Universidad ORT Uruguay

Facultad de Ingeniería

DynamicScan

Interrogador para dispositivo generador de impulsos eléctricos

Entregado como requisito para la obtención del título de Ingeniero de Sistemas

Tomas Dilema – 209316

Milena Dos Santos – 254813

Guillermo Echichure – 151381

Julieta Sarantes – 251105

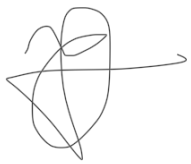
Tutor: Amalia Álvarez

2025

Declaración de autoría

Nosotros, Tomás Dilema, Milena dos Santos, Guillermo Echichure y Julieta Sarantes, declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos el Proyecto Final de Ingeniería en Sistemas;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.



Tomás Dilema

16 de octubre de 2025

Milena dos Santos

16 de octubre de 2025

Guillermo Echichure

16 de octubre de 2025

Julieta Sarantes

16 de octubre de 2025

Agradecimientos

En primer lugar, agradecemos a nuestra tutora Amalia Álvarez por el apoyo, paciencia, orientación y buena disposición durante todo el proceso. También por sacar la mejor versión de cada uno de nosotros. La experiencia de haber trabajado bajo su guía constituyó una instancia de aprendizaje significativa, que contribuyó al desarrollo profesional y personal del equipo.

Extendemos nuestro agradecimiento al cliente Guillermo Kosut y a la empresa *Impulse Dynamics*, por confiar en el equipo y brindarnos la oportunidad de trabajar junto a ellos. Además, agradecer su paciencia y disponibilidad, haciendo el proceso más ameno.

Por otra parte, agradecer a la Dra. Morales y al Dr. Chaia por estar disponibles, y proporcionarnos su punto de vista y conocimiento acerca del entorno médico. De igual manera, a los revisores por sus buenos aportes y consejos.

Por último y no menos importante, a nuestras familias y amigos por su apoyo incondicional, motivación y paciencia, a lo largo de todo este recorrido académico.

A todos, muchas gracias.

Abstract

DynamicScan es un prototipo de *software* desarrollado para asistir en el diagnóstico y seguimiento de pacientes que llevan implantado el dispositivo médico *Optimizer Smart Mini* de la empresa *Impulse Dynamics*, utilizado en terapias de Modulación de la Contractilidad Cardíaca (CCM). Surge de la necesidad del cliente de contar con una herramienta con arquitectura multiplataforma y segura que permita la interrogación del dispositivo médico, la visualización de datos clínicos y la generación de informes automatizados, preservando la trazabilidad y el cumplimiento de estándares regulatorios aplicables al dominio de *software* médico.

La aplicación permite establecer la comunicación con el dispositivo médico implantable, posibilitando la lectura segura de información clínica y técnica de este. Los datos obtenidos son procesados, almacenados localmente y utilizados para generar informes automáticos que facilitan el análisis y la toma de decisiones por parte de médicos, enfermeros y técnicos de *Impulse Dynamics*. El sistema fue diseñado bajo una arquitectura modular y extensible, que separa la interfaz de usuario, la lógica de negocio y la gestión de datos, favoreciendo la mantenibilidad y la evolución del producto.

El proyecto se organizó utilizando un enfoque Incremental e iterativo, gestionando la construcción del producto con prácticas adaptadas de la metodología Scrum, con integración continua, control de versiones mediante Git y Bitbucket, y trazabilidad integral a través de herramientas colaborativas como Jira, Google Drive y Figma. Cada entrega funcional fue validada junto con el cliente, garantizando la conformidad técnica del producto. Además, se aplicaron prácticas de ingeniería de calidad alineadas a las normas IEC 62304, ISO 14971 y Guía de ciberseguridad, asegurando la confiabilidad del *software* y la gestión de riesgos durante su ciclo de vida.

El resultado es un sistema robusto, extensible y mantenible, capaz de operar tanto en entornos conectados como desconectados, brindando una experiencia segura y eficiente para profesionales de la salud. DynamicScan sienta las bases para la evolución futura de la herramienta, la incorporación de nuevas funcionalidades y medios de comunicación alternativos como Bluetooth, sin comprometer la estabilidad ni la trazabilidad del producto.

Palabras clave

.NET MAUI, Android, Aseguramiento de calidad, Dispositivo médico implantable, *Driver*, FDA, FTDI D2XX, Gestión, Guía de Ciberseguridad, IEC 62304, *Impulse Dynamics*, Insuficiencia cardíaca, IPG, ISO 14971, Plan de *software assuranc*e, Procesos, Protocolo, Radiofrecuencia, Riesgo, Riesgo residual, *Software*, Trazabilidad, Validación, Wand, Windows.

Glosario

.NET 8: Plataforma unificada de Microsoft basada en el sucesor de .NET *Core*, abierta y multiplataforma (Windows, Linux y macOS). La versión 8 es de soporte a largo plazo (LTS) e incorpora mejoras en rendimiento, en su *garbage collector* (administrador automático de memoria) y en las bibliotecas base.

.NET MAUI: Marco de trabajo de .NET que permite crear aplicaciones nativas para dispositivos móviles y de escritorio (Android, iOS, macOS, Windows) usando un único código base en C#.

ALCP (Application Level Communication Protocol): Protocolo que especifica el dialogo entre un dispositivo externo y el IPG para ejecutar las ordenes de telemetría requeridas.

API (Application Programming Interface): Interfaz que permite la comunicación entre aplicaciones o componentes de software, definiendo las operaciones disponibles y cómo deben invocarse.

Backend: Capa lógica de un *software* que implementa las reglas de negocio, gestiona los datos y procesa las solicitudes del *frontend*.

Backlog: Lista priorizada de tareas, funcionalidades o requerimientos pendientes del proyecto.

Bluetooth: Es una especificación industrial para redes inalámbricas de área personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia.

Bug: Error o defecto en el software que provoca un comportamiento incorrecto o no esperado del sistema. Los bugs afectan directamente la funcionalidad visible para el usuario y deben ser corregidos para garantizar el correcto funcionamiento del producto.

C# (C Sharp): Lenguaje de programación desarrollado por Microsoft, orientado a objetos y diseñado para la plataforma .NET.

CCM (Cardiac Contractility Modulation): Terapia que mejora la fuerza de contracción del corazón sin alterar el ritmo cardíaco.

CRC (Cyclic Redundancy Check): Código de detección de errores que verifica la integridad de los datos. Se obtiene mediante una división en la que se descarta el cociente y el resto resultante actúa como valor de control para detectar posibles fallos.

CU (Caso de uso): Técnica utilizada en la ingeniería de requerimientos para describir las interacciones entre los actores y el sistema, con el fin de alcanzar un objetivo con resultado observable y de valor. Permite capturar y analizar requerimientos funcionales mediante escenarios, complementando la especificación de requerimientos al aportar contexto, flujo normal y variantes.

D2XX: Librería de FTDI que provee acceso directo a los dispositivos USB de tipo FTDI, evitando el uso de puertos COM virtuales.

Dailies: Reuniones breves y diarias del equipo Scrum donde cada integrante informa qué hizo, qué hará y si enfrenta algún impedimento.

Deuda técnica: Conjunto de decisiones o deficiencias en el diseño, implementación o documentación del software que, si bien no generan fallos inmediatos, pueden afectar la mantenibilidad, escalabilidad o calidad del sistema en el futuro. Su gestión adecuada permite prevenir inconsistencias y asegurar la sostenibilidad del desarrollo a largo plazo.

Driver: Programa que permite la comunicación entre el sistema operativo y un dispositivo de *hardware* específico.

ESRE (Especificación de requerimientos de software): Documento que describe de forma completa los requerimientos funcionales y no funcionales del sistema. Define el comportamiento esperado del *software*, detallando las funcionalidades requeridas, las restricciones del sistema, los atributos de calidad y las interacciones con los usuarios.

Extensibilidad: Capacidad de la solución para incorporar nuevas funcionalidades o variantes sin reestructurar la arquitectura, añadiendo implementaciones detrás de interfaces y manteniendo contratos estables. En este proyecto: nuevas órdenes/vistas, nuevos protocolos por modelo y nuevos medios de transporte se integran agregando estrategias y registrándolas, sin cambiar interfaces públicas.

FDA (Food and Drug Administration): Agencia estadounidense responsable de regular los dispositivos médicos y su *software* asociado.

FP17 (Framing Protocol 17): Protocolo que define la estructura, delimitación y validación de los mensajes intercambiados entre *software* y la antena Wand que implementan la comunicación de telemetría con el dispositivo médico implantable.

FTDI (Future Technology Devices International): Empresa desarrolladora de chips y controladores para comunicación USB a serie, como los usados por la antena Wand.

Frontend: Capa de presentación de un *software* encargada de la interacción con el usuario.

GoF (Gang of Four): Los patrones GoF son soluciones probadas y reutilizables para problemas comunes en el diseño de software.

Interop (Interoperabilidad): capacidad del sistema para comunicarse e intercambiar datos correctamente con componentes heterogéneos

IPG (Implantable Pulse Generator): Dispositivo implantable generador de pulsos eléctricos.

MVVM (Model-View-ViewModel): Patrón de arquitectura que separa la lógica de presentación y la lógica de negocio de una aplicación.

OCP (Open-Closed Principle): Principio SOLID, establece que las entidades de *software*, como clases, módulos y funciones, deben estar "abiertas para su extensión, pero cerradas para su modificación".

Optimizer Smart Mini: Dispositivo implantable de *Impulse Dynamics* que administra impulsos eléctricos para la terapia de Modulación de la Contractilidad Cardíaca (CCM).

PI (Pruebas de integración): Conjunto de pruebas destinadas a verificar la correcta interacción entre módulos, componentes o subsistemas del *software*. Su objetivo es detectar defectos en las interfaces y en el intercambio de información entre las distintas partes integradas del sistema.

Product Owner: Rol en metodologías ágiles responsable de definir las prioridades, gestionar el *backlog* y asegurar que el producto desarrollado satisfaga las necesidades del cliente.

PS (Pruebas de sistema): Pruebas ejecutadas sobre el sistema para verificar el cumplimiento de los requerimientos funcionales y no funcionales. Evalúan la interacción entre componentes y el comportamiento del sistema en condiciones reales de operación.

PU (Pruebas unitarias): Pruebas que verifican el correcto funcionamiento de las unidades individuales de código, como métodos, clases o componentes, de forma aislada.

RF (Requerimiento funcional): Declaración que describe una función o comportamiento que el sistema debe cumplir para satisfacer una necesidad específica. Permite definir qué hace el sistema.

RF (Radiofrecuencia): Es una banda del espectro electromagnético que se utiliza para transmitir información, describe el medio de comunicación de la información entre la antena Wand y el dispositivo implantable.

RFI SCL (RF Interface Script Language): describe el conjunto de comandos, usados para transmitir órdenes del programador u otra aplicación a la Wand que implementa la comunicación RF con el IPG.

Riesgo del producto: Combinación de la probabilidad de ocurrencia de un daño y la severidad de ese daño para el paciente, el usuario o el entorno, resultante de peligros asociados con el *software* o su uso.

Riesgo del proyecto: Evento o condición incierta que, si ocurre, puede tener un efecto positivo o negativo sobre los objetivos del proyecto, tales como el alcance, el cronograma, los costos o la calidad.

Riesgo residual: Riesgo que permanece después de que se han implementado las medidas de control del riesgo.

Scrum Master: Rol responsable de facilitar la aplicación del marco Scrum, asegurando que el equipo siga las prácticas ágiles establecidas.

SO (Sistema operativo): Conjunto de programas que gestionan los recursos de *hardware* y *software* del dispositivo, permitiendo la ejecución de aplicaciones y la interacción con el usuario.

Sprint Planning: Reunión inicial de cada sprint donde el equipo define el objetivo, selecciona las tareas del *backlog* que se abordarán y establece el plan de trabajo para el ciclo.

Sprint Retrospective: Reunión final de cada sprint destinada a analizar el proceso de trabajo, identificar aciertos, oportunidades de mejora y definir acciones concretas para el siguiente sprint.

SQA (Software Quality Assurance): Conjunto de actividades planificadas para garantizar que los procesos y productos de *software* cumplan con los estándares de calidad establecidos.

SQLite: Motor de base de datos liviano, embebido y sin servidor, utilizado para almacenar datos localmente en el dispositivo.

Story Point: Medida relativa usada en metodologías ágiles para estimar el esfuerzo o complejidad de una tarea.

Trends: Sección del sistema que permite visualizar tendencias o variaciones de los parámetros obtenidos del dispositivo a lo largo del tiempo, facilitando el análisis comparativo de resultados.

UI (Interfaz de usuario): Conjunto de elementos visuales e interactivos que permiten al usuario comunicarse con el sistema de forma intuitiva y eficaz.

Wand (Antena Wand): Dispositivo externo que permite la comunicación inalámbrica entre el *software* y el generador de pulsos implantable (IPG).

Wi-Fi: Nombre comercial propiedad de la Wi-Fi Alliance que identifica una familia de protocolos de comunicación inalámbrica basados en el estándar IEEE 802.11 para redes de área local sin cables.

Índice

Índice de ilustraciones

1. Introducción

El presente documento tiene como objetivo describir el proyecto de grado DynamicScan, desarrollado como requisito para la obtención del título de Ingeniero en Sistemas de la Universidad ORT Uruguay, bajo la tutoría de la Lic. Amalia Álvarez.

El proyecto fue llevado a cabo entre octubre de 2024 y octubre de 2025 por los integrantes presentados en la siguiente sección. DynamicScan surge de una necesidad planteada por la empresa *Impulse Dynamics*, dedicada al desarrollo de dispositivos médicos implantables, de contar con una herramienta que facilitara la interrogación segura del dispositivo implantable denominado *Optimizer Smart Mini* (IPG) utilizado en terapias de Modulación de la Contractilidad Cardíaca (CCM).

1.1. Presentación del equipo

El equipo está conformado por Tomás Dilema, Milena Dos Santos, Guillermo Echichure y Julieta Sarantes, estudiantes de la carrera de Ingeniería en Sistemas.

Si bien el equipo no había trabajado en conjunto previamente, sí habían compartido experiencias en subgrupos dentro de distintas asignaturas a lo largo de la carrera. Julieta actuó como nexo entre los integrantes, ya que había trabajado previamente con Milena y, al conversar sobre la conformación del grupo, surgió la incorporación de Guillermo, quien a su vez había colaborado anteriormente con Tomás.

1.2. Elección del proyecto

La elección del proyecto DynamicScan nació de la oportunidad de colaborar con un cliente externo a la Universidad ORT Uruguay, la empresa *Impulse Dynamics*, reconocida internacionalmente por su desarrollo de terapias médicas innovadoras para pacientes con insuficiencia cardíaca. Desde el inicio, esta colaboración representó un enorme desafío y una gran responsabilidad para el equipo, al tratarse de un entorno de alta exigencia donde la precisión y la seguridad son esenciales.

El proyecto fue elegido no solo por su complejidad técnica, sino también por su profundo valor social y aporte a la salud. Trabajar en una solución que contribuye a mejorar la calidad de vida

de personas que padecen insuficiencia cardíaca despertó un fuerte compromiso en cada integrante del equipo. Esta motivación trascendió los objetivos académicos, impulsando al grupo a involucrarse con la realidad clínica y comprender en profundidad tanto la enfermedad como la tecnología detrás del tratamiento.

Para todos los miembros del equipo, DynamicScan significó la primera experiencia desarrollando un *software* médico, lo que implicó aprender nuevas formas de trabajo, incorporar estándares internacionales y adoptar una mentalidad rigurosa centrada en la calidad, la trazabilidad y la seguridad del paciente.

Más allá del aprendizaje técnico, el proyecto se convirtió en una experiencia transformadora. Demostró que la ingeniería de *software* puede ser un motor de cambio real, capaz de generar impacto en la salud y en la sociedad. En este sentido, DynamicScan refleja la vocación del equipo por aplicar la ingeniería al servicio del bienestar humano, uniendo innovación, compromiso y propósito en un mismo camino.

1.3. Objetivos

En esta sección se detallan los objetivos académicos del proyecto, enfocados en el aprendizaje y la aplicación práctica de los conocimientos adquiridos. Los objetivos relacionados con el producto y el proyecto se abordan en la sección 6.1 del capítulo Gestión de la Calidad.

1.3.1. Objetivos académicos

- Adquirir experiencia práctica en el desarrollo y documentación de *software* médico, comprendiendo las particularidades técnicas y regulatorias que implica este tipo de sistema.
- Aplicar los conocimientos y habilidades adquiridos a lo largo de la formación académica en un entorno profesional y de desarrollo real.

1.4. Estructura del documento

El presente documento se encuentra organizado en distintos capítulos que se detallarán a lo largo del mismo, cada uno orientado a abordar un aspecto específico del desarrollo del proyecto de grado DynamicScan.

A continuación, se mencionan junto con una breve descripción de lo que contiene cada uno:

Introducción: Presenta el proyecto, su contexto, la motivación que dio origen a DynamicScan, la composición del equipo y la estructura general del documento.

Descripción del problema y la solución: Expone el contexto del cliente, los marcos regulatorios involucrados, el problema identificado, la solución global y la propuesta.

Ingeniería de requerimientos: Describe el proceso de relevamiento, análisis y especificación de los requerimientos funcionales y no funcionales, junto con las prioridades, y casos de uso.

Arquitectura: Explica la estructura general del sistema, los componentes que lo conforman, las tecnologías utilizadas y las decisiones de diseño adoptadas.

Construcción: Detalla el desarrollo de las funcionalidades, los módulos implementados, los desafíos técnicos, las entregas realizadas (*releases*) y las pruebas realizadas durante la etapa de construcción.

Gestión de la calidad: Presenta las estrategias, herramientas y prácticas aplicadas para garantizar la calidad, seguridad y trazabilidad del *software* médico desarrollado.

Gestión del proyecto: Expone la metodología de trabajo utilizada, la planificación de las iteraciones, la organización del equipo y las herramientas empleadas para el seguimiento del avance.

Gestión de la configuración: Describe los mecanismos utilizados para el versionado, control de cambios, documentación y manejo de entornos del proyecto.

Conclusiones: Resume los resultados obtenidos, las reflexiones finales y el cumplimiento de los objetivos planteados.

Lecciones aprendidas: Menciona los aprendizajes más relevantes del proceso, las fortalezas del equipo y las oportunidades de mejora identificadas.

Referencias bibliográficas: Incluye todas las fuentes consultadas durante el desarrollo del proyecto.

Anexos: Contiene material complementario que amplía o respalda la información presentada en los capítulos anteriores.

2. Descripción del problema y la solución

En este capítulo del documento se explica el contexto del proyecto, el problema y la solución propuesta por el equipo al cliente.

2.1. Descripción del cliente y contexto

En esta sección, se presenta al cliente *Impulse Dynamics*, empresa responsable del desarrollo de la tecnología médica en la que se basa este proyecto. Luego, se describe el contexto clínico y técnico en el que opera la organización, la relevancia terapéutica de su dispositivo médico implantable y los lineamientos regulatorios que enmarcan su desarrollo. Este panorama permite comprender el entorno en el que surge la necesidad abordada por la solución propuesta

Durante todo el proceso de desarrollo del proyecto que abarcó las actividades de ingeniería de requerimientos para gestionar el alcance, diseño, construcción y validación se mantuvo una comunicación constante con el Ing. Guillermo Kosut, representante de *Impulse Dynamics* y Associate Manager de la compañía. Su participación fue clave para orientar las decisiones técnicas, y asegurar la correcta comprensión del funcionamiento del dispositivo y su entorno operativo.

2.1.1. Empresa *Impulse Dynamics*

Impulse Dynamics [1] es una compañía internacional que se dedica al desarrollo de terapias innovadoras para el tratamiento de Insuficiencia Cardíaca, una de las principales causas de hospitalización y mortalidad [2]. La sede principal se encuentra en Estados Unidos con operaciones en distintas regiones de Europa, Asía y América Latina.

La empresa se caracteriza por el diseño de dispositivos médicos implantables que buscan mejorar la calidad de vida de los pacientes que padecen insuficiencia cardíaca crónica, en especial aquellos que no responden adecuadamente al tratamiento farmacológico estándar.

El producto más innovador es el *Optimizer Smart Mini*, un dispositivo implantable que administra la terapia denominada *Cardiac Contractility Modulation* (CCM), reconocida a nivel mundial como una innovación para pacientes con fracción de eyección reducida que no son candidatos a terapia de resincronización cardíaca [3].

Esta técnica está respaldada por múltiples estudios clínicos, aprobada por la *Food and Drug Administration* (FDA) en Estados Unidos y certificada con el marcado CE en Europa, lo que consolida a *Impulse Dynamics* como un referente internacional en la electroterapia cardíaca.

2.1.2. Relevancia clínica y casos de uso

La insuficiencia cardíaca es una enfermedad progresiva que limita severamente la capacidad de los pacientes para realizar actividades diarias. Muchos de ellos experimentan síntomas como fatiga extrema, falta de aire y reducción significativa en su calidad de vida.

El dispositivo *Optimizer Smart Mini* junto con la terapia CCM ofrecen una alternativa terapéutica para este grupo de pacientes, al favorecer la recuperación de funciones previamente limitadas y, en consecuencia, mejorar la calidad de vida [4]. En el Anexo 12.1.1 se puede visualizar más información acerca de esta terapia. Entre los beneficios observados se incluyen:

- Caminar sin fatigarse rápidamente.
- Subir escaleras sin necesidad de detenerse.
- Volver a participar en actividades sociales y familiares.

En estudios clínicos, se observa una mejora de la clase funcional NYHA (New York Heart Association) [5], escala internacional utilizada para clasificar la severidad de la insuficiencia cardíaca en función de la limitación que genera en las actividades cotidianas. La mejora en dicha clasificación se traduce en la posibilidad de realizar esfuerzos físicos con menor fatiga y disnea. Asimismo, se evidencia una mejoría en la prueba de caminata de 6 minutos, que consiste en recorrer la mayor distancia posible durante dicho intervalo, reflejando un incremento en la tolerancia al ejercicio y una reducción de los síntomas asociados a la insuficiencia cardíaca.

En definitiva, el impacto trasciende lo técnico, ya que se orienta a la mejora de la calidad de vida de las personas en un aspecto esencial de su existencia.

2.1.3. Dispositivo implantable

El modelo de dispositivo implantable utilizado en este proyecto corresponde al ***Optimizer Smart Mini***, una versión compacta y avanzada que constituye un generador de pulsos implantable. Este administra impulsos eléctricos en el período refractario absoluto del ciclo cardíaco,

es decir, durante el latido. Mientras que un marcapasos, genera contracciones cardíacas y produce un nuevo latido cuando el corazón no lo logra por sí solo, los impulsos del *Optimizer Smart Mini* no provocan un nuevo latido, sino que modulan la contractilidad del miocardio. De este modo, cada contracción se vuelve más eficiente sin incrementar el consumo de oxígeno, optimizando así la eficacia del bombeo cardíaco.

A diferencia de un marcapasos o de un desfibrilador, el sistema está diseñado para modular la fuerza de contracción del músculo cardíaco en lugar de modificar el ritmo, ofreciendo así una alternativa terapéutica única dentro de las opciones de estimulación cardíaca.

Figura 1: Dispositivo Smart Mini

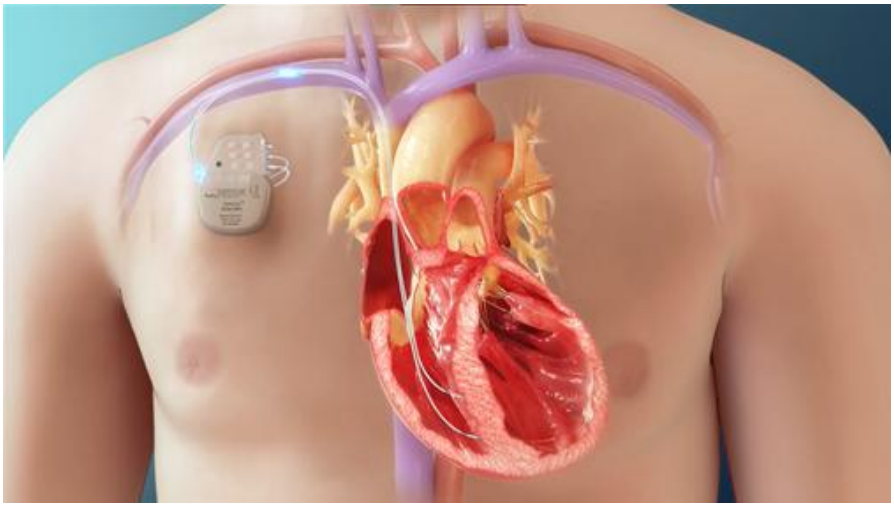


Figura 2: Optimizer Smart Mini implantado

El procedimiento de implante del dispositivo se lleva a cabo bajo anestesia local. Mediante una incisión en la piel se crea una cavidad subcutánea destinada a alojar el dispositivo, habitualmente debajo de la clavícula derecha. Los cables que liberan la energía se introducen por vía

venosa hasta el músculo cardíaco, dos de ellos se posicionan en el ventrículo derecho y, en casos específicos, un tercero se ubica en la aurícula [6].

La terapia CCM no se administra de forma continua, suele programarse para un total de entre 5 y 7 horas diarias, repartidas en intervalos de aproximadamente una hora a lo largo del día, sincronizada con la actividad eléctrica ventricular.

El dispositivo requiere recarga semanal mediante transferencia inductiva transcutánea. Para ello, se coloca el cargador sobre la piel, en la zona del implante, durante aproximadamente una hora, con el fin de mantener la entrega continua de la terapia.

Es importante mencionar que, ante la detección de alguna condición que comprometa la fiabilidad del estímulo como una disminución del voltaje de la batería por debajo del umbral de seguridad, la presencia de ruido en las señales, taquiarritmias, contracciones ventriculares prematuras u otros eventos fuera de los parámetros normales, el dispositivo suspende automáticamente la entrega de la terapia CCM hasta que la condición se resuelva o se restablezca la carga adecuada.

Esta suspensión puede ser transitoria o permanente, dependiendo del tipo de evento detectado. En los casos en que la interrupción sea permanente, se requiere la intervención de un técnico especializado para reconfigurar el dispositivo y reanudar la administración de la terapia.

Cabe destacar que esta situación no representa un riesgo vital inmediato para el paciente, ya que el dispositivo no cumple funciones de soporte cardíaco ni genera latidos. Sin embargo, la interrupción prolongada de la terapia implica una pérdida progresiva de los beneficios clínicos obtenidos, como la mejoría en la tolerancia al ejercicio, la reducción de la disnea y la disminución de la fatiga asociada a la insuficiencia cardíaca.

Actualmente se informan más de 10.000 pacientes implantados con la tecnología Optimizer/CCM en alrededor de 44 países, lo que evidencia la expansión global del dispositivo y el grado de adopción clínica alcanzado [7].

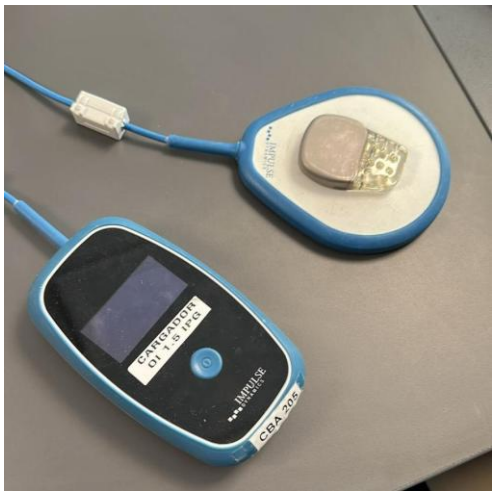


Figura 3: Cargador del IPG

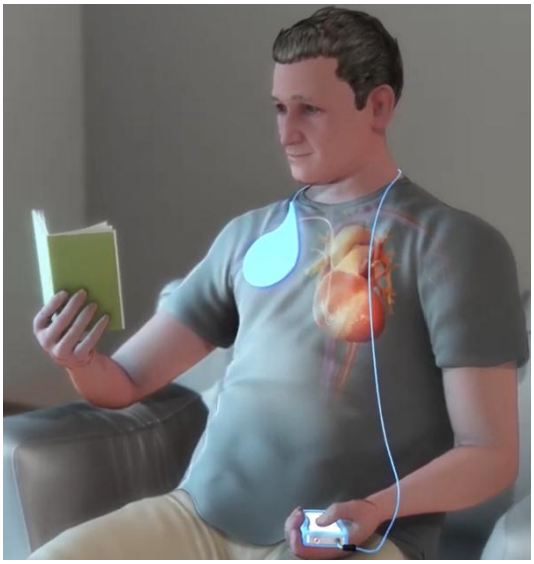


Figura 4: Persona cargando el IPG implantado

2.1.4. Importancia de la comunicación con el dispositivo implantable

La comunicación con el dispositivo implantable resulta esencial para asegurar que la terapia se administra de manera correcta y segura. Es necesario establecer la conexión con el implantable para múltiples finalidades: verificar su funcionamiento; configurar parámetros terapéuticos; obtener información clínica; verificar alertas, como suspensión de la terapia, batería baja, entre otras.

El contexto de uso del dispositivo incluye el acceso al implantable tanto en la fase de implantación como durante el seguimiento. A través de la comunicación se configuran los parámetros de estimulación, se confirma que el sistema administra los impulsos en el período correcto del

ciclo cardíaco y se obtienen registros de funcionamiento. Esto permite disponer de información clave sobre el estado del dispositivo y la respuesta del paciente a la terapia.

La información que se obtiene del dispositivo implantable, también conocido como IPG (Generador de pulsos implantables) se utiliza para evaluar la evolución clínica del paciente, ajustar la programación del dispositivo y garantizar que se cumplen las condiciones de seguridad establecidas. La comunicación permite confirmar que el dispositivo administra la terapia como corresponde, y también, documentar datos relevantes para la toma de decisiones médicas.

La frecuencia de comunicación con el implantable depende de cada caso clínico y del protocolo del centro médico, pero suele realizarse en cada control de seguimiento programado y siempre que se requiera confirmar el estado del dispositivo. La interacción con el mismo la lleva a cabo el equipo médico especializado junto con un técnico de *Impulse Dynamics*, responsables tanto del implante como del seguimiento posterior.

Para establecer la comunicación entre el dispositivo implantable y el *software* clínico se utiliza un accesorio denominado antena Wand, la cual se puede observar en la figura 5. Conectando el sistema y la Wand por medio de un cable USB y esta se comunica con el implantable a través de radiofrecuencia. Se puede observar en la figura 6. La transmisión de datos se realiza a través de un protocolo propietario de comunicación desarrollado por la empresa fabricante, el cual define la estructura y el formato de los mensajes intercambiados. Este protocolo garantiza la seguridad y la integridad de la información médica, constituyendo uno de los principales desafíos técnicos abordados en el desarrollo del proyecto.

La antena Wand transmite información mediante protocolos propietarios, en los cuales cada comando y respuesta se define a nivel de bytes. La comprensión, decodificación y manipulación de esta comunicación constituye uno de los principales desafíos técnicos del proyecto, dado que implica la interpretación de estructuras binarias complejas y la garantía de seguridad de los datos en todo momento.

Figura 5: Antena Wand

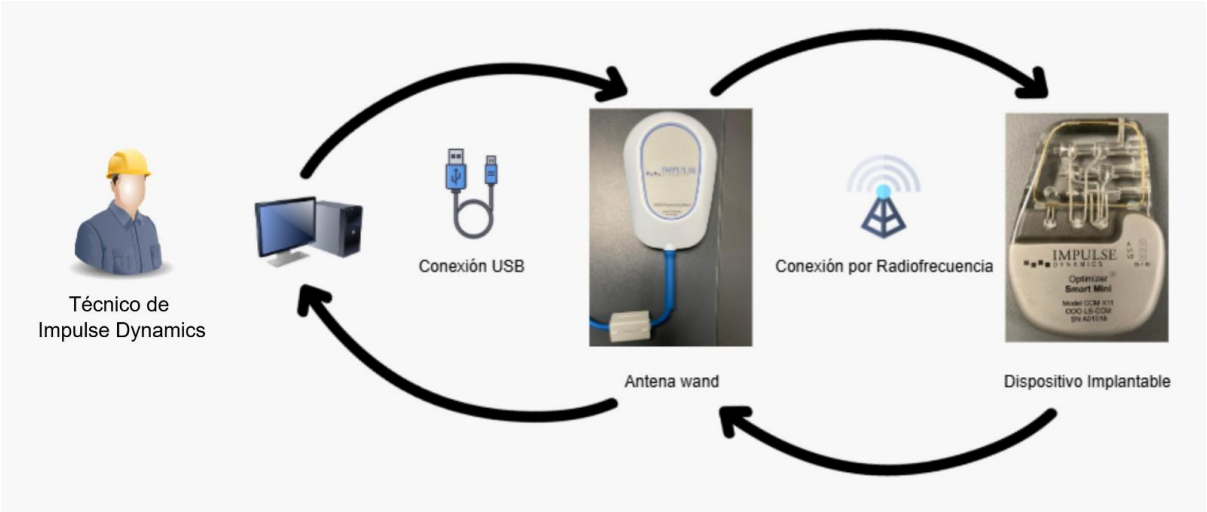


Figura 6: Diagrama de interacción actual

En la figura 6 se muestra el flujo de comunicación: el técnico opera desde el equipo personalizado propio de la empresa, que se conecta por USB a la antena Wand, la cual se comunica por radiofrecuencia con el dispositivo implantable, bajo la estructura que el protocolo de comunicación determina. Las flechas indican un intercambio bidireccional de comandos y respuestas, con el fin de obtener información o configurar el dispositivo implantable.

2.1.5. Marcos regulatorios de *Software* Médico

El desarrollo de *software* en el ámbito médico se encuentra fuertemente regulado, dado que se trata de sistemas que pueden impactar de forma directa en la salud y seguridad de los pacientes. Para que un producto de este tipo se comercialice en Estados Unidos, se exige el cumplimiento de lineamientos establecidos por la *Food and Drug Administration* (FDA) [8]. Entre dichos requisitos se incluye la conformidad con normas internacionales y guías norteamericanas que

definen criterios de seguridad, calidad y gestión de riesgos aplicables al desarrollo de dispositivos médicos.

El marco regulatorio se compone principalmente por tres normas fundamentales:

2.1.5.1. ISO 14971 – Gestión de riesgos en dispositivos médicos

Norma internacional [9], publicada por la *International Organization for Standardization (ISO)*.

Define un proceso sistemático de identificación, evaluación y control de riesgos asociados al uso de dispositivos médicos, incluyendo el *software*. Se documentan los peligros potenciales, se estima su probabilidad y severidad, y se aplican medidas de mitigación que reduzcan los riesgos a niveles aceptables. En el contexto de *software* médico, la norma abarca tanto fallas técnicas (errores de comunicación, vulnerabilidades de seguridad) como riesgos clínicos (uso indebido, pérdida de datos relevantes para el tratamiento).

2.1.5.2. IEC 62304 – Ciclo de vida del *software* médico

Norma internacional [10], publicada por la *International Electrotechnical Commission (IEC)*.

Establece los requisitos de los procesos de desarrollo, mantenimiento y gestión de configuración del *software* que forma parte de un dispositivo médico. Se exige seguir las siguientes actividades independientemente del ciclo de vida elegido, que incluyen Análisis de requerimientos, Diseño de arquitectura, Diseño detallado, Implementación, Pruebas, Liberación y Mantenimiento.

La norma introduce la clasificación del *software* en tres categorías, en función del riesgo asociado a una posible falla:

- Clase A: Falla sin impacto en la seguridad del paciente.
- Clase B: Falla que puede causar daño no grave.
- Clase C: Falla que puede causar la muerte o una lesión grave.

Esta categorización determina el nivel de rigor requerido en el proceso de desarrollo y verificación.

2.1.5.3. Guía de ciberseguridad en dispositivos médicos

Guía norteamericana [11], emitida por la *Food and Drug Administration (FDA)* [8].

Establece las expectativas regulatorias en torno a la protección del *software* y de los dispositivos médicos frente a amenazas cibernéticas. Se incluyen prácticas como:

- Evaluación de vulnerabilidades y amenazas durante todo el ciclo de vida.
- Implementación de controles de seguridad (autenticación, cifrado, validación de integridad).
- Estrategias de actualización segura de *software* y aplicación de parches.

La FDA considera la ciberseguridad como un componente esencial de la seguridad del paciente, ya que un acceso no autorizado o una alteración de datos puede comprometer la eficacia del tratamiento.

El dispositivo implantable utilizado en la solución actual se adecua a este marco regulatorio, lo que garantiza que la terapia pueda aplicarse de manera segura y conforme a los estándares internacionales y regulatorios vigentes.

2.2. Descripción del problema

2.2.1. Situación actual del cliente

Impulse Dynamics dispone de un sistema de configuración e interrogación del dispositivo implantable que funciona únicamente en equipos personalizados propios de la empresa. Dichos equipos están diseñados para uso exclusivo de técnicos especializados, con un flujo de trabajo altamente controlado y dependiente de la logística del fabricante.

Este esquema es efectivo en contextos planificados, donde los implantes y seguimientos se coordinan con antelación y el técnico concurre con el equipamiento preparado. En la práctica, se agenda una cita o se solicita la intervención del técnico de *Impulse Dynamics*, donde el técnico asiste con el equipamiento requerido y configurado según procedimientos internos (imagen controlada, versiones aprobadas, periféricos validados, *checklist* de inicio). El entorno de uso se prepara y verifica: disponibilidad del espacio, comprobación del equipamiento y alineación entre técnico y personal médico antes de iniciar la comunicación con el dispositivo implantable.

Sin embargo, en situaciones no planificadas como emergencias o consultas en domicilio, se evidencian limitaciones importantes. El personal clínico no dispone de un medio inmediato para acceder a la información del dispositivo implantable y depende de la disponibilidad de un técnico y el equipamiento necesario. Ante un evento agudo, se establece comunicación con la línea de emergencia del proveedor, se coordina la visita del técnico y se debe esperar la disponibilidad del equipo dedicado.

El equipamiento que utiliza el técnico es dedicado y provisto por el fabricante, con configuración cerrada y un sistema operativo reducido orientado a la estabilidad, por políticas de seguridad, las actualizaciones son espaciadas y el *software* funciona exclusivamente en ese equipo. Al no existir hoy una vía segura desde dispositivos móviles, cada consulta requiere coordinar equipo y técnico.

A su vez, la comunicación con el implantable pasa hoy por un dispositivo intermediario la antena Wand, provista por un tercero, que se vincula al sistema mediante un controlador específico del proveedor FTDI Chip [12]. Este eslabón adicional agrega puntos de falla, tiempos de coordinación y posibles demoras si el equipo no está operativo, condicionando la continuidad de la atención.

Este esquema resulta adecuado en entornos controlados, pero evidencia una brecha operativa en escenarios no planificados: en escenarios de emergencia, cuando ocurre un evento en domicilio, o cuando no hay personal técnico ni equipamiento dedicado disponible, el equipo clínico no dispone de un medio inmediato para acceder a la información necesaria del dispositivo implantable. La dependencia de equipamiento dedicado y de personal especializado introduce demoras, reprogramaciones y decisiones con información incompleta; coordinar el equipo correcto y su disponibilidad incrementa la latencia entre el evento y la disponibilidad de información pertinente.

A esto se suma que la práctica clínica se desplaza cada vez más hacia contextos móviles y descentralizados: mantener la interacción atada a un equipo único limita la cobertura en domicilio, la respuesta en primera línea y la continuidad de la atención, afectando la experiencia del paciente y la eficiencia del equipo médico, incrementando los costos y el riesgo de indisponibilidad.

2.2.1.1. Limitaciones detectadas e impacto

El modelo actual presenta limitaciones que generan impacto clínico-operativo con demoras y reprogramaciones, impacto económico con elevados costos de soporte y menor flexibilidad:

- **Dependencia de equipamiento específico:** La herramienta funciona exclusivamente en un tipo de equipo con configuración cerrada y sistema operativo reducido en base a Windows 10, resultando incompatible con dispositivos móviles y con otros sistemas operativos fuera del entorno actual, lo que reduce la movilidad y la disponibilidad en campo.

- **Uso restringido a personal especializado:** La presencia de funciones de programación y configuraciones avanzadas dificulta su empleo por enfermería en modo de lectura rápida, y demanda entrenamiento que no siempre está disponible. Además, la interfaz actual está orientada a técnicos y no a perfil clínico (terminología y flujos técnicos), reduciendo la intuitividad y la usabilidad para profesionales ajenos a la tecnología, eleva la carga cognitiva, el tiempo de aprendizaje, y aumenta el riesgo de errores de interpretación en contextos de urgencia.

- **Tiempo de respuesta operativo elevado:** Ante un evento clínico, la coordinación de equipamiento dedicado y la disponibilidad de personal técnico retrasa la interacción con el dispositivo implantable, afectando la oportunidad de atención.

- **Escalabilidad y evolución limitadas:** La comunicación está acoplada al entorno y equipamiento actuales, por lo que incorporar nuevas formas de conectividad o nuevos modelos de dispositivo demanda modificaciones extensas en el sistema, en lugar de extensiones graduales.

- **Capacidad de actualización condicionada:** Aunque el *software* y el protocolo utilizado para comunicarse con el dispositivo implantable puedan modificarse por la organización, dicha modificación implica revalidaciones, posibles actualizaciones en dispositivos implantados y altos costos, lo que limita la velocidad de evolución.

- **Compatibilidad hacia atrás obligatoria:** Los dispositivos implantados (*Smart Mini* y generaciones actuales) deben recibir soporte durante toda su vida útil, estimada en 20

años. Esto exige que cualquier solución futura sea capaz de seguir comunicándose a través de la Wand y el *driver* FTDI existentes.

2.2.1.2. Oportunidades de mejora

Del análisis surgen las siguientes oportunidades de mejora:

- **Acceso oportuno y portable:** Habilitar una vía de interacción rápida desde dispositivos móviles, disponible para enfermería y equipos clínicos de primera línea.
- **Separación clara de responsabilidades:** Organizar la interacción con el dispositivo implantable de forma que la obtención de datos quede independiente de las funciones de configuración, reduciendo riesgos y bajando la barrera de uso.
- **Camino a la conectividad sin cables:** Preparar el terreno para una futura conectividad inalámbrica entre el sistema y el dispositivo implantable, disminuyendo dependencias de puestos fijos y mejorando la cobertura en campo.
- **Experiencia de uso simple:** Focalizar la interfaz y el flujo en el caso de uso clínico de consulta, disminuyendo la carga cognitiva y los tiempos de aprendizaje.
- **Compatibilidad hacia atrás y preparación a futuro:** Preservar la compatibilidad con el equipamiento existente (soporte ~20 años) mientras se habilita una adopción gradual de nuevas formas de conectividad.

Estas oportunidades delinean el estado deseado, disponer de una capacidad de interacción con el dispositivo implantable segura, portable y verificable, accesible en contextos clínicos diversos y preparada para evolucionar.

2.3. Descripción de la solución

2.3.1. Solución global

Se propone una aplicación de uso clínico que permita acceder a la información del dispositivo implantable desde un dispositivo móvil y también desde los equipos actualmente utilizados por *Impulse Dynamics*. La aplicación habilita la consulta de datos y, cuando corresponda, la configuración de parámetros. Está pensada para que personal médico y de enfermería, además del

personal técnico, pueda utilizarla en escenarios planificados y no planificados (emergencias y domicilio).

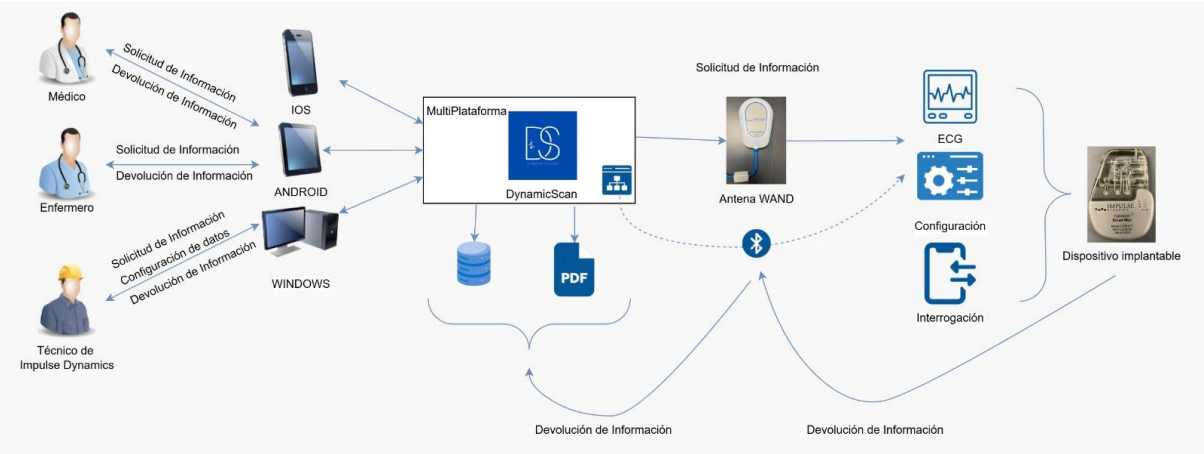


Figura 7: Diagrama de interacción de la solución global

En términos funcionales, la solución global contempla:

- **Interrogación completa del dispositivo implantable:** Obtención de todos los parámetros de terapia y estado almacenados en el dispositivo.
- **Visualización de electrocardiograma (ECG) en tiempo real:** Integración de gráficos dinámicos que reflejen de forma inmediata la actividad eléctrica cardíaca, permitiendo el seguimiento directo de la respuesta del paciente.
- **Gestión de configuraciones terapéuticas:** Permite ajustar parámetros de la terapia CCM bajo acceso restringido, garantizando seguridad y control.
- **Acceso a la información del dispositivo por perfiles diferenciados:** Modo simplificado para enfermería y médicos (lectura rápida de datos clínicos), y modo avanzado para técnicos (configuración y validación).
- **Generación de informes:** Creación automática de reportes en formato PDF que integren datos de interrogación y evolución histórica. Teniendo una versión reducida para médicos y una versión avanzada para técnicos.
- **Persistencia local y portabilidad:** Almacenamiento de datos con encriptación y posibilidad de consulta en contextos sin conectividad.

- **Compatibilidad hacia atrás y a futuro**: soporte obligatorio para el ecosistema actual (Wand, *driver* FTDI, USB), y preparación para nuevas formas de conectividad inalámbrica (Bluetooth, Wi-Fi).

Finalmente, la solución global incluye la preparación de la documentación regulatoria completa exigida por la FDA, en conformidad con IEC 62304, ISO 14971 y guías de ciberseguridad, siendo estos redactados como insumo formal para ser entregado y validado por la FDA.

2.3.2. Solución entregada

La solución consiste en un prototipo funcional que implementa el protocolo de comunicación del implante a partir de la documentación del cliente y se habilita la interrogación y presentación de información clínica para técnicos, médicos y enfermería. Se valida la interoperabilidad con la cadena de comunicación vigente y se preserva la compatibilidad hacia atrás con el parque implantado, a la vez que se deja preparado el camino para incorporar nuevos medios de conectividad sin afectar la lógica clínica. La figura ilustra el flujo de solicitud, interrogación y devolución de información en la aplicación.

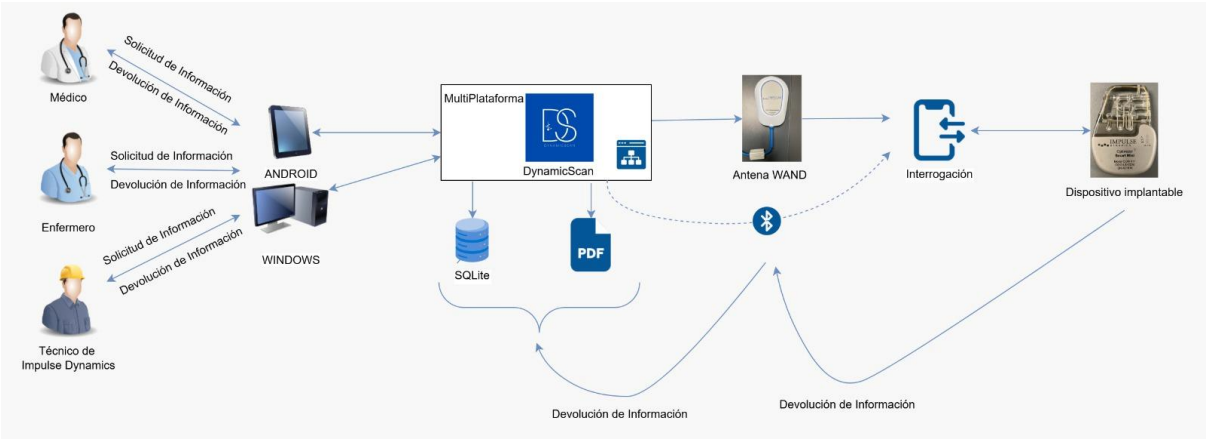


Figura 8: Diagrama de interacción de la solución entregada

- **Comunicación y protocolos**: El protocolo propietario se implementó a partir de la documentación del cliente y se logró la conexión con el dispositivo implantable utilizando la cadena de comunicación existente (Wand/FTDI), implementada en esta etapa en Windows por disponibilidad de *driver*s. La arquitectura modular separa Transporte, Protocolo y Lógica de negocio, facilitando sustituir el medio físico (USB hoy, Bluetooth mañana) sin reescrituras masivas.

- **Aseguramiento de compatibilidad hacia atrás**: La solución entregada garantiza soporte para el ecosistema de comunicación actual, compuesto por la antena Wand y el *driver* FTDI, que son indispensables para interactuar con los dispositivos implantados en pacientes. Esta compatibilidad se mantiene como requisito esencial, dado que los dispositivos existentes deben contar con soporte clínico durante un período aproximado de veinte años, que corresponde a la vida útil estimada de los implantes. La arquitectura desarrollada asegura que, incluso al incorporar en el futuro nuevos medios de comunicación como Bluetooth o Wi-Fi, el sistema continúe siendo capaz de operar con la infraestructura actual sin pérdida de funcionalidad. De este modo, se protege la continuidad de la atención, se reduce el riesgo clínico y se preserva la inversión realizada en los dispositivos ya implantados.

- **Ámbito funcional**: Tras entrevistas y refinamiento del alcance, la versión entregada se reduce a un conjunto acotado de operaciones solo de lectura de datos del dispositivo implantable, suficientes para el caso clínico de consulta. Las funciones de configuración y el módulo de electrocardiograma en tiempo real se posponen por su complejidad, costo de implementación, validación regulatoria, y por el riesgo implicado.

- **Plataformas**: Se prioriza el funcionamiento completo en Windows que es la plataforma actual del cliente y el despliegue inicial en Android. En esta etapa no se contempla iOS debido a la ausencia de *driver*s FTDI, como se puede visualizar en el Anexo 12.1.2 el equipo consultó mediante un correo directamente con los responsables de estos.

- **Interfaz y reportes**: El Prototipo incluye visualización de detalles del IPG, alertas, tendencias y generación de informe PDF con datos obtenidos en las interrogaciones, accesibles en la misma sesión o en consultas históricas. Acotando a un solo informe en formato PDF para todos los perfiles.

- **Documentación**: Se elaboraron artefactos consistentes con IEC 62304, ISO 14971 y Guías de ciberseguridad (requerimientos, diseño y arquitectura, gestión de riesgos, gestión de la trazabilidad, registros de prueba) alineados a dichas normas de la FDA. No constituyen una certificación formal, sino que actúan como insumo para el cliente, con miras a su posterior adaptación a los formatos y procesos regulatorios internos.

3. Ingeniería de requerimientos

En este capítulo se presenta y fundamenta el proceso de ingeniería de requerimientos definido para el proyecto DynamicScan. La etapa se inicia con el propósito de obtener una visión clara, validada y completa de las necesidades del cliente, asegurando el cumplimiento de un subconjunto de normativas regulatorias del *software* médico.

Se considera imprescindible dedicar una fase inicial a la ingeniería de requerimientos. Esta decisión es necesaria debido a varios factores como la complejidad del dominio clínico y tecnológico, la criticidad del sistema en términos de seguridad del paciente, y la necesidad de asegurar la trazabilidad y la alineación con un subconjunto de normas y guías internacionales y norteamericanos como IEC 62304, ISO 14971 y la guía de ciberseguridad de la FDA.

El resultado de este proceso inicial se materializa en una Especificación de requerimientos de *software* (ESRE), que incluye requerimientos funcionales y no funcionales, casos de uso [24] y restricciones generales del sistema. Dicho documento se utiliza como guía para el desarrollo y asegura que las funcionalidades implementadas se alinean tanto con las expectativas del cliente como con las regulaciones normativas.

La etapa de ingeniería de requerimientos no se concibe como una actividad cerrada en una única etapa. A lo largo del proceso de Ingeniería de requerimientos, los requerimientos fueron refinados y ajustados mediante revisiones y validaciones planificadas con el cliente. Esta dinámica permite responder correctamente a posibles cambios que puedan surgir, mejorando la precisión de las especificaciones y garantiza que el sistema se mantenga alineado a las necesidades detectadas.

3.1. Proceso de Ingeniería de requerimientos

El proceso de ingeniería de requerimientos utilizado en el proyecto se estructura en tres actividades: Elicitación, Especificación y Validación. Estas actividades se realizan de manera iterativa, lo que implica que la información obtenida en una actividad se utiliza para las demás. De esta forma, se asegura la coherencia y la completitud de los requerimientos a lo largo del desarrollo del sistema.

El proceso comienza con la elicitación, etapa donde se identifican las necesidades de los interesados mediante entrevistas, talleres con técnico representante de *Impulse Dynamics* y análisis

de documentación técnica y normativa. Esta fase permite comprender en profundidad el dominio del problema, el contexto clínico y las limitaciones tecnológicas propias del entorno de un dispositivo médico implantable.

Posteriormente, los resultados de la elicitación se formalizan en la especificación de requerimientos. Donde se redactan y organizan los requerimientos funcionales y no funcionales, estableciendo identificadores únicos, criterios de aceptación y prioridades. La especificación se plasma en el ESRE (Especificación de Requerimientos de *Software*), que constituye la base de referencia para las siguientes fases de diseño, implementación y prueba.

Finalmente, se realiza la validación, en la cual los requerimientos especificados se validan con el cliente para confirmar que reflejan correctamente sus necesidades. Este proceso se realiza mediante entrevistas, prototipos de distintos niveles de fidelidad y sesiones de retroalimentación con los representantes de la empresa.

Es importante destacar que estas actividades no se realizan de manera estrictamente lineal. En la práctica, se adopta un enfoque iterativo en el que la validación retroalimenta a la elicitación y a la especificación, permitiendo ajustes tempranos y reduciendo el riesgo de inconsistencias. Esta dinámica asegura que los requerimientos evolucionen junto con el entendimiento del problema y con los aprendizajes obtenidos en cada iteración del proyecto.

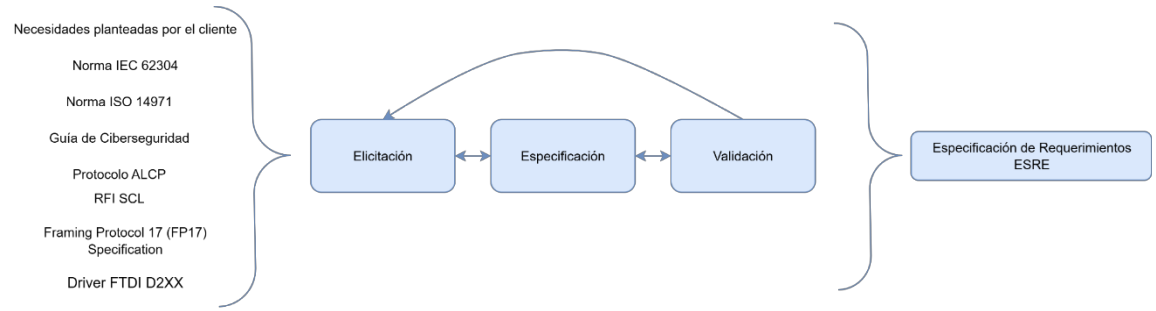


Figura 9: Diagrama de actividades de Ing. de requerimientos

3.2. Actividades del proceso

A continuación, se detallan las actividades del proceso de la fase de Ingeniería de requerimientos.

3.2.1. Elicitación

La actividad de elicitación de requerimientos consiste en recopilar, investigar y definir las necesidades del cliente. Para esto se aplican diversas técnicas en paralelo que permiten determinar tanto las necesidades técnicas del cliente como las normativas y restricciones tecnológicas.

Herramientas de elicitación utilizadas:

- **Entrevistas:** Se llevan a cabo distintas reuniones con representantes de *Impulse Dynamics* con el propósito de relevar las necesidades del cliente, las restricciones de uso y los requisitos técnicos del *software* médico. En algunas de estas instancias, el cliente presenta la interfaz de usuario del *software* actualmente utilizado, con el objetivo de brindar un panorama general sobre el flujo de trabajo y la forma en que los usuarios interactúan con el sistema existente. Este material se utiliza únicamente como referencia conceptual, para comprender las expectativas de usabilidad y el contexto operativo del personal clínico y técnico. A partir de esa observación, se definen los lineamientos generales de diseño de la nueva interfaz, asegurando que mantenga una identidad visual propia, adaptada a los requerimientos del proyecto y a las buenas prácticas de diseño de *software* médico.

- **Talleres:** Se organizan instancias de reuniones entre el equipo de desarrollo y el técnico de *Impulse Dynamics*, en las que se analizan las regulaciones normativas, flujos de trabajo, escenarios clínicos y necesidades de los profesionales de la salud.

- **Revisión de documentación:** Se analizan documentos regulatorios y normativos relevantes, incluyendo la norma IEC 62304 (ciclo de vida del *software* médico), la norma ISO 14971 (gestión de riesgos) y Guía de ciberseguridad aplicables a *software* de dispositivos médicos.

- **Analizar protocolos confidenciales:** Se estudia la documentación técnica de protocolos propietarios como ALCP, FP17 y RF SLS, necesarios para la comunicación entre el prototipo y el dispositivo implantable.

- **Analizar *Driver*s:** se analiza el uso del *driver* FTDI D2XX para la conexión USB de la antena Wand al sistema.

- **Prototipos:** se elaboran prototipos de baja fidelidad (método *Crazy Eights* [13]) para explorar diferentes alternativas de interfaz de usuario y también en Figma con mayor fidelidad.

La combinación de estas técnicas asegura que la elicitación no solo cubra las necesidades explícitas, sino que también contemple los aspectos técnicos y regulatorios que condicionan el desarrollo de la solución.

3.2.2. Especificación

La etapa de especificación consiste en transformar los resultados obtenidos en la elicitación en un conjunto estructurado y formalizado de requerimientos de *software*.

Durante esta etapa se definen tanto los requerimientos funcionales (RF) como los requerimientos no funcionales (RNF). Cada requerimiento se identifica con un código único, se clasifica por prioridad, se describe de manera clara y verificable, y se asocia a criterios de aceptación que permitan validar su cumplimiento.

Entre los aspectos más relevantes de la especificación se destacan:

- La definición de 17 requerimientos funcionales, de los cuales 14 son implementados en la versión entregada del prototipo, priorizados junto al cliente en función de su impacto y criticidad,

- La descripción detallada de cada requerimiento funcional mediante un flujo de pasos que incluye la construcción de comandos en bytes, el envío a través de la antena Wand, la recepción iterativa de respuestas y el mapeo de datos para su interpretación.

- La incorporación de 8 requerimientos no funcionales, que abarcan aspectos de portabilidad, seguridad, variabilidad, integrabilidad, disponibilidad y testeabilidad.

- La elaboración de casos de uso que describen los escenarios de interacción entre actores (médico, técnico, enfermero) y el sistema, asegurando la cobertura de los principales flujos de trabajo.

La especificación se documenta en el ESRE, que constituye el documento de referencia oficial del proyecto. Véase en el Anexo 12.2.1, ESRE *Release* 2.

3.2.2.1. Actores del sistema

En el sistema DynamicScan se identifican los actores principales que interactúan directamente con el *software*. Cada uno de ellos cumple un rol específico en función de sus responsabilidades dentro del entorno clínico o técnico.

Técnico de *Impulse Dynamics*

El actor técnico corresponde al personal especializado de *Impulse Dynamics* o a usuarios con permisos de mantenimiento y diagnóstico avanzado del sistema.

Este perfil cuenta con todas las funcionalidades habilitadas para el médico/enfermero, y además dispone de acceso al menú avanzado, el cual requiere autenticación para garantizar la seguridad y trazabilidad de las operaciones realizadas.

En el modo avanzado, el técnico puede:

- Acceder al "*Summary*" del dispositivo, que resume el estado operativo del IPG, los parámetros de Impedancia del cable uno y dos, RWave de ambos cables y PWave del cable a la aurícula. Ver en el Anexo 12.2.2 un ejemplo de esta pantalla.
- Visualizar los "*Trends*" o tendencias, donde se presentan series históricas de variables relevantes como horas de actividad, voltaje de batería y *leads*, permitiendo evaluar el rendimiento del dispositivo a lo largo del tiempo.
- Verificar las alertas técnicas o de mantenimiento, que informan sobre condiciones del sistema que podrían requerir revisión o reemplazo de componentes.

El acceso a este nivel avanzado se encuentra restringido a personal técnico calificado, conforme a los lineamientos regulatorios y de seguridad definidos para el *software* médico.

Médico / Enfermero

El actor Médico/Enfermero representa al personal clínico autorizado para realizar la lectura de datos del dispositivo implantable en un entorno hospitalario o ambulatorio.

Desde la aplicación, el usuario con este perfil puede establecer una conexión con un dispositivo implantable *Optimizer Smart Mini* (IPG) mediante la antena Wand, ejecutar una interrogación

del dispositivo y obtener los datos de terapia relevantes del paciente en modo reducido en el informe PDF.

Asimismo, tiene la posibilidad de consultar el historial de sesiones anteriores, accediendo a la fecha y hora de cada conexión, el identificador del dispositivo interrogado y el informe PDF generado por el sistema. De esta forma, se dispone de un registro completo de las interacciones realizadas, garantizando la trazabilidad de la información obtenida.

El médico/enfermero también puede visualizar las alertas generadas por el IPG, permitiendo detectar posibles eventos anómalos o situaciones que requieran seguimiento clínico.

Las funcionalidades disponibles para este actor están diseñadas para ofrecer una experiencia segura y simplificada, priorizando la facilidad de uso y la claridad en la presentación de los resultados.

En las siguientes figuras se representa el diagrama de actores y casos de uso asociados al sistema DynamicScan:

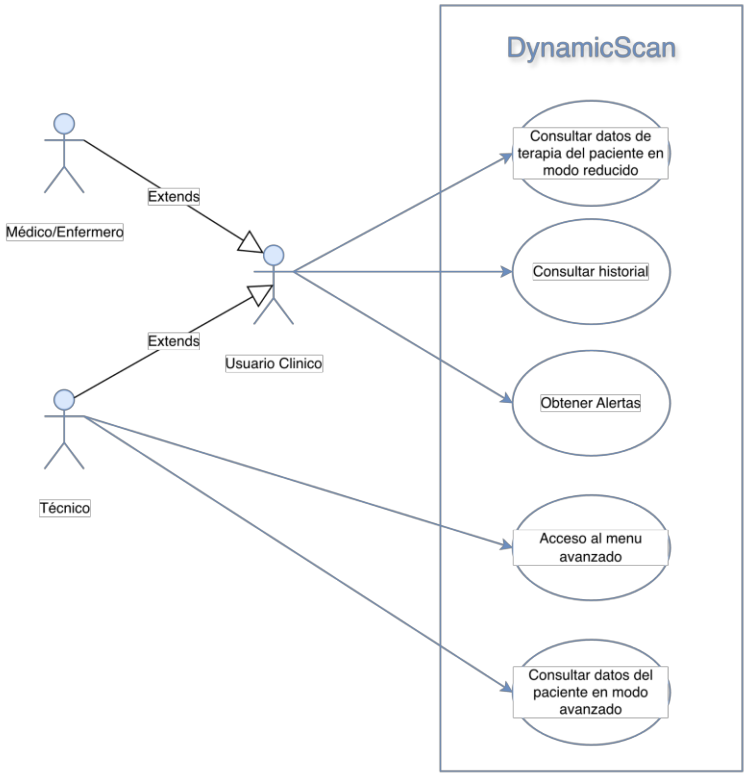


Figura 10: Diagrama técnico con casos de uso

3.2.2.2. Interesados

En el marco de este proyecto se identifican distintos grupos de interesados (*stakeholders*), clasificados según su vínculo con el proyecto y con la solución final.

3.2.2.2.1. Interesados en el proyecto y en la solución

- **Representantes de Impulse Dynamics**

Características: Actores clave de la empresa *Impulse Dynamics*, con interés en validar la factibilidad técnica de la solución.

Interés: Garantizar que el *software* desarrollado cumpla con los lineamientos de calidad y normativas del ámbito médico, aportando conocimiento experto sobre el dispositivo y sus protocolos.

- **Desarrolladores de *Impulse Dynamics***

Características: Equipo interno de ingeniería que tomará la posta del prototipo para integrarlo al ecosistema de la empresa, mantenerlo y evolucionarlo.

Interés: Recibir una base de código legible y modular, con documentación y diagramas actualizados, decisiones de arquitectura trazables, pruebas automatizadas y guías de despliegue. Valoran extensibilidad (nuevos modelos/protocolos y transportes), portabilidad y claridad de dependencias/licencias para continuar el desarrollo sin reestructuraciones.

- **Técnicos de *Impulse Dynamics***

Características: Especialistas en el uso del dispositivo y en la comunicación a través de la antena Wand.

Interés: Evaluar y probar la herramienta en entornos de simulación y uso clínico controlado, verificando que la comunicación con el dispositivo implantable sea confiable y segura.

Interesados principalmente en la solución

- **Médicos**

Características: Profesionales de la salud responsables del seguimiento de pacientes con insuficiencia cardíaca y médicos de emergencia.

Interés: Disponer de un *software* que facilite la interrogación del IPG, mejorando la usabilidad y reduciendo la complejidad técnica para su aplicación clínica.

- **Pacientes con insuficiencia cardíaca**

Características: Beneficiarios finales de la terapia de *Cardiac Contractility Modulation* (CCM).

Interés: Acceder a un tratamiento más seguro, con controles clínicos más eficientes y mayor confiabilidad en la gestión de los datos del dispositivo.

Interesados principalmente en el proyecto

- **Universidad ORT Uruguay**

Características: Institución académica responsable de la evaluación del proyecto de grado.

Interés: Verificar que el trabajo cumpla con los estándares de calidad académica y que los estudiantes adquieran experiencia práctica en proyectos reales de alta complejidad.

- **Equipo de proyecto (estudiantes)**

Características: Integrantes responsables de la planificación, análisis, diseño e implementación de la solución.

Interés: Obtener un producto técnicamente robusto y académicamente válido, cumpliendo con los objetivos del proyecto de fin de carrera.

3.2.2.3. Priorización de requerimientos

El criterio de priorización de los requerimientos se centra en las necesidades del cliente y en la viabilidad técnica de cada funcionalidad dentro del cronograma del proyecto.

La priorización se realiza de manera colaborativa en reuniones periódicas con el equipo técnico de *Impulse Dynamics*, donde se analizan conjuntamente los objetivos funcionales, las dependencias entre módulos y las limitaciones propias del entorno regulado. Cada requerimiento es evaluado según su impacto clínico, su relevancia operativa y el esfuerzo de implementación requerido.

Como resultado de este proceso, los requerimientos se clasifican en tres niveles de prioridad: Alta, Media y Baja, para profundizar ver el Anexo 12.2.3 la Tabla de prioridades.

3.2.2.4. Requerimientos funcionales

En esta sección se presentan todos los requerimientos funcionales identificados durante el proceso de Ingeniería de requerimientos. Para cada requerimiento se incluye su identificador, nombre, prioridad y estado, reflejando su situación actual dentro del proyecto.

Identificador del requerimiento funcional	Nombre	Prioridad	Estado
RF1-000	Establecer la conexión	Alta	Realizado
RF2-000	Cerrar la conexión	Alta	Realizado
RF4-000	Obtención de datos	Alta	Realizado
RF7-000	Visualización de detalles del IPG	Alta	Realizado
RF8-000	Visualización de Informe	Alta	Realizado
RF14-000	Obtener la hora del IPG	Alta	Realizado

RF17-000	Detección de antena Wand	Alta	Realizado
RF9-000	Generación de logs	Alta	Realizado
RF5-000	Visualización de alertas	Media	Realizado
RF11-000	Acceso a sección avanzada	Media	Realizado
RF12-000	Visualización del resumen avanzado	Media	Realizado
RF13-000	*Stats & Trends*	Media	Realizado
RF16-000	Visualización de historial de informes	Media	Realizado
RF18-000	Electrocardiograma	Media	Despriorizado
RF15-000	Generación de informe avanzado	Media	Despriorizado
RF10-000	Visualización de *Splash Screen*	Baja	Realizado
RF3-000	Permitir el cambio de contraseña	Baja	Despriorizado
RF6-000	Recibir datos propios de la Wand luego de establecer la conexión.	Baja	Despriorizado

Dichos requerimientos describen las acciones, procesos y comportamientos que el sistema debe ejecutar para cumplir con las necesidades expresadas por los usuarios y los lineamientos del cliente.

Cabe destacar que el listado incluye la totalidad de los requerimientos relevados, independientemente de su inclusión o exclusión en el alcance final definido en la sección 6.2.4. Gestión del alcance.

3.2.2.5. Requerimientos no funcionales

En esta sección se detallan los requerimientos no funcionales relevados, los cuales establecen las condiciones de calidad, modificabilidad y portabilidad que debe cumplir el sistema.

Estos requerimientos complementan a los funcionales, definiendo cómo debe comportarse el sistema más allá de las funcionalidades que ofrece.

Se incluyen tanto los requerimientos derivados del análisis normativo (IEC 62304, ISO 14971 y guías de ciberseguridad de la FDA) como los identificados a partir de entrevistas y análisis técnico, independientemente de su priorización o inclusión dentro del alcance final.

Identificador del requerimiento no funcional	Nombre	Descripción
RNF-FDA-1-000	Alineación con la Norma IEC 62304	Se deben seguir los procesos, actividades y/o tareas para alinearse con la norma IEC 62304.
RNF-SEG-1-000	Autenticación de usuarios - Ciberseguridad	El sistema debe requerir ingreso con contraseña antes de acceder a la sección avanzada, restringido a técnicos.
RNF-POR-1-000	Portabilidad con arquitectura multiplataforma	El prototipo deberá estar diseñado sobre una arquitectura que favorezca la portabilidad, permitiendo su ejecución en Windows 10 y contemplando, a futuro, su despliegue en Android con modificaciones mínimas. La funcionalidad se

		deberá asegurar en Windows 10, mientras que en Android se validará el despliegue solamente.
RNF-MOD-1-000	Abierto a la extensión a múltiples tipos de dispositivos médicos implantables a futuro	El prototipo no debe limitarse a una única implementación de dispositivo (como el *SMART MINI*), su arquitectura debe permitir, en el futuro, la incorporación de nuevos tipos de dispositivos médicos implantables.
RNF-TES-1-000	Pruebas del prototipo	El prototipo deberá contar con pruebas a nivel: unitario, de integración y/o de sistema según sea el módulo o funcionalidad que se esté probando y la necesidad que se requiera cubrir.
RNF-INT-1-000	Integración futura con nuevas tecnologías de comunicación	El prototipo debe ser abierto a la extensión con nuevas tecnologías de comunicación, como Wi-Fi o Bluetooth, sin impactar negativamente la arquitectura.
RNF-AVA-1-000	Operación offline y persistencia local	El sistema debe permanecer utilizable sin conexión de red ni dispositivo implantable conectado para funciones locales: debe permitir consultar el historial y obtener informes persistidos localmente, sin depender de servicios externos y ser agnóstico al SO.
RNF-AVA-2-000	Disponibilidad del canal de comunicación local	Para operaciones que requieren comunicación con el dispositivo implantable, se debe verificar la disponibilidad del canal antes de ejecutarlas; si no está disponible, se informará de inmediato y la operación no se iniciará. La UI no se debe bloquear, y cada operación de comunicación contará con un límite temporal y cancelable.

3.2.2.5.1. Casos de uso

En el marco de la ingeniería de requerimientos, los casos de uso se adoptan como técnica principal de representación de los requerimientos funcionales. Esta elección se fundamenta en que los casos de uso permiten describir de manera clara y estructurada los escenarios de interacción entre los distintos actores y el sistema, ofreciendo una comprensión accesible tanto para perfiles técnicos como para los interesados no técnicos, como médicos y representantes del cliente.

En este proyecto, los casos de uso se emplearon como un puente operativo entre los requerimientos funcionales (muchos de ellos técnicamente complejos y de alta densidad informativa) y la interacción usuario-sistema. Su incorporación permitió visualizar los flujos reales de uso y derivar criterios de aceptación y pruebas de sistema a partir de los cursos principales y alternativos, alineando la experiencia del usuario con el diseño técnico y la arquitectura. De esta forma, los casos de uso fortalecieron la trazabilidad bidireccional exigida por la norma IEC 62304, facilitando la comprensión y validación de funcionalidades críticas.

Durante el desarrollo del proyecto se definen 17 casos de uso, de los cuales 14 son implementados en la versión actual del prototipo. Cada uno de ellos se construye siguiendo una estructura uniforme que contempla:

1. Identificador y versión.
2. Título descriptivo del caso.
3. Actor o actores involucrados.
4. Objetivo del caso de uso.
5. Precondiciones necesarias para su ejecución.
6. Curso principal.
7. Cursos alternativos o de excepción.

Esta estandarización permite mantener la coherencia documental y facilita la gestión de la trazabilidad técnica.

Asimismo, los casos de uso sirven de base para el diseño de las pruebas de sistema, asegurando que los escenarios verificados correspondan exactamente a los flujos definidos durante la fase de especificación.

En la versión final del ESRE *Release* 2, los casos de uso se documentan en formato extendido, con sus flujos principales y alternativos detallados, cubriendo desde la conexión del sistema con la antena Wand hasta la visualización de tendencias y reportes clínicos.

Dado el nivel de detalle y extensión, la documentación completa de los casos de uso se incluye en los Anexos del presente documento, donde se presentan en su totalidad con la numeración y descripción correspondiente.

Las imágenes de los distintos Casos de uso fueron documentados en Google Drive, como se pueden visualizar en el Anexo 12.2.4. Los Casos de uso descritos se encuentran en el Anexo 12.2.1.2.

3.2.3. Validación

La etapa de validación tiene como propósito confirmar que los requerimientos especificados reflejan de manera fiel las necesidades de los interesados y que resultan factibles de implementar bajo las restricciones técnicas y regulatorias del proyecto.

Las principales actividades desarrolladas son:

- Entrevistas de validación: se llevan a cabo reuniones virtuales con representantes de *Impulse Dynamics*, en las que se revisa y valida cada requerimiento identificado. En el capítulo 6 de Gestión de la calidad se describen más en detalle las validaciones.

- Prototipos interactivos: se utilizan los prototipos elaborados en Figma y los de baja fidelidad como herramienta de validación temprana con el cliente, permitiendo verificar flujos de navegación, claridad de la interfaz y correspondencia con los objetivos clínicos. Véase Anexo 12.2.5, Evidencias de validación con prototipos.

- Interacción continua: los comentarios recibidos en las entrevistas y en la revisión de prototipos se traducen en ajustes sobre la especificación y, en algunos casos, en la redefinición de requerimientos previamente elicitados.

Este enfoque iterativo asegura que la validación no se limite a una etapa aislada, sino que constituya un proceso continuo que acompaña la evolución del sistema teniendo como prioridad las necesidades del cliente.

3.2.4. Gestión del alcance

La gestión del alcance se establece como un proceso colaborativo entre el equipo de desarrollo y los representantes de *Impulse Dynamics*, orientado a garantizar que el producto final responda a las necesidades prioritarias del cliente, dentro de los plazos y recursos disponibles.

Durante la fase de Ingeniería de requerimientos, se desarrolla un proceso de investigación y análisis progresivo, orientado a comprender en profundidad el entorno técnico y regulatorio del sistema, así como las necesidades del cliente. A partir de este trabajo conjunto con los representantes de *Impulse Dynamics*, se construye y consolida el documento ESRE *Release* 2, que formaliza el conjunto de requerimientos acordados y define el alcance final del proyecto. Este puede ser visualizado en el Anexo 12.2.1.

El producto desarrollado corresponde a un módulo de lectura que permite interrogar el dispositivo implantable *Optimizer Smart Mini* (IPG) mediante la antena Wand, procesar los datos obtenidos y presentarlos en una interfaz comprensible tanto para perfiles clínicos (médico/enfermero) como técnicos (especialistas de *Impulse Dynamics*).

En acuerdo con el cliente, se delimita el alcance a operaciones de lectura, excluyendo las funcionalidades de configuración sobre el dispositivo. Esta decisión se fundamenta en criterios de seguridad, cumplimiento normativo y mitigación de riesgos clínicos, en concordancia con los lineamientos de las normas IEC 62304 y ISO 14971.

Asimismo, la arquitectura del sistema se diseña con enfoque multiplataforma, asegurando la operación completa en Windows y el despliegue validado en Android como prueba de concepto, véase en el Anexo 12.2.6. Este enfoque posibilita la portabilidad futura hacia otros sistemas operativos o dispositivos de comunicación, como Bluetooth, sin alterar la estructura base del prototipo.

Durante el desarrollo, se identifican un total de 17 requerimientos funcionales, clasificados según su prioridad en el producto. Tras las instancias de revisión y planificación conjunta con el cliente, se acuerda la implementación prioritaria de 14 requerimientos que conforman el núcleo funcional del sistema.

Los requerimientos restantes (RF3-000, RF6-000, RF15-000 y RF18-000) se despriorizaron de forma acordada con el cliente, a partir de la estimación del esfuerzo requerido y la duración

total del proyecto, lo que permitió determinar tempranamente que no sería posible incluirlos en esta versión del Prototipo.

Asimismo, el requerimiento RF13-000, originalmente denominado "*Stats & Trends*", se ajusta durante el proceso de Ingeniería de Requerimientos, acordándose con el cliente que en esta versión se incluya únicamente la funcionalidad de visualización de tendencias (*Trends*), dejando el componente de estadísticas (*Stats*) previsto para una fase futura del sistema.

Dicha decisión no implica la exclusión definitiva de las funcionalidades, sino su reprogramación para futuras iteraciones, garantizando la continuidad evolutiva del sistema sin comprometer la estabilidad ni la conformidad regulatoria del producto actual.

En todas las instancias, las decisiones de inclusión, exclusión o replanificación son documentadas y aprobadas junto con el cliente, priorizando la relevancia clínica, el valor agregado y la viabilidad técnica de cada funcionalidad.

Este proceso de gestión transparente y controlado permite mantener la trazabilidad entre los requerimientos, los casos de uso y las pruebas de validación, asegurando que el sistema entregado cumpla con los estándares de calidad esperados y con las metas funcionales establecidas en el ESRE *Release* 2.

4. Arquitectura

En este capítulo se presenta la arquitectura de DynamicScan, delimitada por las restricciones y los requisitos no funcionales priorizados con el cliente. Se orienta el diseño a habilitar la comunicación con el dispositivo implantable de forma clara, mantenible y portable; se aíslan las dependencias de plataforma y se deja preparada la evolución del sistema.

Se organiza así: primero se presentan las restricciones y los requisitos no funcionales que guían el diseño; luego se describen los atributos de calidad priorizados y cómo la arquitectura los satisface; a continuación, se detalla la arquitectura del sistema, vista de despliegue, módulos y capas, componentes y conectores, y secuencias representativas; seguidamente se justifican las decisiones y patrones aplicados; por último, se documentan las tecnologías seleccionadas y su justificación.

4.1. Restricciones

Tecnología del cliente. Se impone como restricción mantener la base de código en C# y, preferentemente, permanecer dentro del ecosistema .NET, a fin de facilitar la futura migración de algoritmos existentes.

***Driver*s.** Se debe conservar el uso de esta versión de *Driver* FTDI utilizada en el sistema actual del cliente

Sistema Operativo. Se exige compatibilidad con Windows 10 (ejecutable en Windows 10 o superior) como entorno mínimo

***Testing* Unitario.** Se prueban unitariamente las funcionalidades y módulos críticos, de acuerdo con las exigencias de la FDA.

4.2. Requerimientos no funcionales

Se consideran los requerimientos no funcionales (RNF) el motor y la base de la arquitectura presentada. Se usan como criterios fundamentales para el diseño y la verificación del sistema, y su implementación se refleja en las vistas y decisiones detalladas en este capítulo.

Los siguientes son los RNF que incidieron en la arquitectura:

- **RNF-POR-1-000 — Portabilidad multiplataforma**
- **RNF-MOD-1-000 — Variabilidad por modelo de dispositivo**
- **RNF-INT-1-000 — Integración de nuevos medios de comunicación**
- **RNF-AVA-1-000 — Operación offline local**
- **RNF-AVA-2-000 — Gestión de indisponibilidad del canal**

Para más detalle de los requerimientos no funcionales ver la sección 3.2.2.5.

4.3. Descripción de la arquitectura

Este apartado explica la solución desde el punto de vista arquitectónico: qué piezas la componen, cómo se organizan, cómo colaboran y por qué se tomaron ciertas decisiones, con el objetivo de satisfacer los RNF y los atributos de calidad. El recorrido es de lo general a lo particular: primero el diagrama de contexto para ubicar la solución en su ecosistema; luego despliegue (dónde corre y cómo se conecta); más adelante la vista de capas y componentes (estructura interna y contratos) y, por último, secuencias representativas que muestran el flujo real de una orden.

4.3.1. Diagrama de contexto

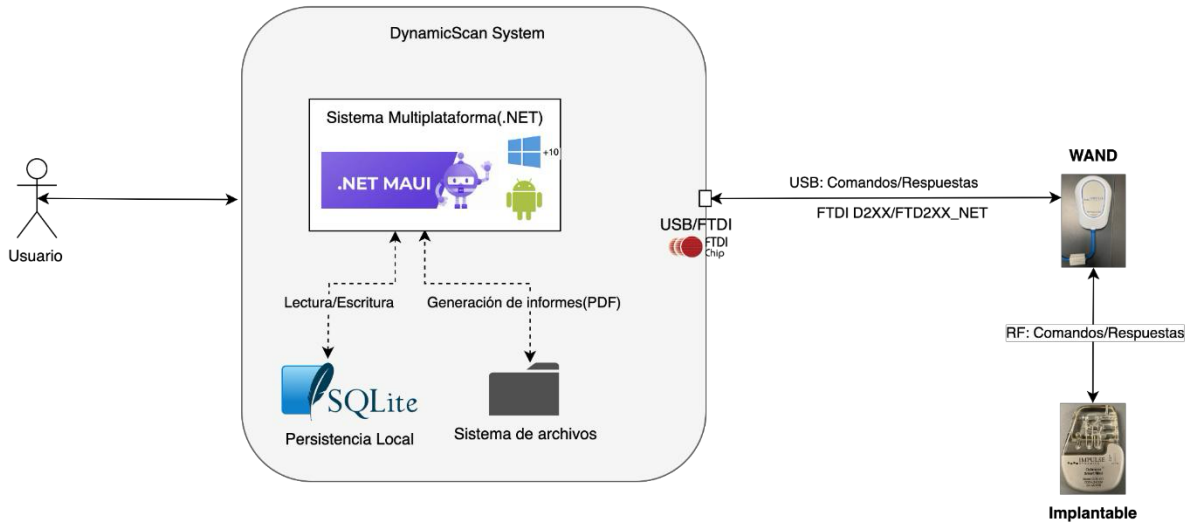


Figura 11: Diagrama de contexto

El diagrama resume el ecosistema de uso:

- Usuario clínico → interactúa con DynamicScan System, una aplicación multiplataforma .NET (MAUI) + NET 8.0 que hoy se ejecuta en Windows 10+ (con pruebas de concepto en Android).

- Dentro del sistema, la aplicación utiliza persistencia local con SQLite y sistema de archivos para la generación de informes en formato PDF.

- La integración con el *hardware* se realiza por USB mediante el *driver* FTDI D2XX / FTD2XX_NET (frontera de integración). Por ese enlace circulan Comandos/Respuestas hacia/desde la WAND.

- La WAND retransmite por RF los mismos Comandos/Respuestas al dispositivo implantable (IPG) y devuelve sus respuestas al sistema.

4.3.2. Diagrama de despliegue

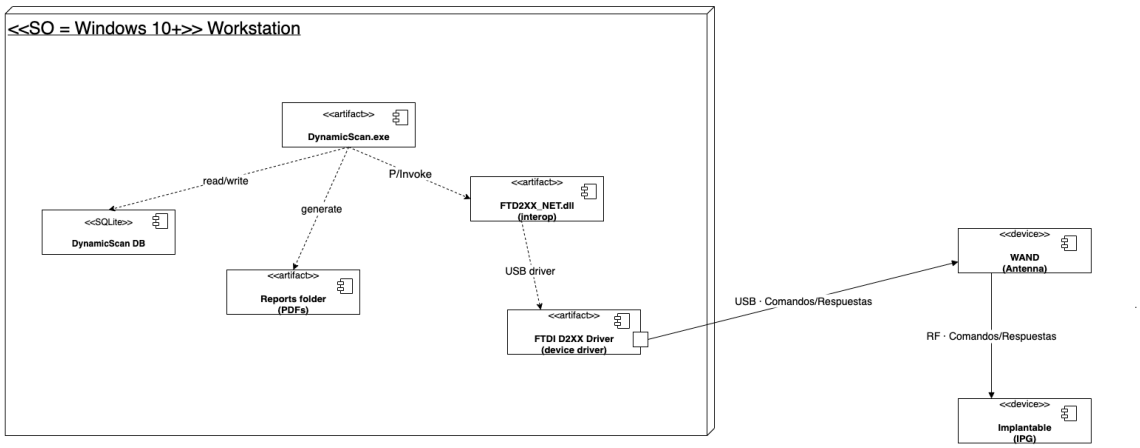


Figura 12: Diagrama de despliegue

*RF: Radiofrecuencia

En la figura 12 se muestra el diagrama de despliegue de la solución que se despliega en una *workstation* Windows 10+, donde se ejecuta el proceso DynamicScan.exe (.NET). Todo el procesamiento se realiza localmente: se persisten datos clínicos en una base SQLite (archivo "DynamicScan DB") y se generan informes en una carpeta de reportes (PDFs) dentro del mismo equipo. No hay servidores ni servicios remotos en esta configuración.

Se establece como frontera de integración con el *hardware* el enlace USB. Desde DynamicScan se invoca, vía *P/Invoke*, la librería FTD2XX_NET.dll (Interop. administrado), y a su vez se utiliza el *driver* D2XX instalado en el sistema operativo para cursar los bytes por USB. Por ese canal viajan Comandos/Respuestas entre la PC y la Wand; por radiofrecuencia (RF) se retransmiten esos mensajes hacia/desde el implantable (IPG). No se interactúa directamente con el dispositivo: todo el acceso físico queda encapsulado en el *driver* USB y la antena.

En funcionamiento, si la Wand no está disponible, se mantiene utilizable la aplicación para tareas locales (consulta del historial, generación de informes ya persistidos). Cuando la Wand está presente, el *driver* expone el dispositivo USB y se establece la sesión; se opera con *timeouts* y cancelación por operación para mantener la UI responsiva. En esta topología, la única dependencia externa relevante en tiempo de ejecución es *driver* USB + Wand; el resto de los artefactos (ejecutable, base SQLite, PDFs) residen en la misma *workstation*.

Finalmente, se refleja en este despliegue el transporte USB/D2XX actual. Se deja abierta la incorporación, en futuras versiones, de otros medios (p. ej., Bluetooth) sin alterar el resto de los artefactos locales: se cambia la pieza de transporte y la cadena USB + Wand, se mantienen intactas la persistencia y el proceso de la aplicación.

4.3.3. Arquitectura en capas

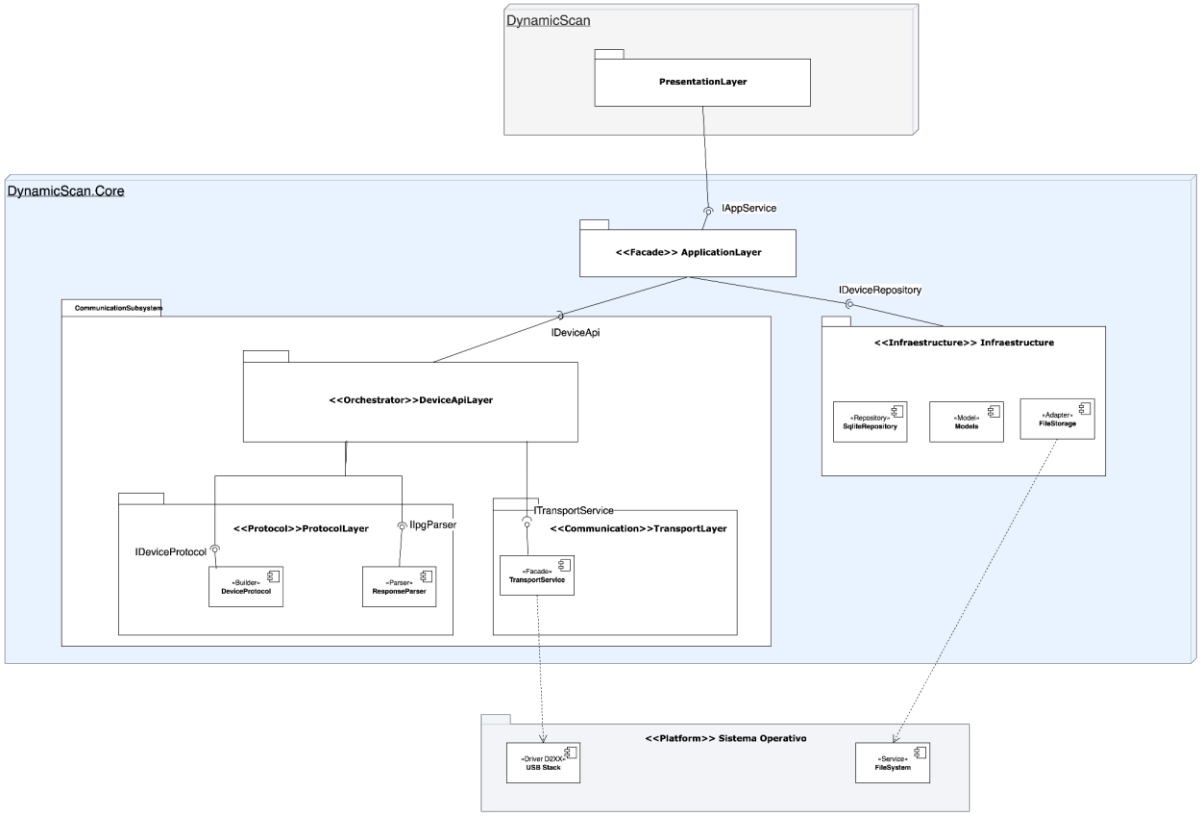


Figura 13: Diagrama arquitectura en capas

Evidencia **RNF-MOD-1-000 y RNF-INT-1-000**

Se organiza la solución en dos módulos: *DynamicScan* (presentación) y *DynamicScan.Core* (núcleo reutilizable). Se consume el núcleo desde la Presentación exclusivamente mediante el puerto de aplicación (*IAppService*).

En el *core*, se orquestan casos de uso desde la capa de Aplicación y se establece como única puerta hacia la UI. El subsistema de comunicación se estructura con *DeviceApi* como mediador sobre dos capas paralelas y desacopladas: Protocolo (*builders/parsers* por modelo; se expone *IDeviceProtocol/IIpgParser*) y Transporte (envío/recepción; se expone *ITransportService*). Se realiza la selección de variantes tras el inicio de sesión mediante fábricas.

Se proveen adaptadores para la persistencia local y el sistema de archivos del sistema operativo desde Infraestructura, encapsulándose esa interacción con la plataforma. Se mantiene como única dependencia directa con el *stack* específico del SO el *driver* USB D2XX usado por Transporte; se mantiene el resto del núcleo aislado detrás de puertos. Con esta organización se deja

preparado el sistema para incorporar nuevos medios o variantes de protocolo cambiando implementaciones detrás de los puertos, sin impactar a la UI ni a la lógica de aplicación.

Componentes principales

Presentación (DynamicScan — MVVM [26])

Se muestran *Views* y *se* coordina la interacción desde *ViewModels*. Se comunica solo con *IAppService* para iniciar sesión, interrogar, consultar histórico y generar informes. No se contienen reglas clínicas ni detalles de comunicación.

Aplicación (*Core* — Fachada/Orquestador)

Se orquestan casos de uso, se validan precondiciones y se traducen resultados/errores a un modelo uniforme para la UI. Se coordina Comunicación e Infraestructura sin exponer detalles de protocolo, transporte ni almacenamiento.

DeviceApi (Fachada del subsistema de comunicación)

Se establece como punto de entrada para órdenes. Se aplica política de precondiciones, *timeout* y cancelación por operación y orquesta la secuencia: construir mensaje → enviar/recibir → parsear respuesta. Se mantienen desacopladas Protocolo y Transporte.

Protocolo

Se construyen tramas y se interpretan respuestas según el modelo del implante. Se elige la variante concreta en tiempo de despliegue/inicialización vía *Factory* tras *StartSession*. Se verifica integridad (p. ej., CRC) y se mantiene agnóstico del transporte.

Transporte

Se abstrae el canal físico de bytes con *ITransportService*. Se gestionan apertura/cierre, E/S no bloqueante, *timeouts* y cancelación. Se mantiene como único punto que toca la plataforma mediante *FTD2XX_NET* / *D2XX* (USB). Se permite sumar otros medios (Bluetooth/Wi-Fi) sin impactar Protocolo ni Aplicación.

Infraestructura (persistencia y utilidades)

Se implementan *SQLiteRepository* y el adaptador de *file system* (PDF, configuración, *logging*). Se cumplen interfaces definidas por Aplicación, manteniéndose la UI y Comunicación independientes de detalles de almacenamiento.

4.3.4. Diagrama de componentes y conectores

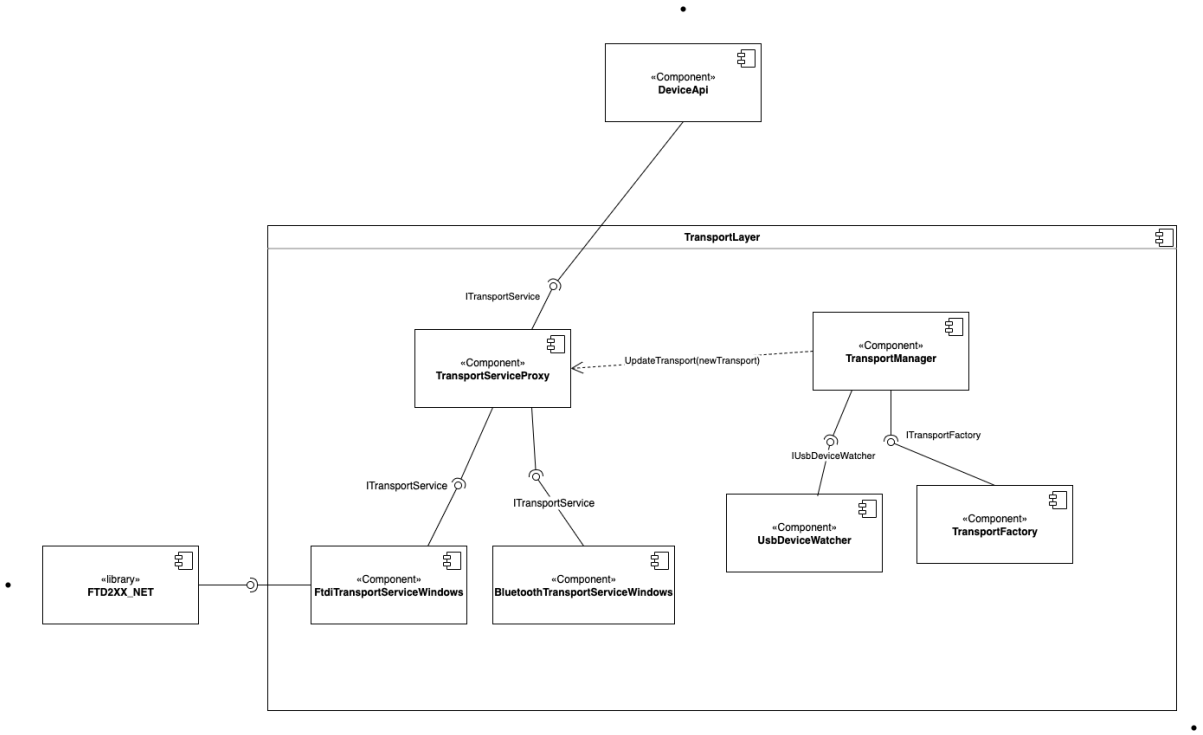


Figura 14: Diagrama de componentes y conectores

Cumple **RNF-INT-1-000**

Elemento	Responsabilidad
ITransportService	Interfaz común para todos los transportes físicos. Define un contrato con operaciones básicas de comunicación (abrir, cerrar, enviar, recibir, configurar, etc.).
TransportServiceProxy	- Centraliza el acceso a transporte físico. - Delegar llamadas al transporte activo - Permitir cambiar dinámicamente la implementación en tiempo de ejecución

TransportManager	Administrar el proxy y actualiza dinámicamente el transporte según eventos y políticas.
TransportFactory	Entregar la implementación concreta de *ITransportService* según plataforma/condiciones *hardware*.
IUsbDeviceWatcher	Define un mecanismo de notificación de conexión/desconexión de dispositivos USB.
UsbDeviceWatcher.**Windows**	Implementación específica de Windows para detectar eventos USB.
Ftdi**Windows*****TransportService***	Implementación USB (FTDI) para Windows basada en FTD2XX_NET.
DeviceApi	Consumir *ITransportService* para ejecutar casos de uso del dispositivo.
FTD2XX_NET (*library*)	Librería de terceros para acceso a dispositivos FTDI desde .NET.

4.3.5. Diagramas de secuencia

Inicio de sesión (*StartSession*)

En el siguiente diagrama de secuencia se describe el flujo completo del proceso de inicio de sesión con el dispositivo IPG, eje central del *Release* 1. Este caso de uso se representa como el primer punto de interacción efectiva entre la aplicación y el dispositivo médico, donde se establece la conexión, se envía un comando binario específico (*StartSession*) y se procesa la respuesta para obtener datos identificatorios del IPG (modelo, número de serie, etc.).

El flujo se inicia desde la capa de presentación (*ViewModel*) y se atraviesa la capa de aplicación (*AppService*) y la capa de comunicación (*DeviceApiLayer*). La respuesta binaria recibida es desempaquetada, parseada y evaluada. En base al modelo recibido y al tipo de conexión activa (*USB* en este prototipo), se utiliza una fábrica de protocolos (*DeviceProtocolFactory*), por medio de la cual se instancian dinámicamente los objetos necesarios para continuar la sesión (protocolo de comunicación y *parser* asociado).

Con este mecanismo se permite desacoplar el código del modelo específico de IPG y del medio de transporte utilizado, habilitando la extensibilidad futura a otros dispositivos y tipos de conexión (Bluetooth, simulación, etc.). El resultado final se almacena en memoria y se utiliza para continuar la comunicación con el dispositivo a lo largo de la sesión activa.

Por el momento, se encuentra preparada la implementación para reconocer y operar con el modelo de IPG; SMART MINI. No obstante, se tiene la capacidad de identificar un dispositivo INTEGRA, en cuyo caso se genera una respuesta adecuada indicando que dicho modelo no es soportado en esta versión.

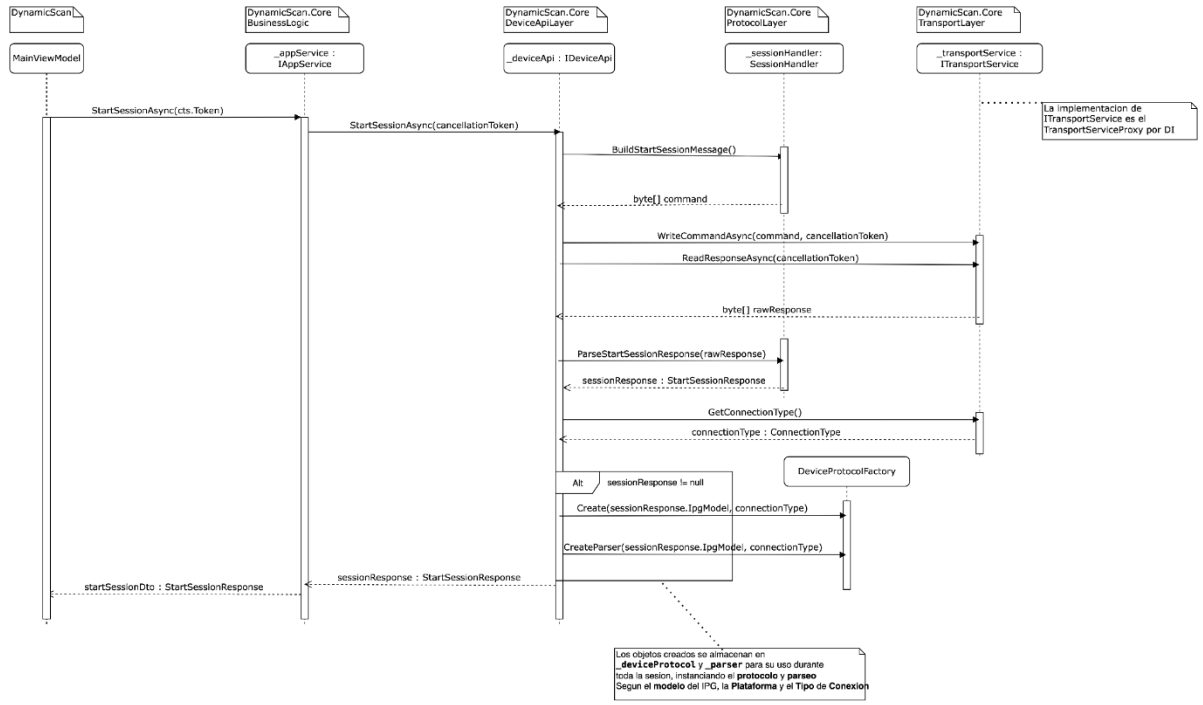


Figura 15: Diagrama de secuencia StartSession

Proceso de Órdenes (*GetIpgTime*)

A continuación, se detalla un diagrama que ilustra el flujo genérico de una orden (ej.: *GetIpgTime*) una vez iniciada la sesión y enlazado el protocolo correspondiente al modelo del dispositivo implantado conectado. Desde la UI/*ViewModel* se invoca el servicio de aplicación y desde allí se invoca el método de la API de dispositivo, que valida que el protocolo esté inicializado y que el canal de comunicación esté abierto. Con esas precondiciones satisfechas, se construye el comando mediante el protocolo activo y se serializa el *frame* correspondiente.

La orden se envía de forma asíncrona por la capa de transporte, con *timeout* y cancelación propios de la operación. Se cursan los bytes a través del *stack* USB del sistema operativo (*driver*

FTDI D2XX) hacia la Wand, retransmitiéndose por radiofrecuencia (RF) al implante. Se recorre el camino inverso para la respuesta hasta el transporte, donde se entrega el *buffer* recibido al protocolo.

Se verifica la integridad en el protocolo (p. ej., CRC), se interpreta la respuesta y se mapea a un resultado de dominio. Se devuelve ese resultado a la capa de aplicación y, cuando corresponde, se persiste o se registra la operación. Si ocurre una condición anómala (transporte no disponible, *timeout*, cancelación, *frame* inválido o error reportado por el dispositivo), se clasifica la causa y se devuelve un error claro sin propagar estados parciales. De este modo, todas las órdenes comparten la misma tubería; lo que varía entre ellas es el *builder/parser* del protocolo y, eventualmente, la política de tiempo y cancelación asociada.

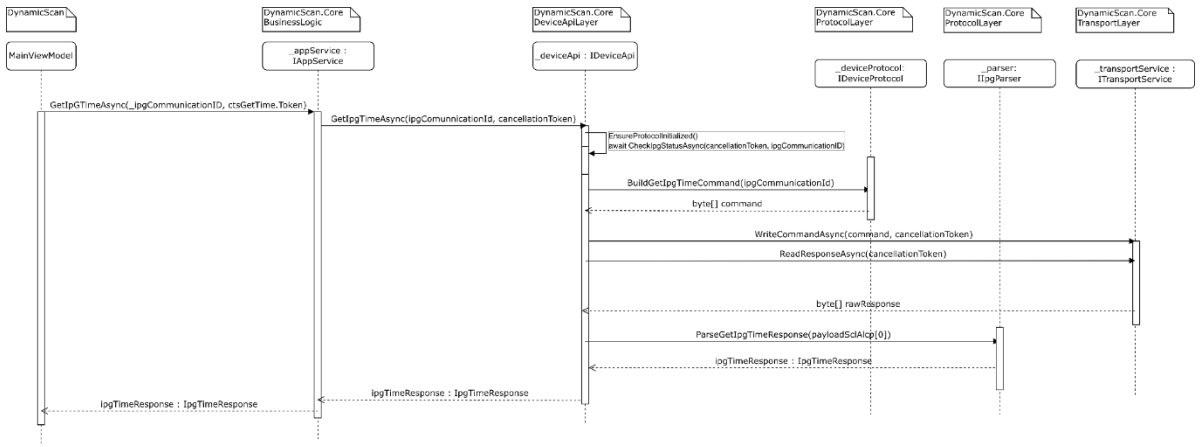


Figura 16: Diagrama secuencia GetIpgTime

4.4. Atributos de calidad

En este apartado se describen los atributos de calidad priorizados y se justifica cómo la arquitectura los satisface mediante tácticas y escenarios. Se apoya el análisis principalmente en el marco de atributos, tácticas y escenarios de *Software Architecture in Practice* [14]. Para cada atributo se presentan escenarios representativos y las decisiones que los habilitan. Se deriva la selección de los RNF acordados con el cliente y se orienta la evolución del sistema a largo plazo, asegurando que las necesidades clínicas y técnicas queden cubiertas de manera consistente y verificable.

4.4.1. Portabilidad

Se establece como requisito clave del cliente desarrollar el prototipo con una tecnología y arquitectura que garantice la portabilidad. Con ello, se permite su ejecución en la versión actual de Windows 10 y se facilita su despliegue en Android, así como una integración sencilla con otros sistemas operativos como iOS.

Dado que se tiene como restricción utilizar tecnología del cliente, como mantenerse en el ecosistema de .NET y desarrollo en lenguaje C#, se realiza una investigación para dar con dicha tecnología y se decide adoptar .NET MAUI [15] como *Framework* multiplataforma del proyecto.

En la primera semana de desarrollo, al crear pruebas unitarias para la lógica, el *test runner* .NET 8 (net8.0) no podía ejecutar ni restaurar correctamente cuando el proyecto de pruebas referenciaba directamente el proyecto MAUI. La causa fue el *multi-targeting* de MAUI (net8.0-android, net8.0-windows10, etc.): al arrastrar dependencias de plataforma (AndroidX, WinUI, etc.), el runner de escritorio net8.0 que no ejecuta sobre Android/iOS ni carga esos *toolchains*, generaba conflictos de restauración/compatibilidad.

Para solventar la incompatibilidad, se extrajo la lógica del dominio y la comunicación a un módulo independiente *DynamicScan.Core* (biblioteca net8.0). Este núcleo expone un puerto de aplicación estable hacia la UI y mantiene sus dependencias técnicas detrás de contratos.

El proyecto MAUI quedó exclusivamente para la presentación y consume ese puerto sin conocer detalles de protocolo, transporte ni persistencia. Las pruebas unitarias referencian ahora el Core net8.0, por lo que compilan y se ejecutan en runners estándar sin *toolchains* móviles; además, el interop FTDI y el acceso a archivos/base se ejercitan mediante adaptadores y dobles de prueba.

Esta separación no solo resolvió el bloqueo del *test runner*; también mejoró la arquitectura: acelera los ciclos de *build*/*test*, reduce la fricción de dependencias, favorece la CI y clarifica responsabilidades (MAUI = UI / Core = negocio y comunicación). El Core queda portable y reutilizable por futuras UIs o servicios, mientras que la UI permanece estable aun cuando se incorporen nuevos medios o variantes de protocolo detrás de las interfaces del núcleo. En síntesis, el cambio convirtió un impedimento operativo en un beneficio de portabilidad y mantenibilidad alineado con SRP/DIP.

4.4.2. Integrabilidad

Se considera otro de los principales requisitos del cliente el deseo de, en un futuro cercano, poder integrar nuevos medios de comunicación al nuevo sistema.

En su sistema actual se cuenta con un único medio de transporte de datos, por medio de la antena Wand con su USB-*driver* FTDI D2XX; se tiene la intención de "comprar futuro" y se apunta a una comunicación inalámbrica entre el sistema y el dispositivo implantable, sacando del medio intermediarios que puedan generar errores, demoras en los procesos y dependencia de *hardware* específico. Se apunta principalmente a una comunicación Bluetooth; por esto se implementa una arquitectura enfocada en favorecer la integración de nuevos medios de comunicación.

Para ello se aplican las siguientes tácticas de integrabilidad:

Usar un intermediario.

Se introduce un puerto de comunicación con contrato mínimo y estable, y se interpone un adaptador por cada tecnología (USB, Bluetooth, Wi-Fi) que traduce particularidades de *driver*, permisos, buffers y códigos de error hacia ese contrato común. Con esto se localiza el cambio en el borde del sistema y se evita que diferencias sintácticas (formatos de *frame*), semánticas (significado de errores/estados) y temporales (*latencias*, *timeouts*) se propaguen a las capas superiores. Como consecuencia práctica, se reduce agregar un medio nuevo a proveer su adaptador y registrarlo, sin reescrituras en lógica de negocio ni en el protocolo.

Orquestar y Enlace diferido

La selección y actualización del medio de comunicación se gestionan desde un único coordinador. Se determina e implementa el medio más apropiado según el contexto. El enlace se difiere al momento de inicialización. Se detecta el entorno (SO, *drivers* presentes, permisos), se descubre y registra/configura el medio correspondiente, por medio de la Inyección de dependencias.

Se centraliza la selección y conmutación del medio de comunicación en un único coordinador; se expone a las capas superiores un punto de acceso único para enviar/recibir datos y se aplican políticas homogéneas de *timeouts*, reintentos y acceso exclusivo al dispositivo. Con esto, se evita que los casos de uso y el protocolo conozcan tecnologías específicas (USB, Bluetooth,

Wi-Fi) o detalles de *drivers* o permisos: toda la variación queda confinada en el borde del sistema.

A partir de ese estado inicial, se orquesta en tiempo de ejecución: se selecciona el medio activo, se conmuta ante fallas o desconexiones según eventos de disponibilidad, y se mantienen invariantes de recursos (colas, buffers, concurrencia) de forma uniforme para todas las tecnologías, cambiando el medio de comunicación cuando así lo determinen la política y las condiciones de operación, de forma transparente para las capas superiores.

En la práctica, incorporar un medio nuevo se limita a proveer su adaptador y ajustar las políticas, conservando intactas la lógica clínica y el protocolo, y cumpliendo el atributo de Integrabilidad.

4.4.3. Variabilidad

Se opera con múltiples dispositivos médicos implantables dentro del alcance de la aplicación; en consecuencia, la arquitectura se concibe como base de una familia de productos (*product line*) en la que cada variante corresponde a un modelo/protocolo distinto. Se procura que los activos compartidos (código, pruebas, documentación) se mantengan estables y que las diferencias se gestionen en puntos de variación predefinidos, de modo que la incorporación de nuevos modelos se realice por selección y adaptación con costo y riesgo acotados. Así, no solo se satisface lo actual, sino que se deja preparada la arquitectura para anticipar e incorporar futuras variantes de dispositivos de forma fluida y controlada.

Se incluyen al menos dos dispositivos en la familia de productos: el ya integrado *Smart Mini* (protocolo ALCP) y el *Integra*. Entre variantes de esta familia se modifican, principalmente, la implementación de los comandos/órdenes y el parseo de respuesta, los campos, los modelos de datos clínicos y ciertos códigos de error/estado. En cambio, otras partes del sistema no se deben cambiar entre variantes: la lógica de negocio clínico expuesta hacia la aplicación, los contratos del "*Core*" y el medio de transporte (que permanece abstracto).

Split module

Se separa la lógica específica de cada variante de protocolo en unidades propias (construcción de comandos y parseo de respuestas por modelo), en lugar de mezclar reglas de múltiples dis-

positivos en un único módulo. Se evita que un cambio en Integra afecte a *Smart Mini*, y viceversa. Como resultado, se mantiene en cada submódulo una responsabilidad única y coherente: "producir y consumir tramas para ese modelo".

Defer Binding

Después de iniciar sesión se conoce el modelo del implante; antes de ese momento no existe información suficiente para decidir qué variante de protocolo utilizar. Por ese motivo se aplica *deferred binding*: la selección y el enlace del protocolo se posponen hasta disponer de ese dato, resolviéndose en tiempo de ejecución y no en el código fijo.

Conocido el modelo y el contexto de conexión/transporte, se determina la variante concreta del protocolo mediante una Fábrica, tanto para construir los comandos/*scripts* como para interpretar respuestas, y se mantienen esas implementaciones activas para la sesión. Ante un modelo desconocido, se falla de forma controlada y se informa que no hay soporte para dicho modelo. Con este enfoque, se permite incorporar nuevos dispositivos registrando su variante en la Fábrica y ajustando configuraciones, sin modificar las capas superiores.

Use an Intermediary

Se coloca una Fábrica de variante entre la detección de modelo y el uso del protocolo. Con ese intermediario se decide qué implementación concreta se emplea sin que los consumidores conozcan el modelo. Como resultado, se localiza el cambio y la incorporación de un nuevo dispositivo se limita a "agregar la variante y registrarla".

4.4.4. Disponibilidad

La disponibilidad se sostiene aplicando tolerancia a fallas: el sistema detecta tempranamente condiciones anómalas y falla rápido mediante validaciones previas a cada orden; aísla la falla, mantiene el estado estable, libera recursos y comunica con claridad cuando una operación no puede ejecutarse.

Antes de cada orden se verifica el estado del transporte. Si el canal USB no está abierto, se procede a abrirlo; si la Wand no está conectada o el dispositivo implantable no está vinculado con la Wand, al intentar ejecutar la orden se muestra un mensaje que alerta de la situación.

Se asegura además que la sesión esté en condiciones válidas antes de procesar órdenes. Se comprueba que el dispositivo no se encuentre en modo *bootloader* y que la versión de protocolo sea compatible con la admitida por el sistema; de no ser así, se devuelve un mensaje de error claro y no se continúa. Se garantiza, mediante un mecanismo de inicialización del protocolo, que la conexión esté establecida y que la variante adecuada esté activa antes de enviar o interpretar datos.

Se establece que las operaciones de comunicación sean asíncronas y cancelables. Se asigna a cada operación su propio *token* de cancelación y un tiempo límite claro. Con este enfoque se evitan bloqueos, se permite abortar las operaciones que tardan mucho o que cambian de estado, se liberan recursos al instante y se mantiene la aplicación funcionando sin problemas. Se registra de forma consistente si una orden finaliza por cancelación o por tiempo excedido; los plazos por orden se parametrizan en *appsettings.json*, pudiéndose ajustar en despliegue.

Durante el intercambio de datos se valida la integridad de las tramas (p. ej., mediante CRC) y se distingue entre respuestas parciales, finales y mensajes de error. Ante corrupción o error reportado, se informa la condición y se evita propagar datos inválidos a capas superiores, preservando el estado operativo de la sesión.

Cuando no hay un dispositivo conectado, se mantiene utilizable el sistema: se permite consultar el historial local y generar informes con datos persistidos. Al intentar operaciones que requieren conexión, se alerta que no hay dispositivo disponible y no se ejecuta la acción.

4.4.5. Testeabilidad

En el dominio de *software* médico, obtener resultados de prueba rápidos y reducir la dependencia de *hardware* disminuye costo y riesgo. La arquitectura se orienta a que los fallos se revelen temprano, sean reproducibles y su causa se acote sin ambigüedad: el "*core*" se prueba principalmente en entorno simulado mediante *mocks* de transporte/protocolo; un conjunto acotado de pruebas de integración con dispositivo real valida la cadena de comunicación. También se realizan pruebas de sistema para verificar la funcionalidad de los requerimientos.

Specialized Interfaces

Se expone funcionalidad mediante contratos pequeños y estables y se compone por *Dependency Injection*, lo que permite sustituir módulos como transporte y protocolo por *mocks* de prueba.

Así se controlan entradas (*frame*s, estados, errores) y se observan salidas sin *hardware*, elevando la probabilidad de que un fallo aparezca en pruebas y sea reproducible. Esta táctica materializa "controlar y observar el estado del sistema" y baja drásticamente el costo de la pirámide de *tests*.

Localize State Storage

El estado de sesión (modelo detectado, versión de protocolo, inicializaciones, identificadores, etc.) se centraliza en un servicio único, con operaciones claras para sembrar, consultar y limpiar el estado. Al concentrar la "verdad" en un punto, se facilita iniciar pruebas en estados arbitrarios, repetir escenarios y acotar diagnósticos cuando algo falla. Esta táctica reduce la dispersión de estado y facilita la preparación, ejecución y reinicio del entorno de pruebas, haciendo más simple configurar casos reproducibles.

4.5. Justificaciones de diseño y patrones

En este apartado se fundamentan las decisiones arquitectónicas tomadas a partir del marco de tácticas de calidad de *Software Architecture in Practice* y su traducción a soluciones concretas del catálogo de patrones (GoF) y compilaciones contemporáneas como en *refactoring.guru* [16]. La aplicación de estos patrones y principios responde a la necesidad de satisfacer los atributos de calidad priorizados.

Transport Layer

En transporte se prioriza la Integrabilidad del medio. Se aplica *Strategy* mediante implementaciones intercambiables del puerto *ITransportService* (USB/FTDI hoy; otros medios mañana). Se resuelve la selección de la estrategia con *Abstract Factory* y *binding* diferido en inicialización (y se puede actualizar en ejecución según disponibilidad). Se expone un *Proxy* con contrato estable y se permite conmutar el transporte activo sin impacto en capas superiores; ante ausencia de medio, se informa indisponibilidad de canal de forma inmediata y no bloqueante. Para reaccionar a eventos físicos (*attach/detach*), se emplea *Observer*, de modo que se reciban señales del entorno y se aplique la política (abrir/cerrar, degradar). Bajo DIP, se depende del puerto *ITransportService*; se encapsula la dependencia con el *driver* USB en el adaptador concreto.

Motiva **RNF-INT-1-000**

Protocol Layer

Aquí se materializa la variabilidad por modelo de implante. Se combinan *Factory* + *Strategy*: tras iniciar sesión y conocer el modelo, se selecciona mediante una fábrica la estrategia de protocolo que define cómo construir tramas, verificar integridad y parsear respuestas. Se mantiene estable la API ofrecida hacia arriba; para añadir un modelo se suma una nueva estrategia y se registra en la fábrica, sin modificar consumidores (OCP). En el protocolo no se depende del transporte: se producen/consumen estructuras y tipos de dominio; el envío/recepción por el medio físico se realiza en la capa de transporte orquestada desde la *DeviceApi*.

Motiva **RNF-MOD-1-000**

DeviceApi

Se establece como Fachada del subsistema de comunicación: se ofrecen operaciones de alto nivel y se orquesta la secuencia de cada orden sin exponer detalles. Se aplican precondiciones (sesión válida, canal disponible), *timeouts* y cancelación por operación, y se clasifican resultados (éxito, cancelación, expiración, error de transporte, error del dispositivo). Bajo DIP, se depende de puertos (*IDeviceProtocol*, *ITransportService*) y no de concreciones; mediante *binding* diferido se asegura la elección del protocolo correcto tras conocer el modelo y se permite conmutar el medio activo sin afectar a consumidores.

Presentation (MVVM)

Se adopta MVVM para separar representación, estado de UI y lógica de interacción. Se renderizan y se enlazan *bindings/commands* desde las *Views*; se coordina la interacción desde las *ViewModels* y se consume el puerto de aplicación (*IAppService*) sin conocer protocolo ni transporte. Con este patrón se mantiene la UI agnóstica de detalles técnicos, se favorece el mantenimiento y la evolución de pantallas, y se facilita el manejo coherente de estados ante cancelaciones y *timeouts*.

4.6. Tecnologías Selccionadas

4.6.1. Estrategia de plataforma y *framework* multiplataforma (.NET 8 + .NET MAUI)

Desde las primeras reuniones con el cliente y a medida que se recaban los elementos que el sistema debe contener, se manifiestan algunos de los primeros requerimientos no funcionales y restricciones. Principalmente se requiere una aplicación multiplataforma que, en lo posible, se restrinja al ecosistema de .NET y, sí o sí, mantenga un desarrollo de la lógica de negocio y órdenes en C#, que es el lenguaje utilizado actualmente y facilita la migración de métodos propios.

Con estas premisas se inicia la investigación tecnológica. Se reduce el conjunto de opciones a .NET MAUI y Uno Platform. Ambas conservan C# y el ecosistema .NET; sin embargo, se adopta .NET MAUI por ser la opción oficial de Microsoft para desarrollo multiplataforma en .NET, por su integración nativa y *tooling* maduro en Visual Studio (IDE utilizado por el cliente) y por su documentación y comunidad activas. Con esta elección se reduce el riesgo, aspecto clave en *software* médico, y se acompaña con la selección de .NET 8 (LTS*, Long-Term Support*) como *runtime*, priorizando estabilidad y ventana de soporte extendida. Se valida la decisión mediante pruebas de concepto: con una misma base de código se generan *builds* para Windows y Android, se ejecutan *smoke tests* en emuladores/dispositivos y, en Windows, se verifica la cadena USB/FTDI.

En un inicio se proyecta concentrar toda la base de código en un único proyecto MAUI; posteriormente, para favorecer el desacoplamiento, la reutilización y la portabilidad, se separa la solución en dos módulos: un *frontend* en .NET MAUI siguiendo MVVM (UI en XAML y lógica de interfaz en C#) y un núcleo en .NET 8 (C#) que concentra la lógica de negocio y las órdenes del dispositivo. Con esta organización se mantiene la interfaz agnóstica del medio físico y del protocolo, se mejora la testeabilidad del núcleo y se simplifican futuras extensiones de plataforma o de comunicación sin reescrituras masivas.

4.6.2. Comunicación (USB + FTDI)

Desde el inicio se identifica la necesidad de sostener la comunicación propietaria con el dispositivo implantable mediante *USB* y *driver* FTDI.

FTDI (*Future Technology Devices International*) es un proveedor de circuitos integrados de interfaz *USB*-a-serie y de controladores asociados; su *driver* D2XX se expone con una API nativa para abrir, configurar, leer y escribir directamente sobre dispositivos basados en sus chips, sin pasar por puertos serie virtuales. Se examina la documentación oficial de FTDI Chip y del controlador D2XX, prestándose especial atención a las funciones básicas que ofrece, como la apertura y cierre del canal, su configuración, y la lectura y escritura de bytes. Sobre esa base se decide integrar el *driver* desde .NET a través de la librería administrada FTD2XX_NET.

En la documentación de FTDI se verifica disponibilidad de controladores para Android y ausencia para iOS; este punto se confirma en un intercambio directo con el proveedor, por lo que se define el alcance de las pruebas de despliegue en Windows y Android.

Se construyen pruebas de concepto para ambas plataformas incorporando los controladores de FTDI correspondientes. El resultado es: en Windows 10+ se logra una prueba de concepto funcional con comunicación correcta a través de USB/FTDI; en Android se observan problemas de compatibilidad al adoptar el *driver* dentro de la aplicación, quedando como trabajo futuro del cliente profundizar esa integración en el prototipo.

4.6.3. Persistencia

Para la elección de la tecnología de persistencia se definen criterios que debe cumplir la solución. Se prioriza una arquitectura multiplataforma que funcione sin conexión a *internet*, requiera administración nula o mínima y mantenga una huella liviana en el puesto clínico. Además, se busca buen desempeño para lecturas locales e inserciones moderadas, y que resulte sencilla de respaldar (copiando el archivo) y amigable para las pruebas.

En un comienzo se evalúa MySQL como motor, por ser el que utiliza el cliente en otros sistemas. Sin embargo, el contexto y la naturaleza de este proyecto son distintos: se apunta a una estación de trabajo local, usada por una persona a la vez, donde la aplicación se ejecuta en la misma PC que se conecta al *hardware* (Wand/USB) y no se comparte la base en simultáneo con otros equipos por red, con operación *offline* y despliegue sin infraestructura adicional.

En MySQL se requiere instalación, configuración, servicios en segundo plano, gestión de usuarios y red, lo que incrementa la complejidad operativa sin aportar beneficios en este escenario.

Por eso se opta por SQLite [17], que es embebido, multiplataforma, requiere cero administraciones, facilita el respaldo (archivo) y ofrece el rendimiento adecuado para lecturas locales e inserciones moderadas, alineándose mejor con los requisitos del prototipo.

Para el acceso a datos se utiliza Dapper [18] como *micro-ORM*: permite escribir SQL de forma clara, tiene muy poca sobrecarga y resulta sencillo de testear. Para los informes se adopta QuestPDF [25], que permite generar documentos de forma local y multiplataforma, con plantillas fáciles de mantener y resultados consistentes entre entornos.

5. Construcción

En esta sección se presenta el análisis del trabajo realizado y las decisiones tomadas por el equipo durante la etapa de construcción, así como la forma en que se planificaron y gestionaron los distintos *releases*.

Desde el inicio del proyecto, el equipo definió una división de responsabilidades: Julieta y Tomás se enfocaron en el desarrollo del *frontend*, mientras que Milena se encargó del *backend*. Sin embargo, esta distribución fue adaptándose a lo largo del desarrollo, ya que en determinadas etapas todos los integrantes debieron colaborar en el desarrollo del *backend* para cumplir con los objetivos planificados.

A excepción de Guillermo el cual asumió desde el principio el rol de *Tester*, manteniéndose parcialmente externo al desarrollo para evaluar el producto desde una mirada más objetiva. Esta perspectiva le permitió detectar fallos, inconsistencias o mejoras que podían pasar desapercibidas para quienes estaban directamente involucrados en la codificación.

La construcción del producto se planificó en dos *releases.* El cliente fue quien definió el alcance del primer y último *release*, estableciendo los requerimientos funcionales prioritarios que debían implementarse en cada una de las versiones. En el Anexo 12.3.1 se puede visualizar la confirmación de las funcionalidades acordadas para el desarrollo.

El primer *release* tuvo por objetivo desarrollar la comunicación con el dispositivo implantable contemplando las siguientes funcionalidades: Establecer conexión con el IPG, Detalles del IPG, Obtener la hora del IPG, Mostrar *Splash Screen* y Detectar antena Wand.

Durante la Fase 1 (la cual fue clave para poder completar con éxito los *releases*), centrada en la investigación y el análisis, se definió la arquitectura del sistema, concebida para ser extensible, de manera que permitiera incorporar nuevos modelos de dispositivos sin requerir una reestructuración del sistema. A partir de esa base, se avanzó hacia la interacción directa con el dispositivo implantable, incorporando mecanismos de intercambio de claves, reconocimiento del dispositivo y respuesta a órdenes específicas.

Las funcionalidades relacionadas con la comunicación entre el IPG y la Wand demandaron un entendimiento técnico más profundo de lo previsto, por lo que inicialmente fueron subestimadas

en su complejidad. Como consecuencia, se optó por dividir las órdenes en tareas más específicas. Por ejemplo, la funcionalidad Establecer conexión, se dividió en cuatro etapas: armar comando, enviar a la antena, recibir respuesta y mapear los datos.

Este requerimiento representó uno de los mayores desafíos del proyecto, ya que implicó adentrarnos en un entorno completamente nuevo. Si la respuesta del dispositivo no llegaba de forma completa, el comando quedaba incompleto, imposibilitando validar la comunicación.

Además, esta funcionalidad resultaba crítica para el avance del desarrollo, dado que sin una conexión establecida con el IPG no era posible continuar con la implementación del resto de los comandos dependientes.

El *Release* final tuvo como objetivo implementar una versión más avanzada con operaciones que implicaban la implementación de las funcionalidades restantes que completaban el alcance del proyecto. Entre ellas se incluyeron la Obtención de datos, la Visualización de alertas, la Visualización de informes, la Generación de logs, el Acceso a la sección avanzada, la Visualización del resumen avanzado, el módulo de *Trends* y la Visualización del historial de informes.

Durante este *release*, entre otras funcionalidades, en la Visualización de informes, la cual dependía directamente de la correctitud de los datos obtenidos a través del requerimiento Obtención de datos (véase Anexo 12.3.2, Control de valores tras la interrogación). Esta funcionalidad resultó clave para representar de forma clara y confiable la información proveniente del dispositivo implantable, asegurando la coherencia entre los valores obtenidos y los mostrados en la interfaz. Muchos de los requerimientos presentaban dependencias directas entre sí, ya que el funcionamiento de cada módulo estaba condicionado por la correcta ejecución del comando perteneciente.

Tanto para el *Release* 1 como para el *Release* 2, fue necesario contar con las matrices de trazabilidad actualizadas, gestionar los riesgos y dar seguimiento a los planes establecidos, lo que garantizó coherencia entre los entregables y cumplimiento de los estándares de calidad definidos. En cada uno de ellos se realizaron pruebas unitarias, de integración y de sistema, verificando el correcto funcionamiento de los módulos y la estabilidad del sistema antes de su entrega al cliente.

Cada *release* fue planificado con el objetivo de entregar versiones estables y funcionales, priorizando aquellas funcionalidades de mayor impacto para el cliente. Durante el desarrollo, el

equipo realizó validaciones internas y pruebas locales, lo que permitió detectar tempranamente errores de integración o comportamientos inesperados antes de cada entrega formal. El *feedback* recibido por el cliente era tomado en cuenta para la misma, lo que permitía que el *release* entregado cumpliera con lo acordado y priorizado.

Esta estrategia incremental facilitó ajustar la dirección del desarrollo, mantener una trazabilidad clara del progreso y asegurar que cada versión liberada cumpliera con los estándares de calidad definidos. Esta estrategia incremental y la retroalimentación temprana del cliente favoreció una mejor trazabilidad y control del avance.

6. Gestión de la calidad

En este capítulo se detallan los objetivos de calidad establecidos para el proyecto, diferenciando entre objetivos de producto y de proceso. Se plantea un plan de la calidad adaptado a las particularidades del desarrollo de un *software* médico, alineando lineamientos normativos requeridos por la FDA (*Food and Drug Administration*), tales como; la Guía de ciberseguridad y las normas ISO 14971 e IEC 62304.

Se describen las actividades realizadas para promover la calidad en cada etapa, que incluyen revisiones técnicas, validaciones y trazabilidad de requerimientos. Asimismo, se incorporan métricas de aseguramiento de calidad las cuales permiten verificar el cumplimiento de los objetivos planteados y se presenta una Matriz de riesgos del producto, fundamental en este tipo de desarrollo para reducir al mínimo el impacto ante posibles fallos. Véase el Anexo 12.4.1 por más detalle de la matriz.

En este proyecto la gestión de la calidad no se limita a la verificación del *software* desarrollado, sino que implica un enfoque integral de aseguramiento, lo cual abarca desde la planificación inicial hasta las pruebas finales, con el fin de garantizar la calidad del sistema en un entorno crítico como en el que se trabajó.

6.1. Objetivos de calidad

En esta sección se describen los objetivos de calidad definidos para el proyecto, los cuales se dividen en objetivos del producto y de proceso. Estos se plantean bajo el enfoque SMART [19] para que sean claros, medibles y alcanzables dentro del ciclo de vida. Entre los objetivos del proceso, se incluye la gestión y mitigación de riesgos, asegurando que el Plan de gestión de riesgos del proyecto y del producto se mantuvieran actualizados de forma continua.

6.1.1. A nivel de producto

Cobertura de pruebas: Se establece como meta alcanzar un porcentaje mayor o igual a 90 porciento de cobertura en pruebas unitarias, medido mediante la herramienta de *Code Coverage* integrada en Visual Studio, previo a la liberación del *Release* final.

Gestión de riesgos del producto: Se asegura que el 100% de los riesgos identificados en el producto finalicen con un nivel residual aceptable, según los criterios de la normativa ISO 14971, al culminar el proyecto.

6.1.2. A nivel de proceso

Alineación a normas: Promover el cumplimiento de un subconjunto de requisitos derivados del marco normativo, previamente acordados con el cliente, garantizando su trazabilidad y verificación dentro del ciclo de vida del proyecto. Estos se pueden visualizar en la sección 6.2.2.3.

Control de retrabajo: Se plantea que el retrabajo asociado a defectos se mantenga por debajo del 10% del esfuerzo total del equipo, medido en horas registradas en Clockify [27] al cierre de cada semana.

6.2. Planificación de la calidad

Se presenta la planificación de la calidad para el proyecto, cuyo propósito es establecer actividades, herramientas y controles necesarios para asegurar el cumplimiento de los objetivos de calidad planteados, así como también garantizar la calidad de todos los entregables en los que se trabaja. La misma se apoya en dos documentos centrales: el Plan de la calidad, el cual se enfoca en la metodología de trabajo, la definición de criterios de aceptación y la finalización de tareas, y el Plan de *software assurance*, el que abarca actividades de aseguramiento, estándares, revisiones, verificaciones, validaciones, pruebas y resolución de problemas, así como métricas relativas a la calidad.

6.2.1. Plan de la calidad

En esta sección se establece cómo se gestionan y controlan las tareas del proyecto para asegurar el cumplimiento de los objetivos planteados, integrando revisiones, validaciones y verificaciones necesarias.

El Plan de la calidad definido incluye las actividades realizadas a lo largo de todas las fases del proyecto: investigación y análisis, diseño, construcción, pruebas y procesos de gestión. La tabla incluye para cada actividad sus entradas, salidas, herramientas y/o técnicas utilizadas y los roles

responsables y participantes, lo que permite visualizar de forma clara cómo se gestionó principalmente cada etapa dentro del Plan de la calidad. Véase el Anexo 14.4.2 por más detalle de la Matriz del plan de la calidad.

6.2.2. Plan de *software assurance*

En este plan se establecen las actividades, estándares y mecanismos aplicados para asegurar la calidad del producto a lo largo de todo el ciclo de vida del proyecto. Este plan contempla la alineación a normas, la realización de revisiones internas y con el cliente, la verificación y validación de los entregables, así como la ejecución de pruebas y la resolución de problemas. Además, integra prácticas de gestión de riesgos y control de trazabilidad, en conformidad con normativas internacionales aplicables al *software* médico (IEC 62304, ISO 14971), procurando que el prototipo desarrollado cumpla con los niveles de calidad necesarios al tratar con este tipo de entorno clínico.

6.2.2.1. Responsables del aseguramiento y control de la calidad

En esta sección se presentan los roles y responsabilidades vinculados al aseguramiento y control de la calidad, con el fin de garantizar una adecuada planificación, ejecución de actividades y resolución de hallazgos a lo largo del proyecto.

Rol	Nombre	Responsabilidades
Líder de SQA	Julieta Sarantes	Es responsable por el aseguramiento de la calidad de los productos generados en el proyecto. También es responsable de la planificación, seguimiento y evaluación de las actividades de este plan.
SQA	Julieta Sarantes, Guillermo Echichure	Es responsable de la ejecución de las actividades definidas en este plan. Asimismo, es responsable de la identificación de hallazgos referentes a la calidad del producto o del proceso y de asegurar su resolución.
Tester	Guillermo Echichure	Es responsable de la planificación y ejecución de las actividades de prueba del producto. Además, es responsable de la identificación de hallazgos referentes a la calidad del producto y de asegurar su resolución.

Figura 17: Tabla de roles

La Figura 17 muestra la asignación de roles vinculados al aseguramiento de la calidad, incluyendo al líder de SQA, al equipo de SQA y el *Tester*, junto con las responsabilidades específicas de cada uno.

El Rol de Tester especializado

En nuestro equipo, el rol de *Tester* especializado fue asumido por Guillermo, quien de manera planificada no participaba en la codificación diaria, con el objetivo de evaluar el producto desde una perspectiva independiente y objetiva. Esa distancia del código le permitió detectar detalles que a los desarrolladores se les escapaban, porque estaban más centrados en la construcción. Fue él quien tomó la responsabilidad de crear el Plan de pruebas y de llevar adelante las pruebas de integración y de sistema, transformándose en una especie de "filtro final" que permitía verificar que el producto respondiera de acuerdo con el comportamiento esperado.

6.2.2.2. Aplicación de estándares

Desde un principio se definieron estándares que guiaron el desarrollo y aseguramiento de la calidad del proyecto. Se incorporaron tanto lineamientos propios establecidos por el equipo, como prácticas y recomendaciones provenientes de normativas internacionales y comunicaciones periódicas con el cliente, con el fin de garantizar consistencia, trazabilidad y alineamiento regulatorio como se establece en los objetivos.

Estándares de codificación

Uso de idioma inglés: Todo el código fuente se escribió en inglés siguiendo los estándares de la industria del desarrollo de *software*. Esto garantiza una mayor claridad, uniformidad y comprensión a nivel global, lo cual facilita la colaboración y el mantenimiento de nuestro proyecto.

Aplicación de principios de diseño y buenas prácticas: Se implementaron los principios *SOLID* [20], fundamentales para la programación orientada a objetos. Estos principios actúan como guías para los desarrolladores y promueven un código modular, mantenible y eficiente, asegurando una mejor estructura y escalabilidad en el *software*.

Asimismo, se adoptaron los lineamientos de *Clean Code* [21], que garantizan una codificación estructurada, clara y de fácil mantenimiento, optimizando el desarrollo y reduciendo costos a largo plazo.

Estándares de documentación

Se definieron también lineamientos para la documentación interna del proyecto, los cuales se definieron por el equipo para asegurar la homogeneidad de estos. Se crearon plantillas que facilitaron el cumplimiento de los estándares definidos, para asegurar la uniformidad y claridad de estos. Estos documentos incluyen información estructurada como título, versión, fecha, responsable y revisores lo que permitió mantener un control formal sobre los cambios y facilitó los procesos de revisiones.

Documento	**Estándar**
Plan de calidad	Template ORTsf modificado por el equipo
Plan de *software assurance*	Template ORTsf modificado por el equipo
Plan de riesgos del producto	Definido por el equipo *
Plan de riesgos del proyecto	Template ORTsf modificado por el equipo
Plan de pruebas	Definido por el equipo *
Matriz de trazabilidad integral	Definido por el equipo *
Matriz de ejecución y resultados de pruebas	Definido por el equipo *
Matriz de riesgos del producto	Definido por el equipo *
Matriz de riesgos del proyecto	Template ORTsf modificado por el equipo
ESRE	Definido por el equipo *
Documentación final	Documentos 302, 303, 304 y 306

(*) La definición de estándares realizada por el equipo se encontró alineada con las normativas aplicables al desarrollo de *software* médico. La norma ISO 14971 establece la gestión de riesgos como eje central, lo que exige que cada requisito, diseño y prueba contemple la identificación y mitigación de riesgos asociados al producto; por lo cual se alineó correctamente al tener las matrices relacionadas al producto mencionadas anteriormente. Por su parte, la norma IEC

62304 se apoya en el Modelo V como referencia para describir las fases del ciclo de vida del *software* médico. Sin embargo, no exige que este modelo se cumpla de manera estrictamente secuencial, sino que requiere que las actividades realizadas en el proyecto se puedan mapear con las establecidas en dicho ciclo de vida. La norma demanda que todos los procesos, actividades y tareas definidos por el Modelo V se encuentren completos y documentados, incluyendo de forma obligatoria la existencia de actividades de verificación y validación, independientemente de la metodología de desarrollo aplicada.

6.2.2.3. Subconjunto de requisitos derivados del marco normativo

Durante la fase de análisis e investigación, se relevaron las principales normas aplicables al desarrollo de *software* médico, incluyendo las recomendaciones de la FDA y la Guía de Ciberseguridad para dispositivos médicos. A partir de su lectura y análisis, se identificaron una serie de puntos clave a cumplir para asegurar el alineamiento del proyecto con dichos estándares:

- Mapear las actividades del Modelo V con las etapas del proyecto, alineado a la norma IEC 62304.
- Gestionar los riesgos del producto por separado
- Contar con un Plan de gestión de riesgos del producto, alineado a la norma ISO 14971, con enfoque en identificación, mitigación y trazabilidad de los riesgos.
- Ejecutar pruebas de integración para los módulos identificados en el diseño que no puedan probarse de manera unitaria o identificados con algún riesgo asociado al producto.
- Realizar pruebas de sistema para cada requerimiento funcional definido en el ESRE.

Puntos clave	Actividades
Mapear las actividades del Modelo V con las etapas del proyecto, alineado a la norma IEC 62304.	Matriz de trazabilidad integral.
Gestionar los riesgos del producto por separado	Matriz de riesgos del producto y matriz de riesgos del proyecto independientes.
Contar con un plan de gestión de riesgos del producto, alineado a la norma ISO 14971, con enfoque en identificación, mitigación y trazabilidad de los riesgos.	Plan de gestión de riesgos del producto.
Ejecutar pruebas de integración para los módulos identificados en el diseño que no puedan probarse de manera unitaria o identificados con algún riesgo asociado al producto.	Pruebas de integración en driver FTDI D2XX, Conexión/Desconexión de USB y acceso a base de datos.
Realizar pruebas de sistema para cada requerimiento funcional definido en el ESRE.	Matriz de ejecución y resultados de prueba.

Figura 18: Tabla de mapeo de subconjunto de requisitos normativos

6.2.2.4. Gestión de los entregables y actividades de calidad

En esta sección contamos cómo organizamos y controlamos nuestros entregables, asegurando que cada avance pasara por actividades de calidad que nos permitieran mantener el rumbo del proyecto y cumplir con lo que habíamos definido junto al cliente.

6.2.2.4.1. Especificación de requerimientos

La especificación de requerimientos se realizó mediante casos de uso, con el fin de que resulten claros, completos y verificables, asegurando su correcta interpretación e implementación. Tanto los requerimientos funcionales (RF) como los no funcionales (RNF) se registraron en el documento ESRE con identificadores únicos, lo que garantizó su trazabilidad y versionado a lo largo del ciclo de vida, contemplando la documentación normativa y técnica relevada durante la fase de investigación y análisis. De este modo, por ejemplo, se consideró la guía de ciberseguridad para la autenticación de usuarios.

Revisión

Para asegurar consistencia y claridad, se realizaron revisiones cruzadas dentro del equipo, de modo que un integrante no involucrado directamente en la especificación aportara una visión externa que permitiera detectar posibles inconsistencias o ambigüedades, evitando retrabajo posterior. A su vez, se llevaron a cabo reuniones periódicas con el cliente, asegurando que los requerimientos reflejaran correctamente las necesidades planteadas y se mantuvieran alineados con los objetivos del proyecto. Véase el Anexo 12.4.3 para ejemplo de reunión de revisión del ESRE.

Estas instancias de revisión permitieron detectar tempranamente omisiones o necesidades de ajuste, reduciendo riesgos o deudas técnicas en etapas posteriores. Según los avances del proyecto, se elaboró una minuta que servía como guía para relevar información y obtener insumos necesarios para la siguiente etapa (Véase el Anexo 12.4.4 para ejemplo de minuta). Mientras que la comunicación cotidiana se reforzó con intercambios ágiles por WhatsApp, lo que facilitaba resolver inquietudes de manera inmediata.

Verificación

Se aseguró que los requerimientos sean claros, completos y consistentes. Para ello se aplicó una Tabla de verificación (ver Anexo 12.4.5), conteniendo una lista de verificación (Véase el Anexo 12.4.6 para Lista de verificación del ESRE) la cual comprobó que los mismos cumplían con la especificación adecuada, por lo que se validó que tanto los requerimientos funcionales (con sus respectivos casos de uso), como los no funcionales contaban con identificadores únicos, garantizando su trazabilidad y versionado a lo largo del ciclo de vida. De esta forma, se obtuvo un documento coherente cumpliendo las normativas y apto para servir como base en las actividades de diseño, implementación y validación.

Validación

La validación del ESRE se realizó en conjunto con el cliente, con el cual se le compartía pantalla por Google Meet, una de las cuales se puede visualizar en el Anexo 12.4.7. De esta manera, se aseguró que los requerimientos especificados reflejaran correctamente sus necesidades y expectativas antes de avanzar con la arquitectura e implementación. El *feedback* recibido en estas instancias se registró en minutas, y se incorporó en las versiones posteriores del documento.

6.2.2.4.2. Diseño arquitectónico

En cuanto al diseño arquitectónico se centró en definir la estructura de alto nivel del sistema, estableciendo los módulos principales, sus responsabilidades y las interacciones entre ellos. Esta etapa permitió asegurar que la solución propuesta tuviera una base sólida, escalable y alineada con los requerimientos funcionales y no funcionales identificados en la etapa anterior.

El resultado de esto es una visión global que orienta las fases posteriores de diseño detallado e implementación, y que construye un buen marco de referencia para garantizar una buena arquitectura.

Revisión

Se llevaron a cabo revisiones con el cliente mediante videollamadas, en las cuales se presentaron los diagramas de arquitectura y se recabó retroalimentación sobre las decisiones tomadas, para así ser capaz de resolver inconcordancias en una etapa temprana. Paralelamente, se mantuvieron reuniones internas del equipo destinadas a analizar alternativas y definir si la estructura global (ej. separar *backend* de *frontend*, persistencia, comunicación con dispositivo, arquitectura en capas) era la adecuada. Véase el Anexo 12.4.8 para ejemplo de reunión de refinamiento de diseño arquitectónico.

Validación

La validación del diseño arquitectónico se realizó en llamada en conjunto con el cliente, mediante la presentación y discusión de los diagramas de arquitectura para confirmar que la estructura planteada reflejaba sus necesidades y expectativas. Así como también los requerimientos no funcionales definidos para la misma. Véase el Anexo 12.4.9 para ejemplo de reunión de validación de arquitectura del *Release* 1.

6.2.2.4.3. Diseño detallado

Se orientó a especificar la estructura interna de los componentes definidos en la arquitectura, describiendo con precisión las clases, métodos, patrones de diseño y estructuras de datos que conforman el sistema.

Revisión

Las revisiones del diseño detallado se realizaron en reuniones internas del equipo, en las cuales se analizaron los diagramas de clases, secuencia y componentes, evaluando la correcta aplicación de principios de diseño y patrones definidos por el equipo. Se discutieron alternativas técnicas, como la utilización de determinados patrones en módulos específicos, así como también el cumplimiento de los atributos de calidad definidos. Véase el Anexo 12.4.10 para ejemplo de reunión de revisión de diseño detallado

Validación

Se realizó la validación mediante la presentación de los diagramas completos al cliente, con el objetivo de confirmar que la estructura planteada reflejaba las funcionalidades esperadas y los escenarios de uso definidos. En caso de existir desacuerdos, se estaba en una etapa temprana lo que evitaba tener desvíos en etapas posteriores. Véase el Anexo 12.4.11 para ejemplo de reunión de validación de diseño detallado y el Anexo 12.4.9 para ejemplo de reunión de validación de arquitectura del *Release* 1.

Verificación diseño arquitectónico y detallado

La verificación del diseño arquitectónico y del diseño detallado se orientó a confirmar en equipo que la estructura propuesta y las clases/métodos definidos cumplieran con los requerimientos establecidos en el ESRE y con los lineamientos normativos. Para ello, se revisó que el diseño contemplara la identificación de módulos principales y la adecuada gestión de riesgos, así como la correcta vinculación entre requerimientos, componentes arquitectónicos y elementos de diseño detallado. Esta relación quedó documentada la sección 6.3.4.2 donde se muestra la Matriz de trazabilidad integral, utilizada como evidencia principal del proceso de verificación.

6.2.2.4.4. Codificación

La etapa de codificación se orientó a transformar el diseño detallado en código fuente ejecutable, siguiendo los estándares de programación definidos y las buenas prácticas establecidas por el equipo. En esta fase, la verificación no se presenta como una sección diferenciada, ya que se encuentra implícita en las actividades de revisión de código. De este modo, los *pull requests*,

las revisiones cruzadas y los *releases* intermedios permitieron comprobar que el código implementado mantenía coherencia con el diseño, cumplía con los estándares establecidos y se integraba correctamente antes de ser validado por el cliente.

Revisión

El equipo adoptó un flujo de trabajo basado en *pull requests* y revisiones cruzadas dentro del equipo. Cada modificación propuesta debía ser evaluada por al menos otro integrante antes de su integración a la rama *develop*, en caso de estar de acuerdo se daba *approve* y quedaba listo para ser integrado. Estas revisiones cruzadas tenían como propósito fomentar la discusión sobre la implementación, a su vez, la persona que no participaba de la implementación tenía otra mirada sobre el código, lo que ayudaba a aumentar la calidad y estabilidad en el mismo.

A continuación, en la figura 19, se presenta una captura de pantalla la cual ejemplifica cómo se gestionaron los *pull requests* en nuestro equipo.

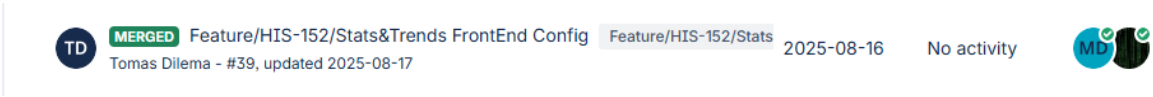


Figura 19: Ejemplo pull request

En esta captura se visualizan algunos ejemplos de los *pull requests* integrados durante el desarrollo. También se registraron casos con estado *declined*, correspondientes a solicitudes que no cumplían con los criterios establecidos. Asimismo, en determinados momentos se utilizaba *draft pull requests*, empleados para compartir avances parciales y permitir revisiones tempranas antes de su integración definitiva. Véase el Anexo 12.4.12 para detalle de un *pull request*.

Validación

Se realizaron validaciones sobre los *releases* generados, siendo el primero y el final los hitos más relevantes. Durante estas validaciones, se revisó junto al cliente que cada funcionalidad cumpliera con el comportamiento esperado y respondiera correctamente a los flujos establecidos en los casos de uso. Estas instancias permitieron comprobar el funcionamiento de la solución, contrastarla con lo esperado por el cliente e identificar ajustes necesarios. Además, permitieron verificar el cumplimiento de los objetivos de cada *release*. Lo que aseguraba que la solución desarrollada respondía a los objetivos definidos y contaba con alto grado de aceptación en su contexto de uso. Véase el Anexo 12.4.13 y 12.4.14 para evidencia de validación del *release* 1 y final, respectivamente.

6.2.2.5. Validación del producto final

La validación del producto final fue una etapa esencial para evaluar que la solución desarrollada por el equipo cumplía con los objetivos y el alcance definido. Por lo cual, se hicieron instancias de validaciones, por un lado, con el cliente, el cual evaluó directamente el producto contemplando todos los casos de uso definidos, y por otro, con potenciales usuarios finales, en este caso cardiólogos, cuya perspectiva y sabiduría permitió enriquecer tanto el contexto, como la solución, dando al equipo una comprensión real sobre la utilidad clínica de la información presentada.

Validación con cliente

El cliente validó de forma directa tanto el primer *release* (ver Anexo 12.4.13) como el *release* final (ver Anexo 12.4.14), lo que aseguró que la solución desarrollada respondiera a los objetivos definidos y contara con un alto grado de aceptación en su contexto de uso. La validación del producto final se llevó a cabo en las oficinas de *Impulse Dynamics*, donde el cliente configuró el dispositivo implantable de manera que los valores obtenidos fueran correctos y coincidieran con lo esperado, permitiendo así realizar pruebas en un contexto de uso más realista. Una de las funcionalidades a destacar tanto en la validación final como en anteriores, fue la de visualizar el informe en formato PDF, le gustó el formato y estaba muy conforme con lo que se había logrado. Véase el Anexo 12.4.15 para el *feedback* del informe PDF generado.

A su vez, el cliente envió un correo de evaluación del proyecto, en el cual manifestó estar muy conforme con los resultados obtenidos, destacando el trabajo realizado por el equipo. Si bien mencionó que no pudo participar tan activamente como hubiera deseado, debido a sus compromisos laborales y entregas simultáneas a nuestra fase de desarrollo, valoró positivamente nuestro producto. Véase el Anexo 12.4.16 para Correo de evaluación del cliente.

Validación con potenciales usuarios finales

Aunque para el cliente no resultaba indispensable realizar validaciones con usuarios finales, se consideró necesario conocer la opinión y perspectiva de los cardiólogos respecto al producto final. De esta instancia se obtuvo información enriquecedora que permitió comprender en mayor profundidad qué datos se presentaban y cuál era su utilidad clínica. Esta validación contribuyó a dimensionar la relevancia del sistema desarrollado y a reafirmar el valor que el producto aporta a los profesionales que lo utilizarán.

Dra. Gabriela Morales - Cardióloga

Un momento clave de la validación se produce en la reunión mantenida con la Dra. Gabriela Morales, cardióloga de la Asociación Española, cuya participación permite confirmar la pertinencia clínica de los datos mostrados en la aplicación. Durante la sesión destaca que *"esos cinco parámetros son las cosas que yo tengo que saber"*, validando así la relevancia de la información presentada en la pantalla de resumen. Asimismo, se sugiere la incorporación de mejoras, *"sería conveniente que las alertas tengan una señal auditiva cuando se activan"*. Estas observaciones aportan información valiosa para la utilidad del sistema en la práctica médica. Siguiendo con sus observaciones se destaca la importancia de poder visualizar la batería restante del dispositivo, la impedancia que tiene cada cable, si le puedo ajustar los mV de la salida del estímulo eléctrico de cada uno de estos, y que, si algo no anda bien que se apague, *"me gusta ver algo así, con la información importante, concreta y que los médicos sepamos entender"*. Véase el Anexo 12.4.17 para capturas de la reunión con la Dra.

Dr. Ángel Chaia - Cardiólogo

La validación con el cardiólogo Ángel Chaia, Cardiólogo de COMERI Normedica y en el hospital de Artigas, nos dio mucha información que el equipo no poseía. Nos contó datos relevantes sobre el funcionamiento del cuerpo y del dispositivo en el mismo, pero también nos dio una buena retroalimentación con respecto al prototipo.

El Dr. Chaia destacó la importancia de contar con un monitoreo continuo y confiable, ya que permite detectar alteraciones en el ritmo cardíaco. Señaló que la posibilidad de que el médico acceda en tiempo real a los parámetros del paciente representa un cambio sustancial en la atención, especialmente en situaciones de emergencia. Mencionó además que este tipo de tecnología mejora la calidad de vida de los pacientes, haciendo que puedan tener un mejor estilo de vida.

En cuanto al prototipo le pareció muy útil la información brindada en informe en formato PDF y el resumen avanzado, *"es la información que debemos tener al alcance para sacar las principales conclusiones del paciente"* señaló Chaia.

En el Anexo 12.4.18 se incluye la minuta de la reunión con el Dr. Ángel Chaia, junto con una del encuentro en el Anexo 12.4.19, que evidencian la instancia de validación realizada con el especialista y respaldan los aportes obtenidos.

6.2.2.6. Resolución de problemas

Como parte del compromiso con la calidad y la mejora continua se definió un proceso de seguimiento de problemas, el cual permitió establecer con claridad cómo registrar, clasificar y dar trazabilidad a los problemas identificados durante el ciclo de vida del proyecto. Este plan constituyó una parte esencial del presente documento, dado que la gestión efectiva de incidentes resultó fundamental dentro de un entorno tan regulado como este. De esta manera, se aseguró que los errores y deudas técnicas se abordaran de forma sistemática y documentada, se mantuviera visibilidad sobre los problemas encontrados y su estado. Además, se contribuyó a la correctitud y transparencia del proceso, fortaleciendo las actividades de verificación y validación. Véase el Anexo 12.4.20 para Proceso de seguimiento de problemas.

Complementariamente, se utilizó una matriz para la trazabilidad de los problemas, con columnas correspondientes a su nombre definido por: el identificador el cual se forma con el tipo de problema y su respectivo número, tipo de problema, descripción, y criticidad, lo que facilitó el seguimiento detallado de cada incidente a lo largo del proyecto. Véase el Anexo 12.4.21 para Matriz de trazabilidad de problemas.

6.2.3. Plan de riesgos del producto

DynamicScan es un *software* que se integra en un contexto de uso médico, por lo que es un requisito gestionar los riesgos del producto, los cuales se definen como la combinación de la probabilidad de ocurrencia de un daño y la severidad de ese daño para el paciente, el usuario o el entorno, resultante de peligros asociados con el *software* o su uso.

Para llevar adelante la gestión del riesgo del producto se estableció un plan basado en la norma ISO 14971 la cual identifica una serie de etapas clave para realizarlas.

Dichas etapas fueron:

- Análisis del riesgo
- Estimación del riesgo
- Valoración del riesgo
- Control del riesgo
- Evaluación del riesgo posterior al control

Esta última etapa implica evaluar el riesgo residual, (en el análisis de riesgos de un *software* crítico siempre existe un riesgo residual tras las medidas de mitigación), se acordó con el cliente que la evaluación del riesgo residual se iba a hacer de forma binaria: hay riesgo residual o no hay riesgo residual para simplificar el análisis y cumplir con los plazos de entrega del proyecto.

En la norma existen dos etapas más: producción y postproducción las cuales fueron omitidas debido a que el producto entregado es un prototipo y no llega a esas etapas.

6.2.3.1. Análisis del riesgo

El análisis del riesgo implicó identificar aquellos peligros que pueden presentarse a lo largo del desarrollo del producto. Este proceso fue difícil porque obliga a los desarrolladores a pensar en términos de peligros para los cuales el equipo no tenía experiencia previa. El enfoque se basó en una serie de categorías que la norma recomienda como forma de poder clasificar los peligros para finalizar con la etapa de estimación del riesgo.

Se comparte aquí dichas categorías:

- Uso previsto y uso erróneo razonablemente previsible.
- Identificación de las características relacionadas con la seguridad.
- Identificación de los peligros y situaciones peligrosas.
- Identificación de los riesgos técnicos o funcionales internos

6.2.3.2. Estimación del riesgo

Para cada riesgo identificado, se calculó su valor en base a dos factores:

- Probabilidad de ocurrencia (escala compartida por el cliente (ver Anexo 12.4.22).
- Severidad del daño potencial causado al paciente (escala compartida por el cliente ver Anexo 12.4.23).

6.2.3.3. Valoración del riesgo

El cliente define el valor del riesgo mediante un producto cartesiano entre (probabilidad x severidad) (ver Anexo 12.4.24) a diferencia de un cálculo numérico clásico. Además de simplificar el modelo, el objetivo principal es determinar la necesidad de acciones de control más que obtener un valor numérico exacto del riesgo.

La gestión del riesgo del producto se documentó en la Matriz de gestión de riesgos del producto, véase la misma en el Anexo 12.4.1.

6.2.3.4. Control de riesgo

Algunas de las medidas de control adoptadas en este prototipo implicaron:

- Validación de integridad de los datos recibidos, verificando la correcta implementación de la función de CRC definida en el protocolo FP17, con el fin de detectar y descartar tramas corruptas.

- Validación de respuestas completas frente a respuestas parciales, implementando los mecanismos de verificación previstos en el protocolo FP17 y configurando adecuadamente los *time out* para permitir el correcto uso de la lectura y escritura.

- Integración controlada con componentes externos, asegurando el uso correcto del *driver* FTD2XX para la comunicación USB y del motor de base de datos SQLite para el almacenamiento local de sesiones.

- Canales de retroalimentación al usuario, mediante mensajes informativos que notifican estados de error o desconexión de la antena Wand y del dispositivo IPG, favoreciendo la detección temprana de incidentes.

- La verificación de estas medidas se ha realizado mediante pruebas unitarias, pruebas de integración, pruebas de sistema y *code review.*

6.2.3.5. Evaluación del riesgo residual

Esta etapa comprende la evaluación del riesgo residual luego de aplicar las medidas de control de mitigación. Los valores son:

- NO – se considera mitigado.
- SI – se considera como riesgo residual no aceptado y se analiza como poder mitigarlo con otra acción.

6.2.3.6. Archivo de gestión del riesgo

La norma establece que el fabricante debe establecer y mantener un archivo de gestión del riesgo que proporcione trazabilidad de cada peligro identificado respecto a:

- El análisis del riesgo
- La valoración del riesgo
- La implementación y verificación de las medidas del control del riesgo
- Los resultados de la evaluación de los riesgos residuales.

Aclaración de la norma: el archivo de gestión de riesgo puede estar en cualquier formato o tipo de medio.

Por lo anterior, el registro de todas estas actividades se lleva a cabo en la Matriz de riesgo del producto creada para tal fin. Por detalles ver sección 6.3.4.1 Matriz de riesgo del producto.

6.3. Pruebas de *software*

Aquí se describe la estrategia de pruebas adoptada para el prototipo DynamicScan. Su propósito es realizar las verificaciones y validaciones producto de las diferentes actividades del Modelo V: Ing. de requerimientos, Diseño arquitectónico y Diseño detallado, y a su vez, las mitigaciones de los riesgos identificados en la Matriz de riesgos del producto. Se trabajaron a tres niveles basados en el Modelo V cada uno con un propósito específico.

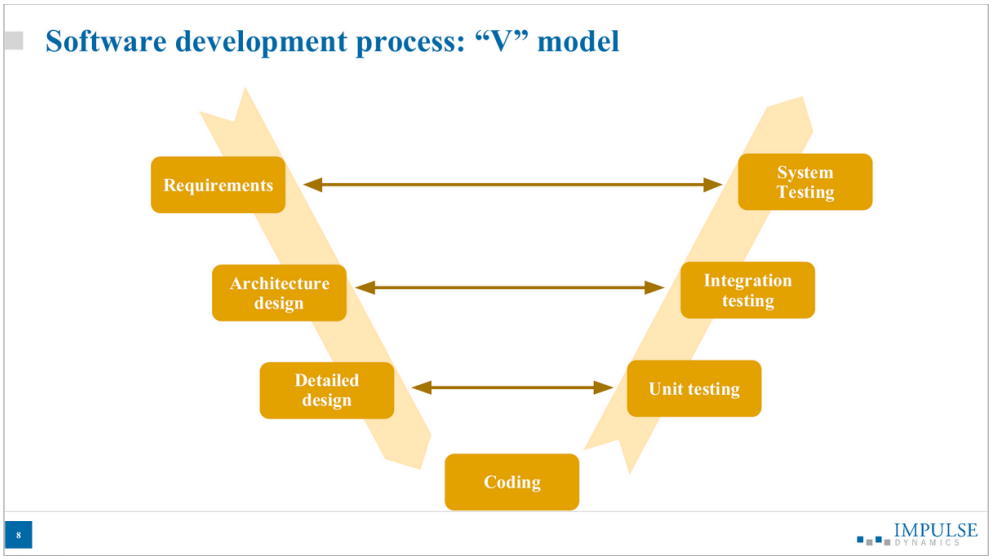


Figura 20: Modelo V

- **Pruebas unitarias:** orientadas a verificar la lógica interna de métodos y funciones individuales en forma aislada. En este modelo se encuentra la relación con el diseño detallado.

- **Pruebas de integración:** orientadas a validar la interacción entre los módulos que dependen del *driver* FTDI y el *hardware* físico de la antena Wand, comprobando la coherencia entre la arquitectura propuesta y su comportamiento real. También se incluyeron pruebas sobre la base de datos SQLite, independientes del *driver*, destinadas a verificar el correcto funcionamiento del motor de persistencia y mitigar el riesgo de errores funcionales en el acceso a los datos.

- **Pruebas de sistema:** dirigidas a comprobar el cumplimiento de los requerimientos funcionales (RF) y los no funcionales (RNF) que dependen de una prueba de sistema, definidos en el ESRE y casos de uso (CU). Ver Anexo 12.2.1 para ESRE *Release* final y Anexo 12.2.1.2 para casos de uso.

En la práctica, el cliente combina las pruebas de integración con las pruebas de sistema, validando la integración al final de la prueba. La norma IEC 62304 establece que: *It is acceptable to combine integration testing and software system testing into a single plan and set of activities.* (*International Electrotechnical Commission*, 2015)

En nuestro caso, seguimos ese mismo enfoque, con la excepción mencionada en las pruebas de integración. Véase Anexo 12.4.25 para motivos de la priorización.

El registro de los resultados queda documentado en la Matriz de ejecución y resultados (véase Anexo 12.4.26) y su trazabilidad con las actividades del brazo izquierdo del Modelo V queda registrado por la Matriz de trazabilidad integral que vincula RF, CU, pruebas y componentes arquitectónicos. Esta se visualiza en la sección 6.3.4.2, Matriz de trazabilidad integral.

6.3.1. Diseño de pruebas

Las pruebas unitarias se desarrollaron siguiendo el patrón *Arrange, Act, Assert* (AAA), un enfoque estructurado recomendado en .NET para separar claramente la preparación del escenario (*Arrange*), la ejecución del comportamiento bajo prueba (*Act*) y la verificación de resultados (*Assert*) [22].

Las pruebas de integración fueron planificadas para cubrir escenarios específicos como Wand conectada, Wand desconectada y estado USB, y también se estructuraron bajo el patrón *Arrange, Act, Assert* (AAA). Estas pruebas se alojaron en un proyecto separado, conforme a las buenas prácticas recomendadas en .NET para separar pruebas de integración y mantener un entorno de pruebas claro y organizado [23].

Para ver la estructura de los proyectos separados véase el Anexo 12.4.27.

Las pruebas de sistema se corresponden directamente con los RF trazados a CU con sus cursos normales y alternativos.

6.3.2. Responsables

El equipo definió los siguientes responsables de pruebas:

Nivel de prueba	Responsable
Pruebas Unitarias	Equipo de desarrollo
Integración	Guillermo (*Tester*)
Sistema	Guillermo (*Tester*)

6.3.3. Registro y documentación

Con el objetivo de mantener la trazabilidad y facilitar el seguimiento, se definió un formato estándar para cada tipo de prueba:

1. **ID de prueba:** PXy-z, donde X indica el tipo (U = unitaria, I = integración, S = sistema), y el número de prueba y, z la versión (se utilizan tres dígitos comenzando del 000).

2. **Requerimiento vinculado:** según el ESRE.

3. **Caso de uso asociado, en caso de corresponder con nomenclatura:** CUx-y, donde "x" indica número de caso de uso e "y" el número de versión (se utilizan tres dígitos comenzando del 000).

4. **Riesgo asociado, en caso de corresponder, vinculado a la matriz de riesgos, con nomenclatura:** RProd-x-y, donde “x” indica el número e “y” el número de versión (se utilizan tres dígitos comenzando del 000).

5. Resultado esperado y obtenido.

6. Observaciones relevantes.

Finalmente, el registro se realizó en la Matriz de ejecución y resultados.

6.3.4. Material de apoyo

El plan se apoya en los siguientes artefactos:

6.3.4.1. Matriz de riesgo del producto

La Matriz de riesgos del producto constituye un insumo fundamental dentro del proceso de verificación y validación del prototipo. Ver el Anexo 12.4.1.

Su elaboración fue progresiva, inició en las etapas tempranas del proyecto y continuó en paralelo al avance del desarrollo y de la adquisición de conocimiento por parte del equipo, donde el mismo se enfrentaba por primera vez a un proceso formal de análisis de riesgos de un *software* en particular médico, y bajo una norma que dicta cómo debe ser ese proceso, ISO 14971.

El propósito inicial de la matriz fue identificar, evaluar y controlar los posibles riesgos que pudieran comprometer la seguridad, la funcionalidad o la confiabilidad del prototipo, considerando tanto los aspectos técnicos del producto (arquitectura, dependencias externas, comunicación *hardware-software*) como los relacionados con su uso previsto.

Estas categorías fueron tomadas propiamente de la guía para apoyar el proceso de identificación el cual fue complejo e iterativo.

El orden de ejecución estuvo condicionado por múltiples factores propios del contexto académico y técnico del proyecto.

Entre ellos, la disponibilidad del *hardware* del cliente (antena Wand y dispositivo implantable - IPG), el avance del código en cada capa, la factibilidad de realizar pruebas unitarias sobre determinados componentes y en algunos casos, la complejidad técnica de las pruebas.

Aún con esas limitaciones, la matriz fue clave para:

- Visibilizar los riesgos que a juicio del equipo fueron las más críticos, especialmente a los asociados al uso del *driver* FTDI D2XX, a la integridad de los datos y a la comunicación Wand-IPG.

- Definir medidas de control concretas y verificables, muchas de las cuales se tradujeron en pruebas de *software* ejecutadas efectivamente.

- Asegurar la cobertura de los riesgos relevantes dentro del conjunto de pruebas del proyecto, garantizando la trazabilidad entre riesgo, mitigación y evidencia de verificación.

En este sentido, la Matriz de Riesgos del Producto no solo presentó un instrumento de gestión de la calidad del *software*, sino también una guía metodológica para las pruebas, adaptadas a las condiciones reales del proyecto y alineadas a la norma ISO 14971.

En el Anexo 12.4.28 se describen tres de los principales riesgos evaluados y la gestión de su trazabilidad.

6.3.4.2. Matriz de trazabilidad integral

La matriz de trazabilidad integral se elaboró para cumplir con una de las premisas establecidas por la norma IEC 62304, la cual exige mantener trazabilidad entre los requerimientos, el diseño, la implementación y las pruebas.

En la figura 21, se presenta la trazabilidad diseñada, donde se diferencian los componentes por color: celeste (Ingeniería de requerimientos), verde (Pruebas: unitarias, de integración o de sistema) y amarillo (Diseño de arquitectura y diseño detallado), representando las diferentes actividades del Modelo V.

Matriz de trazabilidad de integral - Versión 2.0									
Requerimientos				Pruebas				Componente de arquitectura	
ID-Versión	Nombre	Descripción	Tarjetas JIRA vinculadas	ID-Versión	Tipo	Descripción	Casos de Uso o Issues asociados	Diseño de arquitectura	Diseño detallado

Figura 21: Columnas de la Matriz de trazabilidad integral

Versión con pruebas unitarias:

Matriz de trazabilidad de integral - Versión 2.0									
Requerimientos				Pruebas				Componente de arquitectura	
ID-Versión	Nombre	Descripción	Tarjetas JIRA vinculadas	ID-Versión	Tipo	Descripción	Casos de Uso o Issues asociados	Diseño de arquitectura	Diseño detallado
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-65	PU77-000	Prueba Unitaria	GetInterrogateAsync() retorna un objeto InterrogateResponse válido cuando la secuencia de comandos y respuestas es correcta.	CU4-000	Módulo: **CommunicationSubsystem**	Clase: **DeviceApi**
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-66	PU78-000	Prueba Unitaria	GetInterrogateAsync() lanza WandNotFoundException cuando Open() retorna false.	CU4-000 - Curso Alternativo 1	Módulo: **CommunicationSubsystem**	Clase: **DeviceApi**
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-67	PU79-000	Prueba Unitaria	GetInterrogateAsync() lanza InvalidResponseException cuando la respuesta al comando de interrogación es null.	CU4-000 - Curso Alternativo 2	Módulo: **CommunicationSubsystem**	Clase: **DeviceApi**
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-68	PU80-000	Prueba Unitaria	GetInterrogateAsync() lanza InvalidResponseException cuando la respuesta al comando de interrogación tiene longitud inferior a la mínima esperada.	CU4-000 - Curso Alternativo 3	Módulo: **CommunicationSubsystem**	Clase: **DeviceApi**
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-69	PU81-000	Prueba Unitaria	GetInterrogateAsync() retorna null cuando el desempaquetado FP17 entrega null.	CU4-000 - Curso Alternativo 3	Módulo: **CommunicationSubsystem**	Clase: **DeviceApi**
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-70	PU82-000	Prueba Unitaria	GetInterrogateAsync() retorna null cuando el desempaquetado FP17 entrega una lista vacía.	CU4-000 - Curso Alternativo 3	Módulo: **CommunicationSubsystem**	Clase: **DeviceApi**
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-70	PU83-000	Prueba Unitaria	Verifica la correcta construccion del comando Interrogate.	CU4-000	Módulo: **CommunicationSubsystem**	Clase: **AlcpWithFP17**
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-70	PU84-000	Prueba Unitaria	Verifica el correcto encapsulamiento del protocolo RFISCL al comando ALCP	CU4-000	Módulo: **CommunicationSubsystem**	Clase: **AlcpWithFP17**
RF4-000	Obtencion de Datos	Facilitar un botón que permita al usuario descargar la información del dispositivo implantable IPG	HIS-60/HIS-105/HIS-78/HIS-70	PU85-000	Prueba Unitaria	Cubre la rama donde el método falla naturalmente por entrada inválida	CU4-000 - Curso Alternativo 2	Módulo: **CommunicationSubsystem**	Clase: **AlcpWithFP17**
RF4-000	Obtencion de Datos	Se debe proporcionar una sección dentro de la pantalla principal que mostrará la hora interna del IPG.	HIS-60/HIS-105/HIS-78/HIS-70	PU86-000	Prueba Unitaria	Verifica que el método GetInterrogateCommand retorna el byte correspondiente al comando GetBootloaderInfo	CU4-000	Módulo: **CommunicationSubsystem**	Clase: **AlcpScript**
RF4-000	Obtencion de Datos	Se debe proporcionar una sección dentro de la pantalla principal que mostrará la hora interna del IPG.	HIS-60/HIS-105/HIS-78/HIS-70	PU87-000	Prueba Unitaria	Verifica que si response es null o su longitud no es < 3 lanza InvalidResponseException	CU4-000	Módulo: **CommunicationSubsystem**	Clase: **AlcpScript**

Figura 22: Matriz de trazabilidad integral en su versión de pruebas unitarias

Lo relevante de esta matriz es que permite establecer una trazabilidad completa entre cada Requerimiento Funcional (RF), las tarjetas en Jira (HIS) asociadas a su construcción, el (CU) asociado, las pruebas (unitarias en esta versión) correspondiente (identificada mediante su ID-Prueba).

Las columnas Diseño de arquitectura y Diseño detallado muestran las unidades de diseño encargadas de implementar la funcionalidad asociada a cada requerimiento, de acuerdo con la descomposición del sistema establecida en la arquitectura de *software*.

A continuación, se presenta un ejemplo de mapeo y sincronización entre la Matriz de trazabilidad integral y la Matriz de ejecución y resultados:

Datos relevantes de la Matriz de trazabilidad integral:

Requerimiento funcional:

- RF4-000-Obtención de Datos

Tarjeta JIRA vinculada:

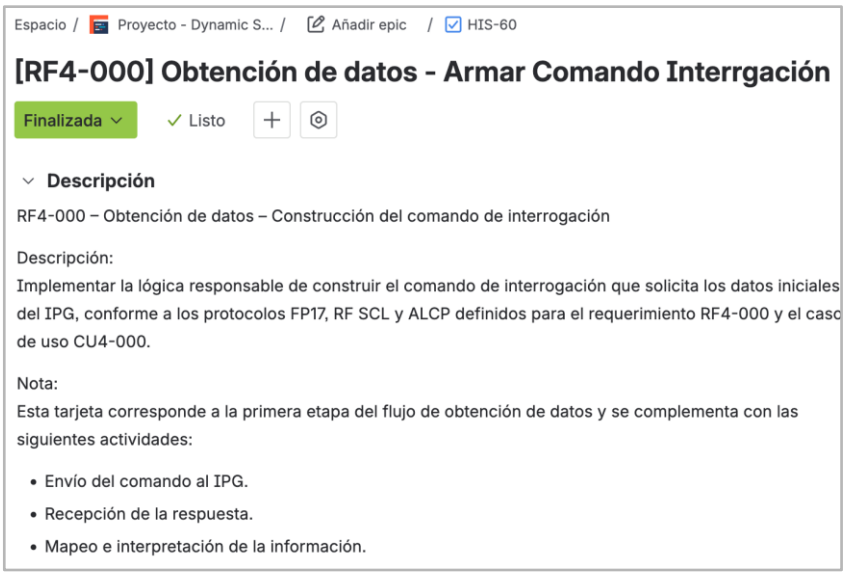


Figura 23: Tarjeta Jira – HIS-60-Obtención de datos - Armar comando interrogación

Caso de uso asociado:

- CU4-000

Diseño de Arquitectura:

- Módulo: *CommunicationSubsystem*

Diseño detallado:

- Clase: *AlcpScript*

Prueba unitaria:

- PU87-000

Hasta aquí quedaron mapeadas todas las actividades del Modelo V, si se necesita información más detallada de la prueba, se realiza una búsqueda en la Matriz de ejecución y resultados (para más detalles de este proceso ver Anexo 12.4.26) por el ID-Prueba.

Tras una búsqueda en la Matriz de ejecución y resultados se encuentra esta fila:

PU87-000	HIS-141	ParseInterrogateResponse_WhenDataIsNullOrTooShort_ThrowsInvalidResponseException	Unitaria	RF4-000	CU4-000	RProd-7-000	Lanzar InvalidResponseException con el mensaje "Payload too short for Interrogate response, < 3"	InvalidResponseException con el mensaje "Payload too short for Interrogate response, < 3."	Verifica que el método lance un InvalidResponseException cuando el parámetro response es null o su longitud es menor de 3 bytes.

Figura 24: PU87-000 trazada en Matriz de ejecución y resultados

Aquí se comparten los datos más relevantes que pueden encontrarse en la fila:

Prueba:

- PU87-000

Título:

- ParseInterrogateResponse_WhenDataIsNullOrTooShort_ThrowsInvalidResponseException

Riesgo vinculado:

RProd-7-000	Interpretación errónea de respuestas parciales.	Recepción de datos incompletos que pueden ser interpretados como válidos.	Si llegan mensajes parciales o incompletos y no hay un buen control en la lectura o los tiempos de espera son muy cortos, el sistema podría tomar datos incompletos como si fueran correctos.	3	2	SI	Implementar lógica de control, validación de mensajes y tiempos de espera adecuados, aun con un riesgo aceptable.	PU68-000/PU69-000, PU79-000, PU80-000, PU81-000, PU82-000, PU87-000	SI	Diseño detallado.

Figura 25: RProd-7-000 - Interpretación errónea de respuestas parciales

En la fila del riesgo RProd-7-000, columna 9 puede apreciarse como PU87-000 es una de las dos pruebas unitarias del conjunto que verifican la mitigación (columna 8) del riesgo.

Resultado esperado:

- El resultado esperado de la prueba.

Resultado obtenido:

- El resultado obtenido de la prueba

Observaciones:

- Las relevantes del caso probado.

Véase el Anexo 12.4.29 para visualizar las demás versiones de la misma Matriz de trazabilidad integral contemplando las pruebas de integración y sistema.

6.3.5. Ejecución del Plan

La ejecución del plan de pruebas se realizó tomando como base el Modelo V adaptado, se denominó de esta forma ya que las ejecuciones de las pruebas fueron en paralelo al desarrollo, lo que nos permitió mantener la trazabilidad entre requerimientos, diseño, implementación y prue-

bas y al mismo tiempo hacer los ajustes necesarios por la naturaleza del proyecto, las limitaciones en cuanto al acceso al *hardware* y los plazos académicos. Véase Anexo 12.4.25 para detalles de la ejecución del plan.

6.3.6. Resultados

6.3.6.1. Cobertura de pruebas unitarias

Como parte de los objetivos del proceso se estableció alcanzar un porcentaje de *code coverage* mayor o igual al 90%. Este objetivo fue superado con éxito, alcanzando un 96% de cobertura de bloques y un 94% de cobertura de líneas. Se destaca un esfuerzo grande por parte del equipo para alcanzarlo, haciendo pruebas exhaustivas y teniendo en cuenta la complejidad del producto que se construyó. Véase Anexo 12.4.30 para *code coverage.* En total se implementaron 429 pruebas unitarias, asegurando un buen porcentaje para la verificación funcional de todos los componentes. Véase Anexo 12.4.31 para pruebas unitarias.

6.3.6.2. Resultados de las pruebas de integración

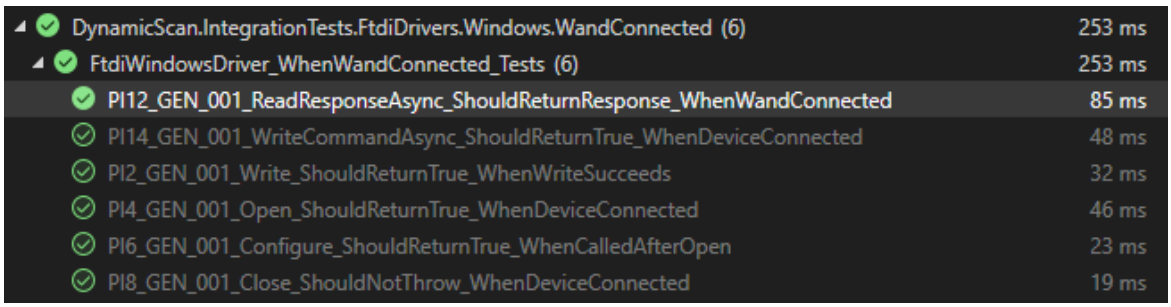


Figura 26: Pruebas integración con antena Wand conectada

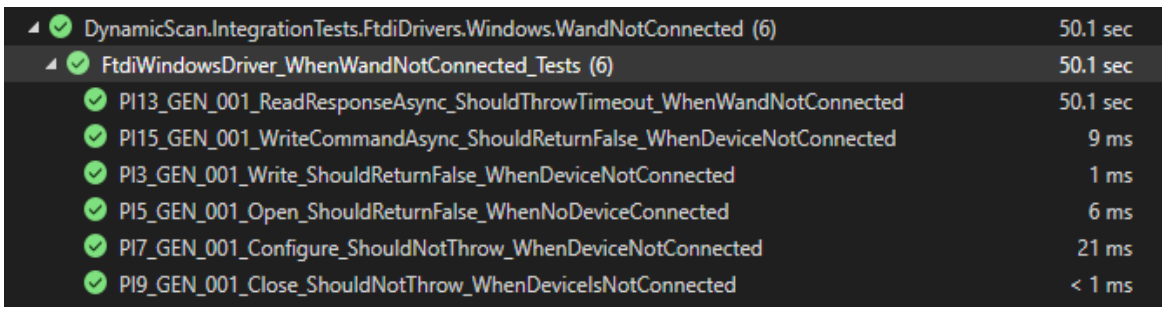


Figura 27: Pruebas de integración con antena Wand desconectada

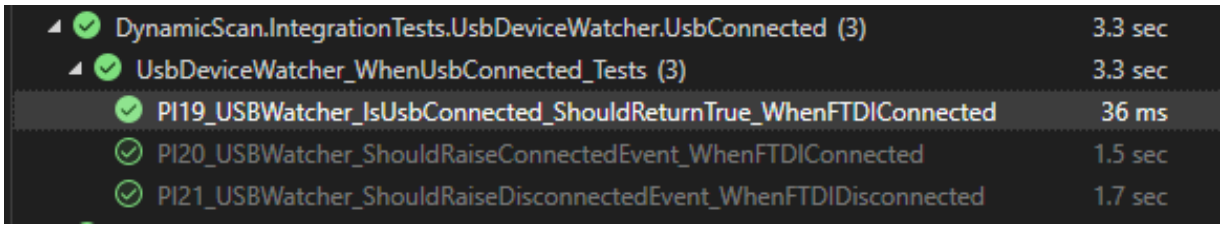


Figura 28: Pruebas de integración USB conectado.

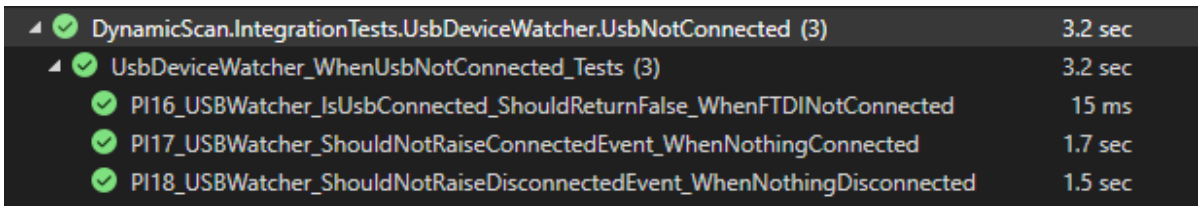


Figura 29: Pruebas de integración USB desconectado.

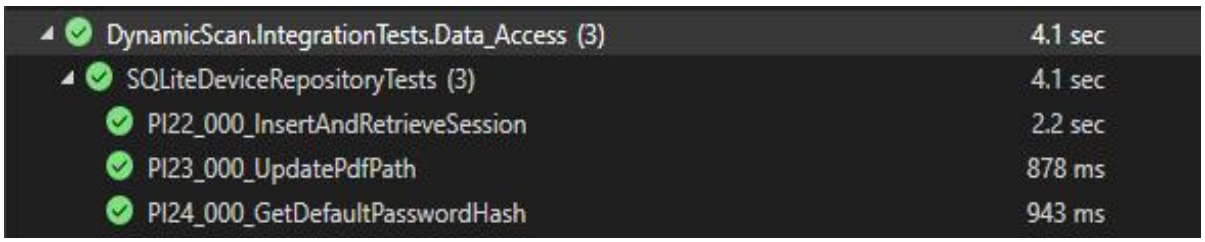


Figura 30: Pruebas de integración a base de datos SQLite

6.3.6.3. Resultados de las pruebas de sistema

Las pruebas de sistema (ver Anexo 12.7.1) tienen su registro en la Matriz de ejecución y resultados, la cual se puede visualizar en el Anexo 12.4.26. Además, se creó una tabla para corroborar la correctitud de los datos para los requerimientos de obtención, comparando nuestros valores con los del cliente. Ver Anexo 12.3.2, Control de correctitud de los datos tras interrogación.

6.4. Métricas

La recolección de métricas constituye un aspecto fundamental para garantizar la calidad, ya que permite contar con información objetiva para la toma de decisiones orientadas al cumplimiento de los objetivos planteados.

En este proyecto se definieron y monitorearon tanto métricas de proceso como métricas de producto, las cuales se detallan a continuación. Esta actividad fue una de las prácticas más importantes implementadas por el equipo para fortalecer el aseguramiento de la calidad.

6.4.1. De proceso

Se describen a continuación las métricas de calidad consideradas clave para el proceso.

6.4.1.1. Distribución del esfuerzo en calidad directa, investigación y otras actividades

La figura 32 muestra cómo se distribuyó el esfuerzo total del equipo entre actividades de calidad directa (pruebas, revisiones de código, retrabajo, trazabilidad y SQA), investigación (lectura de normativas y análisis de estándares regulatorios) y el resto de las actividades de desarrollo, gestión y soporte. De esta manera, puede observarse que una parte significativa del tiempo se destinó a la investigación, la cual fue clave para comprender el marco normativo y garantizar que la solución cumpliera con los requisitos de calidad y seguridad propios del dominio médico. Esto refuerza que la calidad se abordó desde el inicio del proyecto, realizando actividades preventivas tempranamente, teniendo a la calidad como un eje transversal del proyecto desde la fase de investigación y análisis hasta después de la liberación del Prototipo final.

Figura 31: Gráfica esfuerzo calidad directa, investigación y otras actividades

La etapa de investigación resultó un pilar fundamental en el desarrollo del proyecto, ya que permitió establecer las bases necesarias para orientar al equipo sobre cómo y por dónde comenzar. Aunque esta etapa no fue breve, e incluso podría percibirse como más extensa de lo inicialmente previsto, su duración se justificó por la complejidad del dominio y la necesidad de comprender a fondo el problema.

Para el equipo esta etapa de investigación fue clave, sin la misma no se podría haber entregado el producto final con la calidad que posee. Analizar cada documento propietario, normativas e introducir pruebas de concepto era indispensable para este tipo de *software,* el cual era completamente nuevo para el equipo. La misma fue el puente entre la ignorancia inicial del equipo sobre *software* médico y la capacidad de diseñar un prototipo realista y seguro. A su vez, se redujeron incertidumbres y riesgos, se construyó una base sólida, se capacitó en las normas aplicables y documentos propietarios, y se comprendió el contexto clínico y técnico. Gracias a este trabajo, el equipo adquirió una noción clara de los desafíos a los que se enfrentaba.

6.4.1.2. Retrabajo

Para este proyecto, el equipó se puso como objetivo no superar el 10% del total de horas trabajadas dedicadas al retrabajo de los problemas reportados y cambios solicitados por el cliente. Esta métrica se calculaba a partir de los registros de tiempo en Clockify, donde se utilizaba una etiqueta específica de "retrabajo" para identificar las horas destinadas a esta actividad. De esta manera se pudieron comparar las horas de desarrollo con las de retrabajo como se puede visualizar en la siguiente imagen.

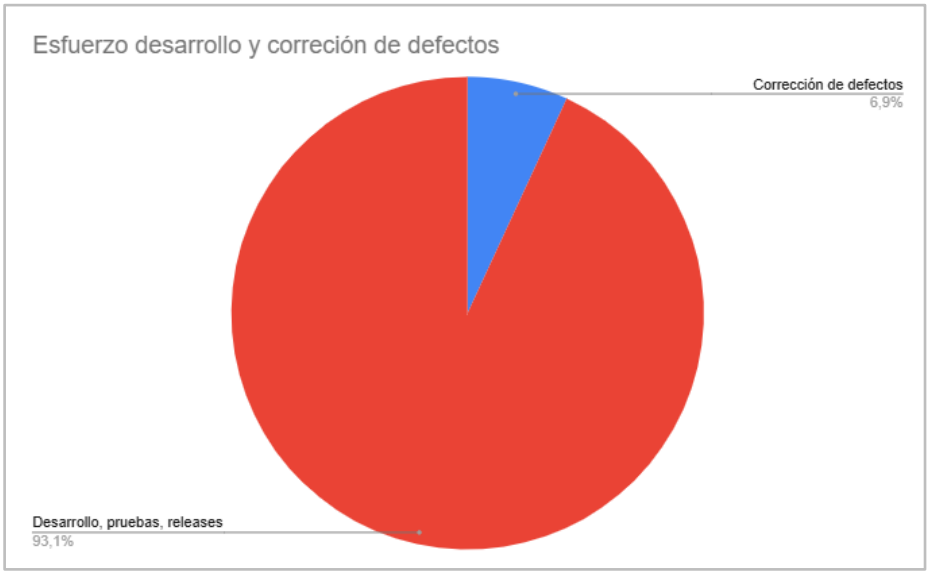


Figura 32: Esfuerzo en desarrollo y corrección de defectos

Tal como se puede ver en la figura 32, el esfuerzo destinado a la corrección de defectos (retrabajo) representó un 6,9% del total de horas trabajadas, cumpliendo con el objetivo de mantener el retrabajo por debajo del 10%. El 93,1% contempla únicamente las actividades de desarrollo, pruebas y *releases*, encontrándose aquí el *code review*, desarrollo *backend*, *frontend*, *release* 1 y *release* final.

Para el equipo los resultados alcanzados demuestran que se logró mantener el retrabajo dentro de los límites definidos. El 6,9% refleja que las actividades de revisión, verificación, validación y comunicación diaria con el cliente fueron efectivas para prevenir defectos y reducir la necesidad de correcciones posteriores, lo que aseguraba estabilidad en el desarrollo.

6.4.2. De producto

Se describen a continuación las métricas de calidad consideradas clave para el producto.

6.4.2.1. Cantidad de problemas reportados por *sprint*

Este indicador es clave para visualizar los problemas reportados en cada iteración y el impacto de las actividades de validación con el cliente.

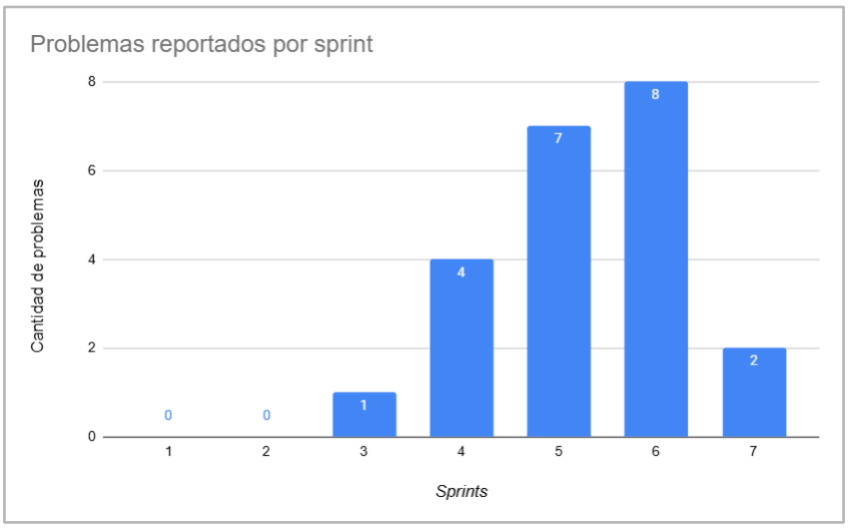


Figura 33: Gráfico problemas reportados por sprint

Tal como se muestra en la figura 33 la mayor cantidad de problemas reportados se concentró en los *sprints* finales en los cuales se tuvo validaciones junto al cliente y a su vez la salida de los *releases*. Como se puede visualizar, el *Sprint* 7 tuvo una reducción debido a que este se destinó principalmente a la estabilización del producto obtenido.

6.4.2.2. Cantidad de problemas acumulados por *sprint*

La figura 35 muestra la evolución acumulada de problemas reportados y resueltos a lo largo de los *sprints*. Se observa que ambas curvas mantienen un crecimiento muy similar, lo que indica que el equipo fue capaz de atender los problemas en tiempos cercanos a su detección. En el

Sprint 6 ambas líneas alcanzan el mismo nivel, reflejando que se resolvieron todos los problemas abiertos hasta ese momento. En total, se reportaron 22 problemas y se resolvieron 21, quedando como deuda técnica únicamente los *warnings* registrados en *Visual Studio.*

Figura 34: Gráfico problemas acumulados por sprint

El incremento a medida que avanzan los *sprints* se debe a que anteriormente se estaba en una etapa temprana del desarrollo en la cual el *software* estaba inmaduro y las validaciones eran limitadas. A partir de los *releases* y las pruebas conjuntas con el cliente, la detección de problemas aumentó significativamente, lo que llevó al equipo a dedicar un esfuerzo especial en resolver los pendientes.

El equipo logró cumplir con el objetivo de corregir los problemas encontrados antes de entregarle el producto al cliente, asegurando que este se mostrara conforme con la calidad alcanzada.

6.4.2.3. Cantidad de *bugs* y deudas técnicas resueltos

En esta sección se visualiza la cantidad por tipo de problemas resueltos durante el desarrollo. Los *bugs* corresponden a errores o problemas visibles que afectan el funcionamiento esperado del sistema. Por otro lado, la deuda técnica agrupa aquellos problemas que, si bien no generan fallos inmediatos de cara al usuario, representan aspectos de calidad interna que requieren atención para evitar problemas o inconsistencias a futuro.

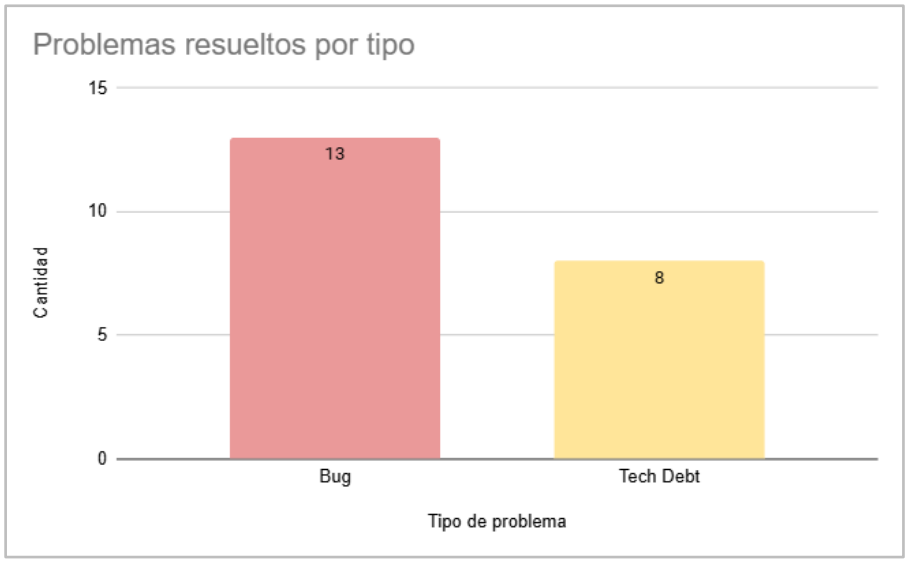


Figura 35: Gráfico problemas resueltos por tipo

En la figura 35, se observa la cantidad de los problemas resueltos durante el proyecto según su tipo. Se corrigieron 13 *bugs* y se abordaron 8 casos de deuda técnica. Los formatos correspondientes pueden visualizarse en los Anexos 12.4.32 y 12.4.33, respectivamente.

6.4.2.4. Cantidad de problemas detectados por criticidad

En esta sección se visualiza la cantidad de problemas encontrados por criticidad. De esta forma, se obtiene una visión más clara sobre qué cantidad de estos representaron más urgencia y cuántos pudieron ser gestionados con un margen menor de ésta.

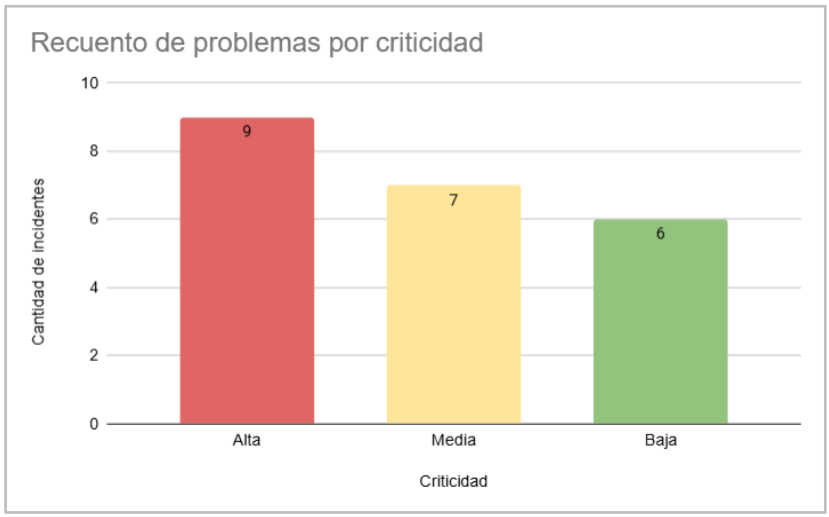


Figura 36: Gráfico recuento de problemas por criticidad

Tal como se observa en la figura 36, se detectaron un total de 22 problemas, de los cuales 9 fueron de criticidad alta, 7 de criticidad media y 6 de criticidad baja. Esto refleja que la mayor atención del equipo debió centrarse en la resolución de problemas críticos.

7. Gestión del Proyecto

Este capítulo describe las principales decisiones adoptadas a lo largo del proyecto, los desafíos que surgieron en el camino y la forma en que se organizaron las actividades del equipo. Se presentan los marcos de gestión aplicados en cada fase, la distribución de roles y responsabilidades, y la evolución temporal del trabajo reflejada en la línea de tiempo del proyecto. Asimismo, se incluye la descripción del ciclo de vida, la gestión de la comunicación y de los riesgos, junto con las métricas de esfuerzo y desarrollo que permitieron monitorear el progreso y evaluar el rendimiento del equipo.

7.1. Roles generales del proyecto

En esta sección se describen los roles generales definidos para el proyecto, asociados a los procesos de ingeniería del *software* y a la gestión técnica y organizativa establecida en el plan del proyecto. Estos roles proporcionaron la estructura base sobre la cual se distribuyeron las responsabilidades principales del equipo.

Gerente de proyecto

Guillermo Echichure asumió la responsabilidad de la planificación, seguimiento y control general del proyecto.

Ingeniero de requerimientos

Milena Dos Santos (principal) y Tomás Dilema (secundario), esta decisión se basó en la complejidad técnica del dominio y la necesidad de comprender los requerimientos del cliente, se estableció una responsabilidad principal en Milena (por su enfoque analítico y capacidad de documentación) y un rol complementario en Tomás, orientado al soporte técnico y a la validación de la factibilidad de implementación de los requerimientos en la arquitectura propuesta.

Arquitecto de *Software*

Tomás Dilema tuvo a su cargo el diseño de la arquitectura y la definición de las principales decisiones de diseño, incluyendo la selección de patrones y tácticas para satisfacer los atributos de calidad priorizados (portabilidad, integrabilidad y mantenibilidad).

Aseguramiento de la Calidad (SQA)

Julieta Sarantes lideró las actividades de aseguramiento y mientras que Guillermo Echichure las de Control de la calidad (pruebas), definidas en el Plan de *Software* Assurance, supervisando la aplicación de estándares de codificación, documentación y revisión.

Fue responsable de verificar la correcta ejecución de las actividades de revisión, verificación y validación en cada etapa del ciclo de vida, de acuerdo con los criterios definidos en el Plan de la calidad y las normas IEC 62304 e ISO 14971.

Además, colaboró estrechamente con el Gerente de proyecto en la evaluación de riesgos asociados a la calidad del producto y en la revisión de los resultados de las pruebas de integración y sistema.

Gestión de la Configuración (SCM)

Milena Dos Santos asumió la responsabilidad de la gestión de la configuración del *software*, asegurando la trazabilidad, integridad y consistencia de todos los artefactos bajo control de versiones.

7.2. Ciclo de vida del proyecto

El desarrollo del proyecto se estructuró siguiendo un modelo de ciclo de vida incremental e iterativo, adecuado para proyectos en los que se requiere validar de manera progresiva tanto la solución técnica como la documentación asociada.

Este enfoque permitió avanzar mediante iteraciones sucesivas, donde cada una incorporó incrementos funcionales y documentales que fueron evaluados, ajustados y validados antes de continuar con la siguiente etapa.

La elección de este modelo se fundamentó principalmente en dos razones:

Gestión de la dificultad técnica y regulatoria

Dado que el proyecto debía integrar un dispositivo médico implantable y cumplir con normas del ámbito sanitario (IEC 62304, ISO 14971 y guías FDA), el enfoque iterativo facilitó la incorporación gradual del conocimiento técnico y normativo, permitiendo ajustar la planificación a medida que se comprendían los requerimientos del cliente y las limitaciones del *hardware*.

Entrega temprana y retroalimentación continua

La naturaleza incremental del modelo hizo posible generar entregables parciales, como el *Release* 1 para poder tener una retroalimentación.

7.3. Fases del proyecto

El tiempo del proyecto se dividió en tres grandes fases: la primera fase tuvo por objetivo comprender el problema y realizar el diseño arquitectónico de la solución, la segunda fase tuvo por objetivo la construcción y prueba de la solución y la tercera, la realización de la documentación final del proyecto.

A continuación, en la figura 37, se presenta la línea de tiempo del proyecto, donde se puede observar cómo evolucionó el trabajo del equipo a lo largo de las distintas fases. En ella se reflejan las actividades más importantes y los hitos principales que marcaron el desarrollo del producto.

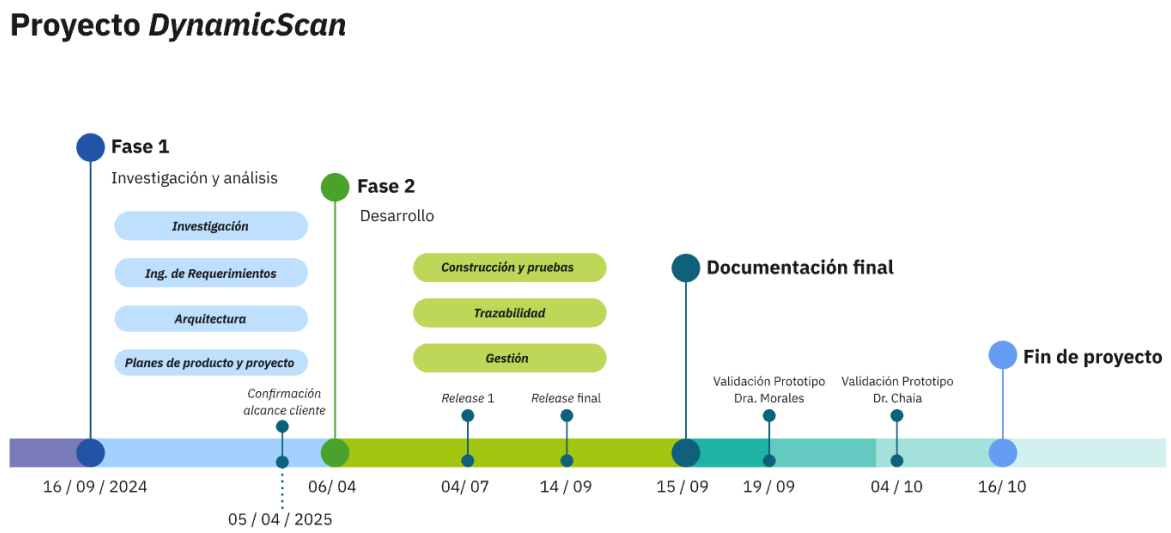


Figura 37: Línea del tiempo

La Fase 1 se extendió durante 26 semanas y 6 días y requirió un esfuerzo total de 638 horas.

La Fase 2 duró 23 semanas y demandó aproximadamente 1256 horas, alcanzando un total de 1894 horas de trabajo. En la sección 7.4 de Métricas de este capítulo se analiza con mayor detalle el esfuerzo.

7.3.1. Fase 1: Investigación y análisis

Esta primera fase tuvo como propósito comprender en profundidad el problema, el entorno tecnológico y las restricciones normativas que condicionaban el desarrollo del prototipo. Desde el punto de vista de la gestión del proyecto, implicó una etapa de alta dificultad, en la que el equipo debió invertir un esfuerzo significativo en investigación, documentación y validación de supuestos, antes de poder avanzar hacia decisiones de diseño o implementación.

Durante este período se priorizó el trabajo exploratorio, el análisis de factibilidad técnica y la recopilación de información sobre los protocolos y marcos regulatorios involucrados, estableciendo así las bases que permitirían reducir riesgos y planificar la siguiente fase con mayor precisión.

Además de investigación, esta fase supuso la consolidación de los procesos fundamentales de ingeniería del *software* que dieron forma al proyecto: Ingeniería de requerimientos, Diseño arquitectónico, Gestión de la calidad y Gestión del proyecto.

7.3.1.1. Ing. de requerimientos

Desde la Ingeniería de requerimientos (ver capítulo 3), se realizaron actividades de Elicitación, Especificación, y Validación de los requerimientos funcionales y no funcionales. Estas actividades permitieron construir la Especificación de requerimientos de *software* (ESRE) y sentar las bases de la Matriz de trazabilidad integral, que posteriormente se utilizó como eje para vincular requerimientos, casos de uso, pruebas y componentes de arquitectura, conforme a la norma IEC 62304.

7.3.1.2. Diseño arquitectónico

Al alcanzar una versión inicial estable del ESRE, que permitió fijar los requerimientos funcionales y no funcionales de referencia, se inició la actividad de Diseño arquitectónico. A partir de esta instancia se definieron las primeras decisiones estructurales del sistema, manteniendo un canal abierto de retroalimentación con la Ingeniería de requerimientos para su posterior refinamiento. Para más detalles de esta actividad ver capítulo 4 Arquitectura.

7.3.1.3. Gestión de la calidad

En este proceso se elaboraron los primeros planes de referencia, incluyendo el Plan de la calidad, el Plan de *software assurance*. Estos documentos definieron los estándares de codificación y documentación, los mecanismos de revisión, verificación y validación, la estructura de control de versiones y la trazabilidad de los artefactos bajo configuración. Finalmente, se elaboró el Plan de riesgos del producto, orientado a identificar, analizar y controlar los riesgos asociados al prototipo, conforme a los lineamientos de la norma ISO 14971, asegurando la trazabilidad entre los peligros identificados, las medidas de mitigación y las pruebas de verificación correspondientes.

7.3.1.4. Registro de la comunicación

Durante el desarrollo de esta fase se consideró esencial disponer de un mecanismo estructurado para registrar y organizar las comunicaciones mantenidas con los distintos actores: cliente, tutora, revisores y equipo de trabajo. Con este propósito, se definió un formato de minuta que permitió centralizar la información relevante de cada reunión.

El formato de minuta incluyó los siguientes campos:

- Título de la reunión
- Autor de la minuta
- Fecha y versión
- Participante

Este mecanismo resultó útil no solo para dejar constancia escrita, sino también para coordinar tareas, preparar próximas reuniones y reducir malentendidos entre los distintos participantes. Véase el Anexo 12.5.1 para ejemplo de minuta con tutora.

Para un mayor de talle de las herramientas utilizadas para la comunicación ver sección 8.2.7 Comunicación, la cual se encuentra dentro del capítulo de Gestión de la configuración.

7.3.1.5. Gestión de riesgos del proyecto

La gestión de riesgos del proyecto permitió identificar, analizar y controlar los factores que podrían afectar el cumplimiento de los objetivos planificados. A lo largo del desarrollo se evaluaron los riesgos más relevantes, se definieron acciones preventivas y se hizo un seguimiento continuo de su evolución, asegurando una respuesta temprana ante posibles desvíos.

Metodología

Marco utilizado:

- Se utilizará un enfoque basado en el PMBOK [19] y en la planilla brindada por la Universidad ORT.

Herramientas o técnicas:

- Análisis cualitativo y cuantitativo.
- Matrices de riesgo y probabilidad e impacto.

Identificación

Para identificar los riesgos del proyecto, el equipo llevó a cabo una sesión inicial de *brainstorming* la cual se puede visualizar en el Anexo 12.5.2 con los riesgos más votados, esta se utilizó para explorar posibles riesgos que podrían surgir durante su desarrollo. Los cuales se pueden visualizar con su descripción en el Anexo 12.5.3.

Así como también su forma de evaluación se puede visualizar en el Anexo 12.5.4., Criterio de evaluación de los riesgos.

Los riesgos del proyecto más votados fueron:

- R1 - Falta de experiencia con tecnologías del proyecto
- R2 - Falta de conocimiento en normas FDA: IEC 62304, ISO 14971, Guía de ciberseguridad
- R3 - Dependencia de *hardware* del cliente
- R4 - Falta de conocimiento de los protocolos propietarios
- R5 - Falta de conocimiento de los cuatro integrantes
- R6 - Error en estimación de tareas.

Monitoreo y control

Al final de cada *Sprint retrospective* se dedicó unos minutos a evaluar el progreso y la aparición de potenciales nuevos riesgos, y a decidir si acciones correctivas en caso de riesgos críticos (evitar, mitigar, transferir, aceptar).

Registro

En el Anexo 12.5.5 se muestran la Tabla de registro de los riesgos con sus valoraciones y su evolución en los diferentes *Sprints*.

Seguimiento

A continuación, se muestra el seguimiento de los riesgos, así como su evolución y se realiza un análisis de dos de los riesgos:

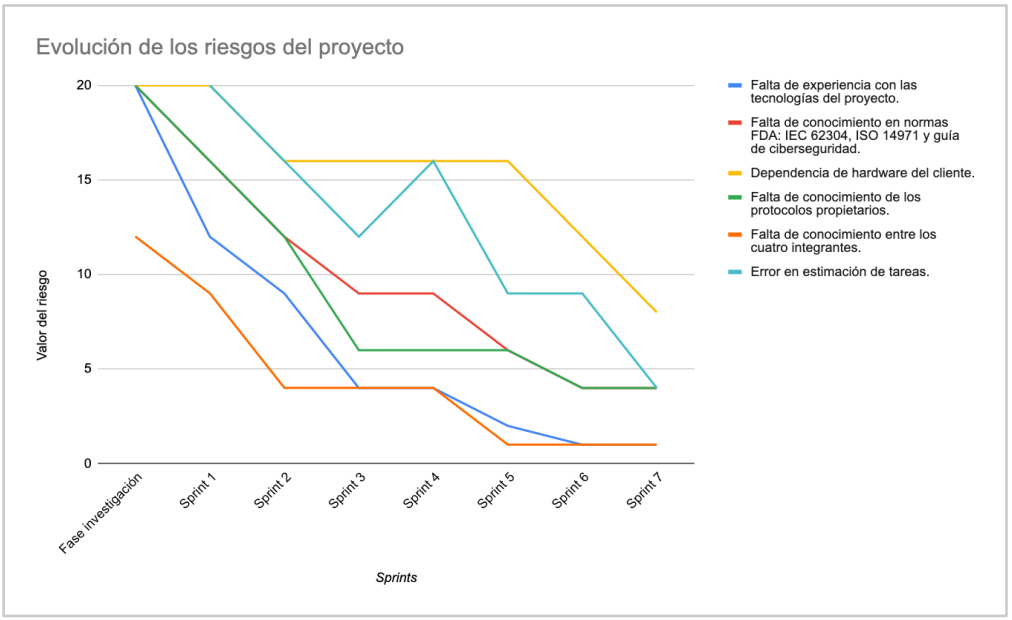


Figura 38: Evolución de los riesgos del proyecto

Finalizado el proyecto se puede visualizar en el Anexo 12.5.6 como finalizaron los riesgos según la tabla RAG.

A continuación, se describe la gestión de tres de los riesgos: R3, R4, R5 y R6:

Dependencia del *hardware* del cliente (R3)

La dependencia del *hardware* del cliente (la antena Wand y el dispositivo implantable IPG) fue un riesgo que se mantuvo durante todo el proyecto. La acción primaria de coordinación de acceso no permitió eliminarlo, debido a esto y con la necesidad de continuar el proyecto, se optó por una táctica de aceptación. Esta restricción afectó especialmente las pruebas de integración y sistema, así como los *merge* de código crítico que debían validarse directamente en el entorno del cliente. Aunque el equipo alcanzó su mayor madurez técnica en los *Sprints* 5 y 6, la disponibilidad restringida del *hardware* del cliente limitó el margen para seguir aumentando la velocidad.

Falta de conocimiento de los protocolos propietarios (R4)

Este riesgo comienza en lo más alto de la gráfica, debido a la dificultad de las tareas y del dominio. Si bien se realizó una etapa importante de estudio y pruebas de los protocolos previas al inicio de los *sprints*, es diferente a enfrentarse a la programación real de los mismos. Es evidente que con el paso de los *sprints* el equipo fue madurando los conceptos no solo teóricos sino a nivel de programación. Hasta el *sprint* 3 el riesgo baja, dado que para ese tiempo ya teníamos casi resuelto el establecimiento de la conexión con el IPG. Luego de eso, se visualiza una meseta, esta ocurre porque se comenzó a trabajar en otro comando importante el de interrogar para obtener los datos del IPG y si bien el equipo ya contaba con experiencia previa este comando supuso un nuevo desafío. Finalmente, si bien como ultimo objetivo del cliente, se tuvo la incorporación de comandos avanzados para la obtención de tendencias, el equipo ya contaba con la experiencia suficiente para afrontar el riesgo con mayor confianza y por esto el valor disminuye.

Falta de conocimiento entre los cuatro integrantes (R5)

La falta de conocimiento previo entre los integrantes se redujo a medida que avanzaron los *sprints*. Las reuniones de retrospectiva y la práctica conjunta fortalecieron la comunicación y la distribución de tareas, lo que permitió un mejor ritmo de trabajo y una mayor capacidad de entrega, especialmente visible en los *Sprints* 5 y 6.

Error en estimación de tareas (R6)

El riesgo vinculado a la estimación de tareas estuvo presente desde las primeras etapas del proyecto. Para el equipo fue todo un desafío estimar las tareas en un dominio tan complejo como

el *software* médico. En los primeros *sprints* se observaron diferencias importantes entre los puntos planificados y los realmente completados, lo que reflejó la dificultad inicial del equipo para dimensionar correctamente el esfuerzo requerido en cada historia. En el *Sprint* 4 donde se alcanzó el pico de la gráfica, se dio una situación contraria: se planificó más trabajo del que se logró completar, en parte porque uno de los integrantes redujo su participación durante tres semanas por la enfermedad de un familiar directo, lo que afectó la productividad general del equipo.

En los *Sprints* 5 y 6 el riesgo se mantuvo alto, aunque con una leve baja, ya que el equipo logró entregar más de lo estimado gracias a un esfuerzo adicional y a un mejor conocimiento de los protocolos y comandos del dispositivo. Sin embargo, las diferencias entre lo planificado y lo entregado continuaron, por lo que la reducción del riesgo esto se debió en parte por el esfuerzo adicional que realizó el equipo en afán de entregar funcionalidades y pruebas.

7.3.1.6. Marco de gestión

A nivel metodológico, y considerando la naturaleza exploratoria de las tareas iniciales, se adoptó un tablero Kanban como marco de gestión, presentado en el Anexo 12.5.7. Esta elección permitió visualizar el flujo de trabajo y priorizar actividades de manera dinámica, adecuándose a la imposibilidad de estimar esfuerzos con precisión en etapas donde predominaban las pruebas de factibilidad, la elaboración de documentos normativos, documento ESRE y los diferentes planes.

7.3.2. Fase 2: Desarrollo

7.3.2.1. Marco de gestión

Para esta segunda fase en donde se debía construir el producto de *software* a dos niveles: prototipo con arquitectura multiplataforma y un conjunto de documentos alineados con las normas FDA se decide utilizar con subconjunto de prácticas de *Scrum* adaptado.

Esta decisión se basó en lo siguiente:

A nivel normativo

La norma IEC 62304 y las guías de la FDA establecen que lo fundamental es asegurar la trazabilidad completa entre requerimientos, diseño, implementación, pruebas y riesgos. En este

marco, cualquier modelo de ciclo de vida puede aplicarse siempre que permita cumplir con dicho principio de trazabilidad. Esto significa que la adaptación que se realizó (usar prácticas de Scrum adaptado), mapeando todas las actividades al Modelo V mediante la Matriz de trazabilidad integral es válida porque cumple con la exigencia central: demostrar correspondencia entre requisitos y pruebas, independientemente de cómo se gestionan los tiempos.

A nivel metodológico

En primer lugar, el plazo estuvo restringido a menos de un año académico al considerar también la Fase 1. Si bien durante esta etapa inicial ya se habían comenzado a cumplir algunas actividades propias del Modelo V, el tiempo total disponible seguía siendo reducido. En este contexto, la incorporación de prácticas ágiles permitió generar un entregable intermedio (que incluyó documentación y funcionalidades iniciales del prototipo) y de esta manera obtener retroalimentación temprana, fundamental para orientar la evolución posterior del proyecto.

En segundo lugar, el equipo estuvo conformado por cuatro integrantes y contó con un único dispositivo de *hardware* (Wand + IPG), lo que exigió una gestión flexible del tiempo del desarrollo y las pruebas. Se trabajó en horarios extendidos y fines de semana, lo que hizo necesario organizar las tareas de manera dinámica para aprovechar cada instancia de uso del dispositivo.

En tercer lugar, cada integrante combinaba responsabilidades laborales y académicas adicionales, por lo que la gestión debía ajustarse a la disponibilidad variable del equipo y a la agenda del cliente.

7.3.2.2. Prácticas de SCRUM adaptadas

Asignación de roles básicos de Scrum:

- ***Scrum Master*:** Se designó a Milena en este rol debido a su experiencia previa en el uso de la herramienta en su ámbito laboral. La elección buscó mitigar riesgos metodológicos, priorizando la aplicación práctica de conocimientos ya adquiridos en lugar de incorporar incertidumbre adicional en un proyecto que de por sí presentaba múltiples desafíos técnicos y organizativos.

- ***Product Owner*:** La asignación del rol a Guillermo respondió a la necesidad de centralizar la coordinación entre las limitaciones del cliente, la disponibilidad del equipo y el

uso del *hardware* único, garantizando así que las prioridades de desarrollo se ajustarán a las restricciones reales del proyecto.

- El uso de un tablero visual para gestionar el *backlog*.
- Reuniones de revisión y retrospectiva unificadas en sesiones de retroalimentación.

Estas prácticas aportaron coordinación, priorización y control del avance, sin perder de vista la trazabilidad exigida la norma IEC 62304

Refinamiento del *Backlog*

De forma periódica, se revisaba y actualizaba el backlog con el objetivo de mantenerlo priorizado y claro. Esta práctica ayudó a dividir historias grandes en tareas más manejables, evitando bloqueos y mejorando la precisión en las estimaciones de los siguientes *sprints*.

Dailies adaptadas

En este caso, las *dailies* no podían llevarse a cabo todos los días debido a compromisos laborales y académicos. Por ello, se decidió realizarlas dos veces por semana. En estas reuniones, de aproximadamente quince minutos, el equipo compartía sus avances y comentaba que iba a abordar en los próximos días. Igualmente, la comunicación mantenida por WhatsApp era diaria, lo que hacía más fácil el mantenerse al día y saber en qué estado estaba cada integrante del equipo.

Sprint planning

Esta actividad se realizaba los domingos, día en el que los cuatro integrantes del equipo estábamos disponibles, sin obligaciones académicas ni laborales. Para estimar el esfuerzo de cada tarea se utilizaron *story points*, una unidad que permite medir la complejidad y el trabajo que implica una actividad, sin expresarlo directamente en horas. De esta forma, la estimación se centra en el nivel de dificultad y no en el tiempo exacto que llevará completarla.

Durante la estimación, se aplicó la técnica de *Planning Poker*, que facilitó la discusión y el consenso entre los miembros del equipo. Además, con base en las estimaciones y los resultados de los *sprints* anteriores, se ajustaba la cantidad de puntos disponibles para el siguiente *sprint*.

Sprint retrospective

La *Sprint retrospective* se realizaba al finalizar cada *sprint* los domingos, antes de la *Sprint planning* y tenía como objetivo analizar el desempeño del equipo, identificar aciertos y aspectos a mejorar. En estas reuniones se fomentaba un espacio de reflexión abierta, donde cada integrante podía expresar su punto de vista sobre la organización, la comunicación y los resultados obtenidos. La herramienta utilizada para dejar documentada cada reunión fue EasyRetro [28].

A partir de estas instancias, se definían acciones concretas de mejora que se implementaban en el siguiente *sprint*, lo que permitió optimizar la coordinación, ajustar la estimación del esfuerzo y fortalecer la dinámica de trabajo del grupo. Véase el Anexo 12.5.8 para ejemplo de reunión.

7.3.3. Fase 3: Documentación

Esta fase comprendió la confección de la documentación oficial de la Tesis, contemplando los estándares definidos por la Universidad ORT: documento 302, 303, 304 y 306.

7.4. Métricas

7.4.1. Registro del esfuerzo

Para registrar el esfuerzo el equipo utilizó la herramienta Clockify, la cual permite obtener reportes de tiempo dadas distintas categorías personalizadas por el equipo (ver Anexo 12.5.9 ejemplo de Clockify). Luego de finalizado cada *sprint* se mapean las horas con su categoría en un documento Google Sheets, lo cual hizo más sencilla la creación de gráficos. A continuación, se detalla el porcentaje de horas dedicadas desde el inicio hasta el fin del desarrollo por categoría, con un total de 1894 horas con 32 minutos.

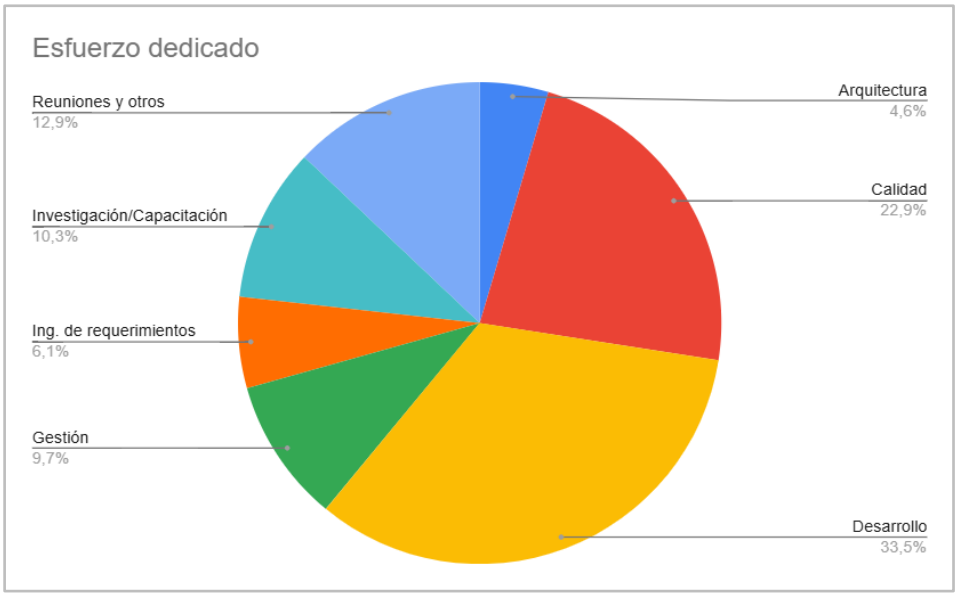


Figura 39: Gráfico de reporte de esfuerzo total

Como se puede apreciar en la figura 39, la etapa de investigación/capacitación representa un porcentaje significativo, junto con calidad donde se encuentran todas las pruebas realizadas por el equipo, los cual se encuentra explicado en el apartado de calidad.

Reuniones y otros hace referencia tanto a reuniones adaptadas de *Scrum* como lo fueron las *dailies, sprint planning y retrospectives*, reuniones con el tutor, reuniones con el equipo y revisiones estipuladas por la Universidad.

7.4.2. Métricas de desarrollo

Durante el desarrollo se completaron 401 *Story points* repartidos en 7 *sprints*, los cuales tuvieron un promedio de 57,3 SP por cada uno de ellos. La complejidad de las tareas fue bastante notoria por lo cual se contempló la escala de Fibonacci en el rango de 1 a 13.

A continuación, se pueden visualizar los *Story points* distribuidos a lo largo de los *sprints* del proyecto, donde se observa un avance sostenido y progresivo del equipo. Se optó por utilizar una *Burnup chart* en lugar de una *Burndown chart*, ya que este permite presentar mejor la evolución del trabajo completado en un contexto donde el alcance fue ajustándose a medida que se avanzaba con los *sprints* y no se contaba al inicio con una cantidad exacta de trabajo total fija.

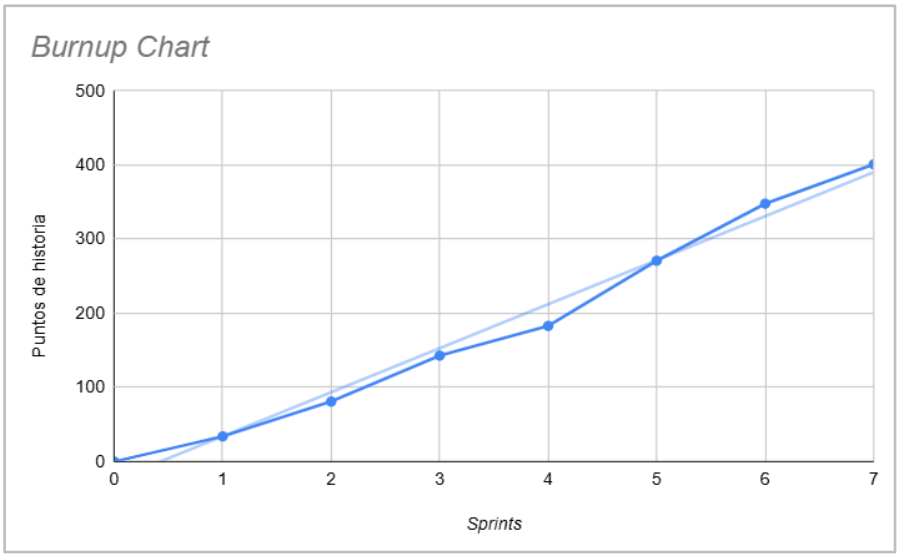


Figura 40: Burnup Chart

Lo primero que se define es el largo de los *sprints* que fueron de tres semanas, a excepción del *Sprint* 1 el cual duró 6 semanas debido a la complejidad de la historia Establecer conexión la cual era bloqueante e impedía el avance de los otros comandos y el *Sprint* 7 que duró 2, por motivo de tiempos y necesidad de arrancar a documentar. En los próximos *sprints* se dividieron los comandos en cuatro pasos. Esta decisión se basó en el hecho de que las historias de usuario relacionadas a la implementación del protocolo contenían un alto grado de dificultad técnica.

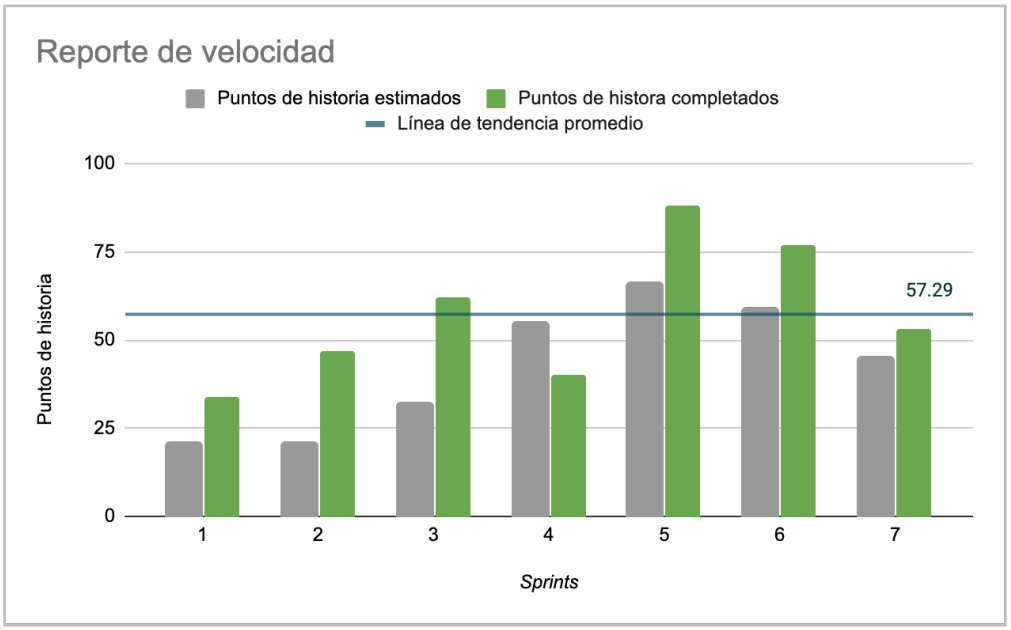


Figura 41: Gráfico reporte de velocidad

Como se puede visualizar en la figura 41, el *Sprint* 1 y 2 fueron interesantes para conocer la velocidad del equipo y saber a qué se enfrentaba.

El *Sprint* 3 implicó la preparación del *Release* 1 que se compuso del prototipo con lo priorizado hasta el momento y documentos relevantes para el cliente.

El *Sprint* 4 tuvo una diferencia apreciable entre lo estimado y lo realmente entregado. Esto se debió a la baja productividad de un integrante por un problema de salud de un familiar directo que duró tres semanas.

Durante los *Sprints* 5 y 6, el equipo alcanzó su mayor nivel de productividad, completando una cantidad de trabajo superior a la planificada inicialmente. Sin embargo, se evidenciaron desvíos en las estimaciones, ya que el esfuerzo real resultó mayor al previsto.

En cuanto al *Sprint* 7 el equipo se concentró en resolver problemas, estabilizar el producto para entregar al cliente y en la preparación de la última revisión.

A nivel general, el análisis de la velocidad y los resultados obtenidos reflejan una evolución sostenida del equipo a lo largo del proyecto. Aunque en algunos *sprints* las estimaciones no fueron del todo precisas, el equipo logró mantener el ritmo de trabajo y completar las tareas planificadas. Las primeras iteraciones estuvieron marcadas por la exploración técnica y la adaptación a un dominio desafiante en lo técnico y en lo documental.

8. Gestión de la configuración

El propósito de este capítulo es describir las estrategias utilizadas para la Gestión de configuración de *Software* (SCM) en DynamicScan, partiendo desde la identificación de los elementos de configuración, continuando con el control de cambios, versionado, documentación, herramientas de apoyo y comunicación.

Es un proceso fundamental dentro del ciclo de vida del proyecto de *software*, especialmente en sistemas críticos como DynamicScan, cuyo dominio está vinculado a dispositivos médicos y protocolos de comunicación especializados.

El objetivo principal de esto fue asegurar la integridad, trazabilidad y calidad de todos los elementos del proyecto, desde el código fuente hasta la documentación, garantizando que cada cambio realizado pueda ser identificado, evaluado y validado. Esto permitió minimizar riesgos, evitar pérdida de información, reducir el retrabajo y aumentar la productividad del equipo.

8.1. Identificación de elementos de configuración

Se consideran Elementos de configuración de *software* (ECS) todos aquellos componentes del proyecto que debían ser gestionados bajo control de versiones y trazabilidad.

Tipo	Elemento	Descripción
Software	Proyecto *DynamicScan*	Código fuente del *frontend* multiplataforma para Windows y Android implementado en .NET MAUI, responsable de la interfaz gráfica y lógica de presentación (patrón MVVM).
Software	Proyecto *DynamicScan.Core*	Código fuente del *backend* implementado en .NET 8, que contiene la lógica principal del sistema, las capas *BusinessLogic*, *CommunicationSubsystem, Models, Utils*.

Software	Proyecto *DynamicScan.IntegrationTests*	Proyecto de pruebas de integración implementado en xUnit
Software	Proyecto *DynamicScan.UnitTests*	Proyecto de pruebas unitarias implementado en xUnit
Software	Base de datos SQLite	Implementación local de la base de datos utilizada por DynamicScan para almacenar sesiones de conexión y credenciales cifradas del menú avanzado.
Infraestructura de configuración (control de versiones)	Repositorio Bitbucket privado de *Impulse Dynamics*.	Repositorio privado proporcionado por el cliente, utilizado como repositorio principal para el control de versiones del código fuente y pruebas del prototipo DynamicScan. Es el entorno oficial de gestión de configuración donde se almacenan las versiones de desarrollo, integración y entrega final.
Infraestructura documental (repositorio privado)	Repositorio Egnyte privado de *Impulse Dynamics*.	Plataforma en la nube para almacenar, compartir y colaborar en archivos de forma segura.
Documento	Prototipos de interfaz de usuario (UI)	Se utiliza la herramienta Figma.
Documento	Riesgos del proyecto	Documento elaborado a partir de sesiones de *brainstorming* y priorización de riesgos, utilizando la herramienta colaborativa Miro para la identificación, análisis y votación de estos.
Documento	Línea de tiempo	Representación visual de los principa-

		les hitos y fases del proyecto, elaborada con Miro para facilitar la planificación y el seguimiento temporal.
Documento	*Framing Protocol 17 (FP 17) Specification, RF Interface Script Language (RFI SCL), Optimizer SM System Application Level Communication Protocol* - ALCP	Documentación técnica confidencial. Familia de protocolos propietarios del cliente. Repositorio privado Egnyte proporcionado por el cliente.
Documento	Norma IEC 62304, Norma ISO 14971.	Documentación de normas confidenciales. Acceso exclusivo en instalación del cliente.
Documento	Guía de ciberseguridad	Documentación disponible en *internet*.
Documento	Plan de gestión de riesgos del proyecto.	Documento que identifica, analiza y evalúa los riesgos asociados a la planificación, ejecución y control del proyecto. Herramienta: Google Docs
Documento	Plan de gestión de riesgos del producto	Documento elaborado conforme a la norma ISO 14971, en el que se identifican, analizan y evalúan los riesgos asociados al *software* como producto médico. Herramienta*:* Google Docs*.*
Documento	Plan de *software assurance*	Documento que define las actividades, estándares y procedimientos aplicados para garantizar la calidad del *software* a lo largo del ciclo de vida del proyecto. Herramienta: Google Docs

Documento	Plan de la calidad	Documento que describe el enfoque general de calidad adoptado por el equipo, las responsabilidades asignadas, los criterios de aceptación de entregables y los métodos de revisión. Herramienta: Google Docs
Documento	Plan de pruebas de *software*	Documento que establece la estrategia, los tipos de prueba (unitarias, integración, sistema) para el prototipo DynamicScan. Herramienta: Google Docs.
Documento	Especificación de requerimientos de *software* (ESRE)	Documento que detalla los requerimientos funcionales y no funcionales del sistema, incluyendo la trazabilidad con casos de uso. Herramienta: Google Docs
Documento	Documento de arquitectura	Documento que presenta la estructura del sistema, las capas y componentes principales, los patrones arquitectónicos aplicados y las decisiones de diseño justificadas. Herramienta: Google Docs
Documento	Matriz de plan de la calidad	Hoja de cálculo asociada al Plan de la calidad, utilizada para registrar el seguimiento de actividades, entregables, revisiones y cumplimiento de criterios de aceptación. Herramienta*:* Microsoft Excel
Documento	Matriz de trazabilidad integral	Documento que mapea las actividades del Modelo V: Ing. de requerimientos,

		Pruebas de *software*, Diseño de arquitectura y Diseño detallado. Permite alinearse a la norma IEC 62304 mediante la trazabilidad completa de las actividades. Herramienta: *Google Sheet*
Documento	Matriz de ejecución y resultados	Registro de los casos de prueba ejecutados, sus resultados, observaciones, permite trazabilidad con los CU asociados, los Riesgos del producto vinculados y/o los problemas asociados en caso de tener. Herramienta: Google Sheets.
Documento	Matriz de riesgos del producto	Tabla elaborada según la norma ISO 14971, en la que se identifican, analizan y evalúan los riesgos asociados al *software* como producto. Incluye la trazabilidad con las actividades donde se evidencian, acciones de mitigación y pruebas de verificación o validación. Herramienta: Google Sheets
Documento	Backlog del producto	Registro de los RF, mantenidos a lo largo de los *sprints* del proyecto. Herramienta: Google Sheets
Documento	Gestión del esfuerzo	Planilla utilizada para registrar la carga horaria semanal por integrante y actividad, permitiendo monitorear la distribución del esfuerzo y la evolución del proyecto. Herramienta: Google Sheets

Documento	Minutas de reunión: cliente, tutoría y equipo de proyecto	Documentos que registran los acuerdos, decisiones y tareas surgidas en las reuniones periódicas con el cliente, el tutor y el equipo interno. Herramienta*:* Google Docs
Documento	Documentación final del proyecto	Contiene la documentación final del proyecto. Herramienta: Google Docs
Documento	Diagramas de arquitectura	Draw.io y Miro

En todos los casos, los ECS fueron sometidos a control de versiones, con políticas de versionado y respaldo definido desde el inicio del proyecto. Se puede apreciar en el Anexo 12.6.1 el código de respaldo y en el Anexo 12.6.2, carpetas de respaldo en Google Drive. Esto permitió asegurar que las distintas versiones del sistema fueran reproducibles y que se pudiera mantener un historial claro de la evolución de cada artefacto.

8.2. Herramientas utilizadas

Para gestionar los distintos ECS se seleccionaron las herramientas en base a criterios como los son: experiencia previa del equipo, facilidad de integración, disponibilidad, soporte para trabajo distribuido y por usabilidad previa del cliente.

8.2.1. Control de cambios

El control de cambios se implementó mediante un sistema de control de versiones distribuido, con el objetivo de garantizar la integridad, la trazabilidad y la correcta gestión de los artefactos bajo configuración.

Las herramientas utilizadas fueron las siguientes:

- **Git:** Sistema de control de versiones distribuido utilizado para gestionar la evolución del código fuente, permitiendo la identificación, registro y seguimiento de cada modificación.
- **Bitbucket:** Repositorio remoto privado, proporcionado por el cliente, empleado como entorno central de almacenamiento y colaboración del equipo de desarrollo.

- **Git Extensions y SourceTree:** Herramientas gráficas de apoyo utilizadas para la visualización y administración de *commits*, *merge* y ramas, facilitando la gestión visual del historial de cambios.

8.2.2. Estrategia de *branching* y versionado

Durante el desarrollo del proyecto se aplicó una versión adaptada de la estrategia GitFlow, con el objetivo de mantener un flujo de trabajo estructurado, trazable y alineado con las necesidades del equipo.

Esta estrategia permitió organizar el desarrollo en ramas principales y auxiliares, de acuerdo con la siguiente estructura:

- ***main***: rama destinada exclusivamente a la publicación de *releases* oficiales, en la que se etiquetaban las versiones estables mediante tags.
- ***develop***: rama de integración utilizada para consolidar las funcionalidades y correcciones completadas antes de su promoción a producción
- ***feature*/HIS-***: ramas creadas para la implementación de nuevas funcionalidades.
- ***bug-id*/HIS-***: ramas destinadas a la corrección de defectos detectados durante las pruebas o el uso del sistema.
- ***tech-debt-id*/HIS-***: ramas dedicadas a la resolución de deuda técnica o a la refactorización de componentes.
- ***hotfix*/HIS-***: ramas empleadas para la corrección inmediata de errores críticos identificados en versiones liberadas.

El trabajo diario del equipo se desarrolló principalmente en ramas (feature, bug-id, tech-debt-id o hotfix), las cuales se integraban posteriormente en *develop*, previo a un proceso cruzado de revisión de código (*Code Review*) que aseguraba la calidad y consistencia de las modificaciones antes de su fusión definitiva.

Una vez alcanzado un estado estable de integración, los cambios se promovían a *main*, donde se generaba la versión liberada correspondiente (*release*) mediante la aplicación de etiquetas de versión (*tags*).

La adaptación realizada al flujo *GitFlow* consistió en la convención de nombres utilizada para las ramas de corrección (*bug-id/**) y de deuda técnica (*tech-debt-id/**), esta decisión se basó en tener una mejor trazabilidad y clasificación de los problemas.

8.2.3. *Code Review*

Como parte de las prácticas de control de calidad del *software*, se implementó un proceso sistemático de revisión de código cruzada (*Cross Code Review)*, orientado a garantizar la corrección, mantenibilidad y consistencia del código fuente antes de su integración en la rama principal de desarrollo.

La revisión de código tuvo como objetivos principales:

- Detectar errores de manera temprana, reduciendo la probabilidad de defectos en etapas posteriores.
- Identificar oportunidades de mejora técnica, tales como refactorizaciones o eliminación de deuda técnica.
- Favorecer la transferencia de conocimiento entre los integrantes del equipo, promoviendo una comprensión compartida de la arquitectura y los estándares de codificación.

El proceso se llevó a cabo mediante la creación de solicitudes de integración (*pull requests*) en el repositorio Bitbucket.

Cada *pull request* implicaba que el código desarrollado fuese revisado por al menos un integrante distinto al autor, y en casos de funcionalidades críticas, por dos revisores.

Figura 42: Ejemplo de code review cruzado

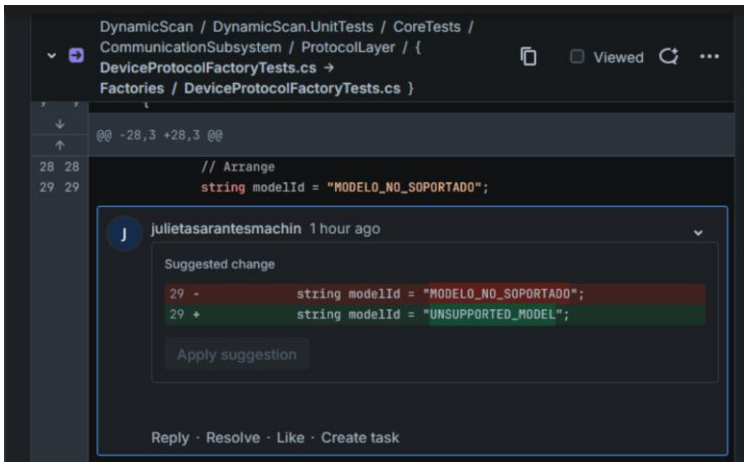


Figura 43: Ejemplo de sugerencia de cambio en revisión de código

La modalidad adoptada fue cruzada, de modo que ningún integrante revisara sus propios cambios, asegurando así una evaluación objetiva y colaborativa.

Una vez que las observaciones eran tratadas y los revisores aprobaban la solicitud, se procedía a realizar el *merge* hacia la rama *develop*, completando el flujo de integración definido en la estrategia GitFlow.

Una vez alcanzado un estado estable en la rama *develop*, se creaba una rama de *release* para consolidar la versión candidata. Tras su validación, los cambios se integraban en main, donde se aplicaba la etiqueta (*tag*) correspondiente a la versión liberada.

Figura 44: Ejemplo de registro de la fusión de ramas en el control de versiones

Este mecanismo permitió mantener la trazabilidad de las revisiones, documentar los comentarios técnicos y consolidar un proceso de verificación previo a cada incorporación de código, contribuyendo significativamente al aseguramiento de la calidad del *software*.

8.2.4. Documentación

Durante el proyecto se utilizó la plataforma Google Drive para la gestión y almacenamiento de la documentación generada, permitiendo el trabajo colaborativo y la disponibilidad permanente de los archivos.

La elección de esta herramienta se basó en su facilidad de uso, acceso remoto, capacidad de edición simultánea y en la experiencia previa del equipo con el entorno de Google.

Dentro del espacio de trabajo compartido se emplearon distintos tipos de documentos según su finalidad:

Documentos de Google: utilizados para elaborar y mantener los principales documentos del proyecto, como el Plan de calidad, el Plan de *software assurance* (SQA), el Plan de gestión de riesgos del proyecto, el Plan de gestión de riesgos del Producto, el Plan de pruebas de *software,* la Especificación de requerimientos de *software* (ESRE), el Documento de arquitectura, las Minutas de reunión y la Documentación final del Proyecto.

Estos archivos se desarrollaron de forma colaborativa, permitiendo registrar observaciones, revisiones y versiones intermedias.

Hojas de Cálculo de Google: utilizadas para la administración de la información estructurada del proyecto, tales como la Matriz de trazabilidad integral, la Matriz de ejecución y resultados de pruebas, la Matriz de riesgos del producto, el Backlog del producto, la Gestión del esfuerzo, así como las tablas de *Sprints* y Priorización de requerimientos.

Estas planillas facilitaron la trazabilidad entre actividades, pruebas y riesgos, contribuyendo al seguimiento de las tareas y a la organización del trabajo.

El uso de Google Drive como repositorio compartido permitió mantener la documentación organizada en carpetas por tipo de entregable y etapa del proyecto, garantizando su acceso controlado y la preservación del historial de versiones.

8.2.5. Diagramas y prototipos

Para la creación de diagramas se utilizó la herramienta Draw.io, seleccionada principalmente por la experiencia previa del equipo con su uso, su facilidad de adopción y su entorno intuitivo. Además, su integración con plataformas colaborativas como Google Drive permitió trabajar de forma simultánea entre los integrantes del equipo, lo que facilitó la revisión y actualización de los diagramas en tiempo real, mejorando la comunicación y la coherencia en la documentación del proyecto.

Para el desarrollo de prototipos de interfaz de usuario, se empleó Figma, una herramienta ampliamente utilizada en la industria del diseño de *software* que posibilita el trabajo colaborativo en línea, el versionado de diseños y la visualización interactiva de las pantallas. Su elección se fundamentó en la necesidad de contar con un entorno que permitiera iterar rápidamente sobre

los diseños, validar propuestas con el cliente y mantener un registro centralizado de los cambios realizados durante las etapas de diseño de la interfaz.

8.2.6. Gestor de tareas

Para la gestión de tareas se utilizó Jira, herramienta que permitió realizar el seguimiento de las actividades del proyecto, incluyendo tarjetas de trabajo, *bugs*, deudas técnicas y documentación.

Cada integrante del equipo registró y actualizó en Jira el estado de sus tareas, lo que facilitó mantener una visión centralizada del progreso y asegurar la trazabilidad entre los distintos componentes del sistema.

Esta herramienta resultó fundamental para coordinar el trabajo del equipo, detectar bloqueos a tiempo y documentar el historial de actividades realizadas durante todo el ciclo de desarrollo, así como también obtener métricas asociadas.

8.2.7. Comunicación

Durante el desarrollo del proyecto se utilizaron diversas herramientas para asegurar una comunicación fluida, organizada y efectiva entre los integrantes del equipo, la tutora y el cliente.

Se utilizó WhatsApp como canal principal de comunicación inmediata, permitiendo coordinar reuniones, resolver dudas puntuales y mantener el contacto cotidiano con la tutora, el cliente y el equipo.

Microsoft Teams fue utilizado para las reuniones internas del equipo, especialmente durante las reuniones de SCRUM adaptadas, así como para mantener un respaldo de información y materiales de trabajo compartidos, a su vez se utilizó como insumo para las reuniones con la tutora. En cambio, las reuniones y los intercambios formales con el cliente se realizaron a través de Google Meet, garantizando un entorno accesible para todos los participantes.

9. Conclusiones

En este capítulo se presentan las principales conclusiones del proyecto y un análisis respecto al cumplimiento de los objetivos definidos al inicio de este.

Al comienzo del proyecto, el equipo no contaba con una definición precisa del tema a desarrollar, pero sí con la convicción de que el trabajo final debía representar un aporte significativo a la sociedad. Se buscó aplicar los conocimientos adquiridos a lo largo de la carrera en un contexto real, con un propósito que trascendiera lo meramente académico. El desarrollo de este proyecto permitió materializar esa intención, constituyéndose en una experiencia profundamente enriquecedora tanto en el plano técnico como en el humano.

El trabajo implicó un doble desafío: por un lado, la complejidad técnica inherente al desarrollo de un *software* médico bajo criterios de calidad y seguridad exigentes; por otro, la rigurosidad en los documentos exigida por el tipo de producto y su entorno regulado. Este proceso favoreció un crecimiento notable del equipo en ambas dimensiones, consolidando competencias técnicas, metodológicas y profesionales que resultarán de alto valor en el ámbito laboral.

Desde una perspectiva académica, el proyecto permitió profundizar conocimientos en todas las etapas del ciclo de desarrollo del *software*. Se fortalecieron las capacidades de análisis y especificación durante la ingeniería de requerimientos, y se amplió la comprensión sobre el diseño arquitectónico al concebir una solución capaz de adaptarse a distintos sistemas operativos y entornos de ejecución, lo que exigió analizar patrones, dependencias y decisiones orientadas a la portabilidad y mantenibilidad. Asimismo, se profundizó en la verificación y validación del producto mediante pruebas unitarias, de integración y de sistema. Finalmente, el proceso de aseguramiento de la calidad (SQA) permitió comprender la relevancia de planificar, documentar y supervisar cada actividad de desarrollo con criterios de trazabilidad, consistencia y conformidad, procurando que los resultados se alinearan con los estándares requeridos por el cliente.

En el plano humano y profesional, el trabajo en equipo resultó una experiencia particularmente enriquecedora. La colaboración constante, el intercambio de conocimientos y la resolución conjunta de desafíos técnicos permitieron afianzar la confianza, la comunicación y la capacidad de adaptación del grupo. Más allá de los resultados obtenidos, el proyecto dejó un aprendizaje

compartido sobre la responsabilidad que implica desarrollar *software* destinado al ámbito médico y sobre el valor que la ingeniería de sistemas puede aportar a la sociedad cuando se orienta a mejorar la calidad de vida de las personas.

9.1. Objetivos académicos

A continuación, se presentan los objetivos académicos cumplidos a lo largo del proyecto.

Objetivo*:* Adquirir experiencia práctica en el desarrollo y documentación de *software* médico, comprendiendo las particularidades técnicas y regulatorias que implica este tipo de sistema.

Se considera que este objetivo fue **cumplido**, dado que el equipo logró incorporarse en un entorno de desarrollo real que demandó el estudio y la aplicación de normativas específicas del ámbito médico, tales como los lineamientos de la FDA, la norma IEC 62304, la norma ISO 14971 y la Guía de ciberseguridad. A lo largo del proyecto se adquirió una comprensión profunda de los procesos de gestión de riesgo, validación y trazabilidad, así como de los desafíos técnicos asociados a la comunicación con dispositivos implantables y la documentación requerida para *software* médico. Esta experiencia permitió consolidar una visión integral sobre el ciclo de vida y las exigencias de calidad y seguridad propias de este tipo de sistemas.

Objetivo*:* Aplicar los conocimientos y habilidades adquiridos a lo largo de la formación académica en un entorno profesional y de desarrollo real.

Este objetivo también se considera **cumplido** satisfactoriamente, ya que el proyecto permitió poner en práctica conocimientos adquiridos en diversas áreas de la carrera, como arquitectura de *software*, programación en .NET, diseño de interfaces, bases de datos, ingeniería de requerimientos, control de versiones, pruebas de *software*, entre otras. La experiencia de trabajar en un proyecto real, con interacción directa con un cliente y un entorno regulado, permitió validar la formación académica recibida, reforzando la capacidad del equipo para integrar teoría y práctica en la resolución de problemas complejos.

9.2. Objetivos de calidad

A continuación, se presentan los objetivos cumplidos a lo largo del proyecto.

9.2.1. A nivel de producto

Cobertura de pruebas: Se estableció como meta alcanzar un porcentaje mayor o igual a 90 porciento de cobertura en pruebas unitarias.

Resultado: **Cumplido**, se logró un 96% de cobertura de bloques y un 94% de cobertura de líneas. En la sección 6.3.6, se encuentra información sobre la cobertura de pruebas.

Gestión de riesgos del producto: Se asegura que el 100% de los riesgos identificados en el producto finalicen con un nivel residual aceptable, según los criterios de la normativa ISO 14971, al culminar el proyecto.

Resultado: **Cumplido**, tras la finalización del proyecto el 100% de los riesgos identificados del producto culminaron con un nivel de riesgo residual aceptable. Véase el Anexo 12.4.1 donde se encuentra la Matriz de riesgos del producto.

9.2.2. A nivel de proceso

Alineación a normas: Se establecieron un subconjunto de requisitos derivados del marco normativo, previamente acordados con el cliente, para el cumplimiento de este objetivo.

Resultado: **Cumplido**:

- Se mapearon las actividades del Modelo V con las etapas del proyecto, y se registraron en la Matriz de trazabilidad integral. Esta se encuentra en la sección 6.3.4.2.
- Se gestionaron los riesgos del producto (ver sección 6.2.3) por separado de los riesgos del proyecto (ver sección 7.3.1.5).
- Se creó un Plan de riesgos del producto. Ver sección sección 6.2.3.
- Se ejecutaron pruebas de integración para módulos que no podían probarse unitariamente y para los identificados con algún riesgo asociado. Véase en la sección 6.3.6.2 para resultados.
- Se realizaron pruebas de sistema para los RF identificados en el ESRE (ver Anexo 12.7.1).

Control de retrabajo: Se logró que el retrabajo asociado a corrección de defectos se mantenga por debajo del 10% del esfuerzo total del equipo.

Resultado: Cumplido por registro del retrabajo. Ver sección 6.4.1.2, Métricas.

10. Lecciones aprendidas

A lo largo del proyecto se obtienen aprendizajes significativos que impactan directamente en la calidad del producto final y en la madurez del equipo. Las principales lecciones identificadas se detallan a continuación.

Colaboración continua con el cliente.

El involucramiento constante del cliente resulta un factor determinante para el éxito del proyecto. Se comprobó que mantener reuniones periódicas y revisar los avances mediante estas permite reducir ambigüedades y alinear expectativas. No obstante, es importante considerar que el cliente también cuenta con sus propias responsabilidades y entregas, lo que en ciertos momentos dificultó la coordinación y la frecuencia de la comunicación. Aun así, este enfoque participativo contribuyó a construir un entendimiento compartido del problema y a fortalecer la toma de decisiones conjuntas.

Gestión del alcance y priorización.

La gestión del alcance se llevó a cabo durante la etapa de Ingeniería de requerimientos, instancia en la cual se definieron, ajustaron y priorizaron las funcionalidades del sistema según su impacto clínico y técnico para el cliente. Este proceso permitió mantener un enfoque claro y ordenado, optimizando los tiempos y concentrando los esfuerzos en los módulos esenciales. Una vez finalizada esta etapa, el alcance del proyecto se mantuvo estable, garantizando la coherencia y trazabilidad de lo definido. La lección aprendida radica en la importancia de una adecuada gestión del alcance y priorización, acompañada de una comunicación continua con el cliente que permita validar cada decisión y asegurar que el producto final responda efectivamente a las necesidades reales.

Integración del enfoque normativo con la práctica de desarrollo.

La aplicación de normas como IEC 62304, ISO 14971 y la guía de ciberseguridad de la FDA se traduce en una comprensión más profunda del entorno regulado en el que opera el *software* médico. Se aprende que el cumplimiento normativo no debe verse como una instancia posterior, sino como un componente integrado desde las etapas tempranas de requerimientos. Este enfoque previene retrabajos y asegura la trazabilidad y seguridad del sistema.

Capacitación técnica y dominio del contexto clínico.

El equipo enfrenta el desafío de trabajar con tecnologías y protocolos propietarios, lo que requiere una fase de estudio y experimentación previa al desarrollo. La investigación de *drivers* y documentación técnica permiten comprender el comportamiento del sistema y definir requerimientos realistas. Este proceso evidencia la relevancia de combinar conocimiento técnico con comprensión del contexto clínico para lograr soluciones seguras y funcionales.

Valor del prototipado iterativo.

El uso de prototipos facilita la comunicación con los interesados y reduce el riesgo de malinterpretar requerimientos. Se comprueba que las demostraciones tempranas permiten validar flujos de interacción, obtener *feedback* inmediato y realizar ajustes antes de la implementación, ahorrando tiempo y esfuerzo en etapas posteriores.

Importancia de la trazabilidad.

Mantener una trazabilidad clara entre requerimientos, diseño, pruebas y resultados se reconoce como una práctica esencial en proyectos regulados. El uso del documento ESRE como insumo principal asegura el control de versiones y la coherencia de las especificaciones a lo largo del ciclo de vida del *software*.

La experiencia adquirida permitirá optimizar futuros proyectos en el ámbito del *software* médico, aplicando prácticas más efectivas de comunicación, trazabilidad y gestión del riesgo.

11. Referencias bibliográficas

[1] *Impulse Dynamics*, "*Impulse Dynamics* – Official Website," *Impulse Dynamics*, 2025. [En línea]. [Accedido: 28-sept-2025]. Disponible en: https://impulse-dynamics.com/.

[2] McMurray JJV, et al. "Mortality in heart failure patients: a contemporary overview." [En línea]. [Accedido: 15-Oct-2025]. [Disponible]: https://pmc.ncbi.nlm.nih.gov/articles/PMC5336901/

[3] *Impulse Dynamics*, "FDA Indications & Patient Selection," *Impulse Dynamics*, 2025. [En línea]. [Accedido: 28-sept-2025]. Disponible en: https://impulse-dynamics.com/fda-indications-patient-selection/.

[4] *Impulse Dynamics*, "Patient Education Materials," *Impulse Dynamics*, 2025. [En línea]. [Accedido: 29-sept-2025]. Disponible en: https://impulse-dynamics.com/patient-education-materials/.

[5] American Heart Association, "Classification of Functional Capacity and Objective Assessment (NYHA)," Professional Heart Daily, 2025. [En línea]. [Accedido: 15-oct-2025]. Disponible en: https://professional.heart.org/en/guidelines-and-statements/classification.

[6] CardioAlianza, "¿Qué es la terapia de modulación de la contractilidad cardíaca (CCM)?," *CardioAlianza*, 2025. [En línea]. [Accedido: 30-sept-2025]. Disponible en: https://cardioalianza.org/que-es-la-terapia-de-modulacion-de-la-contractilidad-cardiaca/.

[7] *Impulse Dynamics*, "*Impulse Dynamics* Surpasses 10,000 Patients Benefiting from CCM Therapy for Heart Failure," *Impulse Dynamics Global Newsroom*, 2025. [En línea]. [Accedido: 02-oct-2025]. Disponible en: https://impulse-dynamics.com/global/impulse-dynamics-surpasses-10000-patients-benefiting-from-ccm-t….

[8] U.S. Food and Drug Administration, "U.S. Food and Drug Administration – Official Website," *U.S. Department of Health and Human Services*. [En línea]. [Accedido: 28-sept-2025]. Disponible en: https://www.fda.gov/.

[9] International Organization for Standardization, *ISO 14971:2019 – Medical Devices — Application of Risk Management to Medical Devices*, Ginebra, Suiza: ISO, 2019. [En línea]. Disponible en: https://www.iso.org/standard/72704.html.

[10] International Electrotechnical Commission, *IEC 62304:2006+A1:2015 – Medical Device Software — Software Life Cycle Processes*, Ginebra, Suiza: IEC, 2015. [En línea]. Disponible en: https://webstore.iec.ch/en/publication/22794.

[11] U.S. Food and Drug Administration, "Cybersecurity in Medical Devices: Quality System Considerations and Content of Premarket Submissions," *FDA*, 2025. [En línea]. [Accedido: 09-oct-2025]. Disponible en: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/cybersecurity-medical-devi….

[12] Future Technology Devices International Ltd., "Driver Licence Terms – Details," *FTDI Chip*, 2025. [En línea]. [Accedido: 12-jun-2025]. Disponible en: https://ftdichip.com.

[13] Google, "Crazy 8's – Design Sprint Kit," Design Sprint Kit, 2025. [En línea]. [Accedido: 15-oct-2025]. Disponible en: https://designsprintkit.withgoogle.com/methodology/phase3-sketch/crazy-8s

[14] L. Bass, P. Clements y R. Kazman, *Software Architecture in Practice*, 4.ª ed. Boston, MA: Addison-Wesley, 2021.

[15] A. Shvets, "Design Patterns," *Refactoring.Guru*, 2025. [En línea]. [Accedido: 11-oct-2025] Disponible en: https://refactoring.guru/design-patterns.

[16] Microsoft, ".NET Multi-platform App UI (.NET MAUI)," .NET, 2025. [En línea]. [Accedido: 15-oct-2025]. Disponible en: https://dotnet.microsoft.com/en-us/apps/maui.

[17] SQLite, "SQLite Home Page," SQLite, 2025. [En línea] [Accedido: 15-oct-2025]. Disponible en: https://www.sqlite.org/.

[18] DapperLib, "Dapper – a simple object mapper for .NET," GitHub Pages, 2025. [En línea]. [Accedido: 15-oct-2025]. Disponible en: https://dapperlib.github.io/Dapper/.

[19] Project Management Institute, *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK)*, 7.ª ed., Newtown Square, PA: Project Management Institute, 2021.

[20] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*, 1st ed., Upper Saddle River, NJ: Pearson/Prentice Hall, 2002.

[21] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ: Prentice Hall, 2008.

[22] Microsoft Corporation, "Arrange your tests," Microsoft Learn — Unit Testing Best Practices in .NET, 2024. [En línea]. Disponible en: https://learn.microsoft.com/en-us/dot-net/core/testing/unit-testing-best-practices.

[23] Microsoft Corporation, "Pruebas de integración en ASP.NET Core," Microsoft Learn, 2024. [En línea]. Disponible en: https://learn.microsoft.com/es-es/aspnet/core/test/integration-tests?view=aspnetcore-8.0&pivots….

[24] IBM Corporation, "Defining use cases," IBM Documentation, 2025. [En línea]. Disponible en: https://www.ibm.com/docs/es/product-master/12.0.0?topic=processes-defining-use-cases.

[25] QuestPDF, "QuestPDF Documentation," QuestPDF, 2025. [En línea]. [Accedido: 15-oct-2025]. Disponible en: https://www.questpdf.com/.

[26] Microsoft Corporation, "MVVM en .NET MAUI," Microsoft Learn, 2025. [En línea]. Disponible en: https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm.

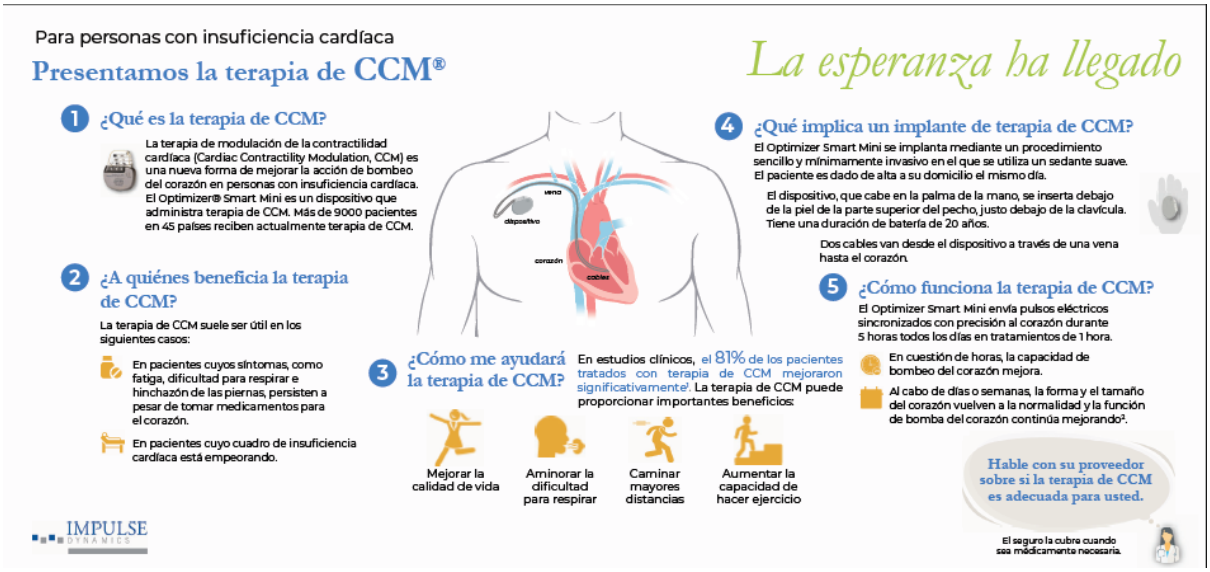
[27] Clockify, "Clockify: Seguimiento de tiempo y productividad," Clockify, 2025. [En línea]. Disponible en: https://clockify.me/es/.

[28] EasyRetro, "EasyRetro – Retrospective Tool," EasyRetro, 2025. [En línea]. Disponible en: https://easyretro.io/.

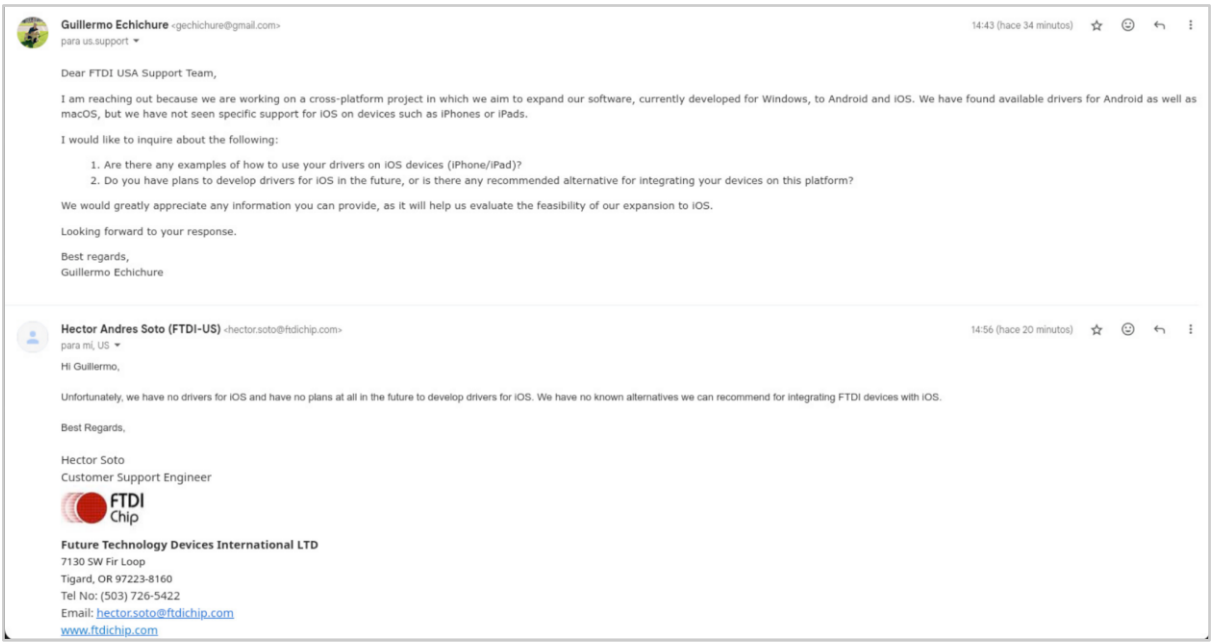
12. Anexos

12.1. Descripción del problema y la solución

12.1.1. Información de la Terapia CCM



12.1.2. Comunicación con FTDI Chips USA y Reino Unido



Guillermo Echichure <gechichure@gmail.com> 14:43 (hace 34 minutos)
para us.support

Dear FTDI USA Support Team,

I am reaching out because we are working on a cross-platform project in which we aim to expand our software, currently developed for Windows, to Android and iOS. We have found available drivers for Android as well as macOS, but we have not seen specific support for iOS on devices such as iPhones or iPads.

I would like to inquire about the following:

1. Are there any examples of how to use your drivers on iOS devices (iPhone/iPad)?
2. Do you have plans to develop drivers for iOS in the future, or is there any recommended alternative for integrating your devices on this platform?

We would greatly appreciate any information you can provide, as it will help us evaluate the feasibility of our expansion to iOS.

Looking forward to your response.

Best regards,
Guillermo Echichure

Hector Andres Soto (FTDI-US) <hector.soto@ftdichip.com> 14:56 (hace 20 minutos)
para mí, US

Hi Guillermo,

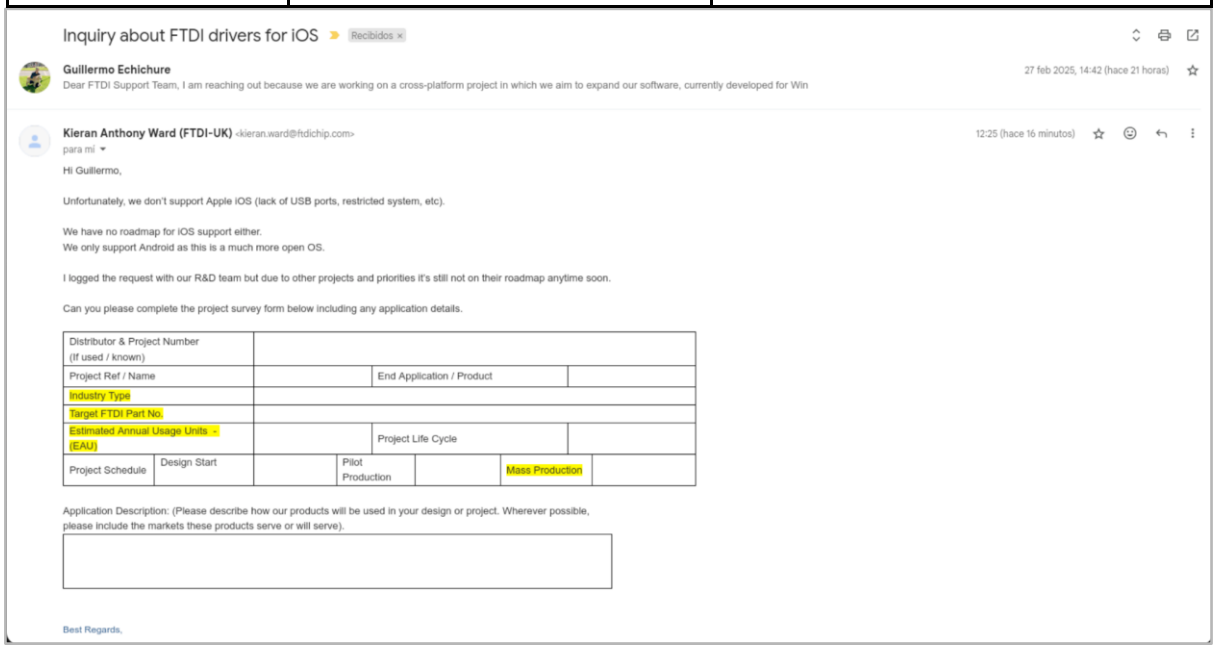
Unfortunately, we have no drivers for iOS and have no plans at all in the future to develop drivers for iOS. We have no known alternatives we can recommend for integrating FTDI devices with iOS.

Best Regards,

Hector Soto
Customer Support Engineer
FTDI Chip

Future Technology Devices International LTD
7130 SW Fir Loop
Tigard, OR 97223-8160
Tel No: (503) 726-5422
Email: hector.soto@ftdichip.com
www.ftdichip.com

	ESRE	
	Milena dos Santos - Tomás Dilema	
	14/09/2025	Versión 2.0



Inquiry about FTDI drivers for iOS

Guillermo Echichure — 27 feb 2025, 14:42 (hace 21 horas)
Dear FTDI Support Team, I am reaching out because we are working on a cross-platform project in which we aim to expand our software, currently developed for Win

Kieran Anthony Ward (FTDI-UK) <kieran.ward@ftdichip.com> — 12:25 (hace 16 minutos)
para mí

Hi Guillermo,

Unfortunately, we don't support Apple iOS (lack of USB ports, restricted system, etc).

We have no roadmap for iOS support either.
We only support Android as this is a much more open OS.

I logged the request with our R&D team but due to other projects and priorities it's still not on their roadmap anytime soon.

Can you please complete the project survey form below including any application details.

Distributor & Project Number (If used / known)					
Project Ref / Name		End Application / Product			
Industry Type					
Target FTDI Part No.					
Estimated Annual Usage Units - (EAU)		Project Life Cycle			
Project Schedule	Design Start		Pilot Production		Mass Production

Application Description: (Please describe how our products will be used in your design or project. Wherever possible, please include the markets these products serve or will serve).

Best Regards,

12.2. Ingeniería de Requerimientos

12.2.1. Documento Especificación de requerimientos

Abstract

Este documento presenta la especificación de requerimientos del prototipo DynamicScan asociado con el *Release final*, derivada de la fase de relevamiento de requerimientos, en la cual se aplicaron diversas técnicas de recolección de información. Entre estas, las entrevistas con el cliente *Impulse Dynamics*, destacaron como la herramienta más efectiva, gracias a su destacada colaboración en el proceso de Ingeniería de Requerimientos.

Introducción

Este documento sintetiza los resultados de la fase de relevamiento de requerimientos. A través de distintas técnicas: análisis de documentación técnica, análisis de normas: IEC 62304, ISO 14971, Guía de Ciberseguridad, y especialmente mediante entrevistas con el cliente *Impulse Dynamics* y el experto en la materia, Ing. Guillermo Kosut, se logró recopilar la información necesaria para iniciar la actividad de ingeniería de requerimientos.

Propósito del ESRE

El propósito de esta especificación de requerimientos es servir como guía completa para el equipo de desarrollo, detallando las funcionalidades requeridas, las restricciones del sistema, los atributos de calidad y las interacciones con los usuarios. Esta especificación garantiza que el prototipo final cumpla con las expectativas del cliente.

Descripción general

El prototipo debe ser capaz de comunicarse con el dispositivo implantable (IPG) a través de la antena Wand por radiofrecuencia. Para lograrlo se deberá implementar el protocolo propietario del cliente.

Identificación de los actores

En este capítulo se incluyen tanto usuarios humanos como sistemas externos, dispositivos, sensores, o cualquier entidad que interactúa con el prototipo DynamicScan.

1. **Profesional de la salud**
    - **Descripción:** Incluye médicos cardiólogos y enfermeros que interactúan directa o indirectamente con el *Software* para consultar datos del dispositivo implantable, monitorear datos del paciente.
2. **Técnico de *Impulse Dynamics***
    - **Descripción:** Este perfil cuenta con todas las funcionalidades habilitadas para el médico/enfermero, y además dispone de acceso al menú avanzado, el cual requiere autenticación para garantizar la seguridad y trazabilidad de las operaciones realizadas.
3. **Antena Wand** *(Actor no humano)*

    - **Descripción:** Dispositivo físico que se conecta por USB al dispositivo donde se está ejecutando DynamicScan y sirve como puente de comunicación entre el

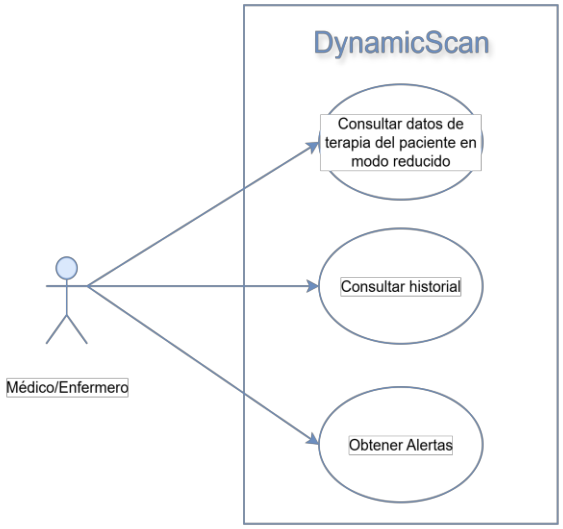
software desarrollado y el dispositivo implantable en el corazón del paciente. Esta última comunicación se da por radiofrecuencia.

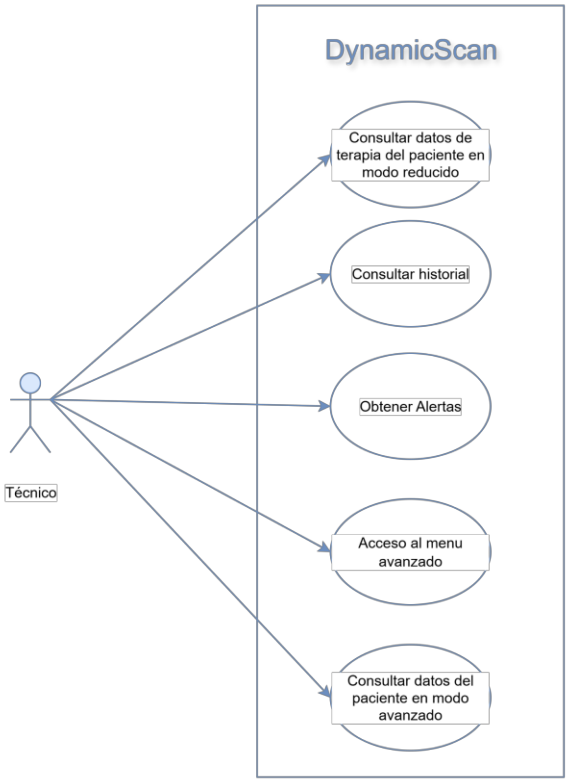
- **Interacción con los componentes del sistema:**
  - Recibe comandos del prototipo y los transmite al dispositivo implantable.
  - Transmite información del dispositivo implantable al prototipo para su procesamiento y visualización.

4. **Dispositivo Implantable *Smart Mini*** *(Actor no humano)*

- **Descripción:** Dispositivo médico implantable diseñado para el tratamiento de la insuficiencia cardíaca.
- **Interacción con los componentes del sistema:**
  - Interactúa con la antena Wand para enviar, a través de radiofrecuencia, datos de desempeño del dispositivo y de la terapia aplicada. En este proyecto, dichos datos serán generados o transmitidos en un entorno de laboratorio con fines de desarrollo y validación. La antena Wand se conecta al prototipo mediante una interfaz USB.

A continuación, se muestra un diagrama de alto nivel de los actores humanos





Restricciones generales

Aquí se detallan las restricciones que tendrá el prototipo:

- DynamicScan sólo podrá interrogar al dispositivo implantable (IPG). Bajo ningún concepto debe permitir configurar datos del mismo. (**R-Prod-001 - Riesgo de escritura no intencionada en uso previsto**) - Documento: Plan de Gestión de Riesgos del producto.

- Se debe asegurar la integridad de los datos.
    - No se podrá modificar ningún dato antes de mostrarle a los usuarios.
- El prototipo deberá estar diseñado con una arquitectura multiplataforma, que permita su despliegue tanto en Windows 10 como en Android.

Nivel de prioridad de los requerimientos

Se establecieron los siguientes niveles de prioridad:

Prioridad	Descripción
Alta	funcionalidades críticas para el funcionamiento del sistema, indispensables para cumplir los objetivos principales definidos por el cliente.
Media	funcionalidades importantes, pero no esenciales para la primera versión del prototipo; pueden ser implementadas en etapas posteriores sin comprometer el uso clínico inicial.
Baja	funcionalidades complementarias o de valor agregado, cuya implementación se considera deseable pero no prioritaria dentro del alcance del presente desarrollo.

12.2.1.1. Especificación de requerimientos

Los requerimientos se trabajarán a dos niveles: el *software* a desarrollar y la documentación asociada para alinearse con el marco regulatorio de la FDA.

Requerimientos funcionales de *software*

RF1-000 - Establecer la conexión

Prioridad: Alta

Descripción:

El sistema deberá permitir al usuario establecer una conexión con el dispositivo implantable IPG a través de la antena Wand.

El proceso de conexión incluirá: la obtención de información básica del dispositivo, la verificación del estado de operación y la compatibilidad del protocolo de comunicación.

También deberá mostrar el estado de la conexión, si está establecida o no aún en la misma pantalla. El componente debe estar diferenciado en colores para la sesión establecida y la no establecida.

Datos a obtener:

- Start Session Time
- Model
- Serial

RF2-000 - Cerrar la conexión

Prioridad: Alta

Descripción:

El sistema deberá permitir al usuario cerrar la conexión con el IPG de manera controlada, asegurando que los recursos sean liberados adecuadamente.

RF4-000 - Obtención de datos

Prioridad: Alta

Descripción:

Facilitar un botón "*Interrogate"* que le permita al usuario obtener la información del dispositivo implantable IPG. A su vez, se deberán guardar los datos en un archivo PDF generado por DynamicScan cuyo nombre será único, con el siguiente formato: "<serial_fecha_hora>.pdf"
Esta funcionalidad estará disponible una vez que la conexión con el IPG haya sido establecida.
Para la correctitud de los datos, se utilizó el documento confidencial **OSM_459 Rev 08 - Optimizer SM Programmer Application.**

Datos a obtener:

CCM Therapy:

- Mode
- CCM Therapy Mode
- CCM hs/day
- Start Time (HH:MM)
- End Time (HH:MM)
- On Time (HH:MM)
- Off Time (HH:MM)
- Extend on Low CCM%
- Magnet Mode

Sensing:

- Atrial Sensivity
- Atrial Sensing Polarity
- Ventricle 1 Sensivity
- Ventricle 1 Sensivity Polarity
- Ventricle 2 Sensivity
- Ventricle 2 Sensivity Polarity

Others:

- Atrial Refractory
- Ventricular Refactory
- Short AV
- Long AV
- Rate Averaging
- LS Alert Start
- LS Alert Width
- Pre Atrial LS Refractory
- Post Atrial LS Refractory
- Pre Ventricular LS Refractory
- Post Ventricular LS Refractory
- Post LS Refractory
- Number of Pulses
- CCM Train Delay
- Balancing Phase Duration
- Compare Phases Current

- Compare Phases Enabled
- Interval
- Phase Duration
- Alert Delivery Mode
- Patient Alert Start Time (HH:MM)
- Patient Alert End Time (HH:MM)
- Battery Recharge Reminder Days
- CCM Safe Mode
- CCM Not Sensing/Noise
- CCM Suspended
- Charger Battery Low
- ChargerFailure
- Battery Low

RF5-000 - Visualización de alertas

Prioridad: Media

Descripción:

El prototipo deberá permitir al usuario visualizar las alertas generadas por el IPG después de completar la obtención de datos de la interrogación (RF4-000). Se debe crear un botón "*Alerts*" el cual despliega un indicador a elección del equipo, de que existen nuevas alertas, así como su cantidad.
Esta funcionalidad estará disponible una vez que la conexión con el IPG haya sido establecida.

Datos a mostrar:

- CCM Safe Mode
- CCM Not Sensing/Noise
- CCM Suspended
- Charger Battery Low
- ChargerFailure
- Battery Low

RF7-000 - Visualización de detalles del IPG

Prioridad: Alta

Descripción:

Una vez resuelta la interrogación, se mostrará en la interfaz de usuario el modelo, número de serie, la fecha y hora de inicio de sesión y la fecha y hora del IPG (RF14-000).

Se deberá controlar si existe una diferencia mayor a 1h entre la hora interna del IPG y la hora en que se estableció la sesión, en caso afirmativo, se debe mostrar una advertencia visual en la pantalla para este caso.

Esta funcionalidad estará disponible una vez que la conexión con el IPG haya sido establecida.

Datos a mostrar:

- Start Session Time RF1-000
- IPG Time RF14-000
- Model RF1-000
- Serial RF1-000

RF8-000 - Visualización de Informe

Prioridad: Alta

Descripción:

El sistema deberá permitir visualizar el informe generado en la sesión actual con los datos obtenidos del IPG tras haber sido interrogado (RF4-000). Se debe crear un botón "*View current interrogate report*" el cual despliega el PDF en el visor de PDF que esté configurado por defecto.
Esta funcionalidad estará disponible una vez que la conexión con el IPG haya sido establecida.
Para la correctitud y cálculo de los datos, se utilizó el documento confidencial OSM_459 Rev 08 - Optimizer SM Programmer Application.

RF9-000 - Generación de Logs

Prioridad: Alta

Descripción:

El prototipo deberá generar un registro persistente de logs de ejecución desde la primera vez que se inicia la aplicación DynamicScan.

Dicho registro deberá conservar un historial completo de cada sesión de uso, incluyendo los eventos asociados a operaciones clave como establecer conexión, obtener el tiempo del IPG e interrogar.

Los logs deben:

- Almacenarse localmente en archivos de texto (.txt) dentro de una carpeta Logs ubicada en la raíz de la solución.
- Acumular la información histórica, sin sobrescritura entre ejecuciones. Cada nueva sesión debe anexar datos a un nuevo archivo o continuar en el último archivo si no se ha superado el límite de tamaño.
- Incluir marcas de tiempo, nivel de log (INFO, ERROR), nombre de la función invocada y datos relevantes para trazabilidad técnica.
- Ser independientes de cuántas veces se haya utilizado la aplicación; cada uso debe ser registrado y persistido.

RF10-000 - Visualización de *Splash Screen*

Prioridad: Baja

Descripción:

El prototipo deberá mostrar una pantalla de carga inicial (*Splash Screen*) con el logo de *Impulse Dynamics* al ejecutar la aplicación (DynamicScan).

El *Splash Screen* deberá ser visualizado por tres segundos.

RF11-000 - Acceso a sección avanzada

Prioridad: Media

Descripción:

Se proporcionará una sección avanzada que solo será accesible para los técnicos autorizados. La misma contendrá datos relevantes para estos. Para acceder a ella, el técnico deberá ingresar una contraseña que validará su autorización. En caso de ingreso incorrecto de la contraseña, se deberá impedir el acceso y registrar el intento fallido.

Por seguridad la contraseña no se ve al digitarse como medida de seguridad.

Tiene dos secciones:

- Summary RF12-000
- Trends RF13-000

RF12-000 - Visualización del resumen avanzado

Prioridad: Media

Descripción:

Dentro de la sección avanzada, se proporcionará una pestaña llamada "*Summary*" que permitirá a los técnicos visualizar un resumen con información clave obtenida del IPG. Esta sección mostrará valores esenciales como: impedancia y amplitud de onda permitiendo un análisis rápido del estado del dispositivo.

Para acceder a la pestaña "*Summary*", el usuario deberá haber ingresado previamente a la sección avanzada, lo que implica que debe haber completado el proceso de autenticación con contraseña.

Para la correctitud y cálculo de los datos, se utilizó el documento confidencial **OSM_459 Rev 08 - Optimizer SM Programmer Application.**

Datos a mostrar:

- Un *tab Summary* contenido los siguientes datos:
    - Impedance V1 (Ω)

- R-Wave Amplitude V1 (mV)
- Impedance V2 (Ω)
- R-Wave Amplitude V2 (mV)
- PWave Amplitude (mV)

RF13-000 - Trends

Prioridad: Media

Descripción:

Dentro de la sección avanzada, se proporcionará una pestaña denominada "*Trends*", la cual permitirá a los técnicos visualizar tendencias obtenidas del IPG.

Esta sección presentará datos relevantes sobre la actividad diaria del paciente, la batería y leads.

Para acceder a la pestaña "*Trends*", el usuario deberá haber ingresado previamente a la sección avanzada, lo que implica que debe haber completado el proceso de autenticación con contraseña.

Para la correctitud y cálculo de los datos, se utilizó el documento confidencial OSM_459 Rev 08 - Optimizer SM Programmer Application.

Datos a mostrar:

- Un segundo *tab Trends* con datos del paciente con las siguientes secciones:
  - *Hours Activity* - Gráficas de actividad del paciente por hora de las últimas 48 horas:
    - *Average* HR (bpm) - promedio frecuencia cardíaca real en latidos por minutos.
    - *Activity:* muestra el nivel de actividad del paciente con las siguientes categorías:*Very Light, Light, Moderate, Hard y Very Hard*.

        - *Position*: muestra las diferentes posiciones del paciente: *Laying, Reclined, Vertical, Active.*

        - *Average* HRV (ms) - promedio de la variabilidad de la frecuencia cardíaca en milisegundos.

    - *Battery*

        - Debe permitir ver el estado actual de la batería pudiendo visualizar el porcentaje y el voltaje.

        - Se debe mostrar el estado de la batería en porcentaje, (los iconos y formas a definir por el equipo de desarrollo) y datos acumulativos del voltaje de la batería en modalidad gráfica.

    - *Leads* - Debe permitir visualizar en gráficas el comportamiento eléctrico de los electrodos conectados al dispositivo implantable.
      Se deben mostrar dos conjuntos de gráficos correspondientes a los canales/cables **V1** y **V2**, donde se representan:

        - Impedancia (Ω): muestra la resistencia medida entre el electrodo y el tejido cardíaco, indicando el estado del contacto eléctrico. Un aumento significativo puede sugerir deterioro del lead o mala conexión.

        - Amplitud (mV): representa la señal eléctrica detectada por cada canal.

        - Cada gráfico presenta la evolución temporal de estos parámetros en función de la fecha, permitiendo al técnico observar tendencias y detectar posibles anomalías en el desempeño de los electrodos.

RF14-000 - Obtener la hora del IPG

Prioridad: Alta

Descripción:

Se debe obtener la hora interna del IPG.

Esta funcionalidad estará disponible una vez que la conexión con el IPG haya sido establecida y será visible en los Detalles del IPG (RF7-000).

RF16-000 - Visualización de historial de informes

Prioridad: Media

Descripción:

El prototipo deberá permitir al usuario acceder y visualizar el historial de informes generados en sesiones previas. Se deberá contar con un botón/icono "*History*" en la pantalla de establecer conexión. Este incluirá una sección histórica con los PDF generados (RF4-000 - Obtener datos) ordenados en forma cronológica.
En caso de no llegar a la instancia de generar el informe, de todas formas, se mostrará la información de las sesiones establecidas previamente.

Datos a mostrar de cada sesión:

- Un identificador único.
- Fecha y hora en que se establece la sesión.
- Fecha y hora interna del IPG.
- Modelo del IPG
- Serie del IPG
- PDF generado en caso de que se haya llegado a la instancia de obtención de datos.

RF17-000 – Detección de antena Wand

Prioridad: Alta

Descripción:

El prototipo deberá detectar automáticamente el estado de conexión de la antena Wand en la pantalla de establecer conexión. Cuando la antena esté conectada a DynamicScan, la interfaz de usuario deberá reflejar el estado "*Wand Connected*". Si la antena está desconectada o no está presente al iniciar la aplicación, la interfaz deberá mostrar "*Wand Disconnected*". Esta detección deberá ejecutarse en tiempo real y sin requerir intervención del usuario.

Requerimientos no funcionales de *software*

A continuación, se presenta la lista de los requerimientos no funcionales relevados con el cliente.

Requerimientos no funcionales obtenidos de la documentación requerida por la FDA

A continuación, se presenta un listado de los requerimientos no funcionales que se relevaron analizando en profundidad la documentación requerida por la FDA: IEC 62304, ISO 14971 y la guía de *ciberseguridad*.

Identificador único: RNF-FDA = Requerimiento no funcional - FDA

RNF-FDA-1-000 - Alineación con la Norma IEC 62304

Descripción: Se deben seguir los procesos, actividades y/o tareas para alinearse con la norma IEC 62304.

Para ello, esta norma no prescribe un modelo de ciclo de vida específico. Los usuarios de esta norma son responsables por seleccionar un modelo de ciclo de vida para el proyecto de *software* y por mapear los procesos, actividades y tareas de esta norma en ese modelo.

Mapear las actividades: diseño arquitectónico, diseño detallado, pruebas unitarias, pruebas de integración y pruebas de sistema, con las del ciclo de vida que se utilice para desarrollar el prototipo.

- **Atributo de calidad más significativo:** *Conformidad normativa*

RNF-SEG-1-000 – Autenticación de usuarios - ciberseguridad

- **Descripción**: El sistema debe requerir ingreso con contraseña antes de acceder a la sección avanzada, restringido a técnicos.

- **Atributo de calidad más significativo:** *Seguridad*

RNF-POR-1-000 – Portabilidad con arquitectura multiplataforma

- **Descripción:** El prototipo deberá estar diseñado sobre una arquitectura que favorezca la portabilidad, permitiendo su ejecución en Windows 10 y contemplando, a futuro, su despliegue en Android con modificaciones mínimas. La funcionalidad se deberá asegurar en Windows 10, mientras que en Android se validará el despliegue solamente.

- **Atributo de calidad:** Portabilidad

RNF-MOD-1-000 – Abierto a la extensión a múltiples tipos de dispositivos médicos implantables a futuro

- **Descripción:** El prototipo no debe limitarse a una única implementación de dispositivo (como el *SMART MINI*), su arquitectura debe permitir, en el futuro, la incorporación de nuevos tipos de dispositivos médicos implantables.

- **Atributo de calidad más significativo:** *Variabilidad*

RNF-TES-1-000 – Pruebas del prototipo

- **Descripción:** El prototipo deberá contar con pruebas a nivel: unitario, de integración y/o de sistema según sea el módulo o funcionalidad que se esté probando y la necesidad que se requiera cubrir.

- **Atributo de calidad más significativo:** *Testeabilidad*

RNF-INT-1-000 - Integración futura con nuevas tecnologías de comunicación

- **Descripción:** El prototipo debe ser abierto a la extensión con nuevas tecnologías de comunicación, como Wi-Fi o Bluetooth, sin impactar negativamente la arquitectura.

- **Atributo de calidad más significativo:** *Integrabilidad*

RNF-AVA-1-000 — Operación offline local

- **Descripción:** El sistema debe permanecer utilizable sin conexión de red ni dispositivo implantable conectado para funciones locales: debe permitir consultar el historial y obtener informes persistidos localmente, sin depender de servicios externos y ser agnóstico al SO.

- **Atributo de calidad más significativo:** *Disponibilidad*

RNF-AVA-2-000 — Gestión de indisponibilidad del canal

- **Descripción:** Para operaciones que requieren comunicación con el dispositivo implantable, se debe verificar la disponibilidad del canal antes de ejecutarlas; si no está disponible, se informará de inmediato y la operación no se iniciará. La UI no se debe bloquear, y cada operación de comunicación contará con un límite temporal y cancelable.
- **Atributo de calidad más significativo:** *Disponibilidad*

12.2.1.2. Casos de uso

La norma IEC 62304 establece la necesidad de documentar y mantener trazables los requerimientos del *software* médico, pero no prescribe un formato único para su representación. En consecuencia, el equipo optó por utilizar casos de uso como técnica principal de especificación, fundamentado en los siguientes aspectos:

Claridad para todas las partes interesadas

Los casos de uso describen escenarios de interacción entre el usuario y el sistema, lo cual facilita la comprensión tanto para perfiles técnicos (desarrolladores, testers) como no técnicos (stakeholders clínicos).

Soporte a la trazabilidad exigida por IEC 62304

Cada caso de uso se vincula de manera directa con requerimientos funcionales y con pruebas de verificación y validación. Esto permite establecer la trazabilidad requerimiento → diseño → implementación → prueba, que la norma considera esencial.

Facilita la gestión de riesgos

La relación de los casos de uso con los posibles riesgos (por ejemplo, fallas en operaciones críticas) permite integrar la gestión de riesgos del producto, (ISO 14971), dentro de los procesos de desarrollo y verificación exigidos por IEC 62304. Matriz de trazabilidad integral.

Soporte a las pruebas de aceptación

Los casos de uso se convirtieron en insumos directos para la definición de pruebas de sistema, garantizando que la validación cubra los escenarios de uso más relevantes.

Estructura

A continuación, se detalla la estructura que contendrán los casos de uso:

1. **Identificador-Versión:** Un identificador único con la última versión**.**
2. **Título del caso de uso**: Nombre descriptivo y claro.
3. **Actor(es)**: Identificación de quién interactúa con el sistema.
4. **Objetivo**: Qué busca lograr el actor.
5. **Precondiciones**: En caso de que existan.
6. **Curso principal**: Los pasos secuenciales para alcanzar el objetivo.
7. **Curso alternativo o de excepción**: Casos de error o rutas alternativas.

CU1 - 000 - Establecer la conexión

Relacionado con requerimiento funcional: *RF1-000 - Establecer la conexión*

1. **Actor**: *Médico/Enfermero/Técnico de Impulse Dynamics*

2. **Objetivo**: Conectarse al IPG a través de la Wand y establecer una sesión de comunicación válida.

3. **Precondiciones**:

    a. El IPG debe estar dentro del alcance de la Wand.
    b. DynamicScan debe estar en ejecución.
    c. La Wand debe conectarse previamente mediante una interfaz USB en el dispositivo donde se está ejecutando DynamicScan.

4. **Curso Principal:**

    a. DynamicScan debe detectar la antena Wand conectada, indicando en la parte inferior de la ventana "*Wand Connected*".
    b. El usuario presiona el botón *"Connect"*.
    c. DynamicScan envía una solicitud de conexión al IPG.
    d. El IPG responde con los siguientes datos:
        i. Start session (fecha y hora de inicio de sesión)
        ii. IPG Time (fecha y hora del IPG)
        iii. Model (modelo del IPG)

iv. Serial (serie del IPG)

e. DynamicScan detecta el estado de la sesión establecida, indicando en la parte superior de la ventana "*Established session*".

f. DynamicScan muestra en la ventana específica la información del detalle del IPG. (RF7-000)

5. Curso Alternativo 1 - Wand desconectada

a. DynamicScan debe detectar la antena Wand desconectada, indicando en la parte inferior de la ventana "*Wand Disconnected*".

b. El usuario intenta conectarse y DynamicScan muestra una alerta informando: *"Wand disconnected! Please connect the Wand before"*.

6. Curso Alternativo 2 - Otro dispositivo USB distinto a una Wand

a. Hay otro dispositivo con interfaz USB distinto a la antena Wand Conectado. - DynamicScan debe detectar la antena Wand desconectada, indicando en la parte inferior de la ventana *"Wand Disconnected"*.

b. El usuario intenta conectarse - **Curso Alternativo 1 - Wand desconectada.**

7. Curso Alternativo 3 - Conexión a otro IPG distinto del *Smart Mini*

a. El usuario conecta un IPG con otro modelo diferente al *Smart Mini* - El usuario intenta conectarse y DynamicScan muestra una alerta informando: "IPG *model <modelo detectado> not compatible detected, please connect an ID_SMART-MINI_IPG*"

8. **Curso Alternativo 4 - Conexión sin IPG**

a. El usuario intenta establecer una conexión sin un IPG conectado a la Wand.

b. DynamicScan muestra una alerta informando: "*CANCELLED. The operation was canceled because no response was received in the expected time*"

CU2-000 - Cerrar la conexión

Relacionado con el requerimiento funcional: *RF2-000 - Cerrar la conexión*

1. Actor: *Médico/Enfermero/Técnico de Impulse Dynamics*

2. **Objetivo:** Finalizar la sesión de comunicación entre la aplicación DynamicScan y el IPG, asegurando una desconexión segura.

3. **Precondiciones**:

    a. La sesión debe estar establecida entre DynamicScan y el IPG.
    b. DynamicScan debe estar en ejecución.

4. **Curso principal:**

    a. El usuario tiene la sesión activa con el IPG.
    b. El usuario hace clic en el botón/icono de salida.
    c. DynamicScan pregunta en un pop-up si está seguro de cerrar la conexión.
    d. El usuario presiona el botón Confirm.
    e. DynamicScan cierra la sesión y se redirige a la pantalla inicio de sesión.
    f. DynamicScan muestra el estado de la conexión en color rojo: "*Session Disconnect*".

5. **Cursos alternativos 1 - Cancelar cierre de conexión**

    a. El usuario tiene la sesión activa con el IPG.
    b. El usuario hace clic en el botón/icono de salida.
    c. DynamicScan pregunta en un pop-up si está seguro de cerrar la conexión.
    d. El usuario presiona el botón Cancel.
    e. DynamicScan no cierra la conexión y retorna a la pantalla actual.
    f. DynamicScan continúa mostrando en la pantalla actual el estado de la conexión en color verde: "*Established Session*".

6. **Cursos alternativos 2 - Cerrar conexión con desconexión de Wand antes de confirmar.**

    a. El usuario tiene la sesión activa con el IPG.
    b. El usuario hace clic en el botón/icono de salida.
    c. DynamicScan pregunta en un pop-up si está seguro de cerrar la conexión.

d. Se desconecta la antena Wand del puerto USB antes de Confirmar.
e. El usuario presiona el botón Confirm.
f. DynamicScan muestra una alerta informando: "*Antenna Wand not found*".
g. DynamicScan cierra la conexión y retorna a la pantalla principal.
h. DynamicScan continúa mostrando en la pantalla actual el estado de la conexión en color rojo: "*Session disconnect*".

CU4-000 - Obtención de Datos

Relacionado con requerimiento funcional: *RF4-000 - Obtención de Datos*

1. **Actor:** *Médico/Enfermero/Técnico de Impulse Dynamics*
2. **Objetivo:** Permitir al usuario obtener la información del IPG (dispositivo implantable).
3. **Precondiciones**:
    a. La aplicación DynamicScan debe estar en ejecución.
    b. Debe haber una sesión establecida con el IPG.
4. **Curso Principal:**
    a. El usuario visualiza la interfaz con una sesión activa con el IPG.
    b. Hace clic en el botón "*Interrogate*".
    c. DynamicScan inicia la solicitud de obtención de información del IPG.
    d. DynamicScan recibe los datos del IPG y habilita el botón "View *current interrogate report*" y *"Alerts".*
    e. DynamicScan guarda el reporte en un PDF cuyo nombre será único, con el siguiente formato: "<serial_fecha_hora>.pdf"
5. **Curso Alternativo 1 - Desconexión de Wand antes de interrogar**
    a. El usuario hace click en el botón interrogar.
    b. DynamicScan muestra una alerta informando: "*Connection Closed. Antenna Wand Not Found*".
    c. DynamicScan retorna a la pantalla inicial mostrando en la UI "*Wand Disconnected*" en color rojo.
6. **Curso Alternativo 2 - IPG desconectado**
    a. El usuario hace clic en el botón "Interrogate".

b. DynamicScan muestra una alerta informando que no hay un IPG conectado.

CU5-000 - Visualización de alertas

Relacionado con requerimiento funcional: *RF5-000 - Alertas*

1. **Actor**: *Médico/Enfermero/Técnico de Impulse Dynamics*
2. **Objetivo**: Permitir al usuario visualizar las alertas generadas por el **IPG** luego de la obtención de datos.
3. **Precondiciones:**
    a. La aplicación DynamicScan debe estar en ejecución.
    b. La sesión con el IPG debe estar establecida.
    c. Se debe haber completado la obtención de datos (RF4-000).
4. **Curso Principal:**
    a. Luego de completar la obtención de datos, el usuario visualiza el botón "*Alerts"* con un contador en color rojo indicando el número de las alertas.
    b. El usuario hace clic en el botón "*Alerts*".
    c. DynamicScan muestra un pop-up con las alertas obtenidas.
5. **Curso Alternativo 1 -No hay alertas generadas**

    a. Luego de completar la obtención de datos, el usuario visualiza el botón "*Alerts"* con un contador en color rojo indicando 0 número de alertas.
    b. El usuario hace clic en el botón "*Alerts*".
    c. DynamicScan muestra un pop-up con las alertas obtenidas, que en este caso estarán en OFF.

CU7-000 - Visualización detalles del IPG

Relacionado con requerimiento funcional: *RF7-000 - Visualización detalles del IPG*

1. **Actor**: *Médico/Enfermero/Técnico de Impulse Dynamics*
2. **Objetivo**: Visualizar en la UI los siguientes datos: el modelo, número de serie, la fecha y hora de inicio de sesión y la fecha y hora del IPG.
3. **Precondiciones**:
    a. La aplicación DynamicScan debe estar en ejecución.

b. La sesión con el IPG debe estar establecida.
c. El modelo, número de serie, la fecha y hora de inicio de sesión y la fecha y hora del IPG, deben haber sido obtenidos correctamente en el proceso de establecer conexión y obtener la hora del IPG (RF1-000 y RF14-000).

4. **Curso Principal**:
    a. Luego de establecer la conexión, el usuario visualiza los datos en la UI.
5. **Curso Alternativo 1 - Hay una diferencia mayor a 1 hora entre la hora interna del IPG y la hora en que se establece la sesión.**
    a. DynamicScan muestra una advertencia en rojo en la hora del IPG e informa por IU: *"The IPG time differs significantly from the system time"*

CU8-000 - Visualizar Informe

Relacionado con requerimiento funcional: *RF8-000 - Visualizar Informe*

1. **Actor**: *Médico/Enfermero/Técnico de Impulse Dynamics*
2. **Objetivo**: Permitir al usuario visualizar un informe con los datos obtenidos del IPG en pantalla en el navegador configurado por defecto.
3. **Precondiciones**:
    a. La aplicación DynamicScan debe estar en ejecución.
    b. La sesión con el IPG debe estar establecida.
    c. Se debe haber completado la obtención de datos (RF4-000).
4. **Curso Principal:**
    a. Luego de completar la obtención de datos, el usuario visualiza el botón *"View current interrogate report".*
    b. El usuario hace clic en el botón *"View current interrogate report".*
    c. DynamicScan abre una nueva pestaña en el visor de PDF configurado por defecto con el informe en formato PDF.

CU9-000 - Generación de Logs

Relacionado con requerimiento funcional: *RF9-000 - Generación Logs*

1. **Actor**: *Técnico de Impulse Dynamics.*
2. **Objetivo**: Permitir que el prototipo genere y almacene logs de ejecución de forma persistente. Para mantener la trazabilidad, soporte o auditoría.
3. **Precondiciones**:
    a. DynamicScan debe haber sido ejecutado al menos una vez.

4. **Curso Principal:**
    a. El técnico de Impulse accede a la opción "*Logs*" en la raíz del proyecto.

CU10-000 - Visualizar Splash Screen

Relacionado con requerimiento funcional: *RF10-000 - Visualizar Splash Screen*

1. **Actor**: *Médico/Enfermero/Técnico de Impulse Dynamics*
2. **Objetivo**:
    a. Visualizar el logo de *Impulse Dynamics* al iniciar la aplicación antes de cargar la pantalla de inicio de sesión.
3. **Precondiciones:**
    a. DynamicScan debe haber sido ejecutado.
4. **Curso Principal:**
    a. El usuario inicia DynamicScan en su dispositivo.
    b. DynamicScan carga el *Splash Screen*, mostrando el logo de *Impulse Dynamics.*
    c. Se mantiene la pantalla por un tiempo determinado.
    d. Una vez completado el tiempo, el prototipo muestra la pantalla de inicio de sesión.

CU11-000 - Acceso a la sección avanzada

Relacionado con requerimiento funcional: *RF11-000 - Acceso a la sección avanzada*

1. **Actor**: Técnico autorizado de *Impulse Dynamic*
2. **Objetivo**: Acceder a datos y métricas del paciente y batería del implantable mediante autenticación con contraseña.
3. **Precondiciones**:
    a. La aplicación DynamicScan debe estar en ejecución.
    b. La sesión con el IPG debe estar establecida.
4. **Curso Principal**:
    a. Hacer clic en el icono del candado arriba a la izquierda de la pantalla.
    b. DynamicScan solicita una contraseña para verificar la autenticación.
    c. El usuario ingresa la contraseña correcta.
    d. DynamicScan concede el acceso a la sección avanzada.
    e. DynamicScan muestra un cartel color rojo en la esquina superior derecha que dice “*Advanced Mode*”.

5. **Curso Alternativo 1 - Contraseña incorrecta**
    a. El técnico autorizado ingresa una contraseña incorrecta.
    b. Se muestra un mensaje indicando que la autenticación ha fallado.
6. **Curso Alternativo 2 - Usuario cancela el acceso**
    a. Se cierra *pop-up* de autenticación.
    b. DynamicScan permanece en la pantalla actual.

CU12-000 - Visualización del Resumen Avanzado

Relacionado con requerimiento funcional: *RF12-000 - Visualización del resumen avanzado*

1. **Actor**: Técnico autorizado de *Impulse Dynamic.*
2. **Objetivo**: Permitir visualizar un resumen con valores clave obtenidos del IPG dentro de la sección avanzada.
3. **Precondiciones**:
    a. La aplicación DynamicScan debe estar en ejecución.
    b. La sesión con el IPG debe estar establecida.
    c. Se debe haber completado la obtención de datos.
    d. El usuario debe haber accedido correctamente a la sección avanzada tras autenticarse con su contraseña correcta.
4. **Curso Principal**:

    a. DynamicScan abre por defecto la pestaña "*Summary"*, carga y muestra los siguientes valores obtenidos del IPG:
        i. Impedance V1
        ii. R-Wave Amplitude V1
        iii. Impedance V2
        iv. R-Wave Amplitude V2
        v. PWave Amplitude
5. **Curso Alternativo 1 - Se desconecta el IPG luego de la interrogación y antes de ingresar al modo avanzado.**

    a. Se muestra un *pop-up* con un mensaje de alerta que dice *"Response invalid or incomplete Wand"*

6. **Curso Alternativo 2 - Antena Wand desconectada.**

a. Se muestra un *pop-up* con un mensaje de alerta que dice: "CONNECTION CLOSED. Antenna Wand not found"

CU13-000 - Trends

Relacionado con requerimiento funcional: *RF13-000 - Trends*

1. **Actor**: Técnico autorizado de *Impulse Dynamic.*
2. **Objetivo**: Permitir al usuario visualizar tendencias del IPG dentro de la sección avanzada.
3. **Precondiciones**:
    a. DynamicScan debe estar en ejecución.
    b. La sesión con el IPG debe estar establecida.
    c. Se debe haber completado la obtención de datos.
    d. El usuario debe haber accedido correctamente a la sección avanzada tras autenticarse con su contraseña.
4. **Curso Principal:**
    a. El usuario hace clic en la pestaña "*Trends*".
    b. Por defecto DynamicScan abre por defecto la pestaña "*Hours Activity" y* muestra la actividad de las últimas 48hs. Se muestran los siguientes datos:
        i. Average HR.
        ii. Activity.
        iii. Position.
        iv. Average HRV.
    c. El usuario hace clic en la pestaña "*Battery*" para ver el estado actual de la batería.
        i. DynamicScan muestra el estado de la batería.
    d. El usuario hace clic en la pestaña "*Leads*".
        i. DynamicScan muestra las gráficas de Impedancia y Amplitud de V1 y V2.

CU14-000 - Obtener la hora del IPG

- Relacionado con requerimiento funcional: *RF14-000 - Obtener la hora del IPG.*
- Este caso de uso está contemplado en el *CU7-000 - Visualizar detalles del IPG.*

CU16-000 - Visualización de Historial de Informe

Relacionado con requerimiento funcional: *RF16-000 - Visualización de historial de informe.*

1. **Actor**: *Médico/Enfermero/Técnico de Impulse Dynamics*
2. **Objetivo**: Permitir al usuario visualizar los informes almacenados de sesiones anteriores ordenados en forma cronológica.
3. **Precondiciones**:
    a. DynamicScan debe estar en ejecución.
4. **Curso Principal:**
    a. El usuario hace click en el botón "*History*" en la pantalla de establecer conexión.
    b. DynamicScan carga y muestra la lista informe ordenada cronológicamente con sus respectivos informes PDF.
    c. El usuario hace click en el botón "*View PDF*".
    d. Se abre el PDF en el visor de PDF configurado por defecto.

CU17-000 – Detectar estado de conexión de la Wand

Relacionado con requerimiento funcional: *RF17-000 – Detección de antena Wand*

1. **Actor:** *Médico/Enfermero/Técnico de Impulse Dynamics*
2. **Objetivo:** Detectar automáticamente si la antena Wand se encuentra conectada o desconectada al iniciar la aplicación, o si ocurre una desconexión/reconexión durante su ejecución, y mostrar dicha información en la interfaz de usuario.
3. **Precondiciones:**
    a. DynamicScan debe estar en ejecución.
    b. El sistema operativo debe tener acceso al puerto USB donde puede conectarse la Wand.
    c. La Wand puede estar conectada o no al iniciar la aplicación.
4. **Curso Principal:**
    a. El usuario inicia la aplicación DynamicScan.
    b. El sistema verifica automáticamente el estado de conexión de la Wand.
    c. El usuario conecta la Wand.
    d. El sistema detecta la conexión y muestra en la interfaz: "*Wand Connected*".

e. El usuario desconecta la Wand durante la ejecución.

f. El sistema detecta la desconexión y actualiza la interfaz a: "*Wand Disconnected*".

g. El usuario vuelve a conectar la Wand.

h. El sistema reconoce nuevamente la antena y actualiza la interfaz a: "*Wand Connected*", sin necesidad de reiniciar la aplicación.

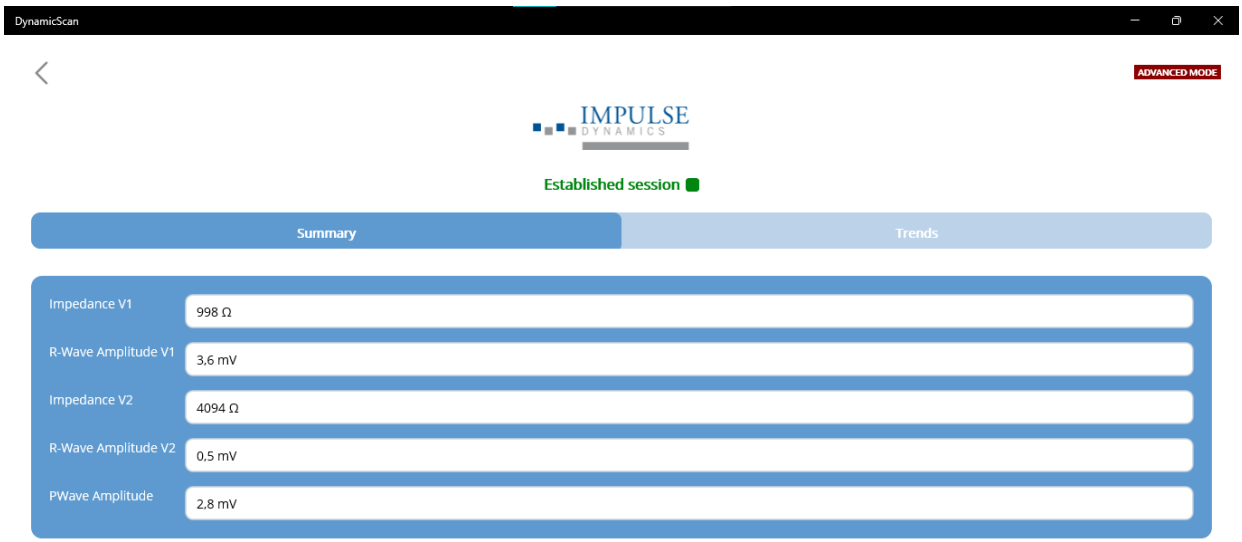
5. **Curso Alternativo 1 – Dispositivo USB no compatible:**

a. Se conecta un dispositivo con interfaz USB que no es una antena Wand al prototipo.

a. El sistema muestra un mensaje de alerta y mantiene el estado "*Wand Disconnected*".

---------- FIN ESRE ----------

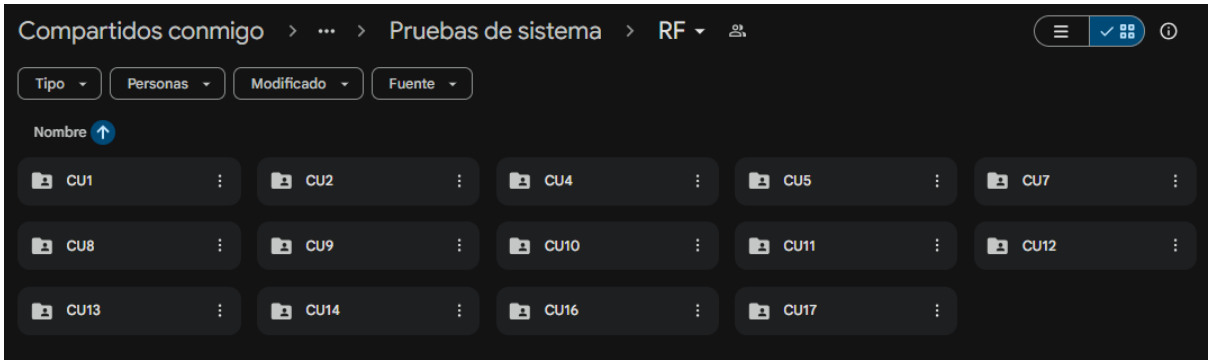
12.2.2. Resumen del menú avanzado

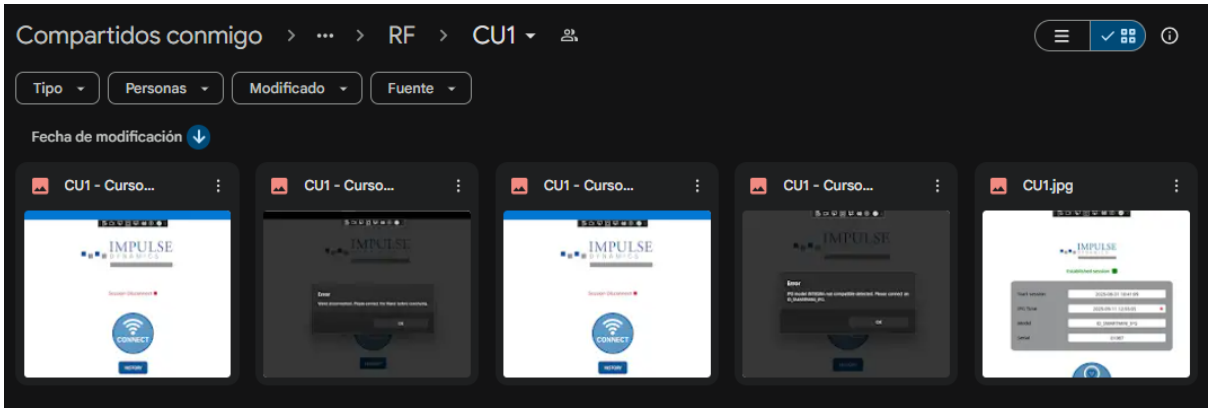


12.2.3. Tabla de prioridades

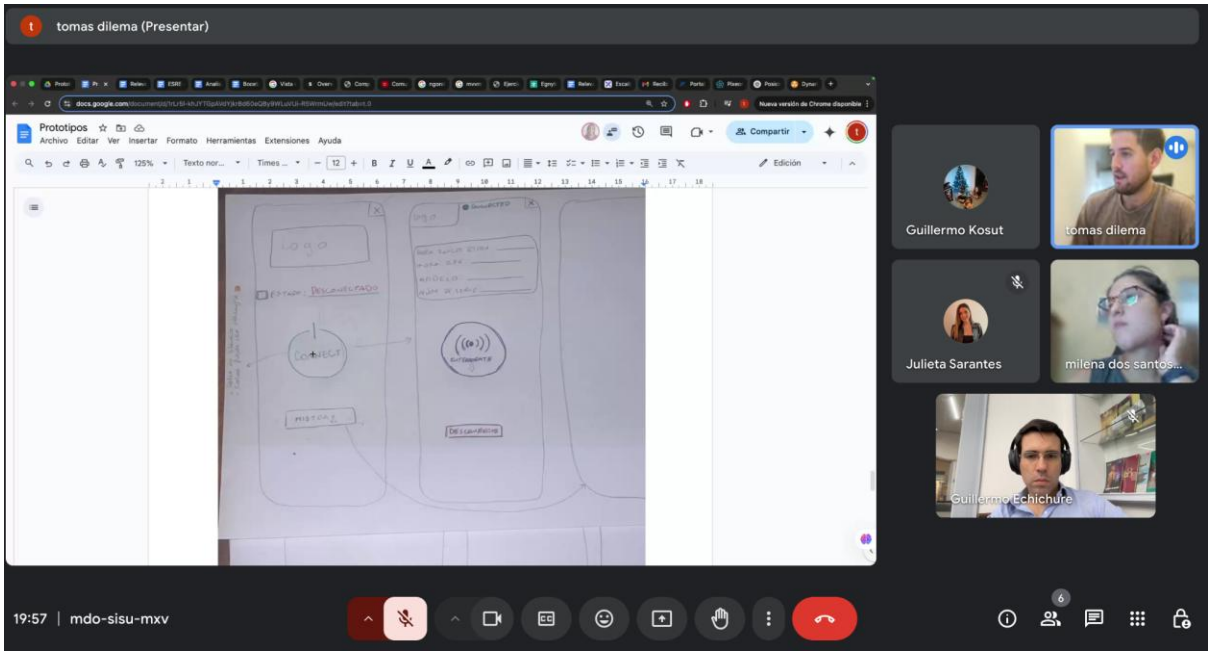
Prioridad	Descripción
Alta	funcionalidades críticas para el funcionamiento del sistema, indispensables para cumplir los objetivos principales definidos por el cliente.
Media	funcionalidades importantes, pero no esenciales para la primera versión del prototipo; pueden ser implementadas en etapas posteriores sin comprometer el uso clínico inicial.
Baja	funcionalidades complementarias o de valor agregado, cuya implementación se considera deseable pero no prioritaria dentro del alcance del presente desarrollo.

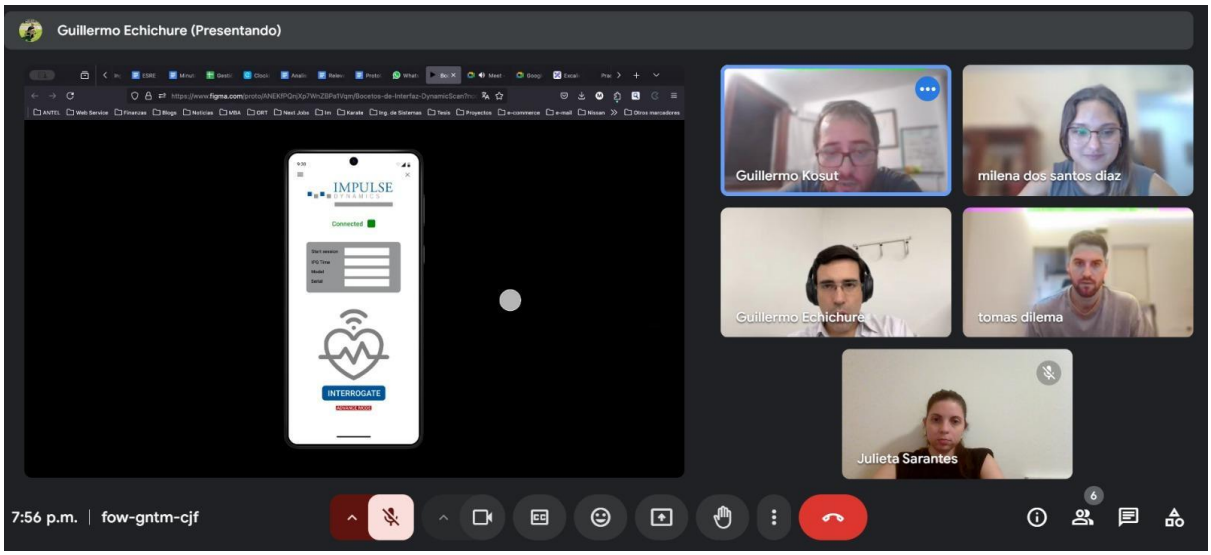
12.2.4. Documentación de casos de uso



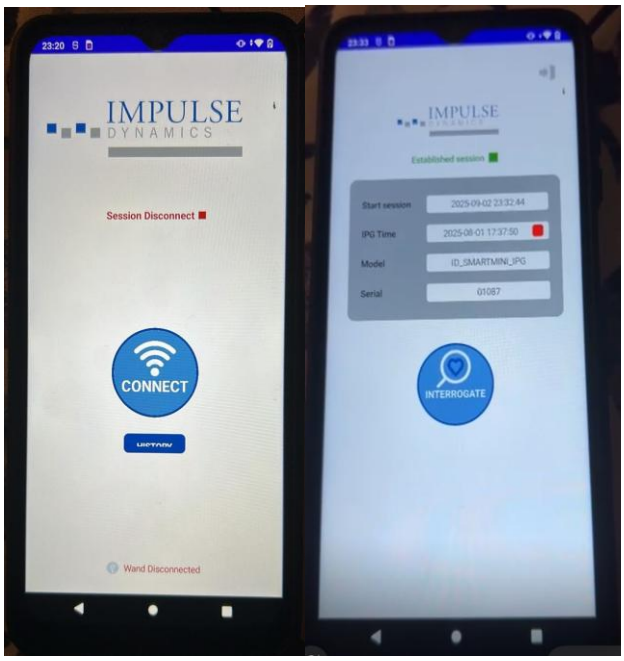


12.2.5. Evidencias de validación con prototipos





12.2.6. Pruebas de Concepto en Android



12.3. Construcción

12.3.1. Funcionalidades acordadas

Funcionalidades acordadas para la fase de desarrollo Recibidos ×

Julieta Sarantes [redacted] 4 abr 2025, 11:34 a.m.

Buenos días, Guillermo:

Tal como se acordó, compartimos a continuación las funcionalidades que se implementarán a lo largo de la fase de desarrollo:

Primer Release:

- Establecer conexión con el IPG
- Detalles del IPG
- **Obtener la hora del IPG**
- Mostrar Splash Screen
- Detectar antena Wand

Release final:

- Obtención de datos
- Visualización de alertas
- Visualización de informe
- Generación de logs
- Acceso a la sección avanzada
- Visualización del resumen avanzado
- Trends
- Visualización del historial de informes

Quedamos atentos ante cualquier comentario o ajuste que consideres necesario.

Saludos cordiales,
Julieta Sarantes
SQA – *DynamicScan*

Guillermo Kosut [redacted] 5 abr 2025, 1:40 p.m.

Buenas, ¿cómo están?

Perdón por la demora en responder.
Acá estamos arrancando la verificación del 1.5 y estamos un poco complicados.

El alcance que me plantean para cada release me parece correcto.
Para mi gusto de esa forma queda bien divido en dos etapas claras.
Una primera donde la mayor complejidad está en implementar los protocolos y la comunicación básica y una segunda de procesamiento, análisis y despliegue de datos recibidos.
Así que vamos con eso.

Aprovecho para avisárseles que los próximos meses probablemente voy a estar un poco más justo de tiempo.
Además, puede pasar que las salas de reuniones se usen como "laboratorios" para pruebas de los IPGs y no siempre estén disponibles.
En todo caso, los días que vayan a estar viniendo por acá, me avisan y vemos como arreglamos.

Saludos,
Guille

Learn How *CCM*® Works – Click Here

Guillermo Kosut
Associate Manager, Software Development

IMPULSE DYNAMICS

12.3.2. Control de correctitud de los datos tras interrogación

Control de valores tras interrogación y generación de informes						
DynamicScan		IMPULSE DYNAMICS TABLA DE VALORES SEGUN DOC. OSM REC_459			RESULTADO	
Campo	Valor Obtenido	Rango válido	Unidad	Valor por defecto	Cumple S/N	Lugar Impulse Dymacis
Mode	ODO-LS-CCM	OOO/OVO-LS-CCM / ODO-LS-CCM		OOO	S	CCM Mode
CCM Therapy Mode	ON	Off / On	Off / On	Off	S	Se configuró en ON
CCM hs/day	13 hs/day	1 to 24 in steps of 1	h/day	5h/day	S	
Start Time (HH:MM)	0:00	00:00 23:59 in steps of 1 min	00:00 23:59 in steps of 1 min	0:00	S	
End Time (HH:MM)	23:59	00:00 23:59 in steps of 1 min	00:00 23:59 in steps of 1 min	23:59	S	
On Time (HH:MM)	1:00	The PSw shall display the CCM On Time and CCM Off Time on the screen but they shall not be programmable by the user. CCM On Time / CCM Off Time Calculation for Display	Calculated	1:00	S	Programado por cliente
Off Time (HH:MM)	0:50		Calculated	3:48	S	Programado por cliente
Extend on Low CCM%	ON	Checked / not checked	Checked / not checked	ON	S	
Magnet Mode	24 hs	Off 1 day / Off	day	Off 1 day	S	Impulse le llama 1 day
Atrial Sensivity	1 mV	0.3, 0.7, 1.0, 1.3, 1.7, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0	mV	1 mV	S	
Atrial Sensing Polarity	Bipolar	Unipolar / Bipolar	Unipolar/Bipolar	Bipolar	S	
Ventricle 1 Sensivity	2 mV	0.3, 0.7, 1.0, 1.3, 1.7, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0	mV	2mV	S	
Ventricle 1 Sensivity Polarity	Bipolar	Unipolar / Bipolar	Unipolar / Bipolar	Bipolar	S	
Ventricle 2 Sensivity	2 mV	0.3, 0.7, 1.0, 1.3, 1.7, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0				
Ventricle 2 Sensivity	Bipolar	Unipolar / Bipolar	Unipolar / Bipolar	Bipolar	S	
Atrial Refractory	249,4 ms	148 to 453 in steps of 7.8	ms	249.4 ms	S	
Ventricular Refactory	249,4 ms	148 to 453 in steps of 7.8	ms	249.4 ms	S	
Short AV	70 ms	23 to 398 in 7.8 ms steps	ms	70 ms	S	
Long AV	397 ms	23 to 398 in 7.8 ms steps	ms	397 ms	S	
Rate Averaging	1 beat	1 / 4 / 8	beat	1 beat	S	
LS Alert Start	0 ms	-40 to 80 in 2 ms steps	ms	0 ms	S	
LS Alert Width	15 ms	1 to 40 in 1 ms steps	ms	15 ms	S	
Pre Atrial LS Refractory	0 ms	0 to 55 in 5ms steps	ms	0ms	S	
Post Atrial LS Refractory	0 ms	0 to 55 in 5ms steps	ms	0ms	S	
Pre Ventricular LS Refractory	0 ms	0 to 39 in 1ms steps	ms	0ms	S	
Post Ventricular LS Refractory	0 ms	0 to 39 in 1ms steps	ms	0ms	S	
Post LS Refractory	30 ms	15 to 250 in 5ms steps	ms	30ms	S	
Number of Pulses	2	1 / 2 / 3		2	S	
CCM Train Delay	30 ms	3 to 140 in steps of 1	ms	30 ms	S	
Balancing Phase Duraton	40 ms	40 to 100 in steps of 10	ms	40 ms	S	
Compare Phases Current	Enabled	Enable/Disable	Enable/Disable	Disable		Programado por cliente
Compare Phases Enabled	Enabled	Enable/Disable	Enable/Disable	Disable		Programado por cliente
Interval	2 ms	0 to 7 ms in steps of 1	ms	2 ms	S	
Phase Duraon	5,14 ms	5.13 / 5.62 / 6.11 / 6.60	ms	5.13ms	S	
Alert Delivery Mode	Schedule	Never, Never, Scheduled	Never, Never, Scheduled	Deliver alert within specified time	S	
Paent Alert Start Time (HH:MM)	9:00	00:00 to 23:59 in 1 min steps	min steps	9:00	S	
Paent Alert End Time (HH:MM)	21:00	00:00 to 23:59 in 1 min steps	min steps	21:00	S	
Battery Recharge Reminder Days	Disabled	Checked / not checked	Checked / not checked	Disable	S	
CCM Safe Mode	OFF	ON/OFF	ON/OFF	OFF	S	Programado por cliente
CCM Not Sensing/Noise	ON	Checked / not checked	ON/OFF	OFF	S	Programado por cliente
CCM Suspended	OFF	ON/OFF	ON/OFF	OFF	S	Programado por cliente
Charger Baery Low	ON	ON/OFF	ON/OFF	OFF	S	Programado por cliente
ChargerFailure	OFF	ON/OFF	ON/OFF	OFF	S	Programado por cliente
Baery Low	ON	ON/OFF	ON/OFF	OFF	S	Programado por cliente

12.4. Gestión de la calidad

12.4.1. Matriz de riesgos del producto

Matriz de Registro y Trazabilidad de los Riesgos del Producto - Versión 2.0									
ID_Riego	Nombre del Riesgo	Descripción	Origen de causa potencial	Probabilidad de ocurrencia (1 al 5)	Gravedad del daño (1 al 5)	¿Es Aceptable ?	Acción preventiva / Medida correctiva	Verificación	Es aceptable el riesgo residual ? SI/NO	Etapa
RProd-1-000	Riesgo de escritura no intencionada en uso previsto	El sistema podría enviar comandos de escritura al IPG de forma no intencionada, lo que implicaría la modificación de parámetros terapéuticos y un potencial daño al paciente.	Prototipo desarrollado por un equipo de estudiantes, con plazos acotados, presión de entrega y sin supervisión experta en software médico, lo que incrementa la posibilidad de errores de implementación.	5	5	NO	El riesgo se mitiga mediante un diseño limitado, excluyendo operaciones de escritura	Revisión del ESRE confirma que solo se implementan operaciones de lectura definidas por el cliente.	SI	Ing. de Requerimientos
RProd-2-000	Uso del software con un dispositivo IPG no compatible	El uso de DynamicScan con un modelo de IPG distinto al Smart Mini puede provocar errores en la interpretación de comandos o en la estructura de los datos recibidos, generando comportamientos no esperados.	Ausencia de validación en el tipo de dispositivo conectado; posibilidad de conexión accidental o intencional con un IPG distinto al previsto. Solo aplicable a RF1-000	4	2	NO	Validar modelo de IPG en clase DeviceProtocolFactory con métodos que controlen la versión.	pruebas unitarias: PU26-000, PU28-000.	SI	Diseño detallado
RProd-3-000	Mal uso razonablemente previsible, persona ajena a Impulse Dynamics	El uso de la seccion avanzada por parte de personal no técnico podría exponer al usuario a datos que este no puede interpretar correctamente, generando errores en el manejo de la información.	Falta de control de acceso por perfil de usuario; ausencia de validación de rol o permisos en la interfaz del sistema.	4	2	NO	Implementar control de acceso a sección avanzada mediante contraseña a nivel de la UI, esconder la contraseña al digitarla.	Pruebas unitarias: PU128-000, PU129-000, PU130-000, PU131-000.	SI	Diseño detallado
RProd-4-000	Riesgo de errores funcionales en el acceso a la base de datos	Si no se prueban directamente las operaciones de escritura y lectura en SQLite, podrían pasar desapercibidos errores en consultas o en cómo se almacenan datos importantes, afectando la información clínica registrada.	No hacer pruebas de integración sobre la base de datos puede hacer que pasemos por alto errores reales en las operaciones del repositorio, que no se detectan con mocks o pruebas unitarias.	4	2	NO	Ejecutar pruebas de integración directas sobre la clase SQLiteDeviceRepository, validando inserciones, actualizaciones y consultas con una base de datos real.	Pruebas de Integración: PI22-000, PI23-000, PI24-000	SI	Diseño de arquitectura
RProd-5-000	FTD2XX es un componente de tercero.	La librería FTD2XX_NET.dll, provista por la empresa FTDI, se utiliza para establecer la comunicación con la antena Wand a través del puerto USB 2.0. Si bien su código es accesible, el proveedor no garantiza la calidad ni el funcionamiento libre de errores del software. Dado que las funciones de comunicación (_ftdiOpen, _ftdiRead, _ftdiWrite, ftdiClose) requieren la presencia física del hardware, no pueden probarse unitariamente de forma aislada.	Un fallo no detectado en esta capa podría propagarse a capas superiores (Protocol, DeviceApi) y generar respuestas inválidas o falsos negativos sin una excepción evidente.	5	2	NO	Ejecutar pruebas de integración directas sobre la clase FtdiWindowsTransportService del módulo TransportLayer, verificando las operaciones críticas del driver (Open, ReadAsync, WriteCommandAsync, Close) en escenarios con hardware conectado y desconectado.	Se definieron pruebas PI2_000 hasta PI15_000, con hardware conectado y desconectado. Las pruebas se ejecutan directamente contra la Wand, verificando el correcto uso de la libreria FTD2XX_NET.dll. Se agregan pruebas para USBWatcher: PI16_000, PI17_000, PI18_000, PI19_000, PI20_000 y PI21_000	SI	Diseño de arquitectura
RProd-6-000	Lectura errónea de datos por corrupción en la transmisión - Nivel modulo	Implementacion incorrecta de la funcion de CRC establecida en el protocolo FP17.	Una implementacion incorrecta de la funcion de CRC establecida en el protocolo FP17 podría incurrir en no detectar mensajes corruptos, afectando la integridad de la información recibida.	5	2	NO	Para mitigar este riesgo se implementó correctamente la función de CRC (Cyclic Redundancy Check) conforme a la especificación del protocolo FP17, y se diseñaron pruebas unitarias exhaustivas (PU2-000 a PU12-000) orientadas a validar su comportamiento en diferentes escenarios: tramas válidas, tramas con errores intencionales, alteraciones de longitud y contenido.	PU2-000, PU3-000, PU4-000, PU5-000, PU6-000, PU7-000, PU8-000, PU9-000, PU10-000, PU11-000, PU12-000	SI	Diseño detallado.
RProd-7-000	Interpretación errónea de respuestas parciales.	Recepción de datos incompletos que pueden ser interpretados como válidos.	Si llegan mensajes parciales o incompletos y no hay un buen control en la lectura o los tiempos de espera son muy cortos, el sistema podría tomar datos incompletos como si fueran correctos.	3	2	SI	Implementar lógica de control, validación de mensajes y tiempos de espera adecuados, aun con un riesgo aceptable.	PU88-000PU89-000, PU79-000, PU80-000, PU81-000, PU82-000, PU87-000	SI	Diseño detallado.
RProd-8-000	Error en la sincronización de hora entre el IPG y la app	Desfase horario entre los datos reportados por el IPG y DynamicScan, que puede afectar la interpretación de la información.	La hora del IPG es interna al dispositivo, y puede no estar alineada con la del sistema donde corre DynamicScan. Una diferencia grande entre ambas horas, podria incurrir en una mala interpretacion de los estados del paciente	5	2	NO	Verificar que la hora del IPG con la hora de inicio de sesión de DynamicScan mantengan una diferencia de tolerancia de <= 1h.	PS24-000	SI	Ing. de Requerimientos. RF7-000 - CU7-000 -CA1
RProd-9-000	Exposición de datos sensibles de pacientes	Riesgo hipotético de asociación entre datos clínicos obtenidos y la identidad del paciente. Actualmente no se almacenan identificadores personales.	Si pudiera establecerse una relación entre los datos clínicos obtenidos tras al interrogación del dispositivo y los datos de pacientes. El prototipo solo almacena metadatos de sesión y paths a archivos PDF, no almacena ningún dato de paciente. No hay riesgo.	1	1	SI	Encriptar la contrasena, como medida preventiva aun sin el riesgo.	PS15-000	SI. Se recomienda que, aunque en la versión actual no se almacenan datos personales de pacientes, la ausencia de encriptación en el almacenamiento podría convertirse en un factor de riesgo en versiones futuras si se incorporan datos adicionales. Se recomienda encriptar todos los datos de la base.	Ing. de Requerimientos RNF-SEG-1-000

12.4.2. Matriz de plan de la calidad

Fase	Actividad	Descripción	Entrada	Salida	Herramienta, técnica o tecnología	Rol encargado	Roles participantes
Investigación y ánalisis	Comprensión del problema	Analizar y entender el problema presentado por el cliente.	Descripción inicial del problema proporcionada por el cliente, ejemplos de documentación técnica y/o regulatoria.	Declaración entendible y clara del problema, objetivos específicos y limitaciones conocidas.	Reuniones con técnicos de Impulse Dynamics.	Gerente de proyecto	Equipo
	Recolección de información	Recopilar los datos, requisitos y normativas a partir de entrevistas y revisión de la documentación.	Normas IEC 62304, ISO 14971, guía de ciberseguridad y documento de requisitos de aplicación de programador. Entrevistas con el cliente y estándares que se requieren, capacitaciones.	Lista inicial de requerimientos y aspectos normativos.	Entrevistas, reuniones, análisis de documentos.	Ingeniero de requerimientos	Equipo
	Análisis de documentos	Revisar, entender y mapear las normas y documentos aplicables al proyecto.	Normas IEC 62304, ISO 14971, guía de ciberseguridad y documentos propietarios.	Documentos relevando cada norma y guía de cyberseguridad. Entendimiento inicial del documento de requisitos de aplicación del programador.	Lectura y análisis de guías normativas (ISO 62304, IEC 14971), guía de ciberseguridad y documentos propietarios.	SQA	Equipo
	Pruebas de concepto	Realizar pruebas iniciales para evaluar la viabilidad técnica de integrar tecnologías, librerías y protocolos necesarios para el proyecto, verificando compatibilidad con hardware, drivers y entornos de desarrollo.	Documentación técnica de protocolos y librerías, requerimientos iniciales del cliente, entornos de desarrollo propuestos.	Resultados de viabilidad técnica documentados, identificación de limitaciones y definición de posibles alternativas de implementación.	Pruebas de integración con drivers (ej. FTDI), entornos de desarrollo .NET MAUI.	SQA	Equipo

Ingeniería de requerimientos	Especificación de requerimientos	Documentar y detallar los requisitos identificados, integrando la comprensión del problema.	Normas IEC 62304, ISO 14971, guía de ciberseguridad y documentos propietarios. Entrevistas con el cliente y estándares que se requieren, capacitaciones.	ESRE - Requisitos funcionales, no funcionales y normativos.	Entrevistas, reuniones, análisis de documentos. Requerimientos funcionales se especificaron mediante casos de uso, los requerimientos no funcionales se documentaron en forma de atributos de calidad asociados cada uno acompañado de una breve descripción y su justificación.	Ingeniero de requerimientos	Equipo, cliente.
	Definición del alcance	Delimitar los objetivos del proyecto y las funciones claves del sistema.	Requisitos iniciales recopilados, y objetivos del cliente claros.	Aprobación del alcance.	Reuniones con el cliente, análisis de requerimientos, talleres de trabajo colaborativo (*workshops*).	Gerente de proyecto	Equipo, cliente.
	Validación de requerimientos	Confirmar que los requerimientos que se recopilaron, son claros, completos y verificables.	Lista de requerimientos inicial.	Aprobación y validación del cliente de los requerimientos presentados.	Revisiones de requerimientos, matriz de trazabilidad.	SQA.	Equipo, cliente.
	Priorización de los requerimientos	Ordenar los requerimientos según priodad para el cliente.	Requisitos validados, objetivos del proyecto.	Lista priorizada de requerimientos.	Herramienta de gestión (Jira)	Ingeniero de requerimientos.	Equipo, cliente.

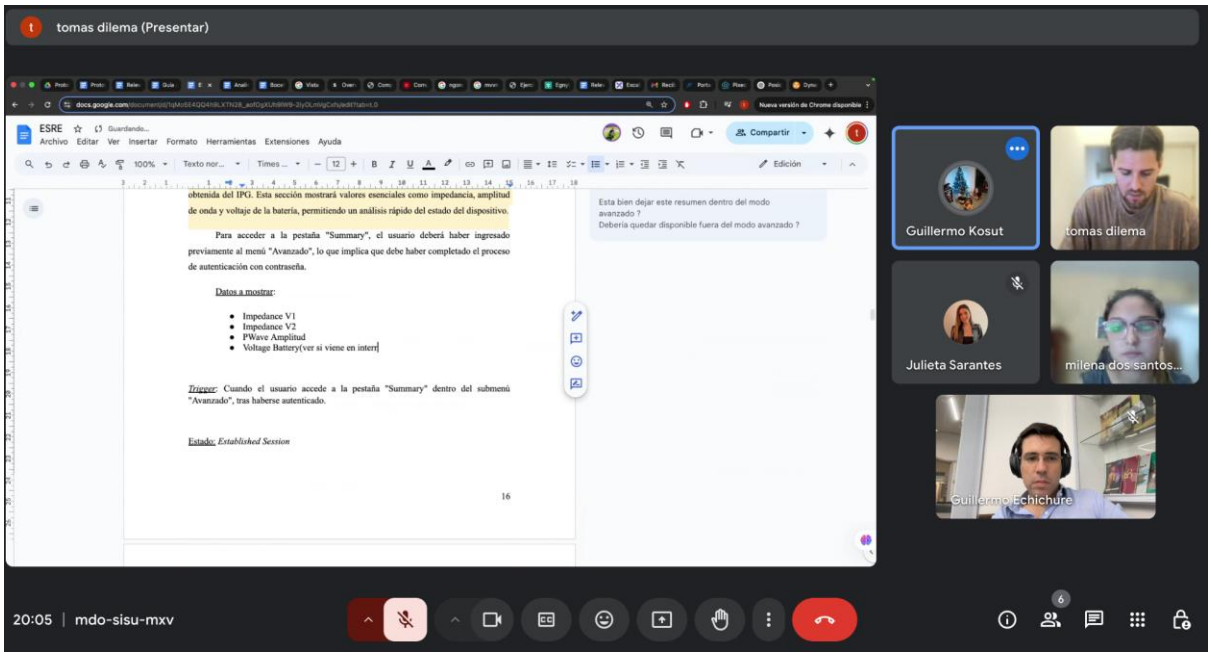
Diseño	Análisis de requisitos de diseño y arquitectura	Definición de los componentes de la solución basados en los requerimientos funcionales y no funcionales.	ESRE, documento de alcance.	Especificación inicial del diseño del sistema.	Digramas, UML, modelado conceptual.	Ingeniero de requerimientos.	Equipo, cliente.
	Evaluación de diferentes soluciones arquitectónicas	Discusión entre los distintos patrones de diseño y arquitectura que mejor se adapten al problema.	Requerimientos funcionales y no funcionales, análisis preliminar del sistema, documentación de estándares (IEC 62304, ISO 14971, guía de ciberseguridad).	Selección de la arquitectura más adecuada (ej. arquitectura por capas con MVVM), justificación de la decisión y lineamientos para el diseño detallado.	Draw.io, Excalidraw	Arquitecto de Software.	Equipo, cliente.

Construcción	Codificación	Desarrollo de los módulos definidos en la etapa de diseño, implementando las funcionalidades priorizadas en el sprint backlog conforme a los estándares establecidos.	Sprint backlog	Código fuente	Visual Studio (.NET MAUI, .NET 8)	Equipo.	Equipo.
	Revisión de código	Evaluación de los cambios realizados mediante revisiones cruzadas, asegurando cumplimiento de estándares, calidad del código y corrección de posibles defectos.	Código fuente y estándares.	Código revisado de forma cruzada.	Bitbuket - *pull requests, code review*	SQA.	Equipo.
	Pruebas unitarias	Implementación y ejecución de pruebas a nivel de módulo, con el objetivo de verificar el correcto funcionamiento de cada unidad de software de forma aislada.	Sprint backlog	Pruebas unitarias implementadas.	XUnit (*Arrange, Act, Assert*)	Equipo.	Equipo.

Pruebas	Pruebas de integración	Validación del correcto funcionamiento de los módulos al interactuar entre sí, verificando la comunicación a través de las interfaces definidas y detectando posibles errores de integración.	Sprint backlog	Pruebas de integración implementadas.	XUnit, *drivers* FTDI D2XX para pruebas de comunicación.	Tester	Tester, líder de SQA.
	Pruebas de sistema	Evaluación del sistema completo como un todo, comprobando que cumpla los requerimientos funcionales y no funcionales establecidos, y que responda adecuadamente en su contexto de uso.	Sistema integrado, ESRE, casos de uso definidos.	Matriz de trazabilidad de issues.	Matriz de trazabilidad de pruebas, validaciones con cardióloga y cliente.	Líder de SQA.	Tester, cliente, cardióloga.

Procesos de gestión (independientes de la fase)	Evaluación de riesgos	Identificar los riesgos normativos, técnicos y operativos que puedan surgir.	Requisitos validados, conocimiento del entorno del proyecto, norma ISO 14971, ESRE, supuestos.	Registro de riesgos con planes inciales de mitigación.	Norma ISO 14971, reuniones de equipo, matrices de riesgo, ESRE.	Gerente de proyecto	SQA, Gerente de proyecto, Cliente.
	Análisis de viabilidad	Evaluar la viabilidad técnica, normativa y económica del proyecto.	Lista de requerimientos validados, análisis de normas, restricciones técnicas.	Informe de viabilidad detallado.	Estudio de factibilidad, análisis costo-beneficio, reuniones con cliente (Google Meet)	Gerente de proyecto	Equipo, cliente.
	Planificación inicial	Diseñar un cronograma inicial y asignar roles y objetivos clave para el desarrollo.	Documento de alcance, registro de riesgos, requerimientos priorizados, restricciones técnicas y normativas.	Cronograma inicial con hitos y responsables asignados.	Herramienta de gestión de proyectos (Jira)	Gerente de proyecto, SQA	Equipo.

12.4.3. Reunión de revisión ESRE



12.4.4. Ejemplo de minuta cliente

	Cuestionario para Cliente 2	
	Guillermo Echichure	
	Fecha 13/01/25	Versión 0.2

Modalidad: Online

Herramienta utilizada: Google Meeting

Hora: 20:00 hs.

Asistentes:

- **Equipo de proyecto**: Tomás Dilema, Milena Dos Santos, Julieta Sarantes y Guillermo Echichure.
- **Cliente**: Ing. Guillermo Kosut

Objetivo

Este cuestionario surgió tras el análisis de la norma IEC 62304, la actualización del ESRE y la incorporación de casos de uso.

- ☑ Qué datos específicos vamos a extraer del dispositivo ?
- ☑ Enumerarlos para el médico (modo simple) como para el técnico de Impulse (modo avanzado).
- ☑ Hay que tener un mecanismo de monitoreo de si todo está todo correcto para saber si está listo para comenzar a utilizarse ?
- ☑ Verificar Casos de Uso - toda la información que está en color rojo.
- ☑ Qué datos vamos a guardar en la base de datos.
- ☑ Usuario y contraseña de los usuarios se guardan en la misma base de datos ?
- ☑ La base de datos tiene que tener un tipo de replicación o backup ?
- ☑ Ver con guillermo si los Casos de uso irían dentro del ESRE. Proponer alternativas si todo está junto o más segregado.

12.4.5. Tabla de verificación

Producto a evaluar	**Criterios/Técnica**	***Responsable***	***Registro de resultados***	***Cuándo se realiza***
ESRE	Uso de lista de verificación para asegurar trazabilidad y cumplimiento de estándares.	Líder de SQA.	Lista de verificación completa.	Luego de las revisiones del ESRE.
Diseño	Se realiza una comparación sistemática entre el diseño propuesto y los requerimientos definidos en el ESRE y la revisión de diagramas de arquitectura con la identificación de posibles inconsistencias.	Ingeniero de Requerimientos	Minuta con observaciones y acciones correctivas.	Antes de iniciar la implementación.
Código fuente	Ejecución de pruebas unitarias y de integración, junto con análisis estático del código para asegurar cumplimiento de estándares de codificación.	Líder de SQA.	Reporte de análisis y descubrimientos.	Durante el desarrollo y previo a la integración.

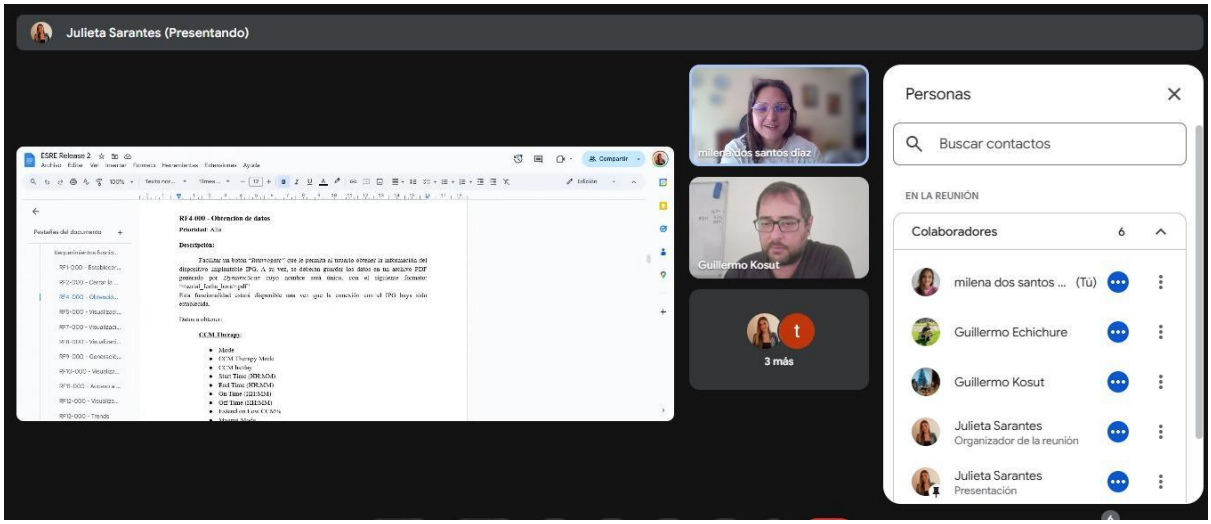
Plan de gestión de riesgos del producto	Se confirma la trazabilidad entre los riesgos identificados en el producto y las medidas de mitigación propuestas. El criterio consiste en verificar que cada riesgo se encuentre registrado en la matriz de riesgos, con un nivel de criticidad definido y vinculado a las actividades de control correspondientes.	Líder de SQA.	Minuta con observaciones y acciones correctivas.	Antes de la aprobación del plan.
Plan de pruebas	Se revisa la alineación del plan con los requerimientos funcionales y no funcionales, verificando que se cubran todos los casos definidos en el ESRE. Se asegura que las pruebas contemplen los distintos niveles (unitarias, integración, sistema).	Líder de SQA	Lista actualizada con casos de prueba.	Previo a la ejecución de pruebas de sistema.
Documentación final	Verificación de coherencia y alineación con las directrices de ORT.	Líder de SQA y equipo.	Minuta de verificación con la tutora.	Antes de la entrega final a ORT.

12.4.6. Lista de verificación ESRE

Nro de ítem	*Ítem*	*Criterio de Verificación*	*Cumple/No cumple*	*Observaciones*
1	Objetivos de seguridad	Metas claras de seguridad; autenticación y control de riesgo de acceso a datos sensibles.	Cumple	Se especifican mecanismos de autenticación y control de acceso.
2	Claridad y completitud	¿Se describe de forma clara cada requerimiento funcionales y no funcionales?	Cumple	No se detectaron ambigüedades relevantes; se ajustaron redacciones menores.
		¿Se evita la ambigüedad y los casos de usos son claros y precisos?	Cumple	
3	Trazabilidad de requerimientos	¿Cada requerimiento tiene su correcta referencia al caso de uso?	Cumple	Se asegura trazabilidad con identificadores únicos (RFx-yyy, RNFx-yyy).
		¿Cada requerimiento se alinea con los objetivos del producto?	Cumple	Validados con el cliente.
4	Normas	¿Los requerimientos tienen concordancia con lo investigado en las normas IEC 62304 e ISO 14971?	Cumple	Se mantiene alineación con gestión de riesgos (ISO 14971) y ciclo de vida (IEC 62304).
5	Requerimientos funcionales	¿Los casos de uso correspondientes a los requerimientos describen a detalle la funcionalidad con sus cursos alternativos?	Cumple	Se revisaron cursos alternativos; un caso de uso requirió ajuste para mayor detalle.

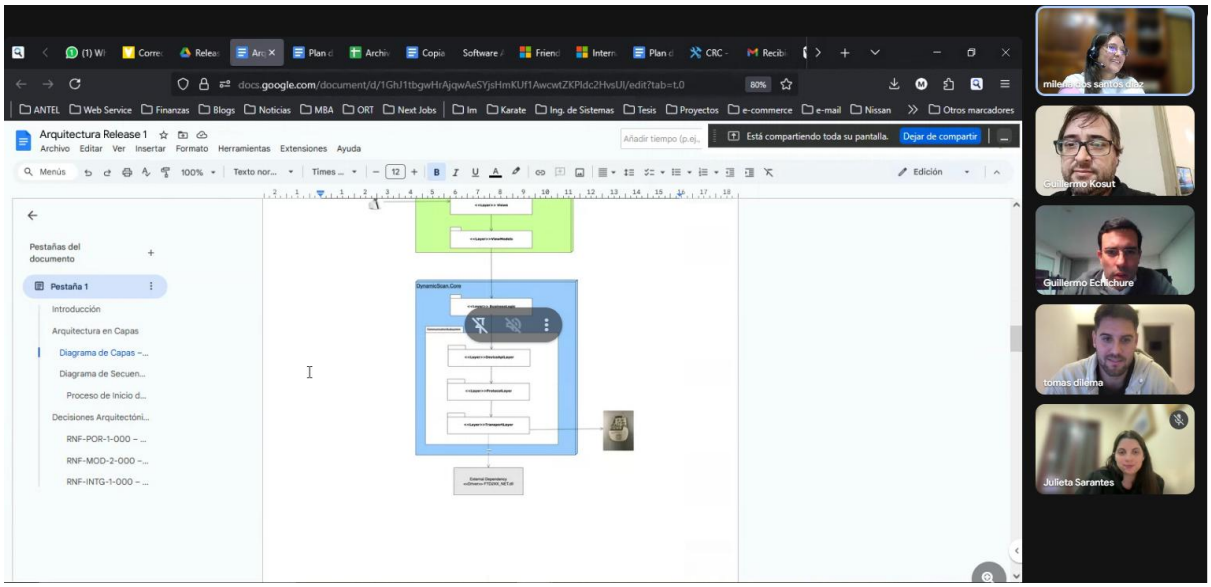
6	Requerimientos no funcionales	¿Estos requerimientos son claros y se definen atributos de calidad?	Cumple	Claridad y completitud validada con cliente.
7	Gestión de riesgos del producto	¿Tiene un enlace el cual nos dirige al plan de gestión de riesgos?	Cumple	Todos los riesgos tienen una correcta trazabilidad.
8	Soporte de Logs	¿Se menciona la necesidad de registrar acciones, tales como logs?	Cumple	Se tiene logs en los principales cursos de la aplicación.
9	Formato	¿Cumple con el formato especificado en estándares de documentación?	Cumple	Se siguió el template definido por el equipo.
	Claridad	¿Cada palabra definida en el mismo es completamente entendible para cualquier lector y en caso de que no, tiene su definición?	Cumple	Contexto y glosario al principio del documento.
10	Priorización	¿Cada requerimiento tiene su nivel de prioridad?	Cumple	Priorización documentada

12.4.7. Ejemplo validación ESRE

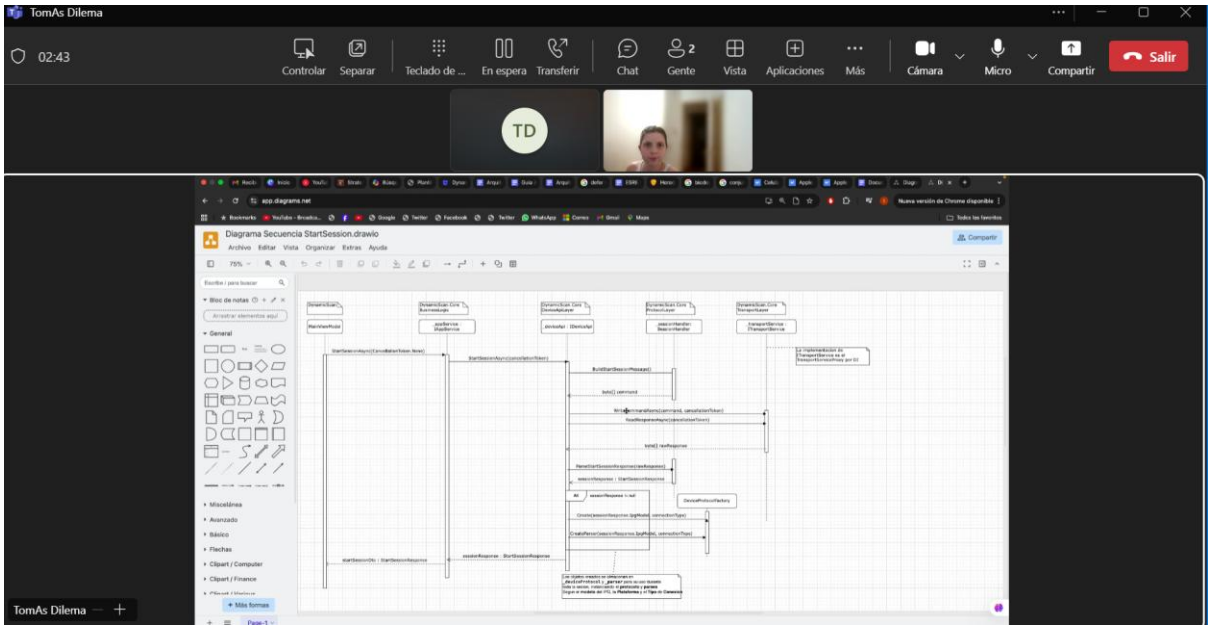


12.4.8. Reunión refinamiento de diseño arquitectónico

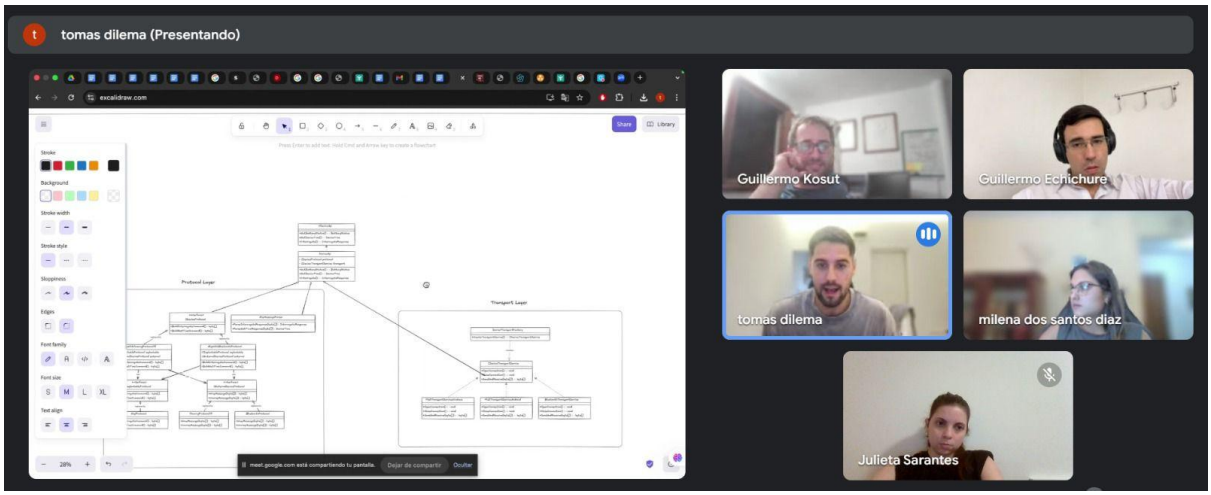
12.4.9. Reunión validación arquitectura *release* 1



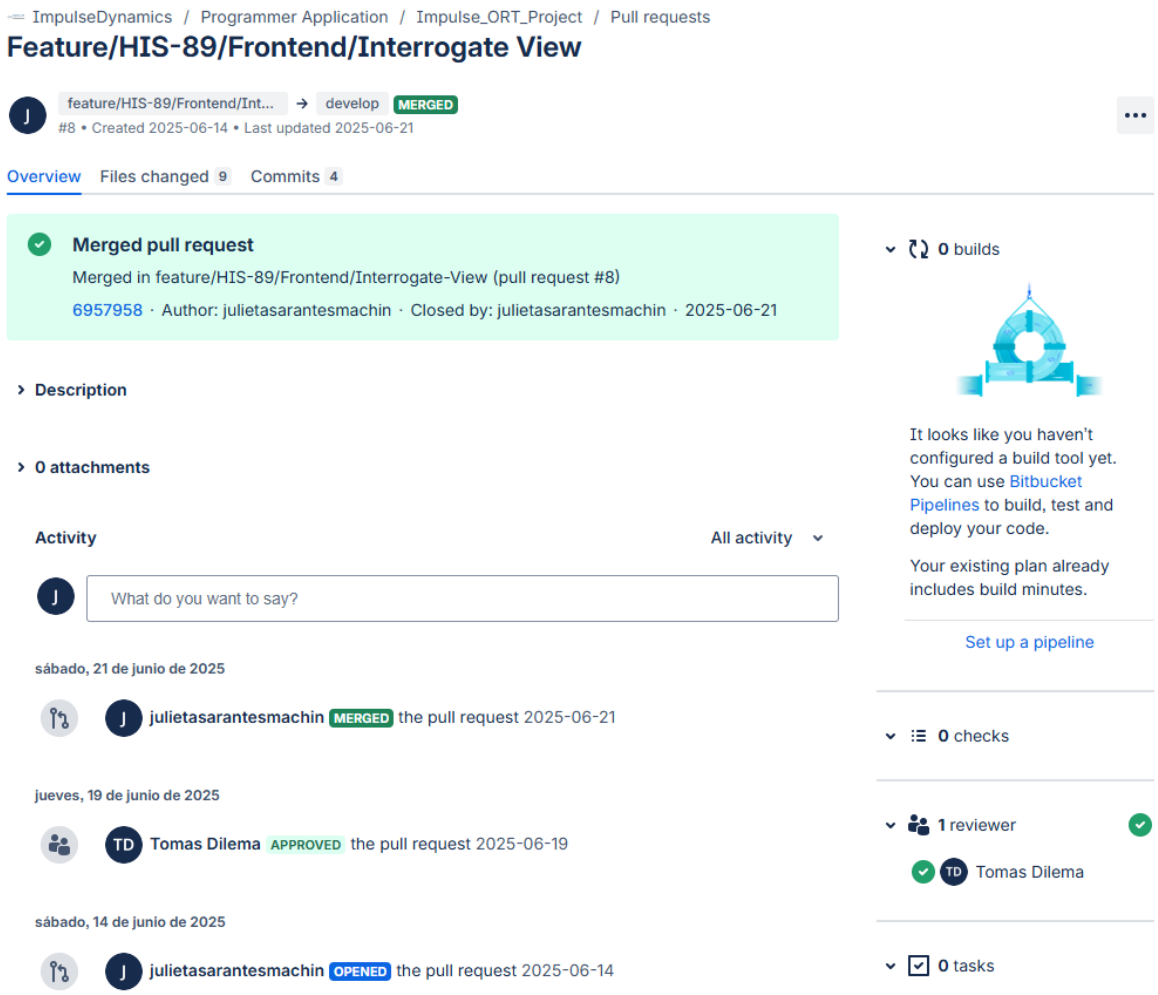
12.4.10. Ejemplo revisión diagrama secuencia



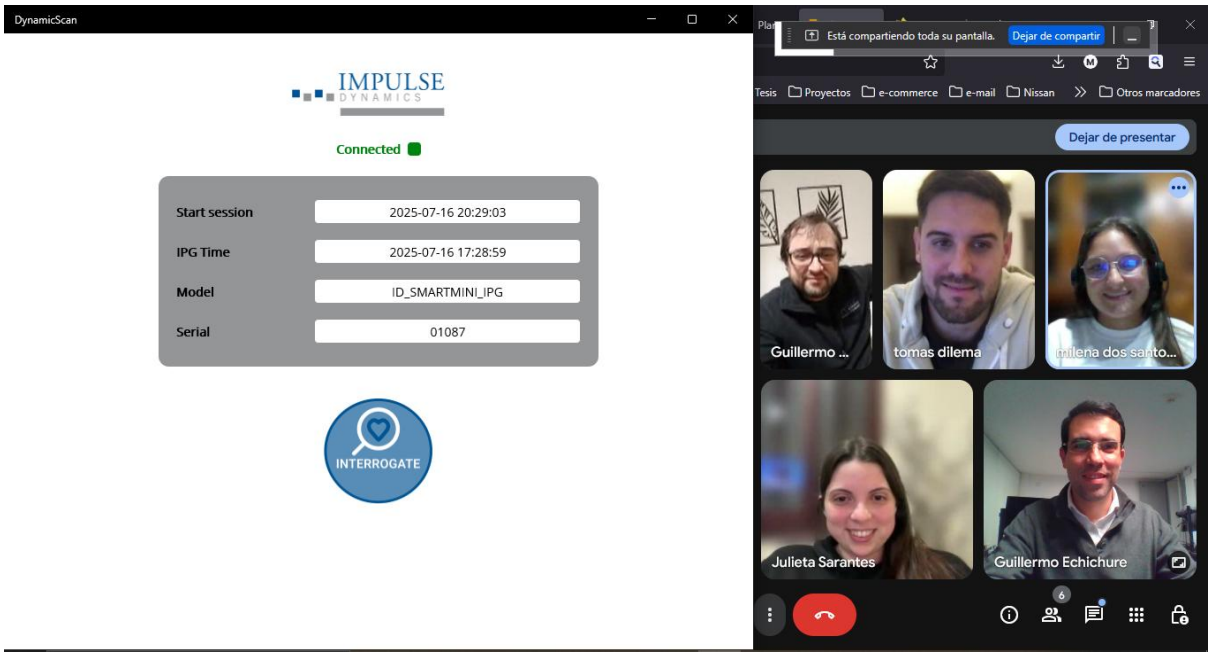
12.4.11. Reunión validación de diseño detallado



12.4.12. Detalle de un *pull request*



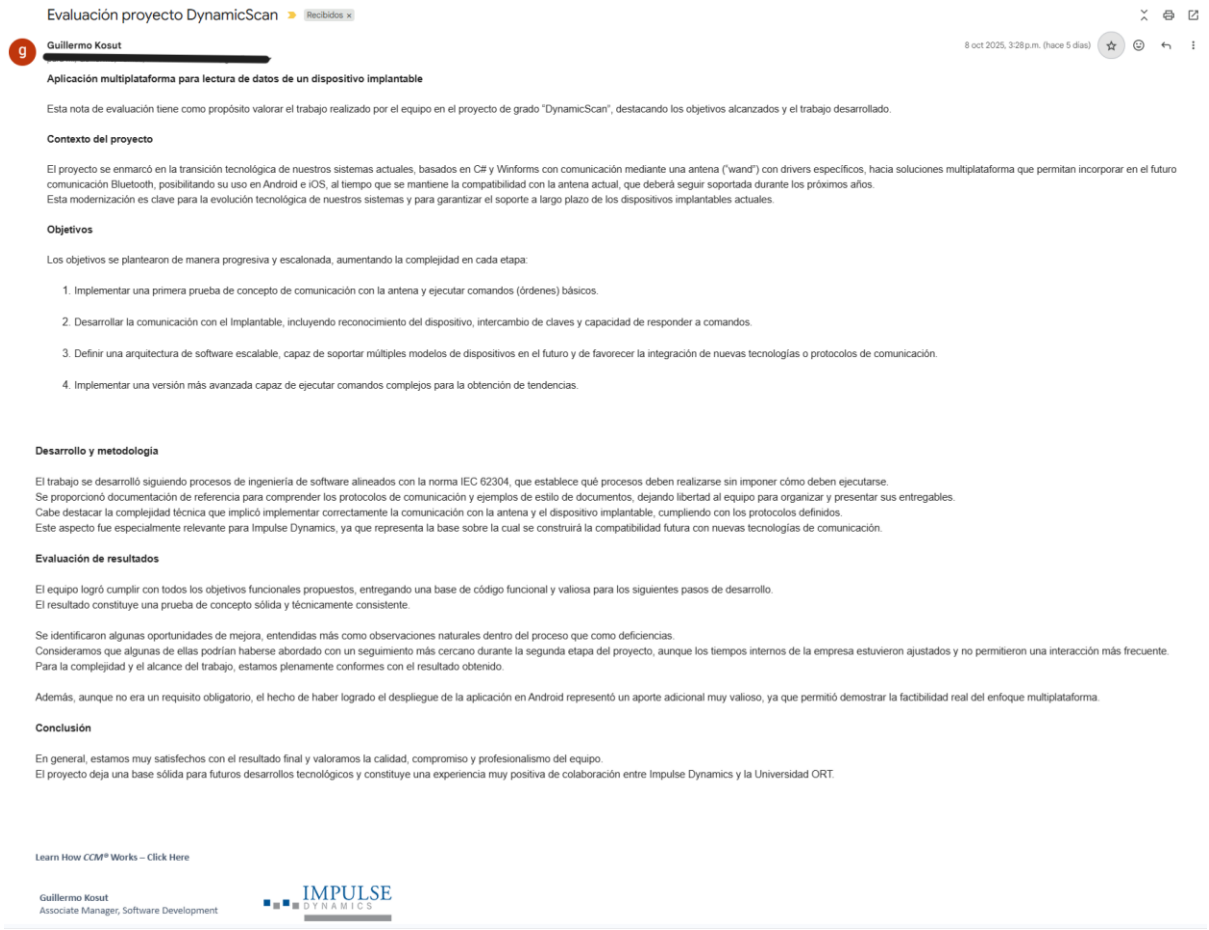
12.4.13. Evidencias de validación con el cliente - *release* uno



12.4.14. Evidencias de validación con el cliente - *release* final

12.4.15. Feedback informe PDF generado

12.4.16. Correo de evaluación del cliente



Evaluación proyecto DynamicScan Recibidos ×

Guillermo Kosut [redacted]

8 oct 2025, 3:28 p.m. (hace 5 días)

Aplicación multiplataforma para lectura de datos de un dispositivo implantable

Esta nota de evaluación tiene como propósito valorar el trabajo realizado por el equipo en el proyecto de grado "DynamicScan", destacando los objetivos alcanzados y el trabajo desarrollado.

Contexto del proyecto

El proyecto se enmarcó en la transición tecnológica de nuestros sistemas actuales, basados en C# y Winforms con comunicación mediante una antena ("wand") con drivers específicos, hacia soluciones multiplataforma que permitan incorporar en el futuro comunicación Bluetooth, posibilitando su uso en Android e iOS, al tiempo que se mantiene la compatibilidad con la antena actual, que deberá seguir soportada durante los próximos años.
Esta modernización es clave para la evolución tecnológica de nuestros sistemas y para garantizar el soporte a largo plazo de los dispositivos implantables actuales.

Objetivos

Los objetivos se plantearon de manera progresiva y escalonada, aumentando la complejidad en cada etapa:

1. Implementar una primera prueba de concepto de comunicación con la antena y ejecutar comandos (órdenes) básicos.
2. Desarrollar la comunicación con el Implantable, incluyendo reconocimiento del dispositivo, intercambio de claves y capacidad de responder a comandos.
3. Definir una arquitectura de software escalable, capaz de soportar múltiples modelos de dispositivos en el futuro y de favorecer la integración de nuevas tecnologías o protocolos de comunicación.
4. Implementar una versión más avanzada capaz de ejecutar comandos complejos para la obtención de tendencias.

Desarrollo y metodología

El trabajo se desarrolló siguiendo procesos de ingeniería de software alineados con la norma IEC 62304, que establece qué procesos deben realizarse sin imponer cómo deben ejecutarse.
Se proporcionó documentación de referencia para comprender los protocolos de comunicación y ejemplos de estilo de documentos, dejando libertad al equipo para organizar y presentar sus entregables.
Cabe destacar la complejidad técnica que implicó implementar correctamente la comunicación con la antena y el dispositivo implantable, cumpliendo con los protocolos definidos.
Este aspecto fue especialmente relevante para Impulse Dynamics, ya que representa la base sobre la cual se construirá la compatibilidad futura con nuevas tecnologías de comunicación.

Evaluación de resultados

El equipo logró cumplir con todos los objetivos funcionales propuestos, entregando una base de código funcional y valiosa para los siguientes pasos de desarrollo.
El resultado constituye una prueba de concepto sólida y técnicamente consistente.

Se identificaron algunas oportunidades de mejora, entendidas más como observaciones naturales dentro del proceso que como deficiencias.
Consideramos que algunas de ellas podrían haberse abordado con un seguimiento más cercano durante la segunda etapa del proyecto, aunque los tiempos internos de la empresa estuvieron ajustados y no permitieron una interacción más frecuente.
Para la complejidad y el alcance del trabajo, estamos plenamente conformes con el resultado obtenido.

Además, aunque no era un requisito obligatorio, el hecho de haber logrado el despliegue de la aplicación en Android representó un aporte adicional muy valioso, ya que permitió demostrar la factibilidad real del enfoque multiplataforma.

Conclusión

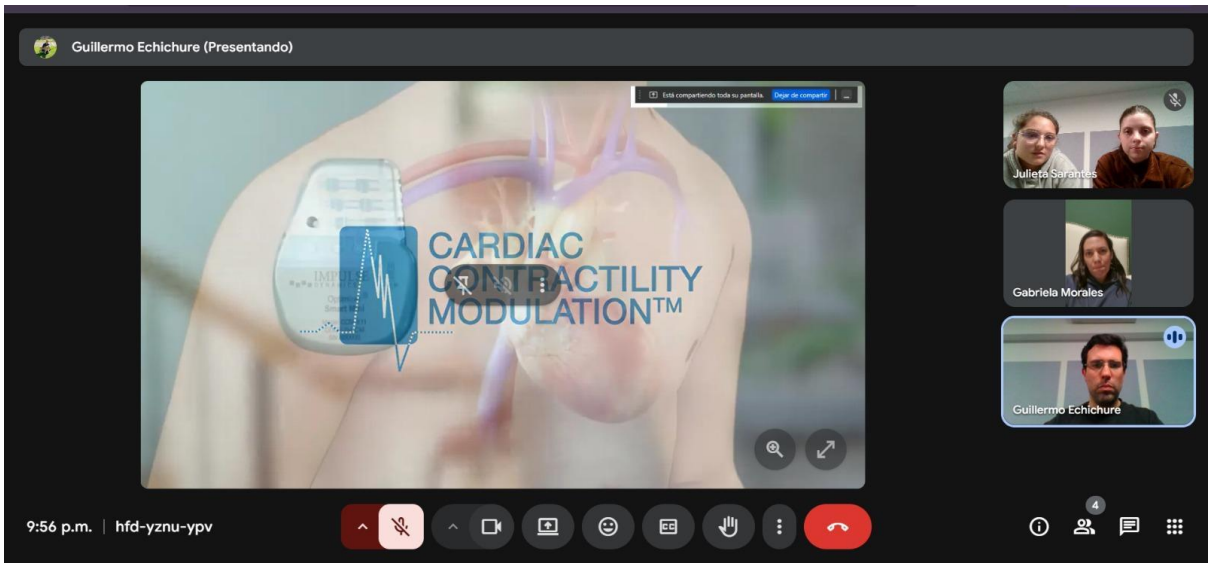
En general, estamos muy satisfechos con el resultado final y valoramos la calidad, compromiso y profesionalismo del equipo.
El proyecto deja una base sólida para futuros desarrollos tecnológicos y constituye una experiencia muy positiva de colaboración entre Impulse Dynamics y la Universidad ORT.

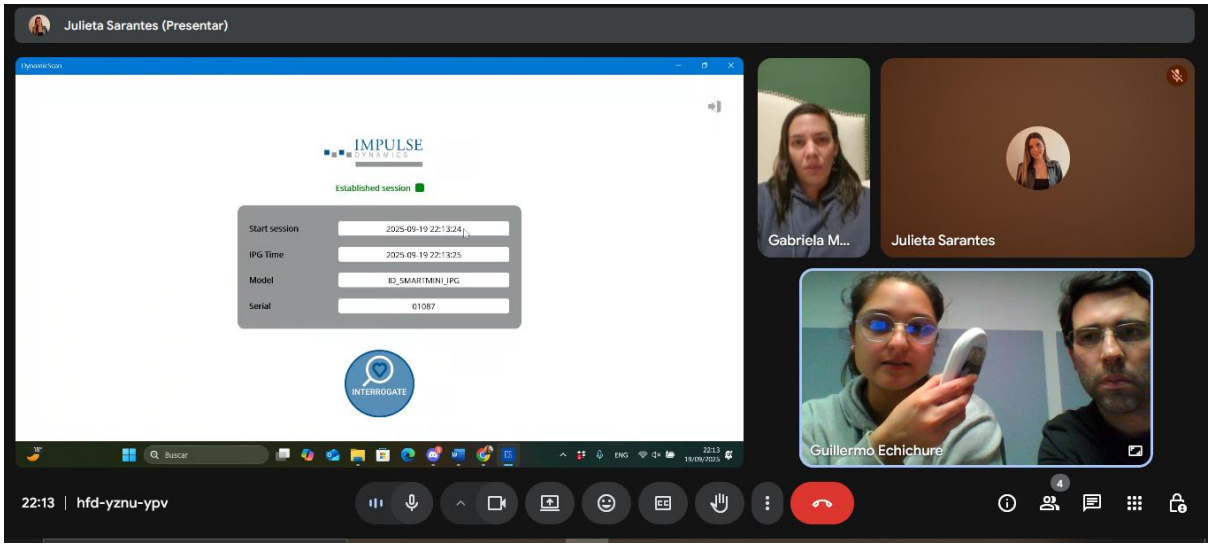
Learn How *CCM*® Works – Click Here

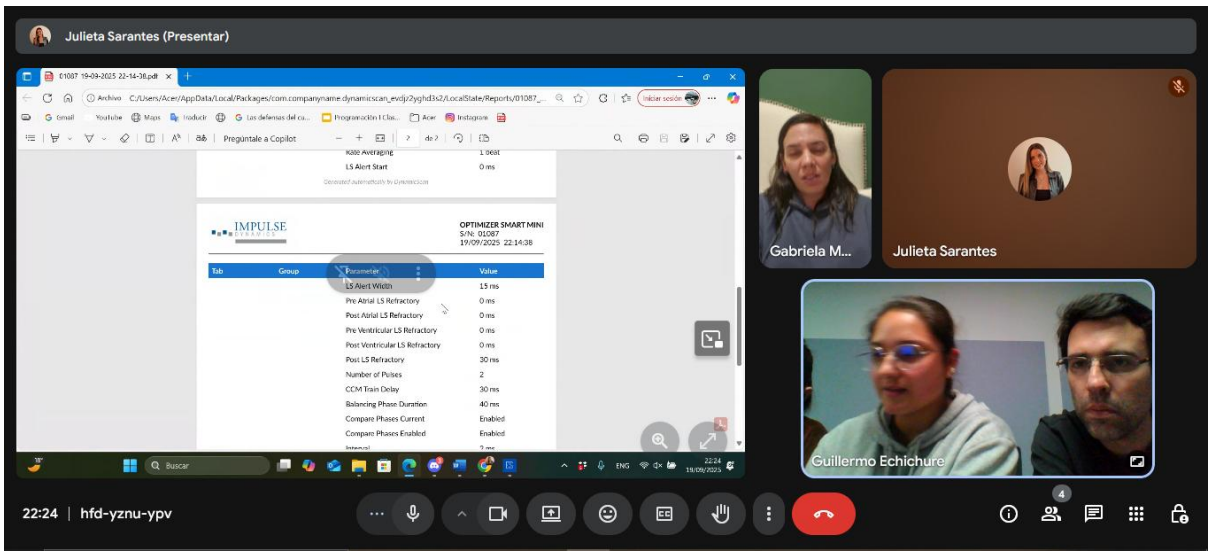
Guillermo Kosut
Associate Manager, Software Development

IMPULSE DYNAMICS

12.4.17. Capturas de la sesión de validación con Dra. Morales







12.4.18. Minuta de reunión con Dr. Chaia

Parte 1:

	Validación con Dr. Chaia - cardiólogo	
	Milena dos Santos	
	Fecha 17/09/25	**Versión -**

Comenzando con una introducción a la cantidad de hospitalizaciones y diagnósticos de IC diariamente, es una enfermedad lamentablemente común donde muchas personas son diagnosticadas diariamente con IC. Muchas enfermedades pueden derivar en esto.

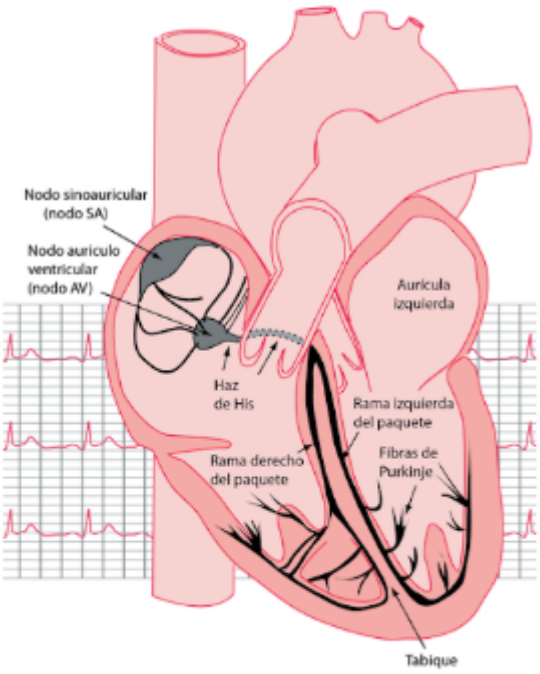
El dispositivo *Optimizer Smart Mini*, ayuda a reducir la disnea, la fatiga y aumenta la tolerancia al esfuerzo, lo que claramente aumenta la calidad de vida de las personas, y como consecuencia de esto disminuye la hospitalización, las internaciones, lo que económicamente es un logro, ya que por cada internación del paciente, se necesitan enfermeros, una junta médica, una cama de hospital disponible y sus aparatos que lo rodean y son controlados por el personal médico, los medicamentos y operaciones en caso de ser necesarias, todo esto incluye un gasto económico muy grande y si logramos que este dispositivo reduzca las hospitalizaciones, ya que administra la terapia completa y es más difícil que los pacientes no la cumplan, lo que reducen las hospitalizaciones y mejora económicamente.

La función cardíaca se mide de muchas maneras. La más práctica y al alcance del médico en la cama del paciente, es la que se determina con un estudio llamado ecocardiograma. La FEVI(fracción de eyección del ventrículo izquierdo) es lo que representa la función cardíaca. Tiene un rango de normalidad en el estudio que va del 50 al 65%. Por debajo del 50%, se dice que la insuficiencia cardíaca es con función o FEVI reducida, que generalmente son las más sintomáticas y graves. En este tipo de insuficiencia, tendría utilidad el tratamiento con el implante del dispositivo de modulación , en conjunto con el resto del tratamiento óptimo con medicamentos. El otro tipo de insuficiencia cardíaca, es el que tiene función o FEVI conservada, que tiene otro encare en el tratamiento y no necesita el dispositivo.

Parte 2:

¿Cómo funciona el dispositivo?

Cuando la persona está con insuficiencia cardiaca, lo que está fallando son los impulsos eléctricos que se generan en el nódulo sinoauricular dentro de la aurícula derecha, esta va con ramificaciones hacia los ventrículos derecho e izquierdo, cuando esto no funciona, es porque en algún lugar se pierde electricidad y es ahí donde entra en juego el dispositivo implantable, donde dos cables van al ventrículo derecho, uno de ellos en la "punta" del corazón para que logre enviar electricidad a ambos ventrículos.



Parte 3:

Información del PDF que es importante para el cardiólogo

Sensing		
	Atrial Sensitivity	1 mV
	Atrial Sensing Polarity	Bipolar
	Ventricle 1 Sensitivity	2 mV
	Ventricle 1 Sensitivity Polarity	Bipolar
	Ventricle 2 Sensitivity	2 mV
	Ventricle 2 Sensitivity Polarity	Bipolar
Others		
	Atrial Refractory	249,4 ms
	Ventricular Refactory	249,4 ms
	Short AV	70 ms
	Long AV	397 ms
	Rate Averaging	1 beat
	LS Alert Start	0 ms

Generated automatically by DynamicScan

1 - Sensibilidad en miliVoltios de la aurícula, es la fuerza de la electricidad que se envía la aurícula, teniendo en cuenta la resistencia que este puede llegar a tener, a mayor resistencia, mayor cantidad de miliVoltios.

2. La polaridad se refiere a los tipos de carga eléctrica, positiva y negativa, que se intercambian en las células durante la transmisión de un impulso. Este cambio de polaridad es lo que permite que el músculo se contraiga y luego vuelva a su estado normal.

3. Sensibilidad del cable 1 al ventrículo

4. Polaridad del mismo

5. Sensibilidad del cable 2 al ventrículo

6. Polaridad del mismo

7. Periodo refractario de la aurícula, esto quiere decir que el músculo está en su momento de "descontracturación", donde si llega algún impulso eléctrico el no lo

Parte 4:

acepta, por eso es refractario, lo devuelve, ya que está en su momento de descanso, que dura unas milésimas más que el periodo de contracción del corazón (momento donde se contrae y bombea la sangre a todo el cuerpo). En el periodo de refracción, el músculo no acepta impulsos eléctricos sino que los rechaza.

8. Lo mismo que lo anterior pero del ventrículo

Nodo AV

El nodo auriculoventricular es un conjunto de células especializadas que se encuentran en la parte inferior del tabique interauricular, justo entre las aurículas y los ventrículos.

Su función principal es recibir el impulso eléctrico proveniente del nodo sinoauricular (SA) y transmitirlo hacia los ventrículos, pero con un ligero retraso controlado.

Los parámetros Short AV y Long AV se refieren a tiempos de conducción auriculoventricular (AV) programables, que controlan cuánto tiempo pasa entre una contracción auricular y una contracción ventricular.

En un corazón normal, el intervalo AV representa el tiempo que tarda el impulso eléctrico en pasar desde el nodo SA (aurícula) hasta el nodo AV y de ahí a los ventrículos.

En los dispositivos, este intervalo se puede programar para definir cuánto espera el aparato entre detectar una señal auricular y estimular el ventrículo (si es necesario).

Short AV (AV corto)

Es el intervalo AV más corto que el dispositivo permite entre la detección auricular y la estimulación ventricular.

Se utiliza, por ejemplo, cuando el dispositivo detecta una frecuencia auricular elevada (como durante el ejercicio).

Permite una respuesta más rápida del ventrículo, acortando el tiempo entre la contracción auricular y ventricular para mantener el gasto cardíaco adecuado.

Long AV (AV largo)

Es el intervalo AV máximo o estándar programado.

Se utiliza en situaciones de frecuencia auricular normal o baja, cuando no hay necesidad de acortar la conducción.

Permite que la contracción auricular se complete y que los ventrículos se llenen bien antes de contraerse.

En general me parece un excelente trabajo, entiendo falta un tiempo para que esto llegue a Uruguay, pero me parece excelente que un médico pueda obtener información, sin tener que esperar a un técnico, dándonos la posibilidad de poder actuar rápidamente ante una emergencia.

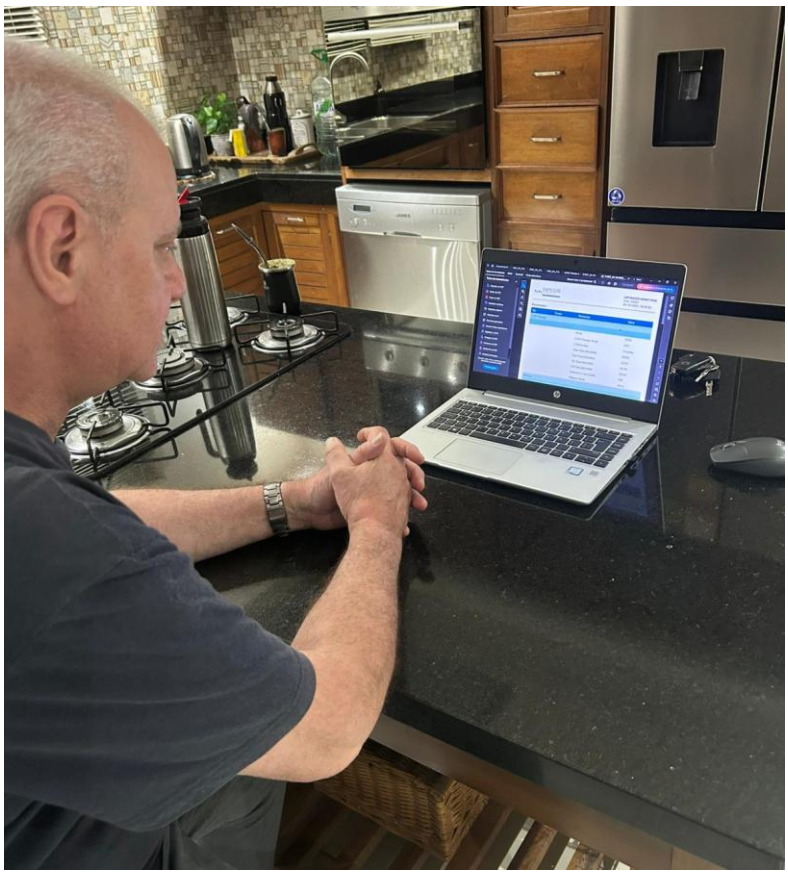
Este dispositivo es capaz de aparte de mejorar la calidad de vida de los pacientes, haciendo que puedan volver a realizar actividades cotidianas, como reuniones familiares o con amigos, donde no solo mejora eso, sino la economía, ya que al reducir las hospitalizaciones estamos ahorrando mucho dinero.

La información que brinda el PDF y el resumen avanzado, es la información que debemos tener al alcance para sacar las principales conclusiones del paciente.

Dato: En un minuto pasa toda la sangre del cuerpo por el corazón

---------- FIN MINUTA ----------

12.4.19. Foto reunión Dr. Chaia



12.4.20. Proceso de seguimiento de problemas

Actividad	Descripción	Herramienta/Técnica	Responsable	Salida esperable
Identificación del problema	Detección de desviaciones, errores, omisiones o vulnerabilidades en cualquier entregable o funcionalidad.	Revisión, Verificación, Pruebas, Análisis	Todo el equipo, Líder de SQA	Problema comunicado al equipo para su registro y seguimiento.
Clasificación y registro	Registro del problema con su respectiva criticidad.	Google Sheets	Líder de SQA	Incidente documentado en Google Sheets.
Problema en Jira	Creación del problema con su correspondiente tipo (*bug o tech-debt)*.	Jira	Equipo	Problema creado en Jira.
Asignación y seguimiento	Asignación del problema a un responsable técnico y seguimiento hasta su resolución.	Jira (asignación y respectivo workflow)	Equipo	Incidente en su respectivo estado; "*ToDo*", "*Doing*", "*Code Review*" y "*Done*"
Origen del problema	Documentación de la forma en la que fue encontrado.	Jira - descripción	Responsable asignado	Comentario del origen del problema.
Resolución y corrección	Aplicación técnica o documental que corrija el incidente.	Código, documentos.	Responsable asignado	Corrección del *issue* en rama, ejemplo: "BUG/HIS-xxx<nombre-del-problema>"
Verificación de solución	Confirmación de que el problema se resolvió correctamente y no generó nuevos problemas.	Revisión cruzada, validación.	Líder de SQA	Problema en "*Done*"

12.4.21. Matriz de trazabilidad de problemas

Matriz de trazabilidad de Issues				
Nombre	**Tipo de Issue**	**Descripción**	**Estado**	**Criticidad**
BUG-01	Bug	"*Device connected*" aparece al conectar cualquier dispositivo USB	Cerrado	Alta
BUG-03	Bug	No tenemos una alerta cuando hay más de una hora de diferencia entre la hora local y la hora del smart mini. El bug surgió de la reunión con el cliente.	Cerrado	Alta
BUG-04	Bug	Actualmente el parseo del Get IPG Time no hace la validación del ACK con la función designada a eso, sino que la hace todo en el mismo parseo.	Cerrado	Alta
BUG-05	Bug	Mensaje incorrecto habla del dispositivo IPG y debe ser al Wand	Cerrado	Media
BUG-06	Bug	Mensaje de alerta al intentar establecer conexion con el IPG y la Wand esta desconectada debe ser en ingles	Cerrado	Baja
BUG-07	Bug	El estado en la UI para la antena Wand conectada o desconectada refiere a un device connected device disconnected, debe ser Wand.	Cerrado	Media
BUG-09	Bug	Manejo de error de Wand desconecta trasnversal a los metodos	Cerrado	Alta
BUG10	Bug	En el tab summary cuando no ha recibido la respuesta total del IPG los datos se muestran a medida que van llegando. No se espera a tener la totalidad de los datos	Cerrado	Alta
BUG11	Bug	No se detecta a nivel de la IU dispositivo IPG no compatible	Cerrado	Media
BUG12	Bug	No se detecta a nivel de UI la desonexion del IPG	Cerrado	Alta
BUG13	Bug	No se controla desconexion de Wand en todos los cursos alternativos	Cerrado	Alta
BUG14	Bug	Nombres en español en la UI	Cerrado	Alta
BUG15	Bug	Logs en español	Cerrado	Baja
TD-01	Tech Debt	Warnings	Abierto	Baja
TD-02	Tech Debt	Clean Code	Cerrado	Media
TD-03	Tech Debt	WriteCommandAsync y ReadResponseAsync ignoran el CancellationToken recibido	Cerrado	Alta
TD-04	Tech Debt	Spinners de carga	Cerrado	Media
TD-05	Tech Debt	Spinner de carga en pantalla Summary	Cerrado	Media
TD-06	Tech Debt	Agregrar response al final de los metodos que no lo tienen	Cerrado	Baja
TD-07	Tech Debt	Crear interfaz SessionStateService	Cerrado	Media
TD-09	Tech Debt	Ordenar los informes historicos ultima fecha arriba	Cerrado	Baja
TD-10	Tech Debt	Agregar scroll en la mainView	Cerrado	Baja

12.4.22. Matriz probabilidad de ocurrencia del daño suministrada por el cliente

PROBABILIDAD DE OCURRENCIA DEL DAÑO			
Rango	**Probabilidad Cualitativa**	**Probabilidad Cuantitativa**	**% Probabilidad**
1	Improbable	< 1 en 1.000.000 (Extremadamente improbable – Nunca se espera que ocurra durante la vida del sistema)	< 0,0001%
2	Remota	< 1 en 100.000 y ≥ 1 en 1.000.000 (Remota – Probable que ocurra menos de una vez en la vida del sistema)	< 0,001% y ≥ 0,0001%
3	Ocasional	< 1 en 10.000 y ≥ 1 en 100.000 (Ocasional – Probable que ocurra menos de una vez al año)	< 0,01% y ≥ 0,001%
4	Probable	< 1 en 1.000 y ≥ 1 en 10.000 (Probable – Puede ocurrir solo unas pocas veces al año por sistema)	< 0,1% y ≥ 0,01%
5	Frecuente	≥ 1 en 1.000 (Frecuente – Probable que ocurra muchas veces al año por sistema)	≥ 0,1%

12.4.23. Matriz de severidad del daño suministrada por el cliente

SEVERIDAD		
Rango	**Descriptor**	**Efecto**
1	Despreciable	Inconveniente o molestia temporal
2	Menor	Provoca lesión o deterioro temporal que no requiere intervención médica
3	Grave	Provoca lesión o deterioro que requiere intervención médica profesional.NOTA: La extracción de un IPG por cualquier motivo se considera Grave.
4	Crítico	Provoca deterioro permanente o lesión potencialmente mortal
5	Catastrófico	Provoca la muerte del paciente

12.4.24. Matriz criterio de riesgos suministrada por el cliente

CRITERIO DE RIESGOS					
	1 Despreciable	**2 Menor**	**3 Grave**	**4 Crítico**	**5 Catastrófico**
5 Frecuente	No aceptable	No aceptable	No aceptable	No aceptable	No aceptable
4 Probable	No aceptable	No aceptable	No aceptable	No aceptable	No aceptable
3 Ocasional	Aceptable	Aceptable	No aceptable	No aceptable	No aceptable
2 Remota	Aceptable	Aceptable	Aceptable	No aceptable	No aceptable
1 Improbable	Aceptable	Aceptable	Aceptable	Aceptable	Aceptable

12.4.25. Detalles de la ejecución de las pruebas y motivos de priorización

***Release* 1 - Comienzo por integración**
La ejecución de pruebas para el *Release* 1 comenzó con las de integración entre el módulo *TransportLayer* (que contiene la clase *Ftdi*Windows*TransportService*), el *driver* de terceros provisto como *assembly* (FTD2XX_NET.dll) y el *hardware* físico (la antena Wand).

Motivos de la priorización:

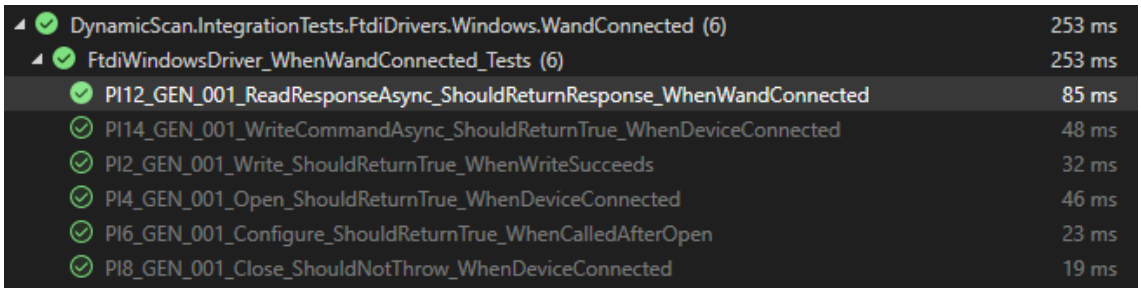
- Riesgo (**RProd-5-000: FTD2XX_NET es un componente de terceros**). Al ser un componente que no controlamos, priorizamos validar cuanto antes la integración real módulo<->*driver*<->Wand.
- Imposibilidad de probar unitariamente las operaciones del *driver*.

Se diseñaron escenarios observables diferenciado:

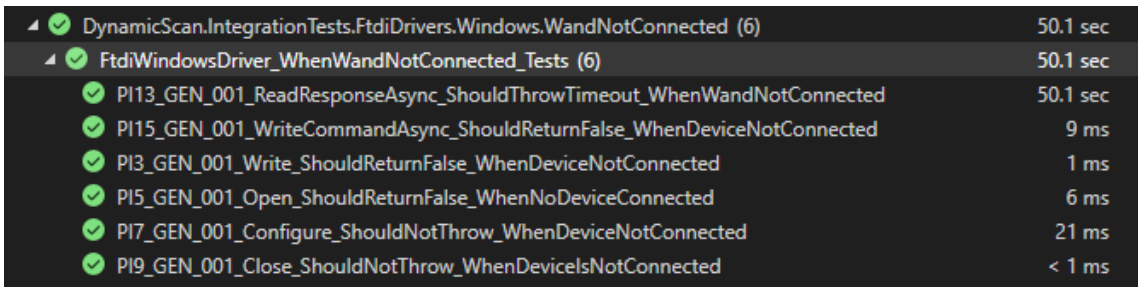
- Wand conectada.
- Wand desconectada.

Los métodos validados incluyeron *Open(), Configure(), Write(), WriteCommandAsync(), ResdResponseAsync().*

Allí la ejecución se organizó de manera secuencial para los escenarios de Wand conectada ya que los métodos del transporte (*Open, Configure, Write, WriteCommandAsync, ReadResponseAsync, Close*) son operaciones básicas del módulo y resultan estado-dependientes. Todas comparten un recurso físico único (la Wand a través del *driver* FTD2XX_NET.dll), por lo que correr pruebas en paralelo generaba condiciones de carrera (por ejemplo, una prueba cerraba la conexión mientras otra intentaba leer o escribir) que terminaban en resultados inconsistentes y distintos al objetivo de la integración. Dado que el escenario real es un único dispositivo ejecutando una operación a la vez, la ejecución secuencial refleja mejor el uso normal y evita introducir concurrencia artificial que no representa el escenario real. Además, el objetivo no era probar concurrencia del *driver* ni simular múltiples antenas Wand, sino validar la integración normal con un único dispositivo tal como se usará.



Para el caso de Wand desconectada, no se presentó este problema.

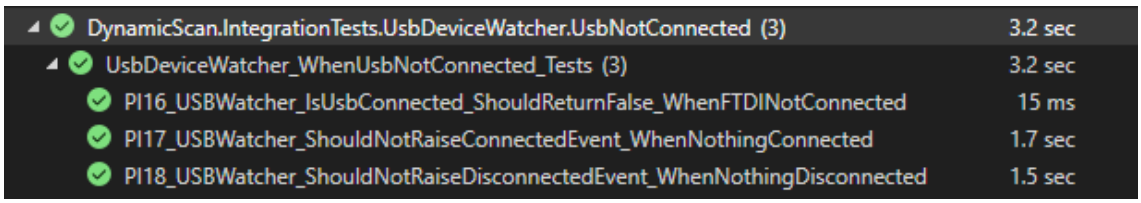


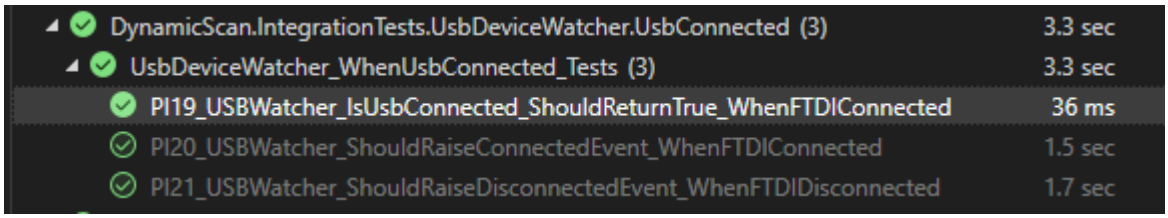
En esos escenarios, los métodos retornan valores controlados o comportamientos coherentes según su diseño:

- *Open() y Write()* devuelven *false*, indicando que no había dispositivo disponible.
- *ReadResponseAsync*() finaliza por cancelación/*timeout* al no recibir datos.
- *Configure*() devuelve true, por que aplica parámetros sobre el objeto FTDI sin validar conexión en ese punto.
- *Close*() no lanza excepción (cerrar sobre un recurso conectado/desconectado es tolerado por diseño).

Luego para la clase encargada de detectar la presencia de un dispositivo USB conectado se plantearon dos escenarios posibles:

UsbDeviceWatcher con USB desconectado:





Aquí, las pruebas se diseñaron para reflejar un escenario mixto: el *Tester* conecta y desconecta físicamente la Wand durante la ejecución. El método *IsUsbConnected()* consulta en tiempo real al *driver* mediante *GetNumberOfDevices ()* y retorna true o false según la presencia del dispositivo. De esta forma se validó que el sistema detecta y notifica correctamente los eventos de conexión y desconexión, reproduciendo el escenario real.

Pruebas unitarias:

- Se realizaron prueba unitaria para los requerimientos trabajados hasta el momento: Establecer conexión con el IPG, Detalles del IPG, Obtener la hora del IPG, Mostrar *Splash Screen* y Detectar antena Wand.

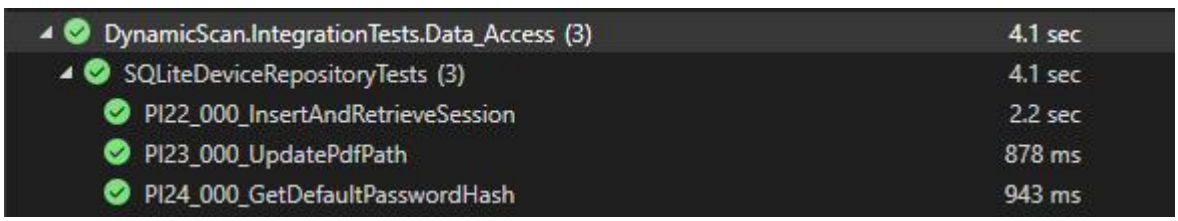
Pruebas de sistema:

- Cinco pruebas de sistema: PS1-000, PS2-000, PS3-000, PS4-000, PS5-000.
- El registro de todas las pruebas (unitarias, de integración y de sistema) se realizó en la Matriz de ejecución y resultados y se mantuvo sincronizado con la Matriz de Trazabilidad integral.

Release final

Al igual que en el *Release* 1, comenzamos con pruebas de integración para trabajar el riesgo identificado en la Matriz de riesgos del producto.

RProd-4-000-Riesgo de errores funcionales en el acceso a la base de datos y se llevaron adelante pruebas de integración contra la base de datos SQLite.



En este conjunto de pruebas se verificó el correcto funcionamiento del repositorio local implementado con SQLite, encargado de gestionar la persistencia de las sesiones establecidas con el dispositivo. Los métodos fueron evaluados de forma aislada utilizando una base de datos temporal creada en disco para cada ejecución, asegurando la independencia entre pruebas.

En estos escenarios, los métodos retornaron valores y comportamientos coherentes con su diseño:

- *AddSessionAsync*() insertó correctamente los registros y devolvió un identificador autoincremental válido.
- *GetAllSessionsAsync*() recuperó los datos almacenados, incluyendo los campos de modelo, número de serie y estado de conexión.
- *UpdatePdfPathAsync*() modificó el campo correspondiente sin afectar la integridad del resto de los datos.
- *GetAdvancedModePasswordHashAsync*() devolvió el hash preconfigurado para el modo avanzado, confirmando la correcta inicialización del esquema y la configuración del valor por defecto.

De esta forma se comprobó la correcta operación de las funciones de lectura, escritura y actualización sobre la base de datos, validando la coherencia de la capa de acceso a datos y su integración con la configuración definida en el sistema. A su vez se siguió trabajando en pruebas unitarias y de sistema. El registro de todas las pruebas (unitarias, de integración y de sistema) se realizó en la Matriz de ejecución y resultados y se mantuvo sincronizado con la Matriz de trazabilidad.

12.4.26. Matriz de ejecución y resultados de pruebas

Esta matriz surge como una necesidad de ampliar el registro de información sobre las pruebas (unitarias, de integración y de sistema) sin sobrecargar la Matriz de trazabilidad Integral con datos operativos. Su propósito es detallar los resultados obtenidos durante la ejecución, manteniendo a la vez la trazabilidad con los riesgos del producto relevados en la Matriz de riesgos del producto.

Aquí se comparte los datos más relevantes de las columnas:

- Id-Prueba

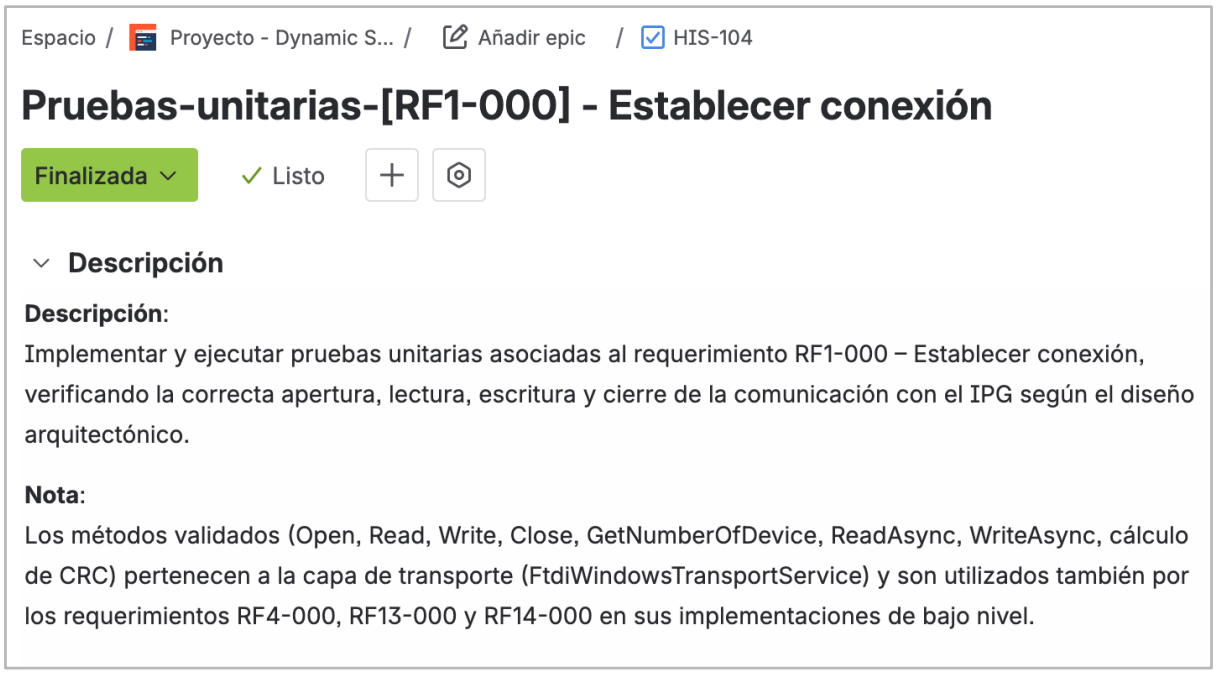
- La tarjeta JIRA donde fue trabajada esa prueba.
- Título de la prueba.
- Los requerimientos funcionales (en capas superiores la trazabilidad es un RF a una prueba, mientras que, en capas inferiores, más de un RF queda mapeado con una prueba por ser código común).
- Los casos de uso asociados (CU) o Issues asociados (en caso de tener).
- Los riesgos de producto vinculados (en caso de tener).
- En la sección ejecución: resultado esperado, resultado obtenido y un campo para observaciones.

Ejemplo de trazabilidad de pruebas unitarias:

Matriz de Ejecución y Resultados de Pruebas - Versión 2.0									
Trazabilidad							Ejecución		
ID de prueba	Tarjeta JIRA vinculada	Nombre de la prueba	Tipo de prueba	Requerimiento vinculado	Casos de Uso o Issues asociados	Riesgo vinculado	Resultado esperado	Resultado obtenido	Observaciones
PU14-000	HIS-104	ParseStartSessionResponse_Should_ThrowInvalidResponseException_WhenFrameListIsEmpty	Unitaria	RF1-000	CU1-000	RProd-7-000	Excepción del tipo InvalidResponseException	Se lanza InvalidResponseException	Cubre escenario mínimo de respuesta vacía. Vinculado parcialmente a RProd-007.
PU15-000	HIS-104	ParseStartSessionResponse_Should_ThrowInvalidResponseException_WhenFrameIsTooShort	Unitaria	RF1-000	CU1-000	RProd-7-000	Excepción del tipo InvalidResponseException	Se lanza InvalidResponseException	Controla frames truncados que podrían dar lugar a interpretaciones erróneas. Cobertura directa del riesgo RProd-007.
PU16-000	HIS-104	LogStart_Should_NotThrowException_WhenCalled	Unitaria	RF1-000	CU1-000	-	No debe lanzar excepción	No lanza excepción	Verifica estabilidad de la función Logger.LogStart.
PU17-000	HIS-104	LogEnd_Should_NotThrowException_WhenCalled	Unitaria	RF1-000	CU1-000	-	No debe lanzar excepción	No lanza excepción	Verifica estabilidad de la función Logger.LogEnd.
PU18-000	HIS-104	LogError_Should_NotThrowException_WhenCalledWithValidException	Unitaria	RF1-000	CU1-000	-	No debe lanzar excepción	No lanza excepción	Verifica que la función Logger.LogError() maneje correctamente excepciones anidadas.
PU19-000	HIS-104	PrintByteArray_Should_NotThrowException_WhenCalledWithValidInput	Unitaria	RF1-000	CU1-000	-	No debe lanzar excepción	No lanza excepción	Verifica que la funcion GenericUtils.PrintByteArray se ejecute correctamente sin lanzar excepcion.
PU20-000	HIS-104	SplitFp17Frames_Should_ReturnThreeFrames_WhenMultipleValidFramesProvided	Unitaria	RF1-000	CU1-000	RProd-7-000	El número de tramas detectadas debe ser 3. El contenido extraído debe coincidir con cada una de las tramas originales. Cada trama debe contener el Header FP17 al inicio , y el Tail FP17 en los bytes finales, el CRC se asume correcto. No es responsabilidad de este método ni de esta prueba verificarlo.	El número de tramas detectadas es 3. El contenido extraído coincide con cada una de las tramas originales. Cada trama contiene el Header FP17 al inicio, y el Tail FP17 en los bytes finales.	Se simulan 3 tramas FP17 válidas. Se agrega ruido (bytes aleatorios) al principio y al final del array para simular una situación real de recepción asincrona o buffer contaminado.
PU21-000	HIS-104	ExtractFP17PayloadSimple_ShouldReturnCorrectPayload	Unitaria	RF1-000	CU1-000	RProd-7-000	Extraer correctamente el payload útil de un frame FP17, omitiendo los primeros bytes (header), los dos últimos Tail FP17 y CRC. El resultado debe coincidir con la sección esperada del payload útil.	Extrae correctamente el payload útil de un frame FP17, omite los primeros bytes (header), los dos últimos Tail FP17 y CRC. El resutado coincide con el payload util.	Verifica la lógica de extracción del contenido de un frame FP17 individual. No se valida el valor del CRC ni la validez del frame completo, ya que no es responsabilidad de este método.
PU22-000	HIS-104	UnWrapMessageFP17_ShouldReturnCorrectPayloads_WhenGivenMultipleValidFrames	Unitaria	RF1-000	CU1-000	RProd-7-000	Una lista de 3 payloads {0x31, 0x00, 0xFF} extraidos correctamente de los frames FP17 válidos. Cada payload debe coincidir con el contenido útil esperado.	Se extrae una lista con tramas FP17 conteniendo dentro 3 payloads . Cada uno coincide con los valores esperados (0x31, 0x00, 0xFF), sin lanzar excepción.	Verifica la extracción del contenido útil desde frames FP17. Cubre casos con datos válidos, con ruido antes y al final del buffer.
PU23-000	HIS-104	GetConnectionType_ShouldReturnUSB_WhenFtdiTransportIsSet	Unitaria	RF1-000	CU1-000	Rpod-14-000	Retornar ConnectionType.USB si el transporte actual en el proxy es una instancia de FtdiWindowsTransportService.	Se retorna correctamente ConnectionType.USB.	Esta prueba requiere acceso al campo _currentTransport, por lo que fue necesario exponerlo temporalmente mediante el método SetCurrentTransportForTesting, declarado como internal. Para permitir su acceso desde el proyecto de pruebas, se utilizó el atributo [InternalsVisibleTo("DynamicScan.UnitTests")] definido en AssemblyInfo.cs, conforme a la documentación oficial de Microsoft.
PU24-000	HIS-104	GetConnectionType_ShouldReturnUnknown_WhenNoTransportIsSet	Unitaria	RF1-000	CU1-000 - Curso Alternativo 1	Rpod-14-000	Retornar ConnectionType.UNKNOW cuando no hay transporte	Se retorna correctamente ConnectionType.UNKNOW	Se evalúa el comportamiento del metodo cuando no se ha establecido ningún medio de transporte, validando que se informe correctamente como tipo de conexión desconocido.
PU25-000	HIS-104	Create_ShouldReturnAlcpWithFP17_WhenModelIsSmartMiniAndConnectionIsUSB	Unitaria	RF1-000	CU1-000	-	Retornar instancia de AlcpWithFP17 si el modelo es ID_SMARTMINI_IPG y la conexión es USB.	Se retorna correctamente instancia de AlcpWithFP17 .	Se verifica correctamente el comportamiento esperado del factory ante combinación soportada.
PU26-000	HIS-104	Create_ShouldThrowNotSupportedException_WhenModelIsUnsupported	Unitaria	RF1-000	CU1-000 - Curso Alternativo 3	RProd-2-000	Lanzar NotSupportedException si el modelo no es ID_SMARTMINI_IPG.	Se lanza correctamente NotSupportedException al usar modelo "MODELO_NO_SOPORTADO"	Verificada robustez ante modelo inválido. Se valida comportamiento esperado del switch del factory.
PU27-000	HIS-104	CreateParser_ShouldReturnAlcpMessageParser_WhenModelIsSmartMini	Unitaria	RF1-000	CU1-000	-	Retornar instancia de AlcpMessageParser si el modelo es ID_SMARTMINI_IPG.	Se retorna correctamente instancia de AlcpMessageParser.	Validación exitosa del parser para el modelo actual soportado.
PU28-000	HIS-104	CreateParser_ShouldThrowNotSupportedException_WhenModelIsUnsupported	Unitaria	RF1-000	CU1-000 - Curso Alternativo 3	RProd-2-000	Lanzar NotSupportedException si el modelo no está definido como válido.	Se lanza correctamente NotSupportedException para modelo "Integra".	Comportamiento ante entrada inválida. Protección frente a uso indebido de un modelo no previsto.

Trazabilidad para **PU26-000**: *CreateParser_ShouldThrowNotSupportedException_WhenModelIsUnsupported*

Tarjeta JIRA: HIS-104



Requerimientos vinculados:

- RF1-000-Establecer conexión

Casos de uso vinculados:

- **CU1 - Curso Alternativo 3 - Conexión a otro IPG distinto de *Smart Mini***

7. Curso Alternativo 3 - Conexión a otro IPG distinto del Smart Mini

a. El usuario conecta un IPG con otro modelo diferente al Smart Mini - El usuario intenta conectarse y DynamicScan muestra una alerta informando: “ *IPG model INTEGRA not compatible detected, please connect an ID_SMARTMINI_IPG*”

Riesgo de producto asociado:

RProd-2-000 - Uso del *software* con un dispositivo IPG no compatible

RProd-2-000	Uso del software con un dispositivo IPG no compatible	El uso de DynamicScan con un modelo de IPG distinto al Smart Mini puede provocar errores en la interpretación de comandos o en la estructura de los datos recibidos, generando comportamientos no esperados.	Ausencia de validación en el tipo de dispositivo conectado; posibilidad de conexión accidental o intencional con un IPG distinto al previsto. Solo aplicable a RF1-000	4	2	NO	Validar modelo de IPG en clase DeviceProtocolFactory con métodos que controlen la versión.	pruebas unitarias: PU26-000, PU28-000.	SI	Diseño detallado

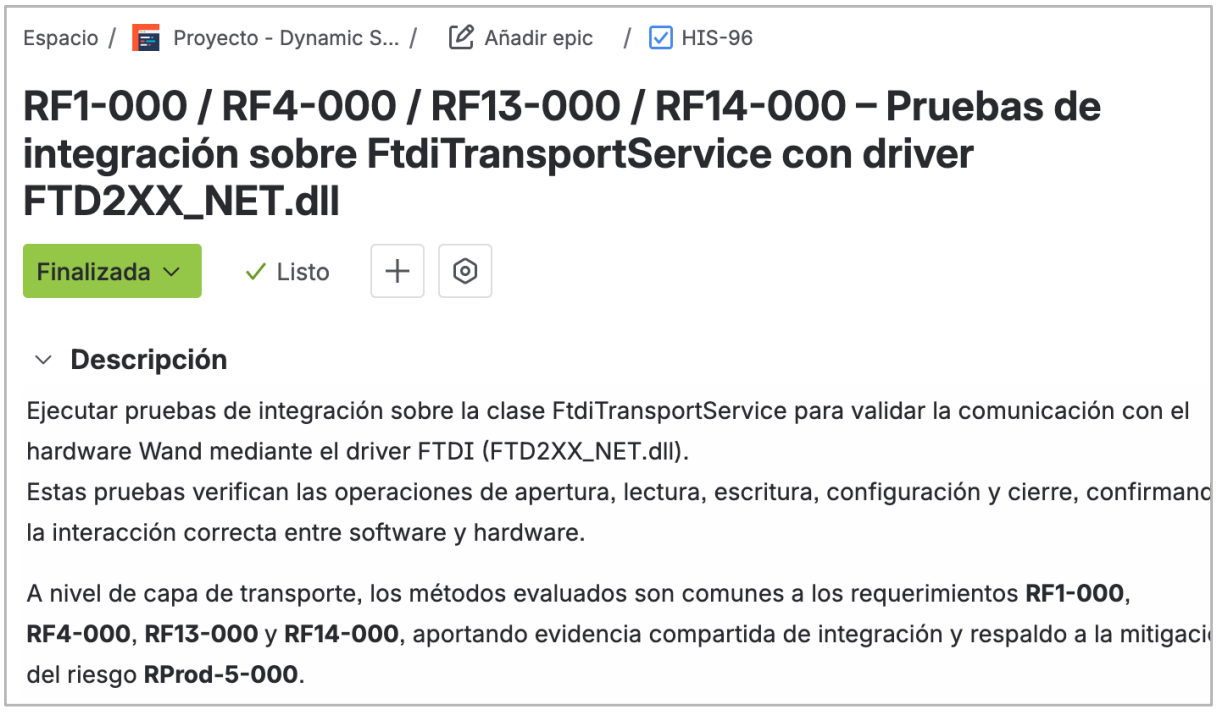
En la fila del riesgo RProd-2-000, columna 9 puede apreciarse como PU26-000 es una de las dos pruebas unitarias que verifican la mitigación (columna 8) del riesgo en la etapa diseño detallado.

Ejemplo de trazabilidad para pruebas de integración:

Matriz de Ejecución y Resultados de Pruebas - Versión 2.0									
Trazabilidad							Ejecución		
ID de prueba	Tarjeta JIRA vinculada	Nombre de la prueba	Tipo de prueba	Requerimiento vinculado	Casos de Uso o Issues asociados	Riesgo vinculado	Resultado esperado	Resultado obtenido	Observaciones
PI2-000	HIS-96	Write_ShouldReturnTrue_WhenWriteSucceeds	Integracion	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método Write debe retornar true cuando logro escribir correctamente en el buffer.	Retorna true.	Valida la escritura correcta en el buffer de datos.
PI3-000	HIS-96	Write_ShouldReturnFalse_WhenDeviceNotConnected	Integracion	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	El método Write debe retornar false si el dispositivo no está conectado	Retorna false.	Validación del flujo alternativo esperable. No se lanza excepción. Control de estado correcto.
PI4-000	HIS-96	Open_ShouldReturnTrue_WhenDeviceConnected	Integracion	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método Open debe retornar true si el dispositivo FTDI está correctamente conectado	Retorna true.	Confirma inicialización y apertura del canal FTDI como se espera.
PI5-000	HIS-96	Open_ShouldReturnFalse_WhenNoDeviceConnected	Integracion	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	El método Open debe retornar false si no hay dispositivo FTDI conectado	Retorna false.	Validación de caso de error esperable. Control de dispositivos FTDI no conectados implementado correctamente.
PI6-000	HIS-96	Configure_ShouldReturnTrue_WhenCalledAfterOpen	Integracion	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método Configure debe retornar true si se ejecuta correctamente	Retorna true.	Validación de configuración FTDI post-conexión. No se detectaron errores al aplicar parámetros operativos.
PI7-000	HIS-96	Configure_ShouldNotThrow_WhenDeviceNotConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	El método Configure() no debe lanzar excepción si no hay dispositivo conectado.	No se lanzó excepción.	Se verifica que la clase maneja el estado desconectado de forma segura.
PI8-000	HIS-96	Close_ShouldNotThrow_WhenDeviceConnected	Integración	RF2-000	CU2-000	RProd-5-000	El método Close debe cerrar la conexión y no debe lanzar excepción si el dispositivo está conectado.	Se cierra la conexión y no se lanzó excepción.	Valida que se cierre correctamente el canal FTDI sin errores de ejecución.
PI9-000	HIS-96	Close_ShouldNotThrow_WhenDeviceIsNotConnected	Integración	RF2-000	CU2-000 - Curso Alternativo 2	RProd-5-000	El método Close no debe lanzar excepción si no hay conexión activa.	No se lanzó excepción.	Prueba el cierre defensivo del recurso en estado desconectado.
PI12-000	HIS-96	ReadResponseAsync_ShouldReturnResponse_WhenDeviceConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método ReadResponseAsync debe devolver una respuesta no vacía con conexión establecida y un mensaje válidos enviado hacia la Wand	Respuesta recibida correctamente.	Esta prueba crea un flujo completo de : open, WriteComandAsync, ReadResponseAsync y Close. Testea el ReadResponseAsync en los Asserts.
PI13-000	HIS-96	ReadResponseAsync_ShouldThrowTimeout_WhenDeviceNotConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	Lanzar TimeoutException cuando no hay conexión activa de la Wand.	TimeoutException correctamente lanzada.	Se prueba el comportamiento esperado del metodo cuando no hay respuesta desde el hardware.
PI14-000	HIS-96	WriteCommandAsync_ShouldReturnTrue_WhenDeviceConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método WriteCommandAsync debe retornar true luego de escribir en el buffer si hay dispositivo conectado.	Escritura exitosa, devuelve true.	Validación de escritura con dispositivo conectado.
PI15-000	HIS-96	WriteCommandAsync_ShouldReturnFalse_WhenDeviceNotConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	Devuelve false si no hay dispositivo conectado.	Escritura fallida, devuelve false.	Validación sin hardware conectado.
PI16-000	HIS-126	USBWatcher_IsUsbConnected_ShouldReturnFalse_WhenFTDINotConnected	Integración	RF17-00	CU17-000	RProd-5-000	El método IsUsbConnected() debe retornar false	El metodo retorna false	Valida que no se detecten dispositivos FTDI inexistentes.
PI17-000	HIS-126	USBWatcher_ShouldNotRaiseConnectedEvent_WhenNothingConnected	Integración	RF17-00	CU17-000	RProd-5-000	No debe invocarse el evento UsbDeviceConnected	No se invoca el evento UsbDeviceConnected	Valida que no haya eventos erróneos cuando no hay acción del usuario.
PI18-000	HIS-126	USBWatcher_ShouldNotRaiseDisconnectedEvent_WhenNothingDisconnected	Integración	RF17-00	CU17-000	RProd-5-000	No debe invocarse el evento UsbDeviceDisconnected	No se invoca el evento UsbDeviceDisconnected	Complementaria a PI17, asegura estabilidad de eventos ante inactividad del puerto USB.

Trazabilidad para **PI3-000**: *Write_ShouldReturnFalse_WhenDeviceNotConnected*

Tarjeta JIRA: HIS-96



Requerimientos vinculados:

- RF1-000-Establecer conexión
- RF4-000-Obtención de datos
- RF13-000-Trends
- RF14-000-Obtener hora del IPG

Casos de uso vinculados:

- CU1-000-Curso Alternativo 1 - Wand desconectada

5. Curso Alternativo 1 - Wand desconectada

a. *DynamicScan* debe detectar la antena Wand desconectada, indicando en la parte inferior de la ventana "*Wand Disconnected*".

b. El usuario intenta conectarse y DynamicScan muestra una alerta informando: *"Wand disconnected! Please connect the Wand before"*.

- CU4-000-Curso Alternativo 1 - Desconexión de Wand antes de interrogar.

5. Curso Alternativo 1 - Desconexión de Wand antes de interrogar

a. El usuario hace click en el botón interrogar.

b. DynamicScan muestra una alerta informando: "*Connection Closed. Antenna WAND Not Found*".

c. *DynamicScan* retorna a la pantalla inicial mostrando en la UI "*Wand Disconnected*" en color rojo.

Riesgo de producto asociado:

RProd-005 - FTDI D2XX es un componente de tercero.

RProd-5-000	FTD2XX es un componente de tercero.	La librería FTD2XX_NET.dll, provista por la empresa FTDI, se utiliza para establecer la comunicación con la antena Wand a través del puerto USB 2.0. Si bien su código es accesible, el proveedor no garantiza la calidad ni el funcionamiento libre de errores del software. Dado que las funciones de comunicación (_ftdi.Open, _ftdi.Read, _ftdi.Write, ftdi.Close) requieren la presencia física del hardware, no pueden probarse unitariamente de forma aislada.	Un fallo no detectado en esta capa podría propagarse a capas superiores (Protocol, DeviceApi) y generar respuestas inválidas o falsos negativos sin una excepción evidente.	5	2	NO	Ejecutar pruebas de integración directas sobre la clase FtdiWindowsTransportService del módulo TransportLayer, verificando las operaciones críticas del driver (Open, ReadAsync, WriteCommandAsync, Close) en escenarios con hardware conectado y desconectado.	Se definieron pruebas PI2_000 hasta PI15_000, con hardware conectado y desconectado. Las pruebas se ejecutan directamente contra la Wand, verificando el correcto uso de la librería FTD2XX_NET.dll. Se agregan pruebas para USBWatcher: PI16_000, PI17_000, PI18_000, PI19_000, PI20_000 y PI21_000	SI	Diseño de arquitectura

En la fila del riesgo RProd-5-000, columna 9 puede apreciarse como PI3-000 es una de las pruebas de integración del conjunto que verifican la mitigación (columna 8) del riesgo en la etapa diseño de arquitectura.

Ejemplo de trazabilidad para las pruebas de sistema:

Matriz de Ejecución y Resultados de Pruebas - Versión 2.0									
Trazabilidad							Ejecución		
ID de prueba	Tarjeta JIRA vinculada	Nombre de la prueba	Tipo de prueba	Requerimiento vinculado	Casos de Uso o Issues asociados	Riesgo vinculado	Resultado esperado	Resultado obtenido	Observaciones
PS24-000	HIS-164	Show_IPG_Details_With_IPGTImeAndStartSessionTimeBigDiference	Sistema	RF7-000	CU7-000-Curso Alternativo 1	RProd-8-000	Cliente configura el smart mini con diferencia horaria. Iniciar dynamicScan, clic en botón connect. se debe mostrar un punto rojo a la derecha del IPG Time y al hacer mouse over el mensaje"The IPG time differs significantly from the system time"	Se muestra un punto rojo a la derecha del IPG Time y al hacer mouse over el mensaje"The IPG time differs significantly from the system time"	Verifica que control para una diferencia mayor a 1h entre el IPG Time y el StartSession Time
PS25-000	HIS-164	Visualize_Advance_Summary	Sistema	RF12-000	CU12-000	RProd-21-000	Iniciar DynamicScan, clic en botón connect, clic en botón interrogate, click en icono candado, ingesar la contraseña correcta. Se debe mostrar el resumen avanzado con los siguientes datos: Impedance V1, R-Wave Amplitude V1, Impedance V2, R-Wave Amplitude V2, PW Amplitude.	Se muestra el resumen avanzado con valores Impedance V1 = 0, R-Wave Amplitude V1 = 2.1 mV, Impedance V2 = 0, R-Wave Amplitude V2 = 2,1 mV, PW Amplitude = 2.2 mV	Los valores en cero de Impedance V1 y V2 son correctos para un IPG no implantado. La correctitud es controlada por tabla de comparación de valores DynamicScan vs App del cliente. Material de apoyo DOC. OSM REC_459
PS28-000	HIS-164	Interrogate_SavingDataToPDF_Without_IPG	Sistema	RF4-000	CU4-000 - Curso Alternativo 2	RProd-19-000	Iniciar DynamicScan, sin un IPG conectado a la Wand, clic en botón connect, clic en botón interrogate. Se debe detectar la ausencia del IPG, informar una alerta con el mensaje pop-up : "Connection Closed. Response invalid or incomplete Wand."	Se informar una alerta con el mensaje: "Connection Closed. Response invalid or incomplete Wand"	Se prueba ausencia de IPG
PS29-000	HIS-164	View_Trends	Sistema	RF13-000	CU13-000	RProd-21-000	Mostrar estadisticas en la Tabs Trends: 3 Tabs mas: Hours Activity / Battery / Trends. Hours Activity: mostrar la actividad de las ultimas 48hs en graficas. Actividad: Average HR (mbg), Activity: rango Very Ligth, Ligth, Moderate, Hard, Very Hard. Position : Laying, Reclined, Vertical, Active. Average HRV (ms).	Se muestra vacio estadisticas en la Tabs Trends: 3 Tabs mas: Hours Activity / Battery / Trends. Hours Activity: mostrar la actividad de las ultimas 48hs en graficas. Actividad: Average HR (mbg), Activity: rango Very Ligth, Ligth, Moderate, Hard, Very Hard. Position : Laying, Reclined, Vertical, Active. Average HRV (ms).	Esto es por que la prueba se realizo con el IPG sin actividad durante 48hs.
PS30-000	HIS-164	View_Historics_Reports	Sistema	RF16-000	CU16-000	RProd-21-000	Iniciar DynamicScan, click en botón History. Se debe mostrar un listado de los reportes historicos conteniendo los siguientes identificadores: ID: incremental ultimo/Date: /IPG Time/ModelID = ID_SMARTMINI_IPG, Serial=01087	Se muestra un listado de los reportes historicos conteniendo los siguientes identificadores: ID: incremental ultimo/Date: /IPG Time/ModelID = ID_SMARTMINI_IPG, Serial=01087. Las fechas son las del IPG Time y StartSession de cada informe.	Se muestran todos los reportes historicos. Y se puede descargar el PDF. La correctitud es controlada por tabla de comparación de valores DynamicScan vs App del cliente. Material de apoyo DOC. OSM REC_459
PS31-000	HIS-164	Visualize_Advance_Summary_Without_IPG	Sistema	RF12-000	CU12-000 - Curso Alternativo 1	RProd-19-000	Iniciar DynamicScan, clic en botón connect, clic en botón interrogate, click en icono candado. Desconectar o alejar IPG, ingesar la contraseña correcta. Se debe motrar un un pop-up con un mensaje de alerta que dice "Response invalid or incomplete Wand"	Tras desconectar o alejar IPG e ingesar la contraseña correcta en la sección avanzada el sistema mostró el pop-up con un mensaje de alerta que dice "Response invalid or incomplete Wand"	Validacion de desconexión de IPG
PS32-000	HIS-164	Visualize_Advance_Summary_Without_Wand	Sistema	RF12-000	CU12-000 - Curso Alternativo 2	RProd-10-000	Iniciar DynamicScan, clic en botón connect, clic en botón interrogate, click en icono candado, desconectar la antena Wand, ingesar la contraseña correcta. Se debe motrar un pop-up con un mensaje de alerta que dice: "CONNECTION CLOSED. Antenna Wand not found"	Tras desconectar la antena Wand e ingesar la contraseña correcta en la sección avanzada el sistema mostró el pop-up con un mensaje de alerta que dice: "CONNECTION CLOSED. Antenna Wand not found"	Validacion de desconexión de antena Wand

Trazabilidad para: **PS24-000:** *Show_IPG_Details_With_IPGTImeAndStartSessionTime-BigDiference*

Tarjeta JIRA: HIS-164

Requerimientos vinculados:

- RF7-000-Visualización detalles del IPG

Casos de uso vinculados:

- CU7-000-Curso Alternativo 1

5. **Curso Alternativo 1 - Hay una diferencia mayor a 1 hora entre la hora interna del IPG y la hora en que se establece la sesión.**
    a. *DynamicScan* muestra un indicador a definir por el equipo de desarrollo en la hora del IPG e informa por IU: *"The IPG time differs significantly from the system time"*

Riesgo de producto asociado:

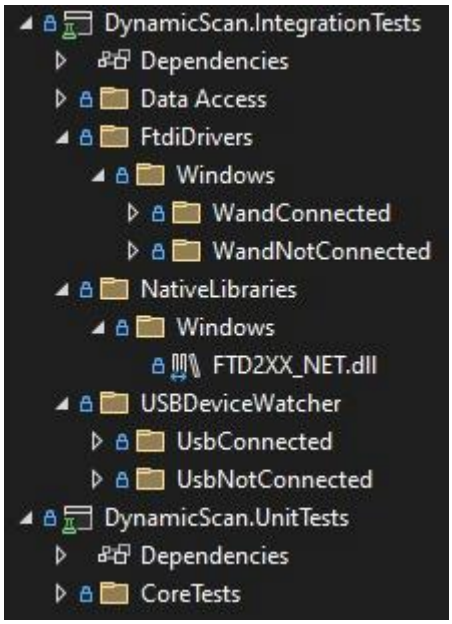
RProd-8-000-Error en la sincronización de hora entre el IPG y la DynamicScan

RProd-8-000	Error en la sincronización de hora entre el IPG y la app	Desfase horario entre los datos reportados por el IPG y DynamicScan, que puede afectar la interpretación de la información.	La hora del IPG es interna al dispositivo, y puede no estar alineada con la del sistema donde corre DynamicScan. Una diferencia grande entre ambas horas, podría incurrir en una mala interpretacion de los estados del paciente	5	2	NO	Verificar que la hora del IPG con la hora de inicio de sesión de DynamicScan mantengan una diferencia de tolerancia de <= 1h.	PS24-000	SI	Ing. de Requerimientos. RF7-000 - CU7-000 -CA1

En la fila del riesgo RProd-8-000, columna 9 puede apreciarse como PS24-000 es la prueba de sistema que verifican la mitigación (columna 8) del riesgo en la etapa ing. de requerimiento.

---------- FIN Matriz de ejecución y resultados de pruebas ----------

12.4.27. Estructura de los proyectos de prueba separados



12.4.28. Ejemplo de gestión y trazabilidad de riesgos del producto

RProd-5-000 — Dependencia del componente externo FTD2XX_NET.dll

Descripción:

Uno de los riesgos más relevantes identificados fue la dependencia del prototipo respecto a la librería FTD2XX_NET.dll, provista por la empresa FTDI para la comunicación entre el sistema y la antena Wand mediante el puerto USB 2.0.

Esta librería cumple un rol esencial como capa de bajo nivel, responsable de las operaciones de apertura, lectura, escritura, configuración y cierre entre otras, del canal de comunicación con el *hardware*.

Aunque el código es accesible, el fabricante no garantiza la calidad (*There are no warranties (or similar) in relation to the quality of the Software.*) ni la continuidad del *software* (ver licencia punto 5.3), lo que implica que su estabilidad no puede asumirse a priori.

A esto se suma la imposibilidad de ejecutar pruebas unitarias sobre las funciones que dependen directamente del *hardware*, ya que requieren la presencia física del dispositivo y acceso al puerto USB.

Por este motivo, el equipo decidió realizar pruebas de integración directas con *hardware* real, verificando el comportamiento del *driver* bajo diferentes condiciones de conexión.

Origen o causa potencial:

Un fallo en esta capa podría propagarse a componentes superiores (*ProtocolLayer*, *DeviceApi*), generando respuestas inválidas o falsos negativos sin una excepción evidente.

Probabilidad de ocurrencia: 5 – Frecuente

Gravedad de daño: 2 – Menor

¿Es aceptable ?: NO

Acción preventiva / medidas correctivas:

Ejecutar pruebas de integración directas sobre la clase *Ftdi*Windows*TransportService* del módulo *TransportLayer*, verificando las operaciones críticas del *driver* (*Open, ReadAsync, WriteCommandAsync, Close*) en escenarios con *hardware* conectado y desconectado.

Verificación:

Se definieron pruebas PI2_000 hasta PI15_000, con *hardware* conectado y desconectado. Las pruebas se ejecutan directamente contra la Wand, verificando el correcto uso de la librería FTD2XX_NET.dll.

Trazabilidad con Pruebas de integración:

PI2-000	HIS-96	Write_ShouldReturnTrue_WhenWriteSucceeds	Integracion	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método Write debe retornar true cuando logro escribir correctamente en el buffer.	Retorna true.	Valida la escritura correcta en el buffer de datos.
PI3-000	HIS-96	Write_ShouldReturnFalse_WhenDeviceNotConnected	Integracion	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	El método Write debe retornar false si el dispositivo no está conectado	Retorna false.	Validación del flujo alternativo esperable. No se lanza excepción. Control de estado correcto.
PI4-000	HIS-96	Open_ShouldReturnTrue_WhenDeviceConnected	Integracion	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método Open debe retornar true si el dispositivo FTDI está correctamente conectado	Retorna true.	Confirma inicialización y apertura del canal FTDI como se espera.
PI5-000	HIS-96	Open_ShouldReturnFalse_WhenNoDeviceConnected	Integracion	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	El método Open debe retornar false si no hay dispositivo FTDI conectado	Retorna false.	Validación de caso de error esperable. Control de dispositivos FTDI no conectados implementado correctamente.
PI6-000	HIS-96	Configure_ShouldReturnTrue_WhenCalledAfterOpen	Integracion	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método Configure debe retornar true si se ejecuta correctamente	Retorna true.	Validación de configuración FTDI post-conexión. No se detectaron errores al aplicar parámetros operativos.
PI7-000	HIS-96	Configure_ShouldNotThrow_WhenDeviceNotConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	El método Configure() no debe lanzar excepción si no hay dispositivo conectado.	No se lanzó excepción.	Se verifica que la clase maneja el estado desconectado de forma segura.
PI8-000	HIS-96	Close_ShouldNotThrow_WhenDeviceConnected	Integración	RF2-000	CU2-000	RProd-5-000	El método Close debe cerrar la conexión y no debe lanzar excepción si el dispositivo está conectado.	Se cierra la conexión y no se lanzó excepción.	Valida que se cierre correctamente el canal FTDI sin errores de ejecución.
PI9-000	HIS-96	Close_ShouldNotThrow_WhenDeviceIsNotConnected	Integración	RF2-000	CU2-000 - Curso Alternativo 2	RProd-5-000	El método Close no debe lanzar excepción si no hay conexión activa.	No se lanzó excepción.	Prueba el cierre defensivo del recurso en estado desconectado.
PI12-000	HIS-96	ReadResponseAsync_ShouldReturnResponse_WhenDeviceConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método ReadResponseAsync debe devolver una respuesta no vacía con conexión establecida y un mensaje válidos enviado hacia la Wand	Respuesta recibida correctamente.	Esta prueba crea un flujo completo de : open, WriteComandAsync, ReadResponseAsync y Close. Testea el ReadResponseAsync en los Asserts.
PI13-000	HIS-96	ReadResponseAsync_ShouldThrowTimeout_WhenDeviceNotConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	Lanzar TimeoutException cuando no hay conexión activa de la Wand.	TimeoutException correctamente lanzada.	Se prueba el comportamiento esperado del metodo cuando no hay respuesta desde el hardware.
PI14-000	HIS-96	WriteCommandAsync_ShouldReturnTrue_WhenDeviceConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000, CU4-000, CU13-000, CU14-000	RProd-5-000	El método WriteCommandAsync debe retornar true luego de escribir en el buffer si hay dispositivo conectado.	Escritura exitosa, devuelve true.	Validación de escritura con dispositivo conectado.
PI15-000	HIS-96	WriteCommandAsync_ShouldReturnFalse_WhenDeviceNotConnected	Integración	RF1-000, RF4-000, RF13-000, RF14-000	CU1-000-Curso Alternativo 1, CU4-000-Curso Alternativo 1	RProd-5-000	Devuelve false si no hay dispositivo conectado.	Escritura fallida, devuelve false.	Validación sin hardware conectado.

¿Es aceptable el riesgo residual ?: SI

Etapa: Diseño de Arquitectura

RProd-6-000-Lectura errónea de datos por corrupción en la transmisión

Descripción del riesgo:

Implementación incorrecta de la función de CRC establecida en el protocolo FP17.

Origen de la causa potencial:

Una implementación incorrecta de la función de CRC establecida en el protocolo FP17 podría incurrir en no detectar mensajes corruptos, afectando la integridad de la información recibida.

Probabilidad de ocurrencia: 5 - Frecuente

Gravedad de daño: 2 – Menor

¿Es aceptable?: NO

Acción preventiva/medida correctiva:

Para mitigar este riesgo se implementa correctamente la función de CRC (*Cyclic Redundancy Check*) conforme a la especificación del protocolo FP17, y se diseñaron pruebas unitarias exhaustivas (PU2-000 a PU12-000) orientadas a validar su comportamiento en diferentes escenarios: tramas válidas, tramas con errores intencionales, alteraciones de longitud y contenido.

Trazabilidad con pruebas unitarias:

Matriz de Ejecución y Resultados de Pruebas - Versión 2.0									
Trazabilidad							Ejecución		
ID de prueba	Tarjeta JIRA vinculada	Nombre de la prueba	Tipo de prueba	Requerimiento vinculado	Casos de Uso o Issues asociados	Riesgo vinculado	Resultado esperado	Resultado obtenido	Observaciones
PU2-000	HIS-104	CRC_ITT_Should_Return_CorrectValue_For_KnownSequence	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	13471 expresado en decimal	13471 expresado en decimal	Validacion con secuencia conocida de cinco bytes (Comando GET_VER) {0xFA, 0x00, 0x35, 0x2F, 0x24} valor de CRC proporcionado por cliente.
PU3-000	HIS-104	CRC_ITT_Should_Detect_LastSingleByteChange	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC original = 13471, CRC cambio ultimo byte = 9406	CRC original = 13471, CRC cambio ultimo byte = 9406	Prueba de sensibilidad a cambio al final del array de datos del mensaje. Original = {0xFA, 0x00, 0x35, 0x2F, 0x24} , Cambio = {0xFA, 0x00, 0x35, 0x2F, 0x25}
PU4-000	HIS-104	CRC_ITT_Should_Detect_FirstSingleByteChange	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC original = 13471 , CRC cambio primer byte = 40654	CRC original = 13471 , CRC cambio primer byte = 40654	Prueba de sensibilidad a cambio al inicio del array de datos del mensaje. Original = {0xFA, 0x00, 0x35, 0x2F, 0x24} , Cambio Original = {0xFB, 0x00, 0x35, 0x2F, 0x24}
PU5-000	HIS-104	CRC_ITT_Should_Detect_MiddletSingleByteChange	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC original = 13471 , CRC cambio en byte del medio = 943	CRC obtenido = 943	Prueba de sensibilidad a cambio al inicio del array de datos del mensaje. Original = {0xFA, 0x00, 0x35, 0x2F, 0x24} , Cambio Original = {0xFB, 0x00, 0x34, 0x2F, 0x24}
PU6-000	HIS-104	CRC_ITT_Should_Return_Initial_When_InputEmpty	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC = 65535 para entrada vacia = 0xFFFF por deficion	CRC = 665535	El objetivo de esta prueba es verificar que la funcion de calculo de CRC no se rompra frente a un mensaje vacio ni tampoco lo identifique como un mensaje corrupto. El valor 0xFFFF es el que establece el procotolo FP17 confidencial.
PU7-000	HIS-104	CRC_ITT_Should_Return_CorrectCRC_For_All2eros	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC esperado 6258 en decimal	CRC = 6258 en decimal	Los mensajes reales pueden contener secciones con datos repetidos (por padding, relleno o buffers inicializados). Se prueba la secuencia { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } y se contrasta con calculadora online
PU8-000	HIS-104	CRC_ITT_Should_Return_CorrectCRC_For_RepeatedValues	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC expresados en hexadecimal para valores repetidos parametrizados deben dar: {0xA6AF, 0x9A55, 0x039A, 0x877E, 0x84C0}	{0xA6AF, 0x9A55, 0x039A, 0x877E, 0x84C0}	Prueba ante valores hexadecimales repetidos, se parametrizan entradas para mejorar cobertura: { 0x35, 0x35, 0x35, 0x35 }, { 0xAA, 0xAA, 0xAA, 0xAA }, { 0x55, 0x55, 0x55, 0x55 }, { 0xF8, 0xF8, 0xF8, 0xF8 }, { 0x00, 0x00, 0x00, 0x00 },
PU9-000	HIS-104	CRC_ITT_InitParameter_Overload_Should_ComputeCorrectCRC	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC calculado con init=65535 e input = 44461	CRC obtenido en decimal = 44461	Se prueba la sobrecarga del metodo cuya firma permite como valores de entrada ademas del imput de un mensaje a calcular, el valor inicial de entrada. En este caso 0xFFFF o 65535.
PU10-000	HIS-104	CRC_ITT_OriginalFunction_Should_ProcessSingleByte_Correctly_WithVariousInputs	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC esperados en Hexa {0xC782, 0xE1F0, 0xFF00, 0xF550, 0xEBA0}	CRC obtenidos en Hexa {0xC782, 0xE1F0, 0xFF00, 0xF550, 0xEBA0}	Firma original de funcion CRC que utiliza como entrada de datos 1 byte y el inicializador.Conjutno de datos parametrizados {0xFFFF,0x31, 0xC782} // ASCII '1', {0xFFFF, 0x00, 0xE1F0 // Byte nulo, {0xFFFF, 0xFF, 0xFF00}// Byte máximo, {0xFFFF, 0xAA, 0xF550}// Alternancia binaria
PU11-000	HIS-104	CRC_ITT_Should_Detect_MissingByte	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC calculado en decimal = 13471 el obtenido debe ser = 280 por estar corrupto	CRC obtenido en decimal = 280	original = { 0xFA, 0x00, 0x35, 0x2F, 0x24 }; corrupto = { 0xFA, 0x00, 0x2F, 0x24 }; // falta el byte 0x35
PU12-000	HIS-104	CRC_ITT_Should_Detect_InsertedByte	Unitaria	RF1-000, RF4-000, RF13-000, RF14-000.	CU1-000 / CU4-000 / CU13-000 / CU14-000	RProd-8-000	CRC calculado en decimal = 13471 se insertan bytes de mas, el corrupto debe dar =54633	CRC obtenido en decimal = 54633	original = { 0xFA, 0x00, 0x35, 0x2F, 0x24 }; corrupto = { 0xFA, 0x00, 0x35, 0x35, 0x2F, 0x24 }; // byte extra 0x35

¿Es aceptable el riesgo residual?: SI

Etapa: Diseño detallado

RProd-10-000 — Detección fallida Sistema–USB–Antena (nivel RF)

Descripción:

La comunicación entre DynamicScan, el puerto USB 2.0 y la antena Wand puede fallar por errores de conexión física, desconexión de la antena, puertos inactivos o detección incorrecta.

Origen de la causa potencial:

El riesgo surge de la dependencia directa del sistema respecto al *hardware* y los controladores USB. Si no existiera un mecanismo que validara el estado de la antena o que notificara al usuario ante una desconexión, la experiencia de uso se vería comprometida y podría confundirse con un error del *software*.

Probabilidad de ocurrencia: 4 - Probable

Gravedad del daño: 2 - Menor

¿Es aceptable?: NO

Acción correctiva/medida preventiva:

Implementar en la UI un indicador visual y un mensaje de estado que informen la conexión o desconexión de la Wand.

Verificación:

Pruebas de sistema: PS1-000, PS2-001, PS3-000, PS4-000, PS5-001, PS10-000, PS12-001, PS32-000

Trazabilidad con pruebas sistema:

PS1-000	HIS-164	StartSession_ShouldEstablishConnectionWithIpg	Sistema	RF1-000	CU1-000	RProd-10-000	Iniciar DynamicScan, clicl en botón connect. Recibir correctamente los datos del IPG: modelo = Hora de inicio de sesion, hora del IPG Time , ID_SMARTMINI_IPG, Serial=01087	Hora de inicio de sesion=2025-08--31 10:41:09, hora del IPG Time = 2025-09-11 12:55:05 , ID_SMARTMINI_IPG, Serial=01087	Prueba diseñada para evaluar al sistema (DynamicScan + Antena Wand + IPG Smart Mini) en el requerimiento RF1 - Establecer conexión
PS2-001	HIS-164	StartSession_ShouldNotEstablishConnectionWithIpg_WhenNonWandDevicePlugged	Sistema	RF1-000	CU1-000 - Curso Alternativo 1 / BUG-05 / BUG-06	RProd-10-000	El sistema (DynamicScan + Antena Wand) no debe establecer una conexion con el IPG con la antena Wand desconectada y debe lanzar un mensaje de alerta que indique: "Wand disconnected! Please connect the Wand before continuing"	El sistema (DynamicScan + Antena Wand) no establece una conexion con el IPG con la antena Wand desconectada y lanza un mensaje de alerta incorrecto: "No connection to the device. Please connnect the Wand before continuing"	La prueba pasa correctamente el mensaje es en Ingles y sobre la Wand. La prueba de realizo por segunda vez para validar esto.
PS3-000	HIS-132	DeviceDetection_ShouldShowConnectedStatus_WhenWandPluggedIn	Sistema	RF17-000	CU17-000	RProd-10-000	Iniciar DynamicScan con la Wand conectada, la UI debe mostrar "Wand Connected".	Se muestra "Wand Conected"	La detección del dispositivo FTDI y su visualización en la interfaz gráfica funcionan correctamente en este escenario. Sistema (DynamicScan + Antena Wand)
PS4-000	HIS-132	DeviceDetection_ShouldShowDisconnectedStatus_WhenWandUnplugged	Sistema	RF17-000	CU17-000	RProd-10-000	Iniciar DynamicScan con la Wand desconectada , la UI debe mostrar "Wand Disconnected". Conectar la Wand, la UI debe mostrar "Wand Connected"	Se muestra "Wand Diconnected" y al conectar la Wand se muestra "Wand connected" en la UI	La detección del dispositivo FTDI y su visualización en la interfaz gráfica funcionan correctamente en este escenario. Sistema (DynamicScan + Antena Wand)
PS5-001	HIS-164	DeviceDetection_ShouldShowDisconnectedStatus_WhenAnotherUSB_DevicePluggedIn	Sistema	RF17-000	CU17-000 - Cursos Alternativo 1 / BUG-01	RProd-10-000	Iniciar DynamicScan, conectar otro dispositivo distinto a una Wand, un receptor de mouse modelo: Logitec - M260, la UI debe mostrar "Wand Disconnected" y debe salir un mensaje de alerta.	Se inicia DynamicScan con otro dispositivo distinto a una Wand conectado en la interfaz USB la UI y muestra "Wand Disconnected" y sale un mensaje de alerta.	Prueba realizada por segunda vez en cliente tras el BUG-01 finalizado, Sistema (DynamicScan + interfaz 2.0 de un mouse inhalambrico)
PS10-000	HIS-164	CloseConnection_FromConnectionEstablished_UnplugingWand	Sistema	RF2-000	CU2-000 - Curso Alternativo 2	RProd-10-000	Iniciar DynamicScan, establecer conexion, luego quitar la antena Wand y hacer clic en botón salir. No se debe cerrar al conexion y se debe volver a la pantalla actual.	No se cierra la conexion y se vuelve a la pantalla actual.	Verifica que la desconexion de la antena no rompa el flujo del programa.
PS12-001	HIS-164	Interrogate_SavingDataToPDF_UnplugingWand	Sistema	RF4-000	CU4-000 - Curso Alternativo 1 /BUG13	RProd-10-000	Iniciar DynamicScan, establecer conexion, desconectar la Wand, clic en botón interrogate. Se informa con un pop-up que dice "Antenna WAND not found"	Se informa con un pop-up que dice "Antenna WAND not found"	Verifica que si la antena WAND se desconecto, no se intente la interrogacion. Y se avise con un pop-up "Connection Closed. Antenna WAND Not Found" Prueba version 2
PS32-000	HIS-164	Visualize_Advance_Summary_Without_Wand	Sistema	RF12-000	CU12-000 - Curso Alternativo 2	RProd-10-000	Iniciar DynamicScan, clic en botón connect, clic en botón interrogate, click en icono candado, desconectar la antena Wand, ingesar la contraseña correcta. Se debe motrar un pop-up con un mensaje de alerta que dice: "CONNECTION CLOSED. Antenna Wand not found"	Tras desconectar la antena Wand e ingesar la contraseña correcta en la sección avanzada el sistema mostró el pop-up con un mensaje de alerta que dice: "CONNECTION CLOSED. Antenna Wand not found"	Validacion de desconexión de antena Wand

¿Es aceptable el riesgo residual?: SI

Etapa: Ing. de Requerimientos. RF1-000 - CU1-000 - CU1-000-CA1 / RF2-000 - CU2-000-CA2 / RF4-000 - CU4-000 - CU4-000-CA1 / RF12-000 - CU12-000-CA2 / RF17-000 - CU17-000

---------- FIN DE EJEMPLOS ----------

12.4.29. Matriz de Trazabilidad integral demás versiones

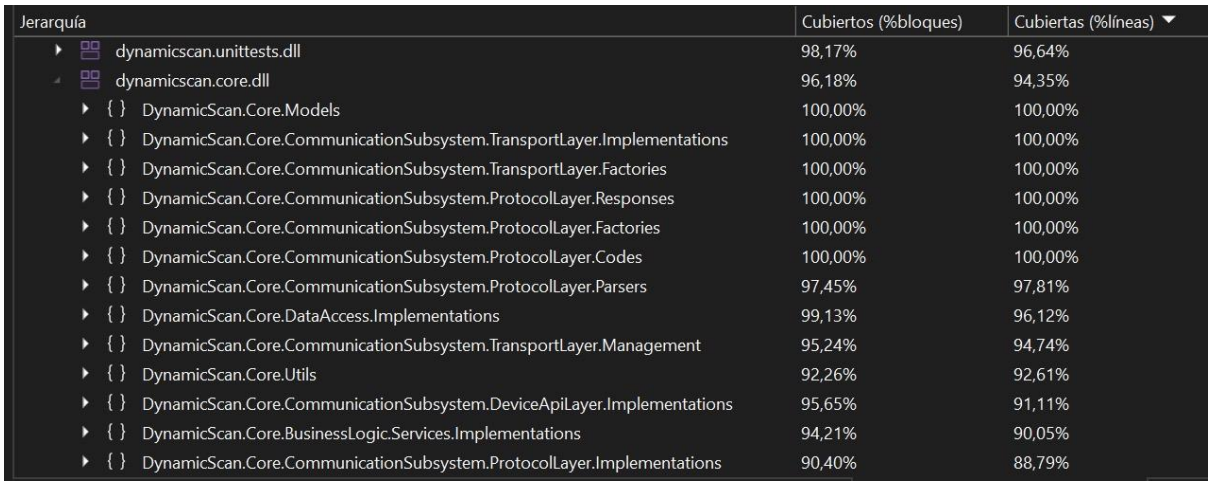
Integración

Matriz de trazabilidad de integral - Versión 2.0									
Requerimientos				Pruebas				Componente de arquitectura	
ID-Versión	Nombre	Descripción	Tarjetas JIRA vinculadas	ID-Versión	Tipo	Descripción	Casos de Uso o Issues asociados	Diseño de arquitectura	Diseño detallado
PRUEBAS DE INTEGRACION									
RF1-000, RF4-000, RF13-000, RF14-000	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI2-000	Integración	El método Write debe retornar true cuando logro escribir correctamente en el buffer.	CU1-000, CU4-000, CU13-000, CU14-000	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI3-000	Integración	El método Write en su curso normal retorna false para una Wand desconectada.	CU1-000, CU4-000, CU13-000, CU14-000	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000.	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI4-000	Integración	El método Open() debe retornar true si el dispositivo FTDI está correctamente conectado	CU1-000, CU4-000, CU13-000, CU14-000	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000.	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI5-000	Integración	El método Open() debe retornar false si no hay dispositivos FTDI conectados	CU1-000-Curso Alternativo, CU4-000-Curso Alternativo 1, CU13-000-Curso Alternativo 2	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000.	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI6-000	Integración	El método debe retornar true si se ejecuta correctamente	CU1-000, CU4-000, CU13-000, CU14-000	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000.	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI7-000	Integración	El método Configure() no debe lanzar excepción si no hay dispositivo conectado.	CU1-000-Curso Alternativo, CU4-000-Curso Alternativo 1, CU13-000-Curso Alternativo 2	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF2-000	Establecer la conexión	Cerrar la conexión	HIS-58 / HIS-93	PI8-000	Integración	Debe cerrar la conexión y no debe lanzar excepción si el dispositivo está conectado.	CU2-000	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF2-000	Establecer la conexión	Cerrar la conexión	HIS-58 / HIS-93	PI9-000	Integración	El método no debe lanzar excepción si no hay conexión activa.	CU2-000 - Curso Alternativo 2	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000.	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI12-000	Integración	Devuelve una respuesta no vacía con conexión establecida.	CU1-000, CU4-000, CU13-000, CU14-000	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000.	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI13-000	Integración	Lanza TimeoutException cuando no hay conexión.	CU1-000-Curso Alternativo, CU4-000-Curso Alternativo 1, CU13-000-Curso Alternativo 2	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000.	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI14-000	Integración	Devuelve true cuando el dispositivo está conectado.	CU1-000, CU4-000, CU13-000, CU14-000	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF1-000, RF4-000, RF13-000, RF14-000.	Establecer la conexión / Obtencion de Datos / Trends / Obtener la hora del IPG	Para multiples descripciones ver el ESRE	HIS-79 / HIS-57 / HIS-81 / HIS-82 / HIS 60 / HIS 105 / HIS 78 / HIS 79 / HIS-152 / HIS-155 / HIS-153 / HIS-147 / HIS-162	PI15-000	Integración	Devuelve false si no hay dispositivo conectado.	CU1-000-Curso Alternativo, CU4-000-Curso Alternativo 1, CU13-000-Curso Alternativo 2	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: FtdiWindowsTransportService, FTDI
RF17-00	Detección de antena Wand	El sistema deberá detectar automáticamente el estado de conexión de la antena Wand.	HIS-76 / HIS-118	PI16-000	Integración	Valida que no se detecten dispositivos FTDI inexistentes.	CU17-000	**Módulos**: TransportLayer (interno), módulo externo FTD2XX_NET (assembly)	**Clases**: UsbDeviceWatcher, FTDI

Sistema

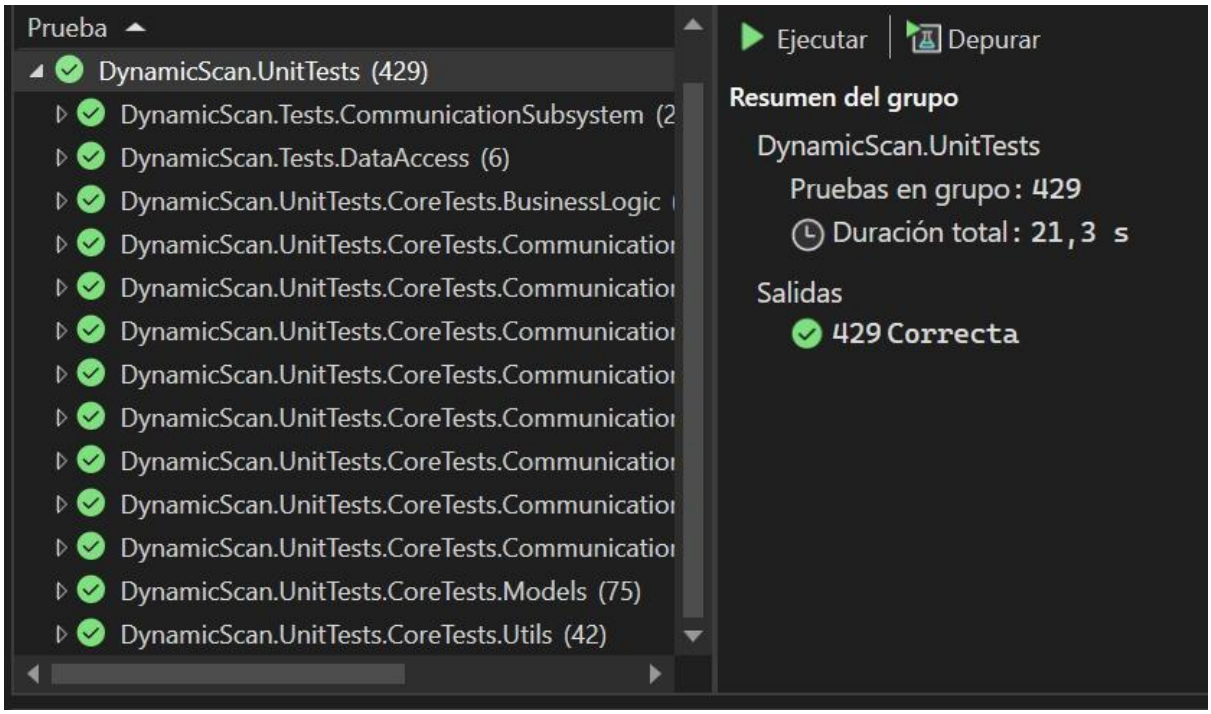
Matriz de trazabilidad de integral - Versión 2.0									
Requerimientos				Pruebas				Componente de arquitectura	
ID-Versión	Nombre	Descripción	Tarjetas JIRA vinculadas	ID-Versión	Tipo	Descripción	Casos de Uso o Issues asociados	Diseño de arquitectura	Diseño detallado
RF2-000	Cerrar la conexion	El sistema deberá permitir al usuario cerrar la conexión de DynamicScan con el IPG de manera controlada	HIS-58/HIS-93	PS16-000	Sistema	Verifiva el curso alternativo 1 , clic en boton cancelar.	CU2-000 - Curso Alternativo 1	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **MainViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs, StartSessionHandler.cs, TransportServiceProxy.cs, FtdiWindowsTransportService.cs**
RF7-000	Detalles del IPG	Visualizar en la UI los siguientes datos: el modelo, número de serie, la fecha y hora de inicio de sesión y la fecha y hora del IPG	HIS-103	PS17-000	Sistema	Verifica el curso normal y el RF14-000	CU7-000	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **MainViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs, StartSessionHandler.cs, FramingProtocol17.cs, AlcpWithFP17.cs, AlcpScript.cs, TransportServiceProxy.cs, FtdiWindowsTransportService.cs**
RF8-000	Visualizar informes	Permitir al usuario visualizar un informe con los datos obtenidos del IPG en pantalla en el navegador configurado por defecto.	HIS-63/HIS-156	PS18-000	Sistema	Verifica curso normal.	CU8-000	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **MainViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs**
RF9-000	Generación de Logs	Permitir que el prototipo genere y almacene logs de ejecución de forma persistente, finalidad: trazabilidad, soporte o auditoría.	HIS-64	PS19-000	Sistema	Verifica curso normal.	CU9-000	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **InitialSessionViewModel.cs, Logger.cs , AppService.cs, SessionStateService.cs, DeviceApi.cs, StartSessionHandler.cs, FramingProtocol17.cs, AlcpWithFP17.cs, AlcpScript.cs, TransportServiceProxy.cs, FtdiWindowsTransportService.cs**
RF10-000	Visualizar *Splash Screen*	Visualizar el logo de Impulse Dynamics al iniciar la aplicación antes de cargar la pantalla de inicio de sesión.	HIS-65	PS20-000	Sistema	Verifica curso normal.	CU10-000	Módulo: **ViewModels, View**	Clase: **SpashScreenView**
RF11-000	Acceso a la sección avanzada	Acceder a datos y métricas de las últimas 48hs del paciente mediante autenticación con contraseña.	HIS-145/HIS106	PS21-000	Sistema	Verifica curso normal.	CU11-0000	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **MainViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs**
RF11-000	Acceso a la sección avanzada	Acceder a datos y métricas de las últimas 48hs del paciente mediante autenticación con contraseña.	HIS-145/HIS106	PS22-000	Sistema	Verifica curso alternativo	CU11-000 - Curso Alternativo 1	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **MainViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs**
RF11-000	Acceso a la sección avanzada	Acceder a datos y métricas de las últimas 48hs del paciente mediante autenticación con contraseña.	HIS-145/HIS106	PS23-000	Sistema	Verifica curso alternativo	CU11-000 - Curso Alternativo 2	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **MainViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs**
RF7-000	Detalles del IPG	Visualizar en la UI los siguientes datos: el modelo, número de serie, la fecha y hora de inicio de sesión y la fecha y hora del IPG	HIS-103	PS24-000	Sistema	Verifica curso alternativo	CU7-000 - Curso alternativo 1	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **MainViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs, StartSessionHandler.cs, FramingProtocol17.cs, AlcpWithFP17.cs, AlcpScript.cs, TransportServiceProxy.cs, FtdiWindowsTransportService.cs**
RF12-000	Visualización del resumen avanzado	Dentro de la sección avanzada, se proporcionará una pestaña llamada "Summary" que permitirá a los técnicos visualizar un resumen con información clave obtenida del IPG	HIS-67/HIS106	PS25-000	Sistema	Verifica curso normal.	CU12-000	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **InitialSessionViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs**
RF4-000	Obtencion de Datos	Se debe proporcionar una sección dentro de la pantalla principal que mostrará la hora interna del IPG	HIS 60 / HIS 105 / HIS 78 / HIS 79	PS28-000	Sistema	Verifica curso sin IPG	CU4-000-Curso Alternativo 2	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **InitialSessionViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs, StartSessionHandler.cs, FramingProtocol17.cs, AlcpWithFP17.cs, AlcpScript.cs, TransportServiceProxy.cs, FtdiWindowsTransportService.cs**
RF13-000	Trends	Permitir al usuario visualizar estadísticas y tendencias del IPG dentro de la sección avanzada	HIS-155 / HIS-153 /HIS-149 /HIS-147 /HIS-162	PS29-000	Sistema	Ver estadísticas en modo avanzado	CU13-000	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **InitialSessionViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs, StartSessionHandler.cs, FramingProtocol17.cs, AlcpWithFP17.cs, AlcpScript.cs, TransportServiceProxy.cs, FtdiWindowsTransportService.cs**
RF16-000	Visualización de Historial de Informe	Permitir al usuario visualizar los informes almacenados de sesiones anteriores ordenados en forma cronológica	HIS-135/HIS148	PS30-000	Sistema	Ver historico de informes, que salgan ordenados, que tengan los identificadoeres correctos y decargar el pdf	CU16-000	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **InitialSessionViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs**
RF12-000	Visualización del resumen avanzado	Dentro de la sección avanzada, se proporcionará una pestaña llamada "Summary" que permitirá a los técnicos visualizar un resumen con información clave obtenida del IPG	HIS-67/HIS106	PS31-000	Sistema	Verifica curso alternativo 1 desconexión de IPG	CU12-000 - Curso Alternativo 1	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **InitialSessionViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs**
RF12-000	Visualización del resumen avanzado	Dentro de la sección avanzada, se proporcionará una pestaña llamada "Summary" que permitirá a los técnicos visualizar un resumen con información clave obtenida del IPG	HIS-67/HIS106	PS32-000	Sistema	Verifica curso alternativo 1 desconexión de IPG	CU12-000 - Curso Alternativo 2	Módulo: **ViewModels, Views, Service, CommunicationSubsystem**	Clase: **InitialSessionViewModel.cs, AppService.cs, SessionStateService.cs, DeviceApi.cs**

12.4.30. Code coverage



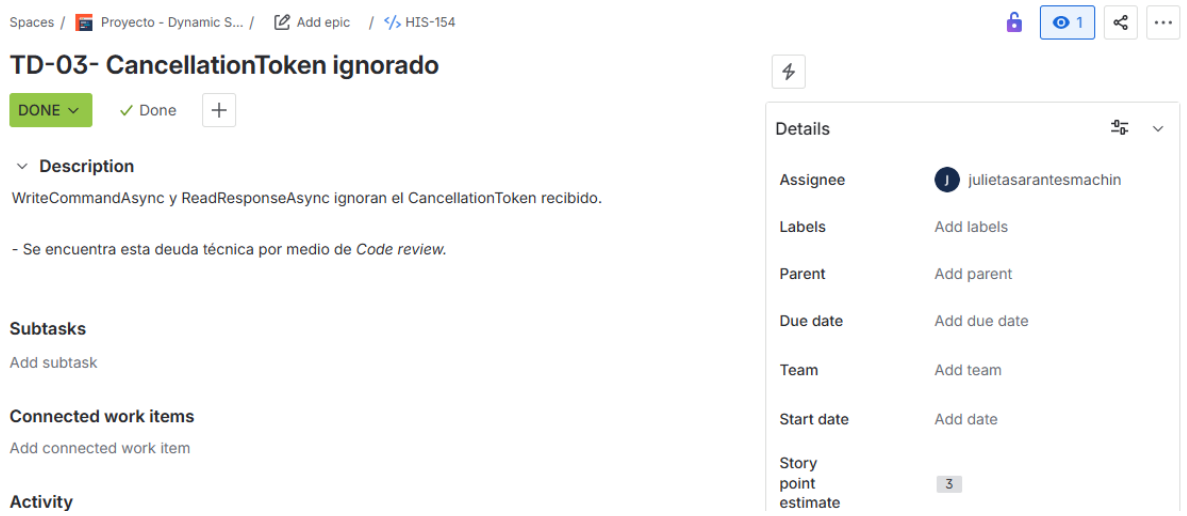
Jerarquía	Cubiertos (%bloques)	Cubiertas (%líneas)
dynamicscan.unittests.dll	98,17%	96,64%
dynamicscan.core.dll	96,18%	94,35%
{ } DynamicScan.Core.Models	100,00%	100,00%
{ } DynamicScan.Core.CommunicationSubsystem.TransportLayer.Implementations	100,00%	100,00%
{ } DynamicScan.Core.CommunicationSubsystem.TransportLayer.Factories	100,00%	100,00%
{ } DynamicScan.Core.CommunicationSubsystem.ProtocolLayer.Responses	100,00%	100,00%
{ } DynamicScan.Core.CommunicationSubsystem.ProtocolLayer.Factories	100,00%	100,00%
{ } DynamicScan.Core.CommunicationSubsystem.ProtocolLayer.Codes	100,00%	100,00%
{ } DynamicScan.Core.CommunicationSubsystem.ProtocolLayer.Parsers	97,45%	97,81%
{ } DynamicScan.Core.DataAccess.Implementations	99,13%	96,12%
{ } DynamicScan.Core.CommunicationSubsystem.TransportLayer.Management	95,24%	94,74%
{ } DynamicScan.Core.Utils	92,26%	92,61%
{ } DynamicScan.Core.CommunicationSubsystem.DeviceApiLayer.Implementations	95,65%	91,11%
{ } DynamicScan.Core.BusinessLogic.Services.Implementations	94,21%	90,05%
{ } DynamicScan.Core.CommunicationSubsystem.ProtocolLayer.Implementations	90,40%	88,79%

12.4.31. Pruebas unitarias



12.4.32. Bug en Jira

12.4.33. Deuda técnica en Jira



12.5. Gestión del proyecto

12.5.1. Ejemplo de formato de minuta de reunión tutora

	Minuta de reunión de tutoría	
	Guillermo Echichure	
	02/01/2025	**Versión 1.0**

Modalidad: Online

Herramienta utilizada: Microsoft Teams

Hora: 19:30 hs.

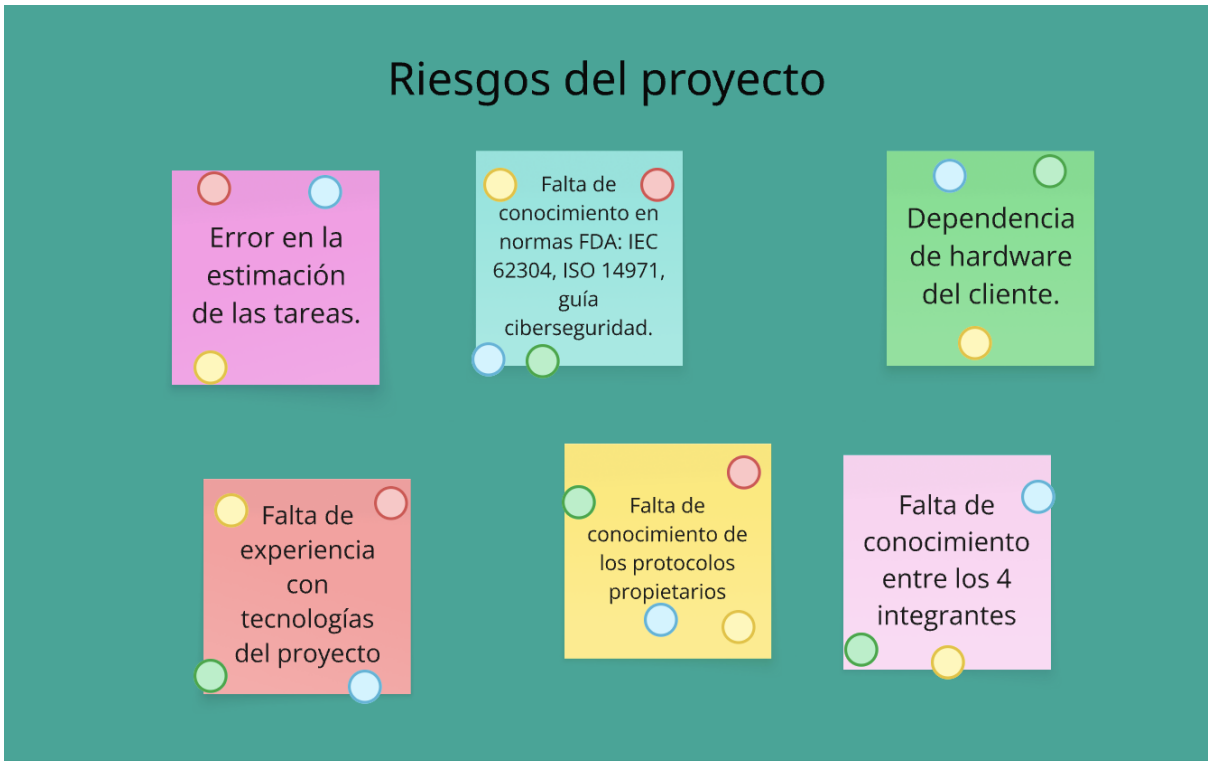
Asistentes:

- **Equipo de proyecto**: Tomás Dilema, Milena Dos Santos, Julieta Sarantes y Guillermo Echichure.
- **Tutor**: Ing. Amalia Álvarez.

Observaciones del tutor:

- ☑ Relevar algún check intermedio, alguna actividad de control intermedia del tipo de SQA quizás en el plan mismo de SQA.
- ☑ El plan de SQA quizás sea un plan de Software Assurance.
- ☑ Cuales son las actividades que vamos a hacer en cada una de las fases y cual es el producto resultante de cada fase. La herramienta que tenemos es el Plan de SQA donde identificamos las fases y sus resultados.
- ☑ Análisis de riesgos del producto, analizaremos los riesgos más críticos que están vinculados a los requerimientos.
- ☑ Se trata de identificar cuales son las acciones de control para reducir el riesgo del producto, incorporando actividades de aseguramiento de la calidad. (conversar con el cliente).
- ☑ Primer ejercicio detallar las fases (ya están identificadas), bajar un nivel de abstracción e identificar las actividades que componen. Luego cuál va a ser el producto resultado de esta fase.
- ☑ Con el diseño arquitectónico lo mismo.
- ☑ Una vez tenido el plan de la calidad pasamos al plan de SQA o de Software Assurance.

12.5.2. Brainstorming



12.5.3. Matriz de riesgos del proyecto

Matriz de Riesgos			
ID	**Riesgo**	**Descripción/Efecto**	**Acciones**
R1	Falta de experiencia con tecnologías del proyecto.	Retrasos en el desarrollo por curva de aprendizaje.	Realizar capacitación con cursos o tutoriales online.
R2	Falta de conocimiento en normas FDA: IEC 62304, ISO 14971, guía de ciberseguridad.	El producto podría desarrollarse aplicando criterios generales sin incorporar completamente las normas requeridas para *software* médico.	Obtener una copia de cada norma y estudiarlas en profundidad para comprender su alcance y aplicar los puntos relevantes al proyecto.

R3	Dependencia de *hardware* del cliente.	El desarrollo depende del acceso a la antena Wand y al dispositivo implantable IPG disponibles únicamente en las instalaciones del cliente.	Coordinar el acceso al *hardware* en horarios no laborables para evitar demoras.
R4	Falta de conocimiento de los protocolos propietarios.	El protocolo de comunicación con el IPG integra tres subprotocolos confidenciales cuya complejidad técnica, sumada al acceso limitado al *hardware*, puede dificultar su correcta implementación.	Definir una etapa inicial del proyecto dedicada al estudio, análisis y pruebas del protocolo, planificando sesiones de pruebas los fines de semana.
R5	Falta de conocimiento de los cuatro integrantes.	Los cuatro integrantes no habían trabajado juntos previamente, lo que podría afectar la coordinación y distribución inicial de tareas.	Fomentar instancias de comunicación y planificación conjunta para consolidar la dinámica de trabajo en equipo. Usar las retrospectivas para identificar oportunidades de mejora.
R6	Error en estimación de tareas.	Falta de experiencia o conocimiento detallado sobre la complejidad de las tareas y tecnologías del proyecto.	Realizar estimaciones utilizando puntos de historia y *planning poker*.

12.5.4. Criterio de evaluación de los riesgos

Para cada riesgo identificado se utilizan las siguientes métricas:

Impacto: Clasificación (Crítico, Alto, Medio, Bajo) en función de las consecuencias.
Probabilidad: Posibilidad de ocurrencia de 1 a 5.
Puntaje del Riesgo: Se calcula multiplicando:

- **Impacto (1-4)** x **Probabilidad (1 a 5)**
- **Riesgos Críticos**: Aquellos con puntajes mayores o iguales a 15.

Criterio:

- **Verde (Bajo):** Valor de riesgo ≤ 5

- **Ámbar (Medio):** Valor de riego $6 \leq IxP \leq 12$
- **Rojo (Alto):** Valor de riesgo ≥ 15

Se utilizará la escala producto de la matriz de probabilidad e impacto, junto con la coloración **RAG** (*red, amber, green*) para representar riesgos alto, medio y bajo, respectivamente. A continuación, se presenta la escala.

	Impacto			
Probabilidad	**1**	**2**	**3**	**4**
1	**1**	**2**	**3**	**4**
2	**2**	**4**	**6**	**8**
3	**3**	**6**	**9**	**12**
4	**4**	**8**	**12**	**16**
5	**5**	**10**	**15**	**20**

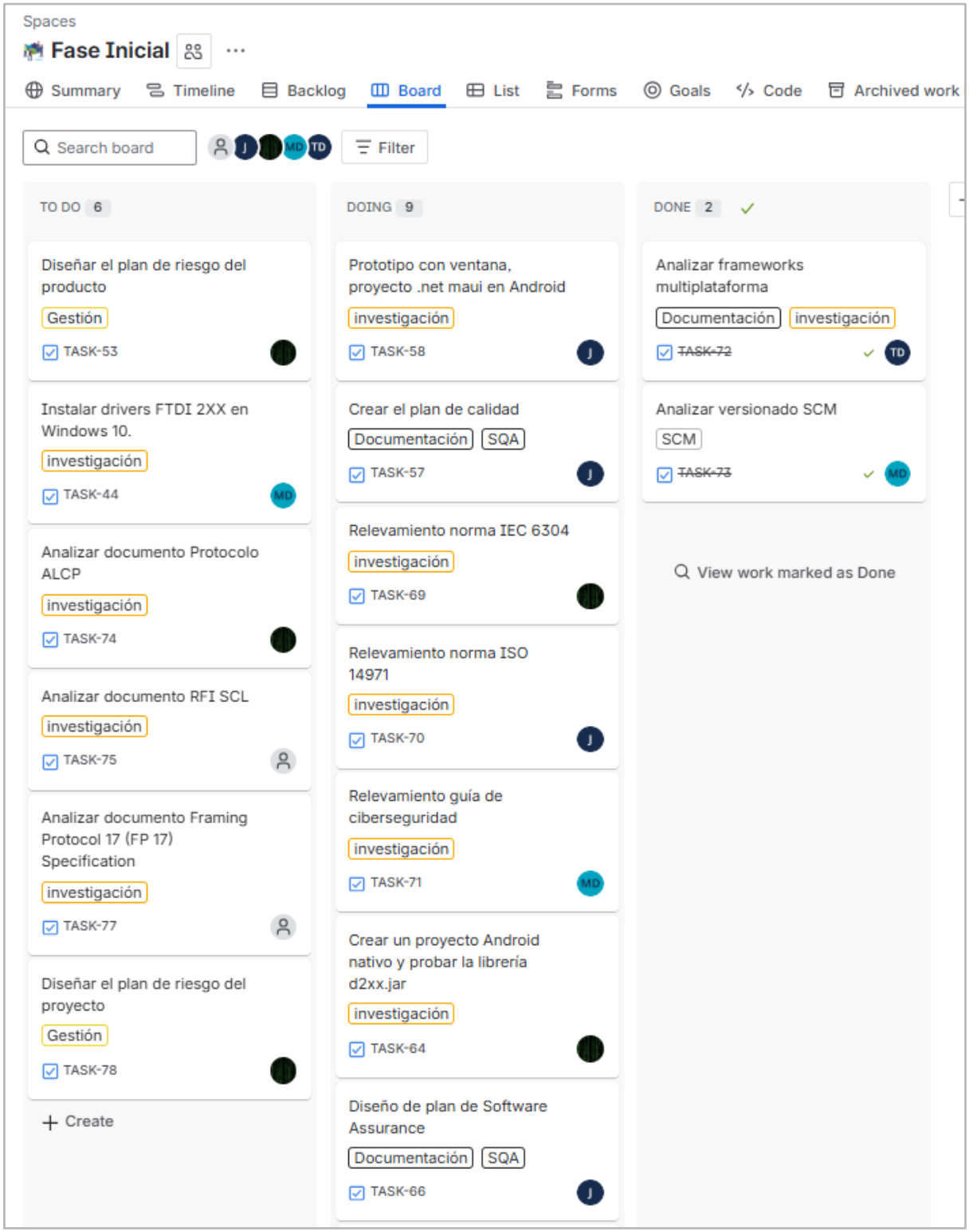
12.5.5. Tabla Registro de riesgos

ID	Riesgo	Fase investigación	*Sprint* 1	*Sprint* 2	*Sprint* 3	*Sprint* 4	*Sprint* 5	*Sprint* 6	*Sprint* 7
R1	Falta de experiencia con las tecnologías del proyecto.	(5,4) 20	(3,4) 12	(3,3) 9	(2,2) 4	(2,2) 4	(2,1) 2	(1,1) 1	(1,1) 1
R2	Falta de conocimiento en normas FDA: IEC 62304, ISO 14971 y guía de ciberseguridad.	(5,4) 20	(4,4) 16	(3,4) 12	(3,3) 9	(3,3) 9	(2,3) 6	(2,2) 4	(2,2) 4
R3	Dependencia de *hardware* del cliente.	(5,4) 20	(5,4) 20	(4,4) 16	(4,4) 16	(4,4) 16	(4,4) 16	(4,3) 12	(4,2) 8
R4	Falta de conocimiento de los protocolos propietarios.	(5,4) 20	(4,4) 16	(3,4) 12	(2,3) 6	(2,3) 6	(2,3) 6	(2,2) 4	(2,2) 4
R5	Falta de conocimiento entre los cuatro integrantes	(4,3) 12	(3,3) 9	(2,2) 4	(2,2) 4	(2,2) 4	(1,1) 1	(1,1) 1	(1,1) 1
R6	Error en estimación de tareas.	-	(5,4) 20	(4,4) 16	(4,3) 12	(4,4) 16	(3,3) 9	(3,3) 9	(2,2) 4

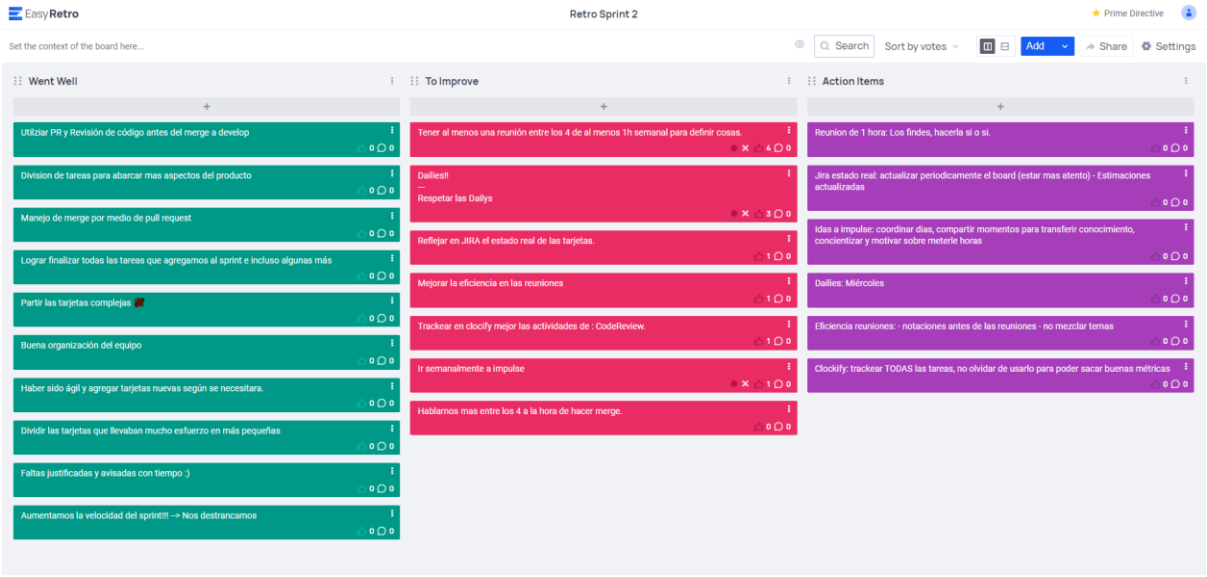
12.5.6. Matriz de probabilidad con los riesgos del proyecto al finalizarlo

	Impacto			
Probabilidad	**1**	**2**	**3**	**4**
1	**R1, R5**			
2		**R2, R4, R6**		
3				
4		**R3**		
5				

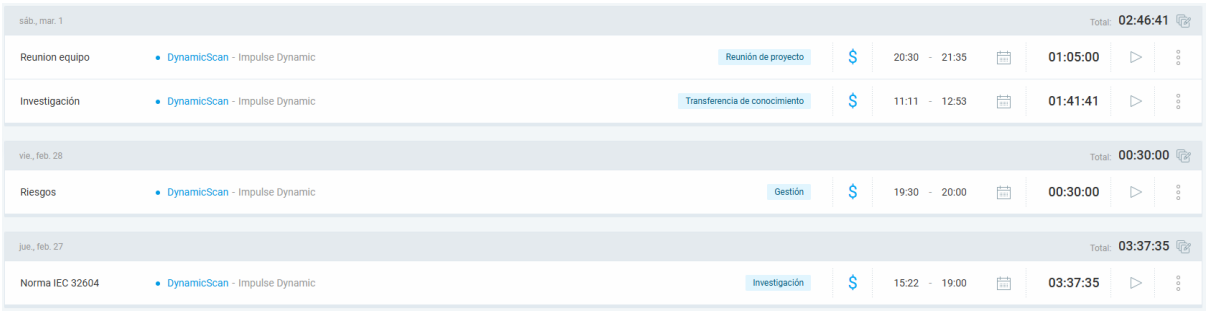
12.5.7. Tablero Kanban



12.5.8. Ejemplo de reunión retrospectiva



12.5.9. Ejemplo registro Clockify

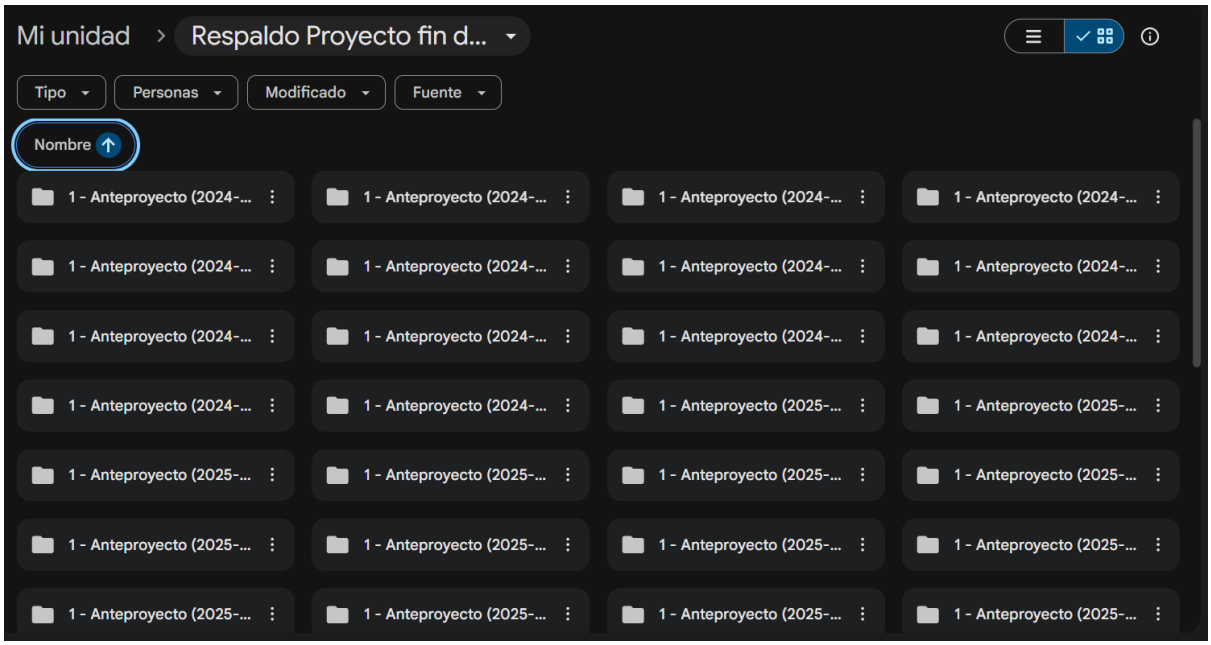


12.6. Gestión de la configuración

12.6.1. Código de respaldo

```
function backupDriveFolder() {
    var folderId = '1iS_8DZDklrMnHJr5X5uc8d4W7vQkdP1S';
    var destinationFolderId = '1rcuIg-JKImDE9bwQsMZICz7ux12FDXpB';
    var sourceFolder = DriveApp.getFolderById(folderId);
    var destinationFolder = DriveApp.getFolderById(destinationFolderId);

    copyAllFolders(sourceFolder, destinationFolder);

    Logger.log('Respaldo completo.');
}

function copyAllFolders(sourceFolder, destinationFolder) {
    var date = new Date();
    var formattedDate = Utilities.formatDate(date, Session.getScriptTimeZone(), 'yyyy-MM-dd');

    var files = sourceFolder.getFiles();
    while (files.hasNext()) {
        var file = files.next();
        var newFileName = file.getName() + ' (' + formattedDate + ')';
        var copiedFile = file.makeCopy(newFileName, destinationFolder);
        Logger.log('Archivo copiado: ' + copiedFile.getName());
    }

    var subfolders = sourceFolder.getFolders();
    while (subfolders.hasNext()) {
        var subfolder = subfolders.next();
        var newSubfolderName = subfolder.getName() + ' (' + formattedDate + ')';
        var newSubfolder = destinationFolder.createFolder(newSubfolderName);
        Logger.log('Carpeta creada: ' + newSubfolder.getName());
        copyAllFolders(subfolder, newSubfolder);
    }
}
```

12.6.2. Unidad de respaldo en Google Drive



12.7. Conclusiones

12.7.1. Resultados de las pruebas de sistema

RF1-000 — Establecer la conexión

CU1-000 — Curso normal

- **PS1-000 — StartSession_ShouldEstablishConnectionWithIpg**
- Se verificó el flujo principal de conexión entre DynamicScan, la antena Wand y el IPG Smart Mini.

CU1-000 — Curso alternativo 1: Wand desconectada

- **PS2-001 — StartSession_ShouldNotEstablishConnectionWithIpg_WhenNon-WandDevicePlugged**
- Se validó que el sistema no establezca conexión si la Wand no está conectada.
  Al intentar conectar, DynamicScan mostró la alerta:
  "*No connection to the device. Please connect the Wand before continuing*".

CU1-000 — Curso alternativo 2: Otro dispositivo USB conectado

- **PS5-001 — DeviceDetection_ShouldShowDisconnectedStatus_WhenAnotherUSB_DevicePluggedIn**

- Prueba compartida con **RF17-000 / CU17-000 - Curso Alternativo 1**.

- Se probó la conexión de un dispositivo USB no compatible (receptor de mouse Logitech M260). DynamicScan detectó correctamente la ausencia de la Wand y mostró el estado "*Wand Disconnected*" junto con un mensaje de alerta.
  La prueba validó la detección de dispositivos no FTDI.

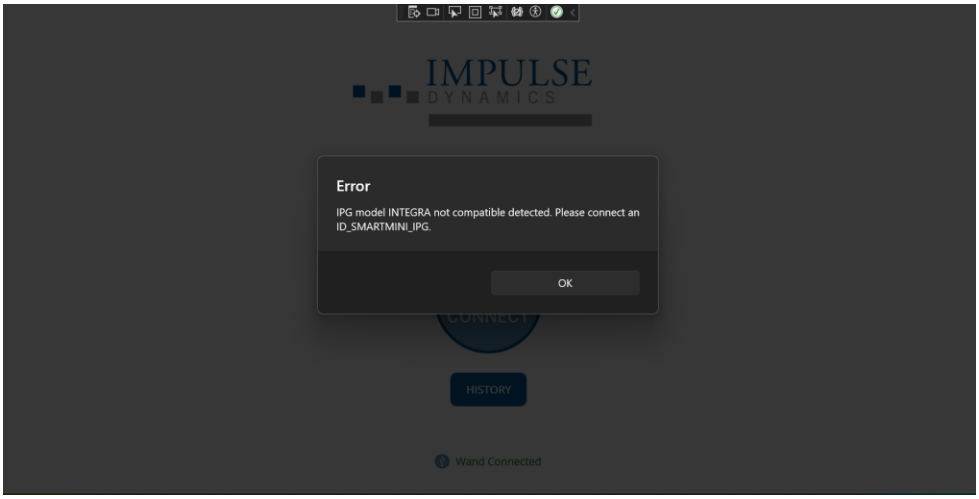
CU1-000 — Curso alternativo 3: Conexión a otro IPG distinto al Smart Mini

- **PS7-000 — StartSession_ShouldNotEstablishConnectionWithAnother_IPG_Model**

- Se evaluó el comportamiento ante un modelo de IPG distinto al *Smart Mini*, se conectó el modelo INTEGRA.

- El sistema mostró un pop-up de alerta:
  "IPG model INTEGRA not COMPATIBLE detected. Please connect an ID_SMART-MINI_IPG."

- Se confirma la validación de compatibilidad del modelo del dispositivo.



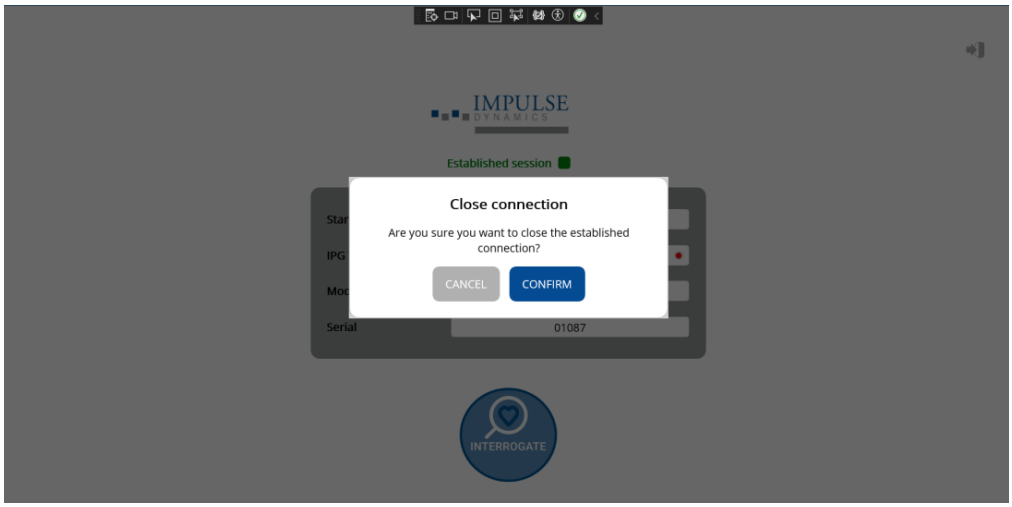
CU1-000 — Curso alternativo 4: Conexión sin IPG

- **PS8-001 — StartSession_ShouldNotEstablishConnectionWithout_IPG**
- Se inició DynamicScan y se presionó *Connect* sin un IPG presente.
- El sistema detectó la ausencia del dispositivo y mostró un pop-up de alerta: "*Do not receive response from IPG in expected time*."
- Se confirma la validación ante ausencia de un IPG.

RF2-000 — Cerrar la conexión

CU2-000 — Curso normal

- **PS9-000 — CloseConnection_FromConnectionEstablished**
- Desde una sesión establecida, al presionar *Exit*, el sistema cerró la conexión correctamente y retornó a la pantalla inicial.



CU2-000 — Curso alternativo 1: Cancelar cierre de conexión

- **PS16-000 — CloseConnection_FromConnectionEstablished_CloseCanceled**
- Al intentar cerrar la sesión y seleccionar *Cancelar* en el pop-up, el sistema mantuvo la conexión activa y no modificó la vista, cumpliendo con el comportamiento esperado.

CU2-000 — Curso alternativo 2: Cerrar conexión con desconexión de Wand antes de confirmar.

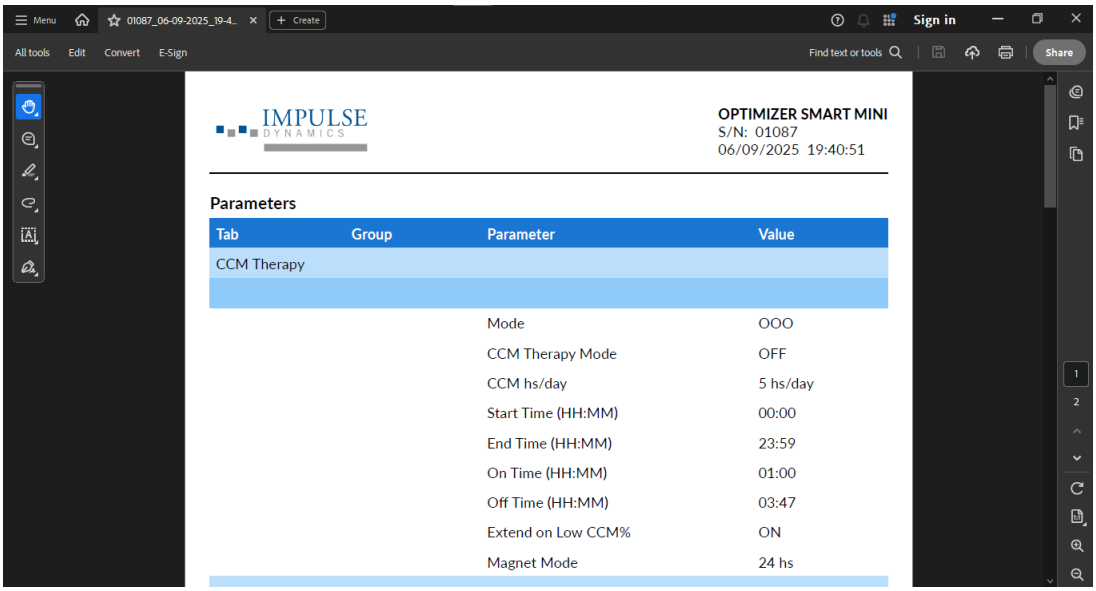
- **PS10-000 — CloseConnection_FromConnectionEstablished_UnplugingWand**
- Se desconectó la Wand durante la sesión y luego se intentó cerrar la aplicación.
- DynamicScan manejó correctamente la excepción, manteniendo la estabilidad y retornando a la pantalla inicial.

RF4-000 — Obtención de datos

CU4-000 — Curso normal

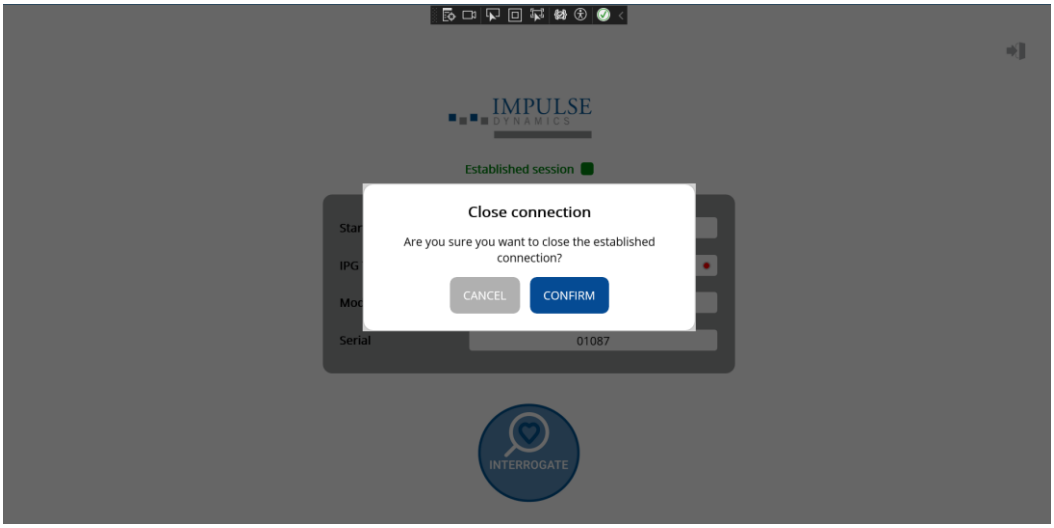
- **PS11-000 — Interrogate_SavingDataToPDF**

- Se ejecutó la función *Interrogate* y se verificó la generación de un archivo PDF en la carpeta *Reports*, con coincidencia total entre los valores obtenidos y los de la aplicación oficial del cliente (*Tabla correctitud de los datos*).

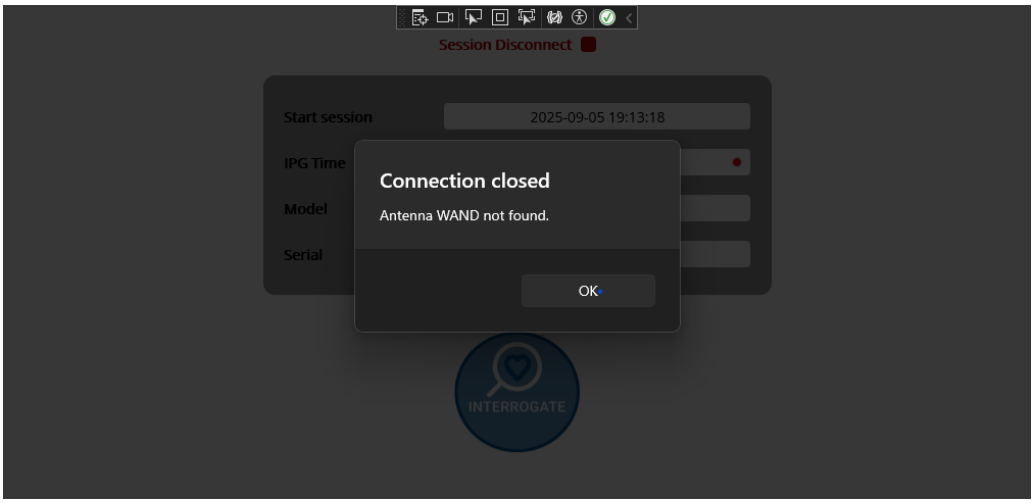


CU4-000 — Curso alternativo 1: Desconexión de Wand antes de interrogar

- **PS12-001 — Interrogate_SavingDataToPDF_UnplugingWand**



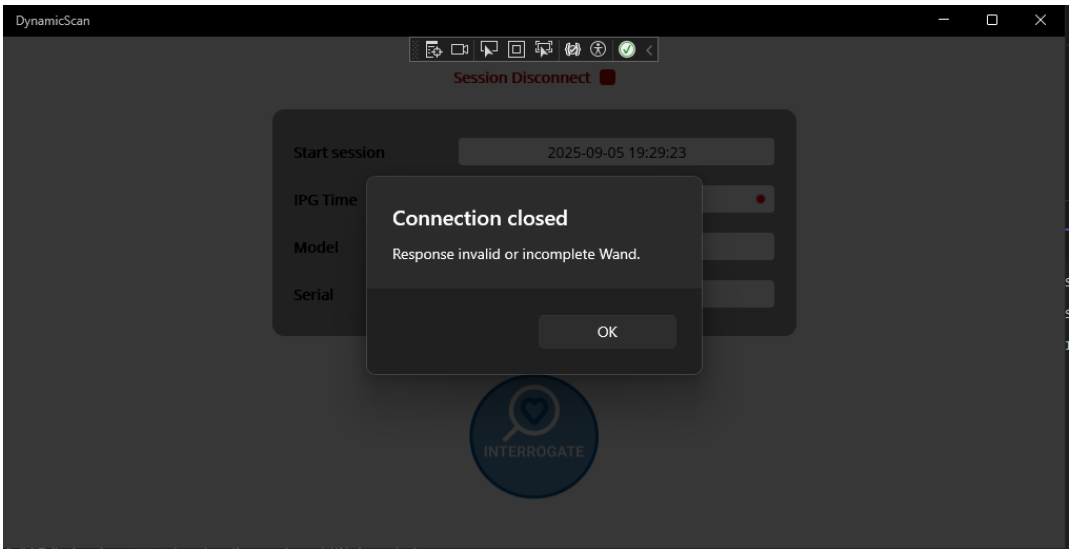
- Con la Wand desconectada, el sistema mostró un pop-up de alerta: "*Antenna WAND not found*", confirmando que no se intentó interrogar sin conexión activa.



CU4-000 — Curso alternativo 2: IPG desconectado

- **PS28-000 — Interrogate_SavingDataToPDF_Without_IPG**

- En ausencia de IPG, el sistema mostró un pop-up de alerta: "*Connection Closed. Response invalid or incomplete Wand*." Se valida la detección del escenario sin dispositivo implantable.



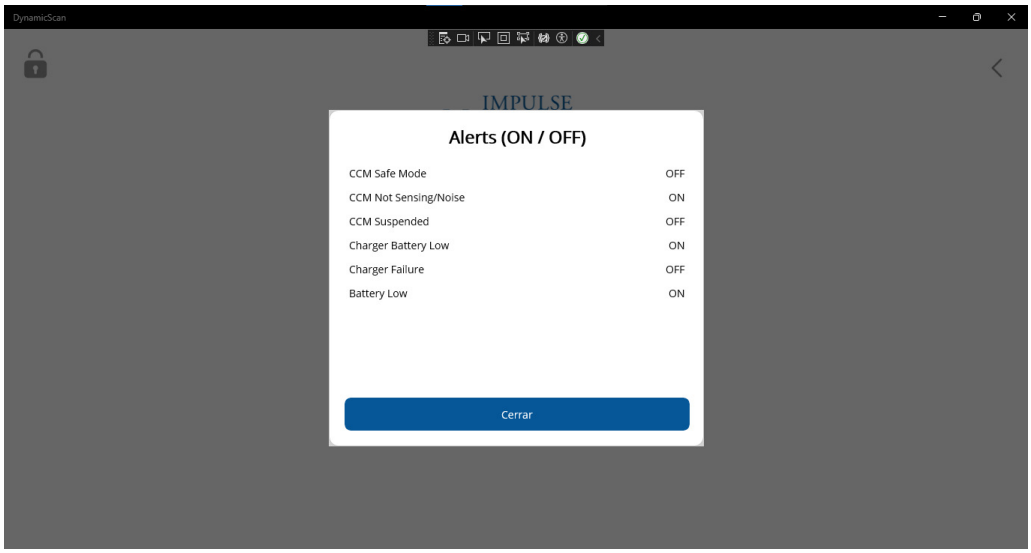
RF5-000 — Visualizar de alertas

CU5-000 — Curso normal

- **PS13-000 — AlertVisualization_FromIpgDataRetrieved**
- Se verificó la correcta lectura y visualización de alertas configuradas por el cliente.
- DynamicScan presentó las mismas seis alertas en ON igual que la aplicación oficial, con coincidencia de estados y número de alertas 6 en rojo en el botón *Alerts*.

CU5-000 — Curso alternativo 1: Sin alertas

- **PS6-000 — AlertVisualization_FromIpgDataRetrieved_NoAlertsToShow**
- Aquí el cliente configuró 3 alertas en ON y 3 en OFF: *CCM Safe = OFF, CCM Not Sensing Noise = ON, CCM Suspender = OFF, Charge Battery Low = ON, Charge Failuer = OFF, Battery Low = ON*. Luego Interrogó el IPG con DynamicScan y se vieron las mismas 3 alertas ON y OFF que en la aplicación oficial. Y a su vez el botón Alerts con un indicador visual en rojo con número 3 indicando que vinieron 3 alertas activas.



RF7-000 — Visualización detalles del IPG

CU7-000 — Curso normal

- **PS17-000 — Show_IPG_Details**
  DynamicScan mostró los detalles del dispositivo (StartSession, IPG Time, Model, Serial), con valores consistentes y sincronizados.

CU7-000 — Curso alternativo 1: Diferencia horaria

- Esta prueba se utiliza también para validar el **RF14-000** que es el que obtiene los datos de la hora.
- **PS24-000 — Show_IPG_Details_With_IPGTimeAndStartSessionTimeBigDifference**
- El cliente configuró el IPG con una diferencia horaria mayor a una hora entre el sistema y el IPG.
- DynamicScan mostró el indicador rojo y el mensaje contextual al hacer *mouseover* sobre el punto rojo: "*The IPG time differs significantly from the system time*."

RF8-000 — Visualización de informe

CU8-000 — Curso normal

- **PS18-000 — View_Report**

- Se verificó la apertura del PDF generado tras la interrogación, visualizado correctamente en el navegador por defecto, con coincidencia total entre los valores obtenidos y los de la aplicación oficial del cliente (*Tabla correctitud*).

RF9-000 — Generación de logs

CU9-000 — Curso normal

- **PS19-000 — Logs_Generated**

- Se comprobó la creación del archivo *DeviceLog_YYYY-MM-DD_HH-MM-SS.txt* en la carpeta *Logs*, confirmando el registro automático de actividad.

RF10-000 — Visualización de *Splash Screen*

CU10-000 — Curso normal

- **PS20-000 — Splash_Screen**

- Al iniciar la aplicación, se visualizó correctamente la pantalla de inicio con el logotipo de Impulse Dynamics.

RF11-000 — Acceso a sección avanzada

CU11-000 — Curso normal

- **PS21-000 — AdvanceMode_Access**
  Se validó el acceso exitoso al

- modo avanzado mediante autenticación correcta.

CU11-000 — Curso alternativo 1: Contraseña incorrecta

- **PS22-000 — AdvanceMode_Access_Incorrect_Password**

- Al ingresar una contraseña errónea, mostró un pop-up de alertas: “Error. Incorrect password.”
  Se confirmó la seguridad del acceso restringido.

CU11-000 — Curso alternativo 2: Cancelar acceso

- **PS23-000 — AdvanceMode_Access_Canceled**
- Al cancelar el ingreso, el sistema permaneció en la pantalla actual sin acceder a la sección avanzada.

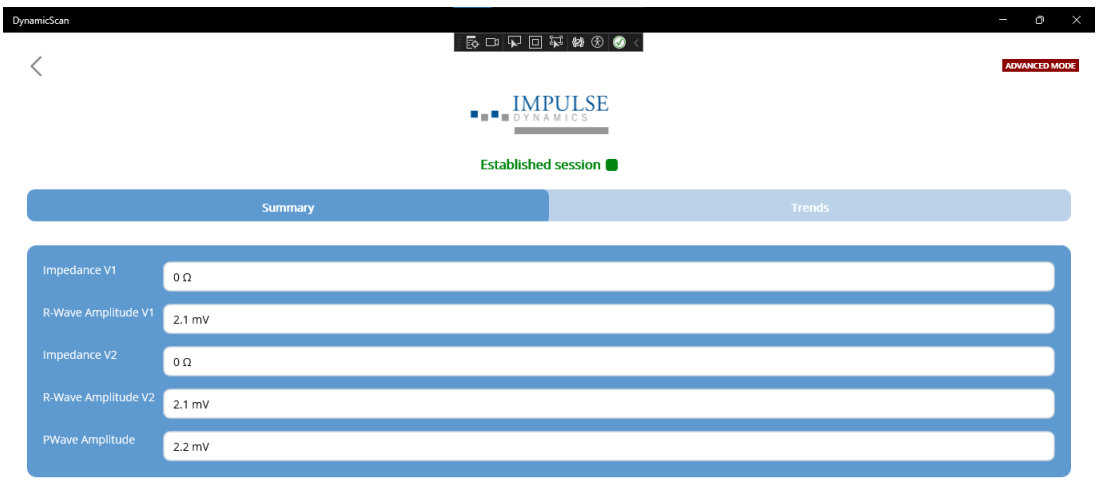
RF12-000 - Visualización del resumen avanzado

CU12-000 — Curso normal

- **PS25-000 — Visualize_Advance_Summary**
- DynamicScan mostró los valores de impedancia (V1, V2 en Ω), amplitud (V1, V2 en mV) y amplitud del pulso (PW). Los valores vienen configurados por el cliente.

CU12-000 — Curso alternativo 1: Se desconecta el IPG luego de la interrogación y antes de ingresar al modo avanzado

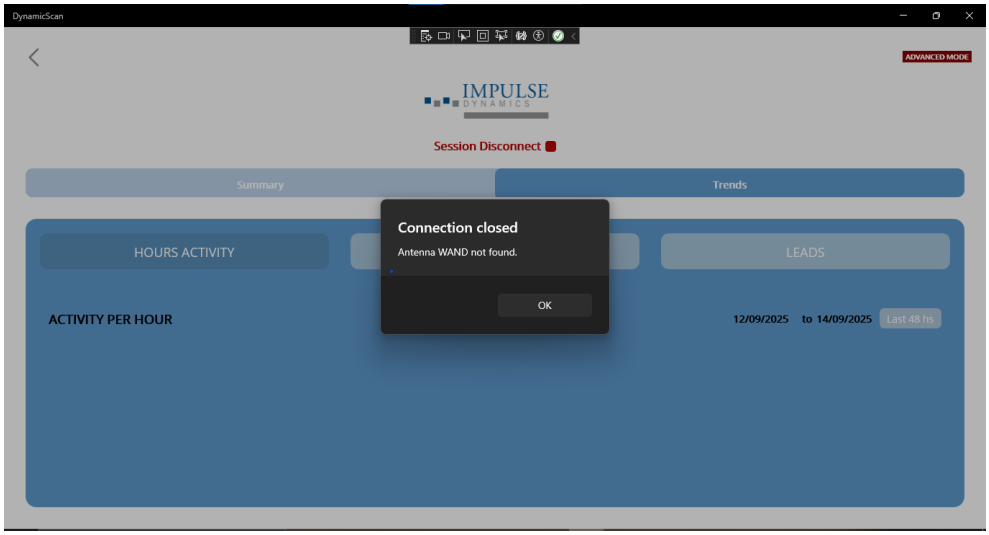
- **PS31-000 — Visualize_Advance_Summary_Without_IPG**
- El sistema detectó la ausencia del IPG y mostró un pop-up de alerta:
  "Response invalid or incomplete Wand.



CU12-000 — Curso alternativo 2: Se desconecta antena Wand.

- **PS32-000 — Visualize_Advance_Summary_Without_Wand**

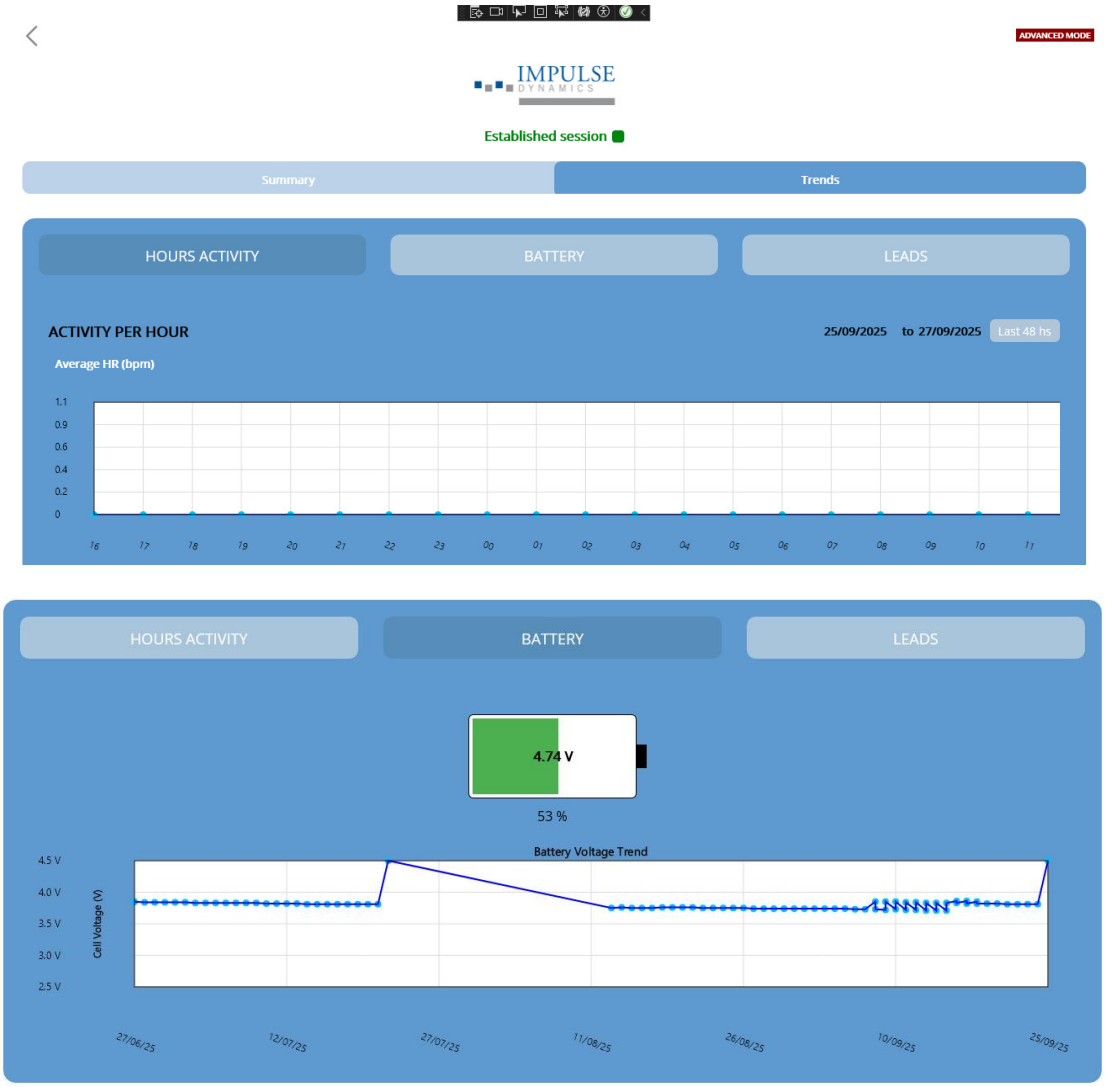
- Ante la desconexión de la Wand, el sistema mostró un pop-up de alerta: “CONNECTION CLOSED. Antenna Wand not found.”



RF13-000 — Trends

CU13-000 — Curso normal

- **PS29-000 — View_Trends**
- Se visualizaron correctamente las pestañas *Activity*, *Battery* y *Trends*.
- Las gráficas mostraron impedancia en Ω y amplitud en mV. Se detectó ausencia de datos de actividad por inactividad prolongada del IPG (>48 h).


ADVANCED MODE
IMPULSE DYNAMICS
Established session
Summary
Trends
HOURS ACTIVITY
BATTERY
LEADS
ACTIVITY PER HOUR
25/09/2025 to 27/09/2025
Last 48 hs
Average HR (bpm)
1.1
0.9
0.6
0.4
0.2
0
16 17 18 19 20 21 22 23 00 01 02 03 04 05 06 07 08 09 10 11
HOURS ACTIVITY
BATTERY
LEADS
4.74 V
53 %
Battery Voltage Trend
Cell Voltage (V)
4.5 V
4.0 V
3.5 V
3.0 V
2.5 V
27/06/25
12/07/25
27/07/25
11/08/25
26/08/25
10/09/25
25/09/25

RF14-000 — Obtener la hora del IPG

- Probado en la prueba anterior **PS17-000 - Show_IPG_Details**, que cubre el RF7-000 donde se muestran estos datos.

RF16-000 - Visualización de historial de informes

CU16-000 — Curso normal

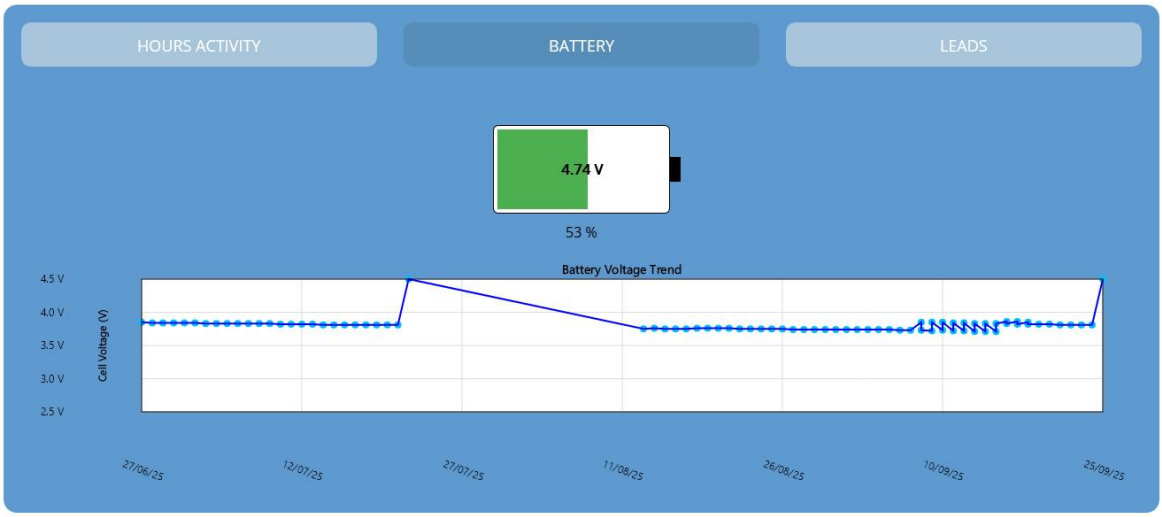
- **PS30-000 — View_Historics_Reports**

- Se visualizó correctamente el listado de reportes históricos con ID incremental, fecha, hora del IPG, modelo y número de serie, ordenados cronológicamente. Se comprobó la descarga funcional de los informes en PDF.

RF17-000 — Detección de la Wand

CU17-000 — Curso normal

- **PS4-000 — DeviceDetection_ShouldShowDisconnectedStatus_WhenWandUnplugged**



- Se validó el comportamiento de la interfaz ante la desconexión y reconexión de la Wand, mostrando correctamente los estados “Wand Disconnected” y “Wand Connected”.

CU17-000 - Curso Alternativo 1

- Prueba compartida con **RF1-000 - CU1-000 - Curso Alternativo 2**

- **PS5-001 — DeviceDetection_ShouldShowDisconnectedStatus_WhenAnotherUSB_DevicePluggedIn**

- Se probó la conexión de un dispositivo USB no compatible (receptor de mouse Logitech M260). DynamicScan detectó correctamente la ausencia de la Wand y mostró el estado “*Wand Disconnected*” junto con un mensaje de alerta.
  La prueba validó la detección de dispositivos no FTDI.

Comentario general:

Las pruebas de sistema verificaron el cumplimiento integral de los requerimientos funcionales, cubriendo tanto los cursos normales como los alternativos definidos en los casos de uso en el ESRE.

A su vez, confirmaron la correcta interacción entre los componentes principales (DynamicScan + Antena Wand + IPG Smart Mini), la estabilidad del sistema ante condiciones alternativas y la consistencia de los resultados frente a la aplicación oficial del cliente y la tabla de valores esperables, (*Tabla correctitud*).