Universidad ORT Uruguay

Facultad de Ingeniería

Framework para procesamiento de archivos DICOM

Entregado como requisito para la obtención del título de

Licenciado en Sistemas de información

Alejandro Horne, 171463

Camila Zinola, 205288

Juan Manuel Ferreira, 203664

Santiago Ferulano, 158632

Tutora: Jimena Saavedra

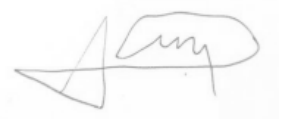
2021

Declaración de autoría

Nosotros, Alejandro Horne, Camila Zinola, Juan Manuel Ferreira y Santiago Ferulano, declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano. Aseguramos, bajo nuestra entera responsabilidad, que:

- La obra fue producida en su totalidad mientras realizamos el proyecto final de la carrera Licenciatura en Sistemas;
- Cuando hemos consultado trabajos publicados por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros autores, hemos indicado las fuentes. Con excepción de dichas 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 no pertenecientes al equipo, hemos explicado claramente qué parte fue contribuida por dichos terceros, y qué parte fue contribuida por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto los casos en que se han realizado las aclaraciones correspondientes.






Juan M. Ferreira	Santiago Ferulano	Alejandro Horne	Camila Zinola
07/10/2021	07/10/2021	07/10/2021	07/10/2021

Agradecimientos

Queremos agradecer a nuestras familias y amigos por el apoyo brindado a lo largo del presente proyecto de fin de carrera, siendo muy importante para nosotros su compañía en este camino.

También queremos agradecer a la persona que nos guió desde el comienzo del proyecto, nuestra tutora Jimena Saavedra. Desde su experiencia y dedicación nos ayudó a realizar el mejor trabajo posible en cada instancia, logrando extraer lo mejor de cada uno de nosotros. Fue importante contar con su disposición y rápida respuesta en todo momento.

Por otro lado, agradecer al laboratorio ORT Software Factory por el apoyo brindado tanto al inicio como a lo largo del proyecto. También a los revisores que nos brindaron sus consejos y comentarios en diferentes etapas del proyecto para mejorar el trabajo realizado.

Necesitamos hacer un agradecimiento y mención especial al profesor de la Universidad ORT Juan Gabito por su invaluable colaboración en momentos críticos del proyecto. Ayudó al equipo a lograr explicar la complejidad del contexto y mostró el camino a seguir para el diseño e implementación del data warehouse. Siempre estuvo dispuesto a colaborar con el equipo. ¡Muchísimas gracias Juan!.

Por último, queremos agradecer a Anthony Figueroa y Mikaela Pisani de Rootstrap por su paciencia y disponibilidad a lo largo de todo el proyecto. Por ayudarnos a aplicar los conocimientos adquiridos durante la carrera en el ámbito académico con un cliente real. Con la satisfacción que demostraban en cada una de las instancias de demostración nos inspiraron a lograr un producto que esté a la altura de sus expectativas y las del equipo.

Abstract

El proyecto fue planteado por Rootstrap [1], una Software Factory con 10 años de experiencia en el mercado, que trabaja para clientes de Estados Unidos y tiene oficinas en Uruguay.

Pensando en sus futuros proyectos, notaron que en el rubro de la medicina existe un formato de archivos llamado “DICOM”, que se utiliza para almacenar estudios de imagenología.

Rootstrap tiene como objetivo aplicar técnicas de machine learning e inteligencia artificial sobre los datos que contienen estos archivos. La problemática actual reside en que dichos archivos pueden tener un gran tamaño, ya que están compuestos por muchas imágenes y metadata, por lo que el procesamiento del archivo "crudo" se hace demasiado lento y dificultoso.

La solución que se plantea es la creación de un framework para el procesamiento de dichos archivos DICOM. Gracias a este framework, el analista de datos como usuario final sólo tiene que preocuparse por el análisis de la información disponible, aplicando las estrategias que desee de manera sencilla y rápida.

El framework se encarga de procesar y almacenar en un data warehouse la información de los archivos. Además, da la posibilidad de realizar transformaciones a los datos almacenados sin modificar los originales. El usuario final puede agregar nuevas transformaciones en base a sus necesidades y la información transformada se hace disponible en data marts para un consumo eficiente.

Las tecnologías utilizadas para el desarrollo del framework son AWS y Python.

El proyecto fue desarrollado por un equipo de 4 integrantes durante 6 meses.

Palabras clave

AWS

Data Lake

Data Mart

Data Warehouse

DICOM

Python

Índice

Glosario

Ambiente de desarrollo

Ambiente que tiene como único propósito la integración de las diferentes partes desarrolladas localmente por los miembros del equipo.

Acuerdo de Nivel de Servicio (SLAs)

Documento habitualmente anexo al Contrato de Prestación de Servicios.

Application Programming Interfaces (API)

En español significa interfaz de programación de aplicaciones.

Beta

Un estado de desarrollo dado a un programa o aplicación que contiene la mayoría de las funciones principales, pero que aún no está completo.

Brainstorming

Aportación de ideas que varias personas ponen en común como punto de partida para un proyecto.

Bug

Error o falla en cualquier programa de computadora.

Data lake

Repositorio de almacenamiento centralizado que contiene big data de varias fuentes en un formato granular y sin procesar. Puede guardar datos estructurados, semiestructurados o no estructurados, lo que significa que los datos pueden conservarse en un formato más flexible para usarlos en un futuro.

Data Mart

Base de datos centrada en un ámbito que en circunstancias es un segmento aislado de un almacén de datos de una empresa.

Entorno de desarrollo integrado (IDE)

Sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario.

Estrella (diseño data warehouse)

Modelo de datos que tiene una tabla de hechos (o tabla fact) que contiene los datos para el análisis, rodeada de las tablas de dimensiones.

Fiber Channel

Tecnología de red utilizada principalmente para redes de almacenamiento.

Git

Sistema de versionado distribuido.

Infraestructura como servicio (IaaS)

Tipo de servicio de informática en la nube que ofrece recursos esenciales de proceso, almacenamiento y redes a petición que son de pago por uso. IaaS es uno de los cuatro tipos de servicios en la nube.

Inteligencia artificial (IA)

Refiere a los sistemas o las máquinas que imitan la inteligencia humana para realizar tareas y que tienen la capacidad de mejorar iterativamente a partir de la información que recopilan.

Internet de las cosas (IoT)

Proceso que permite conectar elementos físicos cotidianos al Internet: desde objetos domésticos comunes, como las bombillas de luz, hasta recursos para la atención de la salud, como los dispositivos médicos; también abarca prendas y artículos personales e incluso los sistemas de las ciudades inteligentes.

iSCSI

Protocolo para comunicación de dispositivos.

Machine Learning (ML)

Disciplina del campo de la Inteligencia Artificial que, a través de algoritmos, dota a los ordenadores de la capacidad de identificar patrones en datos masivos y elaborar predicciones (análisis predictivo).

MVP

Minimum Viable Product o mínimo producto viable.

Open source

Todo software de libre acceso y modificación de su código fuente.

Plataforma como servicio (PaaS)

Conjunto de servicios basados en la nube que permiten a los usuarios empresariales y desarrolladores crear aplicaciones de forma rápida y rentable.

Python

Lenguaje de programación multiparadigma que soporta programación orientada a objetos, programación imperativa y programación funcional.

Repositorio

Sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

SAN

Arquitectura de redes de almacenamiento más común que utilizan las empresas para que sus aplicaciones más relevantes alcancen un alto rendimiento y una baja latencia.

Script

Programa o secuencia de instrucciones que es interpretado por otro programa.

Software

Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

Tester

Quién diseña, ejecuta e informa el resultado de las pruebas sobre un producto de software.

1. Introducción

1.1 Composición del equipo

El equipo de proyecto está integrado por cuatro estudiantes de la carrera Licenciatura en Sistemas de la Universidad ORT Uruguay. Cada uno cuenta con experiencias laborales en diferentes sectores de la industria de tecnologías de la información:

- Alejandro Horne: Desarrollador en Zarasa.
- Camila Zinola: Desarrolladora en ST Consultores.
- Juan Manuel Ferreira: Desarrollador en Rootstrap.
- Santiago Ferulano: Analista en Telefónica.

Alejandro, Camila y Juan trabajaron juntos en el ámbito académico desde el principio de la carrera y se unieron para llevar a cabo el proyecto final. Luego, en el último semestre, se incorpora Santiago para completar el equipo de cuatro integrantes para llevar a cabo el proyecto.

Los perfiles laborales de cada uno eran conocidos por el equipo, pero se tenía incertidumbre respecto a las fortalezas y debilidades de cada uno para la cantidad de roles necesarios a definir en el proyecto. Es por esto que la asignación de las tareas se realizó por consenso según lo que cada uno se sentía cómodo de ejecutar.

1.2 Elección del proyecto

Luego de la feria de proyectos organizada por la Universidad ORT el grupo se postuló a tres propuestas distintas pero ninguna se le fue asignada, por lo que quedó sin proyecto en esa etapa.

Posteriormente surgió la posibilidad de trabajar en el proyecto de la empresa Rootstrap, donde trabaja Juan Ferreira, que había planteado la necesidad en la feria. Se llevó a cabo una reunión en la que el cliente presentó al equipo la idea del proyecto de forma detallada y esta propuesta resultó desafiante en cuanto a investigación y tecnologías a utilizar ya que ninguno tenía experiencia previa trabajando con las tecnologías recomendadas por la empresa.

1.3 Descripción del cliente

Rootstrap es una software factory con oficinas en Uruguay que vende sus servicios a clientes en Estados Unidos. La empresa es reconocida tanto en el medio local como internacional, cuenta con aproximadamente 200 colaboradores y ya ha lanzado al mercado más de 750 productos digitales. La empresa se especializa en desarrollo móvil, desarrollo web, chatbots y machine learning.

1.4 Objetivos

1.4.1 Objetivos del equipo

El objetivo principal del equipo fue aplicar los conocimientos adquiridos durante la carrera de la Licenciatura en Sistemas de Información. Para definir los roles a desempeñar por cada integrante se basaron en la experiencia individual y la afinidad con las tareas relacionadas. Uno de los grandes desafíos que tuvo el equipo fue

enfrentarse a tecnologías desconocidas por todos los integrantes, lo que obliga a un gran esfuerzo en investigación y aprendizaje.

Un segundo objetivo para el equipo fue el de terminar la carrera universitaria, lo que al finalizar el proyecto de forma exitosa, resulta en la obtención del título de Licenciado en Sistemas de Información para cada uno de los integrantes.

1.4.2 Objetivos del proyecto

El objetivo del proyecto fue crear un producto alineado a las necesidades del cliente, cumpliendo con todas las características deseadas en el plazo establecido, agregando valor y generando conocimiento a los integrantes del equipo.

Esto se logró aplicando los conocimientos adquiridos durante la carrera, relevando los requerimientos de forma temprana, realizando pruebas de concepto para obtener feedback del cliente y usuario e investigando las tecnologías a utilizar de manera de poder elegir la mejor para la solución. Todo esto fue posible teniendo constante comunicación con el cliente.

2. Problema y necesidad

En esta sección se introduce la problemática a resolver, se explica la necesidad del cliente y las diferentes investigaciones que fueron necesarias realizar para analizar la viabilidad del proyecto.

2.1 Explicación del problema

Analizando futuros proyectos a desarrollar en la empresa, el cliente encontró que en el rubro de la medicina existían buenas oportunidades de negocio. Identificó la existencia de un formato de archivos específico que se utiliza para almacenar información de estudios de imagenología tales como radiografías, resonancias, tomografías, entre otros.

El problema de Rootstrap estaba relacionado con el procesamiento de estos archivos para analizar su información y así poder resolver la toma de decisiones y diagnósticos médicos. Estos archivos tienen muchas particularidades y complejidades, lo que hacen que el análisis del archivo "crudo" sea lento y dificultoso.

La solución diseñada por el equipo consta de la creación de un framework escalable y customizable para el procesamiento de estos archivos, además de un MVP (Minimum Viable Product). El framework toma los archivos "crudos", los procesa, clasifica la información según diferentes criterios y la vuelca en un data warehouse. Rootstrap consumirá la información desde ese data warehouse para aplicar técnicas de machine learning e inteligencia artificial y así lograr descubrir asociaciones de cualidades, clasificarlas según determinadas características, encontrar avances de enfermedades con el paso del tiempo e incluso poder predecir alguna de ellas.

Estas técnicas ayudan al análisis de información, se basan en valores absolutos y favorecen la eficiencia, además de ser útiles a la hora de extraer reportes.

2.2 Definición de usuario final

Los usuarios finales del sistema son analistas de datos. Estos usuarios tienen conocimiento técnico de programación en lenguajes como Python o R, cuentan con experiencia en el uso de consolas y están familiarizados con el manejo de diferentes tipos de datos, AWS y data warehouses. También tienen experiencia en inteligencia artificial, trabajando con sistemas en la nube y grandes volúmenes de datos.

2.3 Necesidades de los usuarios finales

Los usuarios finales necesitan tener disponible un data warehouse con la información procesada de los archivos, de manera de poder aplicar técnicas de machine learning e inteligencia artificial sobre la información de forma eficiente. Además, requieren que el sistema permita agregar nuevas transformaciones aparte de las ya cargadas, como por ejemplo subir o bajar el contraste de las imágenes. Este mecanismo de extensibilidad del sistema debe ejecutarse de manera simple y concreta, sin necesidad de volver a repensar todo el framework.

3. Contexto

Una vez planteado el problema y la necesidad, es importante comprender algunas definiciones acerca del contexto en el cual se trabajó.

3.1 Archivos DICOM

Los archivos que se estarán procesando y manipulando en el framework se llaman archivos DICOM [2].

Estos son archivos con un formato estándar e internacional para estudios de imagenología. Es gracias a ese formato que se pueden compartir y visualizar los archivos con la calidad y estructura necesaria para el uso clínico.

3.1.1 Estructura

La estructura del archivo se puede dividir en dos grandes partes: Metadata e Imágenes.

3.1.1.1 Metadata

En cada archivo existe una gran cantidad de metadata, en la que se puede encontrar información relativa al estudio como puede ser el tipo de estudio, el modelo de la máquina con la que se realizó, la fecha y hora de inicio, e información relativa al paciente.

Se debe tener en cuenta que dependiendo el tipo de estudio del que se esté hablando, la metadata podría variar. Por ejemplo, un estudio de imágen única (como por ejemplo, una placa) tiene una fecha y hora específica, pero un estudio de varias imágenes (como ser una resonancia magnética) puede tener una fecha y hora de inicio y otra de fin.

3.1.1.2 Imágenes

Cada archivo DICOM tiene una o muchas imágenes (dependiendo del tipo de estudio). Esto puede hacer que, dependiendo la cantidad de imágenes, los tamaños de los archivos sean muy variados.

A modo de ejemplo, se puede ver en la Imagen 3.1 una resonancia magnética de cráneo.

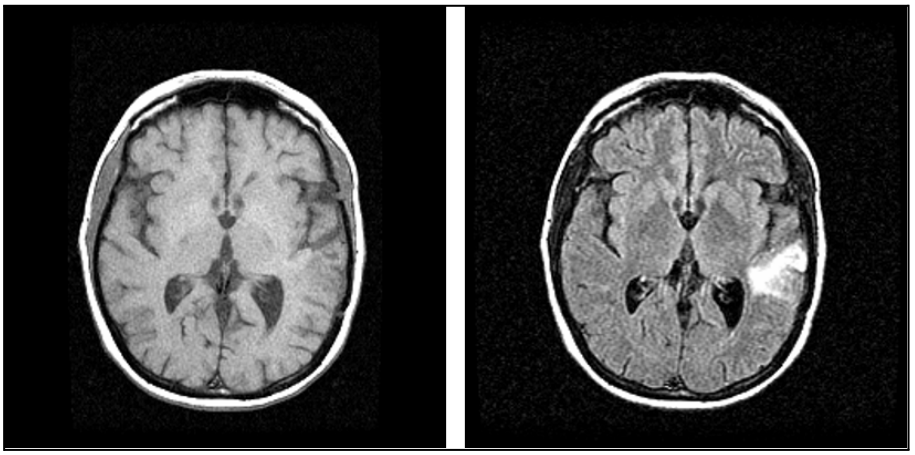


Imagen 3.1 - Resonancia magnética de cráneo.

Cuando el archivo tiene una secuencia de imágenes, también tiene una secuencia de metadata asociada que aumenta su tamaño y su complejidad.

3.1.2 Complejidades

Este formato de archivos tiene diversas complejidades, las cuales son:

- Formato especial: al tener un formato especial y particular, se debe realizar una investigación previa para entenderlo y encontrar la mejor manera de manipular la información.

- Gran tamaño: por la cantidad de imágenes y metadata que pueden llegar a tener estos archivos, su tamaño varía entre pocos MB hasta varios GB. Esto hace que el tamaño de los repositorios pueda escalar a gran velocidad.
- Información variada: dependiendo del tipo de estudio, la información puede variar mucho de un archivo a otro.
- Entendimiento de la información: al tener un formato especial y un gran tamaño, se hace complejo el trabajo de entender la información que hay dentro de cada archivo.
- Secuencias de información: los archivos pueden tener secuencias de información, lo que hace que algunos campos no tengan solo un valor, si no una lista de valores que se deben estudiar para conocer de qué es lo que tratan.

3.2 Tipos de datos

En el universo de los datos existen tres tipos de datos: estructurados, semiestructurados y no estructurados. En los archivos DICOM se pueden encontrar los últimos dos.

3.2.1 Semiestructurados

Los datos semiestructurados de los archivos DICOM son la porción de la metadata. Como se indicó anteriormente, si bien se tiene una parte de metadata que es común para todos los tipos de estudios, existe otra que varía. También, cuando existen secuencias de información, hay atributos que pueden tener un solo valor o una lista de valores. Esto hace que la metadata sea un tipo de dato semiestructurado.

3.2.2 No estructurados

Las imágenes que contienen los archivos DICOM son los datos no estructurados. Todas las imágenes son diferentes y no tienen una estructura definida.

3.3 Repositorios de información

En el proyecto se utilizaron tres tipos de repositorio dependiendo el tipo de información que se desea almacenar: data lake, data warehouse y data mart.

3.3.1 Data Lake

En un data lake se puede guardar cualquier tipo de información, ya sea estructurada, semiestructurada o no estructurada. En este caso se utiliza como almacén para los archivos DICOM que se procesarán inicialmente y para almacenamiento de las imágenes luego de procesar los archivos.

3.3.2 Data Warehouse

El resultado del procesamiento de los archivos DICOM se almacena en dos sitios diferentes. Por un lado, la metadata se almacena en un data warehouse y servirá como “fuente de la verdad”. Por otro lado están las imágenes, que se almacenan en un data lake y en el data warehouse se almacena una relación con un link a cada imagen de dicho data lake.

Este data warehouse es el que luego utilizarán los analistas de datos como punto de partida para sus procesos de análisis de la información.

3.3.3 Data Mart

Un Data Mart es una base de datos que contiene un subconjunto de la información total que se posee y en este caso se utilizará para almacenar la información

transformada. El usuario final puede querer aplicar modificaciones sobre la información extraída de los archivos y, para no modificar esos datos originales, el resultado se almacenará en estos data marts.

3.4 Análisis de datos

El proceso de análisis de datos consta de cinco partes:

1 - Extracción de los datos.

2 - Transformación de los datos.

3 - Carga de los datos.

4 - Análisis de los datos.

5 - Generación de informes.

Los referentes en Rootstrap indican que los primeros tres pasos ocupan el 80% del tiempo de un analista de datos, dejando el 20% para las tareas de Análisis y generación de informes que son propias de su función (ver Imagen 3.2).

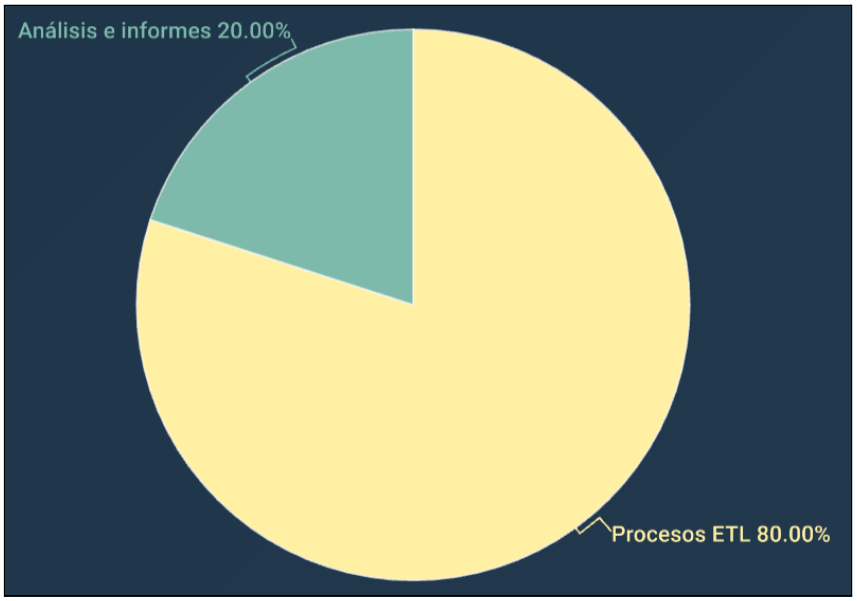


Imagen 3.2 - Análisis de datos.

Analizando la Imagen 3.2, el framework creado automatiza los pasos de color amarillo, ahorrando ese 80% del tiempo que los analistas de datos dedican a estas tareas. Como resultado hará disponible la información en un data warehouse que oficia de punto de partida para comenzar el paso de análisis y la posterior generación de informes.

4. Solución

En esta sección se detalla cuál es el alcance que tuvo el proyecto, se explica la solución creada para la situación planteada por el cliente y cómo cada una de sus partes resuelve una necesidad distinta.

4.1 Alcance

Se definieron los límites del proyecto para comprender el inicio y el fin. A su vez, se dejó explícito cuáles son las entradas y salidas necesarias para que el framework pueda ejecutarse.

La ejecución del framework parte de un repositorio con archivos DICOM, desde el cual se consumen los archivos, se procesa y transforma la información que contienen y finalmente se carga en un data warehouse. Utilizando esta información del data warehouse es que los analistas de datos realizan sus procesos de análisis.

En cuanto a los tipos de estudio soportados por el framework, el alcance del proyecto abarca soportar un solo tipo de archivo DICOM, dejando la estructura necesaria para que Rootstrap pueda extender el framework y agregar soporte para más tipos de archivos.

De igual forma que los tipos de archivos, el alcance del proyecto incluye crear la estructura en el data warehouse para que Rootstrap pueda extenderla sin necesidad de cambiar la lógica de la aplicación.

Finalmente, con respecto a las transformaciones, el alcance del proyecto incluye que se debe implementar un tipo de transformación de imágenes y un tipo de transformación de consultas SQL, dejando creada la estructura para su extensión.

4.1.1 Supuestos

- El usuario provee una cuenta de AWS con un bucket de S3 que contiene los archivos DICOM que se desea procesar, este bucket funciona como punto de partida para el sistema. A su vez este mismo bucket S3 se utiliza para el almacenamiento de las imágenes en la etapa de procesamiento del framework.
- El análisis de la información procesada será hecho por el usuario final, quedando por fuera del alcance de este proyecto.
- Las nuevas transformaciones que se deseen agregar deberán ser implementadas por el usuario final.

4.1.2 Restricciones

- El procesamiento de los archivos DICOM debe ser en la nube debido a su gran tamaño.
- Es prioritario para el cliente que el lenguaje de programación utilizado sea Python, ya que hoy en día cuenta con equipos de desarrollo con conocimientos en dicho lenguaje.
- Por ser un proyecto académico, la fecha de entrega está preestablecida y sin posibilidad de modificación. La fecha límite para la entrega es el 7 de octubre del 2021.

4.2 Fuera de alcance

El alcance del proyecto no involucra la puesta en producción del código. Esto fue definido por el cliente en base a sus objetivos estratégicos anuales y dada la duración del proyecto académico no es viable realizar el despliegue a producción.

De todas formas, la migración a un ambiente productivo está detallado en el plan de transición disponible en el Anexo 1.

También se encuentra por fuera del alcance el desarrollo de la interfaz de usuario. Esto se debe a que el usuario final cuenta con conocimientos suficientes para poder ejecutar el framework utilizando la consola seleccionada para el desarrollo. Esta decisión se basa en que los procesos a ejecutar pueden demorar demasiado tiempo (dependiendo de la cantidad de archivos DICOM a procesar), además de no tener la necesidad de navegar sobre un sitio o página web para poder realizar las tareas. Fue una decisión de común acuerdo con el cliente.

4.3 Entregables

Se acordó con el cliente que los entregables serán tres, cada uno de ellos con diferentes objetivos y características.

4.3.1 Framework

El primer entregable acordado es el framework para el procesamiento de los archivos DICOM. Dicho framework será desarrollado en el lenguaje de programación Python [3]. Constará de dos grandes partes: procesamiento y transformación.

4.3.1.1 Procesamiento

Esta parte del sistema es ejecutada por un analista de datos y consta de un ETL (Extract, Transform & Load), el cual toma los archivos DICOM desde la fuente de datos proporcionada por el usuario, particiona la información del archivo en metadata e imagenes y clasifica la metadata.

Particularmente para la metadata extrae los datos para asociarla a las distintas tablas de hechos y dimensiones previamente creadas en el data warehouse. Por otro lado, para las imágenes es necesario realizar un cambio de formato desde pixel data (formato original) a PNG ya que es uno de los formatos más utilizado para análisis de datos según indica nuestro referente en Rootstrap. Luego del cambio de formato se almacenan en el data lake. Como las imágenes no se almacenan en el data warehouse, se debe guardar en él una referencia a cada una de ellas de manera de poder mantener la asociación con su metadata.

4.3.1.2 Transformación

Las transformaciones también son ejecutadas por un analista de datos y tienen como punto de partida la información del data warehouse.

Este proceso del framework se utiliza cuando un analista de datos tiene la necesidad de realizar algún tipo de modificación sobre la información almacenada en etapa de procesamiento explicada anteriormente.

Se pueden ejecutar transformaciones existentes o crear nuevas. En el caso de necesitar una nueva transformación, se debe codificar la transformación concreta y luego el proceso es exactamente el mismo que en una transformación existente, el cual se explicará a continuación.

Al ejecutar una transformación, el framework copia la información que se indica en el código, realiza la transformación deseada y carga un data mart con la información transformada. Cabe destacar que la información original no se ve alterada en este proceso ya que es la “fuente de la verdad” y bajo ningún concepto debe ser modificada.

4.3.2 Guías de usuario

Se acordó con el cliente la creación de guías de usuario para ejecución del framework, desarrollo de nuevas transformaciones y mantenimiento del código fuente para poder expandirlo.

4.3.2.1 Ejecución del framework

Para ver la guía de ejecución del framework ir al Anexo 2.

4.3.2.2 Agregar una nueva transformación

Para ver la guía para agregar una nueva transformación sobre imágenes ir al Anexo 3 y para transformaciones SQL ir al Anexo 4.

4.3.2.3 Agregar una nuevas tablas de agrupación

Para ver la guía para agregar una nueva tabla de agrupación (codiguera) ir al Anexo 5.

4.3.3 Investigación de plataformas en la nube

Se realizó un documento de investigación de las plataformas en la nube que se compartió con el cliente. El objetivo de dicha investigación fue decidir, en conjunto con el cliente, la mejor plataforma para la realización del framework según los criterios de utilidad para el propósito deseado: costos y rendimiento. El detalle de la investigación realizada se puede ver en el próximo capítulo.

5. Investigación

5.1 Introducción

En vistas del contexto en el que se trabajó, es que se observó la necesidad de comparar distintos servicios en la nube que puedan proveer la infraestructura adecuada para el desarrollo del proyecto.

Para lograr elegir un número acotado de servicios a investigar, el equipo tomó como base el último informe realizado por la consultora Gartner Consulting, la cual es una de las consultoras de TI más prestigiosas del mundo. En el informe realizado por Gartner Consulting, se seleccionan cuidadosamente los proveedores de servicios en la nube y se incluyen o excluyen del análisis en base a varios criterios. Algunos de esos criterios son: participación en el mercado, impulso y tracción del mismo, capacidades comerciales, si tienen ofertas IaaS y PaaS, tipos de clientes que utilizan sus servicios, estándares de seguridad, SLAs de disponibilidad (99,95% o más), ofrecimiento de un portal, documentación, soporte técnico y atención al cliente.

La consultora posiciona a los proveedores seleccionados en un “Cuadrante Mágico” [4] que se divide de la siguiente manera:

- Leaders: Los líderes se distinguen por ofrecer un servicio adecuado para la adopción estratégica y tener una hoja de ruta ambiciosa.
- Challengers: Los retadores están bien posicionados para satisfacer algunas de las necesidades actuales del mercado.

- Visionaries: Los visionarios tienen una visión ambiciosa del futuro y están realizando importantes inversiones en el desarrollo de tecnologías únicas.
- Niche players: Pueden ser excelentes proveedores para casos de uso particulares o en las regiones en las que operan, pero, en última instancia, deben considerarse proveedores especializados de IaaS en la nube.

En la siguiente imagen se puede ver el recuadro de los servicios analizados:

Figura 5.1 - Cuadrante mágico de Gartner.

En este informe se acotó la investigación a Amazon Web Services (AWS), Google Cloud Platform (GCP) y Microsoft Cloud Platform (MCP), los tres líderes en la industria.

5.2 Cuotas de mercado

Amazon Web Services (AWS) fue el proveedor líder de servicios en la nube en el primer trimestre de 2021, con un crecimiento anual del 31%, tiene un 32% de cuota

del mercado. Sin embargo, su crecimiento es menor que el cosechado por sus dos rivales directos.

Microsoft Azure creció un 50% por tercer trimestre consecutivo y acapara ya una participación de mercado del 20%. Mientras, Google Cloud creció un 56% en el último trimestre para representar una participación de mercado del 9% [5].

5.3 Comparativa general de los servicios

	Amazon	Microsoft	Google
Nacimiento del servicio	2006	2010	2011
Backups	3 copias en la misma zona geográfica. Posibilidad de replicar copias en otras zonas.	3 copias en la misma zona geográfica. Posibilidad de replicar copias en otras zonas.	Por defecto realiza copias en todas las plataformas alrededor del mundo.
Disponibilidad mundial	11 centros de datos.	20 centros de datos.	4 centros de datos.
Marketplace	2400 aplicaciones.	707 aplicaciones.	160 aplicaciones.
Soporte	Soporte gratuito usando base de conocimientos. Posibilidad de soporte personalizado con costo.	Soporte gratuito usando base de conocimientos. Posibilidad de soporte personalizado con costo.	Soporte gratuito usando base de conocimientos. Posibilidad de soporte personalizado con costo.
Tipos de servidores	53	25	18
Tipos de discos	Clásicos. SSD. Se pueden personalizar.	Clásicos. SSD. NO se pueden personalizar.	Clásicos. SSD. Se pueden personalizar.
Seguridad	20 certificaciones.	25 certificaciones.	6 certificaciones.
Estabilidad	99,95% de disponibilidad mensual.	99,95% de disponibilidad mensual.	99,95% de disponibilidad mensual.

Tabla 5.1 - Comparativa de servicios.

5.4 Solución

Para el desarrollo del framework se necesitaron principalmente dos plataformas de servicios en la nube: una plataforma para el almacenamiento de las imágenes y un data warehouse para la información procesada.

Para el almacenamiento se encuentran las siguientes plataformas:

- Amazon S3 (AWS).
- Google Cloud Storage (GCP).
- Microsoft Blob Storage (MCP).

Como data warehouse se encuentran las siguientes:

- Redshift (AWS).
- BigQuery (GCP).
- Microsoft Cosmos (MCP).

5.4.1 Almacenamiento

Se pueden dividir en tres categorías:

- Almacenamiento de objetos: es un sistema diseñado para datos no estructurados que los trata como objetos de información granular en lugar de como parte de un archivo o volumen.
- Almacenamiento de archivos: es análogo a los sistemas de archivos utilizados en PC y servidores que almacenan datos en archivos organizados en directorios jerárquicos.
- Almacenamiento en bloque o volumen: es el equivalente en la nube de un volumen de disco sin formato al que se accede a través de una SAN, aunque se accede a través de API en la nube en lugar de los protocolos iSCSI o Fibre Channel.

5.4.1.1 Amazon S3

Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento en la nube que es escalable, basado en la web y de alta velocidad. Está diseñado para realizar copias de seguridad en línea y archivar datos y aplicaciones en *AWS*.

A su vez, cuenta con una API que se ha convertido en un estándar compatible con la gran mayoría de aplicaciones y servicios de terceros.

AWS ofrece la gama más amplia de servicios en almacenamiento, computación, base de datos, análisis, redes, dispositivos móviles, herramientas de administración, herramientas para desarrolladores, IoT, seguridad y aplicaciones empresariales.

S3 proporciona una disponibilidad del 99,999999999% para los objetos almacenados en el servicio y admite varias certificaciones de seguridad y cumplimiento. Un administrador también puede vincular S3 a otros servicios de seguridad y monitoreo de AWS.

Amazon S3 puede ser utilizado por organizaciones de distintos tamaños, desde muy pequeñas hasta gigantes. Las capacidades de escalabilidad, disponibilidad, seguridad y rendimiento de S3 lo hacen adecuado para una variedad de casos de uso de almacenamiento de datos. El que más importa en este caso es el almacenamiento de datos no estructurados, como son las imágenes de nuestros archivos DICOM.

5.4.1.2 Google Cloud Storage

Este servicio de Google combina rendimiento con escalabilidad en la nube con capacidades avanzadas de seguridad y uso compartido.

Está creado para que pueda ser utilizado por empresas de todos los tamaños y almacena cualquier cantidad de datos. A su vez cuenta con accesibilidad mundial y ubicaciones de almacenamiento en todo el mundo, latencia baja, disponibilidad anual del 99,999999999% y redundancia geográfica si los datos se almacenan en una multiregión.

Las características clave de este servicio son:

- Varias opciones de redundancia.
- Facilidad de transferencia.
- Distintos tipos de almacenamiento.

5.4.1.3 Microsoft Blob Storage

Blob Storage permite a los usuarios almacenar grandes cantidades de datos no estructurados en la plataforma de almacenamiento de datos de Microsoft. En este caso, Blob significa Binary Large Object, que incluye objetos como imágenes y archivos multimedia.

La velocidad, la escalabilidad, la facilidad de acceso y la seguridad (tanto de accidentes como de hackers) hacen que dicho almacenamiento en la nube sea muy atractivo para todas las organizaciones medianas y grandes.

5.4.2 Precios por almacenamiento

El precio de los servicios en la nube es bastante complejo. Se debe tener en cuenta que cada proveedor tiene una metodología única de facturación y precios con innumerables variables.

Se hizo foco en los servicios básicos de cada uno, ya que no era necesaria una configuración con características que no aplicarían al contexto.

5.4.2.1 AWS

Los precios de AWS son los siguientes:

S3			
Almacenamiento	Primeros 50 TB/mes	0,04050	USD por Gb
	Siguientes 450 TB/mes	0,03900	USD por Gb
	Más de 500 TB/mes	0,03700	USD por Gb
Procesamiento	PUT, COPY, POST y LIST (x1000 solicitudes)	0,00700	USD
	GET, SELECT y todas las demás (x1000 solicitudes)	0,00056	USD

Tabla 5.2 - Precios AWS.

5.4.2.2 Google

Los precios de Google Cloud Storage son los siguientes:

Google Cloud Storage			
Almacenamiento	Sin franjas/mes	0,03500	USD por Gb
Procesamiento	Operaciones Clase A (x 10000 solicitudes)	0,05000	USD
	Operaciones Clase B (x 10000 solicitudes)	0,00400	USD
	Comentario: Las operaciones Clase A y Clase B son análogas a las que establece AWS en cada franja de procesamiento.		

Tabla 5.3 - Precios Google Cloud Storage.

5.4.2.3 Microsoft Azure Blob Storage

Los precios de Blob Storage son los siguientes:

Microsoft Azure Blob Storage			
Almacenamiento	Primeros 50 TB/mes	0,0424	USD por Gb
	Siguientes 450T/mes	0,0407	USD por Gb
	Más de 500 TB/mes	0,0390	USD por Gb
Procesamiento	Operaciones de escritura (en decenas de millar)	0,1183	USD
	Operaciones de lectura (en decenas de millar)	0,0095	USD
	Operaciones iterativas de lectura (x10000 solicitudes)	0,0095	USD
	Operaciones iterativas de escritura (x100 solicitudes)	0,1183	USD
	Index (Gb/mes)	0,078	USD
	Resto de las operaciones (en decenas de millar)	0,0095	USD

Tabla 5.4 - Precios Microsoft Blob Storage.

Para visualizar los costos en escenarios específicos, se crearon tres casos de uso distintos con diferentes configuraciones de almacenamiento y procesamiento (ver Tabla 5.5).

Caso de uso A			Caso de uso B			Caso de uso C		
Almacenamiento	10	TB/mes	Almacenamiento	100	TB/mes	Almacenamiento	510	TB/mes
Escritura	50000	Solicitudes	Escritura	500000	Solicitudes	Escritura	50000	Solicitudes
Lectura	50000	Solicitudes	Lectura	500000	Solicitudes	Lectura	50000	Solicitudes

Tabla 5.5 - Casos de uso.

Luego, en cada escenario planteado se aplicaron los precios de cada una de las plataformas. Se puede ver el costo de cada escenario por plataforma en la Tabla 5.6.

S3					
Caso de uso A		**Caso de uso B**		**Caso de uso C**	
Almacenamiento	414,72	Almacenamiento	3993,60	Almacenamiento	19322,88
Escritura	0,35	Escritura	3,50	Escritura	0,35
Lectura	0,03	Lectura	0,28	Lectura	0,03
TOTAL	**415,10**	TOTAL	**3997,38**	TOTAL	**19323,26**

Google Cloud Storage					
Caso de uso A		**Caso de uso B**		**Caso de uso C**	
Almacenamiento	358,40	Almacenamiento	3584,00	Almacenamiento	18278,40
Escritura	0,25	Escritura	0,10	Escritura	
Lectura	0,02	Lectura	0,01	Lectura	
TOTAL	**358,67**	TOTAL	**3584,11**	TOTAL	**18278,40**

Microsoft Azure Blob Storage					
Caso de uso A		**Caso de uso B**		**Caso de uso C**	
Almacenamiento	434,18	Almacenamiento	4167,68	Almacenamiento	20367,36
Escritura	0,05	Escritura	0,48	Escritura	0,05
Lectura	0,59	Lectura	5,92	Lectura	0,59
TOTAL	**434,82**	TOTAL	**4174,07**	TOTAL	**20368,00**

Tabla 5.6 - Precios en dólares para cada escenario según la plataforma.

5.4.3 Data Warehouses

5.4.3.1 Redshift

Amazon Redshift es un almacén de datos que forma parte de la plataforma de computación en la nube Amazon Web Services (AWS). Lanzado por primera vez hace 8 años, Redshift permite a los usuarios empresariales comenzar con una pequeña cantidad de datos, unos pocos cientos de gigabytes, y pasar a una escala de petabytes de datos, todo habilitado con el poder de la nube.

Redshift, una herramienta de BI rápida y rentable para todos los datos que utiliza SQL estándar, almacena los datos en un clúster o en forma de bloque, lo que permite consultas más rápidas y sin problemas en comparación con los almacenes de datos tradicionales o locales.

A diferencia de los almacenes de datos tradicionales, Redshift permite realizar consultas analíticas complejas desde gigabytes hasta petabytes de datos estructurados y semiestructurados. Utiliza una sofisticada optimización de consultas, almacenamiento en columnas y ejecución de consultas en paralelo en varios recursos físicos.

Al utilizar clientes conocidos basados en SQL y herramientas de BI que utilizan la Conectividad abierta de bases de datos (ODBC) y la Conectividad de bases de datos Java (JDBC) estándar, Redshift promete capacidades de consulta impresionantemente rápidas, la mayoría de las cuales regresan en segundos.

Características:

- Rápido y rentable.
- Almacena datos en formato de columnas.
- Admite integraciones de IA y ML.
- Menos de la décima parte del costo de las soluciones locales tradicionales.
- Ejecución de consultas en paralelo.
- Permite consultas SQL directamente.
- Seguro con la integración de AWS IAM, control de acceso a nivel de columna, VPC, cifrado y más.

- Utiliza replicación y copias de seguridad continuas para mejorar el rendimiento.

5.4.3.2 BigQuery

Google BigQuery es otro almacén de datos codiciado bajo la propia infraestructura en la nube de Google Cloud. Este almacén de datos está diseñado para manejar datos voluminosos al tiempo que garantiza agilidad y escalabilidad.

Comparte la infraestructura en la nube de Google, por lo que puede paralelizar cada consulta y ejecutarla en decenas de miles de servidores simultáneamente. Al aprovechar la infraestructura computacional y en la nube de Google, los beneficios incluyen la replicación múltiple en todas las regiones y una alta escalabilidad del centro de datos, sin necesidad de ninguna administración por parte del desarrollador.

BigQuery también almacena datos en formato de columnas. Cada columna de la tabla se almacena en un bloque de archivo separado y todas las columnas se almacenan en un solo archivo de condensador que luego se comprime y encripta.

Permite el análisis de datos a través de la nube utilizando SQL estándar. Está completamente administrado y dirigido al procesamiento de datos a gran escala para empresas. Proporciona controladores ODBC y JDBC y por último, proporciona una integración de datos sin problemas con herramientas de Google.

Características:

- Compatible con grandes conjuntos de datos.
- Admite integraciones de IA y ML.

- Facilidad de uso para no programadores.
- Se basa en la propia infraestructura en la nube de Google.
- Utiliza un sistema de archivo para el almacenamiento de datos.
- Rápido y autogestionado.

5.4.3.3 Microsoft Cosmos

Se decidió no investigar en profundidad este data warehouse ya que el almacenamiento de Microsoft resultó ser el más costoso entre los tres que se realizó la comparación.

5.4.4 Seguridad en data warehouses

Cuando se trata de seguridad, Redshift y BigQuery son similares. Redshift usa Amazon IAM para la identidad y BigQuery usa Google Cloud IAM.

5.4.5 Integraciones en data warehouses

Tanto Google como Amazon tienen una gran cantidad de integraciones disponibles. Las principales herramientas de análisis de datos y BI funcionan con ambos almacenes. Redshift se basa en una bifurcación de PostgreSQL, por lo que originalmente tenía más integraciones nativas, pero este aspecto se ha nivelado debido al gran volumen de transacciones de almacén que procesa Google.

5.4.6 Precio en data warehouses

Redshift	
Almacenamiento	306 x TB x mes
Procesamiento	Gratuito

BigQuery	
Almacenamiento	20 x TB x mes
Procesamiento	9 x TB x mes

Tabla 5.7 - Precios de data warehouses.

Todos los valores están expresados en dólares americanos.

De la Tabla 5.7 se desprende que si se almacena 1TB y se procesan menos de 32TB en un mes es conveniente BigQuery. Sin embargo si se almacena 1TB y si se procesan más de 32TB conviene Redshift. Este escenario cambia si el almacenamiento es de más de 1TB ya que el costo de almacenamiento en Redshift es cada 1TB.

5.5 Conclusiones

Como resultado de este informe, se decidió en conjunto con Rootstrap optar por la utilización de los servicios de AWS para la implementación de nuestro framework.

Los motivos de la decisión son:

- AWS ofrece los tipos de plataformas que se necesitan (S3 y Redshift).
- AWS ofrece más de 2400 aplicaciones que solucionan necesidades que puedan surgir a futuro (AWS Lambda, EC2, AWS Elastic Beanstalk y AWS Glue son las más destacables).

- AWS es el más antiguo (está en el mercado desde 2006) y es el que tiene mayor cuota del mercado (con un 32%). Esto lo hace un servicio más maduro y robusto que el de sus rivales [6].
- A pesar de no ser el líder en cuanto a costos, los precios de AWS en los diferentes escenarios planteados no superan en más del 10% a los precios del líder que es GCP (Ver Tabla 5.6).
- El cliente ya cuenta con servicios contratados de AWS (S3 para almacenamiento de los archivos DICOM crudos).

En resumen, AWS tiene las habilidades, los recursos y la motivación para integrarse verticalmente y ofrecer soluciones a los clientes de un extremo a otro. La empresa puede diseñar desde sus servidores hasta los sistemas operativos integrados en los dispositivos periféricos y la pila completa de software en el medio.

Sin embargo, si a futuro se desea optimizar costos se recomienda evaluar la posibilidad de migrar a Google Cloud, ya que del resultado de la investigación se desprende que es el líder en cuanto a costos de almacenamiento y procesamiento en sus servicios de la nube.

6. Ingeniería de requerimientos

6.1 Proceso

El proceso de ingeniería de requerimientos que se utilizó en este proyecto consta de cuatro etapas.

La primera etapa fue la de relevamiento, la cual se llevó adelante con reuniones con el cliente. En esta instancia el equipo tomó conocimiento de los requerimientos funcionales y no funcionales solicitados.

La siguiente instancia que se realizó fue la priorización de cada uno de esos requerimientos, definiendo con el cliente cuáles eran esenciales para que el producto cumpla con las necesidades esperadas por el usuario final.

La etapa de verificación de requerimientos por parte del cliente se realizó mediante la aprobación de un documento en el que se listaron todos los requerimientos relevados y priorizados. En conjunto con los referentes en Rootstrap se analizó el documento y se aprobó. La instancia fue realizada en una reunión virtual.

Finalmente se realizó la etapa de especificación, la cual contó con una investigación por parte del equipo para definir si la especificación se realizaba mediante casos de uso o historias de usuario. Por las características del proyecto se decidió utilizar historias de usuario.

Debido al poco conocimiento que se tenía en el dominio del proyecto y a las herramientas que se debían usar a lo largo del desarrollo, fue necesario iterar varias veces en las etapas nombradas anteriormente para lograr establecer los

requerimientos, ya que en las primeras instancias resultaban ambiguos o poco claros.

En la Imagen 6.1 se puede ver el proceso de ingeniería de requerimientos mencionado anteriormente.

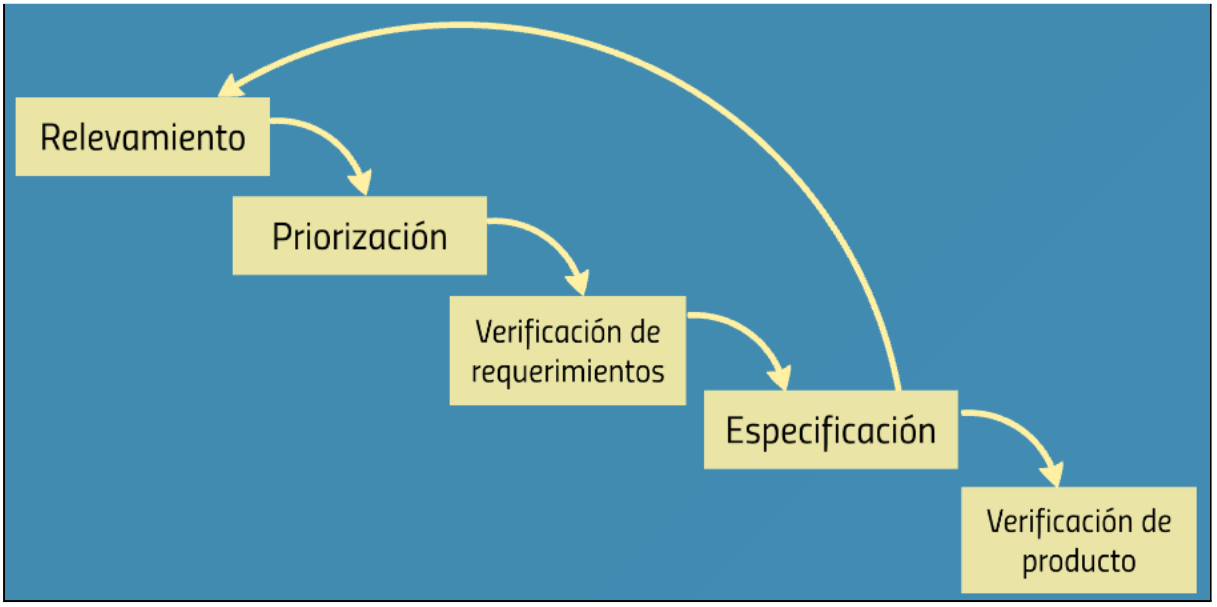


Imagen 6.1 - Proceso de ingeniería de requerimientos.

Cómo se explica en el capítulo 10.1.2, el proceso de requerimientos se realizó al inicio del proyecto.

6.2 Verificación de producto

Una vez que se avanzó en el proceso de desarrollo del framework, se llevaron a cabo instancias de verificación de producto. Estas se dieron cuando se cumplieron los siguientes hitos de desarrollo: procesamiento de metadata, procesamiento de imágenes, pruebas de aceptación con los usuarios finales y cierre del proyecto. En la sección 14.2.3.3 se detallan estas instancias.

6.3 Requerimientos

6.3.1 Requerimientos funcionales

RF1 - Conectar un bucket S3.

Usuario: Analista de datos.

Narrativa: Como usuario quiero poder conectarme a un bucket S3 que contenga mis archivos DICOM para acceder a su información.

RF2 - Consumir desde un bucket S3.

Usuario: Analista de datos.

Narrativa: Como usuario quiero poder consumir mis archivos DICOM desde un bucket S3 para procesar la información que contienen.

RF3 - Preprocesar archivos DICOM.

Usuario: Analista de datos.

Narrativa: Como usuario quiero que el sistema separe los datos que hay en cada archivo DICOM, diferenciando entre imágenes y metadata para cargar su información en los repositorios adecuados.

RF4 - Clasificar archivos en secuenciales y no secuenciales.

Usuario: Analista de datos.

Narrativa: Como usuario quiero que el sistema clasifique los archivos en secuenciales y no secuenciales para poder asociarlos a los que correspondan a un mismo estudio de una persona.

RF5 - Preprocesar los datos sin descargarlos localmente.

Usuario: Analista de datos.

Narrativa: Como usuario quiero que el sistema preprocese los datos sin utilizar el almacenamiento local para lograr un procesamiento más eficiente que localmente.

RF6 - Conectar a Redshift.

Usuario: Analista de datos.

Narrativa: Como usuario quiero poder acceder a Redshift, donde se almacenarán todos los datos preprocesados (imágenes y metadata) de los archivos DICOM para almacenar los datos procesados por el framework.

RF7 - Realizar carga en Redshift.

Usuario: Analista de datos.

Narrativa: Como usuario quiero poder almacenar la información preprocesada de los archivos DICOM (imágenes y metadata) en Redshift para tener la información cargada en un data warehouse.

RF8 - Generar diseño del data warehouse.

Usuario: Analista de datos.

Narrativa: Como usuario quiero crear el diseño del data warehouse automáticamente al procesar archivos DICOM para realizar consultas y análisis sobre los datos cargados.

RF9 - Disponibilizar información.

Usuario: Analista de datos.

Narrativa: Como usuario quiero tener disponible la información (tanto la transformada como la original) para realizar los análisis deseados.

RF10 - Agregar transformaciones sobre los datos.

Usuario: Analista de datos.

Narrativa: Como usuario quiero poder agregar transformaciones sobre los datos del data warehouse sin modificar el código existente para contar con nuevas transformaciones y no introducir errores.

RF11 - Ejecutar transformaciones sobre los datos.

Usuario: Analista de datos.

Narrativa: Como usuario quiero poder ejecutar transformaciones sobre los datos sin modificar los datos originales para contar con los datos transformados y los originales. El resultado de la transformación se hará disponible en un data mart.

RF12 - Carga de archivos correctos.

Usuario: Analista de datos.

Narrativa: Como usuario no podré cargar archivos que generen errores en el proceso de carga o inconsistencias en la información del data warehouse para no introducir errores.

6.3.2 Requerimientos no funcionales

En esta sección se listan los requerimientos no funcionales y los atributos de calidad asociados a cada uno, en el capítulo 7.2, se hace referencia a los mecanismos utilizados para cumplir con estos atributos.

RNF1 - El sistema debe permitir incorporar nuevas transformaciones.

Atributo de calidad asociado: extensibilidad.

RNF2 - Se requiere acceder al sistema desde cualquier PC.

Atributo de calidad asociado: adaptabilidad.

RNF3 - El sistema debe permitir realizar consultas sobre el data warehouse de manera eficiente.

Atributo de calidad asociado: performance.

RNF4- Transformación en paralelo para garantizar tiempos de respuesta adecuados.

Atributo de calidad asociado: performance.

7. Arquitectura

7.1 Introducción

El proyecto parte de un repositorio S3 donde se almacenan los archivos DICOM sin procesamiento previo. El framework obtiene los archivos desde dicho repositorio, los procesa y carga los datos en un data warehouse y las imágenes en un data lake. Además, sobre la información del data warehouse puede realizar transformaciones, las que como salida pueden tener un data lake o un data mart dependiendo de las salidas hayan sido codificadas en la transformación seleccionada.

Por ejemplo, en un hipotético caso en el que el usuario desee procesar un solo archivo que ya está cargado en el repositorio S3, se ejecuta el comando de procesamiento de los archivos y dicho proceso los extrae del repositorio, los procesa y finalmente carga su contenido en los repositorios de salida. La información de este archivo se separa en imágenes y metadata. La metadata se almacena en Redshift, mientras que las imágenes se almacenan en el repositorio S3 y en Redshift se registra un link a esa imagen, manteniendo la trazabilidad de la información y sus imágenes (ver Imagen 7.1).

Se invirtió tiempo en el diseño del data warehouse, de hecho corroborando con la herramienta de planificación, se visualiza que se dedicaron 49 horas entre los ciclos 4 y 5 dedicando cerca de un 30% de los mismos. Se tuvo especial cuidado en la tarea de diseño del data warehouse para lograr que las tablas de hechos y dimensiones se adapten a la información de todos los tipos de estudio. Existen múltiples tablas por cada uno de los archivos.

En las primeras instancias donde se intentó explicar el problema, se cometió el error de transmitir la etapa de procesamiento de archivos en el framework como algo lineal y el corrector interpretó que el procesamiento iba a ser plano. Luego, en la siguiente instancia se logró modificar el enfoque de la explicación, dando a entender cómo es realmente.

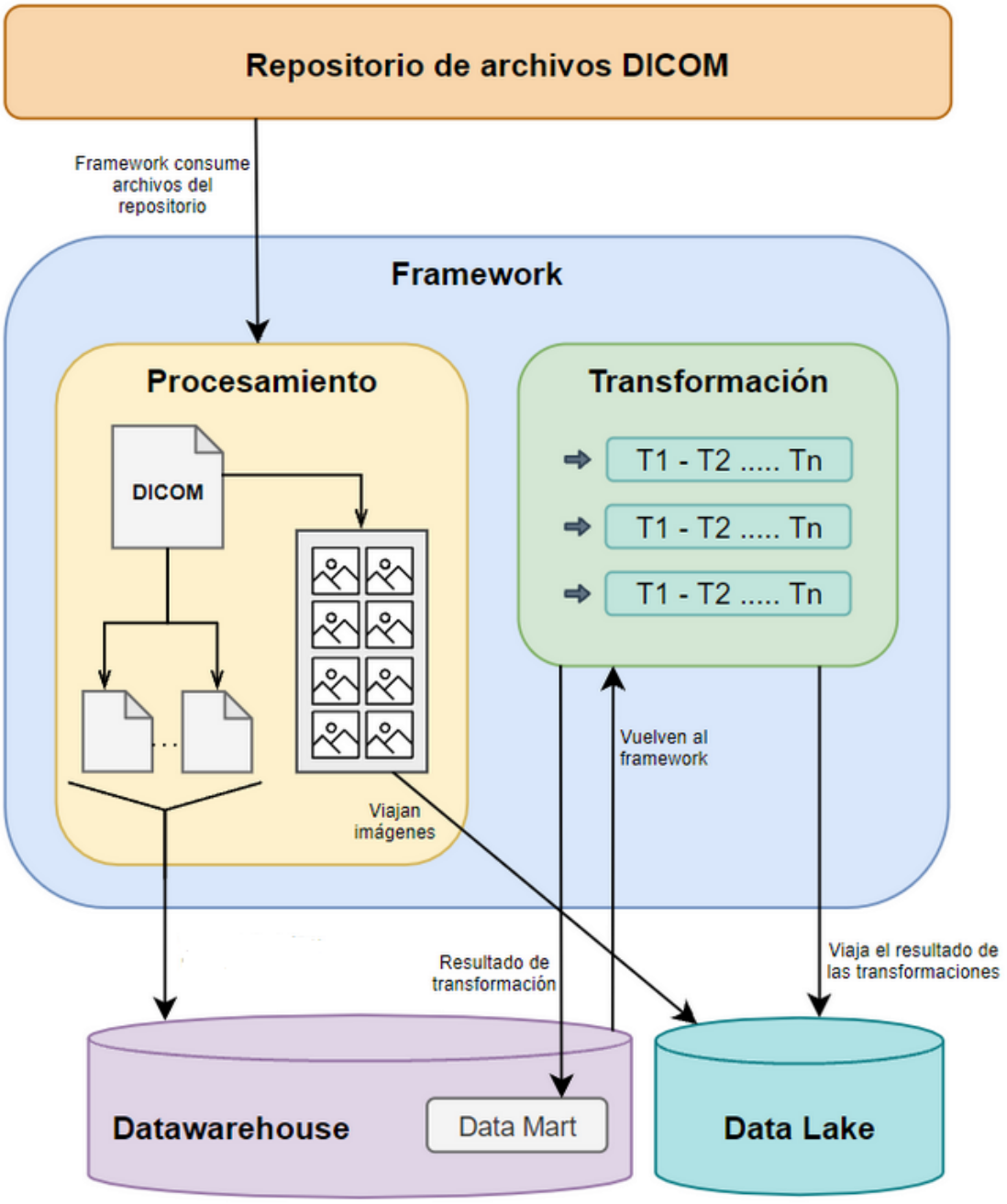


Imagen 7.1 - Arquitectura genérica de la solución.

7.2 Atributos de calidad

7.2.1 Extensibilidad

RNF1 - Permitir nuevas transformaciones.

La mantenibilidad refiere a la capacidad del software para poder modificarse de forma sencilla y rápida como consecuencia de necesidades nuevas o integraciones que se deseen realizar.

El cliente cuenta hoy en día con un repositorio S3 en el cual, se realiza la carga de datos para que sea consumido por Redshift.

Mecanismo para su cumplimiento: El el framework se diseñó de manera que si se desea agregar nuevo comportamiento no se deba modificar el código existente y solo se deba agregar el nuevo código referente al nuevo comportamiento deseado (SOLID - Open/Close Principle) [7].

7.2.2 Adaptabilidad

RNF2 - Posibilidad de ejecutar en cualquier PC.

Mecanismo para su cumplimiento: El único lenguaje de programación utilizado para el desarrollo del framework es Python 3 y al utilizar AWS se podrá acceder desde cualquier PC con internet.

7.2.3 Performance

RNF3 - Consultas eficientes.

Mecanismo para su cumplimiento: El diseño que se utilizó para el DWH será el de estrella para que las consultas sean más sencillas y garanticen un mejor rendimiento.

Se utilizaron las ejecuciones de las pruebas para poder probar y asegurar esto. También se evitan guardar información al disco local. Simplemente se baja, procesa en memoria, transforma y vuelve a subir a la nube sin descargar la información y guardarla.

Como pruebas se pudo obtener que en la primera ejecución se logró procesar un archivo, descomponerlo en 17 imágenes con su metadata asociada. Esto tomó un tiempo de 4.46 minutos, mientras que luego de mejorado el diseño se procesó el mismo archivo y su ejecución tuvo una duración de 3.12 minutos.

7.3 Infraestructura

En cuanto a las tecnologías de infraestructura se utilizó AWS como proveedor de servicios, un Bucket de S3 para guardar la información y el data warehouse de Redshift.

También se utilizó Docker [8] para facilitar el despliegue e instalación del framework.

Por último se utilizó Heroku como plataforma para el uso del servidor, este ambiente es pre productivo y en el plan de transición disponible en el capítulo 15 se explica con detalle cómo realizar el pasaje al ambiente de producción.

La lista completa entonces es:

- AWS.
- Docker.
- Heroku.

7.4 Diagramas

7.4.1 Arquitectura con tecnologías utilizadas

Para visualizar el relacionamiento de las distintas partes del sistema, se generó el siguiente diagrama de arquitectura (Imagen 7.2) con las tecnologías que fueron utilizadas:

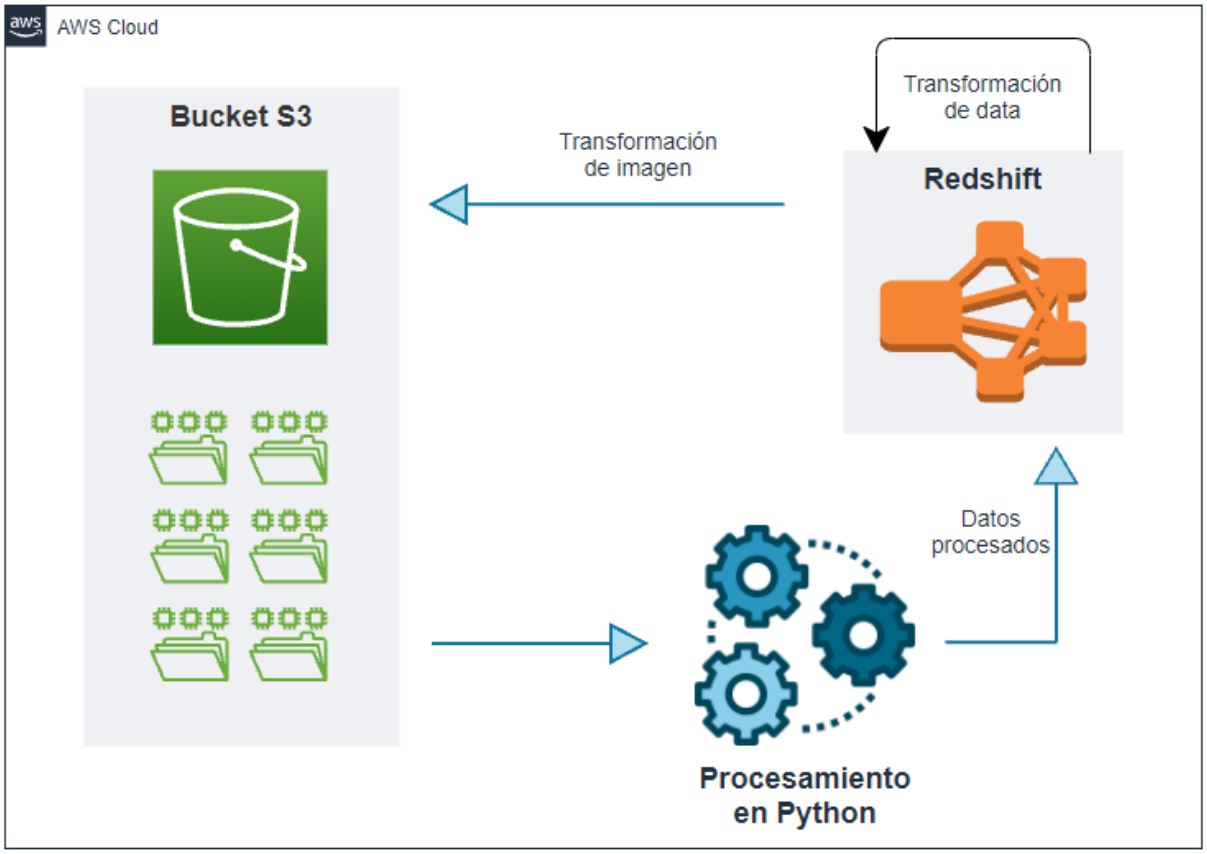


Imagen 7.2 - Arquitectura con tecnologías.

7.4.2 Diagramas de interacción

De manera de visualizar la interacción de las distintas partes del sistema y cómo viaja la información de una a la otra se generó un diagrama de interacción para la etapa de procesamiento (Imagen 7.3) y otro para la de transformación (Imagen 7.4).

7.4.2.1 Procesamiento

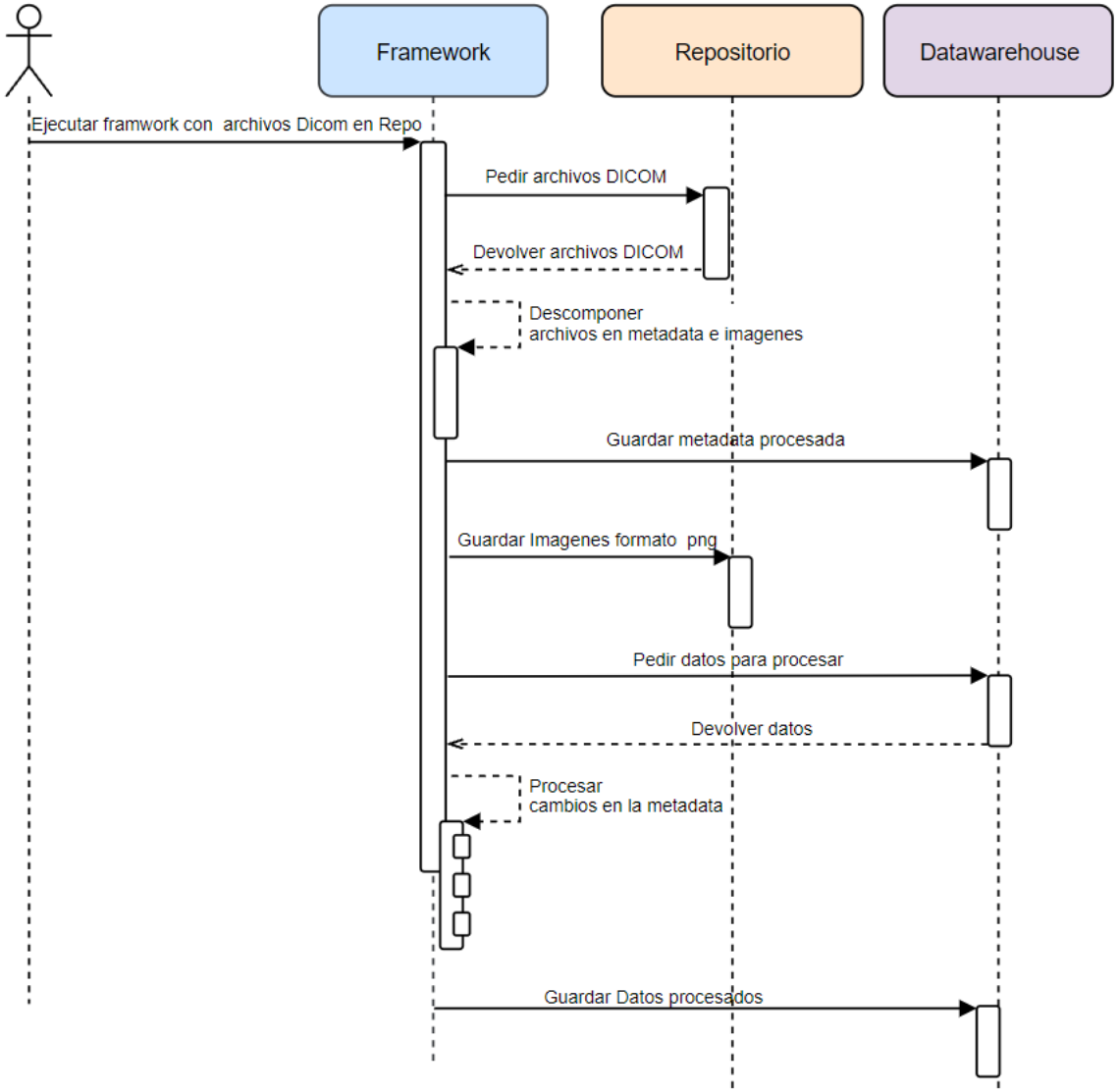


Imagen 7.3 - Diagrama de interacción de procesamiento.

7.4.2.2 Transformación

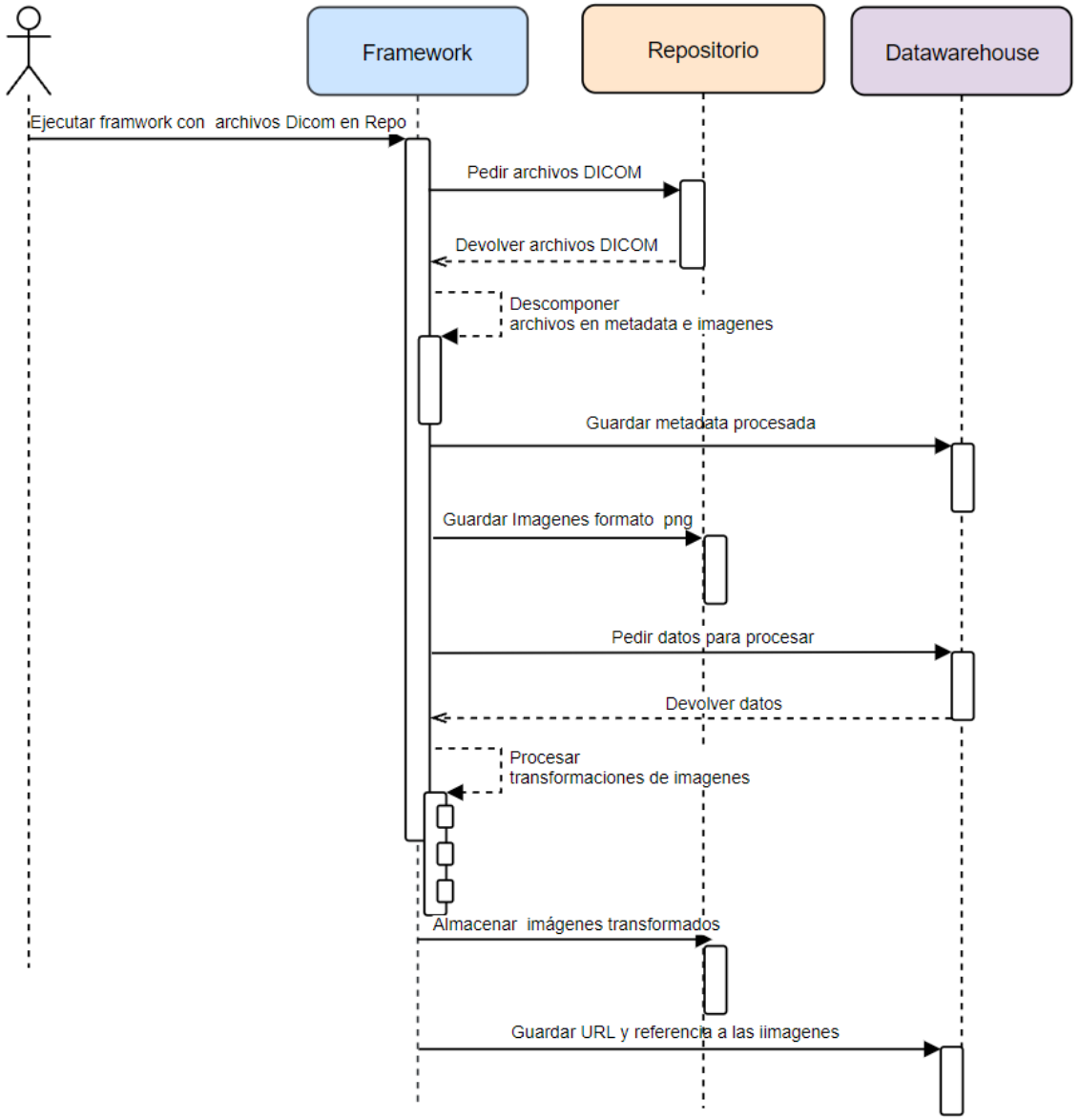


Imagen 7.4 - Diagrama de interacción de transformación.

8. Tecnologías Utilizadas

En esta sección se presentarán y describirán las tecnologías utilizadas durante el desarrollo e implementación de la solución.

Es importante destacar que la solución no cuenta con interfaz de usuario, ya que para ejecutar procesamientos o transformaciones basta con procesar una línea en la consola, lo cual resulta sencillo para el usuario final objetivo, que son los analistas de datos.

8.1 Backend

Se entendió que Python 3 era la mejor opción para desarrollar el backend de la solución. Esto se debe a que ya existen librerías como “pydicom” que ofrecen diferentes funcionalidades que facilitaban la lectura de los archivos DICOM, además de ser un lenguaje de programación con una alta capacidad de cómputo y que facilita la implementación de procesos paralelos y ejecuciones en el background.

8.2 Desafíos

El principal desafío en este proyecto en cuanto al desarrollo, fue la falta de conocimiento de las tecnologías a utilizar. De hecho este es uno de los riesgos identificados por el equipo, que se puede ver más en detalle en la sección de riesgos en el presente documento. Ningún miembro del equipo contaba con experiencia en las tecnologías que se decidieron usar, por lo que se tuvo que invertir muchas horas al desarrollo para generar experiencia y conocimiento sobre las mismas, haciendo que algunas tareas sencillas llevaran más tiempo de lo debido.

Con el objetivo de realizar el mejor producto posible y pensando en su posterior desarrollo y expansión, se decidió investigar y utilizar las tecnologías que más se

adecuaban a las necesidades, y no optar por aquellas en las que el equipo tuviera conocimiento, asumiendo el riesgo que esta decisión conllevaba.

8.3 Tecnologías

En este apartado se describirán algunos detalles relevantes de las tecnologías utilizadas para la implementación de la solución.

8.3.1 AWS

Amazon Web Service (AWS) es un proveedor de servicios en la nube, el cual permite acceder a diferentes características como son almacenamiento, recursos de computación, bases de datos, entre otras de forma remota y sin necesidad de preocuparse por el hardware e infraestructura.

Algunas de las características más llamativas de AWS son la seguridad, accesibilidad y disponibilidad, las cuales para esta solución son bastante importantes. Sobre todo en el aspecto de seguridad, teniendo en cuenta que se trabaja con archivos médicos, los cuales por naturaleza son sensibles.

8.3.2 Amazon Simple Storage Service: S3

Amazon Simple Storage Service (S3) es un servicio de almacenamiento de objetos de gran escalabilidad, disponibilidad y seguridad, además de alto rendimiento.

Estos atributos son realmente importantes para la solución ya que las imágenes obtenidas de los archivos DICOM se almacenarán en un repositorio de S3 con su correspondiente referencia en el data warehouse. Las mismas deberán estar disponibles siempre que el usuario final lo requiera, para poder consultarlas o realizar transformaciones sobre las mismas.

8.3.3 Redshift

Redshift es un almacén de datos del ecosistema Amazon, el cual permite realizar consultas pesadas sobre datos estructurados y semi estructurados mediante el uso de SQL estándar.

Las características más importantes de Redshift para la solución brindada son la escalabilidad y capacidad de procesamiento.

8.3.4 Python

Python es un lenguaje de programación interpretado que tiene como filosofía hacer hincapié en una sintaxis que favorezca un código legible.
En los últimos tiempos este lenguaje ha sido muy utilizado en tareas de procesamiento de información, así como también de machine learning e inteligencia artificial debido a su alta capacidad de cómputo, siendo esta una de las razones por las que se decidió utilizar este lenguaje para el desarrollo de la solución [9].

8.3.5 Heroku

Heroku es una plataforma que permite compartir, disponibilizar, supervisar aplicaciones y alojarlas en la nube. Esta plataforma permite hacer disponible fácilmente código en *Python* 3 y es por eso que se decidió utilizar el mismo. Además, cuenta con servicios gratuitos para aplicaciones pequeñas.

8.3.6 Docker

Docker es un sistema operativo para contenedores. De manera similar a como una máquina virtual virtualiza el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor.

Docker facilita de gran manera los procesos de instalación, ya que el propio desarrollador crea un archivo con los comandos necesarios para la correcta instalacion y ejecucion del programa, permitiendo que desarrolladores externos o usuarios finales, puedan descargar dicho archivo, ejecutarlo y no preocuparse por las dependencias o diferentes configuraciones que podrían ser necesarias para hacer funcionar el software.

9. Diseño del data warehouse

En esta sección se describe la teoría detrás del diseño del data warehouse, ahondando en detalles sobre su construcción.

9.1 Metodología de diseño

Para el diseño del data warehouse se utiliza como guía la metodología de Kimball [10], la cual orienta la construcción del data warehouse desde lo más pequeño a lo más grande.

En este proyecto se comenzó trabajando sobre un tipo específico de archivos DICOM, las angiografías, y en base a ello, se generaron las tablas y columnas del data warehouse. Durante el transcurso del proyecto se agregaron tablas y columnas al data warehouse, todas enfocadas a angiografías.

Esto permitió al equipo poder construir el data warehouse específico para este tipo de archivos en un tiempo razonable, teniendo en cuenta la duración del proyecto.

El software se elaboró de manera que agregar nuevas tablas y columnas al data warehouse sea sencilla. Es decir, utilizar la metodología de Kimball permite al equipo tener un producto tangible y extensible en el tiempo estipulado para el proyecto.

Los pasos ejecutados para lograrlos fueron:

- Enfocarse en el negocio: aplicado al proyecto, el equipo se centró solamente en un tipo de estudio.
- Construir la infraestructura con la información adecuada: los atributos agregados al data warehouse son solamente los necesarios para ese tipo de estudio.

En caso de optar por otro tipo de metodología de diseño como es la de Inmon [11], se debería construir el data warehouse pensando en todos los tipos de archivos DICOM existentes, por lo que es posible que aún se siguiera en las etapas de relevamiento.

9.2 Estructura del data warehouse

En esta sección se describirán las tablas creadas hasta el momento en el data warehouse, categorizadas por: principales, codigueras y temporales.

9.2.1 Tabla principal

La tabla principal del data warehouse es una sola, la cual se nombró "**main_table**". Esta tabla contiene la información que no fue agrupada y extraída a las tablas codigueras. Además, contiene los identificadores a cada una de las tablas codigueras para los atributos que se extrajeron.

Por ejemplo, en esta tabla encontraremos entre otros atributos un campo llamado "image_paths", con valores numéricos incrementales partiendo de 1, donde estos hacen referencia a la tabla de imágenes. Suponiendo que el campo "image_paths" tiene el valor 1 para una tupla, indica que en la tabla de imágenes, la imagen con el identificador 1 es la asociada a esa tupla.

En la siguiente imagen se puede ver a modo de ejemplo lo mencionado anteriormente. Este es el resultado de ejecutar una consulta en la cual se traen algunos atributos de la tabla principal "main_table" y los identificadores de la codiguera para 'patient_table" así como un atributo correspondiente a esta codiguera.

Imagen 9.1 - Ejemplo de tabla principal y codiguera.

9.2.2 Tablas codigueras

Por cada atributo o conjunto de atributos que se separaron y extrajeron de la tabla principal, se crea una tabla codiguera, que mediante identificadores asocia los valores y de la tabla principal con la correspondiente tupla dentro de la tabla codiguera.

En esta ocasión se generaron tres tablas codigueras, una para las imágenes, otra para la información del paciente y finalmente una para el tipo de estudio. Se deja abierta la posibilidad y los mecanismos para continuar extrayendo campos de la tabla principal y creando codigueras de manera simple, sin necesidad de realizar grandes cambios en el código. Para esto se creo un archivo Python en el cual se indica qué campo debe tomarse como identificador para una nueva tabla codiguera y además debe indicarse la estructura de la tabla codiguera, luego el sistema tomará la información de este archivo y creará, completara y subirá la nueva tabla codiguera sin necesidad de realizar cambios en la lógica del software.

9.2.2.1 image_table

En esta tabla se guarda además del identificador para poder relacionarla con la tabla principal, la url al data lake donde se aloja la imagen “físicamente” en formato PNG.

9.2.2.2 patient_table

Esta tabla aloja la información inherente al paciente con su identificador a la tabla principal. En esta tabla se encuentra el sexo, nombre, identificacion y fecha de nacimiento.

9.2.2.3 study_table

Esta tabla contiene la información relacionada al estudio realizado sobre el paciente, además de su correspondiente identificación con la tabla principal. Se pueden encontrar atributos tales como la instancia del estudio y orientación del paciente, entre otros.

9.2.3 Tablas temporales

Las tablas temporales son aquellas que se crean luego de la ejecución de una transformación, y alojan en ella el resultado de la misma. Se utiliza este mecanismo para así nunca modificar mediante transformaciones la tabla principal o las codigueras, haciendo de estas la única fuente de verdad y proporcionando siempre la misma información para futuras transformaciones o consultas.

El contenido de cada transformación va a cambiar de una a otra y va a depender del código de la misma.

10. Plan y gestión del proyecto

En esta sección se describirán las diferentes etapas del proyecto. Se detallarán las particularidades del ciclo de vida utilizado, la metodología de trabajo, además de comentar en detalle el proceso de relevamiento y desarrollo.

10.1 Ciclo de Vida y Metodología

Luego de analizar el contexto de la necesidad planteada, se definió el ciclo de vida y la metodología que se utilizó a lo largo del proyecto.

10.1.1 Ciclo de vida

En base a las necesidades planteadas por el cliente y la falta de experiencia por parte de los integrantes del equipo en las tecnologías a utilizar, se decidió elegir un ciclo de vida iterativo-incremental [12]. En este tipo de ciclo de vida se realizan iteraciones, que dan como resultado un entregable funcional (o un conjunto de ellos), el cual es llamado incremento de valor para el producto. A su vez, en cada una de las iteraciones aumenta el entendimiento del producto y de las tecnologías utilizadas por parte del equipo.

Al final del proyecto se obtiene el producto final, el cual será el acumulado de los incrementos de valor creados en cada iteración.

Este tipo de ciclo de vida es utilizado cuando los requerimientos pueden ser cambiantes, las tecnologías a utilizar son desconocidas por el equipo y/o los objetivos que están poco definidos o son de alta complejidad.

En el proyecto se decidió utilizar demostraciones del producto como una herramienta para hacer estas validaciones por parte del usuario de manera

temprana, permitiendo validar que lo desarrollado hasta el momento estaba alineado con las necesidades.

En la Imagen 10.1 se muestra la modalidad de trabajo en cada iteración y su relación con el siguiente ciclo.

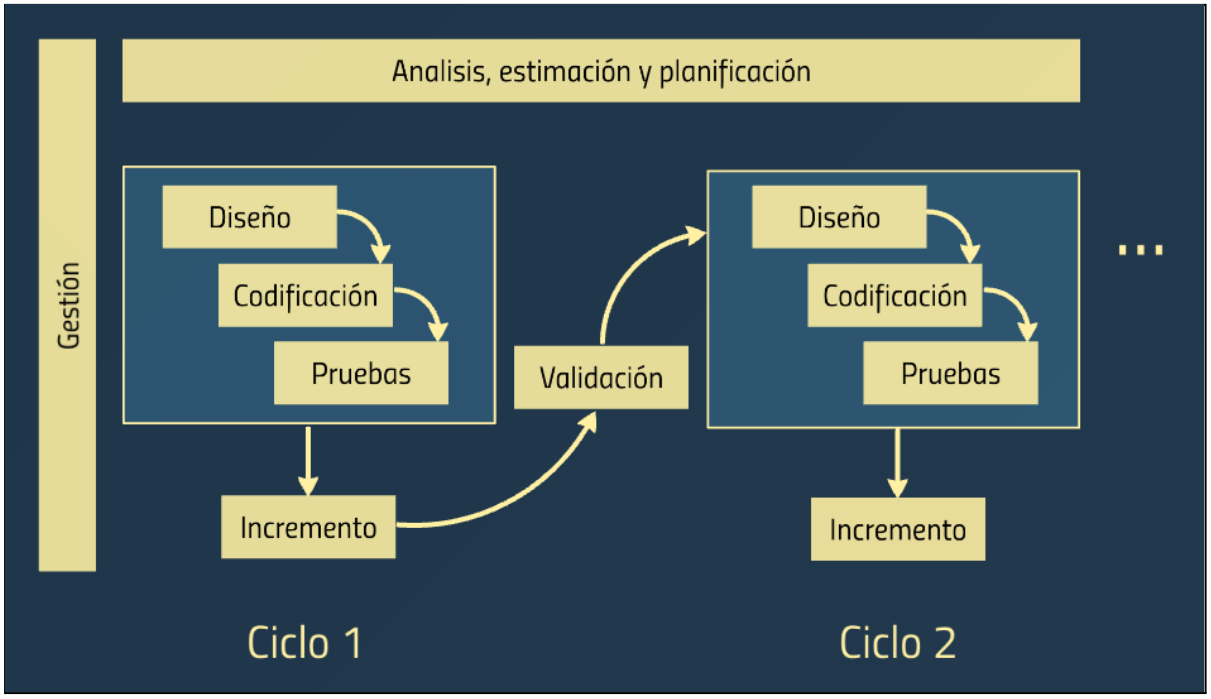


Imagen 10.1 - Ciclo de vida del proyecto.

Al inicio del proyecto se realizó el análisis del dominio y de los requerimientos solicitados por el cliente. También se realizó la estimación y la planificación de las tareas para cumplir con el alcance.

En cada iteración, se realizaron actividades de diseño y codificación, para llevar adelante soluciones a las tareas asociadas a desarrollo planificadas. También se realizaron actividades de pruebas funcionales informales para validar las soluciones implementadas. Eso generaba un incremento que se utilizaba de insumo para la siguiente iteración.

Las validaciones de los incrementos se realizaban con el cliente en las reuniones semanales. El equipo relataba lo realizado y se discutía si la solución era adecuada. El objetivo de esta validación es constatar que el incremento realizado en el ciclo está alineado a las expectativas del usuario.

10.1.2 Metodología

Debido a la complejidad del proyecto y la falta de experiencia en el dominio del problema, se decidió utilizar un híbrido de marcos de trabajo ágiles cómo Scrum [13] y *Kanban* [14], donde a medida que avanzan las iteraciones se va adquiriendo experiencia en las tecnologías utilizadas y se descubren las necesidades del cliente para cada requerimiento. También se utilizaron metodologías tradicionales, por ejemplo al inicio del proyecto se realizó la fase de ingeniería de requerimientos para obtener los mismos, esto es típico de la metodología Waterfall [15]. Otra característica que se utilizó fue la estimación de las tareas en horas en lugar de utilizar puntos de historia como es habitual en metodologías ágiles.

El equipo decidió dividir el trabajo en iteraciones de dos semanas, en donde realizaron reuniones diarias entre los integrantes de equipo para conocer el estado del trabajo de cada uno, reuniones al comienzo de cada iteración para planificar el trabajo dentro de dicho periodo, así como reuniones al finalizar cada iteración en donde se refinaban las tareas existentes en el backlog. Estas son ceremonias definidas por Scrum.

Por otro lado, la asignación de tareas de desarrollo tomó algunas prácticas de *Kanban*, como son el trabajo a demanda.

El marco de trabajo seleccionado fue adaptado en base a la carga horaria de cada uno de los integrantes del equipo y la disponibilidad del cliente con las siguientes características:

- Ciclos de dos semanas comenzando los lunes y finalizando el domingo de la siguiente semana.
- El primer lunes de cada ciclo se realizaba una reunión de planificación del ciclo que estaba comenzando.
- Una o dos reuniones semanales de seguimiento vía Google Meet.
- Una reunión semanal con la tutora del proyecto.
- Una reunión semanal con el cliente.

10.2 Etapas del proyecto

Los seis meses del proyecto se dividieron en ciclos de dos semanas cada uno y un último ciclo de cuatro semanas dedicado a ajustar los últimos detalles del producto y generar la documentación. Como resultado se obtuvieron nueve ciclos de dos semanas y uno final de cuatro semanas.

10.2.1 Estimación

Al principio de cada una de las iteraciones se hicieron revisiones del trabajo planificado y se realizaron ajustes según los siguientes criterios:

- Tareas no finalizadas en los ciclos anteriores.
- Cambios de prioridad a tareas que se planificaron para ciclos posteriores.
- Tareas que no aplican.
- Re estimaciones de horas.

Las estimaciones para cada tarea se realizaron en horas. Todas las estimaciones y re estimaciones fueron realizadas por todo el equipo, contemplando no superar el tiempo total estimado de trabajo del equipo de 160 horas por ciclo. Luego de finalizada una tarea, quien la realizó debía encargarse de ingresar la cantidad de horas reales que invirtió en su desarrollo.

Para las estimaciones de tareas se utilizó la técnica Planning Poker [16], ya que no se contaba con experiencia suficiente como para utilizar otras, tales como el Juicio de Expertos.

10.2.2 Etapa de planificación

10.2.2.1 Asignación de roles y responsabilidades

Al inicio del proyecto se definieron roles específicos para cada miembro del equipo, esto determinó las responsabilidades y los referentes para cada tarea. La selección de cada rol fue basada en la experiencia laboral de cada integrante.

A medida que avanzó el proyecto, si bien se respetaron las responsabilidades de cada uno, si se requería apoyo en cualquier rol se proveía. Esto llevó a que en algunos casos se trabajara en roles diferentes a los asignados anteriormente permitiendo así un crecimiento personal en base a lo experimentado.

En la siguiente tabla se nombran los roles definidos a lo largo del proyecto y los responsables de cada uno:

Rol	Responsable
Analista de requerimientos	Santiago Ferulano, Camila Zinola.
Analista técnico	Camila Zinola.
Desarrollador	Alejandro Horne, Camila Zinola, Juan Manuel Ferreira.
Encargado de calidad	Juan Manuel Ferreira.
Encargado de documentación	Todos.
Encargado de investigación	Alejandro Horne, Santiago Ferulano.
Gestor de configuración	Juan Manuel Ferreira.
Gestor de proyecto	Santiago Ferulano.
Nexo con el cliente	Juan Manuel Ferreira.
Tester	Camila Zinola, Santiago Ferulano.

Tabla 10.1 - Roles.

10.2.2.2 Disponibilidad del equipo

Con la amenaza del COVID durante el transcurso del proyecto, se debió planificar con cautela por el posible riesgo de infección que impidiera la concreción de la planificación original. Sin embargo, la no presencialidad en el ámbito laboral durante ese período permitió contar con flexibilidad horaria.

En principio se estimó que el esfuerzo de trabajo de cada integrante sería de 20 horas por semana. En base a esto, se dividió el proyecto en ciclos de 2 semanas cada uno, lo que nos dejó 40 horas por ciclo por integrante, siendo el total de 160 horas por ciclo para el equipo.

A la hora de estimar y distribuir el trabajo, se evitó superar este máximo de 160 horas por ciclo para dejar algo de holgura en caso de, por ejemplo, alguna infección por COVID como se mencionó anteriormente.

10.2.2.3 Planificación de tareas

La herramienta utilizada para la planificación, seguimiento, gestión y estimación de las tareas durante el proyecto fue Monday [17]. Para lograr disponer de todas las funcionalidades necesarias se debía tener una licencia con costo, por lo que el equipo tuvo que realizar gestiones para lograr conseguir una licencia para estudiantes (con funcionalidades extra a la estándar) hasta Octubre, pudiendo acceder a funcionalidades que los planes gratuitos no contaban.

Para la planificación del proyecto se definió una línea base en Monday en la que se crearon todas las tareas que se pensaban llevar a cabo a lo largo del proyecto, el ciclo en el que se harían y se asignó una estimación en horas a cada una de ellas.

Se categorizaron todas las tareas según ocho categorías que se pueden ver en la siguiente tabla:

Categoría	Descripción
Desarrollo	Actividad asociada al desarrollo del producto.
Documentación	Actividad asociada a la elaboración y revisión de la documentación generada para el proyecto.
Entrega final	Actividad asociada a la preparación y revisión del producto y la documentación para la entrega final.
Ingeniería de requerimientos	Actividad asociada al relevamiento, priorización, verificación y especificación de los requerimientos funcionales y no funcionales.
Investigación	Actividad asociada a las diferentes investigaciones que se llevan a cabo para el conocimiento del dominio del proyecto.
Reuniones	Actividad asociada a las reuniones con el cliente, la tutora y entre los integrantes del equipo.
Revisión	Actividad asociada a la preparación de las instancias de revisiones e informe de avance.
Testing	Actividad asociada a los distintos tipos de pruebas realizadas sobre el producto.

Tabla 10.2 - Categorías de tareas.

Esta categorización permite que a la hora de evaluar el proyecto se conozca el porcentaje del tiempo que se utilizó para cada tipo de tarea.

Además, se definieron estados para realizar seguimiento del avance de cada tarea.

Estos estados se describen en la siguiente tabla:

Estado	Descripción
Creado	Estado por defecto que se asigna cuando se crea la tarea.
En curso	Estado que se asigna cuando uno de los integrantes del equipo trabaja la tarea.
Terminado	Estado que se asigna cuando la tarea está implementada y validada.
Bloqueado	Estado que se asigna cuando la tarea está a la espera de la finalización de otra.
A revisar	Estado que se asigna cuando la tarea debe ser revisada por un miembro del equipo.
Cancelado	Estado que se asigna cuando se decidió que no es necesario ejecutar la tarea.

Tabla 10.3 - Estados de tareas.

Luego, se duplicó la línea base creando nuestro backlog y las tareas del backlog se distribuyeron en cada uno de los ciclos. Esto se realizó para no modificar la línea base creada al inicio y tenerla como punto de comparación a medida que avanzaban los ciclos.

El criterio utilizado para la distribución de las tareas en cada ciclo en la línea base fue en un orden lógico, en el que en principio se realiza la investigación de tecnologías y luego el desarrollo del framework con las tecnologías seleccionadas, comenzando por el procesamiento de los archivos en el framework y finalizando con las tareas de diseño del data warehouse.

Esto hizo que la parte más cargada en cuanto a tiempo y esfuerzo, que era el desarrollo del framework y la creación del data warehouse, se planifique para el final del proyecto. Al notar esto, se decidió reestructurar cada uno de los ciclos, lo cual sirvió como aprendizaje para el equipo. Lo dicho anteriormente se registró como un

riesgo, que en caso de ocurrir pondría en riesgo el cumplimiento del alcance del proyecto.

Se estableció que para ciclos más tempranos se debía tener un framework funcional que procese al menos un tipo de estudio de imagenología y muestre toda su información en un data warehouse. Las tecnologías que se decidió utilizar para realizarlo fueron AWS y Python ya que son herramientas que funcionan para nuestras necesidades. Luego de tener el framework funcional procesando al menos un tipo de estudio, el equipo se centró en investigar y comparar tecnologías, de manera de evaluar ventajas y desventajas de cada una y decidir por la mejor.

En la imagen 10.4 se puede ver una captura de pantalla de la línea base en Monday.

Imagen 10.4 - Línea base y ciclos en Monday.

En la Imagen 10.5 se pueden ver las tareas del ciclo 5 finalizado, con su categoría, estado, ciclo, tiempo real y tiempo estimado.

Ciclo 5 (28/6 - 11/7)	Sub	Categoría	Estado	Tiempo Estimado (hs)
Mejorar el procesamiento en datos del archivo DICOM 3		Desarrollo	Terminado	5 hs
Investigación Python 3		Investigación	Terminado	5 hs
Diseño el data warehouse		Desarrollo	Terminado	20 hs
Investigar alternativas a S3 como repositorio de archivos DICOM		Investigación	Terminado	10 hs
Investigar alternativas para almacenar metadata		Investigación	Terminado	5 hs
Investigar alternativas a S3 para almacenar los datos preprocesados		Investigación	Terminado	5 hs
Comparativa S3 vs Alternativas de almacenamiento de datos		Investigación	Terminado	5 hs
Investigar imágenes de archivos DICOM		Desarrollo	Terminado	15 hs
Comparativa S3 vs Alternativas para repositorio de archivos DICOM		Investigación	Terminado	5 hs
+ Agregar				
		Desa... Investig...		75 hs Total

Imagen 10.5 - Ciclo 5 terminado en Monday.

Monday permitió también ver las actividades en un diagrama de Gantt, lo que ayudó a visualizar un cronograma general del proyecto y tener en cuenta las tareas que se debían realizar en cada ciclo. En la siguiente imagen se muestra un ejemplo del diagrama de Gantt sobre la línea base al comienzo del proyecto.

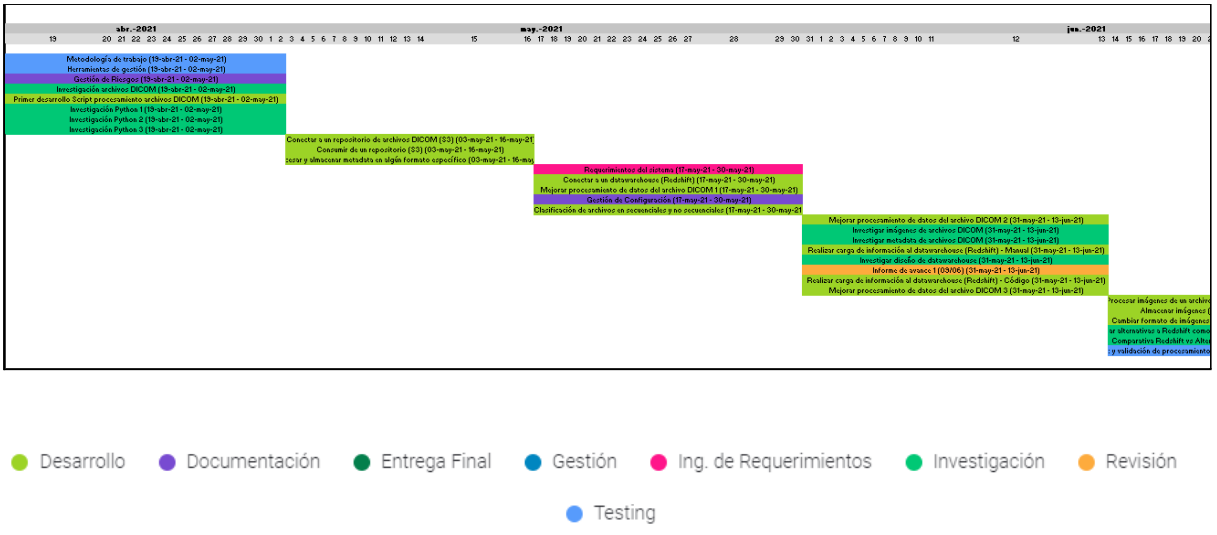


Imagen 10.6 - Diagrama de Gantt.

Este tipo de diagrama permite que el equipo tenga una visión general del estado del proyecto en cuanto al avance y también visualizar los plazos definidos para realizar cada actividad planificada.

Para el seguimiento de las tareas de desarrollo se utilizó *Github*. Cada funcionalidad a implementar se dividió en tareas más pequeñas y se generó un tablero con todas ellas. El tablero se dividió en 3 columnas, cada una representando un estado:

- “To do” para hacer referencia a las tareas que estaban pendientes.
- “In progress” referenciando a las tareas que estaban en desarrollo de manera local por alguno de los integrantes del equipo.
- “Done” para hacer referencia a las tareas que ya estaban listas e integradas a la versión del producto que estaba en el repositorio.

Este tablero permitió que el equipo tenga claro qué funcionalidades estaban listas y cuales faltaban para lograr que el producto cumpla con las necesidades de los usuarios. Cuando todas las tareas relacionadas a una misma funcionalidad eran finalizadas, se actualizaba la correspondiente tarea en Monday.

En la imagen 10.4 se puede ver el tablero en *Github*.

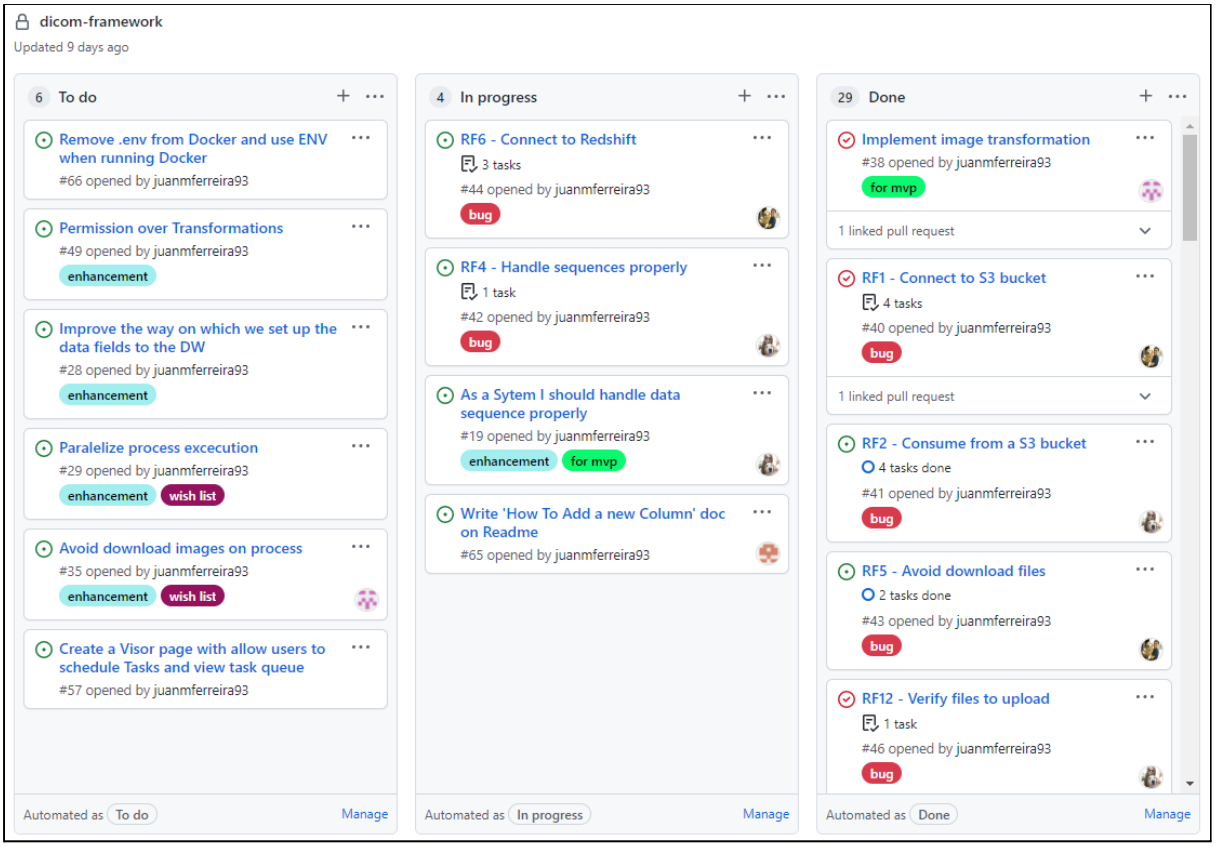


Imagen 10.7 - Tablero en Github.

11. Plan y gestión de comunicación

La gestión de la comunicación fue una tarea fundamental a la hora de llevar adelante el proyecto, ya que se tuvo que realizar un plan acorde a la virtualidad necesaria a causa de la emergencia sanitaria nacional decretada a partir de marzo del 2020.

Al comienzo del proyecto se generó el plan de comunicaciones entre los interesados del proyecto: el equipo, la tutora, el cliente y los referentes de la Universidad ORT (ex profesores de los integrantes del equipo expertos en la materia) a los que acudió el equipo cuando les fue necesario. En el mismo se definieron los diferentes canales de comunicación internos y externos, el formato, la frecuencia y la matriz de dichas comunicaciones. En la matriz se especificaron las actividades realizadas, su frecuencia, el medio por el cual se llevaron a cabo y la responsabilidades de cada interesado para con las mismas.

La comunicación interna fue fundamental porque permitió que el equipo y la tutora estuvieran alineados y coordinados a pesar de la no presencialidad. Los canales más utilizados para esta comunicación fueron Whatsapp de forma diaria y Google Meet en las reuniones semanales.

La comunicación externa entre el equipo y el cliente o referente resultó exitosa a causa del uso de los medios digitales por los cuales se realizó, de forma semanal y eventual.

11.1 Plan y gestión de interesados

11.1.1 Plan de interesados

Un interesado es toda persona, grupo y organización que puede verse afectado o identificarse a sí mismo como afectado, a nivel de decisiones, actividades o resultados del proyecto [18].

Para identificar y clasificar a los interesados más relevantes del proyecto, se generó una tabla basada en indicadores y los actores identificados.

Actor	Interés	Poder	Injerencia	Descripción
Equipo de proyecto	Alto	Alto	Positiva	Equipo que conforman los alumnos de la Universidad ORT para desarrollar el proyecto
Jimena Saavedra - Tutora	Alto	Alto	Positiva	Tutora asignada al equipo para el desarrollo del proyecto.
Rootstrap - Cliente	Alto	Alto	Positiva	Cliente que plantea sus necesidades.
Revisores de la Universidad ORT	Medio	Alto	Neutra	Revisores que evalúan el desarrollo del proyecto.
Correctores de la Universidad ORT	Medio	Alto	Neutra	Correctores que evalúan, corrigen y valoran el proyecto.
Analistas de datos	Alto	Bajo	Positiva	Usuarios finales del producto que se crea como resultado final del proyecto.

Tabla 11.1 - Tabla de interesados.

Los indicadores que se definieron son los siguientes:

- Interés:
    - Alto: Tiene un interés alto por el desarrollo del proyecto.
    - Medio: Tiene un interés medio por el desarrollo del proyecto.
    - Bajo: Tiene un interés bajo por el desarrollo del proyecto.
- Poder:
    - Alto: Tiene un poder alto sobre el proyecto.

    - Medio: Tiene un poder medio sobre el proyecto.
    - Bajo: Tiene un poder bajo sobre el proyecto.
- Injerencia:
    - Positiva: Es afectado positivamente por el desarrollo del proyecto.
    - Negativa: Es afectado negativamente por el desarrollo del proyecto.
    - Neutra: No es afectado por el desarrollo del proyecto.

Luego de haber identificado todos los interesados, se creó una matriz en la que se midió el interés en el eje X y el poder en el eje Y. Los valores se tomaron de la tabla realizada por lo que para ambos ejes pueden ser Alto, Medio y Bajo.

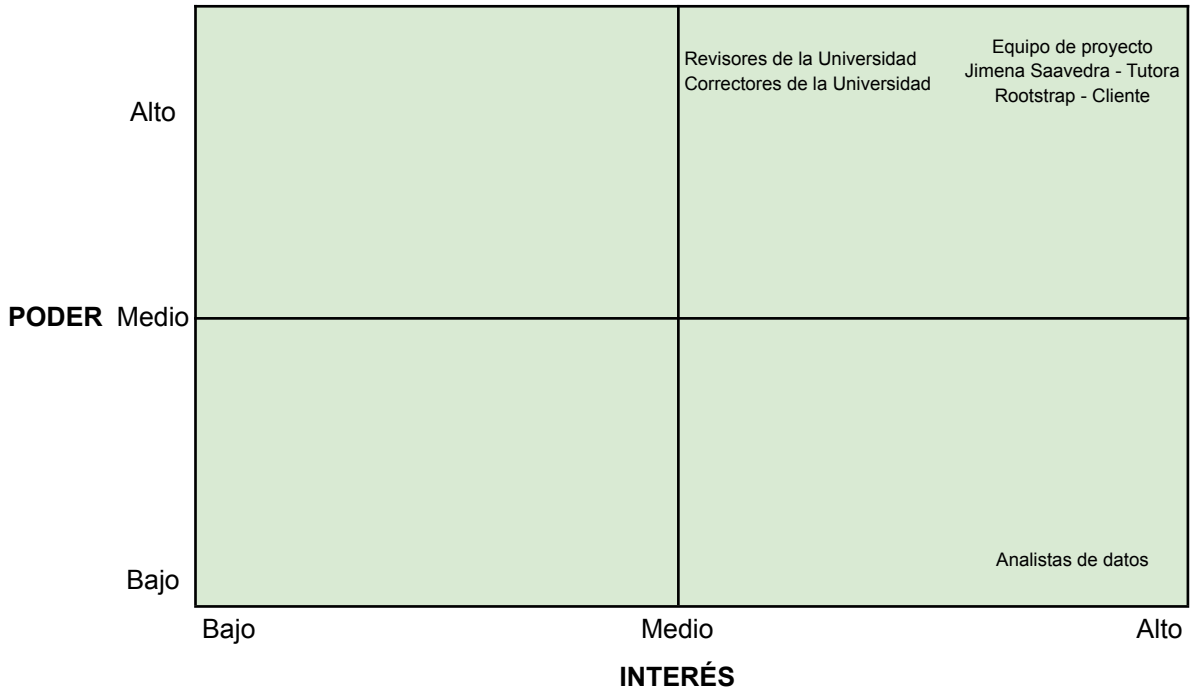


Tabla 11.2 - Matriz de interesados.

Con esta matriz se logró visualizar gráficamente cuál es el grupo de interesados a los que más importancia se les debía dar para definir nuestros objetivos y se pudo descubrir cómo debe tratarse cada uno en base al cuadrante en el que se sitúa (ver Imagen 11.3).

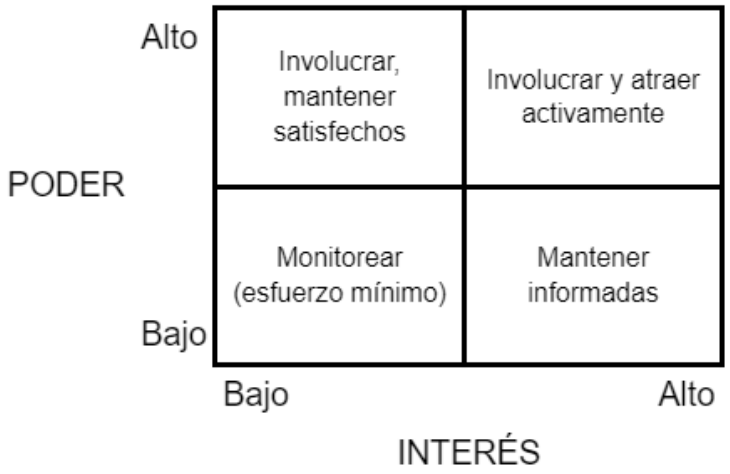


Imagen 11.3 - Matriz de trato de interesados.

11.1.2 Gestión de interesados

Se detectó que la mayoría de los interesados son positivos y solo dos de ellos son neutros. Esto hizo que se tengan que tomar acciones estratégicas para mantener a esos interesados en la posición que estaban en ese momento.

Para esto, como se aprecia en la Imagen 11.3, se precisó involucrarlos y atraerlos activamente. Para lograr esto se realizaron las siguiente acciones:

- Reuniones de equipo una o dos veces a la semana.
- Reuniones semanales con la tutora.
- Reuniones semanales con el cliente.
- Validaciones tempranas para obtener feedback del usuario.
- Instancias de revisión con la Universidad ORT.

11.2 Plan de comunicación

En esta sección se describirán las características de las comunicaciones del equipo entre los integrantes, la tutora, el cliente y los referentes. Se utilizó como punto de partida para este plan el análisis de interesados descrito en el capítulo anterior.

11.2.1 Canales de comunicación

Se utilizó comunicación interna y externa.

11.2.1.1 Comunicación Interna

Para la comunicación entre los integrantes del equipo y la tutora se llevaron a cabo:

- Reuniones virtuales vía Google Meet.
- Coordinaciones de reunión vía Google Calendar.
- Contacto vía Whatsapp y/o correo electrónico.
- Colaboraciones vía Google Drive.

11.2.1.2 Comunicación Externa

Para la comunicación entre los integrantes del equipo y el cliente, así como el equipo y referentes de la Universidad, se llevaron a cabo:

- Reuniones virtuales vía Google Meet.
- Coordinaciones de reunión vía Google Calendar.
- Contactos vía correo electrónico.
- Contactos vía Slack.
- Contactos vía Microsoft Teams.

11.2.2 Formato y contenidos del tipo de información

Las reuniones y el contacto virtual se realizaron con un formato informal. Por otra parte, los correos electrónicos en comunicación con el cliente tuvieron formato semi formal y los documentos que se realizaron a través de Google Docs siguen el formato sugerido por el documento 302 solicitado por la Universidad ORT.

11.2.3 Frecuencia de la comunicación

La frecuencia de la comunicación depende de quiénes son los receptores. En el caso de la comunicación entre los integrantes del equipo la frecuencia fue muy alta. ya que se realizó diariamente. Por otro lado, con la tutora y el cliente se realizó comunicación semanal y/o cada dos a cuatro días.

11.2.4 Matriz de comunicaciones

				Responsabilidades del interesado						
Id	**Actividad**	**Frec.**	**Medio**	**AH**	**CZ**	**JF**	**SF**	**Tutora**	**Cliente**	**Ref.**
1	Coordinación de equipo	D	D | A | CH	D | E	D | E	D | E	D | E	-	-	-
2	Reunión de equipo	S	RV | D | CH	D | E | V	D | E | V	D | E | V	D | E | V	-	-	-
3	Coordinación tutoría	E	RV | A | CH	D | E	D | E	D | E	D | E	D | A | V	-	-
4	Reunión tutoría	S	RV | D | CH	D | E	D | E	D | E	D | E	D | A | V	-	-
5	Coordinación cliente	E	A | CH	D | E	D | E	D | E	D | E	-	D | A | V	-
6	Reunión cliente	S	RV	D | E	D | E	D | E	D | E	-	D | A | V	-
7	Coordinación referentes	E	C | A | CH	D | E	D | E	D | E	D | E	-	-	D | A | V
8	Reunión referentes	E	RV	D | E	D | E	D | E	D | E	-	-	D | A | V

Tabla 11.2 - Matriz de comunicaciones.

Referencias:

- Frecuencia: D (diaria); S (semanal); M (mensual); E (eventual).
- Medio: RV (reunión virtual); C (correo electrónico); D (documento); A (agenda); CH (chat).
- Responsabilidad: D (destinatario); E (emisor); A (autoriza); V (valida).
- Integrantes del equipo: AH (Alejandro Horne); CZ (Camila Zinola); JF (Juan Ferreira); SF (Santiago Ferulano).

12. Plan y Gestión de Configuración

En esta sección se presentará el plan de gestión de la configuración, detallando los elementos de la configuración así como también las herramientas seleccionadas.

12.1 Plan de gestión de configuración

Se describirán y detallarán los diferentes elementos de configuración que se utilizaron durante el proyecto.

12.1.1 Identificación de elementos de configuración

Dentro de los elementos de configuración se pueden encontrar dos grandes grupos:

- Documentación: a este grupo lo conforman todos aquellos documentos creados durante el transcurso del proyecto, que terminarán formando parte de la documentación final del mismo.
- Software: lo conforma el código fuente de la aplicación, el cual está alojado en el repositorio de GitHub.

Tipo	Elemento
Software	Código fuente
	Librerías externas
Documentación	Definición de objetivos
	Informes de avance
	Gestión de riesgos
	Bitácora de reuniones
	Plan de métricas
	Documentación final del proyecto
	Diagramas
	Línea base y Gantt
	Especificación de requerimientos
	Guías de usuario
	Plan de comunicación
	Planillas de ejecución de pruebas
	Documento de investigación

Imagen 12.1 - Elementos de configuración.

12.2 Gestión de la configuración del software

Se presentarán las herramientas seleccionadas para la gestión de la configuración del software que se utilizaron durante la elaboración del proyecto.

12.2.1 Elección de herramientas

Para la elección de las herramientas se buscaron las que al menos uno de los integrantes tuviera experiencia, que permitan la gestión de versiones y el trabajo concurrente.

12.2.1.1 Software

El código de software relacionado al proyecto se alojó en GitHub, el cual es un proveedor de repositorios en la nube que permite gestionar aplicaciones que utilizan sistemas Git.

Git por su parte, facilita todo lo relacionado con la gestión de versiones dentro del proyecto.

Un correcto ensamblaje entre Git y GitHub permite el trabajo concurrente y el control de versiones, haciendo que ante algún inconveniente se pueda dar un paso atrás y restaurar la última versión aceptable del producto.

12.2.1.2 Organización del repositorio

Con respecto al software, se utilizó un solo repositorio en Git, alojado en GitHub. Esto se debe a que el proyecto no requiere de un frontend, por lo que no es necesario tener grandes separaciones de código.

Como estándar de trabajo se siguió el esquema expuesto por Gitflow, con algunas pequeñas modificaciones que no aplicaban al contexto. El resultado de estas modificaciones dió como resultado el siguiente esquema de ramas:

- **develop:** es la rama principal de desarrollo. Entre una liberación y otra, se corta y fusiona a esta rama a cada una de las nuevas funcionalidades. Una vez probado y aprobado el código, se genera una liberación partiendo de esta rama para ser fusionada a la rama main.
- **main:** cada vez que se realiza una release, el código completo de la misma es fusionado a esta rama, haciendo que esta fuera la que reflejaba el trabajo aprobado y liberado en todo momento.

- **feature/número/descripción:** son ramas dinámicas, inherentes a cada una de las funcionalidades a implementar. Esta rama se corta siempre desde develop, se trabaja, y se abre un pull request contra develop, el cual debe ser revisado y aprobado por el resto del equipo. Una vez aprobado, se fusiona sobre develop, agregando una nueva funcionalidad.
- **release:** se utiliza esta rama para pasar el código que está en **develop** a **main**.

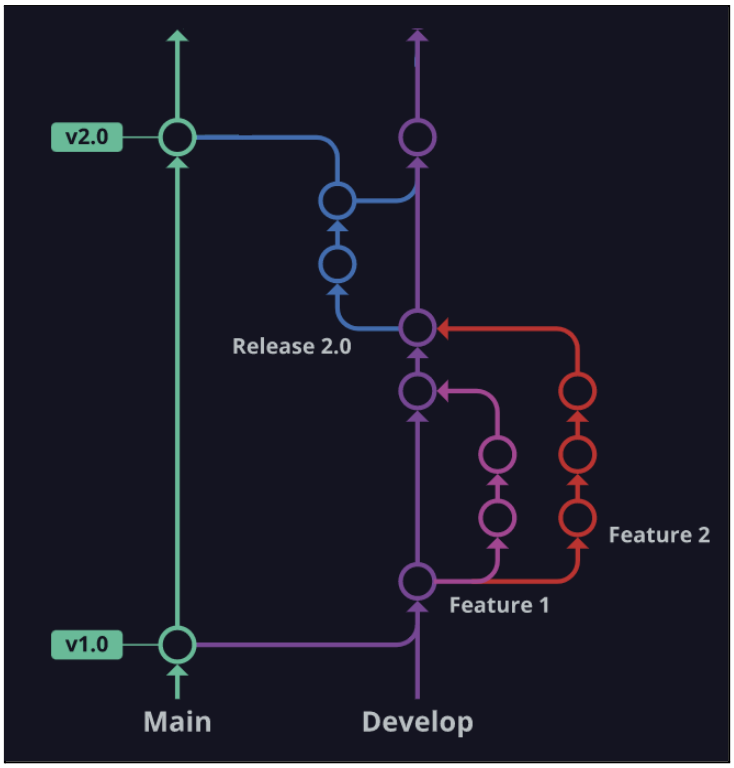


Imagen 12.2 - Esquema de ramas utilizado.

12.2.1.3 Documentación

Todos los documentos del proyecto que no fueran código de software se alojaron en Google Drive, habilitando el trabajo concurrente y dando la posibilidad de agregar comentarios/sugerencias al trabajo existente. Además, este servicio tiene una muy alta disponibilidad y es gratuito, lo que lo hace ideal para este tipo de instancias.

Dentro de Google Drive, se creó una estructura de carpetas y categorías para los diferentes documentos que fueron necesarios crear durante la elaboración del proyecto.

13. Plan y Gestión de Riesgos

Al comenzar el proyecto se realizó una matriz de riesgos para poder anticipar y también mitigar la ocurrencia de diferentes riesgos, ya sean internos o externos al proyecto. Esto llevó a realizar una clasificación de riesgos, y también elaborar planes de contingencia para aquellos que se creyeron relevantes.

13.1 Plan de riesgos

Se elaboró una matriz de riesgos colocando en el eje de las ordenadas la probabilidad de ocurrencia (P) del riesgo y en el eje de las abscisas el impacto (I) que él mismo podría causar en la planificación del proyecto.

P x I	1 - Muy Bajo	2 - Bajo	3 - Moderado	4 - Alto	5 - Muy Alto
0.1	0.1	0.2	0.3	0.4	0.5
0.2	0.2	0.4	0.6	0.8	1.0
0.3	0.3	0.6	0.9	1.2	1.5
0.4	0.4	0.8	1.2	1.6	2.0
0.5	0.5	1.0	1.5	2.0	2.5
0.6	0.6	1.2	1.8	2.4	3.0
0.7	0.7	1.4	2.1	2.8	3.5
0.8	0.8	1.6	2.4	3.2	4.0
0.9	0.9	1.8	2.7	3.6	4.5
1	1.0	2.0	3.0	4.0	5.0

Tabla 13.1 - Matriz de riesgos.

Cada uno de los valores de esta tabla, indica la magnitud (M) de un riesgo, ponderando la probabilidad (P) y el impacto (I) en caso de ocurrencia, esto da una idea de a cuáles riesgos se debió prestar más atención, ya que podrían poner en

riesgo el proyecto si no eran tenidos en cuenta y hubiesen sucedido durante la ejecución del mismo.

Luego se definieron 3 sectores de interés, en base a su magnitud (M), los cuales se colorearon utilizando la siguiente referencia:

- **Verde** (0.1 - 0.8): Baja magnitud de impacto del riesgo.
- **Amarillo** (0.9 - 2.4): Media magnitud de impacto del riesgo.
- **Rojo** (2.5 - 5.0): Alta magnitud de impacto del riesgo.

El color verde indica que el riesgo no necesariamente debería preocupar al equipo, por lo que no se invierte demasiado tiempo en estos. Aunque todos aquellos que podían ser evitables tomando algunas pequeñas acciones preventivas fueron tenidos en cuenta, evaluando costo beneficio para los que correspondiera.

En amarillo están aquellos riesgos que podían ser lo suficientemente complejos como para alterar el alcance y la planificación del proyecto, por lo que para cada uno de ellos se elaboro un plan de contingencia.

Por último, en rojo, están los riesgos críticos, de mayor interés para un correcto desarrollo del proyecto. Para todos los riesgos que se encuentran dentro de esta categoría, al igual que los de amarillo, se les elaboró un plan de contingencia, pero ademas un plan de mitigación, por si no podían ser contenidos en un principio.

Los riesgos fueron revisados y recalculados constantemente durante el proyecto, ya que algunos fueron bajando o su probabilidad de ocurrencia o su impacto en el proyecto, dado a las experiencias generadas durante la ejecución del mismo.

13.2 Gestión de riesgos

13.2.1 Identificación

Los riesgos fueron identificados de manera iterativa, en donde se comenzó listando todos los riesgos potenciales visibles, para luego ahondar en aquellos riesgos que quizás no se apreciaban en una primera instancia. Para este relevamiento se utilizó como técnica principal el juicio de expertos, teniendo en cuenta que la falta de conocimiento en la tecnología no era ponderable, pero si la complejidad de las tareas a desarrollar podrían ser traspoladas a cada una.

Se realizaron varias reuniones, entre el equipo de desarrollo como con el cliente, con el objetivo de lograr una extensa lista de riesgos, capaz de abarcar la mayor cantidad posible de casos, logrando una mejor cobertura de los mismos y beneficiando la salud del proyecto.

Este proceso se repite durante la evolución del proyecto. Esto se debe a que algunos riesgos, con el pasar del tiempo, podrían disminuir o aumentar su magnitud, ya sea por que aumenta la probabilidad de ocurrencia o el impacto del mismo por diferentes razones.

Los riesgos se documentaron, registrando los siguientes atributos:

- **Riesgo**: descripción del riesgo en sí.
- **Debido a**: causa principal del riesgo.
- **Descripción**: breve descripción del riesgo.
- **Impacto**: cuánto afectará a la salud del proyecto en caso de ocurrir.
- **Probabilidad**: probabilidad de ocurrencia del riesgo.
- **Magnitud**: impacto x probabilidad.

- **Plan de mitigación**: descripción del plan para evitar que el riesgo se materialice.
- **Plan de contingencia**: descripción del plan para contener el riesgo en caso de materializarse.
- **Estado**: estado actual del riesgo en el proyecto.

13.3 Riesgos y planes

En este apartado se describirán los riesgos detectados para este proyecto, categorizados por su naturaleza, externos e internos.

Para ver el listado completo de riesgos ver Anexo 6.

13.3.1 Riesgos internos

13.3.1.1 R1-1

	R1-1
Riesgo	Elaborar un producto que no alcance los estándares mínimos de calidad esperados para un producto de software.
Debido a	Falta de conocimiento en la tecnología a utilizar.
Impacto (1-5)	3
Probabilidad	0.5
Magnitud	1.5
Plan de mitigación	Destinar una cantidad importante de horas para investigación y entendimiento de la tecnología.
Se asume	
Plan de contingencia	Que determinados integrantes dediquen el 100% de su tiempo a aprender la tecnología para transmitir su conocimiento al resto del equipo y bajar la curva de aprendizaje.

13.3.1.2 R1-2

	R1-2
Riesgo	Elaborar un producto que no alcance los estándares mínimos de calidad esperados para un producto de software.
Debido a	Falta de experiencia con trabajo en la nube.
Impacto (1-5)	2
Probabilidad	0.6
Magnitud	1.2
Plan de mitigación	Comenzar desde el momento 0 a investigar diferentes tipos de tecnologías que permitan el procesamiento que necesitamos en la nube.
Se asume	
Plan de contingencia	Usar algunas de las tecnologías que conocemos.

13.3.1.3 R2-1

	R2-1
Riesgo	No cumplir las fechas de la planificación.
Debido a	Falta de conocimiento en la tecnología a utilizar.
Impacto (1-5)	4
Probabilidad	0.8
Magnitud	3.2
Plan de mitigación	Separar el proyecto en etapas y dedicar al menos un ciclo dentro de cada etapa para la investigación y generación de conocimientos sobre las tecnologías correspondientes a dicha etapa.
Se asume	
Plan de contingencia	Qué recursos específicos dediquen el 100% de su tiempo a aprender la tecnología para transmitir su conocimiento al resto del equipo y bajar la curva de aprendizaje.

13.3.1.4 R2-2

	R2-2
Riesgo	No cumplir las fechas de la planificación
Debido a	Falta de experiencia con trabajo en la nube.
Impacto (1-5)	3
Probabilidad	0.8
Magnitud	2.4
Plan de mitigación	Separar el proyecto en etapas. En los primeros ciclos realizar pequeñas POC sobre las herramientas de la nube. De esta manera podremos mejorar el entendimiento sobre dichas herramientas.
Se asume	
Plan de contingencia	Solicitar ayuda al DevOps asignado por Rootstrap.

13.3.1.5 R3-2

	R3-2
Riesgo	No cumplir con el alcance del proyecto.
Debido a	Definición de alcance incorrecto.
Impacto (1-5)	4
Probabilidad	0.3
Magnitud	1.2
Plan de mitigación	Entender la naturaleza del problema y las posibilidades de la tecnología. No crear falsas expectativas para el cliente ni ser demasiado ambiciosos.
Se asume	
Plan de contingencia	Acotar la solución a un número finito de tipos de archivos DICOM y generar una interfaz bien documentada que permita la extensión del framework por parte de RS una vez finalizado el proyecto.

13.3.2 Riesgos externos

13.3.2.1 R3-1

	R3-1
Riesgo	No poder realizar el proyecto.
Debido a	No disponibilidad de personas del equipo.
Impacto (1-5)	3
Probabilidad	0.5
Magnitud	1.5
Plan de mitigación	Se tendrá seguimiento diario de las tareas asignadas a cada miembro para evitar que las tareas que está desarrollando sufran retrasos durante su ausencia.
Se asume	
Plan de contingencia	Reasignación de tareas en caso de ser necesario. Se deberá aumentar el trabajo individual en esta instancia.

13.3.2.2 R7-1

	R7-1
Riesgo	Cambio de herramienta.
Debido a	Resultado negativo hacia AWS luego de investigación.
Impacto (1-5)	3
Probabilidad	0.5
Magnitud	1.5
Plan de mitigación	Realizar investigación lo antes posible para poder tener los resultados y tomar decisiones.
Se asume	
Plan de contingencia	Tomar el cambio de herramienta como una recomendación a futuro y seguir desarrollando en la plataforma de la nube en la que estamos trabajando.

13.3.3 Riesgos materializados

Durante el avance del proyecto se fueron materializando algunos de los riesgos mencionados anteriormente.

Los que más inquietaron al equipo y sobre los que se tuvo que tomar medidas fueron aquellos que tenían que ver con la falta de conocimiento en el lenguaje a

utilizar, así como el trabajo en la nube. Estos riesgos se relacionan directamente con el alcance del proyecto, por lo que un desarrollo lento del software podría generar retrasos en el proyecto, haciendo que no se cumplan con las expectativas, tanto de calidad como de funcionalidades implementadas.

Para mitigar el impacto de estos riesgos en el proyecto fue necesario dedicar a un miembro del equipo exclusivamente a adquirir conocimientos en Python y otro a todo lo relacionado con las plataformas en la nube.

13.4 Revisión de riesgos

Como se mencionó anteriormente, el equipo revisó y actualizó mensualmente la matriz de riesgos para detectar eventuales cambios en cada uno.

Se dieron algunos casos de variaciones, que se ven reflejadas por la naturaleza misma del proyecto. Se optó por ejemplificar con los siguientes riesgos, ya que son los que se materializaron durante el proyecto.

13.4.1 R2-1 y R2-2

R2-1: “No cumplir las fechas de la planificación por falta de conocimiento en la tecnología a utilizar”.

R2-2: “No cumplir con las fechas de la planificación por falta de experiencia en la nube” .

Dos riesgos bastante relacionados, enfocados en las tecnologías a utilizar por el equipo y la falta de conocimiento en la misma. Sin dudas, este fue uno de los riesgos más importantes del proyecto y de los cuales se tuvo que tomar acciones tempranamente en la ejecución del proyecto.

Como se puede apreciar en la Imagen 13.2, estos dos riesgos, tenían un nivel de magnitud bastante alto, por lo que el equipo debió dedicar mucho tiempo y recursos a planes de mitigación y contingencia.

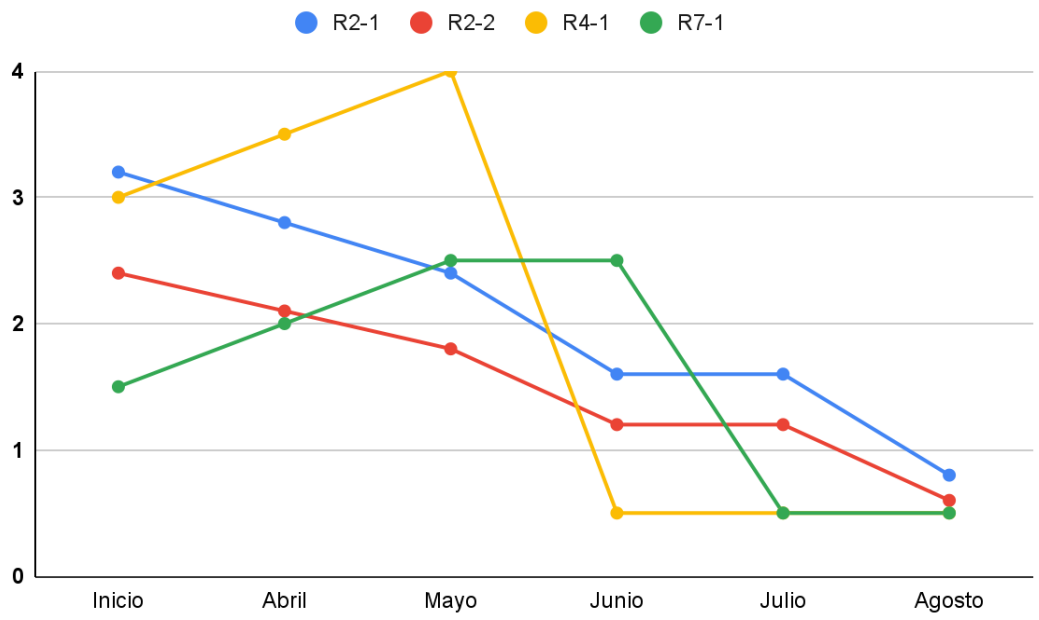


Imagen 13.2 - Evolución de riesgos.

Se puede notar que una vez avanzado el proyecto, estos riesgos comenzaron a bajar su magnitud y en este caso en especial, esto se fue dando porque el equipo fue adquiriendo conocimientos en las tecnologías, por el mismo transcurrir del proyecto, lo que llevó a que la probabilidad de ocurrencia de dichos riesgos, fuera disminuyendo.

La amenaza de estos riesgos nunca se llegó a mitigar, ya que ninguno de los miembros del equipo logró consolidarse como un “experto” tanto en Python como tecnologías en la nube de AWS, pero sí pudieron adquirir conocimientos suficientes para resolver las dificultades planteadas por el proyecto.

13.4.2 R4-1

R4-1: “No cumplir con las expectativas del producto por falta de entendimiento de los requerimientos”.

Este es un ejemplo de uno de los riesgos que se materializaron durante la ejecución del proyecto y en el cual el equipo tuvo que implementar los planes de mitigación para poder alcanzar las expectativas planteadas al comienzo del trabajo.
Tal y como se puede ver en la Imagen 13.2, la magnitud de este riesgo fue aumentando durante la evolución del proyecto. Esto se dio debido a diferentes problemas de comunicación y relevamiento de información durante las primeras etapas del proyecto.

Dada la naturaleza del proyecto, se hizo difícil para los integrantes del equipo entender con claridad cuál era la necesidad y que era lo que se buscaba lograr, en base a lo que necesitaba el cliente.

Para poder bajar la magnitud de este riesgos, fue necesario comenzar a realizar reuniones más seguido, tanto entre los integrantes como con el cliente, validando las ideas con prototipos muy simples y básicos, capaces de demostrar la idea detrás de cada requerimiento, para finalmente lograr el entendimiento colectivo tanto del problema como de la solución a desarrollar.

13.4.3 R7-1

R7-1: “Cambio de herramienta por resultado negativo hacia AWS luego de la investigación”.

Tal como se menciona en el capítulo 5, el cliente dejó en libertad al equipo de utilizar las tecnologías que les parece más correcta para la solución, por lo que se realizaron varias investigaciones al respecto.

Como la investigación es un proceso que podía demorarse, el equipo optó por comenzar a desarrollar la solución utilizando el proveedor de servicios en la nube que más se adapta a las necesidades sin estar completamente seguros de que sería el proveedor final. Para ello, se utilizaron diferentes técnicas y patrones de desarrollo, con la finalidad de estar lo menos acoplado posibles a dicho proveedor, y poder realizar un cambio si el resultado final de la investigación arroja datos relevantes que indican que otro proveedor era mejor o se adaptan mejor a las necesidades.

Este riesgo fue aumentando su magnitud en cuanto se avanzaba con la investigación (ver Imagen 13.2), ya que la misma arroja que el proveedor más económico no es AWS, sino Google, pero se pudo mitigar y no realizar el cambio, ya que la diferencia económica entre uno y otro era escasa, y además el cliente ya cuenta con infraestructura y servicios contratados por AWS, por lo que el mínimo margen de diferencia era desestimable, si se tenía en cuenta el esfuerzo requerido tanto para el equipo como para el cliente migrar a otro proveedor. En caso de tener diferencias más concluyentes, se hubiese decidido cambiar de proveedor.

14. Plan y gestión de calidad

Para asegurar el cumplimiento de las necesidades del cliente durante el desarrollo del producto y las necesidades del proceso, se realizaron tareas de gestión de la calidad a lo largo del proyecto.

En esta sección, se presentarán los objetivos de calidad y estándares que se establecieron para cumplir con los requerimientos y el correcto funcionamiento del framework. Siempre teniendo en cuenta la evolución en el tiempo de las métricas seleccionadas para la evaluación del cumplimiento de dichos objetivos.

Con la información de las métricas, el equipo pudo tomar diferentes decisiones y acciones para revertir situaciones adversas que indicaban que el cumplimiento de un objetivo podría estar en peligro de alcanzarse.

14.1 Plan de calidad

En esta sección se presentarán los distintos objetivos y métricas relacionados a la calidad del producto y del proceso.

14.1.1 Objetivos de calidad

En etapas tempranas del proyecto se establecieron los objetivos de calidad y las métricas con las que se iba a medir su cumplimiento. Dichos objetivos abarcan tanto la parte de los productos desarrollados como los procesos utilizados para llevarlos a cabo. A medida que avanzó el proyecto, algunos de los objetivos fueron modificándose debido a la experiencia adquirida con el paso del tiempo.

A continuación se presentarán los objetivos finales establecidos por el equipo.

14.1.1.1 Objetivos y métricas de calidad en el proceso

1. Para el 6 de Septiembre debe estar finalizado el proceso de desarrollo.

   Criterio de aceptación: Todas las tareas que inicialmente estaban en el backlog están en estado “Terminado” (la cantidad total de tareas en el backlog al inicio es 70).

   Validación: Registro de tareas en Monday.

   Métrica: Cantidad de tareas en estado “Terminado” = 70 y cantidad de tareas en backlog = 0.

2. Porcentaje de desviación de horas del proyecto positivo

   Criterio de aceptación: El porcentaje de desviación de las horas del proyecto es mayor o igual a cero.

   Validación: Registro de horas en Monday.

   Métrica: El porcentaje de desviación de horas del proyecto (relación entre horas estimadas y horas reales dedicadas) es igual o mayor a cero.

14.1.1.2 Objetivos y métricas de calidad en el producto

3. No deben existir defectos de severidad “Crítica” o “Alta”.

   Criterio de aceptación: No existen casos de prueba de severidad “Crítica” o “Alta” en estado distinto a “Ok”.

   Validación: Ejecución de casos de prueba.

   Métrica: Cantidad de casos de prueba de severidad “Crítica” o “Alta” con estado distinto a “Ok” = 0.

4. No deben existir requerimientos funcionales sin implementar.

   Criterio de aceptación: Todos los requerimientos funcionales están implementados.

   Validación: Ejecución de casos de prueba.

   Métrica: Cantidad de requerimientos funcionales implementados.

5. La información de los archivos DICOM originales debe ser la misma que la cargada en el data warehouse.

   Criterio de aceptación: La información original de los archivos DICOM está en el data warehouse.

   Validación: Ejecución de casos de prueba.

   Métrica: Información en Redshift.

14.1.2 Aseguramiento de la calidad

El proceso de aseguramiento de la calidad consiste en especificar las actividades a realizar para asegurar la calidad del software construido. Se detallaron los productos que debían ser revisados y los estándares aplicados, así como los métodos y procedimientos utilizados para controlar que la elaboración del producto se realice según lo establecido.

Para garantizar la calidad del producto entregado al cliente y del proceso utilizado para su creación, se establecieron desde el inicio del proyecto diferentes pautas, actividades y buenas prácticas a seguir. Se decidió utilizar el ciclo de Deming [19] como estrategia, el cual se basa en la mejora contínua de calidad. Dicho ciclo tiene cuatro fases distintas:

- Plan: fase en la que se planifica.
- Do: fase en la que se ejecuta.
- Check: fase en la que se analizan los resultados.
- Act: fase en la que se actúa en base a la etapa anterior.

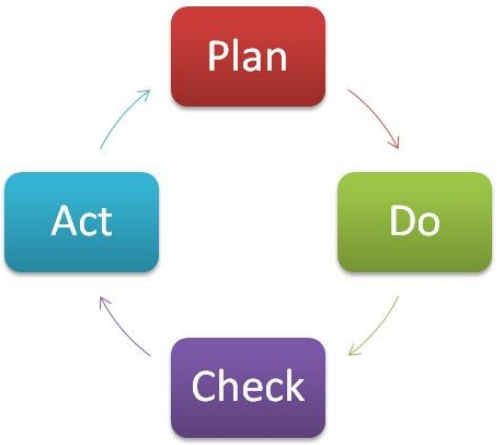


Imagen 14.1 - Ciclo de Deming.

En la fase de planificación se decidió hacer relevamiento de las actividades a realizar y cómo se debía medir cada una de ellas. Ciclo a ciclo se debían realizar las mediciones correspondientes, verificar los resultados y actuar en base a ellos.

14.1.2.1 Estándares de procesos

Se especificaron estándares de proceso para alinear objetivos, formas de trabajo y desempeño sobre las distintas tareas que se llevaron a cabo por el equipo.

14.1.2.1.1 Codificación

Con el objetivo de facilitar la mantenibilidad y entendimiento del código, se utilizó como guía lo establecido en el libro Clean Code [20]. A través de las pautas indicadas en este libro, se logra disminuir los tiempos de interpretación y entendimiento del código escrito por los demás desarrolladores, lo que ayuda en la optimización del tiempo.

Para asegurar el cumplimiento de este estándar se utilizaron las siguientes librerías Python:

- Black: es un formateador de código Python.
- Isort: sirve para ordenar las importaciones alfabéticamente y automáticamente separadas en secciones y por tipo.

Además de configurar Code Climate en el repositorio, una herramienta de CI/CD que asegura la integridad del código antes y después de fusionar nuevos cambios a la rama de desarrollo (“develop”).

14.1.2.1.2 Documentación

Se crearon directorios específicos en Google Drive para todos los documentos llevados a cabo en el proyecto. De esta manera se logró centralizar la documentación generada, siendo de fácil acceso para el equipo.

Asimismo, el presente documento cumple con lo estipulado en los documentos 302 y 303, según lo requerido por la Universidad ORT. Los mismos refieren a estándares en cuanto a estructura y formato para todo el contenido.

14.1.2.2 Métricas e indicadores del proceso y del producto

Dentro del proceso de aseguramiento de la calidad se definieron las métricas a utilizar, las cuales tenían su indicador y su objetivo. Para definir las métricas el equipo se reunió y discutió diferentes puntos de vista hasta llegar a un acuerdo, el cual posteriormente fue validado por la tutora.

14.1.2.2.1 Métricas de proceso

- Cantidad de tareas en estado “Terminado”: El indicador de esta métrica fue el registro de las tareas en Monday, ya que se puede visualizar fácilmente el estado de las tareas.
- Cantidad de tareas en el backlog: El indicador que se utilizó en esta métrica fue el backlog creado en Monday. Se puede ver rápidamente si existen tareas.
- Porcentaje de desviación de horas al finalizar el proyecto: Se utilizó como indicador la relación entre el total de horas estimadas en cada ciclo y la cantidad de horas reales que se dedicaron y el objetivo de la misma es que sea positivo.

14.1.2.2.2 Métricas de producto

- Cantidad de casos de prueba de severidad “Crítica” en estado distinto a “Ok”: Se utilizó como indicador de esta métrica el resultado de las ejecuciones de los planes de prueba y el objetivo de la misma es que sea cero.

- Cantidad de casos de prueba de severidad “Alta” en estado “No OK” o “No implementado”: El indicador de esta métrica es el resultado de las ejecuciones de los planes de prueba y el objetivo de la misma es que sea cero.

- Cantidad de requerimientos funcionales implementados: El indicador de esta métrica es la verificación de que los requerimientos funcionales estén implementados.

- Porcentaje de tests con resultado “Ok”: Como indicador de esta métrica se utilizó la relación entre el total de los casos de prueba ejecutados y la cantidad de casos de prueba con resultado “Ok” y el objetivo de la misma es que el porcentaje sea 90%.

El plan de pruebas se ejecutó en dos oportunidades y luego de cada una de ellas se realizó una evaluación de cada una de estas métricas.

14.1.2.3 Pruebas

Se creó un plan de pruebas que consistió de un conjunto de pruebas base necesarias para lograr asegurar el funcionamiento de los diferentes componentes del framework. A su vez, en el plan se contemplaron pruebas exhaustivas para hacer del framework un software más robusto.

Se vió necesario el rol de Encargado de Calidad, el cual fue asumido en conjunto por dos integrantes del equipo para asumir la responsabilidad de la creación de los casos de pruebas funcionales, reportar los diferentes errores y comportamientos anómalos encontrados, dando seguimiento y asegurando la corrección de cada uno de ellos.

En las siguientes secciones se detallarán los tipos de pruebas realizadas.

14.1.2.3.1 Pruebas funcionales

Se decidió realizar pruebas de caja negra y dentro de este tipo de pruebas se seleccionaron las pruebas funcionales.

Partiendo de los casos de uso definidos para cada funcionalidad, se establecieron casos de prueba para garantizar el cumplimiento de los objetivos por cada

funcionalidad. Se realizaron dos ejecuciones completas del plan de pruebas funcionales y se generó un documento para su registro con los siguientes campos:

- Referencia.
- Caso de prueba.
- Descripción de la prueba.
- Comportamiento esperado.
- Comportamiento obtenido.
- Fecha.
- Comentarios.
- Severidad.
- Resultado.

Para clasificar la severidad se definieron los siguientes criterios:

Severidad	Descripción
Crítica	El caso de uso es crítico. Si no se ejecuta de manera correcta el sistema no cumple su objetivo. Es parte esencial del curso normal del framework.
Alta	Caso de uso con alta probabilidad de ocurrencia que no es parte del curso normal de la ejecución del framework.
Media	Caso de uso de borde con baja probabilidad de ocurrencia que no es parte del curso normal de la ejecución del framework.
Baja	Caso de uso de borde que no es parte del curso normal de la ejecución del framework.

14.1.2.3.2 Pruebas de aceptación

Las pruebas de aceptación se llevaron a cabo en conjunto con el cliente. Lo que se buscó fue evaluar que la construcción del producto se estaba realizando acorde a las expectativas del cliente en etapas tempranas. Para esto se realizaron cuatro instancias de demo con distintas características:

- **Instancia 1**: Solamente visual y ejecutada por el equipo.
- **Instancia 2**: Solamente visual y ejecutada por el equipo.
- **Instancia 3**: El cliente probó el producto interactuando con alguien del equipo en caso de ser necesario. Tuvo a disposición las guías de usuario.
- **Instancia 4**: El cliente utilizó el producto sin interactuar con alguien del equipo. Tuvo a disposición las guías de usuario.

14.1.2.3.3 Pruebas de integración

El objetivo de estas pruebas fue probar el producto por completo tanto en el flujo de procesamiento de archivos como en transformación de la información cargada.

14.1.2.4 Revisiones

Cumpliendo con el cronograma de evaluaciones establecido por la Universidad ORT para los proyectos de fin de carrera, tuvimos la necesidad de generar dos informes de avance. Ambos informes fueron el resultado de dos instancias de revisión independientes. Cada una de estas instancias sirvieron de guía para el equipo y se tuvieron en cuenta desde el inicio para la planificación del cronograma del proyecto.

14.2 Gestión de la calidad

En esta sección se presentarán las distintas actividades relativas a la gestión de la calidad, como son los procesos de gestión de errores, la evolución de las métricas

utilizadas a lo largo del proyecto y los resultados de las pruebas realizadas durante la construcción del producto.

14.2.1 Gestión de errores

Si bien durante el desarrollo del producto se ejecutaron pruebas funcionales, se realizaron dos ejecuciones formales del plan de pruebas funcionales en dos momentos específicos: 24/07 y 01/09.

En la planilla de la ejecución se registraban todos los campos establecidos en el plan de pruebas y al finalizar la ejecución de todo el plan se enviaba un informe de resultados a los desarrolladores. Se tuvo especial cuidado en que el desarrollador que codificó originalmente la funcionalidad afectada sea el mismo que la corrija para tener la menor cantidad de tiempo de retrabajo asociada a la falla reportada. Para acceder a la planilla de ejecuciones del plan de pruebas acceder al Anexo 7.

En la siguiente imagen se puede visualizar una parte de la planilla de Google Sheets utilizada para la ejecución de los planes de prueba:

Ref	Caso de prueba	Descripción	Comportamiento esperado	Comportamiento obtenido	Fecha	Comentarios	Severidad	Resultado
RF1.1	Conectar al bucket S3 con variables correctas y credenciales correctas	Se colocan credenciales y variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -AWS_SECRET_ACCESS_KEY -BUCKET_NAME	Se conecta correctamente al bucket S3	Se conecta correctamente al bucket S3	24/07/2021	No muestra mensaje de conectado con éxito	Critica	Revisar
RF1.2	Conectar al bucket S3 borrando una de las variables	Se colocan credenciales y variables de conexión correctas en el archivo de configuración, se borra una de las variables y se ejecuta. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -BUCKET_NAME	No conecta y muestra mensaje de error en las variables del archivo de configuración	decouple.UndefinedValueError: AWS_ACCESS_KEY_ID not found. Declare it as envvar or define a default value.	24/07/2021	No muestra mensaje de error	Baja	Revisar
RF1.3	Conectar al bucket S3 con usuario inexistente	Se coloca un usuario inexistente en el archivo de configuración. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -AWS_SECRET_ACCESS_KEY -BUCKET_NAME	No conecta y muestra mensaje de error de credenciales	botocore.exceptions.ClientError: An error occurred (InvalidAccessKeyId) when calling the ListObjects operation: The AWS Access Key Id you provided does not exist in our records.	24/07/2021	Corregir mensaje de error	Baja	Revisar

Imagen 14.1 - Planilla de ejecución de pruebas.

14.2.2 Evolución de métricas

A continuación se muestra la evolución de cada una de las métricas definidas.

14.2.2.1 Métricas del proceso

En las siguientes secciones se detallan las diferentes mediciones realizadas sobre las métricas referentes a la calidad del proceso.

14.2.2.1.1 Cantidad de tareas en estado “Terminado”.

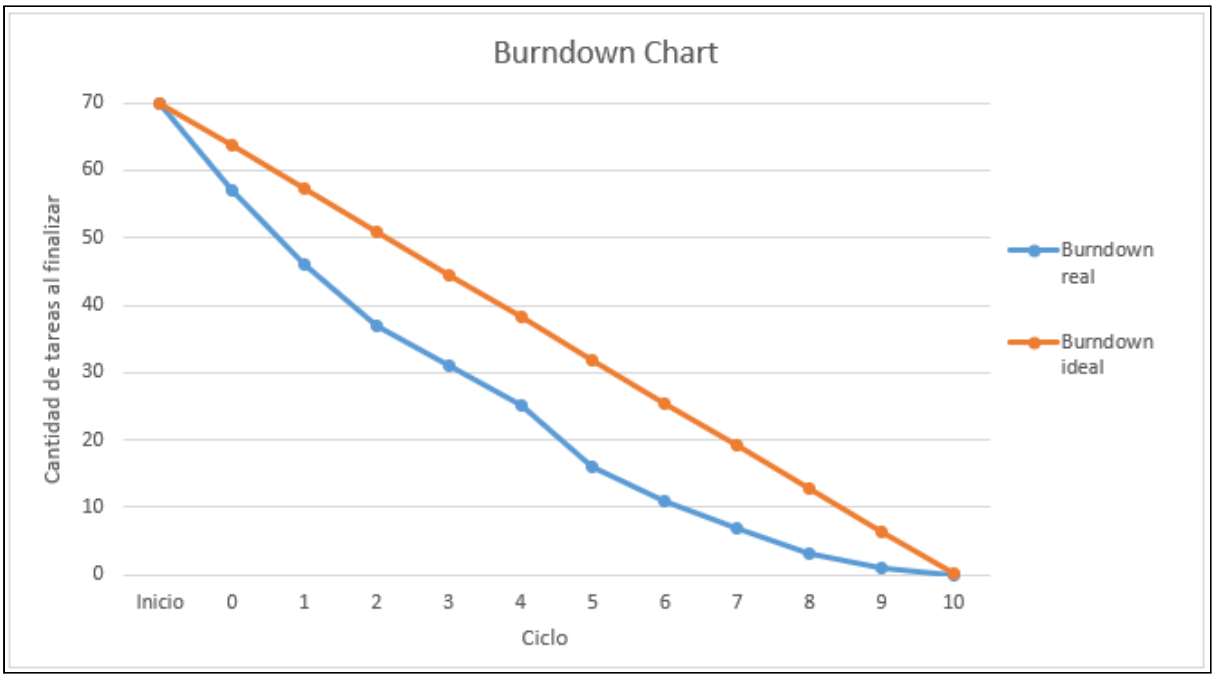
El objetivo de esta métrica es evaluar el cumplimiento del cronograma establecido. Dicho cronograma indicaba que el último ciclo de desarrollo finalizaba el 5 de septiembre, dando comienzo al ciclo 10 (de un mes de duración) el 6 de septiembre. Este último ciclo se reservó para generar la documentación formal del proyecto, por lo que era necesario culminar el desarrollo antes de llegar a dicha instancia.

El estado de cada una de las tareas se registró en Monday. En la siguiente imagen se puede ver el estado de las tareas del ciclo 9 el día 6 de septiembre:

Imagen 14.2 - Ciclo 9 al 6 de septiembre.

En la Gráfica 14.1 se representa en el eje de las X los diferentes ciclos del proyecto y en el eje de la Y la cantidad de tareas restantes en el backlog al finalizar cada uno. La línea naranja representa la velocidad ideal que debía tener el equipo para finalizar la totalidad de la tareas al término del ciclo 10. La línea azul es la velocidad

real que tuvo el equipo, donde se puede apreciar que la cantidad de tareas realizadas por ciclo fue mayor al ideal. Hasta el ciclo 6 se visualiza una mayor dedicación en cuanto a cantidad de tareas por ciclo, esto se debió a la incertidumbre respecto al proyecto y las tecnologías desafiantes. A partir del ciclo 7 se trabajó con menor velocidad a causa de la reestimación de tareas que se detalla en las siguiente secciones, pero de igual manera se logra terminar en el ciclo 10 con la cantidad total de tareas del backlog.



Gráfica 14.1 - Burndown chart.

14.2.2.1.2 Cantidad de tareas en backlog

Esta métrica fue creada con el mismo propósito que la anterior: medir el cumplimiento del cronograma establecido. El objetivo es que al finalizar el proyecto, no existan tareas pendientes en el backlog, de forma de garantizar que todo lo planificado fue llevado a cabo.

En la siguiente imagen se puede apreciar el backlog en Monday el día 6 de septiembre, el cual se encuentra sin tareas:

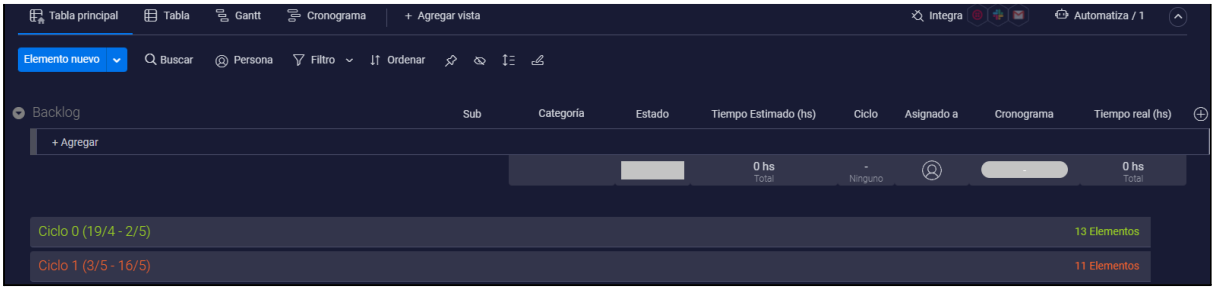
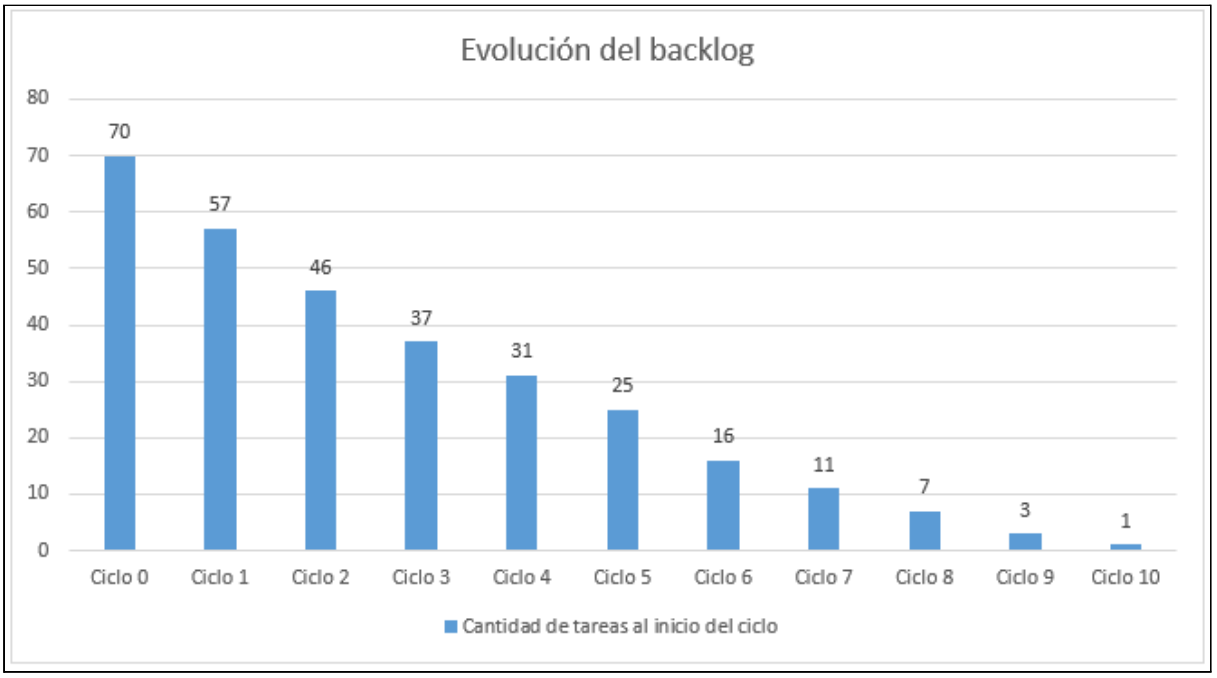


Imagen 14.3 - Backlog al 6 de septiembre.

En la Gráfica 14.2 se puede ver la cantidad de tareas en el backlog al comienzo de cada uno de los ciclos. Al inicio del ciclo 0 el backlog contenía 70 tareas pendientes y al comienzo del ciclo 10 (último ciclo con duración de un mes) restaba únicamente la tarea planificada para la documentación final del proyecto.



Gráfica 14.2 - Evolución del backlog.

14.2.2.1.3 Porcentaje de desviación de horas al fin el proyecto

El objetivo de esta métrica fue incentivar al equipo a realizar mejores estimaciones, de manera que los últimos ciclos pudieran realizarse en los tiempos pautados y cumpliendo con todas las tareas pendientes en cada uno de ellos. Inicialmente se realizaron estimaciones, las cuales no fueron acertadas. Si el equipo hubiese seguido la planificación inicial, los resultados de esta métrica serían los de la siguiente tabla:

	Horas dedicadas	Horas estimadas original	Desviación original
Ciclo 0	97	100	**-3,00%**
Ciclo 1	74	90	**-17,78%**
Ciclo 2	67	60	**11,67%**
Ciclo 3	91	100	**-9,00%**
Ciclo 4	83	144	**-42,36%**
Ciclo 5	85	130	**-34,62%**
Ciclo 6	90	75	**20,00%**
Ciclo 7	80	75	**6,67%**
Ciclo 8	105	70	**50,00%**
Ciclo 9	60	100	**-40,00%**
Ciclo 10			
Total	832	944	-58,42%

Tabla 14.1 - Desviación de horas dedicadas vs horas estimadas originalmente.

Como se puede ver, la desviación final de un total de -58,42%, lo cual estaría muy por debajo del objetivo establecido.

A medida que avanzaba el proyecto, el equipo pudo adquirir experiencia en las tecnologías utilizadas y en el dominio del problema, por lo que las re estimaciones al principio de cada ciclo pudieron ser más certeras. Si se actualiza la tabla anterior con las horas re estimadas, se obtiene la siguiente tabla:

	Horas dedicadas	Horas re estimadas	Desviación re estimada
Ciclo 0	97	95	**2,11%**
Ciclo 1	74	75	**-1,33%**
Ciclo 2	67	69	**-2,90%**
Ciclo 3	94	98	**-4,08%**
Ciclo 4	83	82	**1,22%**
Ciclo 5	85	75	**13,33%**
Ciclo 6	90	95	**-5,26%**
Ciclo 7	80	85	**-5,88%**
Ciclo 8	105	100	**5,00%**
Ciclo 9	60	60	**0,00%**
Ciclo 10			
Total	835	834	2,20%

Tabla 14.2 - Desviación horas dedicadas vs horas re estimadas

Como se puede ver en esta última tabla, el objetivo de que la desviación de horas fuera positiva fue cumplido, teniendo una desviación total positiva de 2,20%.

14.2.2.2 Métricas del producto

En las siguientes secciones se detallan las diferentes mediciones realizadas sobre las métricas referentes a la calidad del producto.

14.2.2.2.1 Cantidad de casos de severidad “Crítica” distintos a “Ok”

Esta métrica se generó con el objetivo de garantizar que las funcionalidades vitales del framework funcionen correctamente. En la primera instancia de ejecución del plan de pruebas, realizada el 24 de julio, los resultados fueron los siguientes:

Severidad	Resultado	Cantidad
Alta	Ok	5
Baja	Ok	1
Baja	Revisar	6
Crítica	No implementado	2
Crítica	Ok	10
Crítica	Revisar	2
Media	No implementado	4
Media	Ok	3
Suma total		**33**

Tabla 14.3 - Primera ejecución del plan de pruebas.

Luego de obtener los resultados, los encargados de calidad reportaron los resultados a los encargados de desarrollo para que pudieran comenzar a planificar las correcciones pertinentes.

Una vez corregidos los errores reportados, el 1 de setiembre se realizó la segunda ejecución del plan de pruebas completa, obteniendo los siguientes resultados:

Severidad	Resultado	Cantidad
Alta	Ok	5
Baja	Ok	7
Crítica	Ok	14
Media	Ok	7
Suma total		**33**

Tabla 14.4 - Segunda ejecución del plan de pruebas.

Según los resultados de esta última ejecución, se puede confirmar que el objetivo de que no existan casos de prueba de severidad “Crítica” en estado distinto a “Ok” se cumplió.

14.2.2.2.2 Cantidad de casos de severidad “Alta” distintos a “Ok”

Esta métrica se generó con el objetivo de garantizar que las funcionalidades importantes del framework funcionen correctamente.

Como se puede ver en la Tabla 14.4, el objetivo de que la cantidad de casos de prueba de severidad “Alta” en estado distinto a “Ok” sea cero, se cumplió.

14.2.2.2.3 Cantidad de requerimientos funcionales implementados

El objetivo de esta métrica fue asegurar que los requerimientos funcionales acordados con el cliente sean desarrollados en su totalidad. En la última etapa de la ingeniería de requerimientos se validó con el cliente el documento con los requerimientos para el producto que se debía entregar. Esto generó un compromiso de alcance del producto que se debía cumplir.

En dicho documento se establecieron 12 requerimientos funcionales, los cuales fueron contemplados en el plan de prueba creado.

Teniendo en cuenta los resultados vistos en la Tabla 14.4, se puede afirmar que el objetivo de que todos los requerimientos funcionales sean implementados se cumplió.

14.2.2.2.4 Porcentaje de pruebas con resultado “Ok”

Al crear esta métrica, el equipo consideró que era de suma importancia tener un control de los resultados de los casos de prueba, particularmente asegurarse de que todos (o la gran mayoría) tengan como resultado “Ok”.

Como se aprecia en la Tabla 11.3, la primera ejecución de pruebas dió un porcentaje de resultados “Ok” de un 45,4%.

Finalmente, en la segunda ejecución (ver Tabla 14.4), el porcentaje de resultados “Ok” fue del 100%. De esta manera se comprueba que el objetivo de que el porcentaje de casos de prueba con resultado “Ok” sea mayor a 90% se cumplió.

14.2.3 Pruebas

Las pruebas realizadas durante el proyecto fueron funcionales, de integración y de aceptación. En las siguientes secciones se detallan los resultados obtenidos en cada una de ellas.

14.2.3.1 Pruebas funcionales

Para el registro de los resultados de las pruebas se utilizó el modelo descrito en el capítulo 14.1.2.3.1, generando una planilla por cada una de las ejecuciones, con los diferentes escenarios de uso para cada una de las funcionalidades del sistema.

Se decidió realizar tres ejecuciones completas del plan de pruebas, de manera de tener resultados en la primera, corregirlos y verificar en una segunda instancia la evolución de los mismos. Llegada la segunda instancia, todas las pruebas dieron resultado “Ok” y no hubo necesidad de realizar una tercera ejecución.

Los resultados de cada una de las ejecuciones, según la severidad de cada caso de uso, se pueden ver en Tabla 14.3 y Tabla 14.4.

14.2.3.2 Pruebas de integración

El equipo no logró definir tempranamente un plan de pruebas de integración, ya que no se tenía claro a priori si se iba a continuar utilizando la misma tecnología. Una vez se definió esto, se pasó a la elaboración del mismo.

Las pruebas de integración se dividieron en tres:

- **Integración del framework con el repositorio de S3**. Para ello se colocaron diferentes archivos en el repositorio y se intentó comenzar la ejecución de las tareas del framework.
- **Integración del framework con Redshift**. Para asegurar la conexión con el data warehouse se ejecutaron tareas desde el framework que terminaron impactando en el data warehouse. Se comprobó meticulosamente que la información llegará al data warehouse de forma correcta y completa.
- **Integración del framework con S3 y Redshift en conjunto**. Finalmente se realizaron ejecuciones del framework desde el punto de partida en el repositorio de S3 hasta la llegada de información al data warehouse.

14.2.3.3 Pruebas de aceptación

Como se indicó en el capítulo 14.1.2.3.2, se realizaron cuatro instancias diferentes de evaluación para lograr comprender si el producto que se estaba desarrollando estaba alineado a las expectativas del cliente.

14.2.3.3.1 Instancia 1

En la primera instancia con el cliente, se realizó una demo del producto mostrando el procesamiento de un archivo DICOM y cómo la información de dicho archivo era almacenada en Redshift. Fue 100% ejecutada por los integrantes del equipo en vivo y expuesta al cliente vía Google Meet.

Al finalizar, el cliente resultó satisfecho con lo expuesto. A su vez confirmó que el producto se alineaba a sus expectativas y confirmó que era lo que necesitaban.

Por último se abordó la discusión sobre si era necesario o no implementar una interfaz de usuario, a lo que el cliente contestó que no era necesario.

14.2.3.3.2 Instancia 2

La segunda instancia con el cliente fue similar a la primera, ya que fue ejecutada por los integrantes del equipo en vivo y expuesta vía Google Meet.

En esta oportunidad se entregó como agregado de valor el procesamiento de las imágenes. Se demostró el mecanismo utilizado para la referenciación de las imágenes con el data warehouse, el cual fue del agrado del cliente, dejándolo sin cambios desde entonces.

14.2.3.3.3 Instancia 3

En esta ocasión un integrante del equipo se reunió con el cliente en sus oficinas. Esta vez el cliente fue quien utilizó el framework.

El cliente descargó el software del repositorio Github y procedió a instalarlo siguiendo las guías elaboradas por el equipo. Ante este proceso de instalación comento que “es bastante sencillo de instalar dado el agregado de Docker”.
Se prosiguió a la configuración y ejecución de tareas. La configuración resultó ser bastante sencilla para el cliente, dado que solamente es necesario agregar las credenciales correspondientes en un archivo, el cual se detalla correctamente en el README.md del repositorio.

Tanto la ejecución de tareas como la configuración, fueron del agrado del cliente, dada su poca complejidad de uso.

Surgieron algunos comentarios sobre los logs de producción en Heroku que no eran de fácil entendimiento.

14.2.3.3.4 Instancia 4

Esta instancia sirvió como cierre del proyecto. Se demostró el producto con todos los requerimientos acordados y se hizo un pasaje por cada uno de ellos.

Se implementaron las mejoras comentadas por el cliente en la instancia 3, así como pequeñas refactorizaciones del código para mejorar su entendimiento y calidad.

El cliente comentó estar satisfecho con el framework logrado, agradeciendo al equipo por su trabajo.

14.2.4 Revisiones

Para presentar los avances del trabajo realizado a lo largo del proyecto, hubo dos instancias de revisión: la primera fue el 9 de junio y la segunda fue el 16 de agosto. Ambas revisiones fueron realizadas por personas ajenas al proyecto y con perfiles diferentes para poder dar puntos de vista distintos.

Las dos instancias fueron de gran importancia para el equipo, ya que las devoluciones obtenidas en cada una de ellas enriquecieron al producto y al proceso de nuestro proyecto. También ayudaron a que los integrantes del equipo puedan explicar de mejor manera los problemas a los que se estaban enfrentando.

En el Anexo 8 se pueden ver los informes de avance realizados para cada una de las instancias de revisión.

16. Conclusiones

16.1 Lecciones aprendidas

Durante los seis meses del proyecto se destinaron esfuerzos para llevar a cabo el desarrollo de la solución y así cumplir con los objetivos planteados. La experiencia fue enriquecedora para todos los integrantes del equipo ya que se pudo llevar a la práctica todo lo aprendido durante la carrera.

También se adquirieron conocimientos y se desarrollaron habilidades a nivel técnico como resultado del trabajo con diferentes tecnologías. El equipo utilizó tecnologías que desconocía y se realizó un esfuerzo en investigación, lo que se tradujo en una curva de aprendizaje positiva. Tanto la experiencia en Python como con AWS, serán fundamentales en el desarrollo profesional de cada integrante del equipo, al ser tecnologías nuevas y en auge.

Se logró implementar una solución funcional al framework solicitado cumpliendo con la implementación de todos los requerimientos acordados con el cliente.

El equipo se adaptó a la nueva realidad, sin reuniones presenciales y desarrollando todo el proyecto únicamente de forma virtual, sin embargo su desempeño fue satisfactorio y permitió cumplir con los objetivos planteados al comienzo del mismo.

Este proyecto obligó al equipo a salir de su zona de confort, no solo desde el punto de vista de actividades o tareas desafiantes y diferentes a lo que cada integrante desempeña en el ámbito laboral, sino que también desde los conceptos trabajados a lo largo del mismo. El proyecto tuvo un fuerte componente de investigación remarcando la habilidad del equipo de superar la inercia y barreras iniciales.

Es importante destacar que las discusiones generadas a lo largo del proyecto se dieron en un ámbito cordial, enriqueciendo el resultado de las mismas con diferentes perspectivas acerca de un problema o una solución.

Con respecto a la interacción con el cliente, más allá de desvíos en algunos plazos, siempre se mantuvo un ambiente de colaboración, diálogo, atención y ayuda, manejando con holgura los compromisos mutuamente acordados al inicio.

16.2 Objetivos de proyecto

Con respecto a los objetivos del equipo en el capítulo 1.4.1 se planteó aplicar los conocimientos adquiridos durante la carrera, fueron puestos en práctica durante toda la duración del proyecto, además se planteó como meta finalizar la carrera al término del proyecto. Dentro del grupo los 4 integrantes finalizaron los cursos y exámenes pendientes y, con el proyecto aprobado, se obtendrá el título de Licenciado en Sistemas de Información para cada uno.

16.3 Objetivos de producto

Dentro de los objetivos del proyecto se planteó un producto alineado a las necesidades del cliente, cumpliendo con todas las características deseadas, en el plazo establecido, agregando valor y generando conocimiento a los integrantes del equipo. Esto también se logró, ya que se realizó una demo final con el cliente el cual quedó a gusto con el producto entregado.

También se puede afirmar que todos los integrantes del equipo generaron grandes conocimientos acerca de cómo gestionar un proyecto y sobre las tecnologías que se usaron para el mismo, las cuales son tecnologías de vanguardia.

17. Próximos pasos

Como el acuerdo con el cliente fue crear un producto mínimo viable que permita extenderse en el futuro, luego de finalizado el proyecto el equipo de Rootstrap continuará con el desarrollo del producto internamente. Como uno de los integrantes del equipo trabaja en Rootstrap, es posible que sea él quien se encargue de transmitir los conocimientos adquiridos durante el desarrollo del software, así como capacitar a sus compañeros para poder continuar con la extensión del producto.

Algunas de las mejoras que se planean introducir son:

- Extender la capacidad del sistema para procesar más tipos de archivos DICOM, es decir, diferentes tipos de estudios médicos.
- Crear más tablas codigueras para una agrupación más detallada y mejorar así también la performance de las consultas sobre el data warehouse.
- Mejorar la performance del sistema agregando ejecuciones en paralelo para los diferentes procesamientos y transformaciones.
- Mejorar la creación de nuevos campos en el data warehouse, permitiendo más flexibilidad y especificaciones en los tipos de campos.
- Agregar más tipos de salidas del sistema, con el fin de ampliar las herramientas de *Machine Learning* que puedan hacer uso del sistema.
- Crear integraciones con diferentes tecnologías de *Machine Learning*.

18. Referencias bibliográficas

[1] Rootstrap, 2021, Abril, [Online].
Disponible:https://www.rootstrap.com/about-us/

[2] DICOM Library, 2021, Abril, [Online].
Disponible:https://www.dicomlibrary.com/dicom/modality/

[3] Python, "Python", 2021, Abril, [Online].
Disponible:https://www.python.org/

[4] Gartner, “Gartner Magic Quadrant for Cloud Infrastructure and Platform Services”, 2020, Septiembre, [Online].
Disponible:https://www.gartner.com/en/documents/3989743/magic-quadrant-for-cloud-infrastructure-and-platform-ser

[5] Xacata, “Microsoft Azure y Google Cloud crecen un 50%, pero AWS sigue siendo el rey: el gasto en la nube supera los 41.800 millones de dólares”, ,2021, Mayo, [Online].
Disponible:https://www.xataka.com/pro/microsoft-azure-google-cloud-crecen-50-aws-sigue-siendo-rey-gasto-nube-supera-41-800-millones-dolares

[6] Muy Canal, “Cómo nació Amazon Web Services”, 2016, Junio, [Online].
Disponible:https://www.muycanal.com/2016/07/05/amazon-web-services-2

[7] Robert C. Martin "*The Open-Closed Principle*", C++ Report, pp. 1 (1996)

[8] Docker. “Docker”, 2021, Julio, [Online].

Disponible:https://www.docker.com/

[9] Ayudaley, “Python en proyectos Big data ¿por qué elegir lenguaje de programación?”, 2020, Junio, [Online].
Disponible:https://ayudaleyprotecciondatos.es/big-data/Python/

[10] Inteligencia de Negocios, “Metodología de Kimball”, 2014, Enero, [Online].
Disponible:http://inteligenciadenegociosval.blogspot.com/2014/01/metodologia-de-kimball.html

[11] Metodología Inmon, “Conceptos de Data Warehouse: enfoque de Kimball vs. Inmon”, 2020, Febrero, [Online].
Disponible:https://www.astera.com/es/type/blog/data-warehouse-concepts/

[12] A guide to the Project Management Body of Knowledge 6th ed, Project Management Institute, 2017, 44-51

[13] La Guía de Scrum, “La Guía Definitiva de Scrum: Las Reglas del Juego”, Noviembre, 2020, [Online].
Disponible:https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf

[14] Kanban Guides, “Kanban Guide”, 2020 [Online].
Disponible:https://kanbanguides.org/html-kanban-guide/

[15] Ian Sommerville. (2005). “*Ingeniería del Software” (séptima edición)*. pp 62. [Online].
Disponible:http://zeus.inf.ucv.cl/~bcrawford/AULA_ICI_3242/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20 Sommerville.pdf

[16] Cohn, M. “*Agile Estimating and Planning*”, Pearson, 2005

[17] Monday, “Monday”, 2021, Marzo, [Online].
Disponible:https://monday.com/lang/es/

[18] Interesados, "Stakeholder analysis", 2021, Abril, [Online].
Disponible:https://www.pmi.org/learning/library/stakeholder-analysis-pivotal-practice-projects-8905

[19] Beetrack, “Ciclo de Deming: ejemplos, etapas, importancia, ventajas y desventajas”, 2020, Noviembre, [Online].
Disponible:https://www.beetrack.com/es/blog/ciclo-de-deming-etapas-ejemplos

[20] R. C. Martin. “*Clean Code*”. Prentice Hall, 2008

Anexos

Anexo 1: Plan de transición

El código perteneciente al framework actualmente está desplegado en un ambiente pre productivo de Heroku. Una vez se comience con el periodo de transición se deberá migrar este código a un ambiente de producción proporcionado por Rootstrap, el cual será de similares características, con la gran diferencia que el ambiente pre productivo se “suspende” luego de cierto tiempo de inactividad y su costo es gratuito.

Para migrar el código a otro ambiente de Heroku, simplemente se debe loguear con el usuario y contraseña del nuevo ambiente de Heroku en la consola y correr el comando “git push heroku origin main”[1] Desde la consola, Heroku se encargará del resto del proceso. Una vez realizado el despliegue, es necesario configurar las variables de entorno dentro de Heroku.

[1] Getting Started on Heroku with python

Anexo 2: Ejecución del framework

Una vez el framework se encuentre disponible en la web, se puede acceder a él mediante consultas HTTP.

Utilizando cURL: `curl -X POST <<dirección del servidor con el framework>>/<<transformación/procesamiento deseado>>` podemos utilizar los endpoints disponibles en el framework. Por ejemplo:

- "`curl -X POST https://dicom-tesis.herokuapp.com/process_dicom`" ejecutara el procesamiento de archivos DICOM alojados en S3, creará las tablas en Redshift y subira las imágenes a S3. Siendo "https://dicom-tesis.herokuapp.com/" el servidor de pre producción en el que está alojado actualmente el framework.
- "`curl -X POST https://dicom-tesis.herokuapp.com/sql_transformation_example`" ejecutara en este caso, una transformación de SQL de ejemplo que se encuentra actualmente en el data warehouse.
- "`curl -X POST https://dicom-tesis.herokuapp.com/image_transformation_example`" ejecutara en este caso, una transformación de imagen de ejemplo que se encuentra actualmente en el data warehouse.

Anexo 3: Agregar una nueva transformación de imagen

Para agregar una nueva transformación de tipo imagen, solamente es necesario crear un nuevo archivo Python que herede de la clase “ImageTransformation” e implementar los métodos definidos como abstractos en la misma y luego crear el endpoint correspondiente para ejecutar la transformación.

La clase ImageTransformation tiene los siguientes métodos:

- run(self): que se encarga de correr la transformación.
- download_images(self, transformation_name): se encarga de descargar cada una de las imágenes para poder ejecutar luego el código que las transforme.
  Recibe como parámetro el nombre de la transformación, el cual usará para crear una carpeta temporal para alojar las imágenes durante el proceso.
  Retorna las imágenes descargadas.
- generate_table(self, transformation_name): se encarga de generar un diccionario con la información para luego convertirla en una tabla.
  Recibe como parámetro el nombre de la transformación, ya que lo utilizara para crear la tabla.
  Retorna el diccionario creado.
- write_table(self, images_dict, transformation_name): se encarga de persistir el diccionario como tabla en el data warehouse.
  Recibe como parámetros el diccionario de imágenes y el nombre de la transformación.
- transform(self, transformation_name): es de tipo abstracto, lo debe sobreescribir el implementador de la nueva transformación en la clase hija.

Contiene el código de la transformación a ejecutar a cada una de las imágenes descargadas anteriormente.

- transformation_name(self): es de tipo abstracto, lo debe sobreescribir el implementador de la nueva transformación en la clase hija. Sirve para identificar a las transformación y es usado en todos los pasos anteriores como ya se mencionó.

Pasos adicionales: si se desea disponibilizar la transformación mediante un endpoint, es necesario importar la nueva transformacion en el “main.py” y luego especificar el endpoint correspondiente a la misma de la siguiente manera:

```
from
dicomframework.transformations.image_transformation_example
import (
    ImageTransformationExample,
)

@app.route("/image_transformation_example", methods=["POST"])
def image_transformation_example():
    logger.info("ImageTransformationExample start running on
background")
    executor.submit(ImageTransformationExample().run)
    flash("ImageTransformation: ImageTransformationExample
started in background")

return redirect(url_for("index"))
```

El código de la transformación puede lucir de la siguiente manera:

```
import numpy as np
from PIL import Image, ImageEnhance
```

```
from dicomframework.transformations.image_transformation
import ImageTransformation

class ImageTransformationExample(ImageTransformation):
    def transform(self, image):
        factor = 2
        image = ImageEnhance.Contrast(image).enhance(factor)
        image = np.uint8(image)
        image = Image.fromarray(image)

        return image

    def transformation_name(self):
        return "image_transformation_example"
```

Anexo 4: Agregar una nueva transformación SQL

Para agregar una nueva transformación de tipo SQL, solamente es necesario crear un nuevo archivo Python que herede de la clase “SqlTransformation” e implementar los métodos definidos como abstractos en la misma y luego crear el endpoint correspondiente para ejecutar la transformación.

La clase SqlTransformation tiene los siguientes métodos:

- run(self): que se encarga de correr la transformación.
- sql_query(self): es de tipo abstracto, lo debe sobreescribir el implementador de la nueva transformación en la clase hija. Es el código inherente a la consulta SQL que se quiere realizar.
- view_name(self): es de tipo abstracto, lo debe sobreescribir el implementador de la nueva transformación en la clase hija. Es el nombre de la vista que se creará en el data warehouse al ejecutar la transformación.
- generate_sql_statement(self, query, view_name): se encarga de generar la consulta SQL en su totalidad, concatenando tanto la query definida por el usuario como el nombre de la vista a ser creada.

  Recibe como parámetros la query generada por el usuario y el nombre de la vista.

  Retorna la SQL final a ejecutarse.
- execute_query(self, query): crea la conexión al data warehouse y ejecuta la query.

  Recibe como parámetros la query final generada en el método “generate_sql_statement” definido anteriormente.

Pasos adicionales: si se desea disponibilizar la transformación mediante un endpoint, es necesario importar la nueva transformacion en el “main.py” y luego especificar el endpoint correspondiente a la misma de la siguiente manera:

```
from dicomframework.transformations.sql_transformation_example
import (
    SqlTransformationExample,
)

@app.route("/sql_transformation_example", methods=["POST"])
def sql_transformation_example():
    logger.info("SqlTransformationExample start running on
background")
    executor.submit(SqlTransformationExample().run)
    flash("SqlTransformation: SqlTransformationExample started
in background")

return redirect(url_for("index"))
```

El código de la transformación puede lucir de la siguiente manera:

```
from dicomframework.transformations.sql_transformation import
SqlTransformation

class SqlTransformationExample(SqlTransformation):
    def sql_query(self):
        sql_query = (
            "select * from public.main_table "
            "JOIN public.image_table "
            "ON public.main_table.image_paths =
public.image_table.id "
            "JOIN public.patient_table "
            "ON public.main_table.patientid =
public.patient_table.id"
        )
```

```
    return sql_query

def view_name(self):
    return "sql_transformation_example"
```

Anexo 5: Agregar nuevas tablas al framework

Los archivos DICOM tienen un gran número de campos en su metadata, los cuales pueden estar relacionados en distintos grupos según criterios predefinidos.

Por ejemplo, podemos tener 100 campos, de los cuales 15 son datos del paciente, por lo que estos 15 datos podrían agruparse con el criterio de “Datos del paciente”.

En el data warehouse tendremos una tabla principal con todos los campos de la metadata de los archivos (y un identificador para cada campo) y por otro lado tendremos tablas más pequeñas con la información agrupada por el criterio que se desee. Llevándolo al ejemplo anterior podríamos tener la tabla principal y la tabla de datos del paciente. En ambas tablas tendremos el ID del paciente y de esa manera se podrá acceder a la información.

Hoy en día en el framework existen algunos criterios de agrupación, pero dependiendo las necesidades o el tipo de estudio podrían surgir nuevas agrupaciones que no están definidas y debemos tomar acciones para poder concretarlas y que esas nuevas tablas se generen al procesar nuevos archivos.

En caso de encontrarnos en esta situación debemos seguir los siguientes pasos:

1. **Detectar los campos involucrados y un identificador.**

En el archivo column_mapping.py ubicado en dicomframework/dicom_generator/ tendremos definidas todas las columnas que tendrá nuestra tabla principal en la función main_cols() de la siguiente manera:

```
def main_cols():
    return [
        {"number": "00080005", "name": "SpecificCharacterSet"},
        {"number": "00080008", "name": "ImageType"},
        {"number": "00080016", "name": "SOPClassUID"},
        {"number": "00080018", "name": "SOPInstanceUID"},
        {"number": "00080020", "name": "StudyDate"},
        {"number": "00080030", "name": "StudyTime"},
        {"number": "00080050", "name": "AccessionNumber"},
        {"number": "00080060", "name": "Modality"},
        {"number": "00080070", "name": "Manufacturer"},
        {"number": "00181001", "name": "EjemploID"},
        {"number": "00181002", "name": "EjemploCampo1"},
        {"number": "00181003", "name": "EjemploCampo2"}
    ]
```

En las últimas 3 líneas podemos ver que hay 3 campos que tienen “numbers” con prefijos similares y uno de ellos tiene la palabra “ID” al final de su “name”. Supongamos que investigamos y encontramos una relación entre estos campos y entonces necesitamos que se extraigan a una nueva tabla. Aquí estamos en presencia de los campos involucrados y su identificador.

2. **Extraer la información hacia una nueva función.**

Una vez detectados los campos se generará en este mismo archivo una nueva función que contenga toda la información que se quiere extraer de la tabla principal. El código que se generaría es el siguiente:

```
def example_cols():
    return [
        {"number": "00181001", "name": "EjemploID"},
        {"number": "00181002", "name": "EjemploCampo1"},
        {"number": "00181003", "name": "EjemploCampo2"}
    ]
```

Al realizar esto, se deberá borrar de main_cols los campos de la tabla que no sean el ID:

```
def main_cols():
    return [
        {"number": "00080005", "name": "SpecificCharacterSet"},
        {"number": "00080008", "name": "ImageType"},
        {"number": "00080016", "name": "SOPClassUID"},
        {"number": "00080018", "name": "SOPInstanceUID"},
        {"number": "00080020", "name": "StudyDate"},
        {"number": "00080030", "name": "StudyTime"},
        {"number": "00080050", "name": "AccessionNumber"},
        {"number": "00080060", "name": "Modality"},
        {"number": "00080070", "name": "Manufacturer"},
        {"number": "00181001", "name": "EjemploID"},
    ]
```

3. **Agregar identificadores.**

En el mismo archivo que venimos trabajando tendremos las siguientes funciones:

```
# Here we have to add all the key field tag for each new child table
def child_id_cols():
    return ["00100020", "0020000d"]


# Here we have to specify for each key value, the table to be created on Redshift
def child_mapping_table():
    return {"00100020": "patient", "0020000d": "study"}
```

A estas se debe agregar el valor del identificador de esta nueva tabla. En el caso del ejemplo quedaría de la siguiente manera:

```
# Here we have to add all the key field tag for each new child table
def child_id_cols():
    return ["00100020", "0020000d", "00181001"]


# Here we have to specify for each key value, the table to be created on Redshift
def child_mapping_table():
    return {"00100020": "patient", "0020000d": "study", "00181001": "example"}
```

4. **Indicar al proceso que se generarán nuevas tablas.**

Debemos ir al archivo processor.py ubicado en dicomframework/dicom_generator/ y en la función "__init__" agregaremos las líneas pintadas con amarillo:

```
def __init__(self):
    # Supported modalities
    self.supported_modalities = ["XA"]

    # Get cols for each table
    self.main_cols = main_cols()
    self.child_ids = child_id_cols()

    ##### Child tables #####
    self.patient_cols = patient_cols()
    self.study_cols = study_cols()
    self.example_cols = example_cols()
    # self.xxx_cols = xxx_cols()
    ##### Finish cols section #####

    # Initialize one dict for each table
    self.main_dict = {col["name"]: [] for col in self.main_cols}

    ##### Child dicts #####
    self.patient_dict = {col["name"]: [] for col in self.patient_cols}
    self.study_dict = {col["name"]: [] for col in self.study_cols}
    self.image_dict = {col: [] for col in ["id", "image_path"]}
    self.example:dict = {col["name"]: [] for col in self.example_cols}
    # self.xxx._dict = {col['name']: [] for col in self.xxx_cols}
    ##### Finish dict initialization #####
```

5. **Agregar la tabla a la función de limpieza**

En la función "clean" del mismo archivo, debemos agregar la línea de código pintada en amarillo:

```
def clean(self):
    # This code deletes dicts
    del self.main_dict

    ##### Child dicts #####
    del self.patient_dict
    del self.study_dict
    del self.example_dict
    gc.collect()
    ##### Finish dict deletion #####
```

Anexo 6: Riesgos

Internos

R1-3

	R1-3
Riesgo	Elaborar un producto que no alcance los estándares mínimos de calidad esperados para un producto de software
Debido a	Mala performance del framework
Impacto (1-5)	4
Probabilidad	0.6
Magnitud	2.4
Plan de mitigación	Diseñar de forma meticulosa el data warehouse para almacenar los archivos DICOM de forma que sea eficiente consumirlos y almacenarlos.
Se asume	
Plan de contingencia	Asignar más recursos al servidor que aloje el framework.

R3-3

	R3-3
Riesgo	No cumplir con el alcance del proyecto
Debido a	No tener información sobre los requerimientos
Impacto (1-5)	5
Probabilidad.	0.1
Magnitud	0.5
Plan de mitigación	Planificaciones a largo plazo para evitar quedar truncados ante un período de indisponibilidad por parte de RS.
Se asume	
Plan de contingencia	Pedirle a Jimena que se comunique con Bootstrap para generar reuniones.

R4-1

	R4-1
Riesgo	No cumplir con las expectativas del producto
Debido a	Falta de entendimiento de los requerimientos.
Impacto (1-5)	5
Probabilidad	0.6
Magnitud	3
Plan de mitigación	Realizar revisiones sobre los requerimientos en cada reunión semanal con Rootstrap para lograr alinearnos en este aspecto.
Se asume	
Plan de contingencia	

R5-1

	R5-1
Riesgo	Retrabajo y/o retraso en actividades clave
Debido a	Mala comunicación entre los integrantes del equipo.
Impacto (1-5)	3
Probabilidad	0.2
Magnitud	0.6
Plan de mitigación	Realizar reuniones diarias de 15 minutos para ponernos al tanto de los avances y dificultades de cada integrante.
Se asume	
Plan de contingencia	

R5-2

	R5-2
Riesgo	Retrabajo y/o retraso en actividades clave
Debido a	Pérdida de información sobre el proyecto.
Impacto (1-5)	5
Probabilidad	0.1
Magnitud	0.5
Plan de mitigación	Realizar una copia descentralizada semanal.
Se asume	
Plan de contingencia	Utilizar la última copia almacenada.

R6-2

	R6-2
Riesgo	Retraso en las etapas de codificación del producto
Debido a	Seguridad archivos DICOM
Impacto (1-5)	3
Probabilidad	0.2
Magnitud	0.6
Plan de mitigación	Investigar con antelación cuales son los requisitos de seguridad de los archivos DICOM.
Se asume	
Plan de contingencia	Pedir a RS que nos provea de archivos que cumplan con las pautas de seguridad establecidas.

R8-1

	R8-1
Riesgo	Falta de integridad en los datos
Debido a	Complejidad de procesamiento
Impacto (1-5)	4
Probabilidad	0.3
Magnitud	1.2
Plan de mitigación	Comenzar cuanto antes a realizar correcciones en los datos que no sean fieles a los originales
Se asume	
Plan de contingencia	

Externos

R6-1

	R6-1
Riesgo	Retraso en las etapas de codificación del producto
Debido a	Indisponibilidad de archivos DICOM
Impacto (1-5)	5
Probabilidad	0.2
Magnitud	1
Plan de mitigación	Investigar diferentes fuentes de datos de archivos DICOM e intentar aprender a utilizarlas desde el comienzo del proyecto.
Se asume	
Plan de contingencia	Pedir a RS que nos provea archivos.

Anexo 7: Ejecuciones del plan de prueba

Primera ejecución del 24 de julio:

Ref	Caso de prueba	Descripción	Comportamiento esperado	Comportamiento obtenido	Fecha	Comentarios	Severidad	Resultado
RF1.1	**Conectar al bucket S3 con variables correctas y credenciales correctas**	Se colocan credenciales y variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -AWS_SECRET_ACCESS_KEY -BUCKET_NAME	Se conecta correctamente al bucket S3	Se conecta correctamente al bucket S3	24/07/2021	No muestra mensaje de conectado con éxito	Crítica	Revisar
RF1.2	**Conectar al bucket S3 borrando una de las variables**	Se colocan credenciales y variables de conexión correctas en el archivo de configuración, se borra una de las variables y se ejecuta. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -BUCKET_NAME	No conecta y muestra mensaje de error en las variables del archivo de configuración	decouple.UndefinedValueError: AWS_ACCESS_KEY_ID not found. Declare it as envvar or define a default value.	24/07/2021	No muestra mensaje de error	Baja	Revisar

RF1.3	**Conectar al bucket S3 con usuario inexistente**	Se coloca un usuario inexistente en el archivo de configuración. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -AWS_SECRET_ACCESS_KEY -BUCKET_NAME	No conecta y muestra mensaje de error de credenciales	botocore.exceptions.ClientError: An error occurred (InvalidAccessKeyId) when calling the ListObjects operation: The AWS Access Key Id you provided does not exist in our records.	24/07/2021	Corregir mensaje de error	Baja	Revisar
RF1.4	**Conectar al bucket S3 con contraseña incorrecta**	Se coloca usuario correcto y contraseña incorrecta en el archivo de configuración. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -AWS_SECRET_ACCESS_KEY -BUCKET_NAME	No conecta y muestra mensaje de contraseña incorrecta	botocore.exceptions.ClientError: An error occurred (SignatureDoesNotMatch) when calling the ListObjects operation: The request signature we calculated does not match the signature you provided. Check your key and signing method.	24/07/2021	Corregir mensaje de error	Baja	Revisar

RF2.1	**Consumir del bucket S3 con un archivo DICOM correcto**	Se ejecuta el framework con un solo archivo DICOM en el bucket S3	El framework se ejecuta correctamente y muestra mensaje de estado	El framework se ejecuta correctamente y muestra mensaje de estado	24/07/2021	-	Crítica	Ok
RF2.2	**Consumir del bucket S3 con más de un archivo DICOM correcto**	Se ejecuta el framework con más de un archivo DICOM en el bucket S3	El framework se ejecuta correctamente y muestra mensaje de estado	El framework se ejecuta correctamente y muestra mensaje de estado	24/07/2021	-	Alta	Ok
RF2.3	**Consumir del bucket S3 con un archivo que no sea DICOM**	Se ejecuta el framework con un archivo PDF en el bucket S3	Mostrar mensaje de que hay archivos incorrectos	pydicom.errors.InvalidDicomError: File is missing DICOM File Meta Information header or the 'DICM' prefix is missing from the header. Use force=True to force reading.	24/07/2021	Debería mostrar mensaje de error	Media	Ok

RF2.4	**Consumir del bucket S3 con un archivo DICOM y uno que no sea DICOM**	Se ejecuta el framework con un archivo DICOM y un PDF en el bucket S3	Procesa sólo los archivos DICOM	Muestra que está procesando y tira el mensaje de error que hay debajo. No genera los CSV. InvalidDicomError("File is missing DICOM File Meta Information " pydicom.errors.InvalidDicomError: File is missing DICOM File Meta Information header or the 'DICM' prefix is missing from the header. Use force=True to force reading. Processing dicom: test.pdf	24/07/2021	Debería procesar solamente los archivos que son DICOM. El resto ignorarlos	Media	Ok
RF2.5	**Consumir del bucket S3 con un archivo DICOM con contenido incorrecto**	Se ejecuta el framework con un archivo DICOM con contenido incorrecto en el bucket S3	Mostrar mensaje de que hay archivos incorrectos	InvalidDicomError("File is missing DICOM File Meta Information " pydicom.errors.InvalidDicomError: File is missing DICOM File Meta Information header or the 'DICM' prefix is missing from the header. Use force=True to force	24/07/2021	Corregir mensaje de error	Media	Ok

				reading.				
RF2.6	**Consumir del bucket S3 sin archivos**	Se ejecuta el framework sin archivos DICOM en el bucket S3	Mostrar mensaje de que no hay archivos en el bucket S3	Queda procesando y luego de un tiempo termina sin mostrar mensaje	24/07/2021	Debemos mostrar un mensaje descriptivo de que no hay archivos en el bucket S3	Baja	Ok
RF3.1	**Preprocesar un archivo DICOM**	Se ejecuta el framework con un solo archivo DICOM en el bucket S3 y se crean los archivos CSV con los datos del archivo DICOM	Se crean los CSV con los datos del archivo DICOM ejecutado	Se crean los CSV con los datos del archivo DICOM ejecutado y muestra mensaje de estado	24/07/2021	-	Crítica	Ok
RF3.2	**Preprocesar más de un archivo DICOM**	Se ejecuta el framework con mas de un archivo DICOM en el bucket S3 y se crean los archivos CSV con los datos de todos los archivos DICOM	Se crean los CSV con los datos de los archivos DICOM ejecutados	Se crean los CSV con los datos de los archivos DICOM ejecutados y muestra mensaje de estado	24/07/2021	-	Alta	Ok

RF4.1	**Procesar archivo con secuencias**	Se ejecuta el framework con un solo archivo DICOM con secuencias de información e imágenes	El framework se ejecuta correctamente y muestra mensaje de estado	El framework se ejecuta correctamente y muestra mensaje de estado	24/07/2021	Todavía no hacemos nada con las secuencias	Crítica	Revisar
RF4.2	**Procesar archivo sin secuencias**	Se ejecuta el framework con un solo archivo DICOM sin secuencias de información e imágenes	El framework se ejecuta correctamente y muestra mensaje de estado	El framework se ejecuta correctamente y muestra mensaje de estado	24/07/2021	-	Crítica	Ok
RF5.1	**Procesar un archivo DICOM sin descargar la información localmente**	Se ejecuta el framework con un solo archivo DICOM	No descarga localmente nada de información del archivo y lo procesa correctamente	No descarga las imagenes pero si la metadata	24/07/2021	Debemos ver la manera de no descargar la información localmente y procesarla directamente en la nube	Crítica	Ok

RF5.2	**Procesar mas de un archivo DICOM sin descargar la información localmente**	Se ejecuta el framework con mas de un archivo DICOM	No descarga localmente nada de información de los archivos y los procesa correctamente	No descarga las imagenes pero si la metadata	24/07/2021	Debemos ver la manera de no descargar la información localmente y procesarla directamente en la nube	Alta	Ok
RF6.1	**Conectar a Redshift con variables correctas y credenciales correctas**	Se colocan credenciales y variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -DSN_DATABASE -DSN_HOSTNAME -DSN_PORT -DSN_UID -DSN_PWD	Carga y procesa los archivos que están en el bucket S3 y almacena la información en las tablas correspondientes de Redshift	Carga y procesa los archivos que están en el bucket S3 y almacena la información en las tablas correspondientes de Redshift	24/07/2021	-	Crítica	Ok
RF6.2	**Conectar a Redshift borrando una de las variables**	Se colocan credenciales y variables de conexión correctas en el archivo de configuración, se borra una variable y se ejecuta. Se deben completar las variables: -DSN_HOSTNAME -DSN_PORT	Da mensaje de error y no procesa	UndefinedValueError('{} not found. Declare it as envvar or define a default value.'.format(option)) decouple.UndefinedValueError: DSN_DATABASE not found. Declare it as envvar or define a	24/07/2021	Deberíamos chequear la conexión ANTES de comenzar a procesar. Si tenemos 100 archivos da error luego de procesar el	Baja	Revisar

		-DSN_UID -DSN_PWD		default value.		último y se pierde esa ejecución.		
RF6.3	**Conectar a Redshift con usuario inexistente**	Se coloca un usuario inexistente en las credenciales y se colocan variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -DSN_DATABASE -DSN_HOSTNAME -DSN_PORT -DSN_UID -DSN_PWD	Da mensaje de error y no procesa	sqlalchemy.exc.OperationalError: (psycopg2.OperationalError) FATAL: password authentication failed for user "qwertyggg1225344" FATAL: password authentication failed for user "qwertyggg1225344"	24/07/2021	Idem anterior	Baja	Revisar
RF6.4	**Conectar a Redshift con contraseña incorrecta**	Se coloca contraseña incorrecta en las credenciales y se colocan variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -DSN_DATABASE -DSN_HOSTNAME -DSN_PORT -DSN_UID -DSN_PWD	Da mensaje de error y no procesa	sqlalchemy.exc.OperationalError: (psycopg2.OperationalError) FATAL: password authentication failed for user "awsuser" FATAL: password authentication failed for user "awsuser"	24/07/2021	Idem anterior	Baja	Revisar

RF7.1	**Procesar un archivo DICOM y verificar que los datos de ese archivo llegan a Redshift**	Se ejecuta el framework con un archivo DICOM y se verifican los datos de ese archivo contra los datos cargados en Redshift	Los datos de Redshift son los mismos que los datos del archivo DICOM original	Varios campos no tienen valor en Redshift. Además varios campos están encriptados	24/07/2021	Debemos desencriptar los datos encriptados y ver que sucede con los datos que no se pasan a Redshift	Crítica	Ok
RF7.2	**Procesar más de un archivo DICOM y verificar que los datos de esos archivos llegan a Redshift**	Se ejecuta el framework más de un archivo DICOM y se verifican los datos de esos archivos originales contra los datos cargados en Redshift	Los datos de Redshift son los mismos que los datos del archivo DICOM original	Varios campos no tienen valor en Redshift. Además varios campos están encriptados. Verificar campos bitsallocated y bitsstored que quedan vacíos y en el archivo original tienen valores encriptados	24/07/2021	Idem anterior. Verificar campos bitsallocated y bitsstored	Crítica	Ok
RF8.1	**Probar creación de diseño de DWH automático con un solo archivo**	Eliminar tablas en Redshift, ejecutar el framework con un archivo DICOM y verificar que las tablas en Redshift se crean y cargan automáticamente	Se ejecuta correctamente y se crea en Redshift las tablas que corresponde con los datos del archivo procesado	Se ejecuta correctamente y se crea en Redshift las tablas que corresponde con los datos del archivo procesado	24/07/2021	-	Crítica	Ok

RF8.2	**Probar creación de diseño de DWH automático con más de un archivo**	Eliminar tablas en Redshift, ejecutar el framework con más de un archivo DICOM y verificar que las tablas en Redshift se crean y cargan automáticamente	Se ejecuta correctamente y se crea en Redshift las tablas que corresponde con los datos de los archivos procesados	Se ejecuta correctamente y se crea en Redshift las tablas que corresponde con los datos de los archivos procesados	24/07/2021	-	Alta	Ok
RF9.1	**Procesar un archivo DICOM y verificar que en Redshift estén disponibles los datos de ese archivo**	Se ejecuta un archivo DICOM y se verifica mediante consulta SQL que la información esté disponible	La información del archivo está en Redshift disponible para consultarse	La información del archivo está en Redshift disponible para consultarse	24/07/2021	-	Crítica	Ok
RF9.2	**Procesar más de un archivo DICOM y verificar que en Redshift estén disponibles los datos de ese archivo**	Se ejecutan dos archivos DICOM y se verifica mediante consulta SQL que la información de ambos archivos está disponible	La información de los archivos está en Redshift disponible para consultarse	La información del archivo está en Redshift disponible para consultarse	24/07/2021	-	Alta	Ok

RF10.1	**Agregar una nueva transformación implementada correctamente**	-	-	-	24/07/2021		Crítica	No implementado
RF10.2	**Agregar una transformación con errores de formato en relación a la interfaz que debe implementar**	-	-	-	24/07/2021		Media	No implementado
RF10.3	**Agregar una transformación con errores de compilación**	-	-	-	24/07/2021		Media	No implementado

RF11.1	**Ejecutar una transformación correcta**	-	-	-	24/07/2021		Crítica	No implementado
RF11.2	**Ejecutar una transformación con errores de formato en relación a la interfaz que debe implementar**	-	-	-	24/07/2021		Media	No implementado
RF11.3	**Ejecutar una transformación que tiene errores de compilación**	-	-	-	24/07/2021		Media	No implementado

RF12.1	**Ejecución de framework con archivos que no sean DICOM o DICOM corruptos**	Ver pruebas RF2.3, RF2.4, RF2.5 y RF2.6	Ver pruebas RF2.3, RF2.4, RF2.5 y RF2.6	Ver pruebas RF2.3, RF2.4, RF2.5 y RF2.6	24/07/2021	-	Crítica	Ok

Segunda ejecución del 1ro de setiembre:

Ref	Caso de prueba	Descripción	Comportamiento esperado	Comportamiento obtenido	Fecha	Comentarios	Severidad	Resultado
RF1.1	**Conectar al bucket S3 con variables correctas y credenciales correctas**	Se colocan credenciales y variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -AWS_SECRET_ACCESS_KEY -BUCKET_NAME	Se conecta correctamente al bucket S3	Conecta a S3 correctamente	1/09/2021	-	Crítica	Ok
RF1.2	**Conectar al bucket S3 borrando una de las variables**	Se colocan credenciales y variables de conexión correctas en el archivo de configuración, se borra una de las variables y se ejecuta. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -BUCKET_NAME	No conecta y muestra mensaje de error en las variables del archivo de configuración	Muestra mensaje de error: "Please check your credential before continue"	1/09/2021	-	Baja	Ok

RF1.3	**Conectar al bucket S3 con usuario inexistente**	Se coloca un usuario inexistente en el archivo de configuración. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -AWS_SECRET_ACCESS_KEY -BUCKET_NAME	No conecta y muestra mensaje de error de credenciales	No conecta y muestra mensaje de error de credenciales	1/09/2021	-	Baja	Ok
RF1.4	**Conectar al bucket S3 con contraseña incorrecta**	Se coloca usuario correcto y contraseña incorrecta en el archivo de configuración. Se deben completar las variables: -AWS_DEFAULT_REGION -AWS_ACCESS_KEY_ID -AWS_SECRET_ACCESS_KEY -BUCKET_NAME	No conecta y muestra mensaje de contraseña incorrecta	No conecta y muestra mensaje de contraseña incorrecta	1/09/2021	-	Baja	Ok
RF2.1	**Consumir del bucket S3 con un archivo DICOM correcto**	Se ejecuta el framework con un solo archivo DICOM en el bucket S3	El framework se ejecuta correctamente y muestra log de la ejecución	El framework se ejecuta correctamente y muestra log de la ejecución	1/09/2021	-	Crítica	Ok

RF2.2	**Consumir del bucket S3 con más de un archivo DICOM correcto**	Se ejecuta el framework con más de un archivo DICOM en el bucket S3	El framework se ejecuta correctamente y muestra log de la ejecución	El framework se ejecuta correctamente y muestra log de la ejecución	1/09/2021	-	Alta	Ok
RF2.3	**Consumir del bucket S3 con un archivo que no sea DICOM**	Se ejecuta el framework con un archivo PDF en el bucket S3	Intenta procesar el archivo y muestra mensaje de error	Da error al intentar procesar el archivo. El mensaje es: "The following file: -nombreArchivo- is not a valid DICOM".	1/09/2021	-	Media	Ok
RF2.4	**Consumir del bucket S3 con un archivo DICOM y uno que no sea DICOM**	Se ejecuta el framework con un archivo DICOM y un PDF en el bucket S3	Procesa sólo los archivos DICOM y da error en el PDF	Procesa el archivo DICOM y da error al intentar procesar el PDF. El mensaje es: "The following file: -nombreArchivo- is not a valid DICOM".	1/09/2021	-	Media	Ok

RF2.5	**Consumir del bucket S3 con un archivo DICOM con contenido incorrecto**	Se ejecuta el framework con un PDF con su extensión modificada a ".dcm"	Mostrar mensaje de que hay archivos incorrectos	Muestra mensaje de error: "Invalid file: -nombreArchivo-."	1/09/2021	-	Media	Ok
RF2.6	**Consumir del bucket S3 sin archivos**	Se ejecuta el framework sin archivos DICOM en el bucket S3	Intenta procesar y no procesa nada	Intenta procesar y no procesa nada	1/09/2021	-	Baja	Ok
RF3.1	**Preprocesar un archivo DICOM**	Se ejecuta el framework con un solo archivo DICOM en el bucket S3 se envían los datos a Redshift para crear/agregar a las tablas	Procesa el archivo y carga las tablas en Redshift	Procesa el archivo y carga las tablas en Redshift	1/09/2021	-	Crítica	Ok

RF3.2	**Preprocesar más de un archivo DICOM**	Se ejecuta el framework con más de un archivo DICOM en el bucket S3 y se envían los datos a Redshift para crear/agregar a las tablas	Procesa los archivos y carga las tablas en Redshift	Procesa los archivos y carga las tablas en Redshift	1/09/2021	-	Alta	Ok
RF4.1	**Procesar archivo con secuencias**	Se ejecuta el framework con un solo archivo DICOM con secuencias de información e imágenes	El framework se ejecuta correctamente y muestra mensaje de estado	El framework se ejecuta correctamente y muestra mensaje de estado	1/09/2021	-	Crítica	Ok
RF4.2	**Procesar archivo sin secuencias**	Se ejecuta el framework con un solo archivo DICOM sin secuencias de información e imágenes	El framework se ejecuta correctamente y muestra mensaje de estado	El framework se ejecuta correctamente y muestra mensaje de estado	1/09/2021	-	Crítica	Ok

RF5.1	**Procesar un archivo DICOM sin descargar la información localmente**	Se ejecuta el framework con un solo archivo DICOM	No descarga localmente nada de información del archivo y lo procesa correctamente	No descarga localmente nada de información del archivo y lo procesa correctamente	1/09/2021	-	Crítica	Ok
RF5.2	**Procesar más de un archivo DICOM sin descargar la información localmente**	Se ejecuta el framework con más de un archivo DICOM	No descarga localmente nada de información de los archivos y los procesa correctamente	No descarga localmente nada de información de los archivos y los procesa correctamente	1/09/2021	-	Alta	Ok
RF6.1	**Conectar a Redshift con variables correctas y credenciales correctas**	Se colocan credenciales y variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -DSN_DATABASE -DSN_HOSTNAME -DSN_PORT -DSN_UID -DSN_PWD	Carga y procesa los archivos que están en el bucket S3 y almacena la información en las tablas correspondientes de Redshift	Carga y procesa los archivos que están en el bucket S3 y almacena la información en las tablas correspondientes de Redshift	1/09/2021	-	Crítica	Ok

RF6.2	**Conectar a Redshift borrando una de las variables**	Se colocan credenciales y variables de conexión correctas en el archivo de configuración, se borra una variable y se ejecuta. Se deben completar las variables: -DSN_HOSTNAME -DSN_PORT -DSN_UID -DSN_PWD	Da mensaje de error y no procesa	Muestra mensaje de error: "Please check your credential before continue"	1/09/2021	-	Baja	Ok
RF6.3	**Conectar a Redshift con usuario inexistente**	Se coloca un usuario inexistente en las credenciales y se colocan variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -DSN_DATABASE -DSN_HOSTNAME -DSN_PORT -DSN_UID -DSN_PWD	Da mensaje de error y no procesa	Muestra mensaje de error: "Please check your credential before continue"	1/09/2021	-	Baja	Ok

RF6.4	**Conectar a Redshift con contraseña incorrecta**	Se coloca contraseña incorrecta en las credenciales y se colocan variables de conexión correctas en el archivo de configuración y se ejecuta. Se deben completar las variables: -DSN_DATABASE -DSN_HOSTNAME -DSN_PORT -DSN_UID -DSN_PWD	Da mensaje de error y no procesa	Muestra mensaje de error: "Please check your credential before continue"	1/09/2021	-	Baja	Ok
RF7.1	**Procesar un archivo DICOM y verificar que los datos de ese archivo llegan a Redshift**	Se ejecuta el framework con un archivo DICOM y se verifican los datos de ese archivo contra los datos cargados en Redshift	Los datos de Redshift son los mismos que los datos del archivo DICOM original	Los datos de Redshift son los mismos que los datos del archivo DICOM original	1/09/2021	-	Crítica	Ok

RF7.2	**Procesar más de un archivo DICOM y verificar que los datos de esos archivos llegan a Redshift**	Se ejecuta el framework más de un archivo DICOM y se verifican los datos de esos archivos originales contra los datos cargados en Redshift	Los datos de Redshift son los mismos que los datos del archivo DICOM original	Los datos de Redshift son los mismos que los datos del archivo DICOM original	1/09/2021	-	Crítica	Ok
RF8.1	**Probar creación de diseño de DWH automático con un solo archivo**	Eliminar tablas en Redshift, ejecutar el framework con un archivo DICOM y verificar que las tablas en Redshift se crean y cargan automáticamente	Se ejecuta correctamente y se crea en Redshift las tablas que corresponde con los datos del archivo procesado	Se ejecuta correctamente y se crea en Redshift las tablas que corresponde con los datos del archivo procesado	1/09/2021	-	Crítica	Ok
RF8.2	**Probar creación de diseño de DWH automático con más de un archivo**	Eliminar tablas en Redshift, ejecutar el framework con más de un archivo DICOM y verificar que las tablas en Redshift se crean y cargan automáticamente	Se ejecuta correctamente y se crea en Redshift las tablas que corresponde con los datos de los archivos procesados	Se ejecuta correctamente y se crea en Redshift las tablas que corresponde con los datos de los archivos procesados	1/09/2021	-	Alta	Ok

RF9.1	**Procesar un archivo DICOM y verificar que en Redshift estén disponibles los datos de ese archivo**	Se ejecuta un archivo DICOM y se verifica mediante consulta SQL que la información esté disponible	La información del archivo está en Redshift disponible para consultarse	La información del archivo está en Redshift disponible para consultarse	1/09/2021	-	Crítica	Ok
RF9.2	**Procesar más de un archivo DICOM y verificar que en Redshift estén disponibles los datos de ese archivo**	Se ejecutan dos archivos DICOM y se verifica mediante consulta SQL que la información de ambos archivos está disponible	La información de los archivos está en Redshift disponible para consultarse	La información de los archivos está en Redshift disponible para consultarse	1/09/2021	-	Alta	Ok
RF10.1	**Agregar una nueva transformación sobre data**	Generar una nueva transformación sobre data siguiendo el documento para dicho fin	Se carga una nueva transformación en el sistema	Se carga una nueva transformación en el sistema	1/09/2021	-	Crítica	Ok

RF10.2	**Agregar una nueva transformación sobre imágenes**	Generar una nueva transformación sobre data siguiendo el documento para dicho fin	Se carga una nueva transformación en el sistema	Se carga una nueva transformación en el sistema	1/09/2021	-	Crítica	Ok
RF10.3	**Agregar una transformación con errores de compilación**	Generar una transformación con errores de compilación y ejecutarla	Se ejecuta y da error	Se ejecuta y da error	1/09/2021	-	Media	Ok
RF11.1	**Ejecutar una transformación sobre data**	Ejecutar la transformación creada sobre data y verificar en Redshift el resultado	Se ejecuta correctamente la transformación y se verifica en Redshift el resultado	Se ejecuta correctamente la transformación y se verifica en Redshift el resultado	1/09/2021	-	Crítica	Ok

RF11.2	**Ejecutar una transformación sobre imágenes**	Codificar una transformación sobre imágenes, incorporar al framework y ejecutarla	Se transforman las imágenes que indica la transformación y se disponibilizar en S3	Se transforman las imágenes que indica la transformación y se disponibilizar en S3	1/09/2021	-	Crítica	Ok
RF11.3	**Ejecutar una transformación que tiene errores de compilación**	Codificar una transformación sobre data, incorporar al framework y ejecutarla	Se transforma la data que indica la transformación y se disponibilizar en S3	Se transforma la data que indica la transformación y se disponibilizar en S3	1/09/2021	-	Media	Ok
RF12.1	**Ejecución de framework con archivos que no sean DICOM o DICOM corruptos**	Ver pruebas RF2.3, RF2.4, RF2.5 y RF2.6	Ver pruebas RF2.3, RF2.4, RF2.5 y RF2.6	Ver pruebas RF2.3, RF2.4, RF2.5 y RF2.6	1/09/2021	-	Crítica	Ok

Anexo 8: Informes de revisión

Como resultado de cada una de las revisiones se realizó un informe detallando fortalezas del equipo, oportunidades de mejora y acciones para implementar las oportunidades de mejora.

Informe de primera revisión

Proyecto: Framework para procesamiento de archivos DICOM

Fecha: 09/06/2021

Carrera: Licenciatura en Sistemas

Revisión nro: 1

Revisor: Gastón Mousqués

Fortalezas del equipo:

1. Integrantes con distintos perfiles (desarrollo, gestión y análisis) que nos permite separar las tareas según la especialidad de cada uno.
2. Capacidad de sortear obstáculos.
3. Comunicación constante y fluida.
4. Correcta elección de metodología

Oportunidades de mejora:

1. Enfocarnos en un ejemplo más concreto en un escenario real para explicar el funcionamiento del framework de punta a punta, nombrando el objetivo del usuario final en dicho escenario.
2. Explicitar qué es lo que se va a entregar al cliente.

3. Separar objetivos del equipo, proyecto y producto.
4. Mejorar explicación de estimación y cronograma.
5. Evaluar la posibilidad de “ejecución de test” que procese una cantidad acotada de archivos y muestre el resultado.

Acciones para implementar las oportunidades de mejora:

1. Generar un ejemplo concreto para la explicación, indicando objetivo.
    a. Fecha estimada: Próxima revisión.
2. Explicitar los entregables que se harán.
    a. Fecha estimada: Próxima revisión.
3. Separar los objetivos según su categoría.
    a. Fecha estimada: Próxima revisión.
4. Mejorar la presentación en cuanto a estimación y cronograma
    a. Fecha estimada: Próxima revisión.
5. Evaluar la posibilidad de contar con una “ejecución de test” que procese una cantidad acotada de archivos y muestre el resultado.
    a. Fecha estimada: Próxima revisión.

Informe de segunda revisión

Proyecto: Framework para procesamiento de archivos DICOM

Fecha: 16/08/2021

Carrera: Licenciatura en Sistemas

Revisión nro: 2

Revisor: Pablo Hernandez

Fortalezas del equipo:

1. Integrantes con distintos perfiles (desarrollo, gestión y análisis) que nos permite separar las tareas según la especialidad de cada uno.
2. Capacidad de sortear obstáculos.
3. Comunicación constante y fluida.
4. Correcta elección de metodología.
5. Problema claramente definido
6. Explicación del formato (contexto) del archivo es muy apropiada.
7. Se evidencia la investigación tecnológica y está bien explicada.
8. Estrategia apropiada de QA y SCM
9. Adecuada selección del Ciclo de vida.

Oportunidades de mejora:

1. Recalcar que el usuario final es un analista de datos, el cual tiene capacidades técnicas para el correcto uso de nuestro framework.
2. Explicar una sola vez los conceptos de data warehouse, data lake y data mart. No repetirlo en la sección de arquitectura.
3. En la sección de arquitectura marcar de qué manera fuimos abarcando cada uno de los requerimientos. Este punto queda sujeto a si tomamos la decisión de cambiar el orden (arquitectura antes o después de nombrar cuáles son los requerimientos).
4. Nombrar cómo se miden los objetivos del equipo y del proyecto.
5. En la sección de contexto dejar explícitos los dos puntos más importantes que hacen a la complejidad de los archivos DICOM: Archivos diferentes y el gran tamaño que puede tener cada uno de ellos.

6. Mejorar contenido visual (agrandando el “80%”, por ejemplo) en la sección de análisis de datos.
7. Cambiar la sigla “UAT” por un término en español. Por ejemplo “Test de Aceptación”.
8. Cambiar de lugar la sección de entregables. Pueden estar al principio en una sección llamada “Alcance”.
9. Simplificar la explicación de las tablas de estimaciones en horas quitando el promedio, variación en horas y no en % y ordenar las columnas de manera más lógica (estimado inicial - re estimado - real)
10. Cuando se da algún dato o información nombrar a qué conclusiones se llega a raíz de lo que se está presentando.
11. Trabajar mejor la explicación de los riesgos diciendo cuales fueron en total, no explicar cada uno de los que se muestra si no centrarse en uno en particular y explicarlo en detalle desde cómo surge, cómo se manejó, cuales eran los planes de mitigación y de contingencia. Explicar por que se muestran esos riesgos que se seleccionan y remarcar cada cuanto se evaluaron los riesgos (mensual).

Acciones para implementar las oportunidades de mejora:

1. Trabajar en las notas de cada uno de los oradores para abarcar las oportunidades de mejora 1, 2, 3, 4, 5, 9, 10 y 11.
    a. Fecha estimada: 07/10/2021
2. Modificar contenido de la presentación acorde a los comentarios hechos en las oportunidades de mejora 4, 5, 6, 7, 8, 9 y 11.
    a. Fecha estimada: 07/10/2021