Universidad ORT Uruguay

Facultad de Ingeniería

iPanel Educativo

Panel de control para la toma de decisiones en el desarrollo de textos educativos

Entregado como requisito para la obtención del título de Licenciado en Sistemas

Martín Martínez – 154327
Guzmán Porro – 160912
Carolina Romero – 143844
Matías Viera – 150196

Tutor: Ignacio Valle

2014

Declaratoria de autoría

Nosotros, Martín Martínez, Guzmán Porro, Carolina Romero y Matías Viera, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

• La obra fue producida en su totalidad mientras realizábamos el Proyecto final de grado de Licenciatura en Sistemas;

• Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;

• Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;

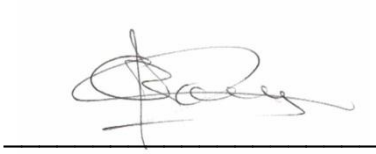
• En la obra, hemos acusado recibo de las ayudas recibidas;

• Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;

• Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Martín Martínez
11/09/2014

Guzmán Porro
11/09/2014



Carolina Romero
11/09/2014

Matías Viera
11/09/2014

Agradecimientos

En primer lugar queremos agradecer a nuestras respectivas familias por el apoyo incondicional durante esta etapa y las transcurridas durante toda la carrera.

A la Universidad ORT por brindarnos todas las herramientas posibles a lo largo de nuestra formación profesional. También agradecer a los expertos de ORTsf por su buena disposición en todo momento.

Especialmente agradecemos a nuestro tutor, Ignacio Valle, por compartir nuestro esfuerzo en este período y estar siempre interesado en nuestro trabajo y en cómo vivíamos la experiencia del proyecto.

Agradecemos también a EDU Editorial por confiar en nosotros para la realización de una herramienta que cumpla satisfactoriamente con sus necesidades. Además queremos hacer una mención especial, a Braulio de León, a quien le agradecemos su colaboración y disponibilidad en todo momento para con el equipo.

Abstract

iPanel Educativo es un panel de control creado a medida para la editorial EDU Editorial, que se dedica a la elaboración de textos educativos inteligentes para las computadoras entregadas bajo el marco del Plan Ceibal en Uruguay.

El panel fue desarrollado a partir de los requisitos planteados por el cliente y se trata de una aplicación web que permite a los usuarios analizar datos del comportamiento de los escolares que interactúan con los contenidos de EDU Editorial, con el objetivo de perfeccionar la calidad de los mismos para un mejor aprendizaje del alumno.

El propósito del proyecto fue crear una herramienta de análisis que acompañe al personal de EDU Editorial en esta primera etapa de pruebas de sus textos, permitiéndoles investigar de manera simple los resultados de la utilización de los mismos y tomar decisiones al respecto.

La aplicación se encuentra desarrollada en *Python* como lenguaje de programación, *Django* como *framework* para el desarrollo web, *Aptana* 3 como *IDE* y *MySQL* Server para la gestión de la base de datos.

Palabras clave

Panel de control, Textos educativos, Plan Ceibal, Gráficas, Aplicación web, *Scrum*, *Python.*

1. Índice

Índice de ilustraciones

Índice de tablas

2. Glosario

A

Aptana Studio 3: entorno de desarrollo integrado de software libre basado en eclipse desarrollado por Aptana Inc., provee soporte para lenguajes como: Php, Python Ruby, CSS, Ajax, HTML y Adobe AIR.

B

Backend: parte del software que procesa la entrada desde el frontend.

Bitbucket: servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de revisiones.

Branch: duplicación de una porción de código fuente existente en el repositorio de versiones con el fin de utilizarlo en paralelo para llevar a cabo otro tipo funcionalidades de desarrollo en el producto de software.

Bug: error o fallo en un programa de computador o sistema de software que desencadena un resultado indeseado.

Burndown chart: representación gráfica del trabajo por hacer en un proyecto en el tiempo, especialmente utilizado para Scrum.

C

Coach: entrenador o director técnico.

Commit: acción de comprometer, refiere a la idea de consignar un conjunto de cambios "tentativos, o no permanentes".

Controller: intermediario o capa intermedia entre la vista y el modelo.

CSS: lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

D

DiffMerge: herramienta que se puede integrar a *SourceTree* para realizar el manejo de versiones de código fuente.

Dump: archivo de texto que contiene una serie de sentencias SQL que permiten crear la estructura de una base de datos y cargarla con ciertos datos.

E

ECS: Elemento de Configuración del Software. Son los elementos creados durante el proceso.

F

Feedback: retroalimentación

Framework (marco de trabajo): define un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Frontend: parte del software que interactúa con el o los usuarios.

Full time: tiempo completo de trabajo.

G

Gantter: herramienta para la elaboración de cronogramas desarrollado por Google Inc.

GIT: software de control de versiones, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Google Calendar: servicio gratuito de calendario en línea desarrollado por Google Inc.

Google Drive: servicio gratuito de almacenamiento de archivos en línea desarrollado por Google Inc.

H

Hangouts: aplicación de mensajería instantánea desarrollada por Google Inc.

Historia de Usuario: representación de un requisito del sistema escrito en lenguaje de usuario.

I

Input: entrada, información recibida.

J

JavaScript: lenguaje de programación que se puede utilizar para construir sitios Web y para hacerlos más interactivos. Puede interactuar con el código HTML, permitiendo a los programadores web utilizar contenido dinámico.

jQuery: biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML.

M

MySql Workbench: es una herramienta visual de diseño de bases de datos que integra desarrollo de software, Administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos *MySQL*.
MVC: patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

O

Ofimática: conjunto de técnicas, aplicaciones y herramientas informáticas que se utilizan en funciones de oficina para optimizar, automatizar, y mejorar tareas y procedimientos relacionados.

ORTsf: Laboratorio ORT software factory.

Output: salida, información enviada.

R

RACI: matriz de la asignación de responsabilidades, Responsable, Aprobador, Consultado, Informado.

Responsive design (diseño responsivo): hace referencia a la capacidad del sistema de responder a distintas condiciones que el usuario puede controlar, como la resolución de su pantalla, el tamaño del navegador, el dispositivo mediante el cual accede, el sistema operativo, entre otro.

S

SaaS: Software como Servicio en la Nube

SCM (Software Configuration Management): refiere a la gestión de la configuración de software en el proyecto.

Script: archivo de órdenes, archivo de procesamiento por lotes que se almacena en un archivo de texto plano, su uso habitual es combinar componentes, interactuar con el sistema operativo o con el usuario.

Scrum: marco de trabajo basado en el desarrollo ágil de software.

Sourcetree: cliente de escritorio de Git para Mac o Windows.

Sprint: nombre que recibe una iteración de Scrum.

Sprint Backlog: conjunto de historias de usuarios que se planifica construir en la iteración.

SQA (Software Quality Assurance): refiere al aseguramiento de la calidad del software en el proyecto.

Stakeholders (interesados): quienes pueden afectar o son afectados por las actividades de una empresa.

Story Points (puntos de las historias): medida de velocidad del desarrollo utilizada en Scrum.

Sublime Text: es un editor de texto y editor de código fuente está escrito en C++ y Python: es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

T

TargetProcess: herramienta web para la gestión del proyecto bajo el marco de trabajo Scrum.

Test A/B: herramienta/proceso para optimizar un sitio web, no solo en lo relativo a conversión, sino en todas las facetas que implican la interacción del usuario con dos o más versiones de una misma página de un sitio web sin que éste lo sepa.

U

Unit Test (prueba unitaria): forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

3. Introducción

3.1 Introducción al proyecto

El presente documento tiene como objetivo describir el proceso realizado durante el proyecto de grado que denominamos: iPanel Educativo. El mismo fue realizado por los alumnos Martín Martínez, Guzmán Porro, Carolina Romero y Matías Viera en el ámbito académico, como requisito para obtener el título de Licenciado en Sistemas en la Universidad ORT Uruguay, en el período de Abril de 2014 a Setiembre de 2014 bajo la tutoría del Ing. Ignacio Valle.

El proyecto fue elegido de entre todos los proyectos presentados a la Facultad de Ingeniería para realizarse como trabajo de fin de carrera. Se trabajó directamente con los socios fundadores de Editorial, cliente que presentó el proyecto, y con el personal designado por los mismos.

El producto abarca el análisis, diseño y desarrollo de una aplicación web que será utilizada por algunos usuarios de EDU Editorial con el fin de obtener información para la mejora continua de los textos educativos que elaboran.

3.2 El equipo

El equipo de proyecto fue conformado por cuatro compañeros de la carrera que se conocieron durante distintas materias compartidas y se consolidaron para trabajar por primera vez como equipo con el fin de realizar el proyecto final.

En el plano laboral, Guzmán se desempeña como desarrollador web UI en Globant, Martín realiza tareas de desarrollo y testing en Xseed, Matías forma parte de KPMG en el área consultoría en tecnología de la información y Carolina trabaja como analista en Pronto! y cuenta con experiencia en gestión de proyectos.

3.3 Objetivos del equipo

Los principales objetivos en los que se enfocó el equipo fueron: la adquisición de experiencia profesional realizando un proyecto, que si bien se encontraron en un marco académico, es muy semejante a la realidad laboral; el aprendizaje de nuevas tecnologías y metodologías de trabajo; la creación de un buen clima de trabajo, colaborativo y que permitiera la retroalimentación y el intercambio de conocimiento entre los integrantes; y no menos importante: la aprobación del proyecto final de carrera para obtener el título profesional.

3.4 ORT Software Factory

El Laboratorio denominado ORT Software Factory, ORTsf, es una organización académica dedicada a la enseñanza de prácticas de Ingeniería de Software, a la mejora de procesos de software, a la transferencia de tecnología a la industria y a la producción de software. [1]

Es en este marco donde el equipo de proyecto se encontró interesado ante el caso de una empresa que había presentado su necesidad de contar con un producto específico que los estudiantes de ORT próximos a recibirse podían desarrollar. Es así como a través del laboratorio se generó el primer contacto que diera motivo al proyecto en cuestión.

Para entender un poco más el entorno conceptual del laboratorio, a continuación compartimos el extracto de la página web de Software Factory sobre su visión, misión, política de calidad y objetivos.

3.4.1 Visión

Ser líderes en la generación de conocimiento sobre la producción de software de calidad. Ser un referente dentro de la Universidad, en el medio y la región, en la creación y aplicación de prácticas de ingeniería de software para la producción de productos de calidad.

3.4.2 Misión

El laboratorio de Ingeniería de Software es una organización abocada a la formación de profesionales que desarrollen productos que satisfagan a sus clientes, focalizando la atención en la producción de software de forma industrial y en proveer tecnología probada al mercado.

3.4.3 Política de Calidad

ORT Software Factory es una organización abocada a la producción de software de forma industrial, focalizando la atención en la formación de profesionales que realicen productos que satisfagan a sus clientes.

La meta perseguida es consolidar una posición de liderazgo en el mercado en la generación de conocimiento sobre la producción de software de calidad.

La mejora continua de la calidad de los productos se quiere lograr a través de la mejora del proceso de desarrollo de software, capacitando y brindando la posibilidad

de que cada integrante pueda desarrollar sus habilidades, y jerarquizando al cliente como el objetivo principal.

Dar tecnología probada al mercado productor de software de la región, y divulgar resultados como forma de mejorar el entorno son algunos de los objetivos de la organización.

3.4.4 Objetivos

Los objetivos académicos de ORTsf son:

- Formar en la producción regida por un Proceso de Software y en técnicas, herramientas y métodos de Ingeniería de Software.

  Todos los desarrollos que se realizan en el marco de ORTsf están regidos por procesos de software adaptados de un proceso rector genérico, el cual es mantenido y mejorado por un grupo de gestión de procesos a nivel de la organización.

- Promover el trabajo en equipo.

  Todos los proyectos realizados en el ámbito de Software Factory son realizados en grupos de entre cuatro y ocho personas, en los cuales hay roles bien definidos y se promueve el espíritu de equipo a través de la realización de evaluaciones grupales a lo largo de la experiencia.

- Transferencia de tecnología a la industria.

  La organización realiza transferencia tecnológica a la industria de varias formas. Desde su creación, ORTsf ha entrenado a más de 300 profesionales que actualmente se desempeñan en cargos técnicos y gerenciales en distintas empresas privadas y estatales. Además, ha desarrollado proyectos originados a iniciativa de empresas nacionales, que envían a su personal a realizar cursos de posgrado y que durante su pasantía por ORTsf realizan proyectos en la propia organización, orientados a la mejora de sus propios procesos. En la actualidad, la mayoría de los proyectos son desarrollos para empresas o para la invención de productos.

Para lograr estos objetivos, en ORTsf se integra a un conjunto de equipos de proyecto al comienzo de cada semestre lectivo, y un conjunto de tutores de rol y de tutores de grupo que tiene por finalidad apoyar a los alumnos a lo largo del semestre. Los grupos están integrados por alumnos de grado y posgrado, y son “tutoreados” por docentes con experiencia en el área de Ingeniería de Software.

Los clientes fundamentales de ORTsf son los alumnos de las carreras de Ingeniería de Sistemas y Licenciatura en Análisis de Sistemas de Información de la Facultad

de Ingeniería de la Universidad ORT Uruguay. Por esta razón sus principales clientes son los alumnos receptores de conocimiento así como la propia Facultad representada por el Decano; en segundo lugar se considera a las organizaciones receptoras de productos de software y de transferencia de tecnología; y en tercer lugar se identifica a la sociedad, como el medio en el cual se insertarán los profesionales al egresar.

3.5 Presentación del cliente

EDU Editorial es la primera editorial educativa de América Latina dedicada exclusivamente a las iniciativas donde se aplica el modelo 1a1 (un laptop por alumno en clase).

La empresa provee textos inteligentes que sincronizan en el aula tecnológica una nueva metodología para la enseñanza y el aprendizaje: adaptada, equilibrada, que facilita el trabajo docente, motiva el alumno y ayuda a alcanzar mejoras en los niveles educativos afectivos, cognitivos y sociales. EDU Editorial abarca las distintas áreas disciplinares, tanto de primaria como de secundaria, y su solución es desarrollada a medida para cumplir con los contextos y currículos nacionales.

Los textos inteligentes propuestos por EDU Editorial están concebidos a partir de una combinación innovadora entre pedagogía, software y contenido educativo. Aprovechan el inmenso potencial tecnológico que está rápidamente creciendo en la región, brindan contenidos digitales e impresos en formato de lección, actividades, evaluaciones, guías didácticas y elevan el estándar técnico en aplicaciones curriculares para el modelo 1a1.[2]

Cuasimodo es el nombre de la plataforma desarrollada por EDU Editorial, que se utiliza para los textos inteligentes, que almacena de forma continua información sobre el uso de los mismos, de la capacidad y aprendizaje del alumno; y del avance del curso, entre otros. Estos datos tienen gran valor si se utilizan para la mejora no solo de la aplicación y la experiencia del alumno sino para la mejora de los textos explicativos, los ejercicios y la propuesta pedagógica.

3.6 Propósito y Alcance

El propósito de este proyecto es crear un panel de control a partir del análisis de los datos que son recogidos por una aplicación de la editorial y transformar los mismos en información útil para el personal de EDU Editorial que buscan la mejora de los libros.

El proyecto se realizó aplicando técnicas y prácticas de Ingeniería de Software adquiridas a lo largo de la carrera. El alcance del mismo fue determinado en gran

parte al comienzo del proyecto con las definiciones que el cliente tenía pensadas para el producto, ajustadas al tiempo que se disponía para la realización del mismo y a la prioridad asignada a estos requerimientos. Sin embargo, durante el transcurso del proyecto el alcance fue sufriendo modificaciones, tanto nuevos requerimientos que se sumaron como la eliminación de otros.

En este documento se describe el producto desarrollado, el proceso y metodología utilizados para su construcción y todas las características propias del mismo.

3.7 Organización del documento

A continuación se describe al lector cómo se encuentra organizado el presente documento.

Descripción del proyecto: se detalla el cliente y su contexto, el producto en sí mismo con sus funcionalidades a al nivel y la elección de la metodología de trabajo.

Ingeniería de requerimientos: describe las técnicas y las estrategias de relevamiento utilizadas en el proceso de ingeniería de requerimientos; detalla los requerimientos funcionales y no funcionales del sistema, su priorización y validaciones.

Arquitectura de software: se explican las decisiones tomadas en cuanto al diseño y la arquitectura del sistema desarrollado, se describen los atributos de calidad, diagramas y vistas correspondientes; las decisiones en cuanto a la tecnología y las estrategias de desarrollo y testing utilizadas.

Aseguramiento de la calidad: el capítulo describe las actividades realizadas en función de la calidad para asegurar la misma durante el proceso de desarrollo y el producto en sí mismo; los estándares definidos, cuáles fueron las validaciones y criterios de aceptación del producto, las pruebas y métricas seleccionadas y sus resultados.

Gestión de la configuración: se detallan las herramientas y los ambientes de trabajo utilizados en el proyecto, los elementos de la configuración, las características del repositorio de código y de documentación, cómo se realiza la gestión de versiones, el control de cambios y los respaldos de todo el proyecto.

Gestión del proyecto: se describe el proceso de la gestión del proyecto; el marco de trabajo utilizado, el ciclo de vida del proyecto, las responsabilidades y actividades de los miembros del equipo, cómo se realizó la gestión de las comunicaciones y la gestión de riesgos, la planificación y evolución que siguieron las iteraciones en el proyecto, los resultados totales en función de las métricas definidas; y las decisiones tomadas en este proceso.

Lecciones aprendidas: se detallan las principales lecciones aprendidas en el transcurso del proyecto.

Conclusiones: se describen las conclusiones finales del proyecto.

Bibliografía: se referencian las fuentes de los materiales bibliográficos utilizados durante el proyecto.

Anexos: se adjuntan documentos complementarios del proyecto.

4. Descripción del proyecto

4.1 Introducción

El objetivo de este capítulo es el de brindar una descripción general acerca del proyecto iPanel Educativo.

Describiremos al cliente y sus necesidades, el objetivo propuesto por el grupo de trabajo y el deseado por el cliente, las características más destacadas en cuanto a funcionalidades del producto, así como también la organización de trabajo por parte del equipo.

4.2 Descripción del cliente y su contexto

EDU Editorial es una organización que se dedica a la creación y optimización de textos digitales inteligentes para el aprendizaje. Es la primera editorial educativa de este estilo en América Latina.

Aplica el modelo 1a1 (un pc por alumno en clase), ayudando a alcanzar mejoras en los niveles educativos gracias al aprovechamiento del potencial tecnológico.

Elabora los textos inteligentes que niños y niñas de 5to y 6to de escuela utilizan en las aulas a través de sus computadoras portátiles entregadas en el marco del Plan Ceibal.

Ilustración 4-1 Pantalla principal de Cuasimodo

Actualmente se encuentra en Uruguay con proyección a expandirse en el resto de Latinoamérica. Este año comenzó con un plan piloto donde algunos maestros de distintas escuelas públicas del país incorporaron al dictado de sus cursos los libros digitales de la editorial.

La plataforma utilizada actualmente se denomina Cuasimodo [3] (aunque en un futuro se denominará Edu). Esta cuenta con varias áreas de conocimiento como ser: ciencias naturales, matemáticas, lenguas y más, junto con sus respectivas materias por ejemplo: física, química, geología, entre otras que saldrán al mercado en el correr del año. Para cada una de estas se encuentra un libro digital donde los niños acceden para leer el material teórico del tema dictado en clase, hacer los ejercicios correspondientes, realizar las tareas domiciliarias y aprender a través de experimentos y juegos de aventura.

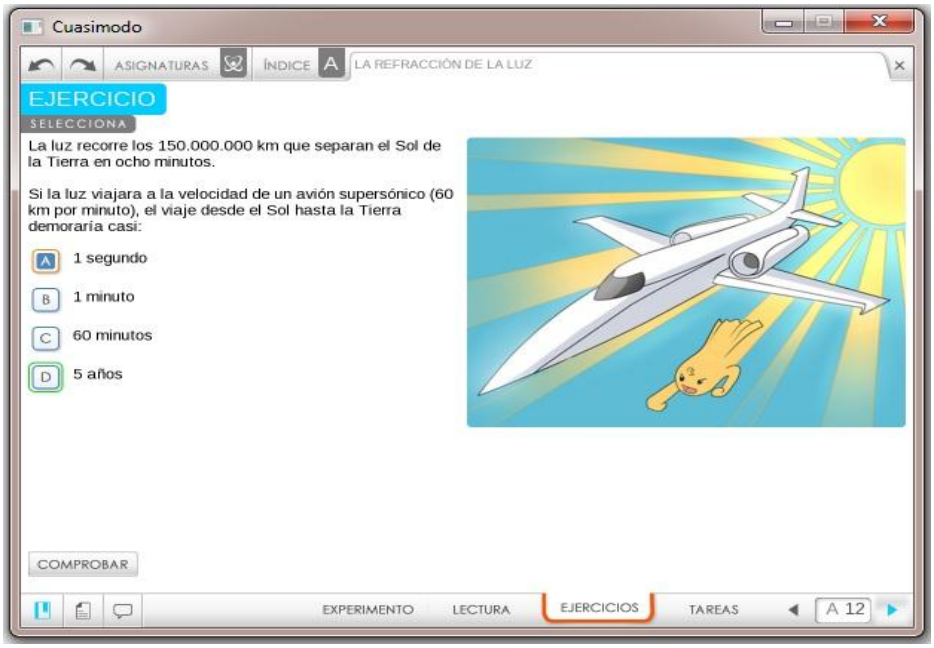


Ilustración 4-2 Ejemplo de ejercicio en Cuasimodo

Al momento de comenzar con el proyecto, EDU Editorial contaba con ciertos registros que la plataforma guarda en una base de datos y la posibilidad de guardar otros más sobre los resultados generados a partir de la experiencia de los niños con el libro. Hasta el momento estos datos que se iban a almacenar para utilizar no habían sido completamente definidos y tampoco estaban agrupados de manera que brindaran información útil para la editorial. No se contaba con ninguna herramienta informática específicamente diseñada para ayudar a visualizar los datos almacenados y que ayudara a la toma de decisiones para la mejora de los libros digitales.

4.3 Objetivos del cliente

Debido a que EDU Editorial cuenta con muchos datos obtenidos de la plataforma educativa que son guardados en una base de datos, pero hasta el momento no tenía forma de analizarlos, se detectaron las principales necesidades:

- Analizar acerca de que datos se almacenaban en la plataforma Cuasimodo.
- Sugerir nuevos elementos de información a almacenar en la base de datos.
- Contar con un sistema a medida que consuma estos datos y proporcione información útil de un modo gráfico y específico a sus intereses.
- El cliente manifestó el deseo de que el sistema tenga una determinada usabilidad, basándose en experiencias personales anteriores, siendo esta una de las premisas fundamentales.
- Interés por parte de la empresa que esa información sea visualizada solo por ellos en sus servidores.
- Implementación de tecnología de código abierto (*php* o *Python* sobre *MySQL*), ya que cuentan con la posibilidad de desarrollar funcionalidades específicas por parte del personal de la empresa capacitado en dichas tecnologías, sin invertir en formación y licenciamiento de herramientas propietarias.

Estos objetivos tenían el propósito de ayudar a que los usuarios del nuevo sistema pudieran tomar decisiones para la mejora de la plataforma en estos aspectos:

- Modificar contenidos y ejercicios del libro, con el fin de mejorar el entendimiento de los textos que se encuentran en Cuasimodo.
- Analizar la experiencia de uso de los alumnos en los ejercicios, según los ejercicios más utilizados, basándose en métricas definidas manualmente y de forma automática según promedio de utilización de los ejercicios.
- Estudiar la propuesta pedagógica dela aplicación Cuasimodo, analizando que áreas de aprendizaje, materias y unidades son las más frecuentes o las menos utilizadas por los usuarios.
- Identificar perfiles de alumnos mediante información gráfica. Esto ayudaría a definir ciertos perfiles de usuarios del libro digital que cuenten con características en común para poder adaptar los textos a sus gustos personales y así motivar su aprendizaje.

4.4 El producto

Definido el objetivo de contar con información capaz de ser analizada para colaborar en la toma de decisiones para la mejora del libro digital, el equipo de proyecto especificó las funcionalidades primarias que debía tener la solución e investigó las distintas posibilidades que podrían satisfacer las necesidades detectadas.

4.4.1 Actores

Los usuarios de la solución son aquellos que pertenecen a la Alta Gerencia de la empresa cuya formación profesional no es relacionada al área de Tecnologías de la Información. A su vez no cuentan con experiencia previa en sistemas de análisis de datos.

4.4.2 Descripción funcional

A continuación se listan las funcionalidades primarias detectadas:

- Analizar información sobre resultados de los ejercicios de los libros según variables guardadas en la base de datos (simples y combinadas).
- Visualizar resultados de búsqueda en gráficas y tablas.
- Generar reportes predefinidos de búsqueda.
- Ordenar los resultados por una determinada variable de búsqueda.
- Alertas automáticas sobre determinadas situaciones encontradas en la aplicación.
- Comparar resultados de búsquedas a través del tiempo.
- Contar con atajos para las búsquedas más frecuentes (favoritos).
- Exportar los resultados obtenidos en distintos formatos (PDF, CSV).

4.4.3 Investigación de posibles soluciones

Tras realizar el relevamiento de requisitos iniciales, el equipo de proyecto investigó acerca de las distintas posibilidades que existían para satisfacer las necesidades del cliente.

Las alternativas posibles fueron la adaptación de una herramienta ya existente o el desarrollo de una solución a medida.

El cliente contaba con conocimientos acerca de algunas herramientas existentes en el mercado. Por políticas internas de la empresa, intentan no adquirir productos prefabricados para los procesos críticos de la empresa, debido a esto manifestó su intención de contar con un desarrollo a medida.

A raíz de estas consideraciones el equipo de proyecto investigó algunas herramientas ya constituidas en el mercado, que podrían satisfacer las necesidades del cliente. Este análisis fue realizado sobre *QlikView* y *Pentaho Community* en base a revisiones en internet, y consultas a expertos en el negocio del análisis de datos mediante *Business Intelligence*.

Si no se tuvieran en cuenta las condiciones que el cliente requirió para la solución, cualquiera de las herramientas analizadas podría haber sido utilizada. Ver Anexo 1 – Investigación de posibles herramientas para la solución.

4.4.4 Condiciones requeridas por el cliente

El cliente definió los siguientes aspectos claves para el desarrollo de la solución a llevar a cabo.

- No se tenía intención en invertir en licenciamiento de herramientas.
- La solución debe ser programada en el lenguaje de programación de código abierto *Python* sobre una base de datos *MySQL*, para que pueda ser mantenida por el personal con que cuenta la empresa.
- La propiedad del código queda enteramente a cargo de la empresa mediante carta de cesión de derechos.
- Dados los buenos resultados personales de la alta gerencia utilizando herramientas de código libre desarrolladas a medida anteriormente, la usabilidad de la solución fue definida como un aspecto fundamental.

4.4.5 Validación de la decisión

Junto al cliente se analizaron las distintas posibilidades investigadas que podrían satisfacer su necesidad. Y en común acuerdo se seleccionó la realización de un panel de control diseñado y desarrollado a medida, en base a criterios de usabilidad validados por EDU Editorial.

Posteriormente se entregó al cliente un documento con la descripción de los principales requerimientos funcionales y no funcionales, las restricciones y compromisos pautados que fueron acordados, con el detalle de las versiones de las tecnologías a utilizar en el proyecto.

5. Ingeniería de requerimientos

5.1 Técnicas y resultados de relevamientos

El proceso de relevamiento y especificación de requerimientos no cuenta con una documentación extensa ya que se utilizó una metodología ágil de desarrollo. Este proceso consistió en realizar entrevistas en primera instancia con los directores de EDU Editorial y el equipo de proyecto. En estas reuniones iniciales se dio lugar al entendimiento del negocio, principales actividades que desarrolla el cliente y la realización de tormenta de ideas, las cuales fueron enriquecedoras para ayudar a definir los primeros objetivos con que debía contar la solución deseada, generando así la especificación de requerimientos a nivel general.

En reuniones posteriores se llevaron a cabo entrevistas entre un referente asignado por los directores de la empresa y el ingeniero de requerimientos del equipo de proyecto. Estas instancias fueron planificadas en base a la información que se necesitaba para la realización de las funcionalidades. Mediante estas se relevaron los requerimientos aumentando el nivel de detalle por parte del cliente, para luego ser analizados en reuniones del equipo y posteriormente validados con el referente asignado.

Una de las características de los requerimientos fue la volatilidad de los mismos, dado que el cliente no tenía en claro que deseaba, debido a esto fue que se consideró la utilización de una metodología ágil.

Además se observó la plataforma Cuasimodo, que datos podía proveer, cómo los alumnos interactuaban con la plataforma y qué resultados se obtenían según las distintas actividades y ejercicios. Esto permitió el entendimiento de cómo era la usabilidad y que comportamientos podrían llegar a ser beneficiosos detectar.

La información, decisiones y opiniones aportadas en estas instancias fueron registradas con el formato de “minutas”, donde se describen participantes, lugar donde se llevaron a cabo, fecha de realización, temas a tratar y aportes sobre los temas tratados.

5.2 Requerimientos

En esta sección se describen las funcionalidades pedidas y sugeridas, los requerimientos no funcionales, priorizaciones, validaciones y restricciones con que se contaba para el desarrollo del producto deseado por el cliente.

5.2.1 Funcionales

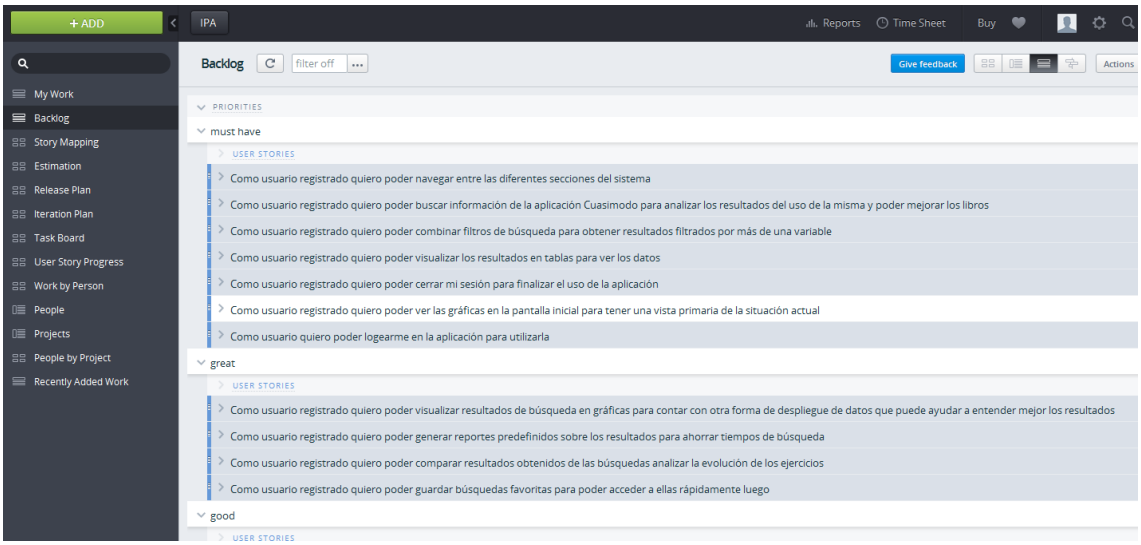


Ilustración 5-1Impresión de pantalla de un segmento del Product Backlog

Acceso autenticado al sistema: como usuario quiero poder ingresar al sistema con mi usuario y contraseña.

Panel Principal: como usuario registrado quiero ingresar al sistema y poder visualizar gráficas acerca de la utilización de los ejercicios.

Alertas: como usuario registrado quiero que el sistema notifique cuando se produzca algún hecho de que la cantidad de ejercicios utilizados promedio o manual es superado.

Consultas: como usuario registrado quiero poder realizar consultas sobre los datos almacenados con distintos filtros aplicados. Los resultados deben mostrar una gráfica y la posibilidad de verlos en tablas.

Consultas rápidas: como usuario registrado quiero poder realizar las consultas por los ejercicios menos utilizados y ejercicios con un puntaje promedio inferior o igual a 0,1 y ejercicios con puntaje promedio mayor o igual a 0,9 rápidamente.

Ayuda: como usuario registrado quiero poder acceder a una sección de ayuda con información útil para la utilización del sistema.

Exportar resultados de búsqueda: como usuario registrado quiero poder exportar los resultados generados por medio de una búsqueda realizada, a archivos con

formato PDF o formato CSV. La gráfica generada debe ser posible exportarla en formato jpg.

Favoritos: como usuario registrado quiero poder guardar combinaciones de uno o más filtros como mis filtros favoritos en las búsquedas.

Comparativas: como usuario registrado quiero poder comparar resultados obtenidos por filtros que seleccione, según intervalos de tiempo que defina, con el objetivo de ver la evolución de los ejercicios.

Operaciones: quiero que el sistema pueda contar, realizar promedio, ordenar información de menor a mayor y viceversa, que me muestre mínimos y máximos y liste resultados.

Ver Anexo 2 – Listado de *User Stories*

5.2.2 No funcionales

Se describen a continuación los requerimientos no funcionales definidos en común acuerdo con el cliente.

- Desarrollo de software: lenguaje de programación *Python* 2.7.6.
- Fuente de datos: base de datos *MySQL* 5.6+.
- Usabilidad: disposiciones visuales claras e intuitivas para el mejor análisis de la información.
- Modificabilidad: desarrollo en módulos con el objetivo de que el impacto en las modificaciones de las funcionalidades, fuera el menor posible.
- Testeabilidad: facilidad del software para mostrar los errores a través de pruebas.

5.2.3 Volatilidad de requerimientos

Debido a la cantidad de cambios realizados a lo largo del proyecto a solicitud del cliente, se describen que requerimientos funcionales y no funcionales se vieron afectados durante el transcurso del mismo.

Principales requerimientos funcionales modificados

Requerimiento	Descripción	Modificación	Motivo
Inteligencia artificial al sistema	Según los datos almacenados aplicar técnicas de inteligencia artificial al panel de control	Quitar este requerimiento del *Product Backlog*	Previo al comienzo de la etapa de desarrollo, EDU Editorial se reunió con un experto del tema el cual les sugirió que dejaran esta idea para cuando la empresa contara con una cantidad de datos mayor
Detección de perfiles de alumno	Según los resultados obtenidos el sistema debía identificar el perfil de ese alumno (inclinado a las ciencias, a lo artístico, etc.)	Dar de baja a esta funcionalidad	En el transcurso del proyecto el responsable de esta funcionalidad se retiró de la empresa. Esta ausencia no fue recuperada hasta el momento
Comparativas en 19 intervalos	El cliente en principio había solicitado que las comparativas fueran mostradas en 19 intervalos	La empresa solicitó que las comparativas fueran vistas en 4 cuadrantes	El *Product Owner* manifestó que el requerimiento original no iba a ser favorable para el buen uso de la aplicación
Información a mostrar en las tablas tras realizar búsquedas	EDU Editorial solicitó que se vieran los resultados obtenidos en los ejercicios tras la realización de consultas	Se solicitó que los resultados se mostraran luego de hacer clic en un enlace de la tabla. Además se solicitó que se mostraran el nombre del docente y la fecha de nacimiento del usuario	Análisis interno que realizó la empresa en cuanto a la utilidad de datos a mostrar

Ubicación de los filtros	La empresa solicitó que las gráficas fueran por encima de los filtros.	Luego no se sabía si era la mejor disposición, por lo cual se definió por realizar test de usabilidad (Test A/B) para definir este requerimiento	Recomendaciones de expertos de negocio a la empresa
Cantidad de alertas visibles rápidamente	El cliente deseaba ver una cantidad determinada de las alertas que el sistema generaba, de forma rápida	Se solicitó que la cantidad de visualizaciones de alertas generadas automáticamente en el sub menú "Alertas" fuese modificable	El *Product Owner* manifestó que se desearía ver una cantidad mayor o menor de alertas dependiendo del período del año, para aumentar la usabilidad

Tabla 5-1Principales requerimientos funcionales modificados

Requerimientos funcionales añadidos

Requerimiento	Descripción	Motivo
Agregar notas rápidas	Posibilidad de, en cualquier momento, ingresar una nota rápida al sistema como ayuda memoria a lo analizado	El equipo de proyecto detectó que ante la realización de algún tipo de análisis podía ser de gran utilidad tomar nota de lo observado para una posterior decisión
Detección de consultas frecuentemente utilizadas	El sistema ofrecerá en el sub menú de "Consultas rápidas" las 5 consultas más utilizadas en la última semana	Ante la repetición de un consulta que el usuario realice y este no la perciba, el sistema se lo sugerirá de forma automática, con el fin de mejorar la usabilidad reduciendo la cantidad de clics que el usuario debe realizar para poder efectuar dicha consulta.
Recuperación de contraseña	En caso que el usuario del panel de control se olvide su contraseña de acceso al sistema, se deberá implementar un mecanismo de recuperación de la misma sin necesidad de la presencia de un usuario administrador	Al no contar con la figura de un usuario administrador del sistema el equipo de proyecto sugirió un mecanismo para que los usuarios finales del panel de control pudieran volver a ingresar nuevamente

Actualización manual de alertas	Dar la posibilidad de que el usuario del panel de control pueda generar una actualización de las alertas que se generan en el sistema	Se evita la espera del período asignado en el sistema para la generación de las alertas

Tabla 5-2 Principales requerimientos funcionales añadidos

Principales requerimientos no funcionales modificados

Requerimiento	Descripción	Modificación	Motivo
Solución en *php*	El cliente solicita que el sistema esté en lenguaje de programación *php*	El cliente solicita que el sistema esté en lenguaje de programación *Python*	Razones propias del cliente debido a que cuenta con personal contratado para el desarrollo de funcionalidades en *Python*
Desarrollo sobre ambiente en servidores Amazon	EDU Editorial comunicó que se iba a disponer de un ambiente de desarrollo en servidores Amazon	Se desarrolló en servidores locales	El cliente no pudo disponer de espacio para configurar un ambiente para el desarrollo de la solución.

Tabla 5-3 Tabla de requerimientos no funcionales modificados

5.2.4 Priorización

Al comienzo de cada *Sprint* se llevaba a cabo la reunión de planificación donde se revisaban las próximas *User Stories* a realizar o cambios de funcionalidades entre el grupo de trabajo y el *Product Owner* en el *Sprint*.

Estos cambios quedaban registrados en las minutas y se modificaba en la herramienta de gestión del proyecto *TargetProcess*.

5.2.5 Validaciones

Al término de cada *Sprint* en la reunión de retrospectiva se realizaba el análisis de las funcionalidades desarrolladas hasta el momento con el referente asignado.

Luego de analizado los comentarios y aportes que realizaba el *Product Owner*, se registraba lo realizado y se comunicaba vía correo electrónico.

6. Arquitectura

Este capítulo refiere a como se planteó la arquitectura del sistema, explicando que aspectos se consideraron para facilitar el entendimiento del mismo.

Se describen los atributos de calidad considerados, las decisiones y fundamentaciones de diseño tomadas, así como también, las vistas arquitectónicas que apoyan dichas decisiones.

6.1 Atributos de calidad considerados

En la primera etapa del proyecto, la cual se centró en el análisis e investigación, se plantearon cuáles eran los atributos de calidad donde se pondría especial atención. Estos se desprenden por la tecnología a utilizar, las necesidades del cliente, características de los usuarios futuros y el tipo de aplicación a desarrollar.

Los atributos de calidad que se consideraron principales fueron:
RNF 1: Usabilidad
RNF 2: Modificabilidad
RNF 3: Testeabilidad

Adicionalmente se definió que se debían cuidar otros atributos de calidad, Seguridad, Portabilidad y Eficiencia. Estos fueron considerados como atributos secundarios, ya que, si bien no se iba a poner especial foco en ellos, se entendió que era correcto tenerlos en cuenta y además beneficiaría al cliente en caso de que en un futuro desee aumentar su atención. Por ejemplo, si el cliente obtiene más datos y necesita optimizar tiempos.

A continuación se detallan las tácticas y acciones que se tuvieron en cuenta para los atributos de calidad de la aplicación a desarrollar.

6.1.1 Atributos de calidad principales

En esta sección se detallan las tácticas realizadas para cada atributo de calidad, de forma de ejecutarlos del modo más adecuado para el equipo.

RNF 1 – Usabilidad

Para este atributo de calidad se consideró que era en el que se tenía que poner mayor esmero y se debía centrar la atención.

Dado que la aplicación es una herramienta que combina una gran variedad de filtros como *inputs*, y un conjunto de gráficas y tablas como *outputs*, se hizo sumamente importante llevar a cabo un correcto desarrollo y que el modo de utilización fuese práctico e intuitivo. De otra forma, sería muy tedioso y cansador realizar análisis en una herramienta como esta.

Esto en principio denotó cierta complejidad, por lo tanto, se plantearon mecanismos para determinar si la interfaz de usuario era la correcta:

- Validaciones de cantidad clics por funcionalidad

Al término de cada *Sprint* se controlaba que la cantidad de clics por funcionalidad principal desarrollada fuese menor a 3, a excepción de la selección de filtros que debía ser 2 clics por filtro seleccionado.

- Validación mediante test A/B:

Se realizó un test A/B con el fin de determinar cómo iban a estar dispuestos los distintos filtros del panel de búsquedas, ya que había dos posibilidades y no se definió formalmente por uno.

Esto era un factor muy importante, ya que, el usuario debe enfrentarse con este panel en dos funcionalidades críticas del sistema como lo son, Consultas y Comparativas. Para más detalles, ver los resultados del test A/B en el capítulo Aseguramiento de la calidad.

- Documentación

Dentro de este atributo de calidad, se tuvo en cuenta cómo hacer que el sistema sea intuitivo y fácil de utilizar, siguiendo como línea base de trabajo las heurísticas planteadas por Nielsen y Norman [4].

A continuación se detallan qué tácticas y lineamientos fueron seguidos para satisfacer este atributo de calidad:

- Información del estado del sistema: esta es visible a lo largo de las actividades que se ejecuten en la aplicación. El usuario es acompañado con mensajes que le informan acerca del estado actual. Por ejemplo, en el momento de guardar un favorito, si ingresó información inválida, entre otros avisos que se despliegan. Se buscó mantener al usuario informado para evitar que se generen incertidumbres.

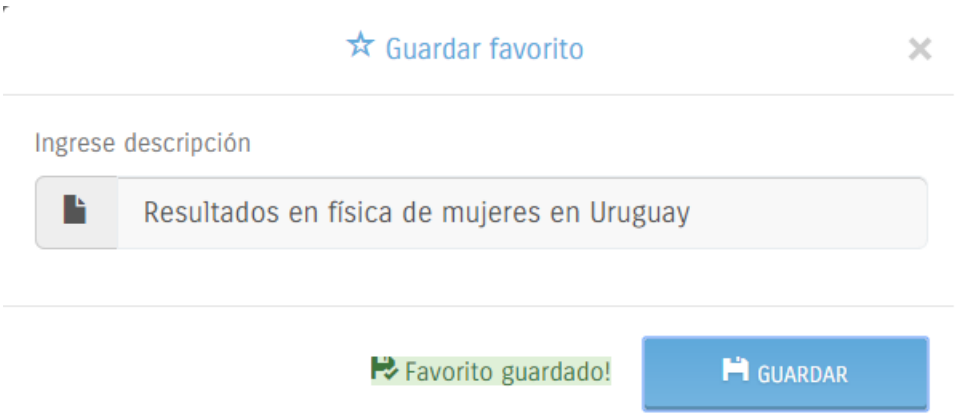


Ilustración 6-1 Favorito guardado

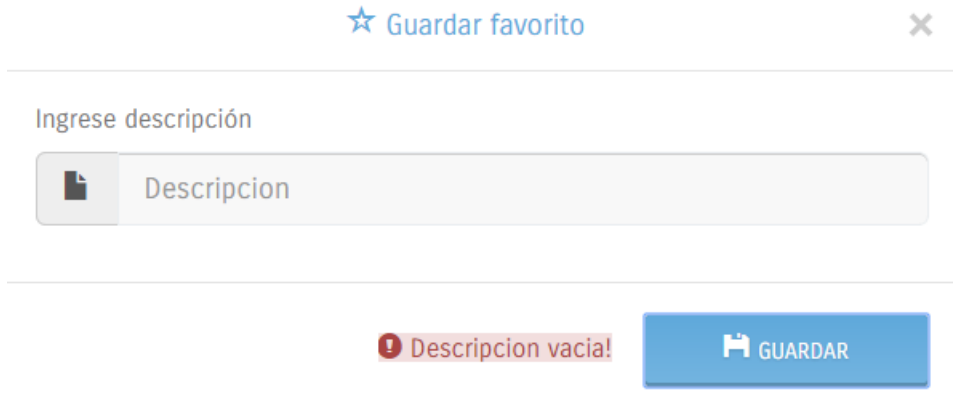


Ilustración 6-2 Campos inválidos

- Coincidencia entre el sistema y el mundo real**:** el sistema fue implementado con lenguaje utilizado por el negocio. Se incluyó terminologías propias del cliente, conceptos particulares y frases utilizadas por los usuarios durante las entrevistas de relevamiento, que fueron replicadas dentro del sistema, de forma de que sea más intuitivo.

Ilustración 6-3 Ejemplo de terminología

A su vez, se buscó acompañar las frases o conceptos con iconos que reflejen lo escrito en lenguaje coloquial. De esta forma, se evita la monotonía de textos y permite al usuario identificar la funcionalidad por otra vía de comunicación.

Ilustración 6-4 Íconos

- Control y libertad para el usuario: refiere a las circunstancias en que los usuarios por error eligen funcionalidades y desean salir o cancelarlas, sin afectar nada. Si bien no se cuenta con tantas funcionalidades que necesiten de lo anteriormente descripto, se entendió que era importante brindarle esa “salida de emergencia” a las pocas que si lo necesitaban. Se muestra una aplicación de lo mencionado en la funcionalidad de “Guardar Favorito”.

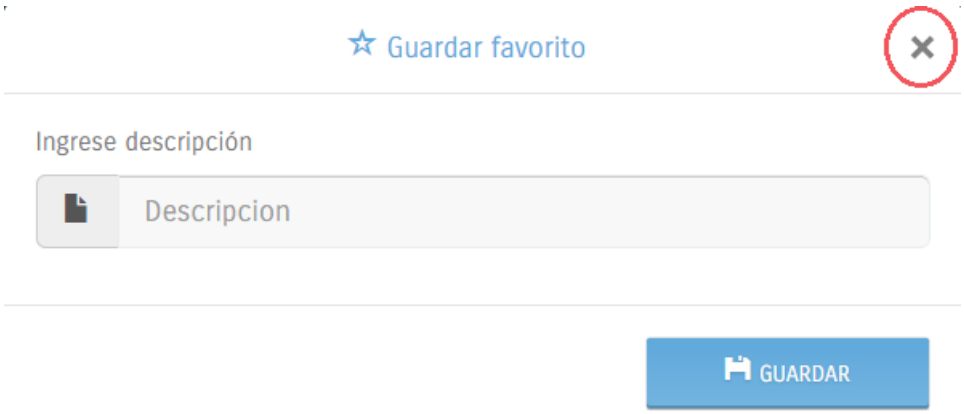


Ilustración 6-5 Posibilidad de cerrar la ventana arriba a la derecha

- Prevención de errores: aunque no se posee procesos críticos en los que se necesite especial foco respecto al cuidado de errores, se implementó esta heurística en aquellas funcionalidades donde un error o desatención del usuario podía causar pérdida de datos. En el ejemplo se muestra en donde se aplica cuando se va a borrar un favorito.

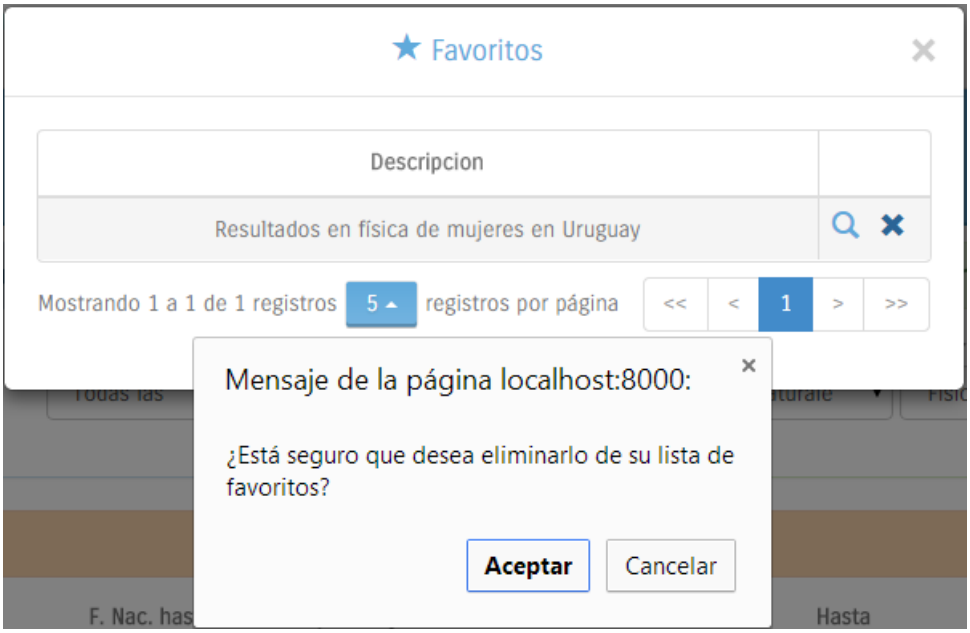


Ilustración 6-6 Mensaje de confirmación

- Ayuda para reconocer, determinar y recuperarse de errores: esta heurística se sigue mediante páginas de error personalizadas, por ejemplo para el error 404, en el que se explica al usuario la situación de forma clara y sencilla, evitando terminologías o códigos extraños.

Ilustración 6-7 Página de error personalizada

- Ayuda y documentación: se implementó una guía de usuario, con ayudas en secciones concretas, lo cual permite erradicar al usuario de dudas en un modo confiable ya que la información proviene de la fuente de desarrollo.

Ilustración 6-8 Ayuda

La primera vez que el usuario ingresa en la aplicación, se ofrece una visita guiada para realizar un recorrido tutelado en el cual se muestra en pocos pasos como utilizarla. De todos modos, se implementó la posibilidad de realizar este recorrido guiado en cualquier momento que se desee.

- Reconocimiento antes que memorización: los atajos siempre son una opción cómoda para los usuarios, es por eso que es conveniente reconocer las acciones que el usuario ha tomado con anterioridad para facilitar su repetición. En la aplicación se implementaron dichos atajos, como en el caso de la funcionalidad de “Consultas frecuentes”, “Alertas”, “Historial de alertas” y “Favoritos”.

Ilustración 6-9 Consultas rápidas

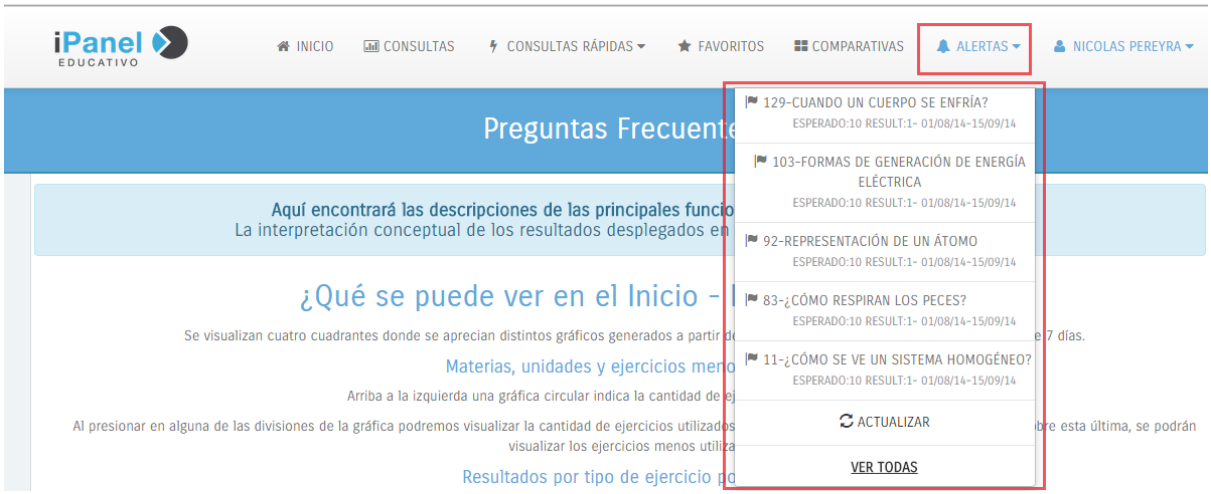


Ilustración 6-12 Alertas

- Manual de usuario: elaboración de un manual de usuario con ejemplos ilustrativos de las distintas funcionalidades que cuenta el sistema. Ver Anexo 3 – Manual de Usuario.

RNF 2 – Modificabilidad

Este atributo de calidad tiene que ver con el costo del cambio y la facilidad con que un sistema se adapta a las modificaciones, teniendo en cuenta sus responsabilidades. Para asegurarlo, se diseñó una arquitectura de capas lógicas, la cual agrupa las responsabilidades a un nivel de abstracción que permite reutilizarlas luego en donde se desee.

Además, se implementó el patrón MVC, de forma de que todos los cambios en la interfaz de usuario no impacten sobre la lógica de la aplicación.

La aplicación cuenta con archivos de configuración para base de datos, archivos estáticos y de ruteo, los cuales permiten cambiar configuraciones o constantes definidas, sin la necesidad de tener que instalar o cambiar componentes.

RNF 3 - Testeabilidad

Respecto a este atributo de calidad, se consideró que el sistema pudiera ser testeado fácilmente para exponer sus defectos. Se implementaron las medidas descriptas a continuación para controlarlo.

Se realizaron casos de prueba funcionales para todos los requerimientos del sistema.

También se realizaron pruebas unitarias para las funcionalidades críticas, Consultas y Comparaciones (Ver Anexo 2 – Listado de *User Stories* - *User Stories* 257 y 259 respectivamente), utilizando las herramientas que provee el *framework Django*[1].XXX

A continuación se muestra un ejemplo del resultado de los Unit Test que se obtuvo de la salida por consola.

```
done.                                  done.
        Consultas                              Compratívas
----------------------------------- -----------------------------------
Ran 25 tests in 16.734s                Ran 74 tests in 10.500s

OK                                     OK
```

Ilustración 6-10 Ejemplo de ejecución de *Unit Test*

1https://docs.djangoproject.com/en/dev/topics/*testing*/

Ambas tácticas, permitieron detectar errores o defectos y mejorar el sistema antes de su entrega.

6.1.2 Atributos de calidad secundarios

En esta sección se detallan las tácticas que el equipo llevó a cabo con respecto a los atributos de calidad secundarios.

RNF4 - Seguridad

Para poder cumplir con la seguridad adecuada de un sistema web como este, se desarrolló teniendo en cuenta acciones tales como:

- Autenticación: el sistema cuenta con un mecanismo de ingreso propio. Por lo tanto, es necesario ingresar con una cuenta y contraseña válida para poder acceder al sistema.

Ilustración 6-11 Acceso al panel

- Claves cifradas: mediante el *framework Django*, se mantienen las claves y contraseña de forma cifrada, lo que genera una seguridad adicional sobre los datos.

Protección CSRF: CSRF[2] (Cross-site-request-forgery) es un método por el cual un usuario malintencionado intenta hacer que otros, sin saberlo, envíen datos o peticiones que no desean enviar. Mediante *Django framework* se maneja y se protege el sistema de estos ataques malintencionados.

RNF 5 - Performance

Para poder abastecer de forma secundaria la performance, se tuvo en cuenta lo siguiente:

- Peticiones *Ajax*: las peticiones que se producen en el *frontend* hacia el *backend*, se realizaron de forma asincrónica aplicando la tecnología *Ajax*[3]. De esta forma, se optimizan los tiempos de peticiones, la aplicación se maneja en la misma vista (sin necesidad de recargar las páginas a menudo) y se opta por la fluidez.

- Índices en base de datos: se crearon índices en las tablas de la base de datos, se optimizó aquellas tablas que se suponían iban a ser las más consultadas para mejorar el tiempo de respuesta de las mismas. Esto también optimizó el sistema desde las capas más bajas de la arquitectura.

RNF 6 – Portabilidad

Para satisfacer con el atributo de Portabilidad se consideró que la tecnología utilizada fuera adecuada para la adaptabilidad del sistema.

Es por eso que se consideró en que ambientes de trabajo funcionó y donde se iba a instalar posteriormente.

Los sistemas operativos compatibles con la aplicación son *Windows* y *Linux*. Los navegadores utilizados para acceder a la aplicación pueden ser *Internet Explorer* 10+, *Mozilla Firefox* 27+ o *Google Chrome* 35+.

La portabilidad en los navegadores se hizo frente usando estándares de CSS. Estos se basan en implementar los selectores mediante técnicas y prefijos específicos.

2 http://es.wikipedia.org/wiki/Cross_Site_Request_Forgery

3 http://es.wikipedia.org/wiki/AJAX

Ilustración 6-12 Ejemplo del uso de los prefijos

Además para fortalecer este atributo se utilizó *Boostrap*, lo que permitió que el sistema fuera visto en monitores de escritorio así como también en dispositivos móviles *Tablets* con resoluciones de 1280x800px con facilidad. Ver Anexo 4 – Capturas de pantalla en distintas resoluciones de dispositivos móviles.

6.2 Decisiones y fundamentación de la arquitectura

El diseño de la arquitectura se vio muy condicionado por los atributos de calidad considerados, la tecnología de desarrollo y el tipo de aplicación. A continuación se especifican los estilos y patrones arquitectónicos que se utilizaron para definir estas decisiones.

6.2.1 Aplicación Cliente-Servidor

El tipo de aplicación, genera que haya dos partes involucradas. Un cliente (navegador de Internet) que es el que envía peticiones y recibe respuestas por parte del servidor; y una segunda que es el servidor, que es quien recibe las peticiones, las procesa y las envía nuevamente al cliente.

6.2.2 Patrón MVC

Para el desarrollo se utilizó el patrón MVC (*Model View Controller*), aplicando el *framework Django*. Esto permitió separar la lógica de acceso a datos y de negocio (*Controllers* y *Models*), con la presentación (*Views*). El marco de trabajo *Django* llama a los *Views* como *Templates*, pero mantiene toda la lógica que estas conllevan. De esta forma, los cambios en la interfaz de usuario no corrompen o impactan sobre las funcionalidades (*backend*) [5] [6].

6.2.3 Separación en capas lógicas

La aplicación fue construida separando las responsabilidades y agrupándolas a un nivel suficiente de abstracción, tal que, permita reutilizarlas en cualquier momento [5]. Las capas implementadas son las siguientes:

- Capa de Presentación: esta capa comprende la presentación del sistema, es decir, todas las páginas que contiene el sitio web.

- Capa de Lógica de Negocio: esta capa es la encargada de realizar los procesamientos de datos y peticiones, comunicándose y enviando datos a la capa de acceso a datos.

- Capa de Acceso a Datos: ofrece las funciones de acceso a los datos y el manejo de los mismos. Brinda a las capas superiores (anteriores) la capacidad de desentenderse de qué forma y como se obtienen y almacenan los datos.

6.3 Vistas arquitectónicas

En esta sección se presenta la arquitectura del sistema aplicando el modelo 4+1 planteado por Philippe Krutchen [7]. El modelo fue planteado parcialmente, generándose las vistas necesarias para mostrar cómo está formado el sistema, su comportamiento y como se despliega. Por lo tanto, estas vistas son un complemento para apoyar las decisiones y fundamentaciones de arquitecturas planteadas, con el fin de facilitar el entendimiento del sistema.

6.3.1 Vista de Lógica

En esta vista se presenta el diagrama de clases del dominio y un diagrama de secuencia referida a una de las funcionalidades críticas, Comparar resultados.

Diagrama de clases del dominio

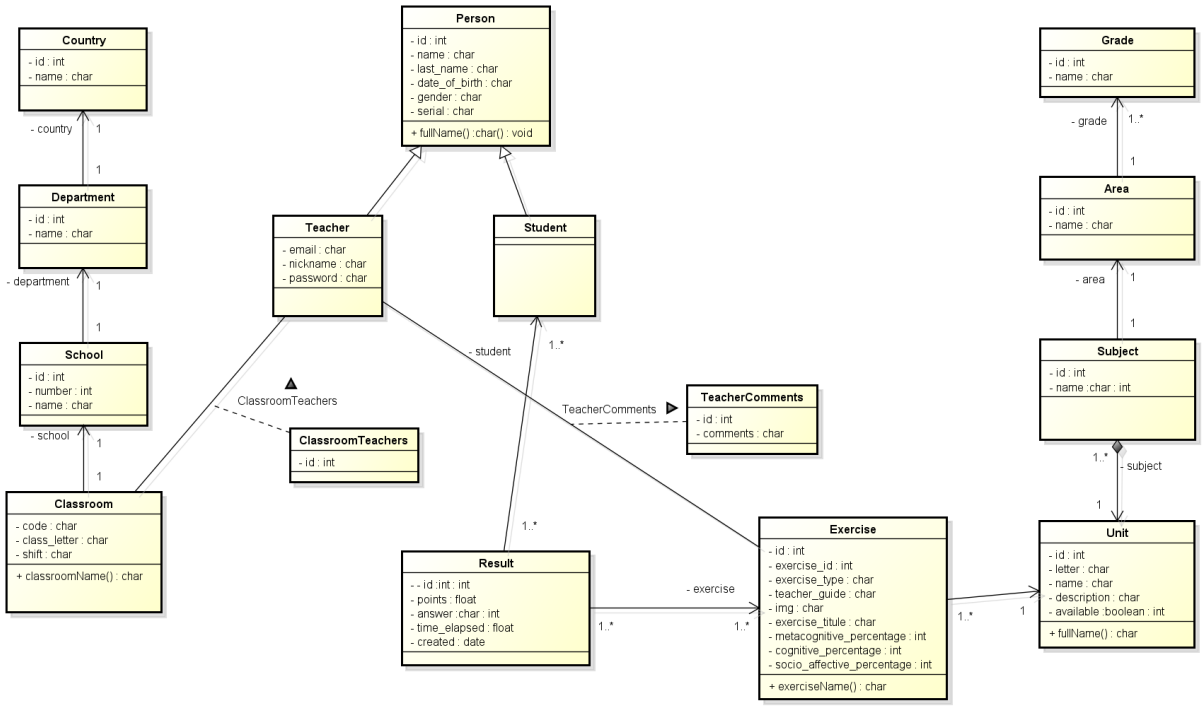


Ilustración 6-13 Diagrama de clases del dominio

Diagrama de secuencia

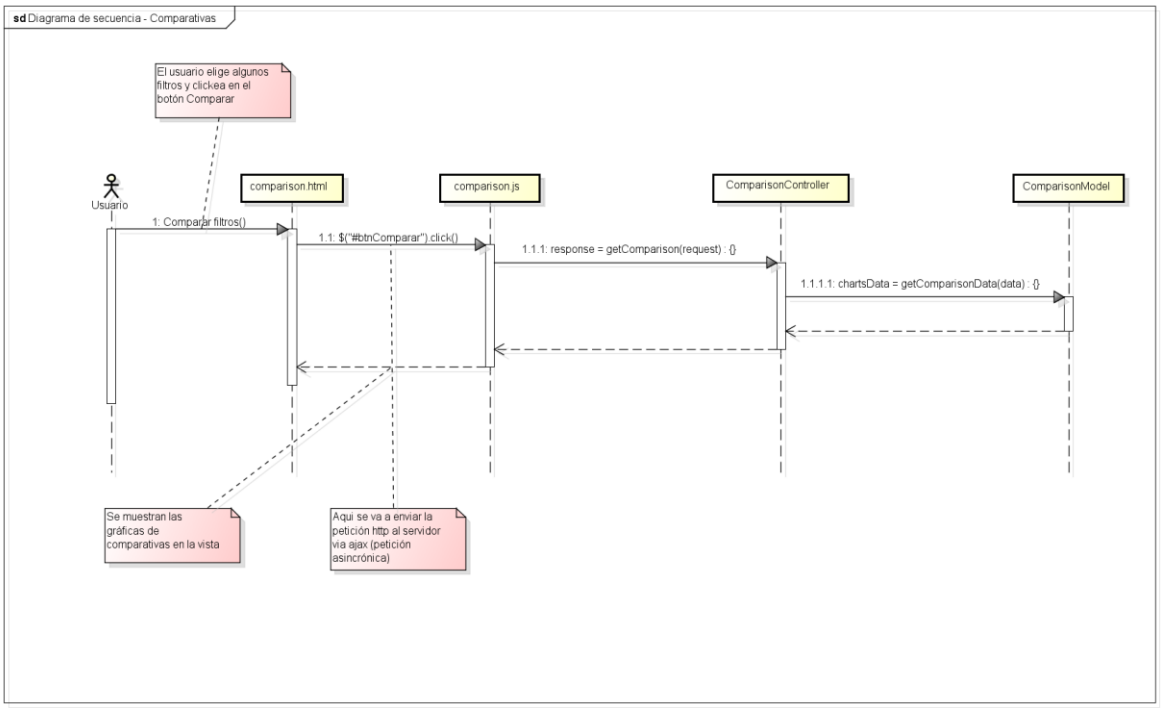


Ilustración 6-14Diagrama de secuencia

6.3.2 Vista de componentes

En esta vista se muestra como está compuesto el sistema en componentes y las dependencias que hay entre los mismos. Aquí se optó por realizar un diagrama de paquetes y un diagrama de componentes y conectores.

Diagrama de paquetes

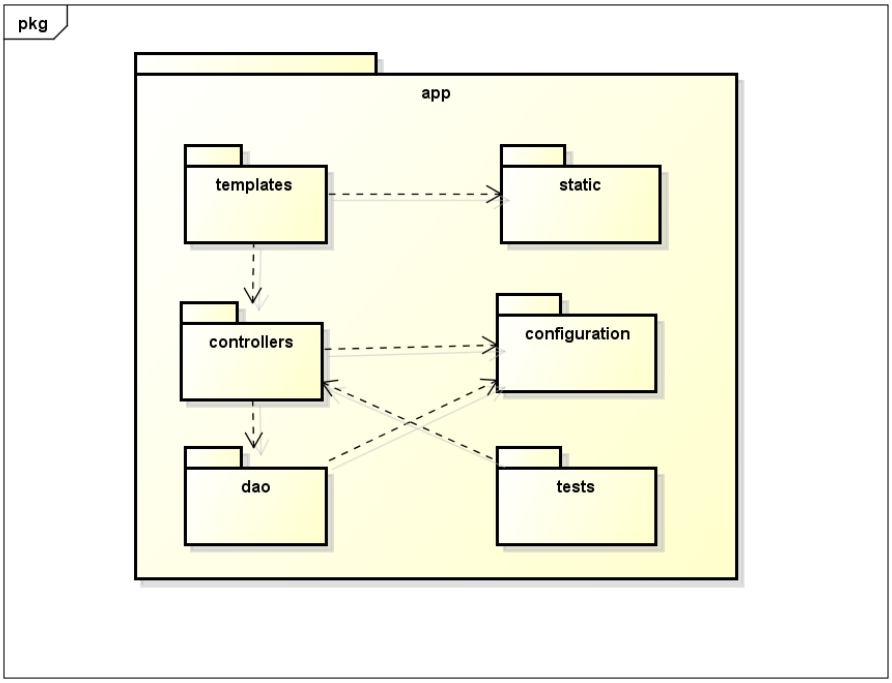


Ilustración 6-15 Diagrama de paquetes de la solución

Catálogo de elementos

Elemento	Descripción
app	El sistema
templates	Contiene todas las páginas web que forman la aplicación.
controllers	Contiene la lógica de negocio y de control de vistas.
dao	Posee la lógica de acceso a los datos.
static	Agrupa todos los archivos estáticos de la aplicación, tales como: imágenes, hojas de estilo, archivos *javascript* y fuentes.
configuration	Este paquete tiene los archivos de configuración del sistema.
tests	Agrupa las clases utilizadas para los test unitarios

Tabla 6-1Catálogo de elementos del Diagrama de paquetes

Diagrama de componentes y conectores

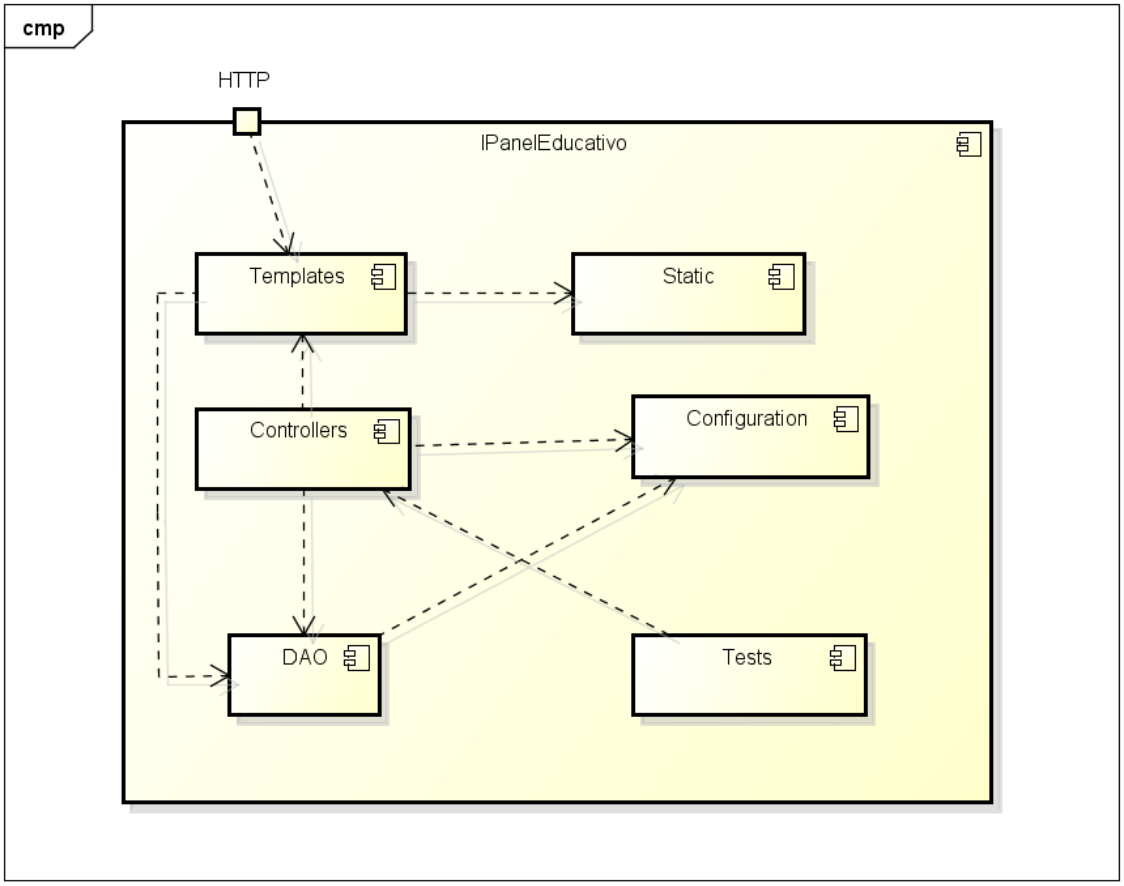


Ilustración 6-16Diagrama de componentes y conectores

Catálogo de elementos

Elemento	Descripción
iPanel Educativo	El sistema.
Templates	Vistas que se despliegan al cliente.
Controllers	Contiene la lógica de negocio y la capacidad de acceso a las vistas y al manejador de datos.
DAO	Lógica de acceso a los datos. Para esto se utiliza el ORM brindado por *Django Framework*.
Static	Agrupa todos los archivos estáticos de la aplicación, tales como: imágenes, hojas de estilo, archivos *javascript* y fuentes.
Configuration	Este paquete tiene los archivos de configuración del sistema. La misma se encuentra expresada como un conjunto de constantes significativas para el *framework* en uso.
Tests	Agrupa las clases utilizadas para los test unitarios.

Tabla 6-2 Catálogo de elementos del Diagrama de componentes y conectores

6.3.3 Vista física

Esta vista se muestra desde la perspectiva física, donde se representan los componentes físicos que conforman el sistema, así como las conexiones y servicios entre los mismos. Para poder evidenciar lo anterior se realizó un diagrama de despliegue.

Diagrama de despliegue

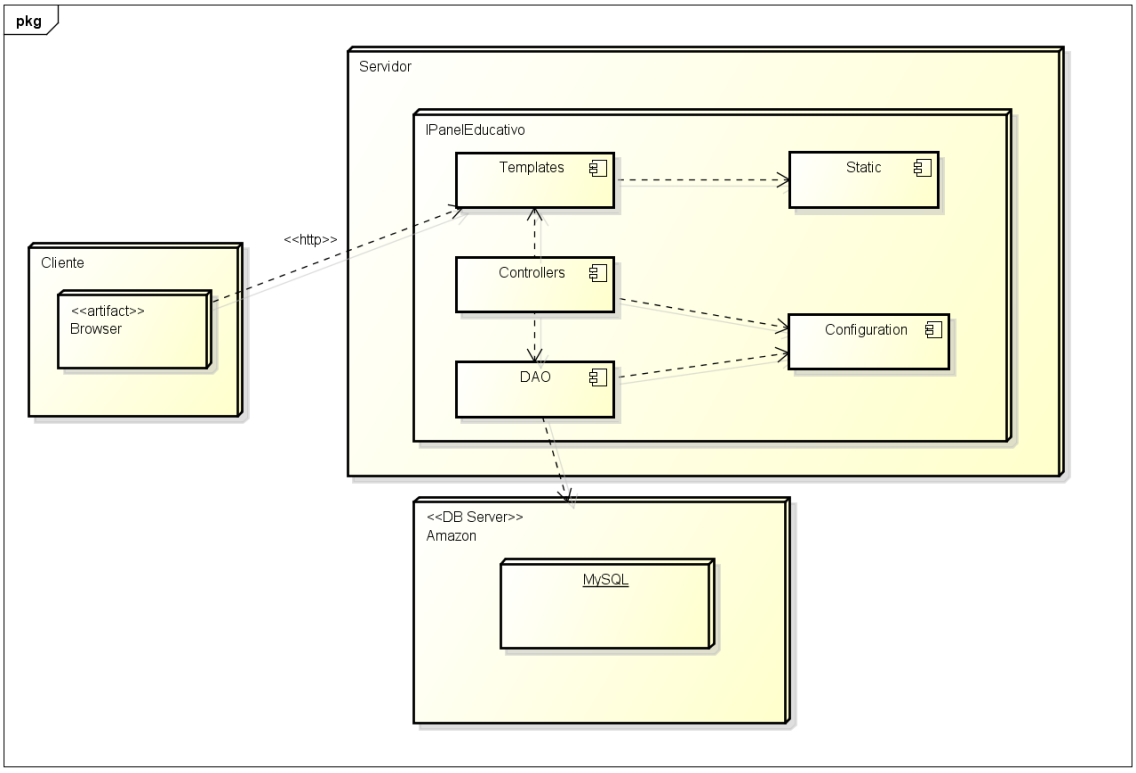


Ilustración 6-17 Diagrama de despliegue

Catálogo de elementos

Elemento	Descripción
Cliente	Equipo desde el cual el cliente va a acceder a la aplicación
Browser	Navegador que se usará para acceder a la aplicación. *Internet Explorer* 10+, *Mozilla Firefox* 27+ o *Google Chrome* 35+.
Servidor	Equipo en el cual el cliente alojará la aplicación.
iPanel Educativo	Aplicación iPanel Educativo
Templates	Vistas que se despliegan al cliente.
Controllers	Contiene la lógica de negocio y la capacidad de acceso a las vistas y al manejador de datos.

DAO	Lógica de acceso a los datos. Para esto se utiliza el ORM brindado por *Django Framework*.
Static	Agrupa todos los archivos estáticos de la aplicación, tales como: imágenes, hojas de estilo, archivos *javascript* y fuentes.
Configuration	Este paquete tiene los archivos de configuración del sistema. La misma se encuentra expresada como un conjunto de constantes significativas para el *framework* en uso.
Amazon	Repositorio en la nube para almacenar en este caso la base de datos.
MySQL	Motor de base de datos.

Tabla 6-3Catálogo de elementos del Diagrama de despliegue

6.4 Justificación de la tecnología seleccionada

En este capítulo se recopilan las investigaciones realizadas al inicio del proyecto, de las cuales, se desprendieron decisiones y elecciones para la tecnología de desarrollo a usar, la herramienta de versionado, entre otras.

6.4.1 Tecnología de desarrollo

La tecnología base de desarrollo no fue seleccionada por el equipo, sino que, el cliente fue el que planteó que la aplicación debía ser creada en *Python*. Dado que, no se contaba con experiencia laboral o académica en dicho lenguaje, se entendió que utilizar un *framework* podía ayudar en la creación de una aplicación más compleja con menos esfuerzo, comparado a hacerla de forma nativa.

Por lo tanto, se investigaron marcos de trabajo con requisitos necesarios para poder aplicar la arquitectura planteada y las funcionalidades deseadas. Se definió *Django* como *framework* porque cuenta con una serie de buenas prácticas ya establecidas y era el sugerido por el cliente. Para más detalles del proceso de investigación, los requisitos de selección, ventajas y desventajas de los *framework*s y decisión final, Ver Anexo 5 – Evaluación de tecnología de desarrollo.

6.4.2 Herramientas de desarrollo

El desarrollo se llevó a cabo con *Aptana Studio 3*[4]. A su vez, se utilizó de forma complementaria el editor de texto *Sublime Text 2*[5].

La base de datos utilizada es *MySQL* 5.6.16. y su administración se realizó con *MySQL Workbench*[6].

4 http://www.aptana.com/products/studio3

5 http://www.sublimetext.com/2

Estas herramientas fueron seleccionadas dado que se contaba con experiencia laboral de casi todo el equipo de proyecto en las mismas.

6 http://www.*MySQL*.com/products/*Workbench*/

7. Aseguramiento de la calidad

7.1 Introducción

El aseguramiento de la calidad se basa en realizar un conjunto de actividades definidas para garantizar la calidad no sólo en el proceso del ciclo de vida del proyecto, sino también en la satisfacción de las necesidades y expectativas del cliente con el producto.

En esta sección se describen los lineamientos, estándares, métricas y pruebas que fueron realizadas para asegurar la calidad en el proyecto; y los resultados obtenidos.

7.2 Objetivos

A continuación se describen los objetivos que fueron definidos en base al producto y al proceso de desarrollo.

Producto

- Lograr la satisfacción del cliente a medida que se va construyendo el producto y al finalizar el proyecto.
- Construir un producto con un porcentaje bajo de defectos.

Proceso

- Cumplir con los estándares definidos, aplicando los controles necesarios que permitan validar los resultados obtenidos en la construcción del producto de acuerdo a las necesidades del cliente.
- Establecer un ciclo de desarrollo acorde al proyecto y las características del equipo, con el fin de asegurar que las tareas se cumplan en tiempo y forma o que las mismas contengan una mejora continua observable capaz de detectar defectos fácilmente para su corrección a futuro.

7.3 Actividades de SQA

Para cumplir con el nivel de calidad esperado y satisfacer las necesidades del cliente se definió una serie de actividades a realizar sobre el producto y el proceso de software, para cada fase de ingeniería.

Fase	Actividad	Producto Resultado	Producto/s Consumidos	Rol Responsable	Roles participantes
Investigación	Capacitación en herramientas tecnológicas	Información sobre tecnologías específicas	Curso de capacitación sobre tecnologías específicas	Arquitecto de Software	Equipo de proyecto
Ingeniería de Requerimientos	Análisis y relevamiento de requerimientos	*Product Backlog*	Entrevistas con el cliente	Ingeniero de requerimientos.	*Product Owner*, Equipo de proyecto
	Validación *Product Backlog*	*Product Backlog* Validado	*Product Backlog*	Ingeniero de requerimientos.	*Product Owner*
Diseño	Diseño de la arquitectura	Vistas de la arquitectura	*Product Backlog*	Arquitecto de Software	Equipo de proyecto
	Revisión de Arquitectura	Vistas de la arquitectura revisadas	Vistas de la arquitectura	Arquitecto de Software	Arquitecto de Software, Tutor Proyecto.
Construcción	Planificación del desarrollo del producto	*Sprint Backlog*	*Product Backlog Planning Meeting*	Equipo de desarrollo	Equipo de desarrollo
	Desarrollo de las funcionalidades	*Releases* intermedias	*Sprint* backlog	Equipo de desarrollo	Equipo de desarrollo
	Cierre de la construcción del producto	*Release* final	*Product Backlog*	Equipo de desarrollo	Equipo de desarrollo
Testing	Diseño de casos de prueba	Planilla de casos de prueba	*Product Backlog*	SQA	Equipo de Testing
	Ejecución de casos de pruebas	Planilla de bugs encontrados	Planilla de casos de prueba	Equipo de Testing	Equipo de Testing
	Cierre de Testing	Planilla de métricas de proyecto	Planilla de casos de prueba y planilla de bugs encontrados	SQA	Equipo de Testing

Procesos de Gestión	Planificación de actividades de gestión	Cronograma de actividades	Reunión del equipo de proyecto	Gerente de proyecto	Equipo de proyecto
	Gestión de Riesgos	Sección de Riesgos en Documento final de proyecto	*Product Backlog*, entrevistas con el cliente	Gerente de proyecto	Equipo de proyecto
	Gestión de SCM	Sección de SCM en Documento final de proyecto	Análisis de diferentes herramientas para la configuración de software	SCM	Equipo de proyecto
	Gestión de SQA	Sección de SQA en Documento final de proyecto	Análisis de diferentes herramientas para el control de la calidad	SQA	Equipo de proyecto

Tabla 7-1Tabla de actividades de comunicación

Estas actividades se basaron principalmente en la selección y verificación de estándares, definición y recolección de métricas. A su vez, las actividades de prevención, evaluación y control del proceso de desarrollo se realizaron desde el inicio y a lo largo de todo el proyecto.

7.3.1 Estándares

Estándares de diseño de software

Se utilizó el entorno de desarrollo integrado *Aptana Studio*, con la utilización para *frontend* de *html5*, *css3*, *javascript*, *jquery*; y en *backend* se utilizó *Python* con la integración del *framework Django*, siguiendo la línea de lo que es el patrón MVC y para el manejo de la interacción de objetos se utilizó el patrón ORM provisto también por *Django*.

Estándares de codificación

Para la codificación de la aplicación web se decidió seguir las convenciones oficiales brindadas por el *framework* de *Django* para programar en *Python.* [8]

Los más importantes utilizados fueron:

- Instrucción try-catch para controlar la mayor parte de las excepciones.

- Nomenclatura *CamelCase* para nombrar a las variables, funciones y clases.

- Nombres nemotécnicos para todas las variables, métodos y clases.

Se agregó para cada método, variable o clase relevante en el sistema un comentario acorde que demuestre su fundamento de creación.

El idioma utilizado fue inglés, a pedido del cliente.

Estándares de control de versiones

Los estándares de control de versiones utilizados se agruparon de la siguiente manera:

- Documentación: se utilizó *Google Drive* para el manejo de modificaciones y la concurrencia en los documentos.

- Código Fuente: se utilizó *Bitbucket* como repositorio con *SourceTree* integrado con *Git* para manejar las versiones de la solución.

Los detalles sobre estos estándares definidos se encuentran en el capítulo Gestión de la Configuración.

Estándares de documentación

Para el desarrollo de la documentación se siguieron los lineamientos provistos por la Universidad en lo que respecta al siguiente listado de estándares:

- Doc. 302 - FI Normas específicas para la presentación de trabajos finales de Carrera de la Facultad de Ingeniería.

- Doc. 303 - Hoja de verificación de pautas de presentación de trabajos finales de carreras de la Facultad de Ingeniería.

- Doc. 304 - Normas para el desarrollo de trabajos finales de carrera.

- Doc. 306 - Orientación para títulos, resúmenes o *abstracts* e informes de corrección de trabajos finales de carrera.

- Doc. 307 - Pautas generales de formato de trabajos finales.

Los documentos pueden encontrarse en el sitio de Bedelías FI en el portal Aulas de la Universidad ORT.

Estándares de usabilidad

Con el objetivo de que el sistema fuera intuitivo y fácil de utilizar, se utilizaron como base las heurísticas planteadas por Nielsen y Norman. [4]

A continuación se listan las mismas que fueron detalladas en el capítulo de Arquitectura del sistema:

- Visibilidad del estado del sistema
- Relación entre el sistema y el mundo real
- Control y libertad del usuario
- Prevención de errores
- Reconocimiento antes que memorización
- Flexibilidad y eficiencia de uso
- Reconocimiento, diagnóstico y recuperación de errores
- Ayuda y documentación

7.3.2 Actividades en el proceso de desarrollo

Con el propósito de garantizar la calidad en el proceso de desarrollo de software y así contribuir a la elaboración de un producto satisfactorio para el cliente es que se realizó una medición de costos de calidad para las diferentes actividades realizadas en el proyecto.

Los objetivos planteados de dicho control sobre el proceso de desarrollo se enfocaron en la obtención de métricas que permitieran evaluar las actividades realizadas en el proyecto y mejorarlas a tiempo en el caso de detectar fallas.

La unidad de medida de esfuerzo utilizada fue de horas / hombre, las cuáles fueron asignadas al finalizar cada *Sprint* para cada actividad correspondiente al proceso de construcción del software. Las mismas (Ver Anexo 6 – Actividades de medición de costos de calidad) se dedujeron de los diferentes roles que integran el equipo, y posteriormente las mismas fueron agrupadas en tres tipos de categorías diferentes:

1. Prevención
2. Evaluación
3. Fallas

El resultado obtenido de esta medición comparado con las horas totales trabajadas en el proyecto arrojó los siguientes resultados que se muestran en la tabla a continuación. El porcentaje deseable de costos de calidad en proyectos de software fue obtenido de un artículo de la docente de ORTsf Amalia Álvarez. [9]

A nivel general, los resultados del equipo para cada categoría especificada anteriormente se encontraron dentro de los rangos esperados.

	Esperado	Real
Prevención	8-25%	17%
Evaluación	3-13%	11%
Fallas	1-8%	7%

Tabla 7-2 Tabla de porcentaje de costos de calidad esperados vs reales

Se concluyó a través de dicho análisis que algunas tareas no agregaban valor productivo al proceso de desarrollo y otras sí resultaron más efectivas para el equipo y el proyecto en cuestión.

7.3.3 Validaciones

Las diferentes validaciones que se realizaron durante el proyecto fueron seguidas de cerca por el rol de SQA y llevadas a cabo con el *Product Owner*, recibiendo *feedback* sobre cada funcionalidad entregada en cada *Sprint*, mientras que para el resto de los procesos tanto de ingeniería de software como de gestión se fueron verificando y validando a través de reuniones con el tutor del proyecto y expertos de ORTsf.

A continuación se detallan los procesos validados más relevantes:

Producto a evaluar	Criterio/Técnica	Responsable	Registro de resultados	Cuándo se realiza
Arquitectura del Sistema	Revisión con el cliente. Consultas al tutor.	Arquitecto de Software	Aceptación formal por parte del cliente y el tutor del proyecto	Durante el proyecto
Validación del alcance y requerimientos	Revisión con el cliente. Primera revisión en ORTsf con Rafael Betancur	Equipo	Aceptación formal por parte del cliente. Informe primera revisión con ORTsf.	Inicio del proyecto

SQA	Reunión con experta de Calidad en ORTsf - Amalia Álvarez.	SQA	Lineamientos sugeridos especificados en capitulo Gestión de calidad.	Inicio del proyecto
	Primera revisión en ORTsf con Rafael Betancur.	SQA	Informe primera revisión con ORTsf.	Inicio del proyecto
Diseño Gráfico de sección Consultas seleccionado luego de Test A/B con grupo de usuarios.	Reunión con *Product Owner* en Ceremonia *Sprint* Review 4	Equipo	Aceptación formal mediante una entrevista con el *Product Owner* del proyecto	*Sprint* 4

Tabla 7-3Tabla de validaciones de procesos principales

7.3.4 Criterios de aceptación del producto

Desde el inicio del proyecto se buscó lograr la aceptación satisfactoria del cliente a mediano plazo para las funcionalidades desarrolladas durante el proyecto. Es por ello que se decidió obtener determinadas aprobaciones en cuanto a las funcionalidades entregadas al cierre de cada *Sprint* por parte del *Product Owner*, con el fin de mantener una tranquilidad y confianza por parte del equipo y para con el cliente en pos de contribuir al compromiso de ambas partes. Ver criterios de aceptación del producto en Anexo 2 – Listado de *User Stories*.

7.3.5 Auditoría repositorios

Se realizaron auditorías periódicas a los repositorios de la documentación y el código fuente del producto con el fin de evaluar la correcta utilización de los mismos. Dichas auditorías fueron realizadas por el responsable de calidad pero también cada integrante del equipo estuvo pendiente de la eficaz utilización y manejo de los repositorios del proyecto.

Las actividades realizadas se basaron en:

- Observar el correcto uso de ambos repositorios de documentación y código fuente.
- Verificar nombres de carpetas, archivos y versionados de documentación.
- Controlar adopción de estándares de nomenclatura en fuentes del sistema.
- Constatar cuál es la última versión de los documentos, fuentes, scripts de base de datos al momento de querer modificarlos.

7.3.6 Pruebas

El equipo entendió como importante la realización del Testing para conseguir los objetivos planteados al inicio del proyecto en cuanto a obtener un producto que satisfaga las necesidades del cliente, siendo intuitivo, fácil de usar y con la menor cantidad de bugs posibles, alineado así a los atributos de calidad como Testeabilidad y Usabilidad definidos junto con al cliente, como se mencionó en el capítulo de Arquitectura.

El testing estuvo a cargo de todo el equipo y la cantidad de horas destinadas para ello fue variable en cada iteración según la cantidad de *User Stories* que se desarrollaban en el *Sprint*.

Para alcanzar las metas planteadas anteriormente se definió aplicar las siguientes técnicas de testing: *units test* (pruebas internas), test funcional y test de usabilidad. La primera sirvió para determinar bugs a nivel de codificación internamente en determinadas funcionalidades de un nivel alto de complejidad; mientras que las pruebas manuales a nivel funcional permitieron encontrar errores del funcionamiento de todo el sistema en general y mejoras para tener en cuenta; y el test de usabilidad permitió identificar la correcta disposición del diseño gráfico del sistema para optimizar su funcionalidad.

Para cubrir el Testing de todas las *User Stories* implementadas, se confeccionó una planilla en *Google Drive* con un conjunto de casos de prueba, donde cada uno de estos se estructuró de la siguiente manera:

Prueba n°: Identificación de la prueba.
Funcionalidad a probar: Nombre de la funcionalidad.
Descripción: Detalle de cómo se probó la funcionalidad.
Resultado esperado: Cómo debió responder la funcionalidad / Cuál sería el resultado ideal.
Resultado obtenido: Cómo efectivamente respondió la funcionalidad a la prueba.

7.3.7 Registro de Bugs

La detección de Bugs así como todas las tareas de testing fueron realizados por parte de todo el equipo, cada integrante que realizaba una *User Story* antes de ser subida debía probar la misma, luego de esto se subía al repositorio de *Bitbucket* y otro miembro del equipo probaba dicha funcionalidad para garantizar la ausencia de bugs en la misma y la correcta integración con el resto de las funcionalidades del sistema.

Durante los *Sprint* 3 y 4 fueron las etapas donde se registraron mayores bugs debido a que se realizaron las funcionalidades críticas del sistema, por lo que la complejidad de las mismas fue mayor respecto al resto.

Para el registro de bugs se utilizó una de las prestaciones que provee la herramienta *TargetProcess*, que en la siguiente imagen se observa un ejemplo.

Ilustración 7-1Ejemplo registro de bugs en *TargetProcess*

Complementariamente también se gestionó una planilla de seguimiento de los mismos a nivel interno en el repositorio en *Google Drive* para una visualización más amplia de la totalidad de los bugs, en la cual se asignaba un responsable para su resolución y la prioridad del mismo.

El reporte de bugs se realizaba durante cada *Sprint* y eran corregidos en el *Sprint* actual o de lo contrario quedaban pendientes para los *Sprint*s siguientes dependiendo de la prioridad del mismo.

7.3.8 Unit test

Se definió realizar *Unit test* solamente para aquellas *User Stories* consideradas críticas para el sistema, acordando con el cliente que no aportaba valor realizar este tipo de test para funcionalidades que no requieren mayor complejidad.

Con dicha técnica para probar el correcto funcionamiento de un módulo de código, se aseguró que cada uno de estos funcionara correctamente de forma separada.

Las *User Stories* de la sección de consultas y comparativas de iPanel fueron las elegidas para aplicar el *unit test* bajo dos tareas: 1- QC - Creación *unit test*, 2- QC - Ejecución *unit test*. Para llevar a cabo esto se utilizó las herramientas provistas por el *framework Django*.

7.3.9 Test A/B

El Testing de usabilidad fue importante para determinar la facilidad e inconvenientes a los que se enfrentaría un usuario al utilizar el sistema.

Dicha prueba de usabilidad fue realizada para las *User Stories* de la sección de consultas y comparativas, ya que fueron las que generaron incertidumbre al cliente respecto al diseño y disposición de elementos web para así lograr optimizar su usabilidad.

Esta tarea no fue sencilla ya que el resultado dependía ampliamente de las características de los usuarios consultados y de su perspectiva. El perfil de dichos usuarios contactados fue de nivel intermedio en cuanto a conocimiento y manejo de herramientas ofimáticas y de tecnología en general.

Para dicho test se ocultó el icono del cliente para mantener la confidencialidad y el resguardo de su identidad ante los usuarios voluntarios que realizaron la prueba, los cuales contaban con nivel de experiencia nula en este tipo de herramientas. De 12 consultados, 10 eligieron la opción B que es la que hoy en día se encuentra en iPanel. Ver Anexo 7 - Test A/B.

7.3.10 Encuesta de satisfacción de usuarios

Se realizó una encuesta de satisfacción entre los usuarios del cliente, con el fin de validar los objetivos del producto definidos al inicio del proyecto y recabar sus comentarios y satisfacción respecto al sistema.

Esta encuesta evaluaban las siguientes características:

- Opinión global sobre el sistema.
- Usabilidad en la sección Consultas.
- Usabilidad en la sección Comparativas.
- Información y contenidos en inicio del Sistema.
- Diseño de resultados en gráficas.

Para cada ítem se pedía elegir entre las opciones predeterminadas, estas eran: Muy de acuerdo, de acuerdo, en desacuerdo y muy en desacuerdo. Ver Anexo 8 – Encuesta de satisfacción de usuario.

7.4 Métricas

En esta sección se detallan las métricas que fueron definidas y evaluadas para las distintas actividades a lo largo del proyecto. A partir de las mismas se logró observar el avance del proyecto en distintas áreas e identificar ciertas actividades que no se realizaban de la forma correcta o esperada.

Por un lado se tomaron métricas con el objetivo de asegurar la calidad del proceso de desarrollo de software en el transcurso del proyecto. Para registrarlas se decidió utilizar una herramienta web llamada *TargetProcess*.

A partir de la incorporación de dicha herramienta para la gestión de los proyectos de software de manera ágil, se pudo realizar un seguimiento continuo de los avances, manteniendo reuniones asiduas de validación y revisión en el equipo, adquiriendo un completo estado de las tareas en desarrollo y obteniendo continuo *feedback* del cliente debido a las entregas periódicas del producto al final de cada *Sprint*. Con este marco de trabajo se detectaron fácilmente las fallas y se logró la mejora continua en las siguientes iteraciones, corrigiendo y ajustando la métrica de retrabajo en el caso requerido.

Para la decisión de la elección de la herramienta *TargetProcess* se hizo principal hincapié en el contexto del proyecto de Software Factory con la utilización de *Scrum*, siendo este el principal punto que se tuvo en cuenta. Otra de las razones que ayudó a tomar la decisión fue su facilidad de uso y simpleza para comenzar a trabajar en ella. La modalidad de asignación de tareas y registro de esfuerzo en el programa es otro de sus puntos fuertes, ya que brinda la posibilidad de ir agregando a una tarea a más de un miembro del equipo e indicando sus horas estimadas y reales, las mismas también fueron comentadas y discutidas para aumentar la retroalimentación del equipo en función del cumplimiento de cada *User Story*.

Por otro lado, se tomaron métricas respecto al producto de software en sí mismo como la cantidad de bugs o errores encontrados, solucionados y pendientes en cada *Sprint*.

Para todas las métricas se registró su objetivo: ¿Qué se pretende lograr al registrar esa métrica? Y un valor de referencia: ¿Qué valor es esperado de la métrica al finalizar el *Sprint*?

Los resultados de las métricas consideradas fueron las siguientes:

Métrica: Esfuerzo estimado (horas/hombre)						
Objetivo: Finalizar todas las tareas definidas para cada *Sprint*						
Valor de referencia: Horas definidas para realizar las tareas						
Sprint 1	*Sprint* 2	*Sprint* 3	*Sprint* 4	*Sprint* 5	Promedio	Total
180	200	200	220	200	200	1000

Tabla 7-4 Tabla de esfuerzos estimados por sprint

Métrica: Esfuerzo real (horas/hombre)						
Objetivo: Cumplir con las tareas definidas de las *User Stories* para cada *Sprint*						
Valor de referencia: Esfuerzo estimado						
Sprint 1	*Sprint* 2	*Sprint* 3	*Sprint* 4	*Sprint* 5	Promedio	Total
224	214	215	220	214	217	1087

Tabla 7-5 Tabla de esfuerzos real por sprint

Métrica: Cantidad de tareas realizadas por *Sprint*						
Objetivo: Estimar correctamente las tareas de acuerdo a las horas estimadas para cada *Sprint*						
Valor de referencia: Tareas definidas en la iteración						
Sprint 1	*Sprint* 2	*Sprint* 3	*Sprint* 4	*Sprint* 5	Promedio	Total
18	24	30	36	32	28	140

Tabla 7-6 Tabla de tareas realizadas por sprint

Métrica: Cantidad de tareas pendientes por *Sprint*						
Objetivo: Evitar la acumulación de tareas pendientes de *Sprint*s finalizados						
Valor de referencia: No mayor a 3						
Sprint 1	*Sprint* 2	*Sprint* 3	*Sprint* 4	*Sprint* 5	Promedio	Total
0	2	4	2	0	1,6	8

Tabla 7-7 Tabla de tareas pendientes por sprint

Métrica: Porcentaje de cobertura de pruebas						
Objetivo: Probar todas las funcionalidades existentes						
Valor de referencia: 100%						
Sprint 1	*Sprint* 2	*Sprint* 3	*Sprint* 4	*Sprint* 5	Promedio	Total
100%	95%	95%	95%	95%	96%	96%

Tabla 7-8 Tabla de cobertura de pruebas por sprint

Métrica: Cantidad de bugs solucionados						
Objetivo: Corregir todos los bugs encontrados en cada *Sprint*						
Valor de referencia: 100% de los encontrados						
Sprint 1	*Sprint* 2	*Sprint* 3	*Sprint* 4	*Sprint* 5	Promedio	Total
8 (100%)	18 (78%)	28 (72%)	16 (67%)	15(100%)	17	85

Tabla 7-9 Tabla cantidad de bugs solucionados por sprint

Métrica: Cantidad de bugs pendientes						
Objetivo: Evitar la acumulación de bugs pendientes de solucionarse de *Sprint*s finalizados						
Valor de referencia: 0% de los encontrados						
Sprint 1	*Sprint* 2	*Sprint* 3	*Sprint* 4	*Sprint* 5	Promedio	Total
0 (0%)	5 (22%)	11 (28%)	8 (33%)	0 (0%)	4,8	24

Tabla 7-10 Tabla de bugs pendientes por sprint

7.4.1 Conclusión de las métricas

Las métricas permitieron el análisis objetivo del avance del proyecto en general, obteniendo datos sobre la evolución de equipo en el proceso de desarrollo y del producto en sí mismo.

Respecto a las horas trabajadas fueron mayores a las estimadas pero no constituyeron una diferencia significativa para el desarrollo del proyecto.
La cantidad de tareas pendientes de *Sprint* anteriores pudieron resolverse sin problemas en las siguientes iteraciones y no ocasionaron grandes desviaciones en los tiempos del proyecto.

En cuanto al proceso de testing, el escenario ideal era cubrir la totalidad de las funcionalidades del sistema pero dadas sus características, donde existen muchas casuísticas de las búsquedas y comparaciones de datos, sumado a los tiempos acotados resultó difícil conseguir un 100% de cobertura de pruebas en todos los *Sprint*s. De todos modos, el cliente quedó satisfecho al ver todas las pruebas realizadas y al contar con personal técnico capacitado para arreglar un eventual defecto.

La cantidad de bugs encontrados aumentó en consecuencia a las funcionalidades desarrolladas y su complejidad, lo que provocó que los tiempos estimados no alcanzaran para solucionar los mismos, sobre todo en los *Sprint* 2 al 4 que se disparó una alerta en el equipo para analizar la situación. Finalmente se decidió comprometer tiempo específico durante el *Sprint* 5 para cumplir con los pendientes.

8. Gestión de la Configuración del Software

8.1 Introducción

En este capítulo se describe cuál fue la estrategia definida a la hora de Gestionar la Configuración del Software, abordando temas referentes con las herramientas utilizadas para el desarrollo del producto de software y documentación durante el proyecto, elementos de configuración detectados, ciclo de control de cambios y tácticas de versionado en estos, configuración de ambientes de trabajo y respaldos.

Las tareas inherentes a dicha área de SCM fueron llevadas a cabo durante todo el ciclo de vida del proyecto y son consideradas actividades de protección, siendo controladas íntegramente por un sólo miembro del equipo pero contando con el apoyo y disposición constante de todo el equipo de proyecto.

Uno de los objetivos establecidos al comienzo del proyecto para el SCM fue definir un conjunto de reglas que iban a marcar un camino seguro y transparente a seguir referente al manejo de versionados, almacenamiento, control de cambios de los elementos de configuración, herramientas a utilizar para que los integrantes del equipo se encuentren alineados a un fin común y tengan en claro cómo trabajar de manera concurrente y sincronizada.

8.2 Herramientas y ambientes de trabajo

El ambiente de desarrollo fue instalado en cada una de las máquinas de los integrantes del equipo, para no depender de un espacio físico en especial y con la ventaja de lograr cierta autonomía para el equipo a la hora de realizar tareas principalmente de desarrollo del producto de software.

Herramientas utilizadas para el desarrollo del producto de Software:

- IDE: *AptanaStudio* 3
- Sublime Text
- *MySQLWorkbench*
- *SourceTree*
- GIT
- *Bitbucket*

Herramienta para gestionar la documentación del proyecto:

SaaS de *Google Drive* con toda la gama de servicios que provee en la manipulación de documentos a nivel online y desktop en su versión para Windows precisamente.

8.3 Identificación de elementos de configuración

Se identificaron dos tipos de Elementos de Configuración del Software (**ECS**) durante el proyecto, los mismos fueron:

Tipo	Elementos
ECS Software: Son todos los archivos que forman parte del producto de software, como ser código fuente, esquemas, respaldos de Base de Datos	● Componentes Web (*.html, .css,.scss,.less .js*) ● *Python* (*.py*) ● Imágenes (*.png)* ● *MySQL* (*.sql*)
ECS Documentación: Son los elementos de documentación realizados y mantenidos a lo largo de todas las iteraciones del proceso de desarrollo de software.	● Imágenes (.jpg, .jpeg, .png, .gif) ● Documentos web ● Diagramas (.uml) ● Documentos de texto (.txt, .pdf) ● Documento con texto enriquecido (.docx) ● Documento de planilla (.xls) ● Documento de presentación (.ppt, .pptx)

Tabla 8-1 Tabla de elementos de configuración

8.4 Definición del repositorio y tecnologías

El repositorio es un medio de almacenamiento instantáneo donde se van a manipular las diferentes versiones de los elementos de configuración de software a lo largo del proceso de desarrollo del producto y proyecto de software.

En el caso del producto de Software se utiliza *Bitbucket* como medio de almacenamiento para el código fuente y para la gestión de documentación utilizamos *Google Drive* como se mencionó anteriormente en las herramientas utilizadas.

8.5 Gestión de la documentación

Para lo que fue la designación de una herramienta encargada del control de todos los archivos inherentes a la documentación del proyecto, se intentó seleccionar una que no sólo posea una gama variada de servicios calificado en ofimática sino la posibilidad de que sea colaborativa, manipulable en tiempo real por más de una persona, accesible para que cada integrante pueda utilizarla en todo momento. Por ello se decidió utilizar *Google Drive*, en base a que cada integrante del equipo ya contaba con cuenta de *Gmail* y experiencia con la plataforma.

Cada integrante del equipo contaba con la versión desktop de *Drive*, por lo que accedía al repositorio de la nube en una carpeta local en sus computadoras. Cuando un integrante del equipo modificaba, creaba o eliminaba un archivo, el cambio impactaba de forma simultánea en todas las computadoras. La única restricción para ello fue que se debía estar conectado a internet para que el cambio fuera actualizado en el repositorio de manera instantánea.

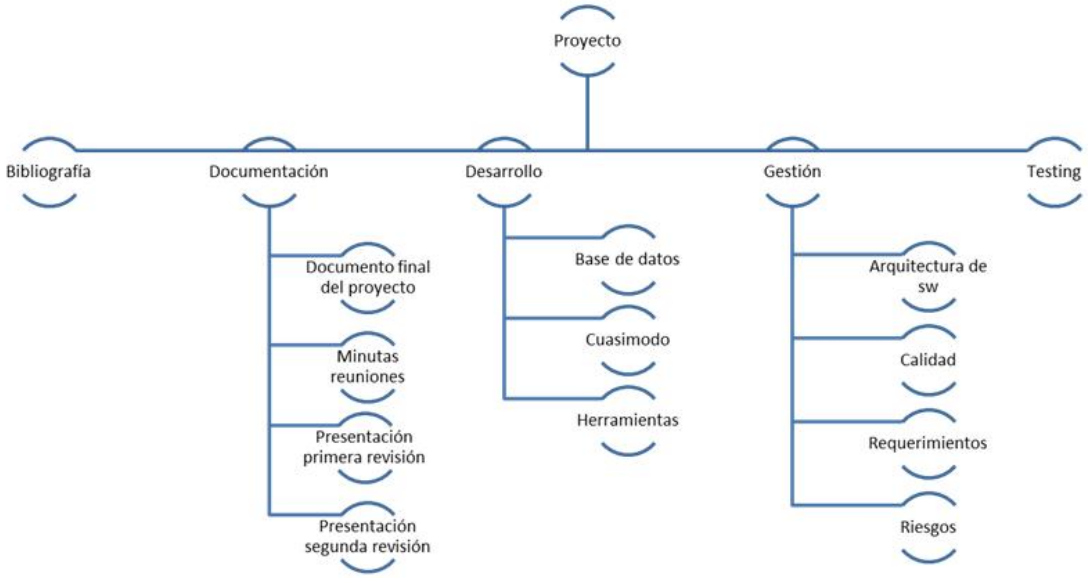


Ilustración 8-1 Estructura Repositorio de Documentación en Google Drive

8.6 Gestión de las Piezas de Software

Para la elección de un determinado repositorio que nos brinde seguridad y accesibilidad en todo momento para almacenar el código fuente se pusieron sobre la mesa dos tecnologías diferentes al inicio del proyecto *SVN* y *Bitbucket*.

Se comenzó por un análisis de la herramienta de *SVN*, la cual no era conocida por todo el equipo, lo que podría insumir mayores tiempos de adaptación y capacitación sobre la misma.

Por otro lado, *Bitbucket* se adaptaba mejor al equipo por ser conocida por todos los integrantes, siendo un servicio en la nube que ofrecía crear repositorios públicos y privados, fue la mejor opción con un almacenamiento gratis para un máximo de 5 usuarios, garantizaba mayor disponibilidad en todo momento comparado con un medio físico de almacenamiento. Otro factor tenido en cuenta fue que la herramienta ya se utilizaba en el cliente EDU Editorial, informado en las reuniones iniciales del proyecto por nuestro *Product Owner*.

Para el control distribuido de versiones y gestión del código fuente se optó por GIT integrado en *AptanaStudio* y *SourceTree* como cliente de *Git* para Windows.

Se utilizaron dichas herramientas para el manejo de conflictos por la agradable prestación de notificación para quienes intentan realizar un cambio. Otro criterio de elección fue que el IDE (*AptanaStudio*) elegido para el desarrollo del producto integra dentro de sus elementos de instalación a GIT. En el caso de *SourceTree* fue por el hecho de poseer una interfaz gráfica intuitiva y accesible fácilmente para los integrantes del equipo como se encuentra para Windows.

A través de *SourceTree* se realizaba la gestión de versiones del producto de software, el equipo manipulaba las diferentes modificaciones, sincronizaciones y unificaciones de archivos de código fuente de cada uno de los repositorios locales del equipo con el *branch* principal ubicado en *Bitbucket*.

Dicha herramienta nos permitía que dos o más personas estén trabajando sobre el mismo archivo, donde luego de aplicar los cambios pertinente, ésta realizaba una integración automática o bien se podía realizar manualmente por cada uno de los usuarios a través de la funcionalidad *DifMerge* integrada en *SourceTree*.

Al principio del proyecto este manejo de conflictos ocasionaba algunas tardanzas a la hora de comenzar a trabajar con una versión estable y consistente al resto del equipo por la inexperiencia en la herramienta de algunos de los integrantes del equipo. Luego progresivamente se fue logrando un avance y conocimiento total en la forma de trabajo distribuido para que cada integrante pudiera estar desarrollando en su máquina, sin tener que estar al tanto de las implementaciones del resto del equipo para evitar problemas de sincronización.

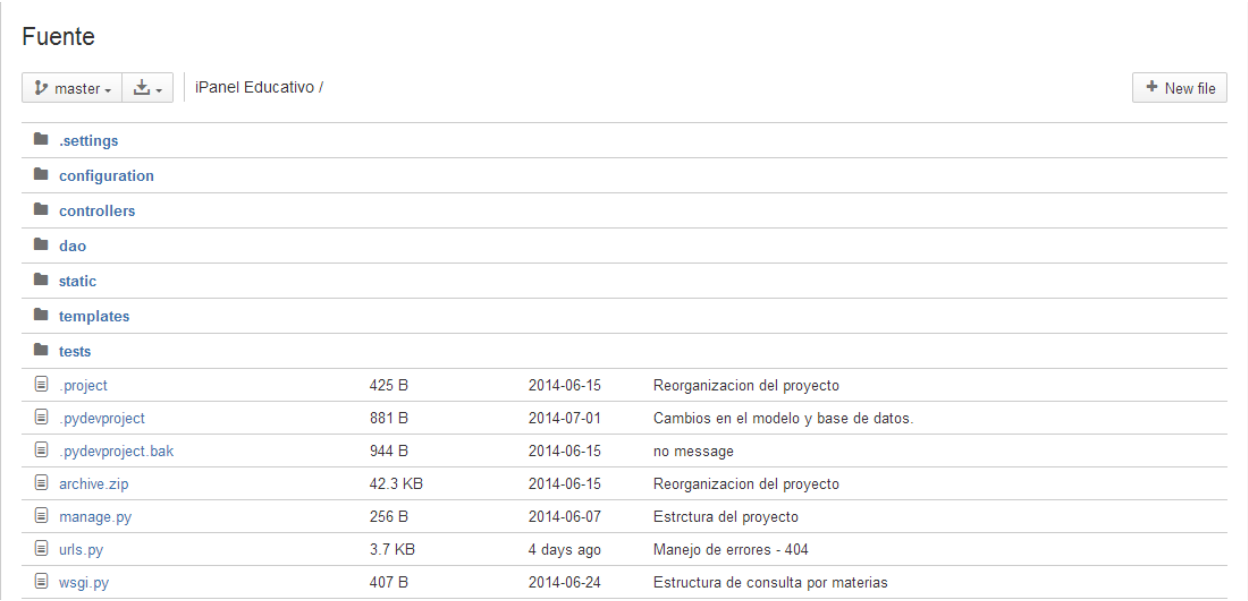


Ilustración 8-2 Estructura Repositorio de Código Fuente en Bitbucket

8.7 Gestión de versiones

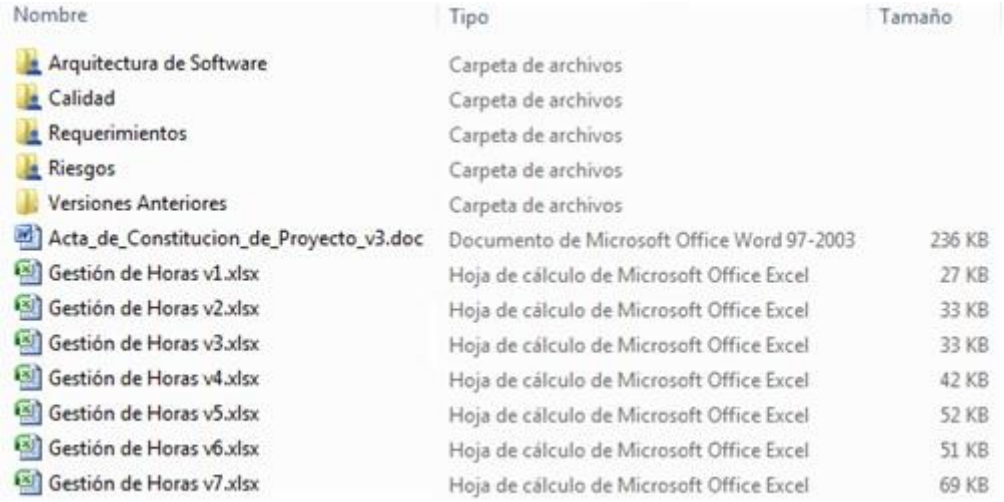
8.7.1 Versiones de Documentación

Cada integrante del equipo poseía permisos de escritura y lectura sobre la carpeta de almacenamiento de Documentación del proyecto creada en *Google Drive*.

Todos los documentos existentes en el repositorio poseen acompañado al nombre la letra “v” y un número de versión correlativo con el fin de llevar un histórico para todos los archivos y poder volver fácilmente a una versión funcional.

La nomenclatura designada fue: nombre Documento**v**X: siendo x el número correlativo de versión para cada archivo. Por ejemplo “matriz de riesgos v1.doc” es la primera versión del documento “matriz de riesgos.doc”.

Con esto se definió inicialmente que cada vez que se debía modificar un archivo existente se debería de bajar del repositorio de *Google Drive* o utilizar la carpeta instalada en cada máquina local y realizar las modificaciones pertinentes, renombrar el archivo (siguiendo las convenciones de nomenclatura) y volverlo a subir al repositorio online.



Nombre	Tipo	Tamaño
Arquitectura de Software	Carpeta de archivos	
Calidad	Carpeta de archivos	
Requerimientos	Carpeta de archivos	
Riesgos	Carpeta de archivos	
Versiones Anteriores	Carpeta de archivos	
Acta_de_Constitucion_de_Proyecto_v3.doc	Documento de Microsoft Office Word 97-2003	236 KB
Gestión de Horas v1.xlsx	Hoja de cálculo de Microsoft Office Excel	27 KB
Gestión de Horas v2.xlsx	Hoja de cálculo de Microsoft Office Excel	33 KB
Gestión de Horas v3.xlsx	Hoja de cálculo de Microsoft Office Excel	33 KB
Gestión de Horas v4.xlsx	Hoja de cálculo de Microsoft Office Excel	42 KB
Gestión de Horas v5.xlsx	Hoja de cálculo de Microsoft Office Excel	52 KB
Gestión de Horas v6.xlsx	Hoja de cálculo de Microsoft Office Excel	51 KB
Gestión de Horas v7.xlsx	Hoja de cálculo de Microsoft Office Excel	69 KB

Ilustración 8-3 Ejemplo manejo de versiones de documentos en Google Drive

8.7.2 Versiones código fuente

Para las versiones del código fuente se comenzó estableciendo los permisos que iba a poseer cada integrante del equipo en el acceso al repositorio del código fuente en *Bitbucket*, por lo que al encontrarse todo el equipo involucrado con tareas de desarrollo, se determinó que todos tuvieran permisos de lectura y escritura sobre los elementos almacenados en el repositorio.

Inicialmente y para generar un flujo transparente y seguro de trabajo con las versiones del producto de software, se definieron las siguientes reglas:

- Al comenzar a trabajar en la solución, se debían bajar los cambios de la misma y en el caso de ocurrir conflictos correspondía resolverlos antes de comenzar con el desarrollo.

- Al finalizar una *User Story* debía ser *commiteada*, para que la misma estuviera disponible en el repositorio virtual para el resto del equipo.

- No se podía hacer *commit* al repositorio en *Bitbucket* si el proyecto contenía errores de compilación.

- No se podía hacer *commit* al repositorio en *Bitbucket* si el proyecto contenía errores de ejecución conocidos, tales como pruebas unitarias o integración insatisfactorias.

- Registrar comentarios al momento de subir cambios al repositorio, a modo de mantener una claridad en las diferentes modificaciones realizadas por parte de los integrantes del equipo.

- Al subir un cambio sobre una funcionalidad/tarea o bien una nueva *User Story* del *Sprint* actual, se debían probar el resto del sistema a modo de garantizar consistencia e integridad a modo general.

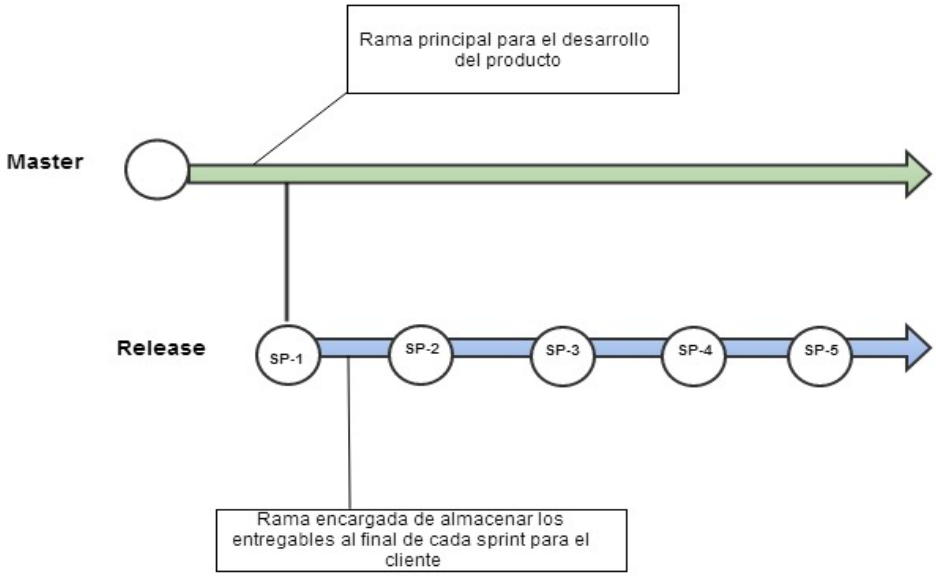


Ilustración 8-4Estructura de ramas del código fuente

Como se observa en la anterior figura, se contaba con una estructura de 2 ramas para el código fuente. Una denominada “Master”, la principal que es la que se iban realizando todas las funcionalidades para el producto de software. Luego, paralelamente a esta, se contaba con la rama de “Release” en donde se almacenaban los entregables parciales al cierre de cada *Sprint* con el Cliente.

El equipo definió realizar un *branch* en el repositorio bajo el nombre de RELEASE, con el fin de volver a una versión incremental y funcional si es necesario de forma sencilla, mediante comandos provistos por la herramienta GIT. En los comentarios de la versión subida al repositorio se nombra a qué *Sprint* pertenece la versión. Por lo tanto, el equipo trabajaba sobre un *branch* principal de desarrollo (llamado master) y al fin de cada *Sprint* generaba una nueva actualización en el *branch* RELEASE con el entregable indicado y el comentario correspondiente.

8.8 Control de cambios

El ciclo de control de cambios de desarrollo en el producto de software se enfocaba en seguir una línea por parte de todo el equipo para no generar retrabajo de desarrollo e inestabilidad en las versiones existentes en los equipos locales de cada integrante.

Por lo tanto cada vez que se iba a trabajar en la solución, se realizaba una descarga (*Pull*) de los últimos cambios efectuados para contar con lo último subido a *Bitbucket*. Posteriormente se realizaban las tareas pertinentes (nuevos cambios, mejoras de funcionalidades) y a continuación se debía asegurar que lo realizado funcionara de forma correcta y no afectara al resto de las funcionalidades de iPanel Educativo.

Finalmente y con el aval de que la solución se encontraba sin problemas, se realizaban los *commit* necesarios para los archivos utilizados en *SourceTree* y se subían (*Push*) los mismos para sincronizarlos con el repositorio virtual en *Bitbucket*. De existir algún problema de ejecución o compilación del sitio, se volvía a la etapa de Realización de Cambio o Mejora.

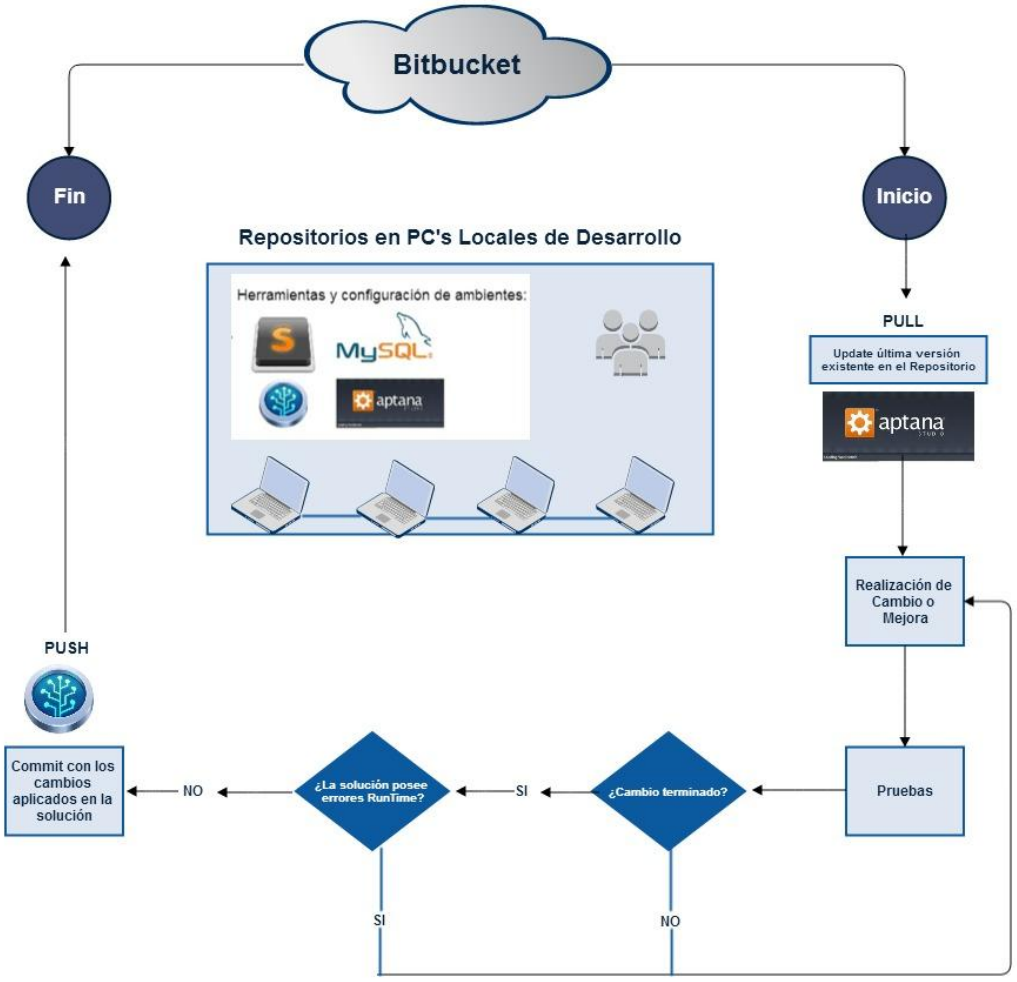


Ilustración 8-5 Diagrama de flujo para reflejar control de cambios

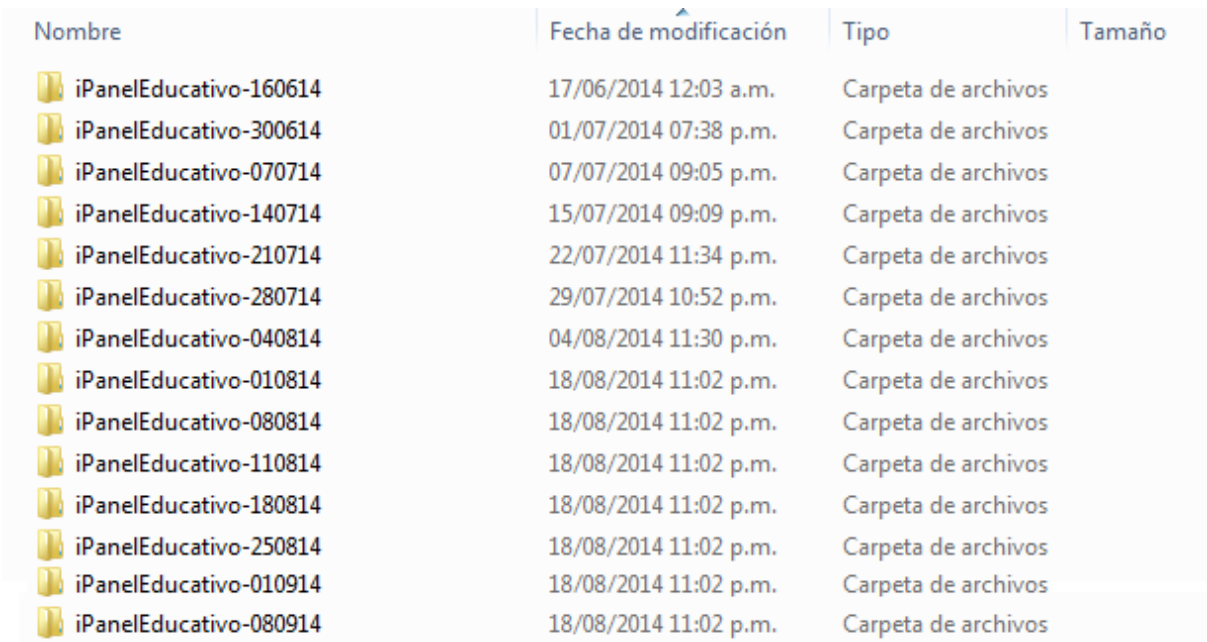
8.9 Respaldos

Se definieron una serie de políticas de respaldos para el rol del SCM, no sólo para el caso del código fuente almacenado en el repositorio en *Bitbucket*, sino también para la documentación en *Google Drive* y la base de datos en *MySQL Workbench*.

En el caso del código fuente de la solución, cada integrante contaba con una copia en sus equipos locales para el desarrollo diario, además de la versión común existente en el repositorio de *Bitbucket*.

Para la documentación, la herramienta de *Google Drive* garantizaba seguridad para su almacenamiento en el repositorio. Sin embargo cada uno de los miembros del equipo poseía la versión desktop con la carpeta local instalada en sus equipos con la versión actualizada de la documentación en la nube.

De todos modos se decidió realizar a partir del *Sprint* 2 en adelante, semanalmente un respaldo de la carpeta inherente a la documentación, copia de la solución del código fuente en *Bitbucket* y también extraer un *dump* de la base de datos local en *MySQL Workbench* en un disco extraíble físico por parte del encargado de SQA/SCM del proyecto.



Nombre	Fecha de modificación	Tipo	Tamaño
iPanelEducativo-160614	17/06/2014 12:03 a.m.	Carpeta de archivos	
iPanelEducativo-300614	01/07/2014 07:38 p.m.	Carpeta de archivos	
iPanelEducativo-070714	07/07/2014 09:05 p.m.	Carpeta de archivos	
iPanelEducativo-140714	15/07/2014 09:09 p.m.	Carpeta de archivos	
iPanelEducativo-210714	22/07/2014 11:34 p.m.	Carpeta de archivos	
iPanelEducativo-280714	29/07/2014 10:52 p.m.	Carpeta de archivos	
iPanelEducativo-040814	04/08/2014 11:30 p.m.	Carpeta de archivos	
iPanelEducativo-010814	18/08/2014 11:02 p.m.	Carpeta de archivos	
iPanelEducativo-080814	18/08/2014 11:02 p.m.	Carpeta de archivos	
iPanelEducativo-110814	18/08/2014 11:02 p.m.	Carpeta de archivos	
iPanelEducativo-180814	18/08/2014 11:02 p.m.	Carpeta de archivos	
iPanelEducativo-250814	18/08/2014 11:02 p.m.	Carpeta de archivos	
iPanelEducativo-010914	18/08/2014 11:02 p.m.	Carpeta de archivos	
iPanelEducativo-080914	18/08/2014 11:02 p.m.	Carpeta de archivos	

Ilustración 8-6 Captura Respaldo en Disco Extraíble Físico

9. Gestión del proyecto

9.1 Marco de trabajo

Al inicio del proyecto se evaluaron dos metodologías de gestión del proyecto: tradicional y ágil; ambas basadas en la experiencia de trabajo de los integrantes del equipo, unos con mayor conocimiento sobre la gestión de proyectos en base al marco de trabajo del *Project Management Institute* (*PMI*) y otros con mayor conocimiento del marco ágil *Scrum*. Luego de reuniones con el cliente y comenzar a analizar su problema, se decidió seleccionar *Scrum* como marco de trabajo para el proyecto. El mismo no se utilizó en su versión más pura sino que fue adaptado a la realidad y particularidades del proyecto.

Este marco permitió mostrarle al cliente periódicamente los avances del desarrollo del producto mediante la liberación de nuevas funcionalidades del sistema. Es así como el cliente pudo probar su sistema, pedir cambios sin impactar negativamente en el proyecto y solicitar nuevos requerimientos que iban surgiendo con el uso del mismo.

Entre las múltiples ventajas del marco, se puede destacar la gestión regular de las expectativas del cliente, los resultados anticipados, la flexibilidad y adaptación, el alineamiento entre el equipo y el cliente, la temprana mitigación de riesgos, el equipo se mantiene motivado y se mantiene en continua mejora de su productividad y calidad en el trabajo.[10]

Luego de varias reuniones con el cliente, donde se decidió la solución a implementar, se especificó una visión general del producto de la cual se fueron desprendiendo requerimientos que se almacenaron en un artefacto utilizado en *Scrum* llamado *Product Backlog* (Ver Anexo 2 – *Product Backlog*). A su vez, se definió el rol de *Product Owner*, una persona responsable de este documento, fundamental para alcanzar el éxito del producto.

9.1.1 Roles de Scrum

Product Owner: es la persona responsable del éxito del producto desde el punto de vista de los *stakeholders*, determina las características del producto, las prioriza, acepta/rechaza las versiones construidas y provee el *feedback* al equipo de desarrollo.

En nuestro caso particular, una persona perteneciente al cliente desarrolló este rol en el proyecto y fue intermediario entre el equipo de proyecto y los socios de la editorial. Si bien, era una reciente incorporación a la editorial y estaba comenzando su primera experiencia en este rol, cumplió satisfactoriamente esta posición estando siempre disponible y atento a evacuar dudas.

ScrumMaster: es el *coach* del equipo, un líder por ser un ejemplo a seguir, un facilitador por fomentar contextos de apertura y discusión donde todos pueden expresar sus opiniones y lograr consensos comunes.

Es responsable por el correcto empleo y evolución de *Scrum*, facilita su uso como la remoción de los impedimentos, asegura la cooperación y comunicación dentro del equipo. Este rol fue cumplido por Guzmán Porro, quien ya contaba con experiencia en el mismo.

Equipo de Desarrollo: está formado por todos los individuos necesarios para la construcción del producto, es auto-organizado, provee las estimaciones de cuánto esfuerzo será requerido para cada una de las características del producto, se compromete al comienzo de cada *Sprint* a construir un conjunto determinado de características en el tiempo que dura el mismo y es responsable por la construcción y la entrega del producto terminado al finalizar cada *Sprint*.

Todos los integrantes del equipo de proyecto formaron parte del Equipo de Desarrollo de *Scrum*.

9.1.2 Actividades de *Scrum*

Las iteraciones en *Scrum* se conocen como *Sprint*s. En el proyecto, estos comenzaban los lunes y finalizaban los domingos. La duración definida fue de 3 semanas, basándonos en consultas realizadas al *Product Owner* y a un experto en el tema de ORTsf.

La decisión fue tomada considerando que el equipo de proyecto no contaba con una dedicación full time para el mismo; y siendo el objetivo de las iteraciones exponer los avances del producto y realizar los encuentros formales con el cliente para visualizar el progreso del mismo, se acordó con el *Product Owner*, en base a los tiempos que contaba el cliente, que la periodicidad ideal para realizar reuniones de planificación y revisión era de 20 días.

La estimación del esfuerzo requerido para las tareas de desarrollo se realizó mediante *Story Points*, inicialmente en base a la experiencia individual de los integrantes del equipo y luego se fue ajustando en comparación con las tareas finalizadas a lo largo del proyecto.

Adicionalmente, se estimó esfuerzo en horas hombre para equilibrar el trabajo en todos los miembros del equipo por igual y contar con la métrica de cuánto tiempo se insumía para las distintas actividades de todo el proyecto.

A continuación se describe en detalle la dinámica de reuniones utilizada.

Planificación del *Sprint*

Al comienzo de cada *Sprint* se realizó una reunión presencial de planificación, con una duración máxima de 6 horas. Esta se dividió en dos partes, la primera trató de ¿Qué trabajo será realizado?, mientras que la segunda fue sobre ¿Cómo será realizado el trabajo?

Para la primera se acordó con el Equipo de Desarrollo y el *Product Owner* el alcance del *Sprint* en cuestión, basado en los requerimientos priorizados del *Product Backlog*.

En la segunda, el equipo determinó la forma de llevar adelante el trabajo, identificando y auto-asignando tareas y responsabilidades. De aquí se desprende el *Sprint Backlog* que contiene las funcionalidades que serán implementadas en la iteración.

El equipo se reunió cada tres lunes para realizar esta ceremonia, durante los 5 *Sprint*s del proyecto. La estimación de *Story Points* de cada tarea fue realizada utilizando la herramienta *PlanningPoker* [11], en base al conocimiento de esfuerzo de desarrollo de cada integrante del equipo y posteriormente a la comparación con métricas anteriores.

Scrum Diario

Scrum promueve reuniones informales diarias, de no más de 15 minutos, donde el Equipo de Desarrollo pueda contestar las siguientes preguntas cara a cara:

- ¿Qué hice desde la última reunión diaria hasta ahora?
- ¿En qué voy a estar trabajando desde ahora hasta la próxima reunión diaria?
- ¿Qué problemas o impedimentos tengo?

Se considera un espacio de comunicación entre todos los miembros del equipo y no un reporte de avance.

Estas pautas fueron adoptadas en el proyecto con la particularidad de que, al no trabajar juntos diariamente todos los integrantes del equipo, las reuniones diarias fueron realizadas mediante *Hangouts* a excepción de aquellos días donde el equipo se reunía en un mismo lugar físico.

Revisión del *Sprint*

Al final de cada *Sprint* se realizó una reunión presencial no mayor a 3 horas, donde el *Product Owner* evaluó el incremento funcional generado por el Equipo de Desarrollo. Se obtuvo el *feedback* del cliente al respecto y en base a este se realizaron adaptaciones al *Product Backlog* cuando fue necesario incluir nuevos requerimientos, así como cambios en las funcionalidades desarrolladas. En algunos casos la revisión del producto era aceptada tal cómo estaba construida y no había cambios en las funcionalidades o nuevos requerimientos.

Retrospectiva del *Sprint*

La reunión de retrospectiva consistió en una oportunidad para el Equipo de Desarrollo de inspeccionarse a sí mismo y reflexionar sobre la forma en la que realizó el trabajo y los acontecimientos que sucedieron en el *Sprint* concluido. Las mismas se realizaron informalmente entre los miembros del equipo, con una duración máxima de 3 horas, y se acordaron las oportunidades de mejora para llevar a cabo en el siguiente *Sprint*.

La siguiente figura obtenida de *ScrumAlliance*, muestra gráficamente el ciclo de *Scrum* que se repite en cada iteración. [12]

Ilustración 9-1 Ciclo de Scrum

9.1.3 Adaptación de *Scrum*

Scrum está pensado originalmente para un grupo de personas que trabajan diariamente en conjunto en el mismo espacio físico. Ésta y otras características se adaptan a cada proyecto u organización para poder utilizar el marco de trabajo ajustado a cada realidad; pero siempre teniendo como base los principios y valores de *Scrum [13]*. Los mismos fueron respetados por el equipo (Ver Anexo 9 – Principios y Valores de *Scrum*) y en la tabla siguiente se describen las adaptaciones realizadas para el proyecto en cuestión.

Características de *Scrum* puro	**Adaptaciones de *Scrum* a este proyecto**
Figura de *Product Owner*.	Se contó con la presencia de una persona que integra el equipo del cliente, dedicada a las siguientes tareas realizadas por la figura de *Product Owner* de *Scrum*: definición, priorización y aprobación de requerimientos y versiones del producto.
Contacto directo y frecuente con el cliente y el equipo.	El punto de contacto con el cliente fue siempre el *Product Owner* que mostró total disponibilidad para reuniones, por mail, vía chat y teléfono. El equipo se mantuvo siempre en contacto realizando reuniones presenciales, vía *Hangouts*, mail y/o teléfono.
Planificación por iteración.	Se realizaron reuniones de planificación al comienzo de cada una de las iteraciones, donde se determinaron las funcionalidades a desarrollar para el *Sprint* y los criterios de aprobación conjuntamente con el *Product Owner*.
Requerimientos poco definidos y/o cambiantes.	Al comienzo del proyecto se tenía definido un alcance a alto nivel, junto al cliente, pero este se encontraba abierto al cambio y las funcionalidades requerían de mayor detalle y definición para su implementación.
Entregas frecuentes.	Al finalizar cada iteración, el *Product Owner* podía probar la funcionalidad desarrollada, proponer los cambios necesarios o aprobarla en caso de que la considerara completa y cerrada.
El equipo de desarrollo trabaja junto en un mismo lugar físico.	Si bien no se trabajaba físicamente en un mismo lugar todos los días, el equipo se mantuvo en diaria comunicación.

Comunicación cara a cara entre los miembros del equipo.	Las *Daily Mettings* se realizaban en forma presencial durante las 3 o 4 reuniones semanales, mientras el resto de los días las comunicaciones eran a distancia vía *Hangouts*.
A intervalos regulares, el equipo reflexiona sobre cómo ser más efectivo para ajustar y mejorar su comportamiento.	A partir del *Sprint* 0, se adquirió una primera experiencia sobre la estimación y el trabajo de una iteración, obteniendo métricas al respecto y permitiendo mejorar y ajustar el funcionamiento del equipo. En las siguientes iteraciones se replicó el proceso, haciéndolas más productivas.
Realizar las tareas que aportan valor al cliente.	Las funcionalidades desarrolladas fueron definidas, priorizadas y aceptadas por el *Product Owner* contemplando las necesidades del cliente. Los requerimientos y documentación sobre el producto que el equipo de proyecto entregó al cliente correspondieron a estas mismas necesidades.
Colaboración abierta de todo el equipo.	Durante todo el proyecto se contó con el apoyo de todos los integrantes, independientemente del problema que hubiera que enfrentar y de los roles asignados.
*Sprint*s de entre 1 y 4 semanas de duración.	La duración de los *Sprint*s fue de 3 semanas, con excepción del *Sprint* 0 y el *Sprint* 5 que se desarrollaron en 4 semanas, debido a que el comienzo y fin del proyecto requirieron más tareas para cumplir con requerimientos del proyecto académico como tal.
Tamaño de equipo entre 4 y 8 personas.	El equipo estuvo conformado por 4 personas.
El diseño y la arquitectura del producto son emergentes.	El diseño y la arquitectura del producto fueron evolucionando constantemente con el transcurso del proyecto.

Tabla 9-1 Características y adaptaciones de Scrum

9.1.4 Ciclo de Vida

Se definió la utilización del ciclo de vida incremental iterativo para el proyecto. Esto se corresponde a que la gran mayoría de los requerimientos fueron establecidos al comienzo del proyecto, para cumplir con el objetivo académico de contar con un

alcance apropiado, ajustado al tiempo de duración del proyecto y a los integrantes que conforman el mismo.

Debido a que el cliente aún no contaba con todos los insumos definidos para el proyecto, existía cierta volatilidad de los requerimientos que debía contemplarse y considerarse como un riesgo. Para ello, se decidió utilizar iteraciones que permitieran a lo largo del transcurso del proyecto relevar e incorporar nuevas funcionalidades así como aprobar las ya realizadas y descartar otras según consideraciones del cliente.

9.1.5 Asignación de responsabilidades

Para el caso de las áreas de Ingeniería de Software, cada integrante del equipo eligió un rol en base a su experiencia y preferencia, para desempeñarse como responsable de las tareas que le corresponden al mismo. Si bien las decisiones fueron tomadas por el equipo en conjunto y en base a consenso, el responsable de determinado rol tuvo más injerencia sobre sus actividades específicas.

Área	Responsable
Ingeniería de requerimientos	Matías Viera
Arquitectura de software	Guzmán Porro
SQA y SCM	Martín Martínez
Gerencia de proyecto	Carolina Romero
Desarrollo	Matías Viera, Guzmán Porro, Martín Martínez, Carolina Romero
Testing	Matías Viera, Guzmán Porro, Martín Martínez, Carolina Romero

Tabla 9-2 Asignación de responsabilidades

Se elaboró una matriz de asignación de responsabilidades con formato *RACI* para ilustrar las conexiones entre las actividades realizadas y los miembros del equipo del proyecto y demás interesados. De esta manera se obtuvo una guía desde el comienzo del proyecto sobre los responsables (R) de las principales actividades, aquellos a los que se debía consultar (C), informar (I) y requerir de su aprobación (A). [14]

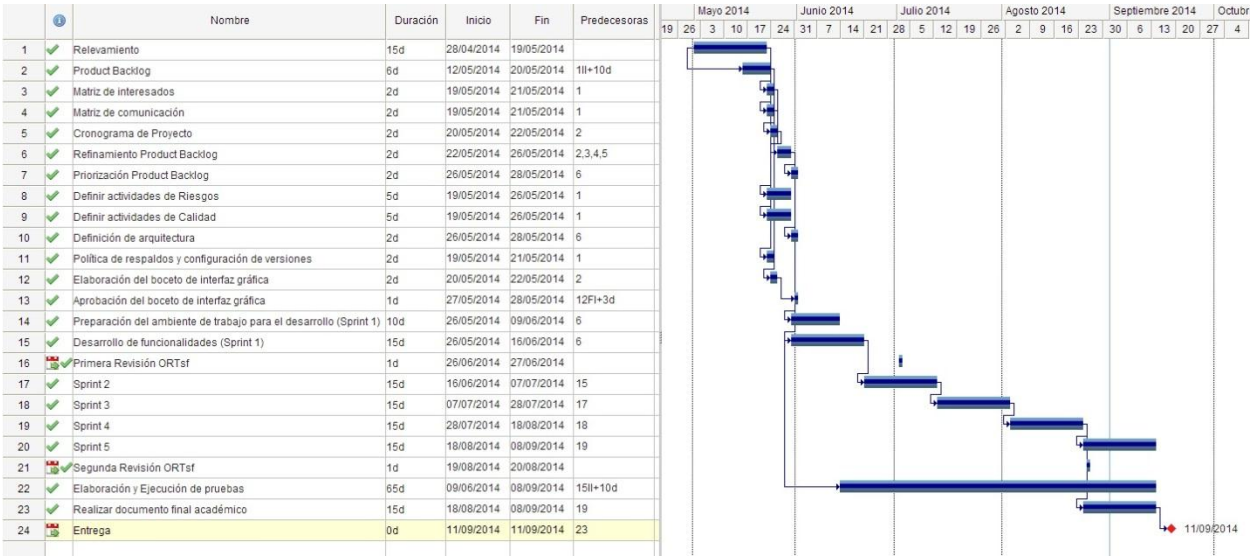
Rol	Cliente / Stakeholder	Cliente / Product Owner	Arquitecto de software	SQA / SCM	Gerente de proyecto	Ingeniero de requerimientos / Diseñador gráfico	Tutor de proyecto	Docente de ORTsf
Actividades / Nombre	**Nicolás Pereyra**	**Braulio de León**	**Guzmán Porro**	**Martín Martínez**	**Carolina Romero**	**Matías Viera**	**Ignacio Valle**	
Definición de arquitectura	I	C/A	R	I	I	I	C	C
Definición de actividades y métricas de Calidad			I	R	I	I	C	C
Identificación y gestión de Riesgos			I	I	R	I	C/I	C
Cronograma de Proyecto	I	A	I	I	R	I	C/I	C
Matriz de interesados		I	I	I	R	I	C	
Cuadro de comunicaciones		I	I	I	R	I	C	
Definición de Product Backlog	C	A/R	R	R	R	R	C/I	I
Definición de Sprint Backlog		A/R	R	R	R	R	C/I	
Definición de política de respaldos y configuración de versiones			I	R	I	I	C	
Elaboración de casos de prueba			I	I	I	R	C	
Elaboración de boceto de interfaz gráfica	C	A/C	I	I	I	R	I	
Desarrollo de funcionalidades			R	R	R	R	C/I	
Ejecución de pruebas			R	R	R	R	C/I	

Tabla 9-3 Matriz RACI

9.1.6 Cronograma

Al inicio del proyecto se realizó un cronograma de alto nivel de las principales actividades que serían desarrolladas durante el mismo, con el objetivo de tener una visión global que nos permitiera asignar en la agenda, eventos como ser las revisiones académicas, las fechas de inicio y fin de *Sprint*, reuniones con el cliente, entre otros. Para ello utilizamos la herramienta *Gantter* de *Google Drive* que permitió a todos los integrantes del equipo acceder al mismo y modificarlo cuando era necesario.

En la siguiente figura se observa la impresión del cronograma utilizado.



	Nombre	Duración	Inicio	Fin	Predecesoras
1	Relevamiento	15d	28/04/2014	19/05/2014	
2	Product Backlog	6d	12/05/2014	20/05/2014	1II+10d
3	Matriz de interesados	2d	19/05/2014	21/05/2014	1
4	Matriz de comunicación	2d	19/05/2014	21/05/2014	1
5	Cronograma de Proyecto	2d	20/05/2014	22/05/2014	2
6	Refinamiento Product Backlog	2d	22/05/2014	26/05/2014	2,3,4,5
7	Priorización Product Backlog	2d	26/05/2014	28/05/2014	6
8	Definir actividades de Riesgos	5d	19/05/2014	26/05/2014	1
9	Definir actividades de Calidad	5d	19/05/2014	26/05/2014	1
10	Definición de arquitectura	2d	26/05/2014	28/05/2014	6
11	Política de respaldos y configuración de versiones	2d	19/05/2014	21/05/2014	1
12	Elaboración del boceto de interfaz gráfica	2d	20/05/2014	22/05/2014	2
13	Aprobación del boceto de interfaz gráfica	1d	27/05/2014	28/05/2014	12FI+3d
14	Preparación del ambiente de trabajo para el desarrollo (Sprint 1)	10d	26/05/2014	09/06/2014	6
15	Desarrollo de funcionalidades (Sprint 1)	15d	26/05/2014	16/06/2014	6
16	Primera Revisión ORTsf	1d	26/06/2014	27/06/2014	
17	Sprint 2	15d	16/06/2014	07/07/2014	15
18	Sprint 3	15d	07/07/2014	28/07/2014	17
19	Sprint 4	15d	28/07/2014	18/08/2014	18
20	Sprint 5	15d	18/08/2014	08/09/2014	19
21	Segunda Revisión ORTsf	1d	19/08/2014	20/08/2014	
22	Elaboración y Ejecución de pruebas	65d	09/06/2014	08/09/2014	15II+10d
23	Realizar documento final académico	15d	18/08/2014	08/09/2014	19
24	Entrega	0d	11/09/2014	11/09/2014	23

Ilustración 9-2 Captura de pantalla del cronograma de proyecto

El Decano de Facultad de Ingeniería comunicó la asignación de proyectos el día 23/04/2014 por lo que el lunes siguiente (28/04) dimos por comenzado el proyecto con la primera reunión para conocer personalmente al cliente en las oficinas del mismo.

El equipo definió un *Sprint* 0 como el inicial del proyecto, previo al comienzo del desarrollo, para analizar el problema del cliente e investigar las soluciones posibles. Posteriormente se prosiguió con las siguientes iteraciones hasta el final del proyecto.

En el último *Sprint* las tareas restantes de desarrollo y Testing fueron centralizadas en su mayoría en dos integrantes del equipo mientras que los otros dos miembros se dedicaron a la documentación del proyecto para cumplir con la entrega final del 11/09. Una vez finalizado el *Sprint* 5 todo el equipo dedicó sus horas de trabajo a culminar con dicha documentación.

En la siguiente tabla se muestran los hitos establecidos en el proyecto.

Tarea	Duración (días)	Inicio	Fin
Sprint 0	28	28/04/2014	25/04/2014
Planificación *Sprint* 1	-	26/05/2014	26/05/2014
***Sprint* 1**	21	26/05/2014	15/06/2014
Cierre *Sprint* 1	-	15/06/2014	15/06/2014
Planificación *Sprint* 2	-	16/06/2014	16/06/2014
***Sprint* 2**	21	16/06/2014	06/07/2014
Primera revisión	-	26/06/2014	26/06/2014
Cierre *Sprint* 2	-	06/07/2014	06/07/2014
Planificación *Sprint* 3	-	07/07/2014	07/07/2014
***Sprint* 3**	21	07/07/2014	27/07/2014
Cierre *Sprint* 3	-	27/07/2014	27/07/2014
Planificación *Sprint* 4	-	28/07/2014	28/07/2014
***Sprint* 4**	21	28/07/2014	17/08/2014
Cierre *Sprint* 4	-	17/08/2014	17/08/2014
Planificación *Sprint* 5	-	18/08/2014	18/08/2014
***Sprint* 5**	28	18/08/2014	07/09/2014
Segunda revisión	-	19/08/2014	19/08/2014
Cierre *Sprint* 5	-	07/09/2014	07/09/2014

Tabla 9-4 Hitos del proyecto

9.2 Gestión de la comunicación

La buena comunicación en un proyecto basado en un marco de trabajo ágil es fundamental. Para ello se utilizaron las siguientes herramientas que permitieron mantener la coordinación en el equipo y con el cliente.

Herramienta para el desarrollo con *Scrum* – Se utilizó *TargetProcess*, una aplicación en la web que permitió ingresar las tareas de desarrollo y testing a realizar con los responsables asignados, su esfuerzo y horas, estimado y real, su estado (abierta, en progreso, en testing, terminada), sus respectivos bugs para solucionar, entre otras. Esto permitió al equipo obtener toda la información centralizada en un mismo lugar y asegurar la integridad de la misma sin duplicar datos ni esfuerzos.

Reuniones – El equipo realizó reuniones presenciales tres o cuatro veces por semana para además de trabajar en las tareas asignadas, evaluar el avance del proyecto, solucionar impedimentos en el desarrollo, planificar y coordinar acciones futuras. En una de estas reuniones realizadas en la sede Centro de la Universidad, se contaba con el apoyo del tutor del proyecto para evacuar dudas, aportar ideas y recibir *feedback* sobre el progreso del mismo.

Mensajería instantánea, mail y video llamadas – Durante todo el proyecto se mantuvo el contacto vía *Hangouts*, llamadas telefónicas o simples mensajes de *Whatsapp*, tanto entre los miembros del equipo como con el tutor y el *Product Owner*. Estos fueron los medios alternativos utilizados cuando no era posible concretar reuniones con todos los involucrados presentes y una forma de mantener una comunicación ágil y efectiva.

El siguiente cuadro fue utilizado al principio del proyecto, cuando recién se establecía la dinámica de trabajo, y permitió servir de guía para el equipo de proyecto de qué comunicar y cómo.

Qué vamos a comunicar?	A quién?	De qué manera?
Cuadro de comunicaciones	Equipo de proyecto, *Product Owner* y Tutor	Correo electrónico
Cronograma de proyecto	Equipo de proyecto, *Product Owner* y Tutor	Correo electrónico
Product Backlog	Equipo de proyecto, *Product Owner* y Tutor	Reunión presencial al inicio de cada *Sprint*
Avances del producto	*Product Owner*	Reuniones presenciales cada 21 días
Problemas en el desarrollo	Equipo de proyecto	Reuniones en Hangouts
Consultas sobre la documentación	Tutor y Equipo de proyecto	Reuniones presenciales semanales

Consultas generales sobre el proyecto y avance del mismo	Tutor y Equipo de proyecto	Reuniones presenciales semanales y Hangouts
Planificación del *Sprint*	Equipo de proyecto y *Product Owner*	Reuniones presenciales cada 21 días
Scrum Diario	Equipo de proyecto	Reuniones presenciales y Hangouts
Riesgos del *Sprint*	Equipo de proyecto	Al comienzo de cada *Sprint*
Riesgo ocurrido	Equipo de proyecto, Tutor y *Product Owner* si es necesario	Reunión presencial eventual o llamada telefónica
Cierre de *Sprint*s	Equipo de proyecto y *Product Owner*	Reuniones de revisión y retrospectiva al finalizar cada *Sprint*

Tabla 9-5 Guía de comunicación

9.3 Gestión de riesgos

Según la Guía del PMI (PMBOK® Guide), un riesgo se define como una condición o evento incierto que, si ocurre, tiene un efecto positivo o negativo en uno o más objetivos del proyecto como son el alcance, tiempo, costo y calidad. [14]

El proceso de gestión de riesgos consistió en la identificación de los mismos, un análisis cualitativo y cuantitativo de cada uno de ellos, el diseño de un plan de respuesta y el seguimiento y control de los riesgos a lo largo del proyecto. Siendo *Scrum* el marco de trabajo utilizado para el proyecto, se adaptó la gestión de riesgos para realizar un monitoreo al comienzo de cada iteración. De esta manera los riesgos fueron evaluados en la Planificación del *Sprint* para que todo el equipo tuviera en cuenta los riesgos definidos como prioritarios en el *Sprint* que se estaba por comenzar.

9.3.1 Identificación de riegos

Una primera identificación de riesgos fue realizada durante el *Sprint* 0 mediante una tormenta de ideas por parte de todo el equipo y posterior juicio experto, consultando con líderes de proyecto de software y otros grupos que realizaron el proyecto de fin de carrera en años pasados.

De estas reuniones se elaboró una lista de riesgos que permaneció abierta hasta el último *Sprint* del proyecto debido a que los mismos estaban sujetos a posibles nuevos requerimientos en el *Product Backlog* y/o a la eliminación de otros; y a cambios que afectaran al proyecto, etc. Ver Anexo 10 – Riesgos del proyecto.

9.3.2 Análisis cualitativo y cuantitativo de riesgos

Para cada uno de los riesgos se realizó un análisis cualitativo y cuantitativo con el objetivo de priorizar los mismos, medir su magnitud y monitorearlos. Para ello se determinó la probabilidad de ocurrencia de cada riesgo y su respectivo impacto en el proyecto.

A continuación se detallan las escalas utilizadas en este análisis.

9.3.3 Probabilidad de ocurrencia

Representa las chances de manifestación de un riesgo en el proyecto. Para ello se utilizó una escala de 0 a 1 que se describe en la siguiente tabla.

Escala	Descripción
0.1	Muy baja Es una situación muy poco probable que suceda.
0.3	Baja. La probabilidad que suceda el riesgo es baja.
0.5	Media. La probabilidad que se manifieste el riesgo en algún momento del proyecto es media. Se deberá considerar alguna acción correctiva en el caso de que el riesgo cuenta con un gran impacto en el proyecto.
0.7	Alta. Son riesgos que tienen altas chances de ocurrir y deberán contar con acciones que mitiguen sus causas.
0.9	Muy Alta. Es muy probable que ocurra el riesgo debido a los motivos que lo causan, por lo que hay que prestarles especial atención durante toda la gestión del proyecto.

Tabla 9-6 Probabilidad de ocurrencia

9.3.4 Impacto sobre el proyecto

Representa la importancia del riesgo materializado en el proyecto. Para ello se utilizó una escala de 1 a 5 que se describe en la siguiente tabla.

Escala	Descripción
1	Muy bajo. El impacto es leve para el proyecto.
2	Bajo. El impacto es menor sobre el proyecto.
3	Medio. El impacto es moderado sobre el proyecto pero debe ser controlado porque podría causar desvíos en la planificación.
4	Alto. El impacto es grave sobre el proyecto. Podría poner en peligro el logro de los objetivos de alguno de los *Sprint* del proyecto.

5	Muy Alto. El impacto es severo sobre el proyecto. Ocasionaría atrasos mayores en la ejecución de los *Sprint* y consecuencias críticas en el proyecto que hasta podría resultar en la cancelación del mismo.

Tabla 9-7 Impacto sobre el proyecto

9.3.5 Magnitud

Luego de analizar los riesgos y calcular su probabilidad e impacto en el proyecto, se registró la magnitud de la manifestación de un riesgo a partir de la fórmula (Probabilidad * Impacto) que se muestra en el siguiente cuadro. De este modo se establece la prioridad en la que serán gestionados dedicando especial control a aquellos que se clasificaron en rojo con una magnitud igual o mayor a 2.

Probabilidad					
0,9	0,9	1,8	2,7	3,6	4,5
0,7	0,7	1,4	2,1	2,8	3,5
0,5	0,5	1	1,5	2	2,5
0,3	0,3	0,6	0,9	1,2	1,5
0,1	0,1	0,2	0,3	0,4	0,5
	1	2	3	4	5
	Impacto				

Prioridad
Alta
Media
Baja

Tabla 9-8 Matriz de magnitud de riesgos

9.3.6 Respuesta a los riesgos

Una vez realizado el análisis cualitativo, se evaluaron las siguientes estrategias de respuesta a los riesgos, en base a lo señalado en el PMBOK® Guide. [14]

- Evitar: Cambiar el plan de gestión del proyecto para eliminar la amenaza que representa un riesgo adverso, aislar los objetivos del proyecto del impacto del riesgo.

- Transferir: Consiste en trasladar el impacto negativo de una amenaza, junto con la propiedad de la respuesta, a un tercero. Simplemente se le da a otra parte la responsabilidad de su gestión; no lo elimina.

- Mitigar: Implica reducir la probabilidad y / o el impacto de un evento de riesgo adverso a un umbral aceptable.

- Aceptar: Esta estrategia indica que el equipo del proyecto ha decidido no cambiar el plan de gestión del proyecto para hacer frente a un riesgo, o no ha podido identificar ninguna otra estrategia de respuesta adecuada.

Posterior a este análisis se decidió mitigar los riesgos del proyecto y para cada uno de ellos se definieron acciones preventivas que fueron llevadas a cabo para reducir la probabilidad de ocurrencia de los mismos. Ver Anexo 10 – Riesgos del proyecto.

9.3.7 Seguimiento y control

El monitoreo de los riesgos fue llevado a cabo durante todo el proyecto. Al inicio de cada *Sprint* se analizaron los riesgos con mayor exposición, pero ninguno de ellos se concretó o impactó en el desarrollo o resultado del proyecto. Este seguimiento permitió observar la evolución de los riesgos a lo largo del proyecto, como se muestra en la siguiente figura para los principales identificados al comienzo del proyecto.

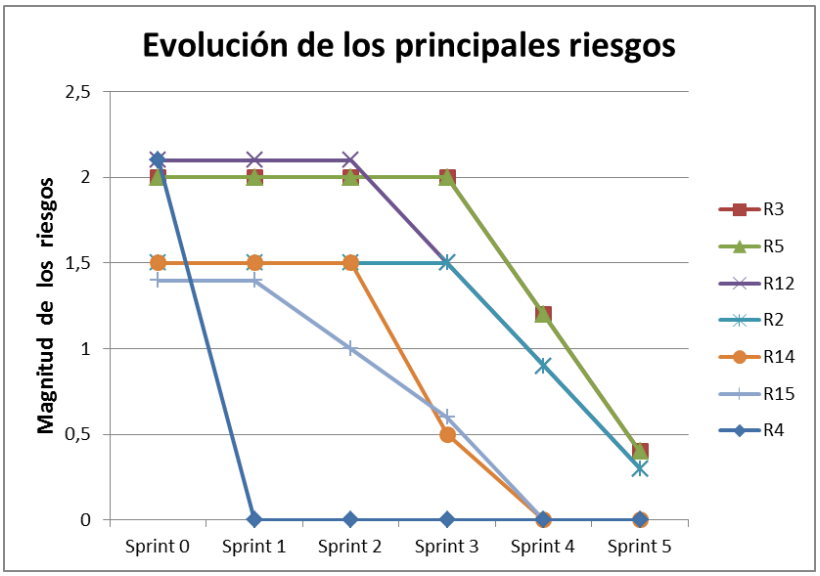


Ilustración 9-3 Evolución de los principales riesgos

Id	Descripción	Sprint 0	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5
R3	No cumplir con un alcance acorde a los tiempos del proyecto académico	2	2	2	2	1,2	0,4
R5	Requerimientos poco claros	2	2	2	2	1,2	0,4
R12	Estimaciones equivocadas sobre el esfuerzo de las tareas	2,1	2,1	2,1	1,5	0,9	0,3
R2	Conflictos internos entre los integrantes del equipo	1,5	1,5	1,5	1,5	0,9	0,3
R14	No contar con la arquitectura de datos definidos en la base de datos para el desarrollo del sistema	1,5	1,5	1,5	0,5	0	0
R15	Desarrollar el software en el lenguaje de programación Python	1,4	1,4	1	0,6	0	0
R4	No poder desarrollar un modelo de inteligencia artificial	2,1	0	0	0	0	0

Tabla 9-9 Evolución de los principales riesgos

Para visualizar todos los riesgos identificados en el proyecto y su correspondiente evolución ver Anexo 11 - Evolución de riesgos.

A continuación se detallan los principales riesgos descritos anteriormente.

Id	Nombre del Riesgo	Causa / Descripción / Efecto del Riesgo
R3	No cumplir con un alcance acorde a los tiempos del proyecto académico	Causa: Malas estimaciones de tiempo y/o alcance del proyecto.
		Descripción/Efecto: Si el alcance no se puede realizar en el tiempo establecido para el proyecto deberá recortarse y se habrá generado una mala expectativa en el cliente.
R5	Requerimientos poco claros que atrasen el desarrollo del producto	El cliente no está seguro de lo que quiere y/o el equipo no puede entender sus necesidades.
		Descripción/Efecto: Si los requerimientos son ambiguos o mal especificados el alcance quedará mal definido por lo tanto fracasará el proyecto.
R12	Estimaciones equivocadas sobre el esfuerzo de las tareas	Causa: Falta de experiencia en la estimación del trabajo.
		Descripción/Efecto: Se estiman tiempos equivocados para el desarrollo de las tareas provocando retrasos en el proyecto y sobrecarga de trabajo en el final del mismo.
R2	Conflictos internos entre los integrantes del equipo	Causa: Los integrantes del equipo trabajan juntos por primera vez y en momentos de estrés los intercambios de opinión pueden generar conflictos.
		Descripción/Efecto: Discusiones sobre distintas formas de realizar el trabajo o tomar decisiones pueden ocasionar problemas a la hora de ponerse de acuerdo y ejecutar el trabajo. Esto a su vez genera retrasos en el proyecto.
R14	El cliente no cuente con la arquitectura de datos definidos en la base de datos para el desarrollo del sistema.	Causa: Demoras en finalizar la edición de las tablas de la base de datos de la aplicación del cliente (Cuasimodo).
		Descripción/Efecto: El sistema consumirá los datos de la base de datos del cliente. Si la misma no contiene datos no se puede visualizar el panel de control con información del cliente.
R15	Desarrollar el software en el lenguaje de programación *Python*	Causa: Los miembros del equipo no cuentan con experiencia con el lenguaje *Python*.
		Descripción/Efecto: El desarrollo del sistema puede verse demorado por la falta de experiencia con el lenguaje de programación, lo que provocaría retrasos en el cronograma.
R4	No poder desarrollar un modelo de inteligencia artificial	Causa: Los miembros del equipo no han realizado trabajos previos con inteligencia artificial.
		Descripción/Efecto: Abordar el tema de IA sin tener sólidos conocimientos al respecto lleva al fracaso del proyecto.

Tabla 9-10 Principales riesgos

9.4 Desarrollo de los *Sprints*

9.4.1 Sprint 0

Inicio: 28/04/2014 - Cierre: 25/05/2014

Se definió una duración de 4 semanas para este *Sprint* con el propósito de favorecer la adaptación de todo el equipo de proyecto; y el análisis del problema del cliente y la solución a implementar. La mayor parte del esfuerzo fue dedicado a realizar tareas de investigación, análisis, gestión, definición e implementación de las pautas de trabajo.

Las tareas realizadas fueron las siguientes:

- Análisis de la situación del cliente
- Investigación de las posibles soluciones a implementar
- Definición de actividades de gestión y marco de trabajo
- Definición de actividades de Riesgos
- Definición de matriz de riesgos
- Definición de matriz de responsabilidades
- Justificación de tecnologías utilizadas
- Definición primaria de Diseño y Arquitectura
- Definición de actividades de SQA
- Definición de actividades de SCM
- Elaboración y priorización de *Product Backlog*
- Investigación y definición de herramientas de software para el apoyo a la gestión del proyecto

En la siguiente gráfica se visualiza el porcentaje de trabajo realizado para cada categoría definida.

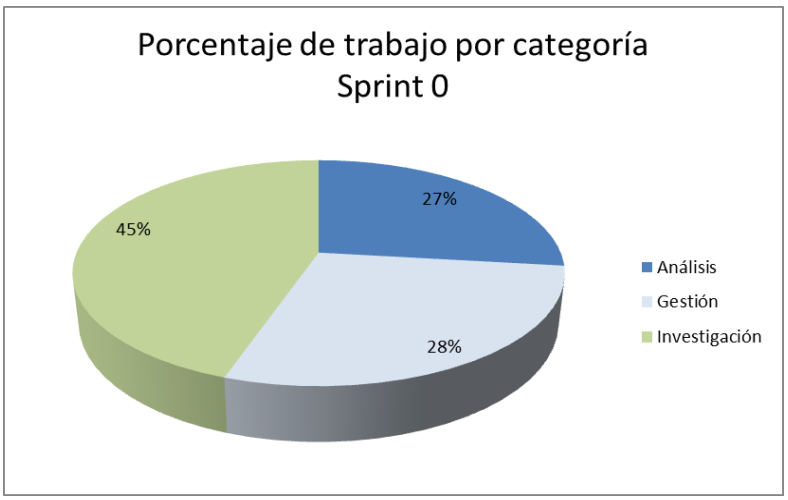


Ilustración 9-4 Porcentaje de trabajo por categoría del Sprint 0

9.4.2 Sprint 1

Inicio: 26/05/2014 - Cierre: 15/06/2014

Este fue el primer *Sprint* donde se trabajó en tareas de desarrollo de software y Testing. Sin embargo, el mayor esfuerzo realizado fue en la capacitación del lenguaje de programación y el *framework* para el desarrollo, superando lo previamente estimado para estas actividades.

Sprint Backlog:

- Configuración ambiente integrado para el producto (*Aptana Studio* con *PythonDjango*, *MySQL*)
- Configuración Repositorio *Bitbucket* con *Git*
- Control Riesgos
- Elaboración de bocetos iniciales - *Photoshop*
- Capacitación en *Python*: Curso *Code Academy*
- Capacitación *Django*: *Django*project.com
- Configuración de *TargetProcess* para el desarrollo con *Scrum*
- Construcción y prueba de la conexión con la base de datos
- Medición de métricas de calidad
- Definición de RNFs
- Capacitación Git - *Bitbucket*
- Construcción *User Story* 229 - Log in
- Construcción *User Story* 326 - Navegación entre secciones
- Construcción *User Story* 232 - Log out
- Pruebas unitarias de las funcionalidades desarrolladas
- Actividades de gestión del proyecto

Métricas generales

***Sprint* 1**		
Métrica	**Valor de referencia**	**Resultado**
Esfuerzo estimado (horas/hombre)	Horas definidas para realizar las tareas	180
Esfuerzo real (horas/hombre)	Esfuerzo estimado	224
Cantidad de tareas realizadas	Tareas definidas en la iteración	18
Cantidad de tareas pendientes	No mayor a 3	0
Porcentaje de cobertura de pruebas	100%	100%
Porcentaje de bugs solucionados	100% de los encontrados	8 (100%)
Porcentaje de bugs pendientes	0% de los encontrados	0 (0%)

Tabla 9-11 Métricas generales Sprint 1

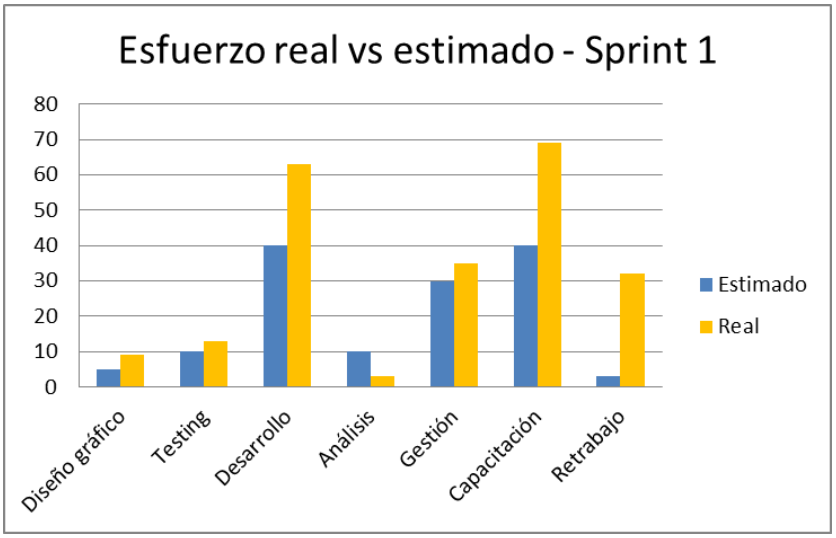


Ilustración 9-5 Esfuerzo real vs estimado del Sprint 1

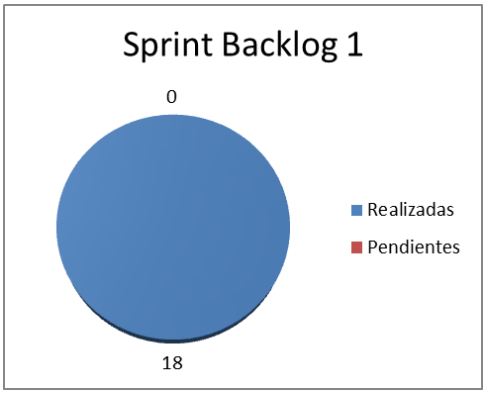
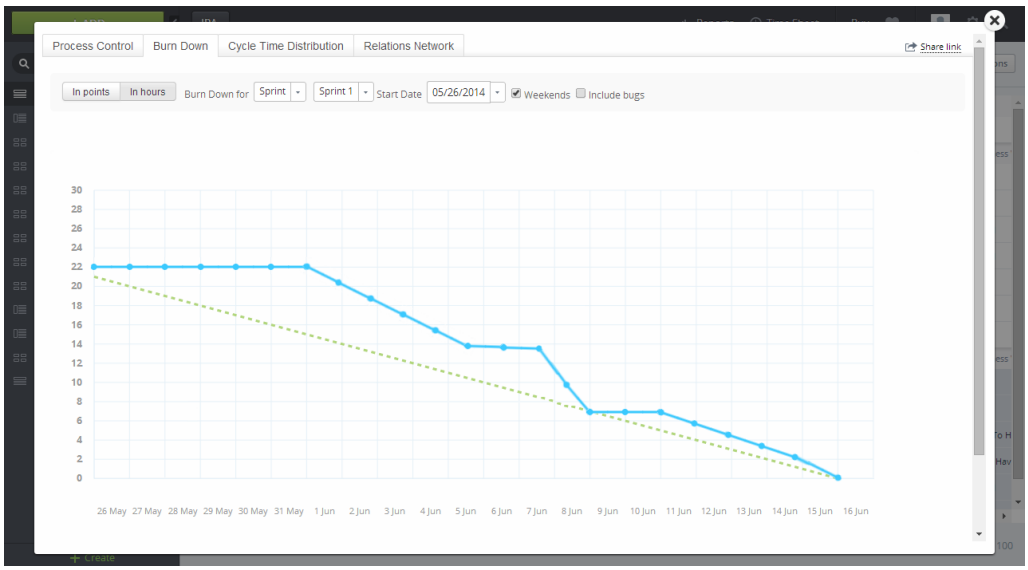


Ilustración 9-6 Sprint Backlog 1



Línea de progreso actual Línea de progreso ideal

Ilustración 9-7 Burndown chart Sprint 1

El principal problema ocurrido en este *Sprint* fue el cambio de tecnología luego de haber avanzado en el mismo. En la reunión de planificación de este *Sprint* el *Product Owner* solicitó utilizar *php* como lenguaje de desarrollo y *Yii* como *framework* del mismo. Sin embargo no todos los interesados del cliente estuvieron de acuerdo, por lo que avanzado el *Sprint* 1, se pidió al equipo, se debía utilizar *Python* con *Django*.

Esta situación provocó muchas horas de retrabajo, ya que se había instalado y configurado el ambiente de trabajo para la tecnología anteriormente definida, sumado a la investigación que se había realizado, ya que no todos los miembros del equipo dominaban la misma.

En la reunión de revisión se acordó con el *Product Owner* que los cambios de decisiones en medio de los *Sprint*s retrasaban el trabajo y tomamos como lección aprendida, en el caso de que volviera a ocurrir, no replanificar durante el *Sprint* en curso, sino darlo de baja y comenzar uno nuevamente de cero teniendo en cuenta que el alcance podría verse afectado. Respecto a las funcionalidades desarrolladas, se obtuvo la satisfacción por parte del cliente sin cambios pendientes.

En la reunión retrospectiva, el quipo analizó de qué otra manera se podría haber gestionado la situación ocurrida y en líneas generales estuvo de acuerdo con el trabajo realizado para este *Sprint*.

9.4.3 Sprint 2

Inicio: 16/06/2014 - Cierre: 06/07/2014

Sprint Backlog:

- Priorización de *User Stories*
- Control Riesgos
- Construcción de datos de prueba para la base de datos
- Medición de métricas de calidad
- Gestión de proyecto
- Capacitación *Django*
- Decisiones de arquitectura
- Realización de presentación Primera Revisión ORTsf
- Construcción *User Story* 324 - Búsquedas simples de datos
- Construcción *User Story* 257 - Búsquedas combinadas de datos - 10 (Tareas sin finalizar que quedaron pendientes para el siguiente *Sprint*)
- Construcción *User Story* 251 - Diseño gráfico de resultados
- Pruebas unitarias

Métricas generales

Sprint 2		
Métrica	**Valor de referencia**	**Resultado**
Esfuerzo estimado (horas/hombre)	Horas definidas para realizar las tareas	200
Esfuerzo real (horas/hombre)	Esfuerzo estimado	214
Cantidad de tareas realizadas	Tareas definidas en la iteración	24
Cantidad de tareas pendientes	No mayor a 3	2
Porcentaje de cobertura de pruebas	100%	95%
Porcentaje de bugs solucionados	90% de los encontrados	18 (78%)
Porcentaje de bugs pendientes	10% de los encontrados	5 (22%)

Tabla 9-12 Métricas generales Sprint 2

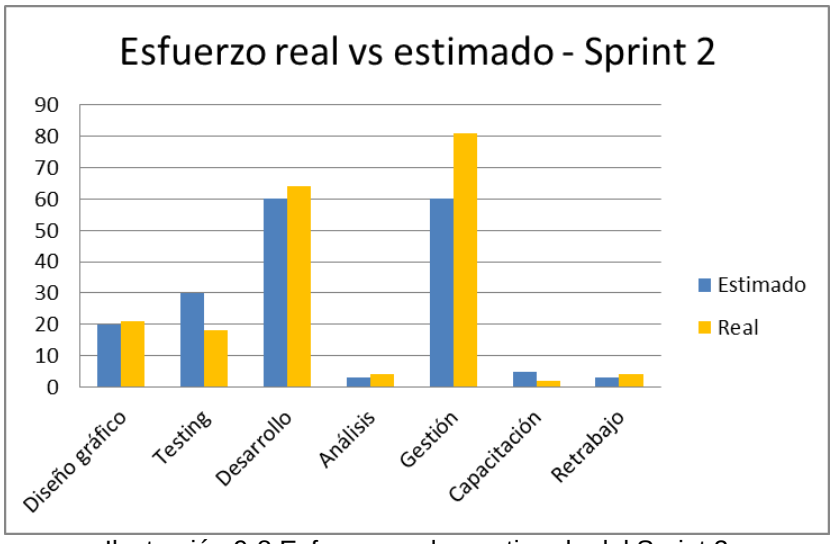


Ilustración 9-8 Esfuerzo real vs estimado del Sprint 2

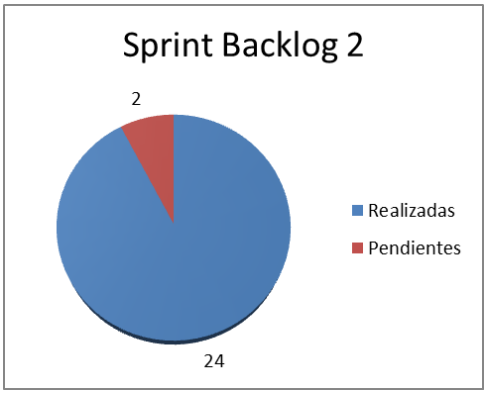


Ilustración 9-9 Sprint Backlog 2

Ilustración 9-10 Burndown chart Sprint 2

En la reunión retrospectiva se destacó la buena estimación general del esfuerzo dedicado al *Sprint*, sin embargo no se tuvo en cuenta la presentación de la primera revisión académica que insumió tiempos de gestión para prepararla y ensayarla pensando en una versión inicial de la presentación final del proyecto.

Respecto a los pendientes, se esperaba tener mayor cantidad de *bugs* solucionados pero el desarrollo de los requerimientos consumió el tiempo restante para ello.

El principal problema surgido en el *Sprint* fue la estimación de los *Story Points* de las *User Stories* a realizar, donde si bien se tenía de base la estimación pasada, las tareas eran muy distintas y la complejidad aumentaba exponencialmente, por lo que el equipo encontró dificultades en ponerse de acuerdo a la hora de asignar el esfuerzo de desarrollo.

Para intentar conseguir mayor exactitud se consultó opiniones y experiencias de otros programadores y del tutor del proyecto.

9.4.4 Sprint 3

Inicio: 07/07/2014 - Cierre: 27/07/2014

Se comenzó con la finalización de las tareas pendientes que bloqueaban el desarrollo de otras del *Sprint* y el cierre de los bugs del *Sprint* anterior.

Sprint Backlog:

- Estimación de las *User Stories*
- Monitoreo y seguimientos de riesgos
- Construcción de datos de prueba para la base de datos
- Medición de métricas de calidad
- Actividades de gestión del proyecto
- Finalización *User Story* 257 - Búsquedas combinadas de datos
- Construcción *User Story* 262 - Visualizar los resultados de búsquedas en gráficas (tareas pendientes para el siguiente *Sprint*)
- Construcción *User Story* 261 - Reportes predefinidos (tareas pendientes para el siguiente *Sprint*)
- Construcción *User Story* 256 - Favoritos
- Construcción *User Story* 250 - Reiniciar búsquedas
- Ejecución de pruebas

Métricas generales

***Sprint* 3**		
Métrica	**Valor de referencia**	**Resultado**
Esfuerzo estimado (horas/hombre)	Horas definidas para realizar las tareas	200
Esfuerzo real (horas/hombre)	Esfuerzo estimado	215
Cantidad de tareas realizadas	Tareas definidas en la iteración	30
Cantidad de tareas pendientes	No mayor a 3	4
Porcentaje de cobertura de pruebas	100%	95%
Porcentaje de bugs solucionados	80% de los encontrados	28 (72%)
Porcentaje de bugs pendientes	20% de los encontrados	11 (28%)

Tabla 9-13 Métricas generales Sprint 3

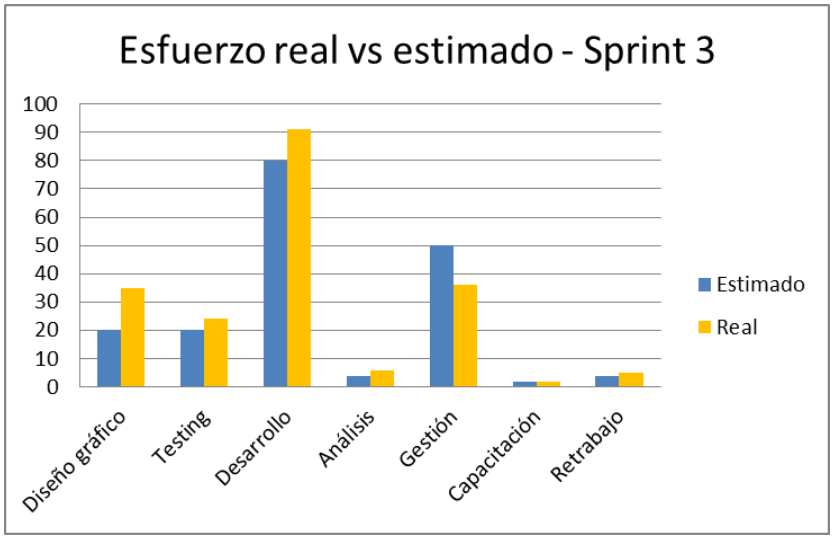


Ilustración 9-11 Esfuerzo real vs estimado del Sprint 3

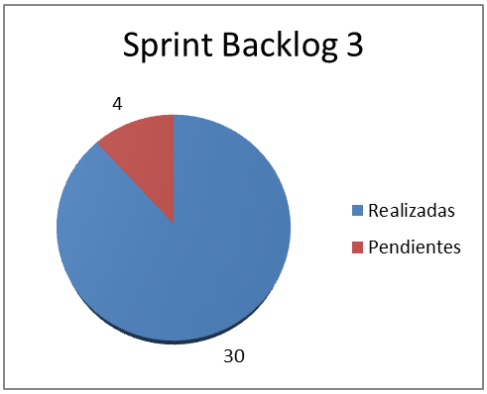


Ilustración 9-12 Sprint Backlog 3

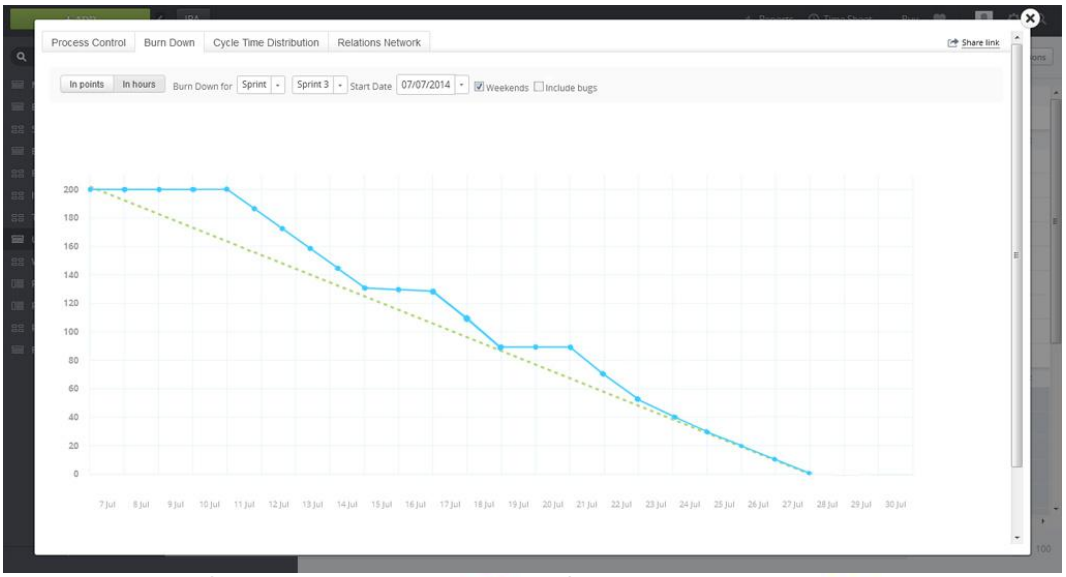


Ilustración 9-13 Burndown chart Sprint 3

El mayor desafío del *Sprint* resultó ser la visualización de gráficas del panel que generó los mayores pendientes para el siguiente *Sprint*.

Las horas reales fueron superiores a las estimadas, sobre todo en las tareas de desarrollo y diseño gráfico, pero disminuyendo en las tareas de gestión.

En la reunión retrospectiva, se destacó como positivo el ritmo logrado para el desarrollo de las funcionalidades y la sinergia conseguida por el equipo; y como negativo las horas insuficientes para cubrir el desarrollo de los bugs encontrados.

Respecto a este último punto se decidió reservar horas exclusivas para ello en el *Sprint* 5, debido a que por asuntos personales y laborales no se pudo dedicar horas extras a las planificadas, y se observó una tendencia a repetirse la situación en el siguiente *Sprint*.

El principal problema fue presentado por el gestor de versiones, donde no quedaron bien realizados la integración del código realizado por cada integrante y hubo que realizarlo manualmente entre el equipo, consumiendo más tiempo del previsto.

9.4.5 Sprint 4

Inicio: 28/07/2014 - Cierre: 17/08/2014

Este *Sprint* comenzó con la realización de las tareas que no habían podido terminarse en el *Sprint* anterior.

Sprint Backlog:

- Estimación de las *User Stories*
- Monitoreo y seguimientos de riesgos
- Construcción de datos de prueba para la base de datos
- Medición de métricas de calidad
- Actividades de gestión del proyecto
- Construcción *User Story* 339 - Desarrollar Notas para el panel
- Construcción *User Story* 338 - Parametrizar alertas
- Construcción *User Story* 260 - Desarrollar Alertas
- Construcción *User Story* 259 - Desarrollar Comparativas
- Construcción *User Story* 253 - Exportar a CSV
- Construcción *User Story* 252 - Exportar a PDF (tareas pendientes para el siguiente *Sprint*)
- Ejecución de pruebas

Métricas generales

***Sprint* 4**		
Métrica	**Valor de referencia**	**Resultado**
Esfuerzo estimado (horas/hombre)	Horas definidas para realizar las tareas	220
Esfuerzo real (horas/hombre)	Esfuerzo estimado	220
Cantidad de tareas realizadas	Tareas definidas en la iteración	36
Cantidad de tareas pendientes	No mayor a 3	2
Porcentaje de cobertura de pruebas	100%	95%
Porcentaje de bugs solucionados	80% de los encontrados	16 (67%)
Porcentaje de bugs pendientes	20% de los encontrados	8 (33%)

Tabla 9-14 Métricas generales Sprint 4

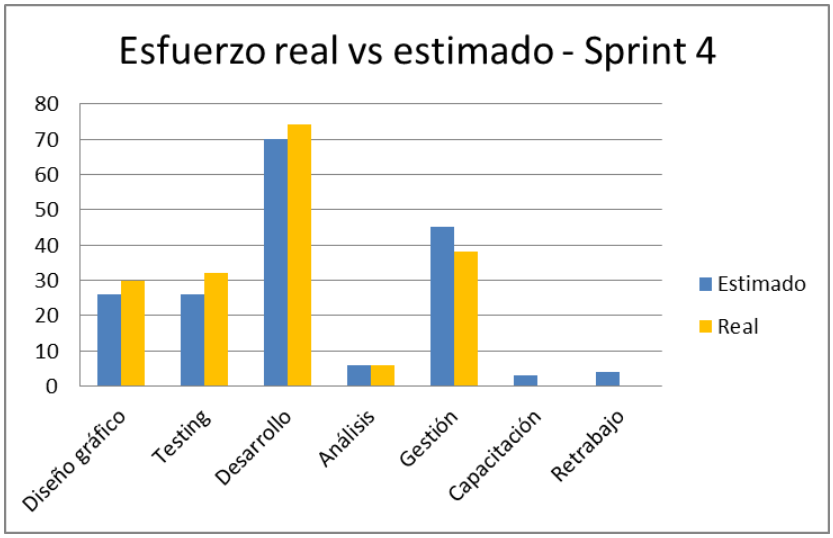


Ilustración 9-14 Esfuerzo real vs estimado del Sprint 4

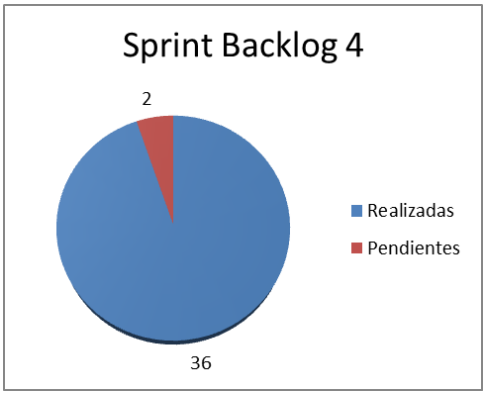
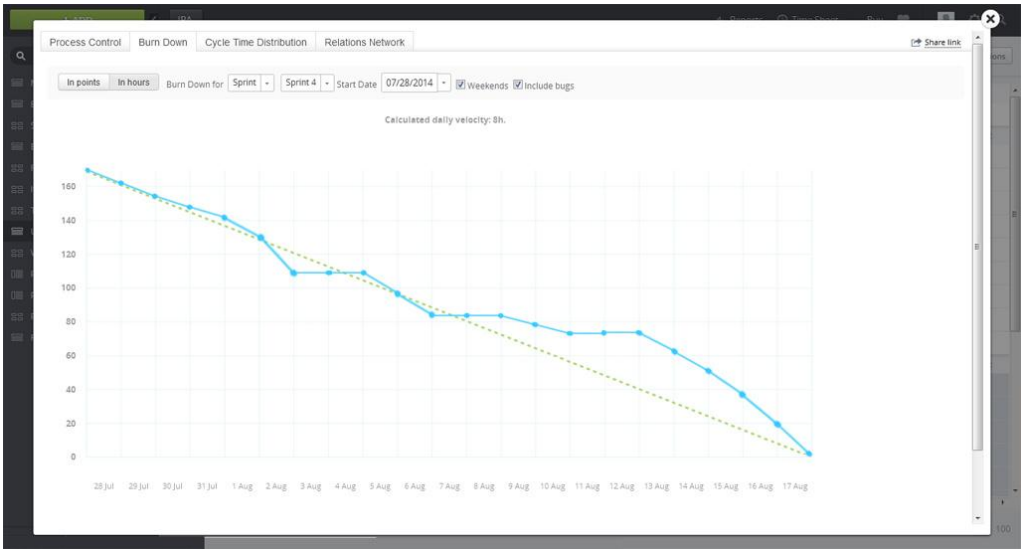


Ilustración 9-15 Sprint Backlog 4



Línea de progreso actual Línea de progreso ideal

Ilustración 9-16 Burndown chart Sprint 4

En el *Sprint* 4 las horas reales totales coincidieron con las estimadas, sin embargo fueron distribuidas diferentes a lo pensado para cada categoría, disminuyendo en las tareas de gestión y sin registrarse tiempo de capacitación y retrabajo.
Respecto a las tareas pendientes quedaron solamente 2 para el siguiente *Sprint*, dentro de nuestro marco de tolerancia.

En la reunión retrospectiva, se discutió sobre la cobertura de pruebas del software que desde el inicio del proyecto se había fijado la meta de llegar al 100% y en las últimas iteraciones no se había podido lograr debido a las múltiples casuísticas dadas por la combinación de los datos de búsquedas. Este tema fue hablado con el *Product Owner*, quien consideró más que aceptable las pruebas realizadas por el equipo y no lo creyó un problema a resolver.

En la puesta a producción, con datos reales en la base de datos del cliente se harían los ajustes que fueran necesarios en el caso de alguna validación que no fuera contemplada a lo largo del proyecto.

En la reunión de revisión, el *Product Owner* quedó conforme con los requerimientos terminados y demostró su agrado general con lo desarrollado hasta el momento. Esto fue en base a la buena usabilidad que visualizó en la herramienta y las funcionalidades sugeridas.

9.4.6 Sprint 5

Inicio: 18/08/2014 - Cierre: 07/09/2014

Luego de la reunión de planificación, donde el *Product Owner* hizo algunas modificaciones sobre las funcionalidades restantes del *Product Backlog*; se desarrollaron los bugs y tareas pendientes del *Sprint* 4 y se asignaron al *Sprint Backlog* todas las *User Stories* remanentes.

Sprint Backlog:
- Estimación de las *User Stories*
- Monitoreo y seguimientos de riesgos
- Construcción de datos de prueba para la base de datos
- Medición de métricas de calidad
- Actividades de gestión del proyecto
- Construcción *User Story* 347 - Consultas realizadas frecuentemente
- Construcción *User Story* 349 - Visita guiada del sistema
- Construcción *User Story* 255 - Ordenar resultados
- Construcción *User Story* 231 - Ayuda
- Construcción *User Story* 230 - Pantalla inicial
- Ejecución de pruebas

Métricas generales

***Sprint* 5**		
Métrica	**Valor de referencia**	**Resultado**
Esfuerzo estimado (horas/hombre)	Horas definidas para realizar las tareas	200
Esfuerzo real (horas/hombre)	Esfuerzo estimado	214
Cantidad de tareas realizadas	Tareas definidas en la iteración	32
Cantidad de tareas pendientes	No mayor a 3	0
Porcentaje de cobertura de pruebas	100%	95%
Porcentaje de bugs solucionados	100% de los encontrados	15 (100%)
Porcentaje de bugs pendientes	0% de los encontrados	0 (0%)

Tabla 9-15 Métricas generales Sprint 5

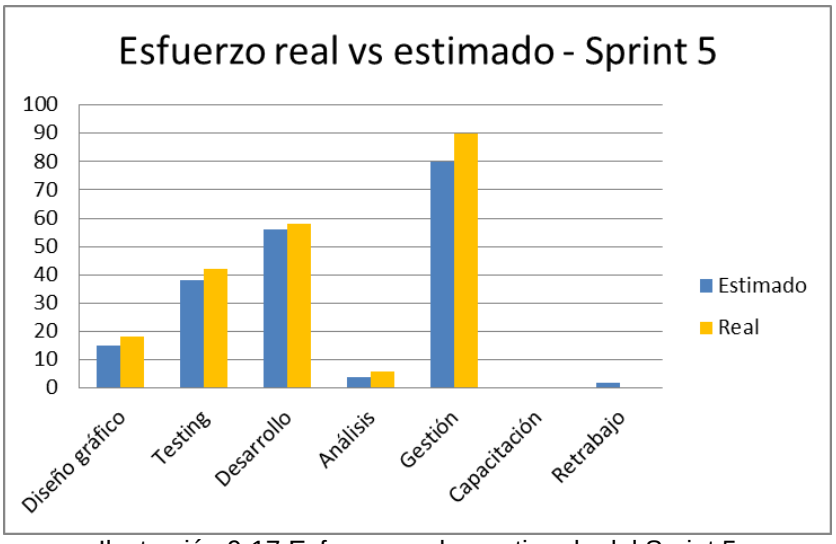


Ilustración 9-17 Esfuerzo real vs estimado del Sprint 5

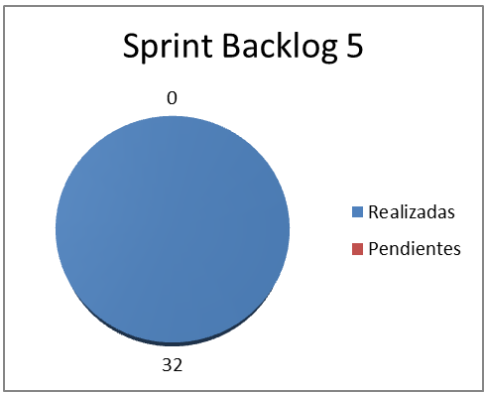
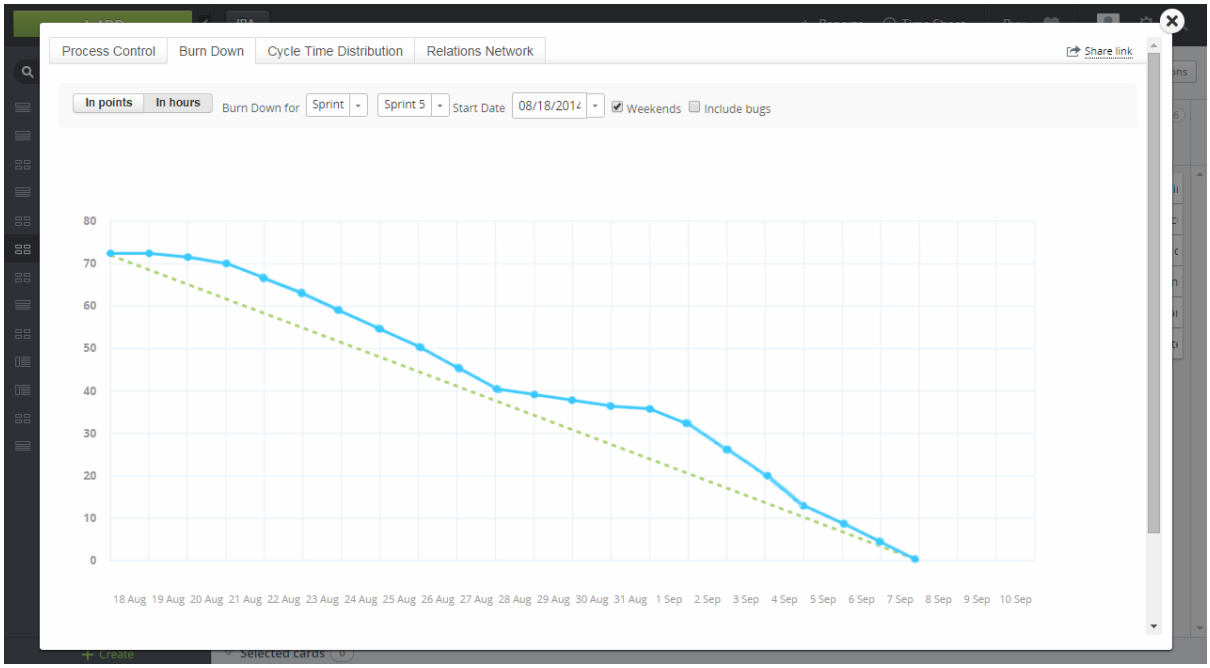


Ilustración 9-18 Sprint Backlog 5



Línea de progreso actual Línea de progreso ideal

Ilustración 9-19 Burndown chart Sprint 5

En la reunión retrospectiva, se consideró un detrimento en la comunicación de todo el equipo debido a que las tareas de este *Sprint* fueron realizadas de manera muy individual y se optimizaron las horas de trabajo al máximo. De todos modos, el equipo siempre tomó todas las decisiones en conjunto, se llevaron a cabo las ceremonias de *Scrum* y se cumplió con las tareas y plazos definidos.

9.5 Conclusiones generales

El *Sprint* 3 fue la mayor bisagra del proyecto, hasta entonces no se tenía información suficiente sobre el trabajo del equipo, pero en esta iteración ya se contaba con cierta experiencia que permitió proyectar el comportamiento futuro del proyecto y notar ciertas tendencias que ayudaron a anticiparse a planificar las soluciones para determinadas situaciones.

Al final de cada *Sprint* el cliente demostró su aceptación con lo desarrollado hasta el momento, indicando que la comunicación constante fue uno de los pilares importantes en el proyecto.

El *Sprint* 0 registró 209 horas de esfuerzo que no fueron estimadas inicialmente, debido a que el equipo no contaba con datos que pudiera utilizar para ello.

El proyecto en total, insumió 1296 horas con una desviación aproximada de un 15% mayor al esfuerzo estimado para el mismo; y su velocidad se registró en un promedio de 41 *Story Points* por *Sprint*.

El panel de control desarrollado puede verse a través de las impresiones de pantalla adjuntas en el Anexo 13 – Imágenes de iPanel Educativo.

9.6 Resultados finales

Las actividades realizadas en el proyecto se agruparon en categorías para monitorear el esfuerzo del trabajo. A continuación se muestran las mismas con sus respectivos resultados expresados en porcentaje al último *Sprint* del el proyecto.

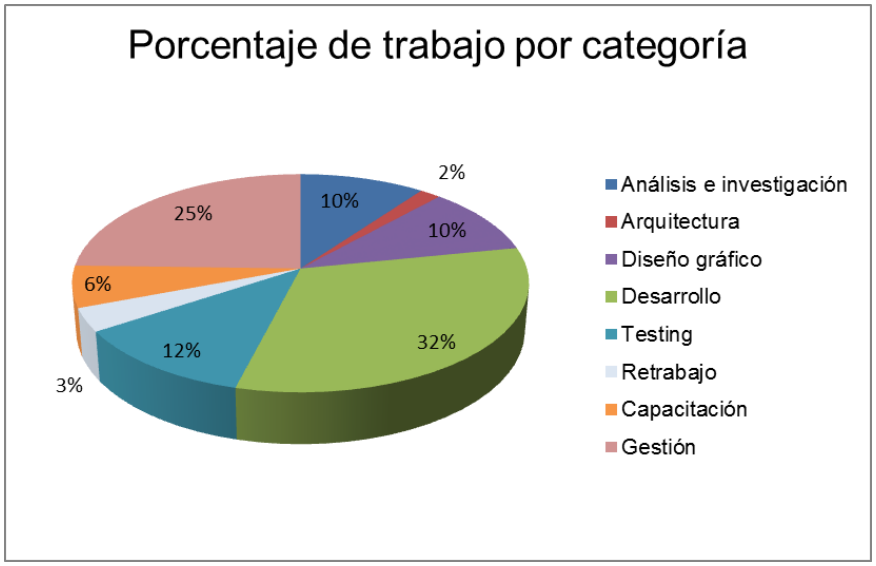


Ilustración 9-20 Porcentaje de distribución de trabajo por categoría

A continuación se observan las horas totales estimadas para cada categoría y las horas totales reales trabajadas.

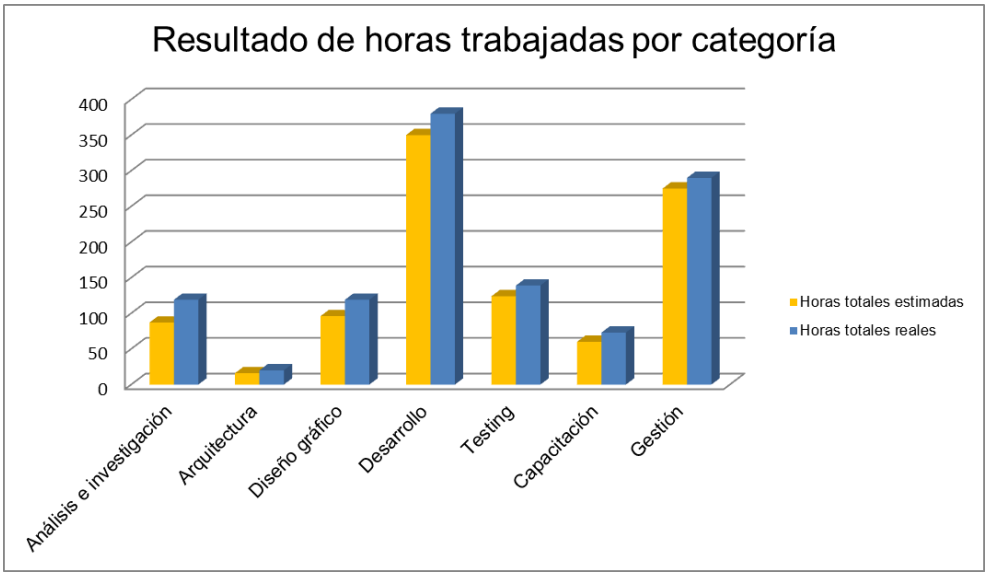


Ilustración 9-21 Resultado de horas trabajadas por categoría

En la siguiente ilustración se observan las horas totales trabajadas por cada *Sprint* del proyecto.

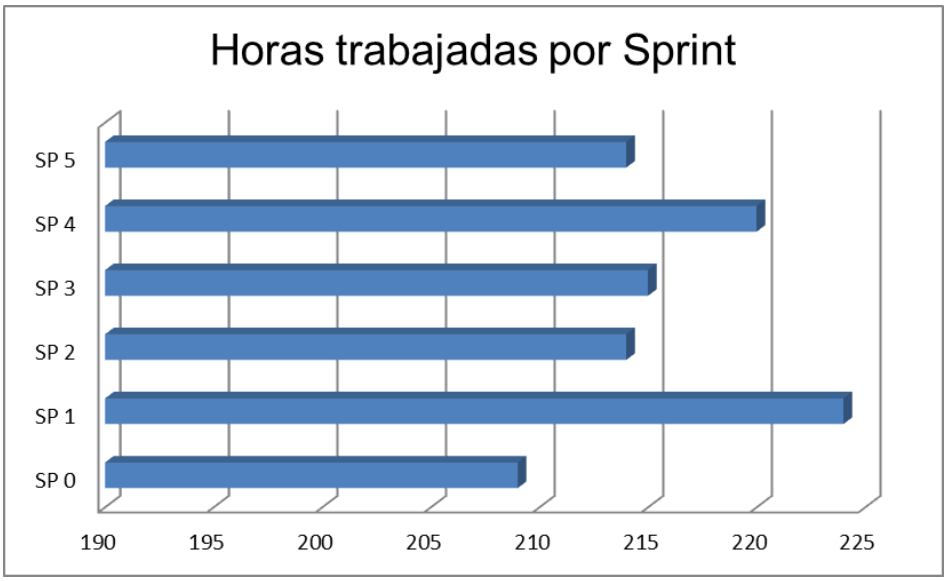


Ilustración 9-22 Horas trabajadas en total por Sprint

Se visualizan la cantidad de bugs reportados por testing en cada *Sprint* detallando la cantidad de bugs pendientes, los cuales se planificaron resolver en el siguiente *Sprint*.

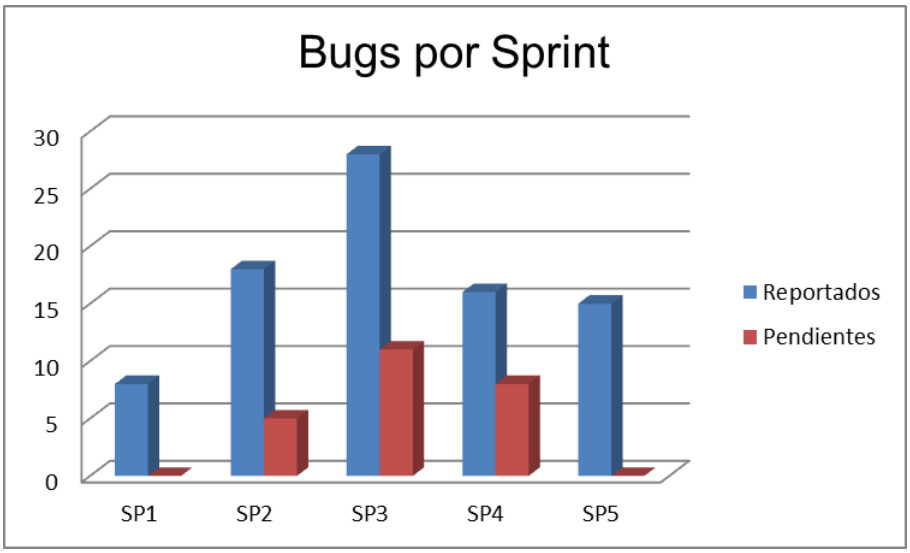


Ilustración 9-23 Cantidad de bugs por Sprint

En la siguiente imagen se muestra la cantidad de tareas realizadas por cada *Sprint* y la cantidad que permanecieron pendientes de realizar para la iteración siguiente del proyecto.

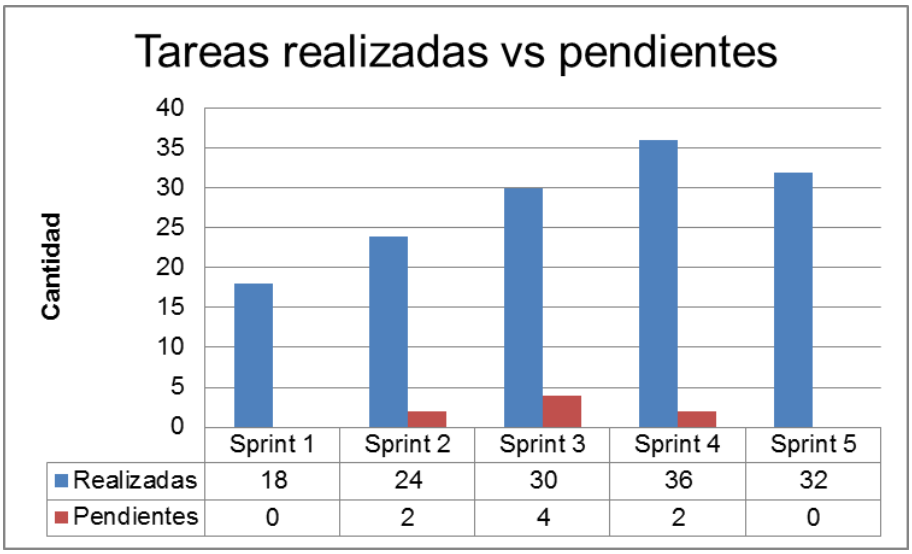


Ilustración 9-24 Tareas realizadas vs. pendientes

La siguiente ilustración presenta la velocidad alcanzada durante el proyecto expresada en *Story Points*.

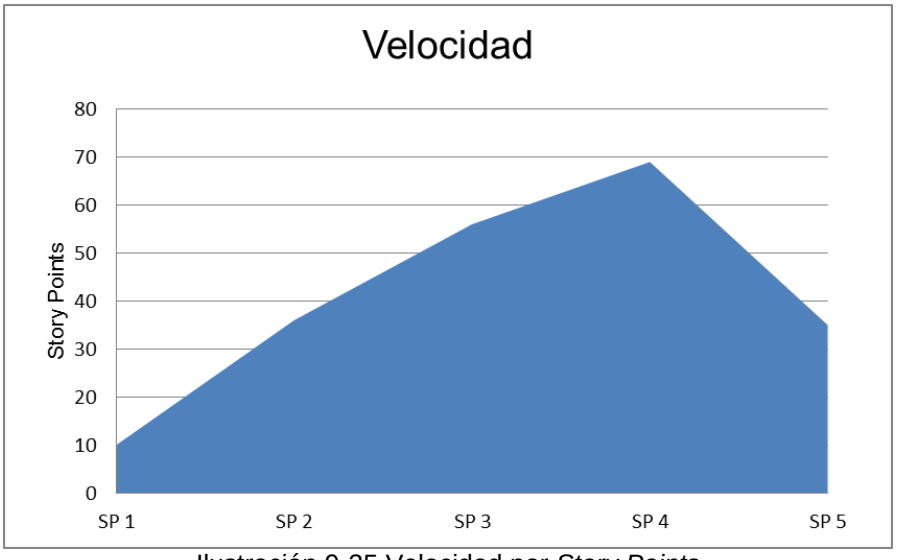


Ilustración 9-25 Velocidad por *Story Points*

10. Lecciones aprendidas

El equipo estuvo de acuerdo en reconocer que el proyecto contribuyó a la formación de todos los integrantes tanto desde un punto de vista académico, como tecnológico y de negocio, que fueron brindados por esta experiencia.

El proyecto permitió a sus integrantes aplicar los conocimientos adquiridos a lo largo de la carrera, recurriendo muchas veces a conceptos aprendidos previamente para ponerlos en práctica. La gestión integral del proyecto contó con importantes desafíos que supieron ser trabajados en conjunto y con un resultado satisfactorio para el equipo.

Entre las tantas lecciones aprendidas de estos últimos meses destacamos el relacionamiento entre los miembros del equipo, que siempre se mantuvo cordial y basado en el respeto, buscando el consenso en todas las decisiones tomadas y siendo esta una clave para el buen desarrollo del proyecto.

Se consideró muy positivo el hecho de realizar parte de la documentación del proyecto en el transcurso del mismo, ya que permitió recordar justificar todas las decisiones tomadas y terminar de elaborar el documento final con tiempo antes de la entrega.

Respecto a la metodología, realizar actividades de SCM, SQA, gestión de riesgos y comunicaciones resultaron beneficiosas dentro del marco de trabajo ágil tanto para alinear y seguir las mismas pautas dentro del equipo como para tener un mayor control sobre el proyecto que también se vio reflejado en el relacionamiento con el cliente.

Por último y no menos importante, la estimación de esfuerzo de las actividades fue una clara dificultad al comienzo del proyecto que las reuniones retrospectivas permitieron afinar y disminuir el margen de error en instancias de discusión donde el equipo compartió sus opiniones sobre el tema y mejoró notoriamente la tarea en base a la experiencia.

11. Conclusiones

Se cumplieron satisfactoriamente los principales objetivos del equipo para este proyecto, como ser la adquisición de experiencia profesional, el aprendizaje de nuevas tecnologías y metodologías de trabajo, la práctica de un buen clima de trabajo y el intercambio de conocimiento entre los integrantes.

EDU Editorial manifestó su conformidad (Ver Anexo 14 – Carta de conformidad) ante el producto desarrollado y la experiencia de haber realizado un proyecto en conjunto con estudiantes que están finalizando su carrera. Además el cliente comunicó su interés de implementar a la brevedad el panel de control para comenzar a analizar sus datos, transformarlos en información y poder tomar decisiones asesorados por el sistema.

12. Bibliografía

[1] Laboratorio ORT Software Factory, Setiembre 2014. [En línea]. Disponible: http://fi.ort.edu.uy/innovaportal/v/1545/5/fi.ort.front/ort_software_factory_ortsf.html

[2] ¿Qué es EDU Editorial?, Abril 2014. [En línea]. Disponible: http://editorialedu.com:8080/wordpress/que-es-edu-editorial

[3] Cuasimodo, Abril 2014. [En línea]. Disponible: http://editorialedu.com:8080/wordpress/features/cuasimodo

[4] J. Nielsen, Enero 1995. [En línea]. Disponible: http://www.nngroup.com/articles/ten-usability-heuristics/

[5] L. Bass, P. Clements y R. Kazman, Software architecture in practice,Massachusetts: Addison-Wesley, 1998. - Third edition

[6] E. Gamma, R. Johnson, J. Vlissides y R. Helm, Design patterns, Addison-Wesley,Ed., Massachusets, 1994.

[7] Artículo de Philippe Kruchtens “Architectual Blueprints – the “4+1” View Model of Software Architecture”, 2014. [En línea]. Disponible: http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

[8] Django Project, 2014. [En línea]. Disponible: https://www.djangoproject.com/

[9] Artículo de Amalia I. Álvarez, “Una experiencia de medición de los costos relativos a la calidad en la producción de software”, 2014. [En línea]. Disponible: http://aulas.ort.edu.uy/mod/resource/view.php?id=57007

[10] Beneficios de Scrum, 2014 [En línea]. Disponible: http://www.proyectosagiles.org/beneficios-de-Scrum

[11] PlanningPoker, 2014 [En línea]. Disponible: http://www.planningpoker.com

[12] WhyScrum?, 2014 [En línea]. Disponible: https://www.Scrumalliance.org/why-Scrum

[13] La Guía de Scrum™, 2013 [En línea]. Disponible: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-ES.pdf

[14] Guía de los Fundamentos de la Dirección de Proyectos, PMBOK® Guide - FourthEdition, 2009

13. Anexos

13.1 Anexo 1 - Investigación de posibles herramientas para la solución

Se analizaron dos herramientas existentes que contaban con las funcionalidades requeridas por el cliente. La elección fue en base a la información a la que se podía acceder mediante expertos.

A continuación se describen brevemente cada una de ellas y una tabla que compara las distintas características de ambas.

QlikView: es un tipo de software de Business Intelligence. Según se describe en su sitio web: “Permite dejar de adivinar y empezar a conocer realmente cómo tomar decisiones mejores y más rápidas.”[7]

Pentaho Community: es un esfuerzo continuo por la comunidad de código abierto para proporcionar a las organizaciones las mejores soluciones para las necesidades de Business Intelligence (BI). El Proyecto BI Pentaho abarca las siguientes áreas de aplicación principales: informes, análisis, paneles de control, minería de datos, entre otras.[8]

Característica / Herramienta	QlikView	Pentaho Community
Usabilidad*	Fortaleza	Debilidad
Licencia Desarrollar funcionalidades	USD 1000 a USD 2000	USD 0.-
Licencia Desarrollar interfaz	USD 1000 a USD 2000	USD 0.-
Lenguaje de programación	java	java
Período de implementación**	3 meses o menos	3 meses o más
Tiempo de aprendizaje usuario final	1 semana	2 semanas
Tiempo de aprendizaje desarrollador de funcionalidades	1 mes	1 mes a 2 meses

* La descripción de la usabilidad se realizó en base al Cuadrante Mágico de Gartner[9] donde se indica que en este primer semestre *QlikView* se ha centrado en

7 http://qlikview.com.uy/)

8 http://community.pentaho.com/faq/general.php

9 http://www.gartner.com/technology/reprints.do?id=1-1QLGACN&ct=140210&st=sb

satisfacer a los usuarios de negocio de las empresas las necesidades detectadas de mejorar la facilidad de uso de la herramienta. Además comenta que para el segundo semestre esta empresa tiene planificado hacer un completo rediseño de su visualización, para que sea más fácil para los usuarios descubrir y compartir nuevos conocimientos.

En tanto para *Pentaho* se indica que un porcentaje más alto que el promedio de los usuarios finales y desarrolladores manifestó su disconformidad sobre la facilidad de uso de esta herramienta.

** Tiempo de implementación aproximado aplicado a grandes negocios.

13.2 Anexo 2 - Listado de *User Stories*

Introducción
A continuación se detalla el *Product Backlog* de la aplicación y se describen los *User Stories* correspondientes.

Product Backlog
La herramienta *TargetProcess* fue la utilizada para la gestión del proyecto bajo el marco de trabajo *Scrum*. La misma numeraba las *User Stories* de forma automática y según un criterio interno, siendo esta la causa de que la identificación de cada una de ellas fuera salteada y comenzara en el número 229.

USER STORIES	ID
Como usuario quiero poder logearme en la aplicación para utilizarla	229
Como usuario registrado quiero poder ver las gráficas en la pantalla inicial para tener una vista primaria de la situación actual	230
Como usuario registrado quiero poder cerrar mi sesión para finalizar el uso de la aplicación	232
Como usuario registrado quiero poder visualizar los resultados en tablas para ver los datos	251
Como usuario registrado quiero poder combinar filtros de búsqueda para obtener resultados filtrados por más de una variable	257
Como usuario registrado quiero poder buscar información para analizar los resultados	324
Como usuario registrado quiero poder navegar entre las diferentes secciones del sistema	326
Como usuario registrado quiero poder guardar búsquedas favoritas para poder acceder a ellas rápidamente luego	256
Como usuario registrado quiero poder comparar resultados obtenidos de las búsquedas analizar la evolución de los ejercicios	259
Como usuario registrado quiero poder generar reportes predefinidos sobre los resultados para ahorrar tiempos de búsqueda	261
Como usuario registrado quiero poder visualizar resultados de búsqueda en gráficas para contar con otra forma de despliegue de datos	262
Como usuario registrado quiero acceder a una ayuda sobre el uso de la aplicación para evacuar dudas	231
Como usuario registrado quiero poder reiniciar los filtros de búsqueda para comenzar una nueva búsqueda	250
Como usuario registrado quiero poder exportar los resultados obtenidos por consulta a formato PDF para poder transferirlos	252
Como usuario registrado quiero poder exportar los resultados obtenidos por consulta a formato CSV para poder transferirlos	253
Como usuario registrado quiero poder ordenar los resultados obtenidos de una consulta para visualizarlos ordenados por determinada variable	255
Como usuario registrado quiero poder contar con alertas sobre los resultados para conocer comportamientos interesantes de los mismos	260
Como usuario registrado quiero poder cambiar los parámetros configurables de las alertas del panel para visualizar las alertas que me interesan	338
Como usuario registrado quiero poder agregar notas mientras utilizo el panel para utilizar como ayuda memoria	339
Como usuario registrado quiero visualizar y poder acceder a las cinco consultas realizadas mas frecuentemente en el ultimo mes	347
Como usuario registrado quiero acceder a una visita guiada para observar las principales funcionalidades del sistema	349

Ilustración 13-1 *Product Backlog* de la aplicación

Listado de Users Stories con la descripción de sus condiciones de satisfacción.

ID	Descripción	Condiciones de satisfacción
229	Como usuario quiero poder ingresar al sistema con mi usuario y contraseña	- en caso de error que me avise y si olvide mi contraseña tener un medio de comunicación con un administrador para que me pueda resolver el inconveniente - al presionar “Cerrar sesión” el sistema me debe enviar a la página de acceso al sistema y si presiono el botón “Atrás” en el navegador que me solicite autenticación para poder ingresar nuevamente
230	Como usuario registrado quiero ingresar al sistema y poder visualizar gráficas acerca de la utilización de los ejercicios	- visualizar los ejercicios menos utilizados - visualizar cantidad de ejercicios utilizados por tipo de ejercicio en cada materia - visualizar la cantidad de ejercicios utilizados a lo largo del sistema - visualizar la cantidad de puntos obtenidos por género y edad
231	Como usuario registrado quiero poder acceder a una sección de ayuda con información útil para la utilización del sistema	- poder ver la descripción de las funcionalidades de alertas, consultas y comparativas
232	Como usuario registrado quiero poder cerrar mi sesión para finalizar el uso de la aplicación	- se cierra la sesión correctamente - al intentar volver a iniciar sesión yendo hacia atrás desde el navegador me pida usuario y contraseña nuevamente para poder ingresar
250	Como usuario registrado quiero poder reiniciar los filtros de búsqueda para comenzar una nueva búsqueda	- poder deseleccionar los filtros aplicados en las consultas realizadas - contar con un botón que permita realizar una consulta completamente nueva
251	Como usuario registrado quiero poder visualizar los resultados en tablas para ver los datos	- luego de realizar una consulta poder desplegar los resultados en tablas con filas y columnas
252	Como usuario registrado quiero poder exportar los resultados generados por medio de una búsqueda realizada, a archivos con formato PDF	- visualizar los archivos descargados en formato PDF desde un equipo sin necesidad de tener conexión a internet
253	Como usuario registrado quiero poder exportar los resultados generados por medio de una	- visualizar los archivos descargados en formato CSV desde un equipo sin necesidad de tener conexión a internet

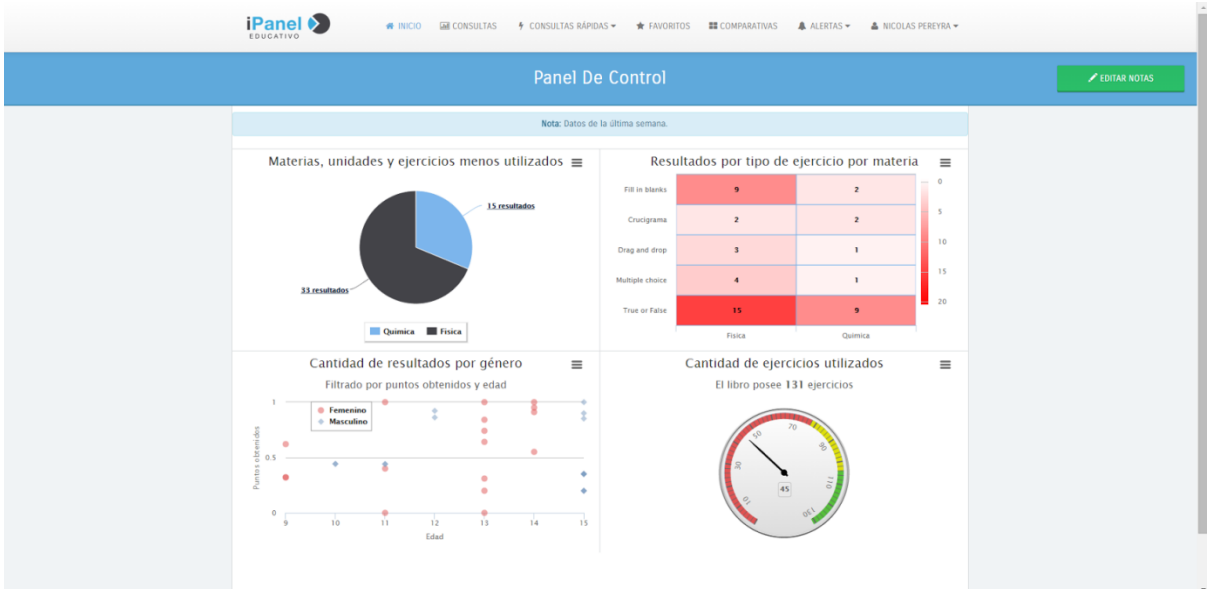
	búsqueda realizada, a archivos con formato CSV	
255	Como usuario quiero que el sistema pueda contar, realizar promedio, ordenar información de menor a mayor y viceversa, que me muestre mínimos y máximos y liste resultados	- que las operaciones de mínimos y máximos, y de contar se visualicen en la utilización de ejercicios y de resultados obtenidos en la página de inicio una vez ingresado al sistema. - que los resultados de las búsquedas se vean en listados
256	Como usuario registrado quiero poder guardar combinaciones de uno o más filtros como mis filtros favoritos en las búsquedas	- al iniciar una nueva sesión poder seleccionar los filtros guardados con anterioridad
257	Como usuario registrado quiero poder realizar consultas sobre los datos almacenados con distintos filtros aplicados.	- poder aplicar filtros en base a: País, Departamento, Escuela, Aula, Usuarios, Nivel, Área de conocimiento, Materias, Capítulos, Tipo de ejercicio, Género, Tiempo de uso, Rango de fechas y Puntaje obtenido - los resultados deben mostrar una gráfica y la posibilidad de verlos en tablas
259	Como usuario registrado quiero poder comparar resultados obtenidos por filtros que seleccione, según intervalos de tiempo que defina, con el objetivo de ver la evolución de los ejercicios	- que se vea en esa sección 4 intervalos de tiempo según los filtros seleccionados
260	Como usuario registrado quiero que el sistema notifique cuando se produzca algún hecho de que la cantidad de ejercicios utilizados promedio o manual es superado	- mantener registros históricos sobre las alertas generadas por el sistema - cuando se cumplan las condiciones de puntajes esperados por debajo de un nivel de tolerancia determinado se emita una alerta automáticamente cada 30 minutos - tener la posibilidad de que se generen alertas de forma manual sin tener que esperar un intervalo de tiempo - poder configurar un tiempo de espera adicional para la generación de alertas automáticas
261	Como usuario registrado quiero poder realizar las consultas por los ejercicios menos utilizados y ejercicios con un puntaje promedio inferior o igual a 0,1 y ejercicios con puntaje promedio mayor o igual a 0,9 rápidamente	- poder realizar una de las consultas en menos de 3 clics
262	Como usuario registrado quiero poder visualizar resultados de	- disponer de varios tipos de gráficas como ser de columnas, circulares y

	búsqueda en gráficas para contar con otra forma de despliegue de datos	lineales
324	Como usuario registrado quiero poder buscar información para analizar los resultados	- en la tabla de resultados poder encontrar un resultado que cumpla con los criterios de búsqueda
326	Como usuario registrado quiero poder navegar entre las diferentes secciones del sistema mediante enlaces	- al hacer clic en los enlaces del menú acceder a una nueva sección
338	Como usuario registrado quiero poder cambiar los parámetros configurables de las alertas del panel para visualizar las alertas que me interesan	- poder acceder a una sección en la aplicación donde se puedan cambiar los parámetros del intervalo de fechas de las consultas rápidas - poder acceder a una sección en la aplicación donde se puedan cambiar los parámetros de las alertas siendo estas fechas desde, hasta, tolerancia, media esperada por ejercicio y media calculada automáticamente.
339	Como usuario registrado quiero poder agregar notas mientras utilizo el panel para utilizar como ayuda memoria sin necesidad de cambiar de sistema	- sin cambiarme de sección del sistema ingresar notas o comentarios y que quede almacenado para poder consultarlo posteriormente
347	Como usuario registrado quiero visualizar y poder acceder a las cinco consultas realizadas más frecuentemente en el último mes	- quiero que el sistema registre las consultas que más utiliza cada usuario - poder utilizar esta funcionalidad en menos de 3 clics
349	Como usuario registrado quiero acceder a una visita guiada para observar las principales funcionalidades del sistema	- quiero que al ingresar al sistema por primera vez, poder seguir mediante señalizaciones en pantalla, una guía de las funcionalidades más importantes de la aplicación - la duración máxima será de hasta 1 minuto - poder repetir la visita guiada al hacer clic sobre algún enlace

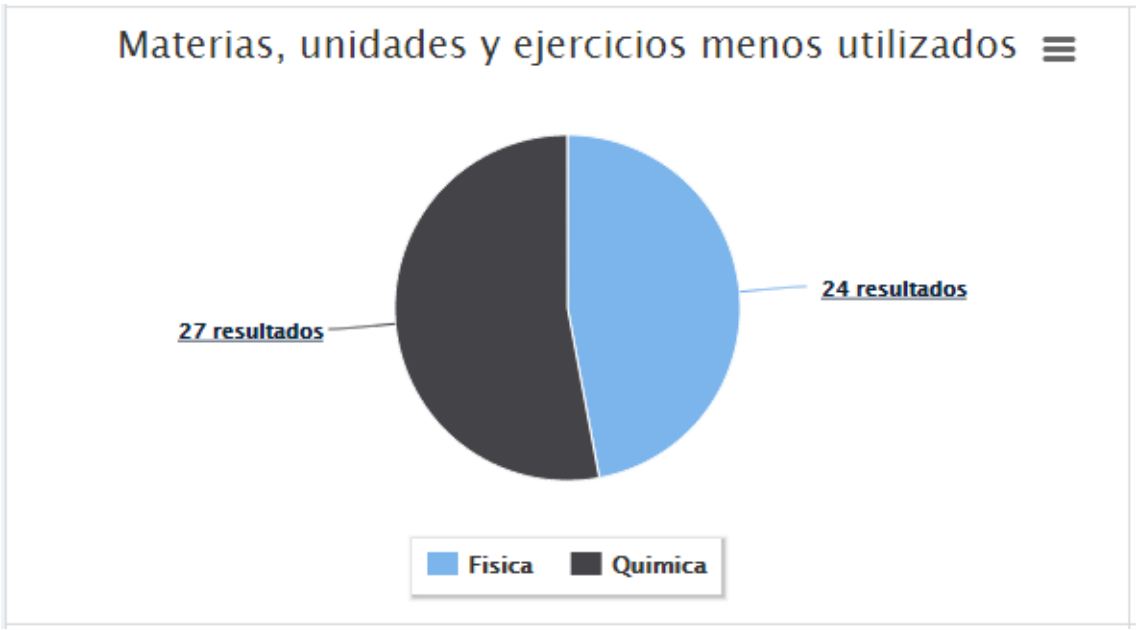
13.3 Anexo 3 – Manual de Usuario

¿Qué se puede ver en el Inicio - Panel de Control?

Se visualizan cuatro cuadrantes donde se aprecian distintos gráficos generados a partir de los resultados obtenidos en la última semana de 7 días.

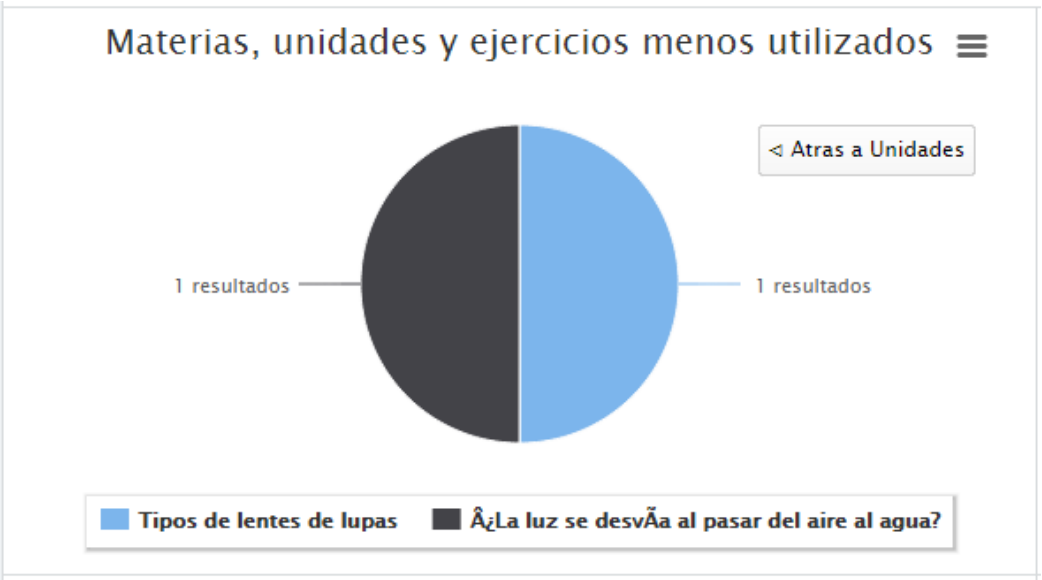


Materias, unidades y ejercicios menos utilizados
Arriba a la izquierda una gráfica circular que indica la cantidad de ejercicios utilizados por Materia.



Al presionar en alguna de las divisiones de la gráfica podremos visualizar la cantidad de ejercicios utilizados por Unidad. Por ejemplo si presionáramos sobre la división celeste que representa a la Materia Física, veríamos una nueva gráfica circular similar a la siguiente:

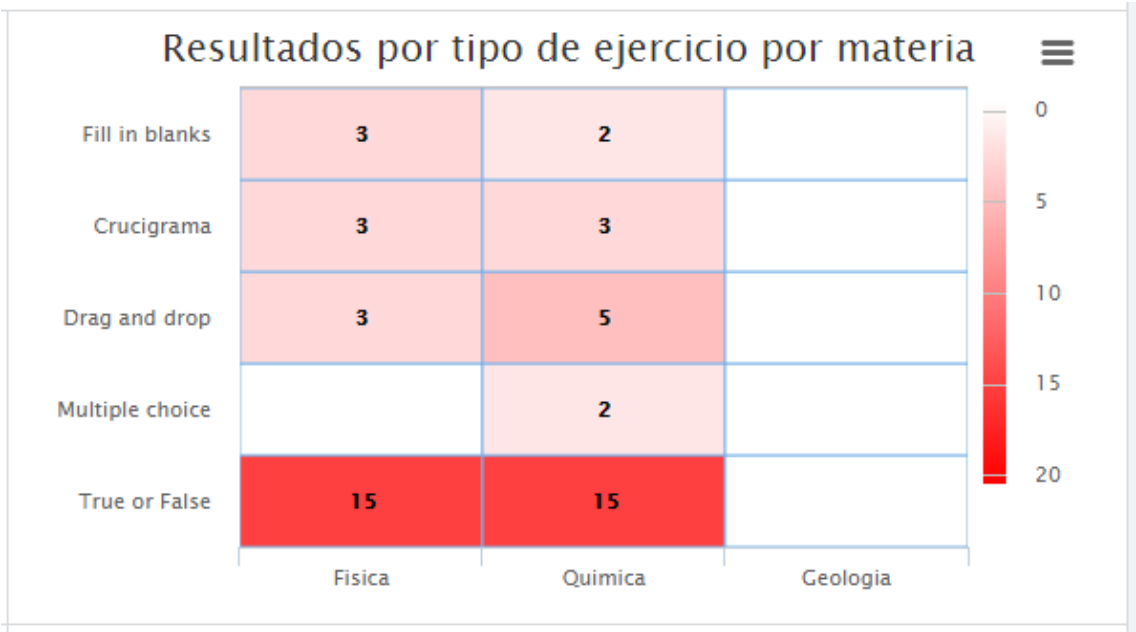
Por último si presionamos sobre la división celeste que representa "La Refracción de la luz", donde se visualiza la cantidad de veces utilizado cada ejercicio.



Resultados por tipo de ejercicio por materia

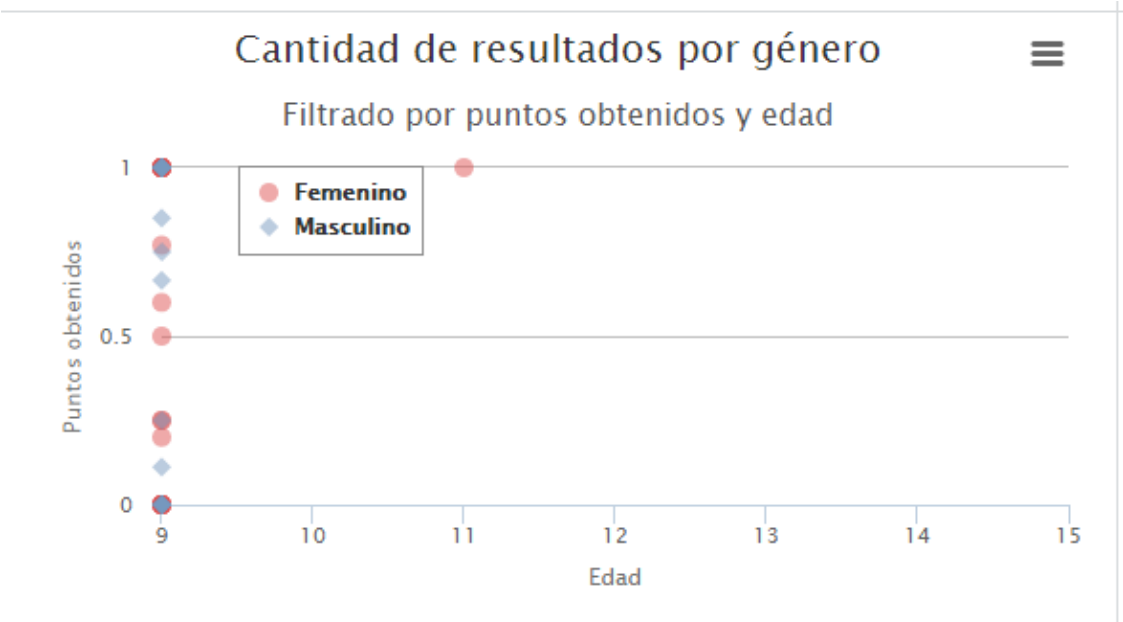
En este cuadrante podemos ver un mapa de calor que refleja la cantidad de ejercicios utilizados según el tipo de ejercicio según las materias que integran el libro.

Esta gráfica indica por ejemplo que del tipo de ejercicio Crucigrama, en la materia Física, la utilizaron 3 personas, en la materia Química 5 y en Geología no fue utilizado.



Cantidad de resultados por género

En el cuadrante ubicado abajo a la izquierda de la pantalla se puede visualizar un diagrama de dispersión el cual indica los puntajes que obtienen los usuarios según su género y detallado por edad.

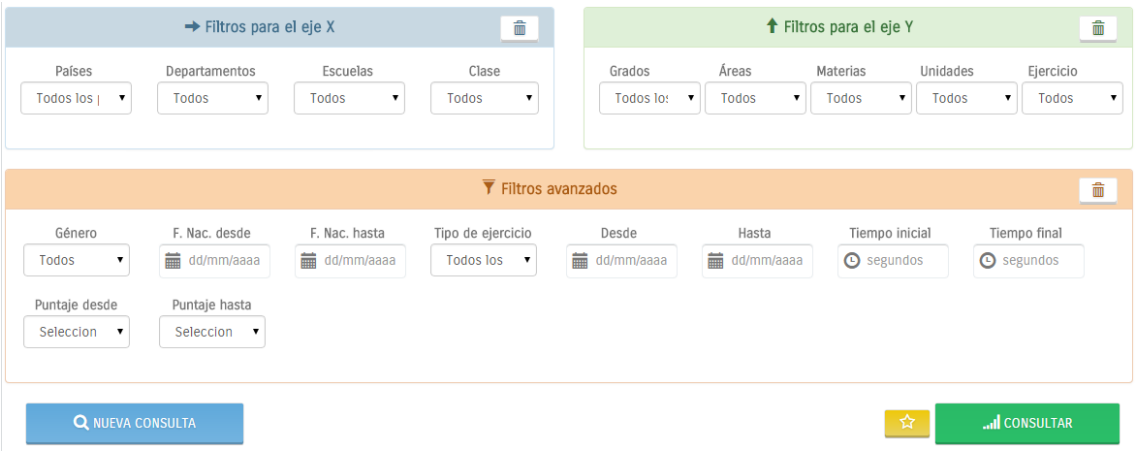


Cantidad de ejercicios utilizados

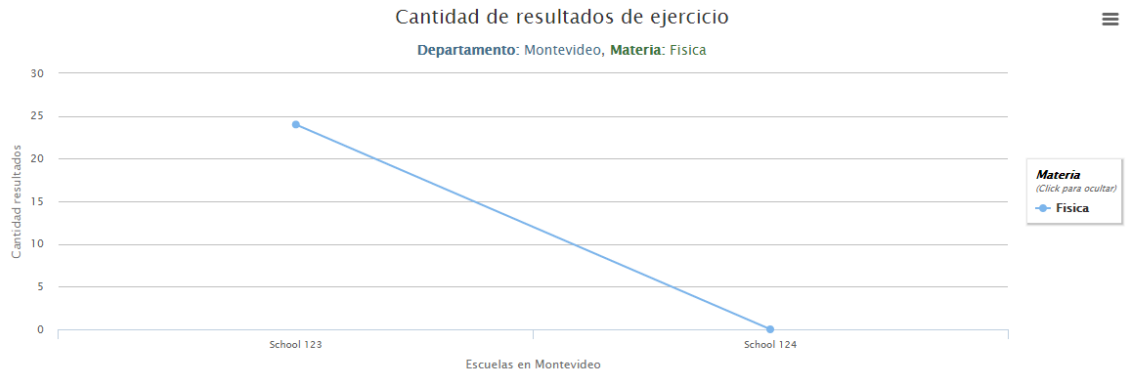
Podremos visualizar un acelerómetro con la cantidad total de ejercicios con que cuenta el libro y la cantidad de ejercicios que fueron utilizados en este último período.

¿Qué puedo hacer en la sección Consultas?

En la sección Consultas se podrán analizar los resultados almacenados hasta el momento mediante la selección de diversos filtros como ser País, Departamento, Materia, Unidad, Género, entre otros.

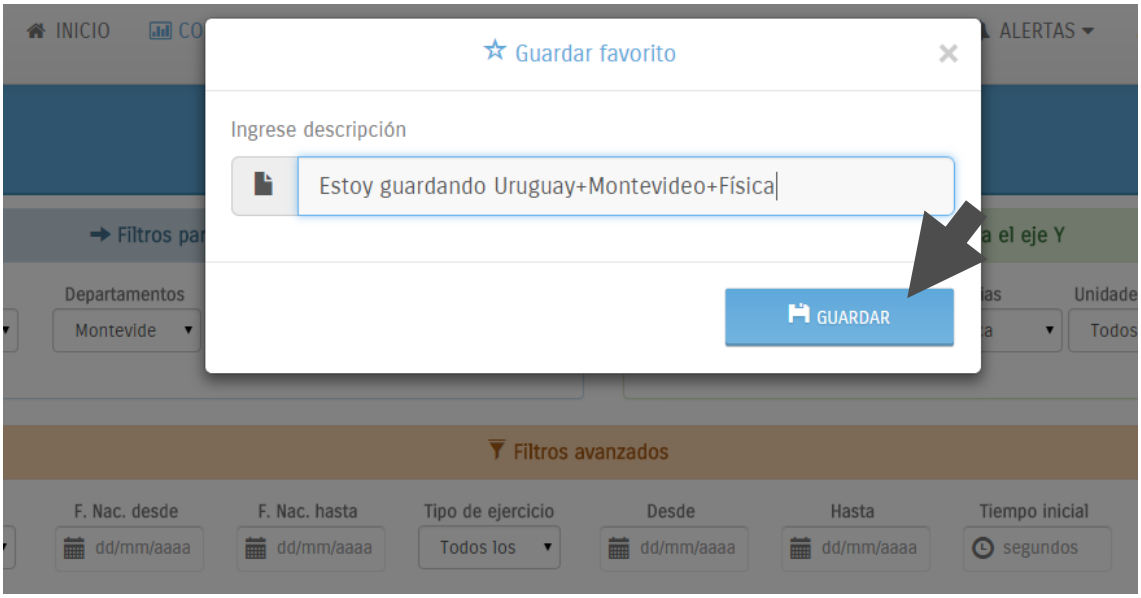


Al presionar el botón “Consultar” se podrá visualizar un gráfico de los resultados obtenidos como se visualiza a continuación.



¿Cómo puedo guardar los filtros aplicados a esta consulta en Favoritos?
Al presionar sobre “Volver a ver los filtros” próximo al botón “Consultar” se puede visualizar un botón con fondo amarillo y una estrella.

Al presionarlo se desplegará en pantalla una ventana que nos pide ingresar una descripción del filtro que queremos guardar como “Favorito” y luego al presionar “GUARDAR” los filtros aplicados a esta consulta serán guardados en el sistema para una posterior utilización.



¿Cómo ver los resultados obtenidos en la gráfica?
Al presionar sobre “Volver a ver los filtros” al lado del botón “Consultar” se puede visualizar un nuevo botón “Ver tabla”. Al presionar sobre él, se despliega la tabla de resultados junto con otras funcionalidades para mejorar la visualización.

Tabla de resultados

EXPORTAR A CSV EXPORTAR A PDF

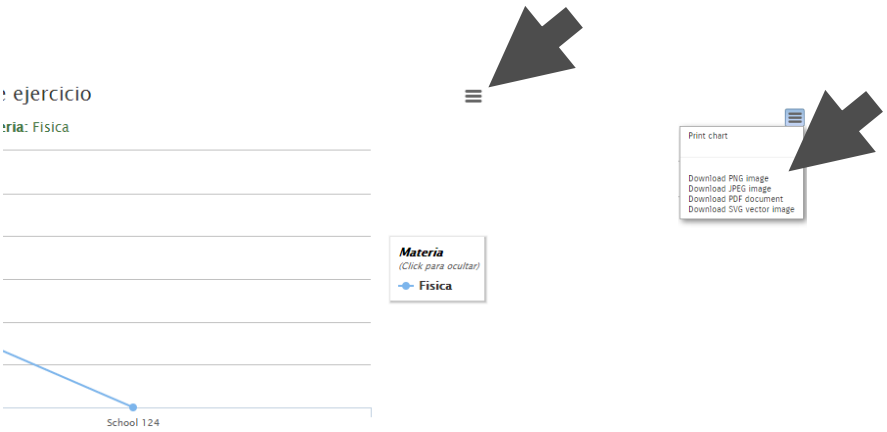
Buscar

Estudiante	Género	Fecha de nacimiento	Maestra	Ejercicio	Puntos obtenidos	Tiempo realización
Juan Perez	Masculino	2005-12-04	Estela Perez	11 - Crucigrama imÃ¡nes y brÃºjulas	0.85714	55
Paola Lopez	Femenino	2005-07-04	Estela Perez	10 - Se puede obtener un iman con un solo polo	0.2	20
Paola Lopez	Femenino	2005-07-04	Estela Perez	7 - Masa y peso no son lo mismo	1	1
Esteban Gomez	Masculino	2005-02-04	Estela Perez	7 - Masa y peso no son lo mismo	1	88
Esteban Gomez	Masculino	2005-02-04	Estela Perez	4 - CuestiÃ³n de principios	0.666	3
Esteban Gomez	Masculino	2005-02-04	Estela Perez	1 - La inercia de Mateo	0.111223	10
Gustavo Viera	Masculino	2005-02-09	Estela Perez	4 - CuestiÃ³n de principios	1	10
Gustavo Viera	Masculino	2005-02-09	Estela Perez	1 - Tipos de lentes de lupas	1	12
Rosina Rodriguez	Femenino	2005-01-09	Estela Perez	7 - Masa y peso no son lo mismo	1	6
Rosina Rodriguez	Femenino	2005-01-09	Estela Perez	8 - Â¿La luz se desvÃa al pasar del aire al agua?	1	15

Mostrando 1 a 10 de 24 registros 10 registros por página

<< < 1 2 3 > >>

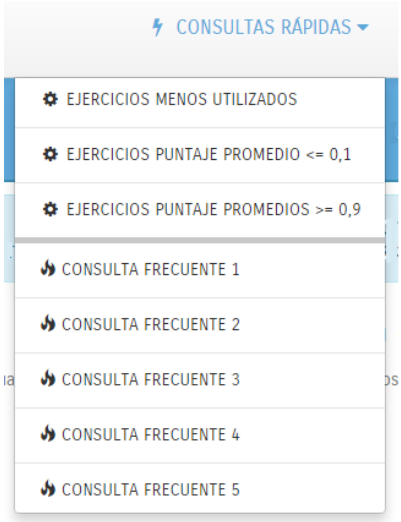
¿Cómo puedo descargar la gráfica o la tabla de resultados?
Para descargar la imagen generada se debe presionar sobre el icono que aparece arriba a la derecha y luego seleccionar el formato deseado.



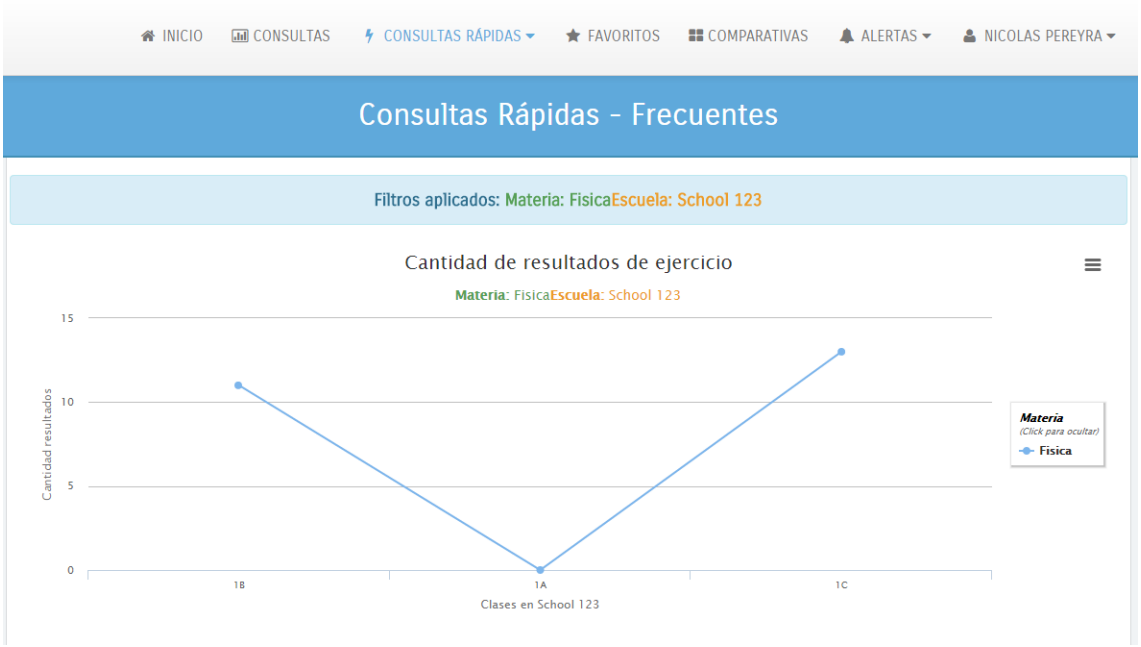
Para descargar los datos de la tabla en formato CSV y/o formato PDF solamente se debe presionar en los botones ubicados arriba del buscador de resultados de la tabla.

¿Cuáles son las Consultas Rápidas?
En este sub menú se puede ver que las primeras tres consultas fueron las solicitadas por EDU Editorial para el período configurado.

En cambio las siguientes cinco son aquellas que el usuario realiza con mayor frecuencia en el mes anterior.



Al hacer clic en cualquiera de ellas el sistema desplegará un gráfico acerca de la consulta realizada.

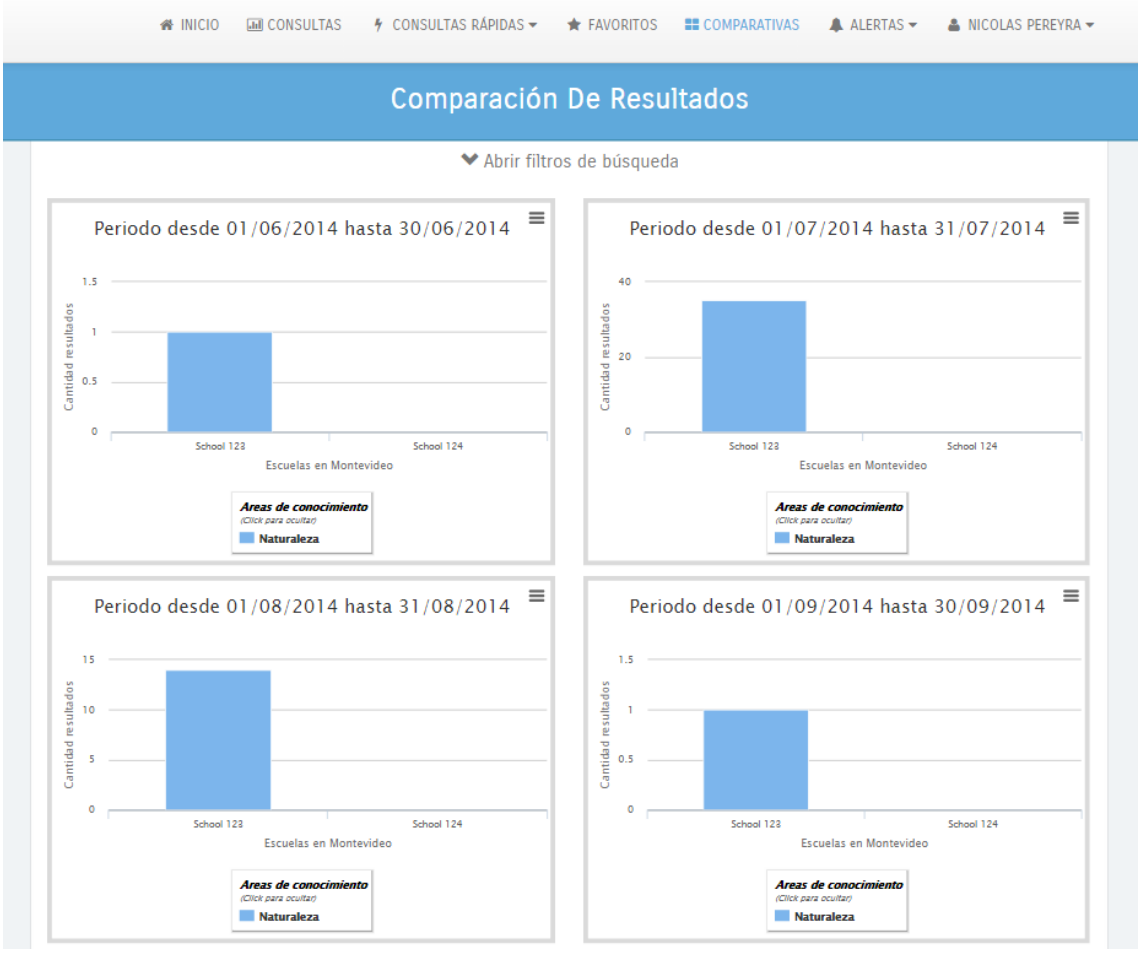


¿Cómo se utilizan los Favoritos?
En esta sección se podrán visualizar los los filtros de búsqueda guardados en el sistema como favoritos.

Al presionar sobre el ícono con forma de lupa el sistema cargará automáticamente los filtros en la sección Consultas.

¿Cuál es el funcionamiento de las Comparativas?
El sistema permite realizar comparativas en distintos intervalos de tiempo para los filtros seleccionados.

Al ingresar los filtros deseados y determinar los períodos de tiempo que se podrán visualizar el sistema desplegará los resultados en 4 cuadrantes con un gráfico cada uno.



¿Cuándo y por qué el sistema emite Alertas?
Las alertas son generadas por el sistema automáticamente cada 30 minutos. Adicionalmente también se generan luego de que transcurriera el Tiempo de Actualización y se cumplieran otras condiciones configuradas en los parámetros del sistema.

Se visualiza una cantidad determinada en la configuración de parámetros, al desplegar el menú Alertas.

También se pueden generar manualmente presionando en “Actualizar”.
Recuerde que una vez que

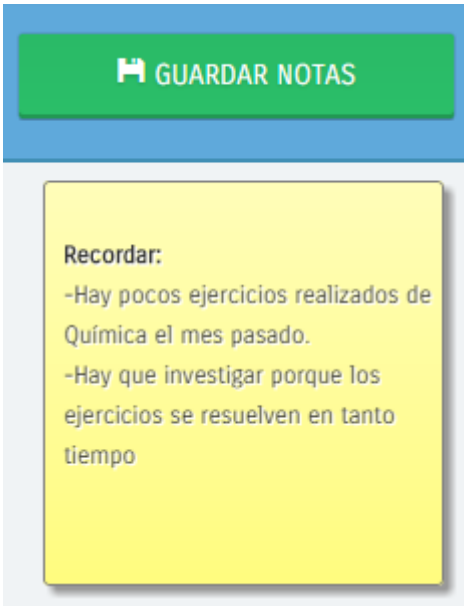
¿Se registra un histórico de las Alertas?
El histórico de Alertas se puede visualizar presionando sobre “Ver todas” en el sub menú, lo que nos dirige a una sección con todos los registros de las Alertas que se generaron.

¿Dónde se configuran los parámetros de las Alertas y las Consultas Rápidas?
En el menú presionado sobre su nombre de usuario se visualizarán una serie de opciones, siendo la primera “Configurar parámetros”

En esta opción se despliegan los parámetros configurables del sistema. Estos pueden ser modificados según los criterios que desee, para las Alertas o para las Consultas rápidas.

Al presionar el botón Guardar Cambios, se ingresan las modificaciones de parámetros al sistema.

¿Cómo guardar notas rápidamente sobre lo analizado en el sistema?
A la derecha de nuestra pantalla en cualquier sección de la aplicación se visualiza una pestaña de color amarillo con un icono de un lápiz dentro. Al presionarlo se despliega un área que permite el ingreso de anotaciones y/u observaciones que se quieran dejar en el sistema para una posterior utilización.



13.4 Anexo 4 - Capturas de pantalla en distintas resoluciones de dispositivos móviles

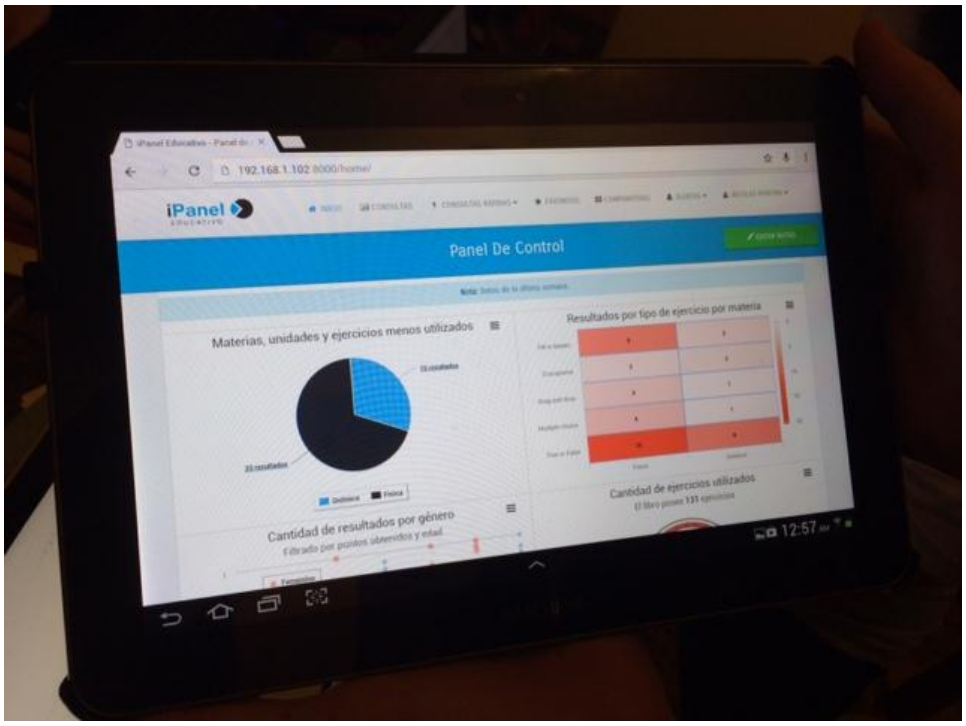


Ilustración 13-2 Foto 1 de utilización de iPanel en Tablet Android, navegador Google Chrome 37 y resolución 1280x800

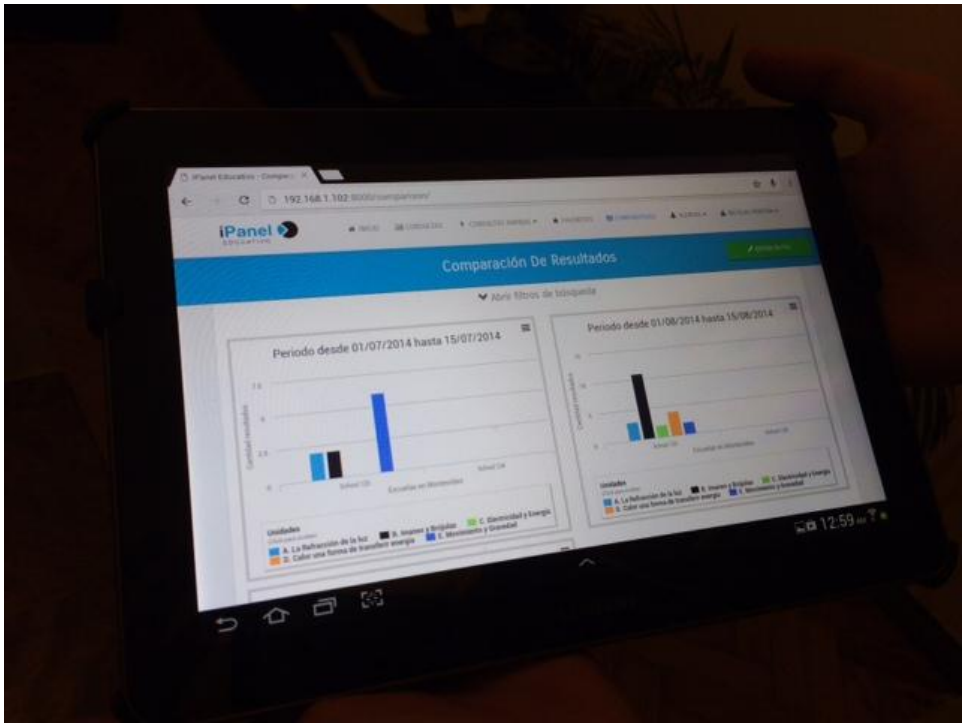


Ilustración 13-3 Foto 2 de utilización de iPanel en Tablet Android, navegador Google Chrome 37 y resolución 1280x800

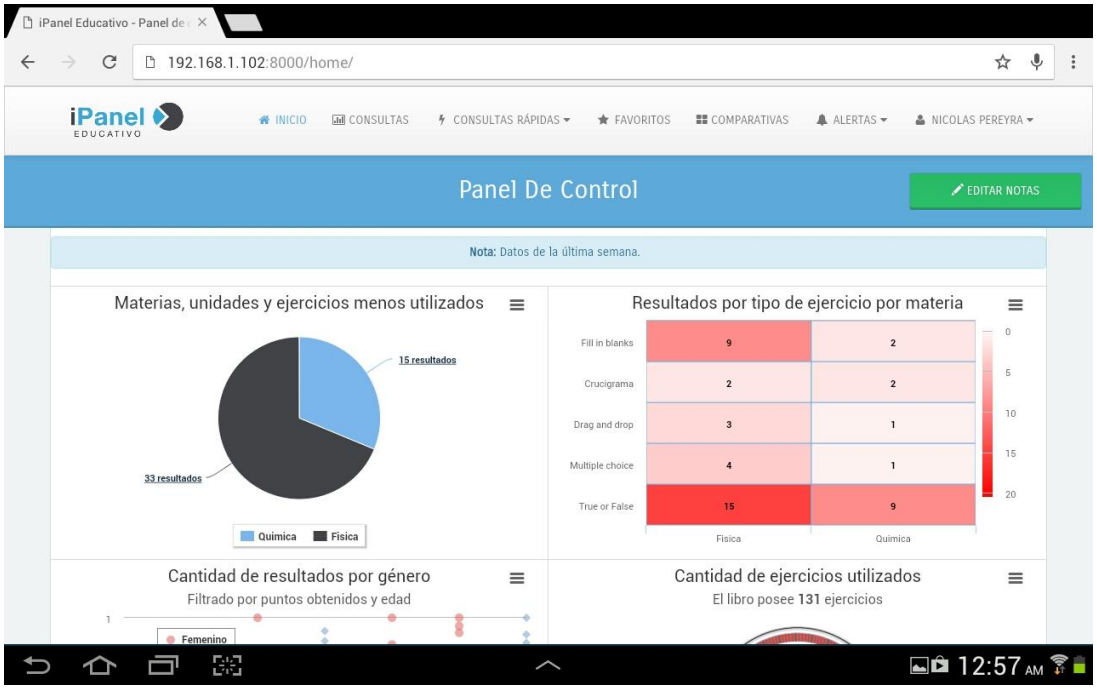


Ilustración 13-4 Captura de pantalla 1 de utilización de iPanel en Tablet Android, navegador Google Chrome 37 y resolución 1280x800

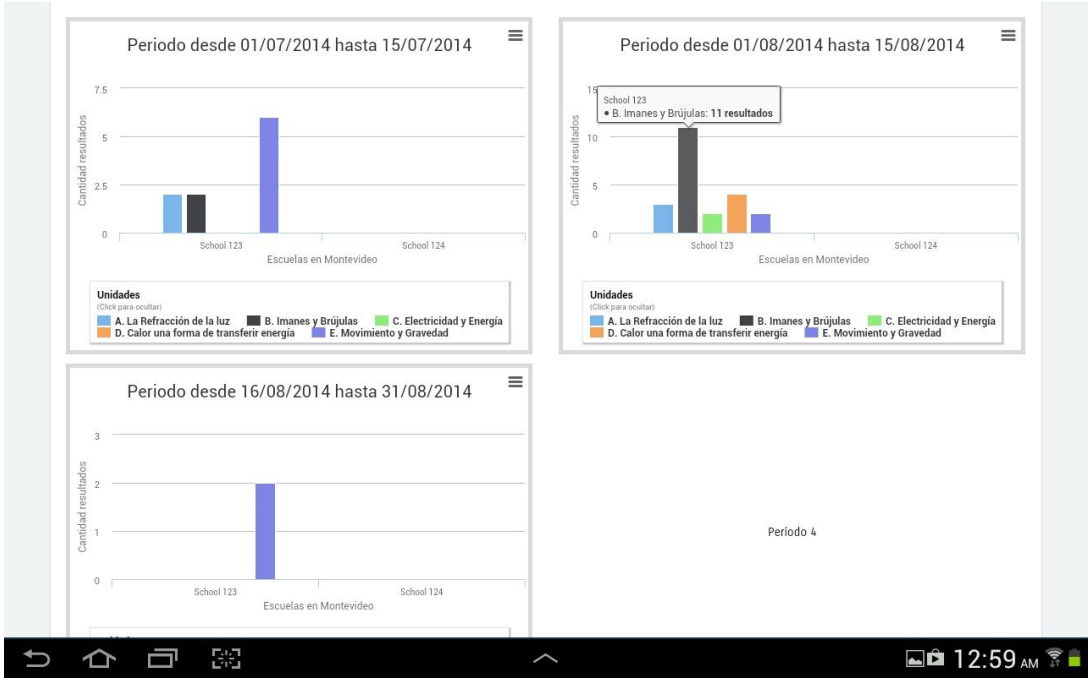


Ilustración 13-5 Captura de pantalla 2 de utilización de iPanel en Tablet Android, navegador Google Chrome 37 y resolución 1280x800

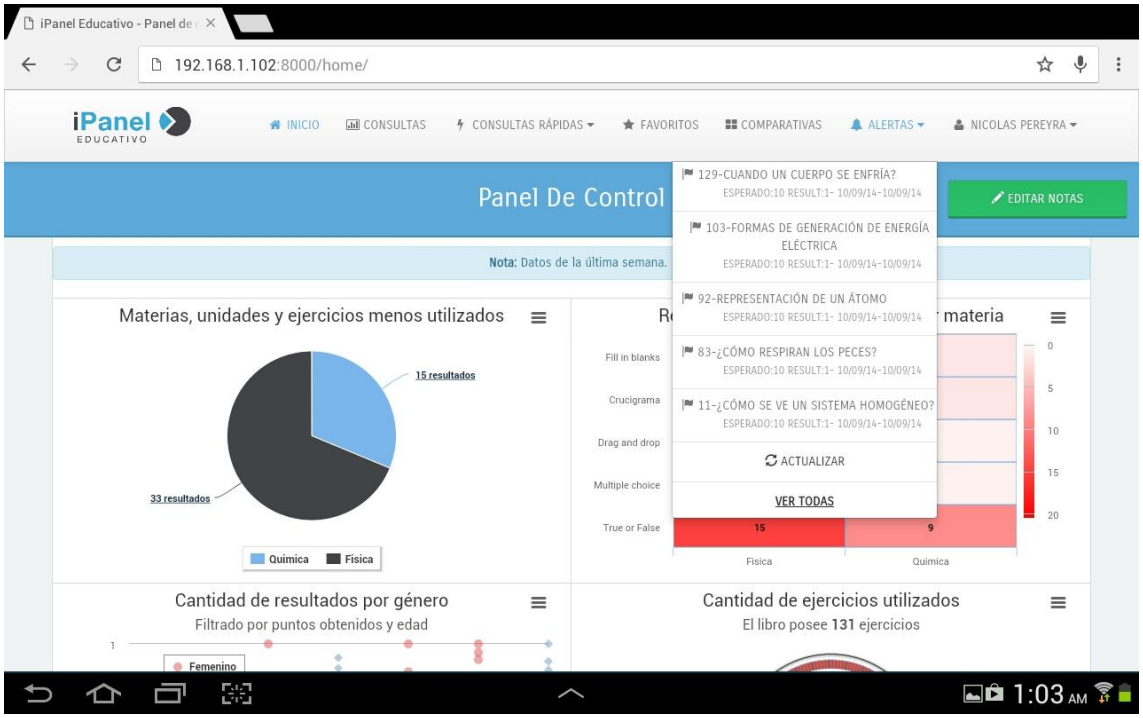


Ilustración 13-6 Captura de pantalla 3 de utilización de iPanel en Tablet Android, navegador Google Chrome 37 y resolución 1280x800

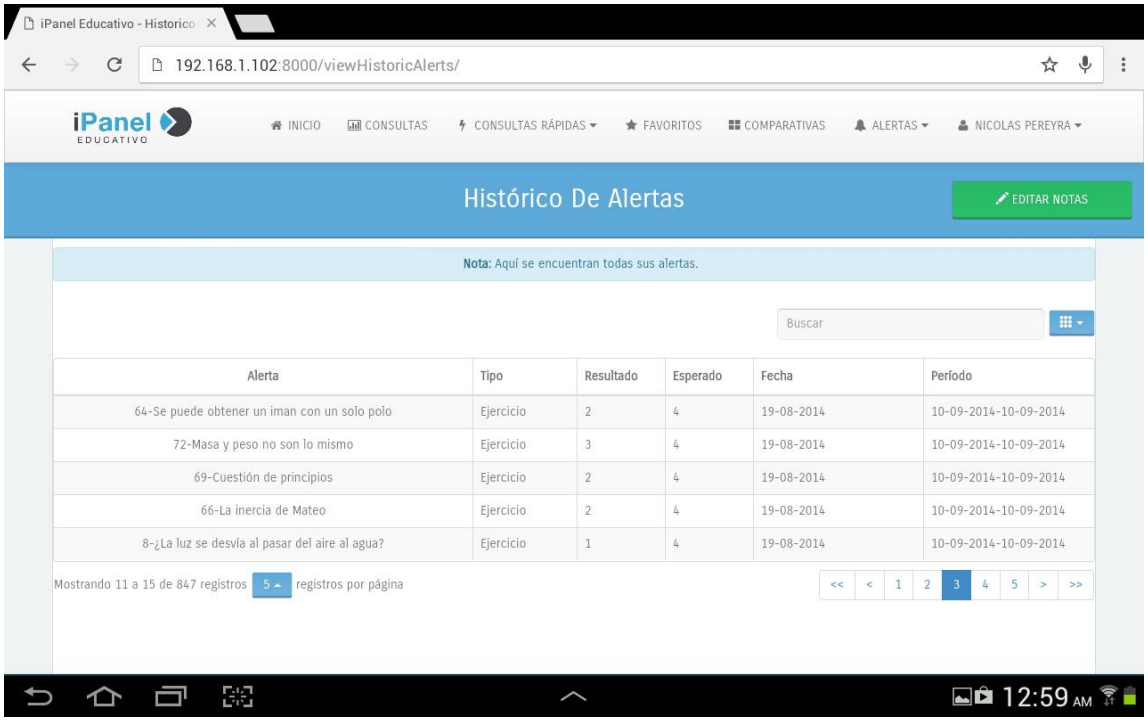


Ilustración 13-7 Captura de pantalla 4 de utilización de iPanel en Tablet Android, navegador Google Chrome 37 y resolución 1280x800

13.5 Anexo 5 – Evaluación de tecnología de desarrollo

El punto de partida fue dado por el cliente, que planteó que la aplicación debía ser web y desarrollada bajo el lenguaje de programación *Python*, específicamente la versión 2.7.6.

Python es un lenguaje de programación abierto, multiplataforma e interpretado. Tiene la particularidad de soportar varios estilos de programación, como por ejemplo: orientada a objetos, funcional e imperativa. Este posee una filosofía que busca que la sintaxis sea limpia y legible[10].

Dado que el equipo no contaba con vasta experiencia en el lenguaje, se decidió que la mejor opción era utilizar un framework, ya que, estos marcos de trabajo por lo general ofrecen componentes que facilitan funcionalidades comunes en todos los grandes proyectos. A partir de allí, el equipo investigó cual era el framework Python más conveniente.

Dado que hay varios frameworks disponibles para Python, nos planteamos ciertos criterios que debían contar los mismos para poder ser tenidos en cuenta en la selección. De esta forma, pudimos descartar varios y la muestra se redujo. En el siguiente listado, se presentan los mismos:

- Versión de Python soportada: era vital que los frameworks soporten la versión de Python solicitada.
- Estabilidad: se buscaron solamente frameworks que tuvieran una versión estable máxima de un año atrás.
- Open Source (código abierto).
- Multi plataforma.
- Que soporte el patrón MVC, necesario para la arquitectura planteada.
- Que contenga módulos para facilitar las tácticas de los atributos de calidad planteados. Por ejemplo: unit tests, soporte de capas, manejo de solicitudes/respuestas http, autenticación, autorización, logging, caché, manejador de urls, entre otras.

A continuación se detallan los frameworks investigados (aquellos que cumplieron todas las precondiciones), teniendo en cuenta sus principales ventajas y desventajas con respecto a los otros, finalizando con una conclusión referida a la decisión final.

10 https://www.python.org/about/

1. Pyramid

Pyramid [11] es un framework minimalista, open source y orientado a crear aplicaciones web de forma sencilla. Inspirado en tecnologías tales como: Zope, Pylons y Django.

Ventajas

- Ofrece varias opciones para el lenguaje a usar en los templates (vistas), para integrar la lógica de backend, estos son: Jinja, Mako, Chameleon.
- Servicios RESTful.
- Ofrece un mecanismo de comentarios propio.
- Ofrece gran cantidad de extensiones y plugins.

Desventajas

- No posee un ORM por defecto, para utilizar esta funcionalidad el usuario debe buscar e integrar el suyo si lo desea.
- No posee generación de código automáticamente.
- No ofrece una vista de administración para crear funcionalidades de forma completa o semi automática.
- El sitio web oficial cuenta con documentación escasa y poco legible.
- Pocos ejemplos de uso y/o tutoriales.

2. Web2Py

Este framework de código abierto tiene como objetivo principal dar soporte a la creación de aplicaciones web de forma ágil, segura; y enfocadas al uso de bases de datos. Sigue el patrón MVC, y fue inspirado por otras tecnologías tales como: Ruby on Rails (imitando el enfoque de desarrollo rápido) y Django (copiando varias características y funcionalidades).[12]

Ventajas

- Tracking de errores.
- Soporta una gran cantidad de motores de base de datos, por ejemplo: PostgreSQL, MySQL, Oracle, SQLite, MongoDB, MSSQL, IBM DB2, etc.
- Servicios RESTful.
- Posee la capacidad de crear funcionalidades parciales o páginas completas a través de un área de administración. Muy útil para crear sitios para acciones CRUD.
- Ofrece un mecanismo de comentarios propio.

11 http://www.pylonsproject.org/projects/pyramid/about

12 http://www.web2py.com/init/default/what

Desventajas

- El sitio web oficial cuenta con documentación escasa y poco legible.
- Pocos ejemplos de uso.
- Si bien tiene la posibilidad de extenderse a través de *plugins*, posee muy pocos.

3. Django

Es un *framework* de desarrollo web de código abierto, escrito en *Python*, que tiene una filosofía enfocada a crear aplicaciones complejas de forma ágil, apostando al reuso, la extensibilidad de componentes. Hace fuerte hincapié sobre el principio de "Don't repeat yourself", apuntando claramente a facilitarle el desarrollo al usuario[13].

Ventajas

- El sitio web oficial ofrece mucha documentación, bien organizada y legible.
- El sitio web oficial posee varios tutoriales para guiar al usuario en cómo utilizar el *framework*. Esta sección, muestra con ejemplos claros y detallados como crear una aplicación simple pero completa en pocos pasos.
- Es un *framework* mucho más maduro que los anteriores, fue lanzado en 2005. Lo cual a priori, brinda mayor respaldo en cuanto a la estabilidad de sus versiones.
- Puede generar ciertas porciones de código automáticamente, por ejemplo, en formularios, validaciones, etc.
- Al igual que el *framework Web2Py*, también posee un área de administración para generar funcionalidades mayores automáticamente o con muy pocas configuraciones.
- Posee una comunidad muy fuerte, lo cual hace que al realizar preguntas o consultas, la velocidad de respuesta sea rápida; lo que se traduce en más vías para aprender o perfeccionarse en el marco de trabajo[14].
- Tiene soporte para gran cantidad de herramientas de versionado, por ejemplo: *Git, Mercurial, Subversion*, entre otras.
- Normalización de urls, con la posibilidad de incluir protección CSRF (falsificación de petición en sitios cruzados).
- Manejo de sesiones.

Desventajas

- Al ser tan completo, tiene una curva de aprendizaje un poco superior a otros *frameworks*.
- No posee un mecanismo propio de comentarios.

13 https://www.djangoproject.com/

14 https://www.djangoproject.com/community/

4. Conclusiones

Tanto *Pyramid* como *Web2Py*, son *frameworks* robustos que si bien no poseen la gran diversidad de opciones que ofrece *Django*, están en pleno crecimiento y madurez.

De todas formas, el equipo se decantó por *Django*, ya que, no solo ofrece a priori muchas más funcionalidades, sino que también, mucho apoyo a la hora de aprender y durante el transcurso del desarrollo.

Es importante recordar que el equipo casi no posee experiencia académica o laboral en *Python*; por lo cual *Django* permitió a través de sus tutoriales, documentación y su comunidad oficial poder entender y aplicar el lenguaje y el marco de trabajo de forma más sencilla y amena.

Además el hecho de que el cliente utilice *Django*, en una de sus aplicaciones en desarrollo, hace que le evite a posteriori tener que aprender o capacitarse en otro *framework* sino se hubiese seleccionado el mismo.

13.6 Anexo 6 - Actividades de medición de costos de calidad

Actividades	Rol	Categoría
Capacitación brindada por ORTsf	General	Prevención
Capacitación interna	General	Prevención
Planificación de la función calidad	SQA	Prevención
Elaboración plan de métricas	SQA	Prevención
Preparación de inspecciones, revisiones y pruebas	SQA	Prevención
Personalización del proceso	SQA	Prevención
Definición de estándares	SQA	Prevención
Validación de arquitectura	Arquitecto	Prevención
Evaluación de herramientas y plataformas	Arquitecto	Prevención
Diseño de software	Arquitecto	Prevención
Validación del diseño	Arquitecto	Prevención
Diseño del repositorio	SCM	Prevención
Implantación del repositorio	SCM	Prevención
Administración del repositorio	SCM	Prevención
Elaboración Plan de SCM	SCM	Prevención
Armado de encuestas de relevamiento	IR	Prevención
Procesamiento de encuestas de relevamiento	IR	Prevención
Elaboración de prototipos	IR	Prevención
Validación de requerimientos	IR	Prevención
Análisis de métricas	GP	Evaluación
Validación del producto con el cliente	IR	Evaluación
Verificación de producto	IR	Evaluación
Verificación del diseño	IR	Evaluación
Revisiones por pares de casos de uso	General	Evaluación
Armado de encuestas de satisfacción para clientes y usuarios	IR	Evaluación
Procesamiento de encuestas de satisfacción de clientes	IR	Evaluación
Test A/B	SQA	Evaluación
Pruebas unitaria	General	Evaluación
Retrabajo de codificación	General	Fallas
Retrabajo de diseño (debido a defectos)	General	Fallas
Retrabajo relacionado con volver a relevar necesidades con un cliente o usuario y analizar la solución debido a errores encontrados	General	Fallas
Retrabajo asociado a la elección incorrecta de tecnología	General	Fallas
Retrabajo debido a otras causas	General	Fallas

13.7 Anexo 7 - Test A/B

Se analizaron los resultados obtenidos de la muestra a los usuarios entre la opción A y la opción B, siendo ganadora la opción B por 83,33%.

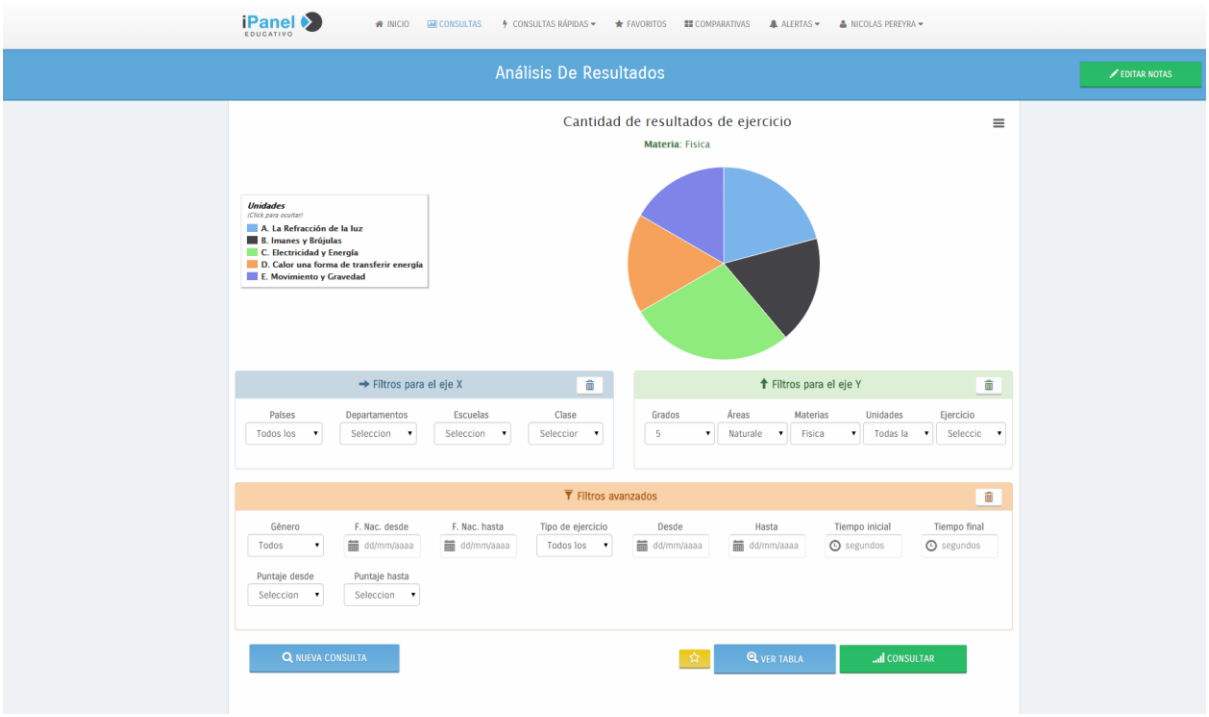


Ilustración 13-8 Opción A: los filtros por debajo del gráfico

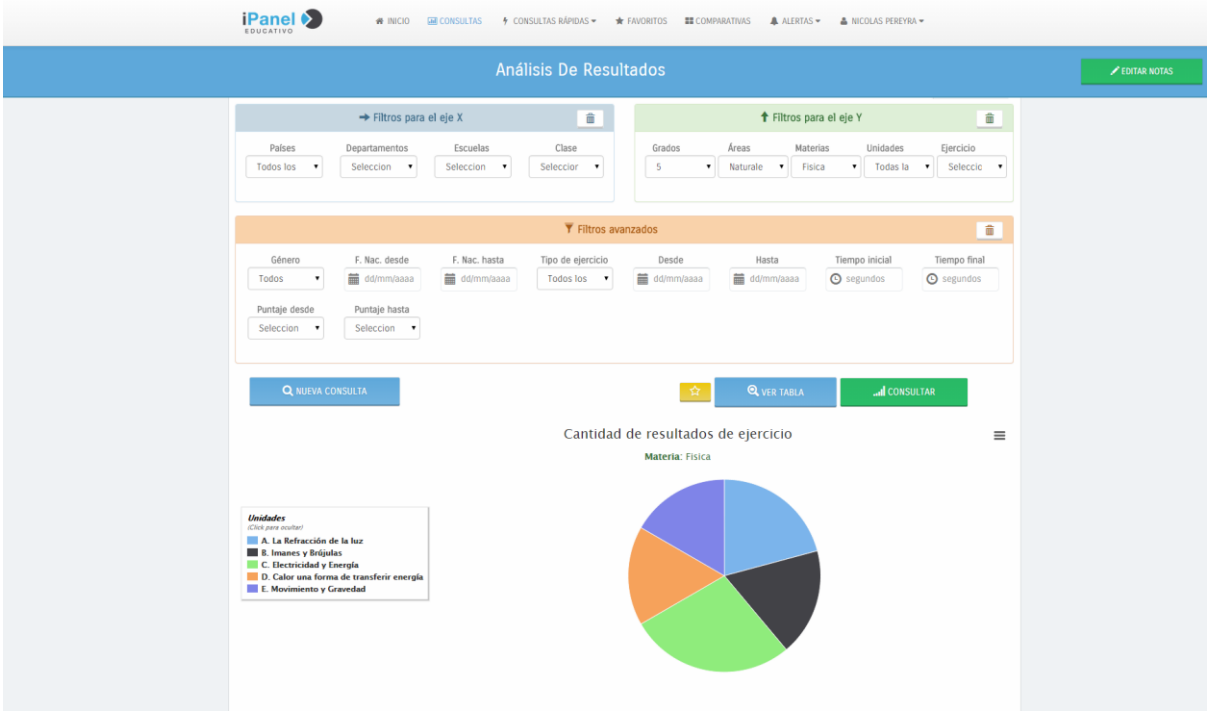


Ilustración 13-9 Opción B: los filtros por encima del gráfico

13.8 Anexo 8 – Encuesta de satisfacción de usuario

Encuesta de satisfacción

Nos interesaría saber su opinión acerca de la realización del proyecto de final de la carrera de Licenciatura de Sistemas, efectuado por los estudiantes de la Universidad ORT (Carolina Romero, Guzmán Porro, Martín Martínez y Matías Viera) para la empresa Edu Editorial.

Por favor lea con atención cada consulta e indique la opción que represente mejor su opinión.

Desde ya agradecemos su aporte.

¿La comunicación durante el proyecto fue la esperada?

- En total desacuerdo ○
- En desacuerdo ○
- De acuerdo ○
- Totalmente de acuerdo ●

¿Los tiempos previstos para el proyecto fueron los esperados?

- En total desacuerdo ○
- En desacuerdo ○
- De acuerdo ○
- Totalmente de acuerdo ●

¿El equipo de proyecto comprendió las necesidades de Edu Editorial para sus textos inteligentes?

- En total desacuerdo ○
- En desacuerdo ○
- De acuerdo ○
- Totalmente de acuerdo ●

¿La solución desarrollada satisface las necesidades detectadas?

- En total desacuerdo ○
- En desacuerdo ○
- De acuerdo ●
- Totalmente de acuerdo ○

¿Cómo evaluaría el nivel de usabilidad de la solución?

- No usable ○
- Poco intuitivo ○
- Muy intuitivo ●
- Totalmente usable ○

¿Cómo valoraría la experiencia de trabajar con este equipo de proyecto? (siendo 10 la máxima puntuación)

1 2 3 4 5 6 7 8 9 (10)

Comentarios o sugerencias

Nos pareció muy bueno el desarrollo del proyecto. Se cumplieron los requerimientos solicitados y está en nuestra planificación a futuro agregar más funcionalidades al producto

13.9 Anexo 9 - Principios y Valores de *Scrum*

Principios de *Scrum*

Valores del Manifiesto Ágil

Scrum es el más conocido de los *frameworks* Ágiles. Es la fuente de gran parte del pensamiento que se encuentra detrás de los principios y valores del Manifiesto Ágil, que a su vez forma una base común a todos estos enfoques.

Los valores del manifiesto Ágil tienen un correlato directo en *Scrum*:

- Individuos e interacciones sobre procesos y herramientas. *Scrum*, como todos los *frameworks* y métodos Ágiles, se basa directamente en la confianza puesta en los equipos, los individuos dentro de esos equipos y la manera en que éstos interactúan. Los equipos resuelven qué hay que hacer, cómo hay que hacerlo y finalmente lo hacen. Éstos identifican qué se interpone en su camino y asumen la responsabilidad de resolver todas las dificultades que se encuentren dentro de su alcance. Los equipos trabajan en conjunto con otras partes de la organización para resolver asuntos que están más allá de su control. Esto es crucial. Intentar hacer *Scrum* socavando este foco primario sobre la responsabilidad de los equipos generalmente conduce a problemas.

- Software funcionando sobre documentación extensiva. *Scrum* requiere un incremento de producto completo y funcionando como resultado final de cada Sprint. Ciertamente habrá trabajo de análisis, diseño, pruebas, el cual quizás deba ser documentado. Pero es el software funcionando lo que permite a la organización guiar al proyecto hacia el éxito. Esto es crucial. Los equipos *Scrum* deben producir un incremento de producto en cada Sprint.

- Colaboración con el cliente sobre negociación contractual. El *Product Owner* de *Scrum* es el punto de contacto principal del Equipo *Scrum* con los eventuales usuarios finales del producto y con las partes de la organización que necesitan el producto. El *Product Owner* es un miembro del equipo y trabaja colaborativamente con el equipo para decidir qué debe hacerse. Como resultado de esta colaboración el *Product Owner* selecciona el trabajo que debe realizarse a continuación, asegurando que el producto tenga el valor más alto posible en todo momento. Esto es crucial. El *Product Owner* debe construir una fuerte colaboración con el equipo.

- Respuesta ante el cambio sobre el seguimiento de un plan. En *Scrum* todo está diseñado para asegurarse que todos tengan la información que necesitan para tomar las mejores decisiones sobre el proyecto. El avance del mismo está representado por un incremento de producto real y que funciona. El *backlog* (pila) de cosas por hacer está disponible para que todos lo vean. El avance, tanto general como de *Sprint* a *Sprint*, se encuentra claramente visible. Los problemas y preocupaciones son discutidos abiertamente y resueltos de forma inmediata. Esto es crucial. *Scrum* funciona bien para

equipos que "inspeccionan" lo que sucede de forma abierta y "adaptan" sus acciones a la realidad. Funciona de forma muy pobre para los que no.

Valores de *Scrum*

Para trabajar en *Scrum* se necesita una base firme de valores que sirvan como fundamento para el proceso y los principios del equipo. A través del uso del trabajo en equipo y la mejora continua, *Scrum* tanto crea como depende de estos valores. Éstos son Foco, Coraje, Apertura, Compromiso y Respeto.

- Foco. Porque nos enfocamos en sólo unas pocas cosas a la vez, trabajamos bien juntos y producimos un resultado excelente. De este modo logramos entregar ítems valiosos antes.
- Coraje. Porque no estamos solos, nos sentimos apoyados y tenemos más recursos a nuestra disposición. Esto nos da el coraje para enfrentar desafíos más grandes.
- Apertura. Durante el trabajo en conjunto expresamos cotidianamente cómo nos va y qué problemas encontramos. Aprendemos que es bueno manifestar las preocupaciones, para que éstas puedan ser tomadas en cuenta.
- Compromiso. Porque tenemos gran control sobre nuestro destino, nos comprometemos más al éxito.
- Respeto. A medida que trabajamos juntos, compartiendo éxitos y fracasos, llegamos a respetarnos los unos a los otros, y a ayudarnos mutuamente a convertirnos en merecedores de respeto.

Si una organización permite a *Scrum* hacer su trabajo, descubrirá sus beneficios y comenzará a comprender por qué estos valores son tanto requeridos como generados por *Scrum*.

13.10 Anexo 10 – Riesgos del proyecto

Id	Nombre del Riesgo	Causa del Riesgo	Descripción / Efecto del Riesgo	Acciones preventivas	Probab.	Impacto	Magnitud
1	Falta de disponibilidad de tiempo por parte del tutor de proyecto	Situaciones personales que no le permitan al tutor estar disponible el tiempo planificado para el proyecto.	Disponer de menos tiempo del estimado para consultas y/o reuniones con el tutor de proyecto puede ocasionar demoras en algún momento del transcurso del proyecto.	Todos los miembros del equipo deben informar formalmente al resto de los involucrados aquellas fechas o periodos de tiempo en los cuales saben que van a tener una menor disponibilidad de tiempo a la definida inicialmente.	0,3	3	0,9
2	Conflictos internos entre los integrantes del equipo	Los integrantes del equipo trabajan juntos por primera vez y en momentos de estrés los intercambios de opinión pueden generar conflictos.	Discusiones sobre distintas formas de realizar el trabajo o tomar decisiones pueden ocasionar problemas a la hora de ponerse de acuerdo y ejecutar el trabajo. Esto a su vez genera retrasos en el proyecto.	Someter a votación por mayoría a las decisiones que sean necesarias. Hablar entre los integrantes en reuniones distendidas para que las discusiones sean productivas y no pasen al plano personales.	0,5	3	1,5
3	No cumplir con un alcance acorde a los tiempos del proyecto académico	Malas estimaciones de tiempo y/o alcance del proyecto.	Si el alcance no se puede realizar en el tiempo establecido para el proyecto deberá recortarse y se habrá generado una mala expectativa en el cliente.	Mediciones de esfuerzo y velocidad constantes para evaluar el desempeño del equipo y continua comunicación con el cliente.	0,5	4	2

4	No poder desarrollar un modelo de inteligencia artificial	Los miembros del equipo no han realizado trabajos previos con inteligencia artificial.	Abordar el tema de IA sin tener sólidos conocimientos al respecto lleva al fracaso del proyecto.	Consultar a un experto sobre el tema para luego poder implementarlo un modelo.	0,7	3	2,1
5	Requerimientos poco claros	El cliente no está seguro de lo que quiere y/o el equipo no puede entender sus necesidades.	Si los requerimientos son ambiguos o mal especificados el alcance quedará mal definido por lo tanto fracasará en proyecto.	Exhaustivo proceso de ingeniería de requerimientos y validaciones con el cliente durante todo el proyecto.	0,5	4	2
6	Sobrecarga de trabajo en miembros del equipo	Tareas mal asignadas dentro del equipo.	La sobrecarga del tiempo de algunas personas conlleva retrasos en el cronograma y descontento por parte del equipo.	Utilizar juicio experto para la asignación de recursos a tareas.	0,5	3	1,5
7	Retraso en el cronograma del proyecto	Estimaciones de esfuerzo equivocadas. Ausencia de un integrante del equipo.	El retraso en las tareas puede ocasionar esfuerzos desmedidos sobre el final del proyecto o hasta el fracaso del mismo.	Utilizar un ciclo de vida evolutivo que permite gestionar de mejor manera el cronograma y detectar antes los atrasos.	0,5	3	1,5
8	Problemas informáticos	Virus, robo de material, pérdida de respaldos.	La pérdida del trabajo realizado puede provocar el fracaso del proyecto.	Realizar varios respaldos físico y en la nube de todo el trabajo por parte de diferentes personas.	0,3	4	1,2

9	Ausencia o abandono de un integrante del equipo	Situaciones personales imprevistas.	En el caso de que falte alguien del equipo por un período de tiempo específico o indeterminado provocará atrasos en el cronograma y/o recortes de alcance pudiendo llegar al fracaso del proyecto.	Compromiso por parte de los miembros de equipo para avisar las ausencias y realizar las tareas asignadas a su retorno y asignación de responsables suplentes para cada rol en el proyecto.	0,3	5	1,5
10	No cubrir expectativas del cliente	Requerimientos mal definidos. Alcance no completado.	Si el cliente no resulta conforme con el trabajo realizado perderá la confianza en el equipo y no se podrá catalogar al proyecto como exitoso.	Se deben realizar pruebas con prototipos de bajo nivel con los usuarios para que todos los aspectos que no concuerden con las necesidades salgan a la luz de forma temprana.	0,3	4	1,2
11	Falta de disponibilidad de tiempo por parte de los integrantes del equipo	Situaciones personales y/o laborales que consuman más tiempo del planificado	Disponer de menos tiempo del estimado para realizar las tareas del proyecto puede sobrecargar a otros miembros del equipo o provocar el fracaso del proyecto.	Todos los miembros del equipo deben informar formalmente al resto de los involucrados aquellas fechas o periodos de tiempo en los cuales saben que van a tener una menor disponibilidad de tiempo a la definida inicialmente.	0,3	5	1,5

12	Estimaciones equivocadas sobre el esfuerzo de las tareas	Falta de experiencia en la estimación del trabajo.	Se estiman tiempos equivocados para el desarrollo de las tareas provocando retrasos en el proyecto y sobrecarga de trabajo en el final del mismo.	Deben realizarse las estimaciones basándose en el juicio experto y en caso de no disponer dicho recurso, utilizar la analogía. Agregar un tiempo extra dentro de cada iteración para asegurar que las desviaciones de tiempo no afecten la iteración.	0,7	3	2,1
13	Problemas de comunicación con el cliente	Poca disponibilidad de tiempo por parte del cliente.	No se consiguen intercambios de información a tiempo	Desarrollar un plan de comunicación acordado con el cliente.	0,3	4	1,2
14	El cliente no cuente con la arquitectura de datos definidos en la base de datos para el desarrollo del sistema.	Demoras en finalizar la edición de las tablas de la base de datos de la aplicación del cliente (Cuasimodo).	El sistema consumirá los datos de la base de datos del cliente. Si la misma no contiene datos no se puede visualizar el panel de control con información del cliente.	Realizar un seguimiento exhaustivo del trabajo en la base de datos por parte del cliente.	0,3	5	1,5
15	Desarrollar el software en el lenguaje de programación Python	Los miembros del equipo no cuentan con suficiente experiencia con el lenguaje Python.	El desarrollo del sistema puede verse demorado por la falta de experiencia con el lenguaje de programación, lo que provocaría retrasos en el cronograma.	El equipo deberá capacitarse mediante cursos on line en el lenguaje y desarrollar algunos ejercicios para adquirir práctica.	0,7	2	1,4

16	Problemas con las tecnologías durante el desarrollo del software	Debido a los distintos perfiles laborales de los miembros de equipo, no se cuenta con un nivel parejo de experiencia en las tecnologías que se pretenden utilizar para el proyecto.	Los distintos niveles de experiencia frente al desarrollo del software con determinadas tecnologías pueden provocar demoras en el proyecto dependiendo de las curvas de aprendizaje de los miembros del equipo.	Definir tempranamente las tecnologías a utilizar para que cada miembro del equipo pueda capacitarse y practicar en lo que crea que cuenta con menos experiencia que el resto.	0,3	3	0,9

13.11 Anexo 11 – Evolución de riesgos

Inicio del proyecto

Id	Probabilidad	Impacto	Magnitud
4	0,7	3	2,1
12	0,7	3	2,1
3	0,5	4	2
5	0,5	4	2
2	0,5	3	1,5
11	0,3	5	1,5
6	0,5	3	1,5
7	0,5	3	1,5
9	0,3	5	1,5
14	0,3	5	1,5
15	0,7	2	1,4
8	0,3	4	1,2
10	0,3	4	1,2
13	0,3	4	1,2
1	0,3	3	0,9
16	0,3	3	0,9

Sprint 1

Id	Probabilidad	Impacto	Magnitud
12	0,7	3	2,1
3	0,5	4	2
5	0,5	4	2
2	0,5	3	1,5
11	0,3	5	1,5
6	0,5	3	1,5
7	0,5	3	1,5
9	0,3	5	1,5
14	0,3	5	1,5
15	0,7	2	1,4
8	0,3	4	1,2
10	0,3	4	1,2
13	0,3	4	1,2
1	0,3	3	0,9
16	0,3	3	0,9
4	0	3	0

Sprint 2

Id	Probabilidad	Impacto	Magnitud
12	0,7	3	2,1
3	0,5	4	2
5	0,5	4	2
2	0,5	3	1,5
11	0,3	5	1,5
6	0,5	3	1,5
7	0,5	3	1,5
9	0,3	5	1,5
14	0,3	5	1,5
15	0,5	2	1
8	0,3	4	1,2
10	0,3	4	1,2
13	0,3	4	1,2
1	0,3	3	0,9
16	0,3	3	0,9
4	0	3	0

Sprint 3

Id	Probabilidad	Impacto	Magnitud
3	0,5	4	2
5	0,5	4	2
12	0,5	3	1,5
2	0,5	3	1,5
11	0,3	5	1,5
6	0,5	3	1,5
7	0,5	3	1,5
9	0,3	5	1,5
8	0,3	4	1,2
10	0,3	4	1,2
13	0,3	4	1,2
1	0,3	3	0,9
16	0,3	3	0,9
15	0,3	2	0,6
14	0,1	5	0,5
4	0	3	0

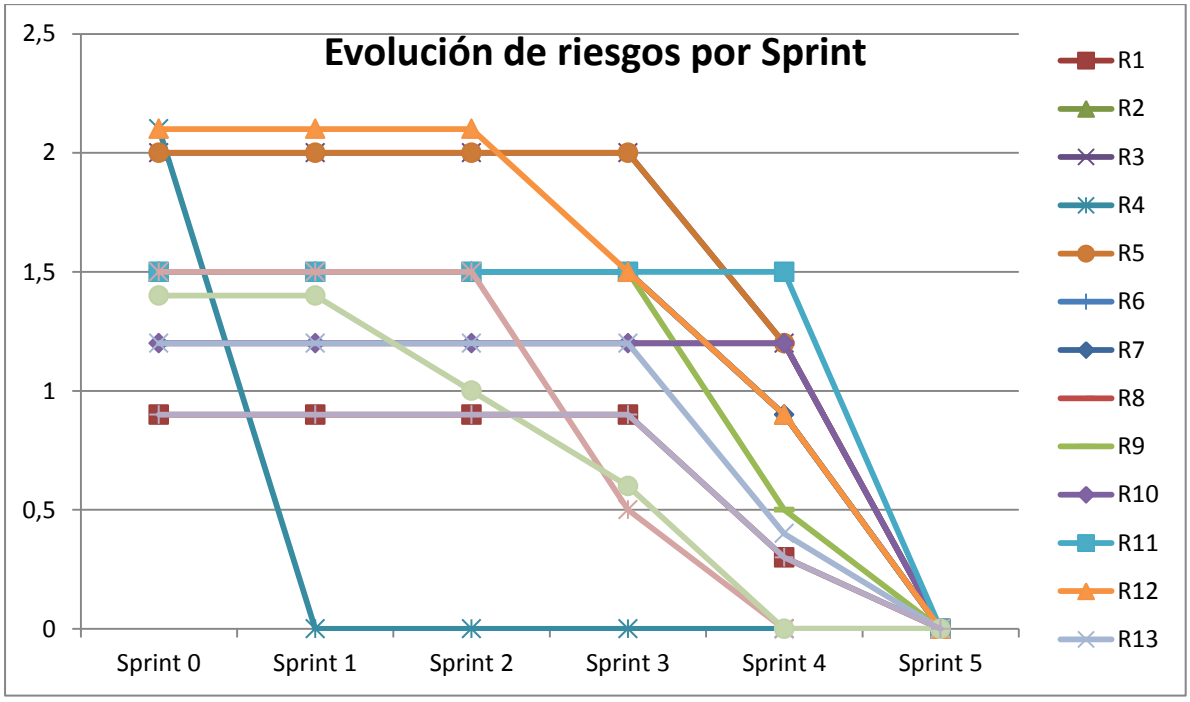
Sprint 4

Id	Probabilidad	Impacto	Magnitud
11	0,3	5	1,5
3	0,3	4	1,2
5	0,3	4	1,2
8	0,3	4	1,2
10	0,3	4	1,2
12	0,3	3	0,9
2	0,3	3	0,9
6	0,3	3	0,9
7	0,3	3	0,9
9	0,1	5	0,5
13	0,1	4	0,4
1	0,1	3	0,3
16	0,1	3	0,3
14	0	5	0
15	0	2	0
4	0	3	0

Sprint 5

Id	Probabilidad	Impacto	Magnitud
11	0,1	5	0,5
9	0,1	5	0,5
3	0,1	4	0,4
5	0,1	4	0,4
8	0,1	4	0,4
12	0,1	3	0,3
2	0,1	3	0,3
6	0,1	3	0,3
7	0,1	3	0,3
1	0,1	3	0,3
16	0,1	3	0,3
10	0,1	4	0,1
13	0,1	4	0,1
4	0	3	0
14	0	5	0
15	0	2	0

Id	Nombre del Riesgo
1	Falta de disponibilidad de tiempo por parte del tutor de proyecto
2	Conflictos internos entre los integrantes del equipo
3	No cumplir con un alcance acorde a los tiempos del proyecto académico
4	No poder desarrollar un modelo de inteligencia artificial
5	Requerimientos poco claros
6	Sobrecarga de trabajo en miembros del equipo
7	Retraso en el cronograma del proyecto
8	Problemas informáticos
9	Ausencia o abandono de un integrante del equipo
10	No cubrir expectativas del cliente
11	Falta de disponibilidad de tiempo por parte de los integrantes del equipo
12	Estimaciones equivocadas sobre el esfuerzo de las tareas
13	Problemas de comunicación con el cliente
14	El cliente no cuente con la arquitectura de datos definidos en la base de datos para el desarrollo del sistema.
15	Desarrollar el software en el lenguaje de programación *Python*
16	Problemas con las tecnologías durante el desarrollo del software



13.12 Anexo 12 – Imágenes de iPanel Educativo

Ilustración 13-10 Acceso al panel de control

Ilustración 13-11 Pantalla principal

Ilustración 13-12 Menú Consultas Rápidas

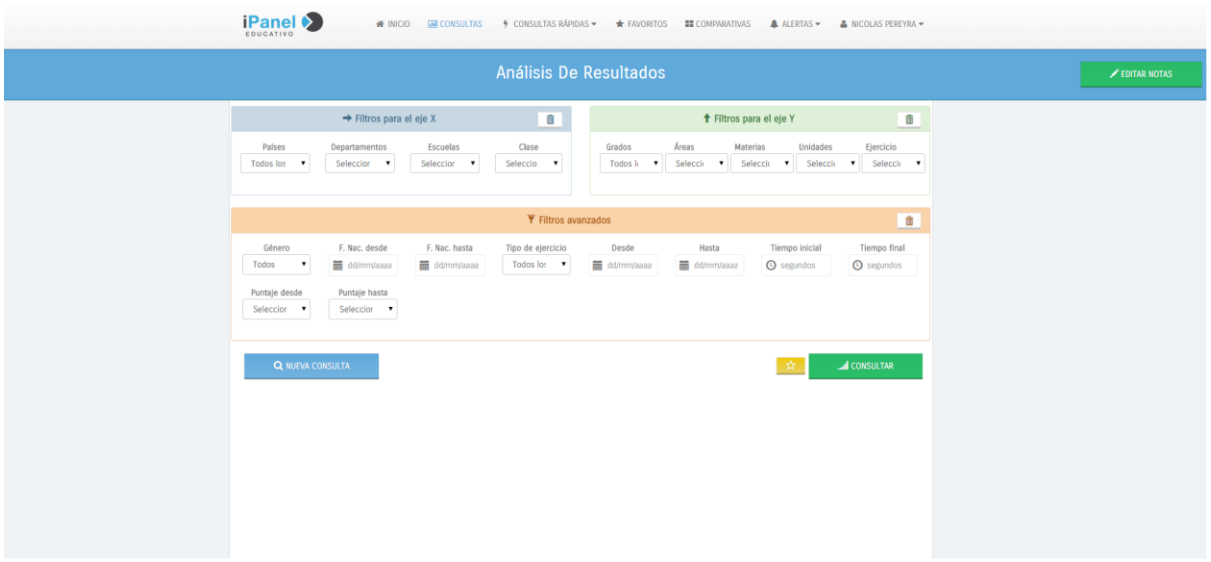


Ilustración 13-13 Pantalla de Consultas

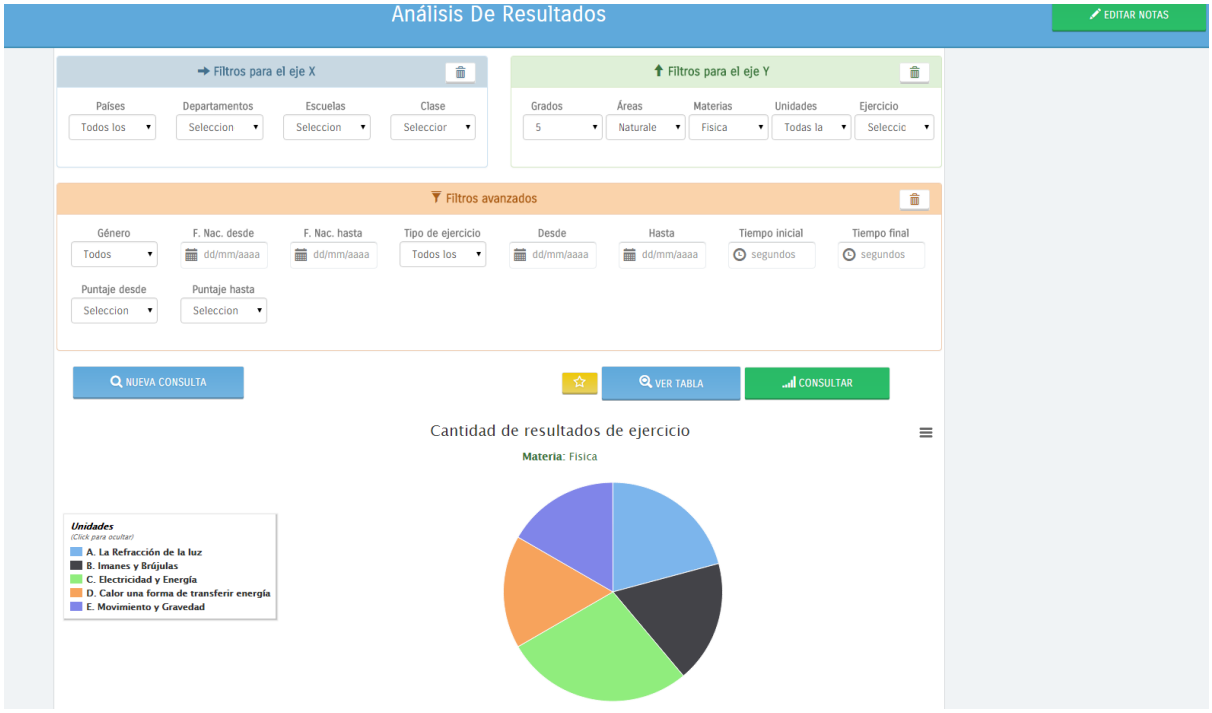


Ilustración 13-14 Pantalla de resultado de una Consulta

Ilustración 13-15 Pantalla de Comparación de resultados

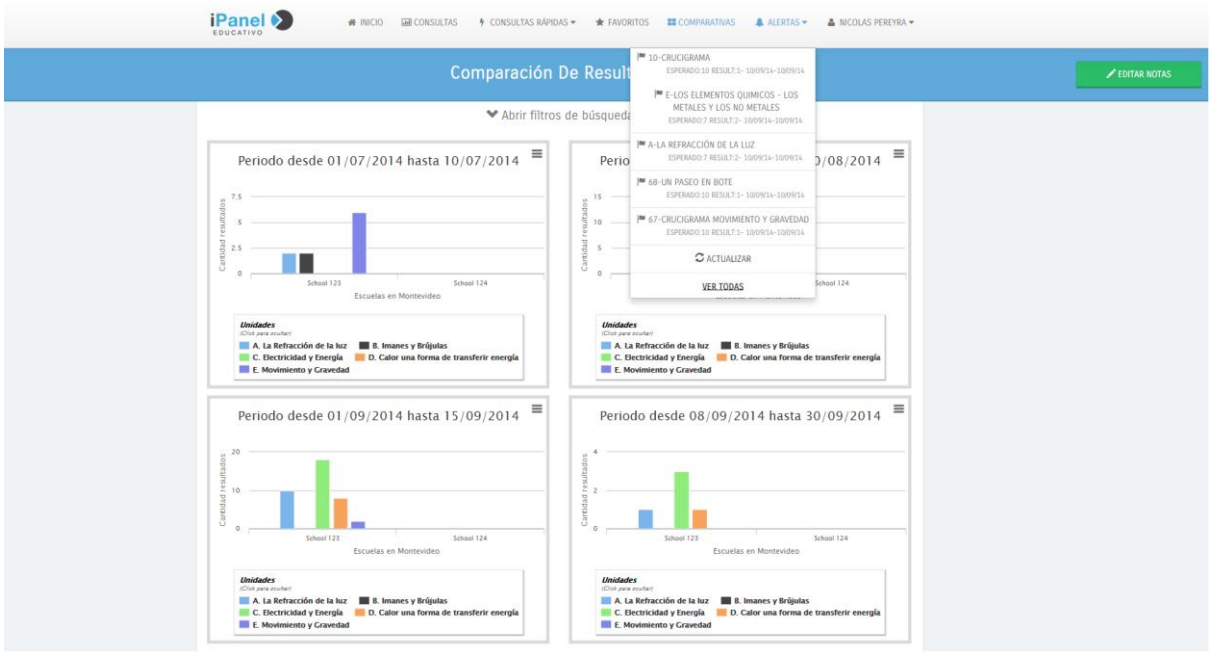


Ilustración 13-16 Pantalla de Comparación de resultados obtenidos y visualización de Alertas

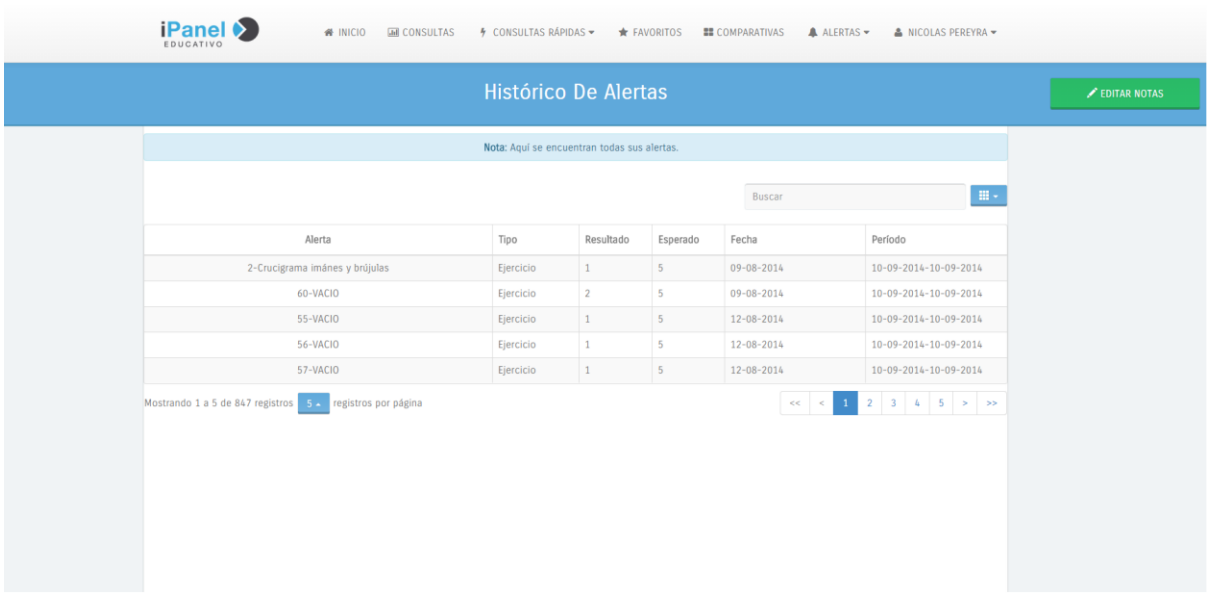


Ilustración 13-17 Pantalla visualización del registro histórico de las alertas

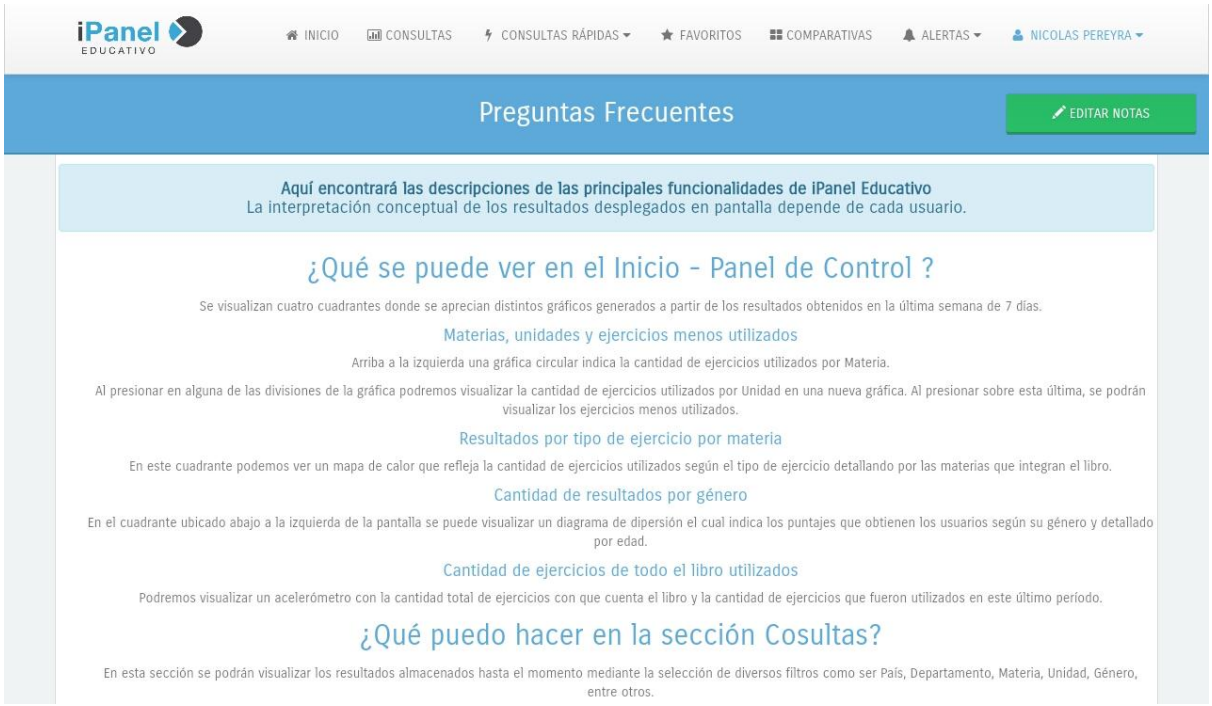


Ilustración 13-16 Pantalla de ayuda al usuario con las preguntas frecuentes

13.13 Anexo 13 – Carta de Conformidad

Braulio De León
Edu Editorial

Presente

Montevideo, 08 de Setiembre de 2014

De nuestra mayor consideración:

Recientemente hemos finalizado el Proyecto de Final de la Carrera de Licenciatura en Sistemas de la Universidad ORT. El mismo constó de un panel de control desarrollado a medida, para la toma de decisiones en el desarrollo de textos educativos para la empresa Edu Editorial.

En este sentido, presentamos la presente carta a fin de conocer su conformidad con el trabajo realizado.

Quedaremos muy agradecidos si se nos confirma la recepción de la presente carta y aprobación de nuestro trabajo, firmando y devolviendo la copia adjunta de esta carta.

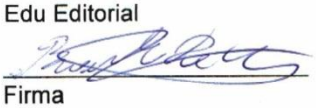
Agradeciendo de antemano su respuesta quedando a las órdenes ante cualquier consulta que estime conveniente.

Equipo de Trabajo

Carolina Romero
Guzmán Porro
Martín Martinez
Matías Viera

RECIBIDO Y ACEPTADO:

Edu Editorial



Firma

BRAULIO DE LEÓN

Aclaración

08/09/2014