



UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

**TECNOLOGÍA ESPECÍFICA DE
INGENIERÍA DEL SOFTWARE**

TRABAJO FIN DE GRADO

TheProductOwnerd:

Herramienta de ayuda a la gestión del Product Owner

Víctor Pérez Piqueras

Mayo de 2020





UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE
INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

TheProductOwnerd:

Herramienta de ayuda a la gestión del Product Owner

Autor: Víctor Pérez Piqueras

Directores: Pablo Bermejo López

Miguel Ángel Sánchez Cifo

Mayo de 2020

A mi familia, por haberme apoyado en todo momento y darme la posibilidad de poder dedicarme a lo que me apasiona. A ellos les debo todo.

A D. Pablo Bermejo López, por ser uno de los mejores docentes que he tenido la suerte de conocer.

A D. Miguel Ángel Sánchez Cifo, por haberme guiado y ayudado desde que inicié este camino el primer año hasta el día en que escribo estas líneas.

Y a ambos, por ser mis directores, por su gran esfuerzo y excelente trabajo.

A D. José Antonio Gallud Lázaro, por imbuir en mí un espíritu de inconformismo y autosuperación.

A mis amigos, que tuve la suerte de conocerlos en aquellos primeros meses de clase y me han acompañado hasta el final.

A todos los profesores y compañeros con los que he compartido escuela durante estos cuatro años.

A ella, por ser lo más bonito que me llevo de este grado.

Declaración de Autoría

Yo, Víctor Pérez Piqueras, con DNI 48259328-S, declaro que soy el único autor del Trabajo Fin de Grado titulado “TheProductOwnerd: Herramienta de ayuda a la gestión del Product Owner” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 7 de junio de 2020

Fdo.: Víctor Pérez Piqueras

Resumen

En la actualidad, el creciente uso de metodologías ágiles para la gestión de proyectos software muestra poco a poco sus efectos en el sector tecnológico, siendo uno de estos la entrega rápida y de valor al cliente. Así lo indica uno de los pilares del movimiento ágil: “Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor”. En la metodología Scrum, la figura que carga sobre sus hombros la responsabilidad de maximizar el valor entregado al cliente es el Product Owner. Pero priorizar las características que más valor aportan al cliente es solo una de sus funciones. Scrum establece sus cimientos sobre la transparencia, siendo de especial importancia en un desarrollo ágil la generación y visualización de datos sobre el estado de un proyecto y su evolución. Del buen uso que el Product Owner haga de esta información, realizando estimaciones, predicciones e informes, dependerá que un equipo mejore su rendimiento, se detecten problemas estructurales o un proyecto cumpla su fecha límite.

El auge de las metodologías ágiles ha provocado la aparición en el mercado de cientos de herramientas y servicios enfocados en facilitar esta gestión. A pesar de este gran número de servicios y aplicaciones dedicadas a la gestión ágil y de la gran importancia que tiene el buen tratamiento y uso de la información generada en un desarrollo ágil, pocos son los servicios que proporcionan métodos eficaces de proyección y las herramientas predictivas son ínfimas.

Con el fin de resolver esta carencia nace TheProductOwnerd, una aplicación web centrada en el Product Owner y la gestión ágil de proyectos. Esta aplicación permite al usuario gestionar proyectos, incorporando la funcionalidad tradicional de gestión del Product Backlog, creación de sus elementos y gestión de equipos. Para suplir la falta de métodos predictivos y proyecciones la aplicación incluye dos módulos: una sección de informes de proyecto, que analiza diversos aspectos del desarrollo generando informes gráficos del estado actual; y la sección de predicciones, que incorpora modelos predictivos que otorgan al Product Owner la capacidad de trabajar con la información conocida hasta la fecha para predecir el progreso de un proyecto, detectar tendencias y problemas, adelantarse a los acontecimientos y, en definitiva, aumentar las probabilidades de éxito del proyecto.

Agradecimientos

Como un gran rascacielos que decora la ciudad, mi Trabajo Fin de Grado no habría llegado a alzarse en el firmamento sin cimientos, sin los pilares que soportando una alta presión han hecho de este trabajo una realidad. Aunque es complicado condensar en pocas líneas la gratitud que siento por toda la gente que me ha apoyado durante estos cuatro años, haré un esfuerzo, pues merecen esto y más.

Empezando por quienes han vivido este trabajo más de cerca conmigo, quisiera darle las gracias a D. Pablo Bermejo López por su gran apoyo dentro y fuera del aula. Recuerdo que uno de los últimos días de clase de segundo nos deseó suerte para la dura etapa del tercer curso dibujando en la pizarra dos símbolos que expresan con gran claridad el durante y el después de dicho curso: “:(” y “:)”. Aunque parezca mentira, dos puntos y un cierre de paréntesis me dieron fuerza para poder soportar todos los golpes, superar el curso y finalmente volver a asistir a sus clases un año y medio después.

A D. Miguel Ángel Sánchez Cifo quiero agradecerle toda la ayuda que me ha proporcionado desde el día en que llegué a la escuela, desorientado y perdido, y se presentó como mi mentor portando unas hojas de evaluación; hasta hoy, día en que finaliza esta etapa de mi vida.

A Carolina le quiero agradecer todo el apoyo y ánimos que me ha brindado y que han sido vitales para superar todos los retos que he encontrado desarrollando este Trabajo Fin de Grado. A pesar de la distancia siento sus fuerzas y ánimos como si estuviera junto a mí.

Quiero darle las gracias a mi familia, porque sin ellos nada de esto habría sido posible.

También quiero darle las gracias a todos los amigos con los que he compartido experiencias, viajes y anécdotas durante el grado y, sobre todo, carcajadas, pues al final son estos momentos los que uno recuerda con el tiempo.

Por último, quiero agradecer a los compañeros y profesores del grado su labor, el trato y la implicación que han dedicado, convirtiendo el gris edificio de la Politécnica en un hogar para mí.

Índice de Contenido

Declaración de Autoría	III
Resumen.....	V
Agradecimientos.....	VII
CAPÍTULO 1. Introducción	1
1.1. Contexto	1
1.2. Motivación	1
1.3. Objetivos	2
1.4. Competencias	2
1.5. Estructura de la memoria	4
CAPÍTULO 2. Antecedentes y estado de la cuestión.....	5
2.1. Introducción	5
2.2. El Product Owner	5
2.3. Entregas, velocidad y estimaciones	6
2.4. Herramientas y servicios.....	14
2.5. Conclusiones	16
CAPÍTULO 3. Metodología y desarrollo.....	19
3.1. Introducción	19
3.2. Metodología	19
3.3. Tecnologías y arquitectura	21
3.4. Especificaciones	23
3.5. Pipeline y herramientas.....	26
3.6. Artefactos del desarrollo	29
3.7. Conclusiones	34
CAPÍTULO 4. Documentación de desarrollo y pruebas de usabilidad.....	35
4.1. Introducción	35
4.2. Documentación de sprints.....	35
4.3. Pruebas de usabilidad.....	59
4.4. Conclusiones	62
CAPÍTULO 5. The Product Ownerd App.....	63
5.1. Introducción	63
5.2. Gestión de usuarios.....	63

Índice de contenido

5.3. Overview	64
5.4. Backlog	68
5.5. Forecasts	71
5.6. Comparativa entre proyectos	73
5.7. Conclusiones	73
CAPÍTULO 6. Conclusiones y trabajo futuro	75
6.1. Introducción	75
6.2. Conclusiones	75
6.3. Trabajo futuro.....	78
Bibliografía.....	79
Anexos	81
ANEXO A. Guion de pruebas de usabilidad	81
ANEXO B. ISO/IEC 25062 Common Industry Format for Usability Test Reports	85
ANEXO C. Sugerencias de usuarios.....	99
ANEXO D. Listado de Product Backlog Items	101
Contenido del CD.....	105

Índice de Figuras

Figura 2.1: Razones para las entregas	7
Figura 2.2: Gráfico burn-down de entregas con predicciones	9
Figura 2.3: Proyección del alcance basada en el burn-down chart	10
Figura 2.4: Distribución de probabilidades basada en la simulación de Monte Carlo	12
Figura 2.5: Composición de la Velocidad con el Tiempo	13
Figura 2.6: Logos de Zenhub, Jira y Yodiz	15
Figura 3.1: Arquitectura general	22
Figura 3.2: Diagrama de clases	23
Figura 3.3: Diagrama Entidad-Relación	24
Figura 3.4: Diagrama de despliegue	25
Figura 3.5: Pipeline de DevOps	26
Figura 3.6: Pipeline de Integración Continua, Entrega Continua y Despliegue	27
Figura 3.7: Dependencias del proyecto	30
Figura 3.8: Descripción del repositorio	30
Figura 3.9: Tabla de etiquetado de PBIs	30
Figura 3.10: Cronograma de alto nivel	31
Figura 3.11: Project Burndown Chart	32
Figura 3.12: Kanban de ZenHub	33
Figura 4.1: Sprint 2 - User Story Mapping	37
Figura 4.2: Sprint 2 - Entregas del User Story Mapping	38
Figura 4.3: Sprint 3 - Build en Travis	39
Figura 4.4: Sprint 4 - Mensaje automático de Slack	40
Figura 4.5: Sprint 4 - Predicción por velocidad	41
Figura 4.6: Sprint 7 - Interfaz de listado de proyectos	44
Figura 4.7: Sprint 8 - Vista principal del proyecto	45
Figura 4.8: Sprint 8 - Vista del Product Backlog	46
Figura 4.9: Sprint 8 - Vista detallada de un PBI	46
Figura 4.10: Sprint 8 - Interfaces de Listado y Detalles de PBIs	47
Figura 4.11: Sprint 10 - Vista de Forecast by Velocity	49
Figura 4.12: Sprint 11 - Configuración de Despliegue de Heroku	50
Figura 4.13: Sprint 12 - Interfaz de Overview	52
Figura 4.14: Sprint 12 - Email de invitación	53
Figura 4.15: Sprint 12 - Pestaña de About	53
Figura 4.16: Sprint 13 - Modelo de Regresión Lineal	54
Figura 4.17: Sprint 13 - Modelo de Regresión Polinómica	55
Figura 4.18: Sprint 13 - Predicción por velocidad	55
Figura 4.19: Sprint 13 - Predicción por regresión lineal	56
Figura 4.20: Sprint 14 - Comparativa de proyectos por velocidad relativa	57
Figura 4.21: Resultados subjetivos globales de las pruebas de usabilidad	62
Figura 5.1: Login de usuarios	64
Figura 5.2: Cabecera de un proyecto	65
Figura 5.3: Project Burndown Chart y Equipo	65
Figura 5.4: Email de invitación	67
Figura 5.5: Bug Status, PoC y Visión	68
Figura 5.6: Vista del Backlog	69

Índice de figuras

Figura 5.7: Vista de un PBI - 170

Figura 5.8: Vista de un PBI - 270

Figura 5.9: Predicción por velocidad71

Figura 5.10: Modelo de regresión lineal72

Figura 5.11: Modelo de regresión parabólica.....72

Figura 5.12: Comparativa entre proyectos73

Figura B.1. Resultados subjetivos globales.....96

Índice de Tablas

Tabla 3.1: Listado de herramientas y servicios	28
Tabla 5.1: Matriz de roles-permisos	66
Tabla B.1. Participantes en la prueba	87
Tabla B.2. Resultados de la Tarea 1	90
Tabla B.3. Resultados de la Tarea 2	91
Tabla B.4. Resultados de la Tarea 3	91
Tabla B.5. Resultados de la Tarea 4	91
Tabla B.6. Resultados de la Tarea 5	92
Tabla B.7. Resultados de la Tarea 7	92
Tabla B.8. Resultados de la Tarea 7	92
Tabla B.9. Resultados de rendimiento combinados	93
Tabla B.10. Resultados subjetivos.....	94

CAPÍTULO 1. INTRODUCCIÓN

*“The agile way is more adapt to changes but shall not lose the sight of big picture”
- Pearl Zhu.*

1.1. CONTEXTO

Actualmente se ha extendido el uso de las metodologías ágiles en el desarrollo de proyectos software. Estas metodologías, como Scrum [1], permiten aportar una mayor flexibilidad al producto y entregar un mayor valor al cliente. Para poder generar un producto de forma competitiva y rápida es necesaria una buena gestión de la información que genera el propio desarrollo. Es de esta gestión de la que se encarga el Product Owner, uno de los roles de Scrum. De su labor dependerá que se prioricen las actividades más productivas para el cliente y se maximice el valor entregado, optimizando así el trabajo realizado por el Equipo de Desarrollo.

1.2. MOTIVACIÓN

En la actualidad existen numerosas herramientas y servicios que facilitan la gestión ágil de proyectos. No obstante, estas aplicaciones suelen enfocarse en las necesidades de los desarrolladores, descuidando sus métodos de gestión, informes y predicciones. Ante la evidente carencia de una herramienta que proporcione a un Product Owner la capacidad de gestionar proyecciones de manera automatizada, nace una propuesta de aplicación que posteriormente se desarrolla en este trabajo. Esta aplicación, además de incluir las funcionalidades básicas de gestión de proyectos, aportaría al Product Owner diversos tipos de proyecciones y modelos predictivos.

1.3. OBJETIVOS

Los objetivos de este trabajo pueden dividirse en dos secciones. Por una parte, se encuentran los objetivos y propósitos de establecer una metodología de desarrollo específica para la realización de la aplicación. Adicionalmente, se encuentran los objetivos propios del producto a desarrollar, que buscan resolver una problemática concreta.

1.3.1. OBJETIVOS DEL DESARROLLO

El objetivo de este trabajo no es únicamente producir una aplicación de calidad y funcional, sino también aplicar una metodología de desarrollo ágil que siga las vías de DevOps [2]. Para esto, se aplicarán los siguientes puntos:

- Utilizar Scrum como marco de trabajo para la gestión del proyecto.
- Establecer un pipeline de Integración Continua que ejecute pruebas automáticas tras cada incremento del producto.
- Extender este pipeline con Despliegue o Entrega Continua [3], actualizando la aplicación final en producción tras cada incremento.

1.3.2. OBJETIVOS DEL PRODUCTO

El producto software generado busca cumplir los siguientes objetivos:

- Proporcionar las herramientas básicas de gestión del Product Backlog y de los Product Backlog Items encontradas en la mayoría de las aplicaciones de gestión ágil.
- Generar proyecciones e informes del estado actual del proyecto para facilitar su gestión.
- Crear modelos predictivos en base a la información de los últimos sprints para apoyar la toma de decisiones del Product Owner.

1.4. COMPETENCIAS

A continuación, se listan las competencias asociadas a la intensificación de Ingeniería del Software que se han desarrollado y aplicado en la realización de este Trabajo Fin de Grado, relacionando estas con las actividades llevadas a cabo durante el desarrollo del trabajo.

- **[IS1]** Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente,

sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la Ingeniería del Software.

- **[IS2]** Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.
- **[IS3]** Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.
- **[IS4]** Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.
- **[IS5]** Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.

La planificación y gestión del proyecto asocia la competencia **[IS5]**, útil para realizar una planificación de alto nivel del proyecto, evaluar los riesgos y su nivel de impacto y poder afrontar las posibles soluciones y alternativas a estos.

La competencia **[IS2]** se aplicó durante la captura inicial de requisitos y durante la actualización de estos tras cada uno de los sprints del desarrollo. Esta competencia fue especialmente útil a la hora de conciliar las necesidades y requisitos encontrados para cada tipo de usuario y poder priorizar estos según su relevancia.

El diseño, modelado e implementación de la solución aplicaron las competencias **[IS3]** e **[IS4]**, aportando la capacidad de analizar los problemas que se deben solucionar, elegir el procedimiento más eficiente en términos de usabilidad y rendimiento y saber documentar estas actividades de forma correcta y estandarizada.

La evaluación y documentación del producto software desarrollado enlazan las competencias **[IS1]** e **[IS4]**, dando la capacidad de desarrollar productos software fiables y eficientes de acuerdo con estándares de calidad y buenas prácticas de la Ingeniería del Software.

1.5. ESTRUCTURA DE LA MEMORIA

La memoria se divide en capítulos que conforman la siguiente estructura:

- **Capítulo 1: Introducción.** En este capítulo se introduce de forma resumida el Trabajo Fin de Grado.
- **Capítulo 2: Antecedentes y estado de la cuestión.** En este segundo capítulo se exponen las bases teóricas sobre las que parte la propuesta y, seguidamente, se pone en contexto la problemática que se pretende resolver mediante este trabajo.
- **Capítulo 3: Metodología y desarrollo.** Se describe la planificación, gestión y desarrollo del trabajo, resumiendo la metodología aplicada y justificando las tecnologías y arquitecturas seleccionadas para la implementación de la solución.
- **Capítulo 4: Documentación de desarrollo y pruebas de usabilidad.** En este capítulo se reúnen los documentos descriptivos de cada sprint del desarrollo y se adjunta la descripción y resultados de las pruebas de usabilidad llevadas a cabo durante el desarrollo.
- **Capítulo 5: The Product Ownerd App.** A lo largo de este capítulo se expone la aplicación resultado del Trabajo Fin de Grado, mostrando las vistas y funcionalidades que la aplicación provee al usuario.
- **Capítulo 6: Conclusiones y propuestas.** En este capítulo se exponen las conclusiones extraídas tras la realización del Trabajo Fin de Grado, así como las posibles mejoras y trabajos futuros que se proponen para la aplicación.
- **Anexo A: Guion de pruebas de usabilidad.** En este anexo se adjunta el guion proporcionado a los participantes de la prueba de usabilidad realizada durante el desarrollo de la aplicación.
- **Anexo B: ISO/IEC 25062 Common Industry Format for Usability Test Reports.** En este anexo se adjunta el informe generado para documentar los resultados de las pruebas de usabilidad.
- **Anexo C: Sugerencias de usuarios.** En este anexo se listan todas las propuestas, modificaciones y anotaciones que los participantes de las pruebas entregaron tras dichas pruebas de usabilidad.
- **Anexo D: Listado de Product Backlog Items.** Este anexo adjunta en una tabla los Product Backlog Items creados durante el desarrollo, incluyendo título e identificador de cada uno de estos.

CAPÍTULO 2. ANTECEDENTES Y ESTADO DE LA CUESTIÓN

*“If you don’t know where you are going, how can you expect to get there?”
– Basil S. Walsh.*

2.1. INTRODUCCIÓN

En este capítulo se pone en contexto el estado de la cuestión, desarrollando varios de los elementos más importantes y dando unas pinceladas sobre la problemática a resolver. Primeramente, en la sección 2.2 se introduce la figura del Product Owner (PO) y algunas de sus tareas y necesidades. En la sección 2.3 se trata la importancia de los despliegues en un proyecto software y los distintos métodos de realizar estimaciones. Seguidamente, en la sección 2.4 se centra el estudio en las distintas herramientas a las que puede acceder el PO, analizando el estado actual del mercado de dichas herramientas y valorando varios de los servicios más utilizados. Por último, se sintetizan las cualidades de las herramientas estudiadas que más aportan a la labor del PO y se finaliza el capítulo tras un resumen de conclusiones.

2.2. EL PRODUCT OWNER

El PO, uno de los tres roles de Scrum [1], además del Scrum Master y el Equipo de Desarrollo, es el encargado de maximizar el valor del producto, el cual es el resultante del trabajo del equipo de desarrollo. Además, el PO tiene sobre sus hombros la responsabilidad de gestionar de una manera inteligente el Product Backlog. Para llevar a cabo dicha tarea, deberá ser capaz de definir de forma clara los Product Backlog Items (PBIs), priorizar estos en función de las necesidades del cliente y optimizar el valor que el equipo de desarrollo realiza en sus entregas. Scrum depende en gran parte de la transparencia. Las decisiones que

CAPÍTULO 2. Antecedentes y estado de la cuestión

el PO tome se basarán en información empírica y en datos obtenidos del trabajo realizado, incluyendo los artefactos de Scrum, de forma que la carencia de transparencia se traducirá en un aumento del riesgo del proyecto y en una disminución del valor entregado al cliente.

Uno de los artefactos que mayor retroalimentación aportan sobre el estado de un proyecto es el Product Burndown Chart (PBCh) [1]. Con este artefacto, el PO puede realizar predicciones sobre la situación actual del proyecto, detectar problemas en el transcurso de este y tomar decisiones que minimicen los posibles riesgos. Para simplificar y facilitar la gestión de estos artefactos se pueden encontrar en Internet diversas herramientas orientadas a metodologías ágiles que ofrecen funcionalidad relacionada con la creación de PBIs, priorización y gestión de los distintos artefactos que Scrum desarrolla. El principal problema es que, de las soluciones que se han podido utilizar, pocas, si las hubiera, implementan mecanismos que permitan al PO gestionar un PBCh y realizar predicciones sobre este.

2.3. ENTREGAS, VELOCIDAD Y ESTIMACIONES

En esta sección se centra el análisis en varios de los elementos más relevantes para el Product Owner. En el apartado 2.3.1 se describen las distintas razones para realizar entregas. La sección 2.3.2 desarrolla el concepto de velocidad en un proyecto. Seguidamente, los apartados 2.3.3-2.3.5 desarrollan varios tipos de estimaciones aplicando distintas métricas y métodos de aproximación. El último subapartado desglosa los tipos de velocidad que se pueden encontrar en un proyecto.

2.3.1. ENTREGAS

Un Product Owner profesional sabe que la única forma de crear valor es mediante las entregas [4]. Probar los productos directamente en el mercado es algo fundamental para la agilidad de la organización. Pero las entregas son más o menos importantes y valiosas en función de las razones por las que estas se realizan. En la Figura 2.1 se muestra una lista de las razones para hacer una entrega ordenadas de menos a más importante en términos de agilidad, innovación y satisfacción del cliente.

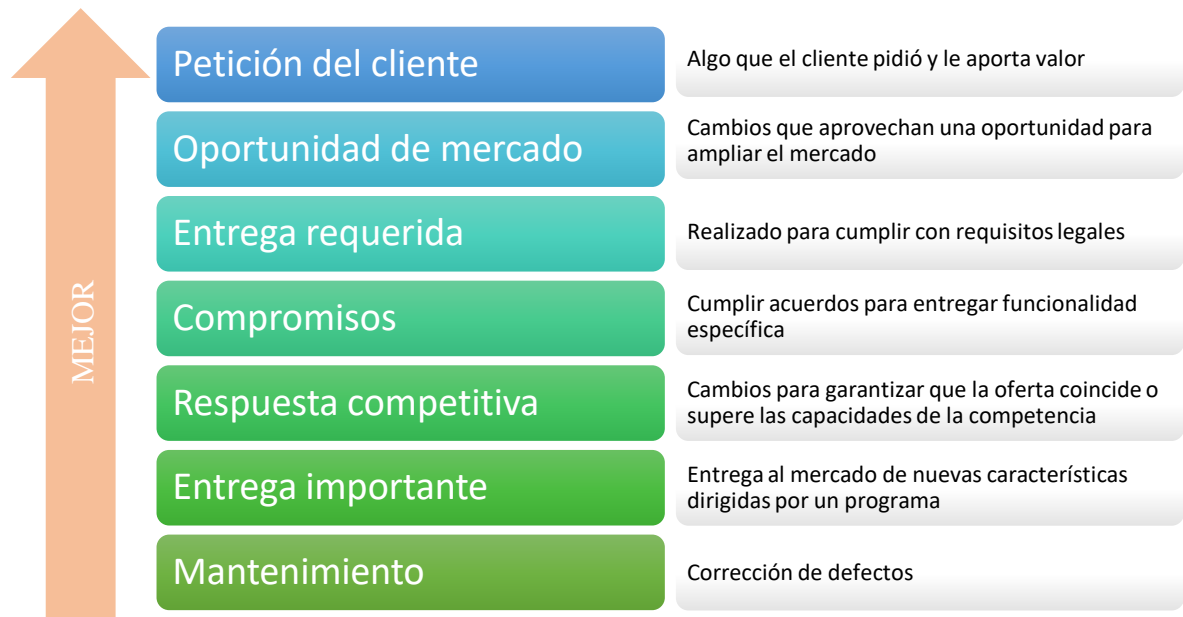


Figura 2.1: Razones para las entregas

Cuanto más se desciende en la lista de la Figura 2.1 menos ágil se es en términos de entrega al mercado, innovación y satisfacción del cliente.

2.3.2. VELOCIDAD

La velocidad es otro factor importante a tener en cuenta. La velocidad representa la capacidad para hacer progresos [4]. Es el factor determinante para saber cuanto tiempo llevará realizar una tarea. Esta velocidad siempre variará dependiendo de las personas implicadas, cambios en las soluciones, conflictos, fallos en el hardware, etc. Además, a la hora de hacer previsiones sobre el trabajo de un equipo, hay que tener en cuenta que la velocidad de un equipo estable es mucho más segura en términos de precisión que la de un equipo recién formado. Eligiendo una buena medida de velocidad se puede dar al cliente estimaciones de gran valor sobre el progreso del trabajo.

Por poner un ejemplo:

- Un equipo de desarrollo estable lleva una velocidad media de 8 PBIs/sprint durante el transcurso del proyecto, suponiendo que la duración de un sprint es de dos semanas. El cliente quiere conocer qué nivel de desarrollo se alcanzará dentro de 6 meses:

$$8 \text{ PBIs/sprint} \cdot 12 \text{ sprints} = 96 \text{ PBIs realizados del Product Backlog}$$

CAPÍTULO 2. Antecedentes y estado de la cuestión

- Inversamente, si el cliente quiere conocer cuánto tiempo llevará realizar 50 items, la fórmula sería la siguiente:

$$\frac{50 \text{ PBIs}}{8 \text{ PBIs/sprint}} \cdot 2 \text{ semanas/sprint} = 12,5 \text{ semanas}$$

Esta forma de utilizar la velocidad para realizar cálculos sobre el proyecto es demasiado sencilla para ser fiable por sí sola. Se deben tener en cuenta 3 factores de riesgo:

1. La falsa sensación de seguridad que pueden generar este tipo de fórmulas, ya que puede inducir al cliente la sensación de precisión y exactitud cuando realmente no la hay.
2. La falta de datos empíricos que ocurre cuando los equipos de trabajo son nuevos. En este caso ninguna fórmula servirá para predecir situaciones.
3. Los tamaños variables entre las formas de medir los ítems del Product Backlog. Cada equipo posee una forma diferente de medir, por lo que cada ítem puede requerir un esfuerzo muy distinto.

Todos estos factores se relacionan con la transparencia. El objetivo de utilizar estas fórmulas es aplicarlas iteración tras iteración mientras se obtienen más datos. Así, las previsiones y planes se ajustarán conforme la situación cambie.

A la hora de lidiar con los tamaños variables de los PBIs existen varias alternativas. Un modelo sencillo consiste en asignar un peso a cada PBI en función de su estimación. Una secuencia popularmente utilizada para estas estimaciones es la sucesión de Fibonacci. Aunque los números de la secuencia pueden ser cualesquiera, se utiliza esta sucesión en concreto debido a que la diferencia entre un número y el siguiente crece de forma gradual. Esto se adapta perfectamente al hecho de que, a más grande la estimación, más inexacta se vuelve esta.

Utilizando estas técnicas se pueden crear planes de entrega con fechas sin estimar los PBIs en unidades de tiempo, ya sean horas, días o semanas. Los estudios muestran que los humanos son bastante buenos comparando cosas (resulta más sencillo comparar dos edificios

que estimar la altura de uno solo). Aprovechando esta habilidad y la experiencia se puede mejorar la precisión de las estimaciones. En un estudio realizado por Rally [5], en el que se analizaron más de 70.000 equipos de Scrum, se deduce que los equipos que peor rendimiento tenían realizaban las estimaciones en horas, seguidos por los que no realizaban ninguna estimación. Los equipos con mejor rendimiento utilizaban estimaciones por puntos relativos. Conociendo esto, se puede concluir que no hacer estimaciones resulta menos perjudicial que realizar estimaciones en horas o días [6].

La velocidad debe ser considerada una medida neutral dedicada exclusivamente al equipo Scrum. Aunque la transparencia en el progreso general es importante, estos valores numéricos no deben considerarse significativos para interesados externos al equipo.

2.3.3. GENERACIÓN DE INFORMES

Un Product Backlog bien mantenido contiene toda la información necesaria para realizar informes. Debido a que Scrum da la posibilidad de completar incrementos del producto con valor en cada sprint, se abre un amplio abanico de posibilidades en torno a la generación de informes. El Equipo de Desarrollo y el Product Backlog serán suficientes para realizar la generación de informes mientras este último se mantenga constantemente refinado y sea único para el producto. El gráfico de la Figura 2.2 muestra el progreso a través de los sprints para un producto.

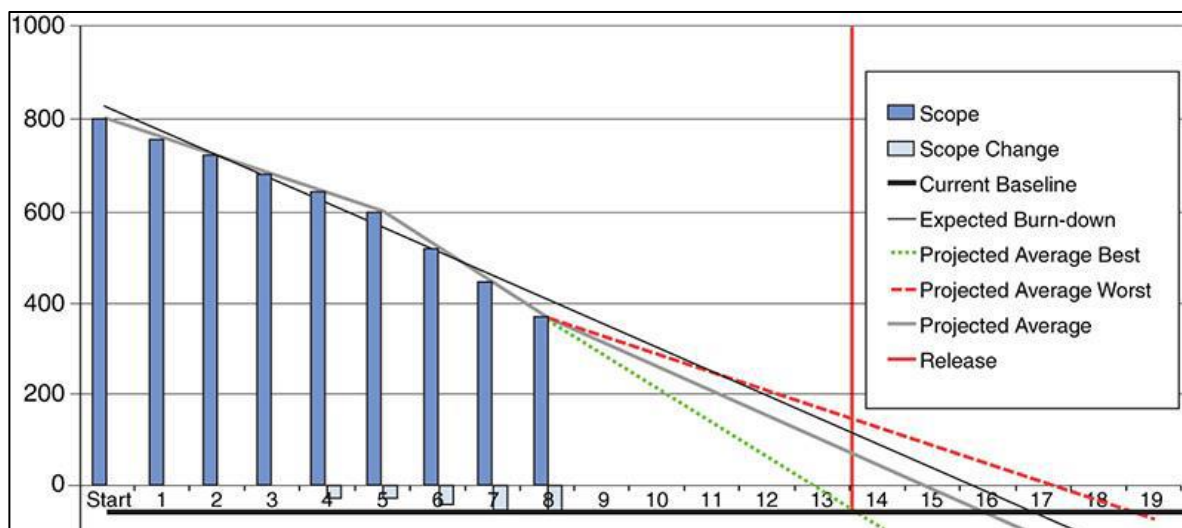


Figura 2.2: Gráfico burn-down de entregas con predicciones

La fina línea negra representa la tendencia de quemado esperada (Expected Burn-down), basada en la regresión de mínimos cuadrados [7] sobre todos los puntos conocidos (sprints). La gruesa línea gris representa la velocidad promedio extrapolada (Projected Average) del último punto de datos conocido (último sprint). Las líneas discontinuas son, respectivamente, el promedio de las 3 velocidades más altas y bajas de los sprints. La extensión de estas líneas se conoce como “Cono de incertidumbre”. A más lejos se mire en el futuro, menos precisas serán las predicciones. Por esto, se debe utilizar este cono como vehículo para recordar a los Stakeholders la existencia de la incertidumbre y la carencia de la precisión.

Estudiando el ejemplo de la Figura 2.3, la entrega está programada al final del sprint 13. Ya que el gráfico se creó tras el sprint 8, el equipo tiene cinco sprints para completar el Product Backlog. Las opciones que se pueden llevar a cabo son las siguientes:

- **Cambiar la fecha de la entrega:** hay que tener en cuenta implicaciones del cambio de fecha, planes de contingencia, etc.
- **Incrementar la velocidad para cumplir el alcance:** sopesar la inclusión de nuevos desarrolladores, eliminar distracciones de estos, mejorar las infraestructuras y herramientas o incluir otros equipos en el desarrollo.
- **Cumplir con el objetivo:** analizar lo absolutamente necesario para la entrega (Minimum Viable Product).

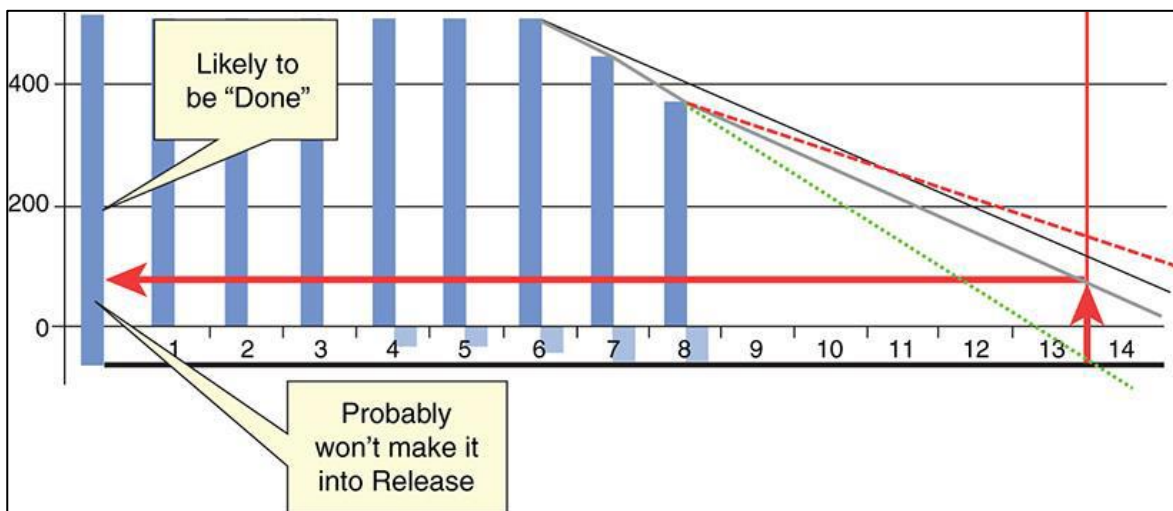


Figura 2.3: Proyección del alcance basada en el burn-down chart

Inspeccionando el burn-down chart de la entrega se puede concluir que la línea gris es una previsión razonable. Trazando la línea roja vertical desde la fecha final hasta que cruce la línea de previsión se puede estimar qué franja del alcance será completada a tiempo y cuál no. Esta valiosa información permite al PO tomar decisiones importantes acerca del alcance del producto. Por ejemplo, se podría simplificar parte de la funcionalidad que seguramente sea completada para dejar espacio a características que se estimaron que no llegarían a realizarse.

En general, los planes de entrega de este tipo son valiosos debido a la visualización y la capacidad de comunicar el nivel de incertidumbre existente (transparencia). Así mismo, permiten al Equipo Scrum y a todos los interesados enfrentar la realidad de su situación de forma temprana (inspección) para luego comenzar a hacer planes de ajuste (adaptación).

Comparar entre diferentes Equipos Scrum se vuelve más complicado. Cada equipo tiene su propia escala para estimar el Product Backlog, lo cual hace que comparar equipos por velocidad no sea buena idea. Una opción viable es asumir que cada equipo estima de una forma distinta, y centrarse en la pendiente de la velocidad como medida del progreso. Siendo independiente la escala de medida de cada equipo, la pendiente sí es comparable. Utilizando gráficas que representen estas pendientes y eliminando unidades de medida se puede analizar el progreso entre distintos equipos y tomar decisiones de gestión relacionadas con, por ejemplo, la distribución del trabajo entre equipos.

2.3.4. PORCENTAJE DE COMPLETITUD

Una forma de medida fácil de entender, aunque no siempre sencilla de calcular, es el Porcentaje de Completitud (PoC). Para llevar a cabo esta medida se debe conocer el número y peso de todos los Product Backlog Items. Conociendo estos y sabiendo cuántos se han completado hasta el momento, se puede realizar una estimación del porcentaje de finalización actual. Para más interés del cliente, se pueden dividir estos porcentajes en los módulos que compongan el producto. De esta forma el cliente puede saber en qué nivel de desarrollo está cada componente del producto final. Estos informes pueden ser de gran interés si se presentan durante las Sprint Reviews.

2.3.5. SIMULACIÓN DE MONTE CARLO

La simulación de Monte Carlo es una técnica de mitigación de riesgos para problemas que requieren una respuesta numérica pero que son demasiado complejos para resolverse de forma analítica [4]. Para abordar este problema, se aplican números aleatorios y estadística para explorar la probabilidad de cada posible resultado de una decisión, desde un extremo al otro.

Esta estimación a los Product Backlogs funciona bajo la premisa de que cada Product Backlog Item se encuentra bajo un rango de estimación entre pesimista y optimista. Esto se repite en todos los elementos del Product Backlog y los valores aleatorios se suman. Esta suma resultante representa un punto en el eje X (ver Figura 2.4). El eje Y representa la frecuencia de sumas con el mismo valor. Esta simulación se realiza por lo menos 10.000 veces. La gráfica resultante es una distribución en el tiempo en la que el área es la probabilidad de que el Product Backlog se complete en ese tiempo.

Además, hay 2 puntos de interés en el eje X o del tiempo: e_{50} y e_{95} , que son respectivamente los tiempos en los que hay un 50% y un 95% de posibilidades de completar el Product Backlog.

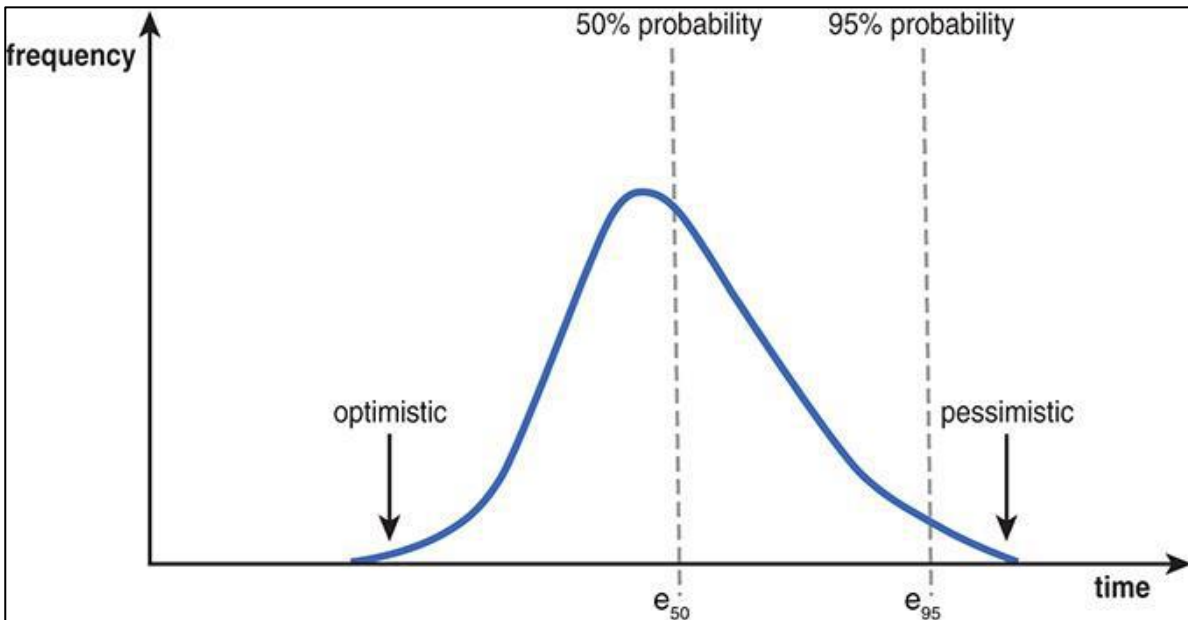


Figura 2.4: Distribución de probabilidades basada en la simulación de Monte Carlo

2.3.6. COMPOSICIÓN DE LA VELOCIDAD

Conforme el tiempo avanza, la velocidad de un equipo puede mantenerse estable, pero eso no implica que así lo haga el valor aportado al producto. Factores como la ratio de innovación del equipo pueden afectar al rendimiento en el tiempo. Para visualizar esta problemática se pueden categorizar los PBIs y la velocidad resultante como se muestra en la Figura 2.5:

- **Nuevas características:** son positivas. Generan valor atrayendo nuevos clientes o manteniendo los anteriores.
- **Deuda tecnológica:** no tan positiva, debido a que aparece a causa de elegir soluciones sencillas que deben ser modificadas a largo plazo. Aún así, las malas decisiones del pasado se deben solucionar para evitar mayor trabajo en el futuro y poder centrar esfuerzos en nuevas características.
- **Infraestructura:** es trabajo que no forma parte de nuevas características ni deuda tecnológica. Por ejemplo, configurar un cluster o actualizar entornos.
- **Errores:** siempre son negativos. Se manifiestan debido a la incapacidad de entregar un producto de alta calidad.

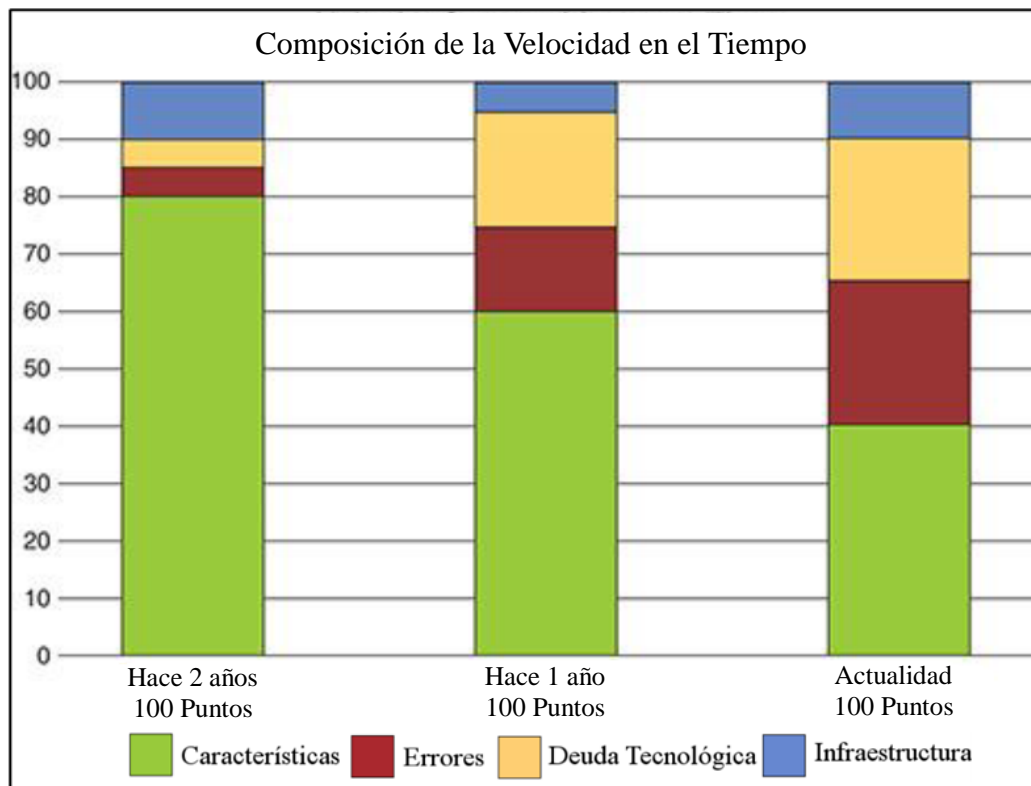


Figura 2.5: Composición de la Velocidad con el Tiempo

CAPÍTULO 2. Antecedentes y estado de la cuestión

En el ejemplo de la Figura 2.5, el Equipo de Desarrollo sigue trabajando igual de bien que hace 2 años. Sin embargo, el valor entregado por este ha disminuido sustancialmente. Las nuevas características entregadas han disminuido en 40 puntos, mientras que la deuda tecnológica y los errores han crecido.

Lo importante de esta información es controlarla a lo largo del tiempo, ya que puede ser un buen indicador de que algo no anda bien en el transcurso de un proyecto o, al contrario, de que se está siguiendo un buen ritmo de trabajo y entrega de valor.

2.4. HERRAMIENTAS Y SERVICIOS

En Internet se puede encontrar una gran variedad de aplicaciones que facilitan el trabajo y gestión en un equipo que aplique metodologías ágiles como Scrum. Se analizarán los tipos de aplicaciones, así como algunas de las herramientas más usadas para intentar encontrar los puntos fuertes de estas que las hacen realmente útiles a sus usuarios.

Las aplicaciones de gestión de metodologías ágiles se pueden agrupar en 2 conjuntos claramente diferenciados:

- **Aplicaciones de escritorio:** las cuales el usuario descarga e instala en su dispositivo, siendo estas cada vez menos utilizadas debido a la poca flexibilidad que proporcionan, la alta compatibilidad que requieren y los problemas que pueden generar a los usuarios.
- **Servicios web:** son las aplicaciones online que se acceden a través de navegador. La gran mayoría de aplicaciones que triunfan actualmente son servicios online, debido a la facilidad con la que los usuarios pueden acceder a estos, pudiendo tener la misma funcionalidad en la nube que una aplicación de escritorio, pero evitando cualquier tipo de instalación.

Del inmenso número de aplicaciones de gestión ágil y Scrum que hay en el mercado podemos destacar varias de las más utilizadas (en la Figura 2.6 se muestran los logos de cada una):

- **ZenHub** [8]: además del Kanban básico incluye vistas del PBCh, de gestión de la velocidad del proyecto, entregas y de flujo acumulativo de las historias de usuario

del producto. Si bien dispone de varias opciones sobre informes, siendo esto una novedad sobre otras herramientas del mercado, la herramienta principal de que dispone el PO para realizar estimaciones es la línea de quemado ideal del PBCh, que se autogenera con el gráfico de quemado y no aporta gran información. Otro de los informes que presenta es el Informe de Entregas y el de Velocidad.

- **Jira** [9]: permite la gestión de Kanban e historias de usuario o incidencias, como así las llaman en esta herramienta, así como una hoja de ruta que permite gestionar de forma temporal las épicas del producto. Carece de predicciones y vistas del PBCh y no dispone de versiones gratuitas de la misma.
- **Yodiz** [10]: una herramienta bastante completa que incluye un gran número de opciones tanto gráficas como roadmaps y flujos cumulativos. Aporta gran información sobre el estado del proyecto y las velocidades media, máxima y mínima. Aún así, sigue careciendo de opciones más flexibles a la hora de realizar predicciones sobre el proyecto y los sprints.



Figura 2.6: Logos de Zenhub, Jira y Yodiz

Cabe remarcar que entre todas las herramientas siempre hay varios campos que se mantienen dentro de las historias de usuario, como: una descripción, un responsable, etiquetas y relación con otras historias. Un punto interesante es la opción de diferenciar entre el informador de la incidencia/historia de usuario y el responsable de realizarla.

Como se ha visto en el apartado 2.3, el PO puede hacer uso de los artefactos que proporciona Scrum para obtener información valiosa sobre el estado de un proyecto. Varias utilidades se encuentran en la velocidad y en la generación de informes de predicciones. Aquí se reúnen varias de las técnicas de estimación y previsiones descritas en dicho apartado. Pero es curioso

CAPÍTULO 2. Antecedentes y estado de la cuestión

ver que, del conjunto de herramientas más utilizadas del mercado, muy pocas incorporan módulos de estimación más allá de la línea básica que estima la finalización del Product Backlog. Y lo que es peor (para el PO y, por consecuencia, para el proyecto) ninguna de las aplicaciones encontradas llega a aportar opciones avanzadas de estimación o generación de previsiones. Esto limita el flujo de información y, por ende, el rendimiento y valor de las entregas del producto de forma sustancial. ZenHub, por ejemplo, amplía los informes que presenta, pero incluye algunos aspectos que no se adecúan al uso que se hace de las metodologías ágiles.

Su informe de entregas obliga al usuario a crear entregas con fecha de inicio y fin, para poder generar el informe conveniente. Esto es bastante incómodo ya que en muchas ocasiones el proyecto no trata las entregas con la importancia que le da esta aplicación. Cuando los despliegues se realizan a diario esta utilidad pierde su sentido. Otra inconveniencia es la imposibilidad de tener un PBCh, esto se debe a que ZenHub utiliza Milestones, el equivalente a los sprints en Scrum. Además, limita la generación de reportes a un solo Milestone. Esto impide ver el progreso durante el proyecto salvo que se utilice un único Milestone para todo el proyecto.

2.5. CONCLUSIONES

A lo largo de este capítulo se ha situado el centro del análisis en la figura del PO, resumiendo sus funciones y objetivos en el marco de Scrum. Se ha puesto en evidencia la importancia de la gestión de las entregas, en especial para el cliente, a la vez que se introducía el término de la velocidad, pues esta será una de las métricas utilizadas para la generación de informes. Tal y como se describe en el capítulo, hay gran variedad de proyecciones entre las que elegir: desde predicciones por velocidad, pasando por porcentaje de completitud y llegando a utilizar simulaciones estadísticas como el método Monte Carlo. Por último, se hace hincapié en el análisis de la composición de la velocidad. Dicho análisis permite detectar si el desarrollo tiende a aportar un mayor o menor valor al cliente.

El conjunto de todas las métricas y técnicas descritas hace esperar que existan herramientas o servicios que proporcionen uno o varios de estos informes, que han probado ser de gran utilidad para la gestión satisfactoria de un proyecto. No obstante, analizando el mercado de

aplicaciones y herramientas se puede detectar que pocos son los servicios que proveen alguna de las predicciones mencionadas y prácticamente ninguna aplicación presenta una fuente robusta de informes y métricas. Esta situación lleva a plantear la necesidad de diseñar un servicio que aporte a un gestor de proyectos y, en definitiva, a un PO, las herramientas que tan valiosas son para la consecución de los objetivos de un proyecto y, por ende, para su éxito.

CAPÍTULO 3. METODOLOGÍA Y DESARROLLO

“Walking on water and developing software from a specification are easy if both are frozen” – Edward V Berard.

3.1. INTRODUCCIÓN

Una vez se ha puesto en contexto la problemática presente en torno a la figura del PO y sus necesidades y, tras analizar las herramientas existentes, se procede a documentar el método de organización, gestión y planificación del desarrollo de la solución. Con este fin, se introduce en la sección 3.2 la metodología seguida en el desarrollo del trabajo, profundizando en las modificaciones que se han tenido que tomar para adaptar esta al desarrollo de un Trabajo Fin de Grado. La sección 3.3 describe la arquitectura y tecnologías que se han utilizado para facilitar la comunicación y planificación, así como el desarrollo. La sección 3.4 incluye las especificaciones del sistema. Las herramientas y el pipeline se describen en el apartado 3.5. Seguidamente, en la sección 3.6 se incluyen los artefactos del desarrollo, finalizando este capítulo con las conclusiones.

3.2. METODOLOGÍA

La metodología seguida durante el desarrollo del proyecto se basa en Scrum [1]. Para adaptar las necesidades del proyecto a este marco de trabajo, se aplican varios de los conceptos y artefactos de Scrum con ciertas restricciones.

Roles

En Scrum, los equipos de desarrollo están formados por entre 3 y 9 personas. En este contexto, el equipo lo forma una única persona: el estudiante autor del trabajo.

Los roles que se identificarán en la metodología aplicada serán los siguientes:

- *Product Owner*: para maximizar el valor entregado al interesado del proyecto, que

es el Tribunal encargado de revisar y evaluar dicho trabajo, el cargo de PO lo formarán ambos directores. Esto se debe a que ambos conocen en mayor profundidad los requisitos que deben cumplir tanto la memoria como el producto software para que el proyecto se desarrolle de manera satisfactoria. Este rol traspasará la gestión del Product Backlog al Equipo de Desarrollo, y será dicho equipo el que se encargue de crear los Product Backlog Items, así como priorizarlos y gestionarlos en general.

- *Scrum Master*: este rol lo cumplen los directores del trabajo, debido a que son los encargados de que se aplique Scrum correctamente y a su amplio conocimiento en este campo. Además, al igual que la figura original del Scrum Master, tienen la tarea de eliminar los impedimentos que vayan surgiendo.
- *Equipo de Desarrollo*: está formado única y exclusivamente por el estudiante y autor del trabajo.

Eventos

Los eventos y reuniones típicos de Scrum se mantendrán idénticos en su mayor parte, salvo algunas adaptaciones necesarias.

- *Sprints*: el desarrollo del proyecto se dividirá en sprints. Estos tendrán una duración aproximada de 2 semanas, pudiendo reducir o alargar dicho plazo si fuera necesario. Cada sprint tendrá asociado un objetivo específico del sprint, que definirá de forma breve los puntos clave a cumplir en dicho evento.
- *Reunión de planificación, revisión y retrospectiva*: se reúnen los directores y el estudiante una vez con el objetivo de finalizar un sprint e iniciar el siguiente para tratar los siguientes temas en este orden:
 - Revisión de los resultados del sprint finalizado → Revisión.
 - Tratar problemas encontrados en el desarrollo y alguna posible mejora → Retrospectiva.
 - Planificar las tareas a realizar para el siguiente sprint → Planificación.
- *Scrum diario*: la reunión diaria del equipo de desarrollo en la que se resumen los avances del trabajo se suprime por estricta necesidad, ya que el equipo de trabajo lo forma una única persona.

Artefactos

- *Product Backlog*: siendo responsabilidad del PO, delegará la gestión de este al Equipo de Desarrollo, que se encargará de priorizar y refinar los PBIs que se vayan generando.
- *Sprint Backlog*: no se actualiza diariamente, debido a que el Equipo de Desarrollo está formado por una única persona. No obstante, siguiendo los valores de Scrum de transparencia, inspección y adaptación, se mantiene online a disposición del PO.
- *Incremento*: fruto del desarrollo se originan dos incrementos: incremento del producto software e incremento de la memoria del trabajo.

3.3. TECNOLOGÍAS Y ARQUITECTURA

La arquitectura de la aplicación gira en torno a tres tecnologías que se han elegido por las capacidades y beneficios que aportan al proyecto. Estas tecnologías se dividen en los siguientes campos: Front-end, Back-end y Base de Datos. Para cada uno de estos componentes de la arquitectura se justifica la elección de la tecnología asociada.

Back-end: De entre las diversas tecnologías que soportan el lado del servidor, como puedan ser Java, Python o PHP, se ha decidido utilizar Node.js. Las razones son simples: Node.js [11] es un entorno de ejecución asíncrono y orientado a eventos que ejecuta el código mediante un intérprete en tiempo real. Se programa en lenguaje JavaScript, lo cual facilita las tareas de desarrollo, al tener un paradigma y lenguaje similar al Frontend. Este diseño sin hilos facilita la concurrencia del sistema debido a que no bloquea los procesos existentes. Esto, junto a la alta escalabilidad que se podría alcanzar si se implementasen clústeres, proporciona una flexibilidad que otras tecnologías no alcanzan a aportar. Además, proporciona un gestor de módulos que permite importar gran cantidad de librerías externas.

Para generar las conexiones que se realizarán con el cliente se ha utilizado el framework Express [12], por su flexibilidad y facilidad de uso a la hora de diseñar infraestructuras en aplicaciones web.

Base de Datos: Los datos que se deben almacenar en el sistema están bien definidos y estructurados, así como las relaciones entre estos. Es por esto que se decide utilizar un modelo relacional de bases de datos. El gestor de bases de datos escogido es MySQL [13]. Esto se debe a la extensa documentación e integración que aporta este gestor.

Front-end: La tecnología del lado del cliente es Angular 8 [14]. Angular aporta una estructura de directorios y modelo de trabajo que, por un lado, facilita el desarrollo modular y extensible del código y, por otro, gestiona de manera eficiente las acciones y eventos que debe ejecutar el navegador web.

El código se escribe en lenguaje TypeScript [15], el cual es una versión tipada de JavaScript que simplifica las tareas de desarrollo y posteriormente se compila a JavaScript para que lo pueda interpretar cualquier navegador. La versión 8 de Angular no es la última versión, pero sí es de las más actuales y dispone de la estabilidad y características suficientes para satisfacer el desarrollo del cliente web.

Estas tecnologías se comunican entre sí siguiendo los principios de las arquitecturas ReST [16]. El Back-end proporciona una interfaz uniforme de comunicación con el cliente vía HTTP sin guardar ningún contexto, sesión o petición de este (*stateless*). La comunicación se realiza mediante una API que devuelve al Front-end los datos requeridos en formato JSON. Estos datos son manipulados y enviados al cliente tras realizar una consulta en la base de datos. Esta arquitectura se caracteriza por un gran desacople entre el cliente y el servidor, permitiendo que cada componente se implemente independientemente del resto sin tener más relación que el servicio de APIs. La Figura 3.1 muestra de forma gráfica un esquema de esta arquitectura.



Figura 3.1: Arquitectura general

3.4. ESPECIFICACIONES

En esta sección se reúnen los diagramas y especificaciones generadas que describen varios aspectos de la estructura del sistema. Es importante remarcar que los diagramas descritos conforman la versión final de estos, habiendo sufrido diversos cambios durante la realización de los sprints.

3.4.1. MODELO DE CLASES

La Figura 3.2 muestra el diseño del modelo de clases general del sistema. En dicho modelo se representan las clases y atributos del sistema, así como las relaciones entre estas. Es importante remarcar que este modelo es conceptual y que no se implementa al pie de la letra en el código por los siguientes motivos: en el cliente se utilizan modelos reducidos para facilitar la representación de estos en las vistas; en el lado del servidor, al utilizar un lenguaje no tipado se prescinde de la definición de clases, dejando el tipado de atributos y relaciones en manos de la base de datos.

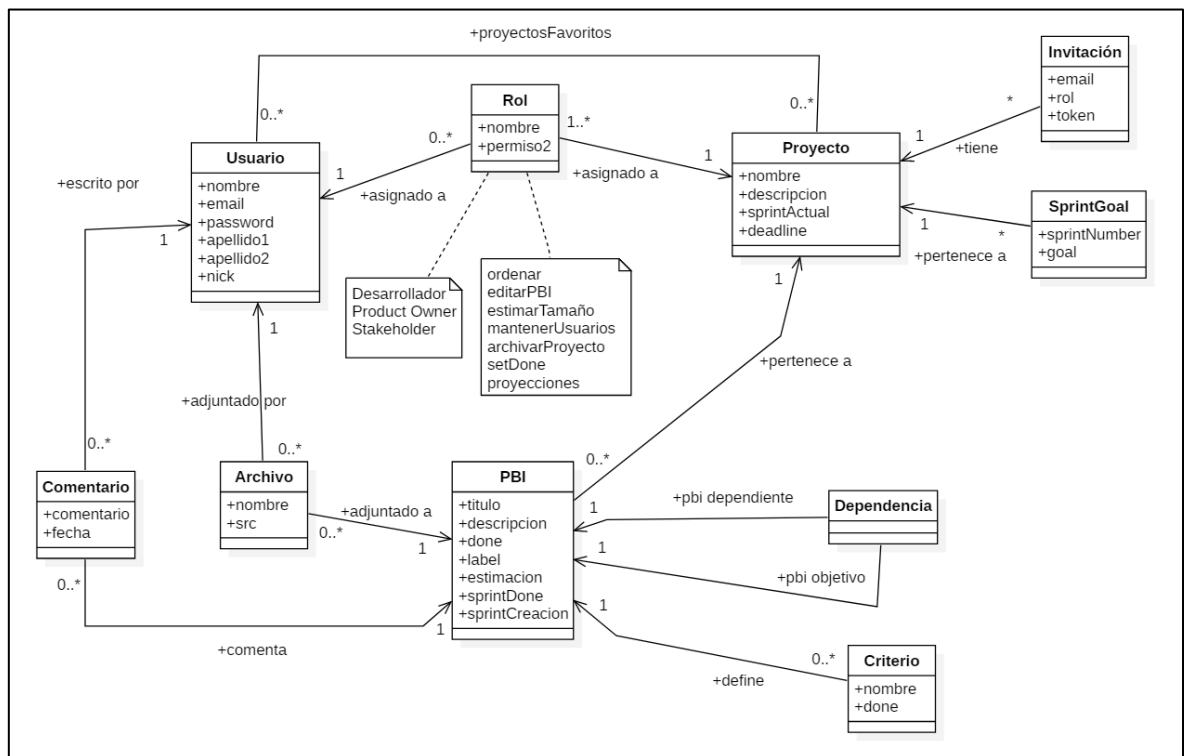


Figura 3.2: Diagrama de clases

3.4.2. BASE DE DATOS

Para definir la estructura de la base de datos se utiliza un diagrama Entidad-Relación, el cual se muestra en la Figura 3.3. Para generar esta estructura en la base de datos se incluye en el repositorio un script de generación de tablas dentro del servidor, concretamente en el fichero *backend/db/seeder.js*.

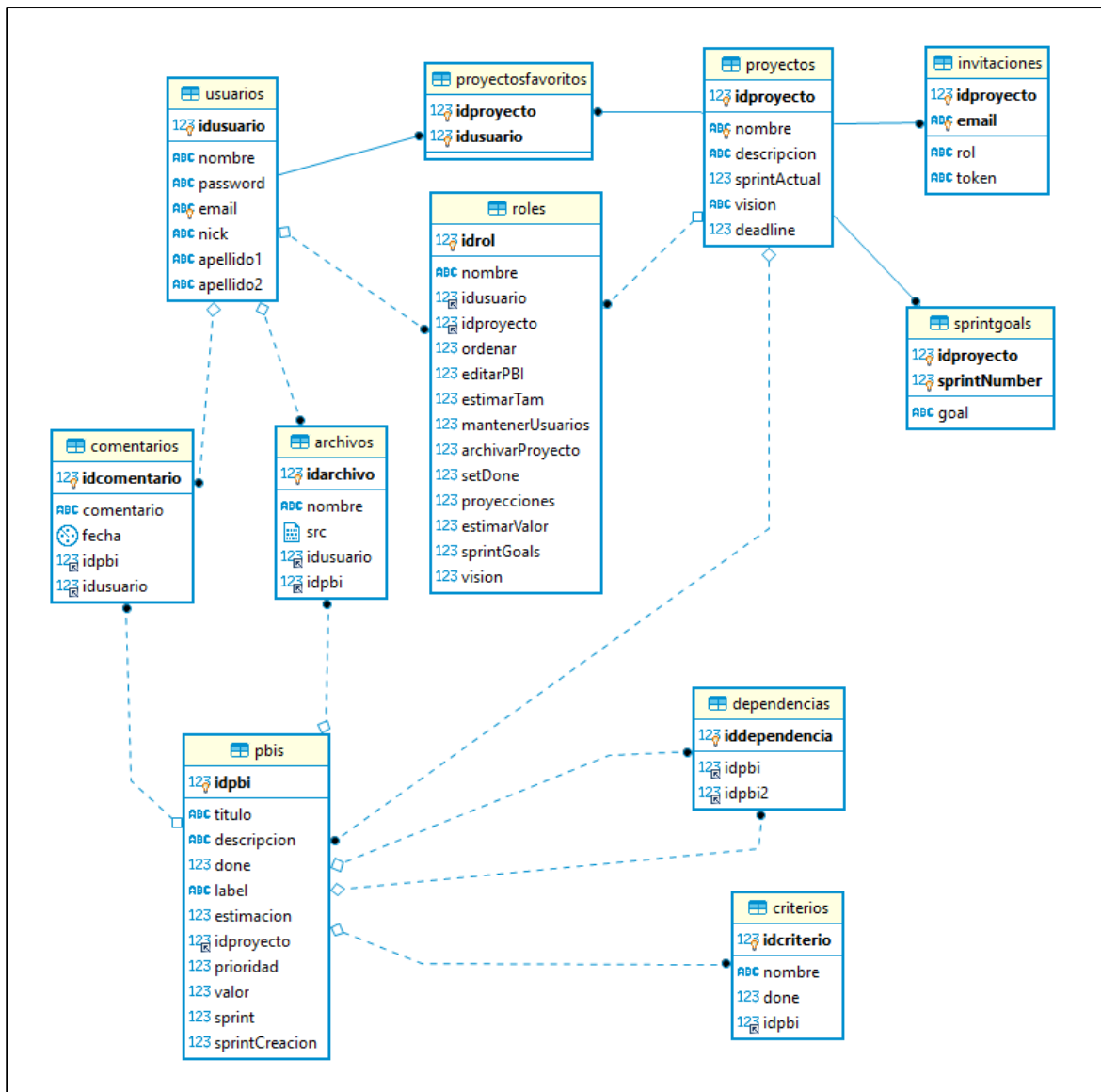


Figura 3.3: Diagrama Entidad-Relación

3.4.3. DESPLIEGUE

El despliegue de la arquitectura en Heroku se muestra en la Figura 3.4. El cliente, utilizando la web desde un navegador, se comunica mediante el servicio que se requiera según el contexto de uso. En el diagrama se abrevian los artefactos incluyendo únicamente los relacionados con *proyectos*, siendo esta estructura similar para cualquiera de los restantes artefactos de acceso a datos (usuarios, PBIs, etc...).

El cliente envía una petición HTTP generada por el servicio correspondiente. Esta es interceptada por el artefacto *server.js*, que se encuentra a la escucha de peticiones en el puerto que se haya configurado. A su vez, este artefacto delega el manejo de esta petición en un gestor de rutas (*routes/api.js*) que, en función de la url recibida en la petición, vuelve a delegar la petición al artefacto correspondiente (*routes/proyectos.js*). Este artefacto verifica que los parámetros de la petición son correctos y vuelve a delegar la petición al controlador requerido (*controllers/proyectos.js*). El controlador se encarga de realizar el acceso a la base de datos. Tras obtener una respuesta, se adjunta esta a la petición HTTP, enviándola de vuelta al cliente desde el fichero de ruta correspondiente.

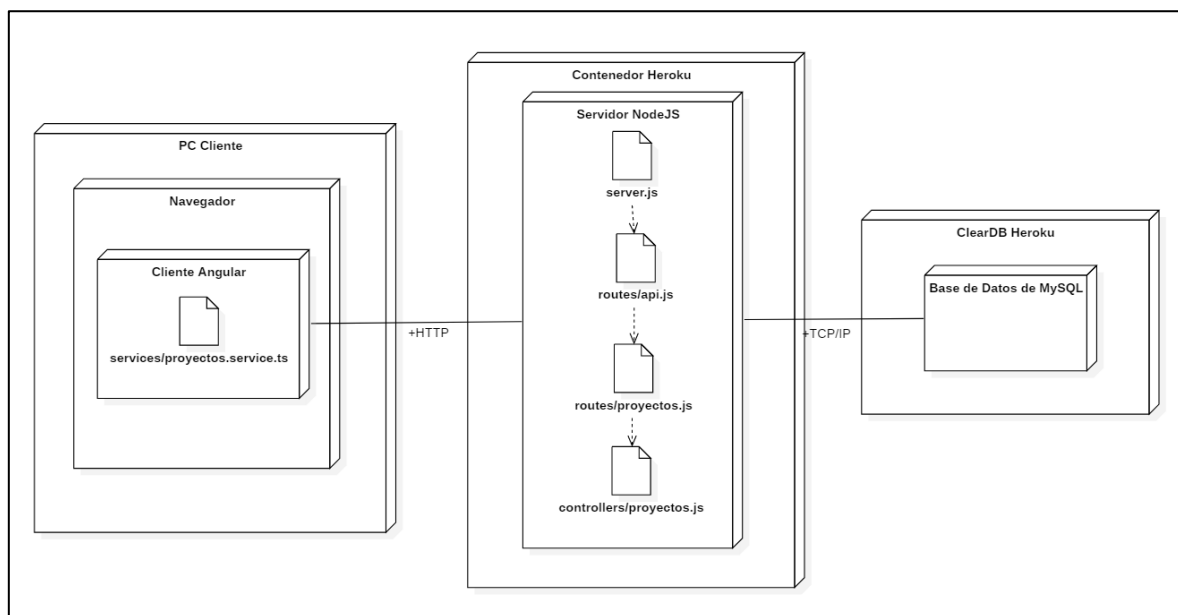


Figura 3.4: Diagrama de despliegue

3.5. PIPELINE Y HERRAMIENTAS

Una de las prácticas que DevOps [2] propone en su Primera Vía es la “Aceleración del flujo”. Para aumentar la velocidad y progresos en el desarrollo de un producto, se deben automatizar los pasos a seguir desde que se implementa el software hasta que se lanza a producción, pasando por las distintas fases de pruebas y compilaciones. Si se consiguen eliminar las barreras entre las tareas normalmente asociadas al Desarrollo y las asociadas al equipo de Operaciones, se genera un pipeline fluido y automático. Esto permite, no solo acelerar el flujo de desarrollo, sino evitar fuentes comunes de errores, que se suelen encontrar en la cantidad de traspasos que sufre el código. Otro punto a favor es que facilita las labores de despliegue, por lo que se puede alcanzar una mayor tasa de despliegues en el tiempo.

Para llevar a cabo un pipeline similar al mostrado en la Figura 3.5, se han integrado varias de las herramientas utilizadas en el desarrollo, las cuales se describen en la Tabla 3.1. El código almacenado en el repositorio de GitHub [17] se enlaza con el servicio de TravisCI [18] mediante una configuración previa y un archivo de código que configura esta integración. Al realizar un push al repositorio, este a su vez lanza la ejecución de una máquina virtual en TravisCI, que ejecuta el script de configuración anterior. Esto permite ejecutar pruebas de forma automática (Integración Continua) para cada commit, y saber si estos cambios funcionan correctamente o no.

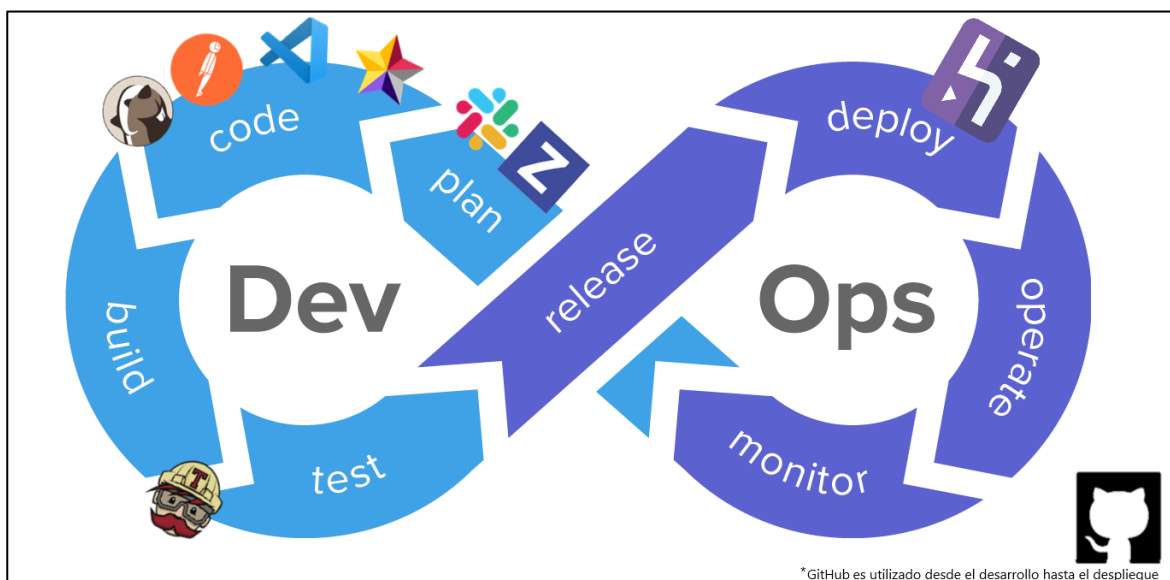


Figura 3.5: Pipeline de DevOps

Adicionalmente, se puede comprobar la cobertura de código de las pruebas del cliente web de forma manual, utilizando la opción `--code-coverage` al ejecutar las pruebas unitarias. La última versión del cliente tiene un nivel de cobertura de 32,6%.

A su vez, el fichero de configuración y el repositorio de GitHub tienen enlazada una aplicación en Heroku [19]. De esta forma, cuando TravisCI ejecuta las pruebas de forma satisfactoria, encadena un despliegue automático (Despliegue Continuo) en la aplicación de Heroku, actualizando el contenido de la aplicación cada vez que un commit es realizado con éxito.

Para que los directores del proyecto estén al tanto de las actualizaciones de contenido de la web, TravisCI también integra el servicio de notificaciones de Slack [20]. Cada vez que un despliegue nuevo es realizado con éxito, un bot de Slack notifica de un nuevo despliegue en el canal (ver Figura 3.6).

Para coordinar el desarrollo, además de las herramientas mencionadas en el sistema de integración del código, se han utilizado otras herramientas y servicios que facilitan la comunicación entre los directores y el estudiante. El listado de servicios se especifica en la Tabla 3.1.

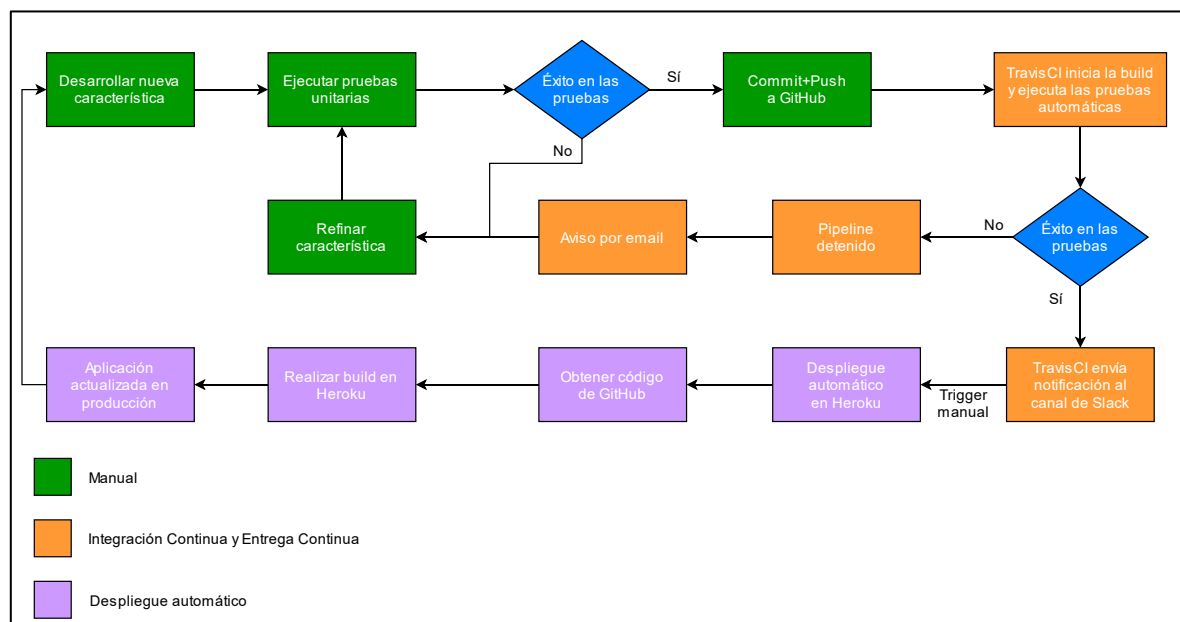


Figura 3.6: Pipeline de Integración Continua, Entrega Continua y Despliegue

Nombre	Icono	Descripción
Slack [20]		Aplicación de escritorio y móvil para la comunicación de equipos de trabajo mediante la cual se mantiene una comunicación constante entre los miembros. Muy útil tanto para resolver dudas y proponer fechas de reuniones, como adjuntar ficheros y documentos varios.
MS Teams [21]		Debido a las dificultades de atender en persona a algunas de las reuniones programadas, se ha utilizado esta herramienta para mantener conferencias online, como sustituto de las presenciales.
MS Word [22]		Ha sido el editor de texto elegido para realizar el desarrollo de la memoria del trabajo, debido a la comodidad que el estudiante tiene utilizando dicha herramienta.
Mendeley [23]		Es una biblioteca y gestor de bibliografías y citas que facilita la inserción de estas en documentos. Adicionalmente, cuenta con integración en Word.
StarUML [24]		Una herramienta de modelado de diagramas UML muy simple y completa, ha sido elegida para el proyecto por su eficacia y sencillez.
VS Code [25]		Es el editor de código elegido para el desarrollo de la parte tecnológica del proyecto. Gracias a su gran cantidad de plugins e integración y facilidad de uso, reduce el tiempo de programación dedicado.
Postman [26]		Es una plataforma de desarrollo de APIs. Se ha utilizado para probar las APIs desarrolladas durante el proyecto y comprobar su funcionamiento de forma simple. Permite importar las configuraciones para utilizarlas en distintos entornos.
DBeaver [27]		Esta herramienta de gestión de bases de datos es la elegida para todo lo relacionado con la gestión de la base de datos del proyecto.
GitHub [17]		Se utiliza esta plataforma para gestionar el control de versiones y mantener un repositorio online del proyecto. En este repositorio se controlan tanto el código como las versiones, historias de usuario y clasificaciones, además de integraciones con otros sistemas.
ZenHub [8]		Aplicación que facilita la gestión de proyectos con metodologías ágiles. Permite integrarlo con GitHub y trabajar las historias de usuario, así como varios gráficos.
TravisCI [18]		Es un servicio de Integración Continua utilizado para aplicar CI al desarrollo del proyecto.
Heroku [19]		Es un servicio de despliegue de aplicaciones en la nube que, de forma gratuita, te permite subir tus aplicaciones a Internet y hacerlas disponibles a todo el mundo.

Tabla 3.1: Listado de herramientas y servicios

3.6. ARTEFACTOS DEL DESARROLLO

Este apartado resume los artefactos utilizados y generados durante el desarrollo, poniendo en contexto su uso, además de las aportaciones y beneficios de estos al proceso de desarrollo.

3.6.1. REPOSITORIO

El repositorio en el que se almacena el código de la aplicación, pruebas e integración se encuentra ubicado en el siguiente enlace disponible en GitHub:

- <https://github.com/victorperezpiqueiras/TheProductOwnerd>

Se trabaja con desarrollo basado en trunk: todos los cambios son aplicados directamente sobre la rama máster.

El sistema de ficheros y la estructura de directorios inicial se creó utilizando un generador de proyectos de Angular llamado NgxRocket [28], el cual incluye buenas prácticas de programación y diversas librerías y scripts para facilitar el desarrollo. El sistema de archivos se organiza de la siguiente manera:

- *backend*: código del servidor de la aplicación en Node.
- *docs*: documentación sobre varios de los artefactos, técnicas y estándares de programación utilizados.
- *documentación_tfg*: documentos entregables del Trabajo Fin de Grado.
- *reports*: contiene los últimos informes de pruebas y cobertura de código generados.
- *readme_images*: imágenes utilizadas para la descripción del repositorio.
- *wikis*: contiene las definiciones de métodos del API y de las tablas de la base de datos.
- *src*: código del cliente de Angular.
- *.travis.yml*: código de integración continua de TravisCI.
- *readme.md*: fichero con la descripción del repositorio.

Lo primero que se visualiza al acceder al repositorio es el fichero *Readme.md* que, como se muestra en la Figura 3.8, muestra información sobre el estado de la última compilación realizada, los items o PBIs sin resolver y la actividad reciente en el repositorio. También se listan las dependencias principales del proyecto y se proporciona un enlace a cada recurso en la web (ver Figura 3.7). Por último, en la Figura 3.9 se especifica el esquema de etiquetado seguido para clasificar los distintos PBIs según su naturaleza.


Main Dependencies

Backend

- [Node](#)
- [Express](#)
- [Mustache](#)
- [Nodemailer](#)
- [Mysql2](#)
- [Jsonwebtoken](#)

Frontend

- [Angular 8](#)
- [Jasmine](#)
- [Highcharts](#)
- [ML-regression](#)



TheProductOwnerd

build passing open issues 15 commit activity 26/month

Table of Contents

- [Introduction](#)
- [Showcase](#)
- [Main Dependencies](#)
- [Get Started](#)
- [Documentation](#)

Introduction

TheProductOwnerd is a web application focused on giving the Product Owner the tools to successfully manage software projects using an agile methodology. Using this application the Product Owner will be able to:

- Manage the Product Backlog of a project
- Invite new members to collaborate in the Scrum team
- Control the Product Burndown Chart and project specific metrics such as velocity
- Perform forecasting and predictions of the project and team performance

Figura 3.7: Dependencias del proyecto

Figura 3.8: Descripción del repositorio

Documentation

Sprint Reports

Every Sprint has a documentation issue associated to it. The report file is posted inside the issue.

Labelling

Label	Description
feature	New features requested for the application
defect	Bugs
enhancement	Enhancements for application features
flow	Integration and flow related issues
retro	Issues generated in the Scrum retrospective
debt	Technical Debt that has to be paid
memoria	Report related issues
Epic	Epic issues
Sprint 1	Issue selected for Sprint X

Figura 3.9: Tabla de etiquetado de PBIs

3.6.2. PLANIFICACIÓN Y QUEMADO DEL TRABAJO

El quemado de trabajo se recoge desde el inicio del proyecto tras la firma del documento de mutuo acuerdo y la configuración inicial de las herramientas de desarrollo, y fluye hasta la fecha final, que se corresponde con la defensa del proyecto. En la Figura 3.10 se muestra el cronograma planificado de actividades, divididas en tres categorías: Aplicación, que asocia trabajo relacionado con programación de la aplicación web; Documentación, que engloba el desarrollo de los capítulos de la memoria, así como de la presentación del trabajo; y los Hitos, que indican las fechas más relevantes en el calendario del proyecto.

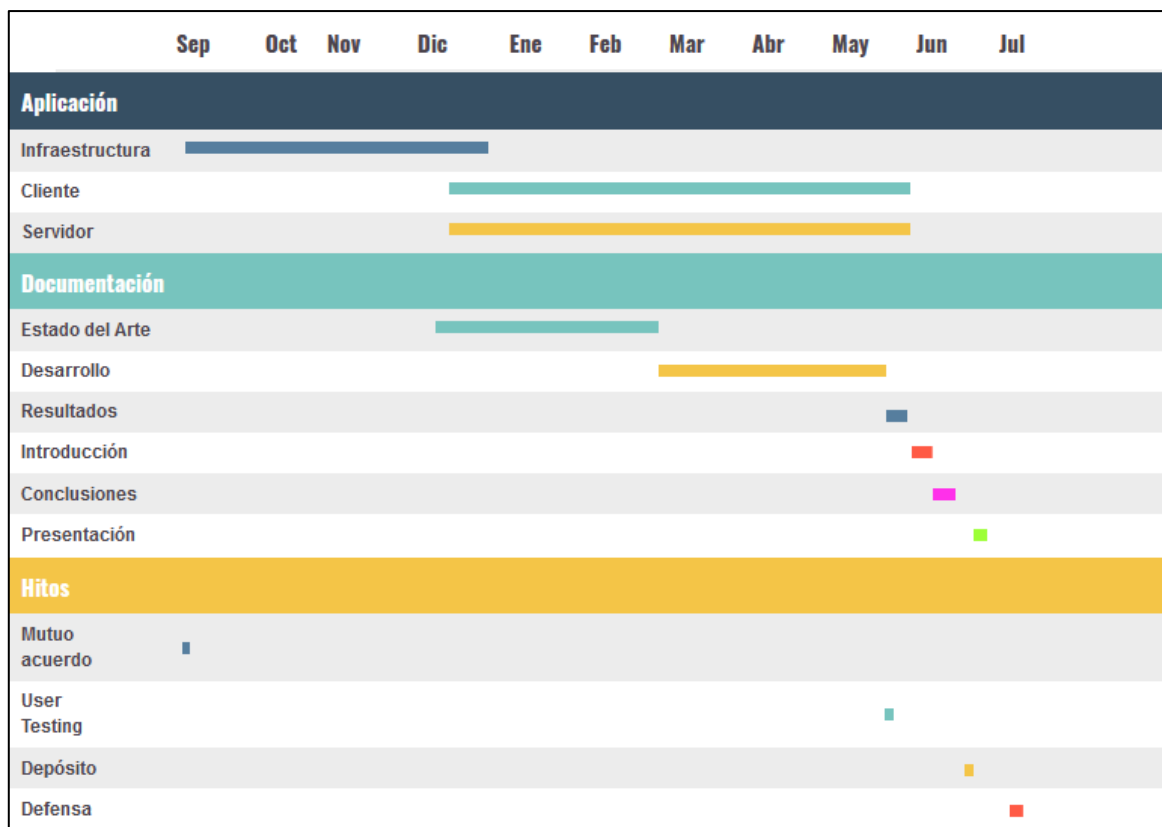


Figura 3.10: Cronograma de alto nivel

El PBCh mostrado en la Figura 3.11 resume el trabajo total registrado en la aplicación de ZenHub. Como se puede observar en las fechas tempranas, el tamaño del trabajo pendiente es bastante alto y desciende poco hasta fechas cercanas a marzo. Esto se debe a que los primeros sprints tuvieron poco peso debido a la alta carga del trabajo del estudiante. Una vez se alivió esta carga, los sprints comienzan a ser más densos, reflejándose esto tanto en la inclinada pendiente que se presenta entre las fechas de marzo y mayo como en los resultados obtenidos en la aplicación. La práctica totalidad del desarrollo tuvo la recta ideal de quemado de trabajo por debajo del trabajo actual. Esto fue causado por la continua creación de PBIs,

extendiendo así el número de sprints en el que el trabajo quemado no alcanzó al ideal.

A partir del sprint 7, que corresponde con el inicio de una mayor disponibilidad del equipo de desarrollo, la pendiente de quemado cambia y se comienza a quemar una mayor proporción de puntos historia por sprint, consiguiendo para finales del sprint 13 alcanzar el quemado ideal.

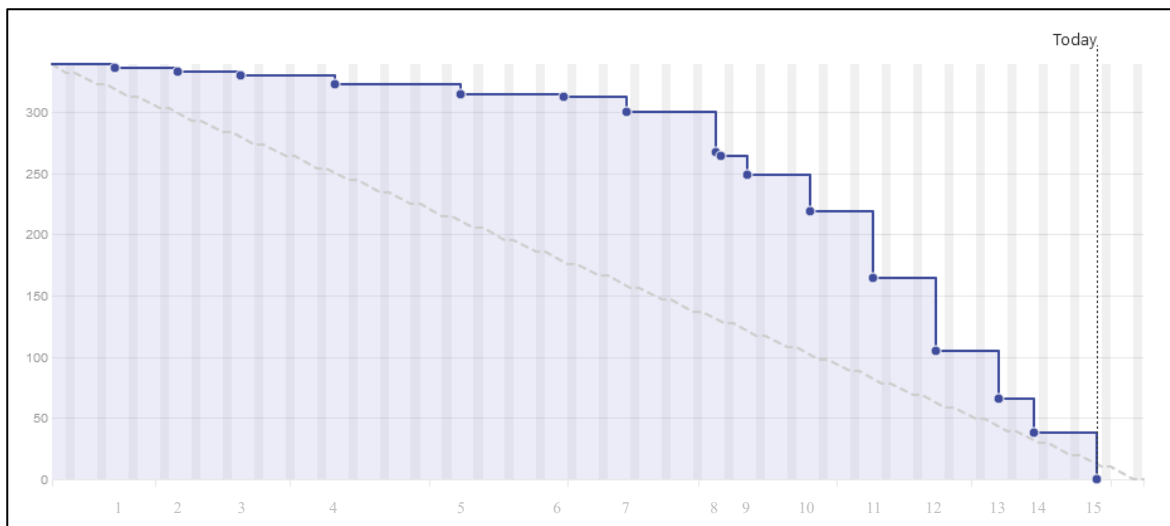


Figura 3.11: Project Burndown Chart

Este artefacto recoge la información de trabajo quemado de las historias de usuario e ítems del Product Backlog. Este es gestionado desde ZenHub a través de un tablero Kanban. El tablero (ver Figura 3.12), previamente configurado para la metodología de trabajo seguida por el estudiante, contiene 5 columnas. La primera columna reúne los PBIs del Product Backlog, que se ordenan por prioridad. La segunda representa el Sprint Backlog, que incluye los PBIs elegidos por el equipo de desarrollo para un sprint. Las columnas de “In Progress” y “Dev Done” sirven para marcar los PBIs que están siendo realizados y los que ya se han finalizado. Durante la revisión del sprint, los PBIs que se consideren Done son movidos a la columna “Closed”, en caso contrario se vuelven a colocar en el Sprint Backlog.

Cada PBI tiene asociado un identificador numérico que se utilizará como referencia [ANEXO D], y varios de ellos contienen comentarios y documentos asociados para describir más detalladamente sus requisitos. Se puede acceder al tablero entrando al repositorio de GitHub del proyecto si se tiene instalada la extensión de ZenHub u obteniendo permisos directamente en la web de la herramienta.

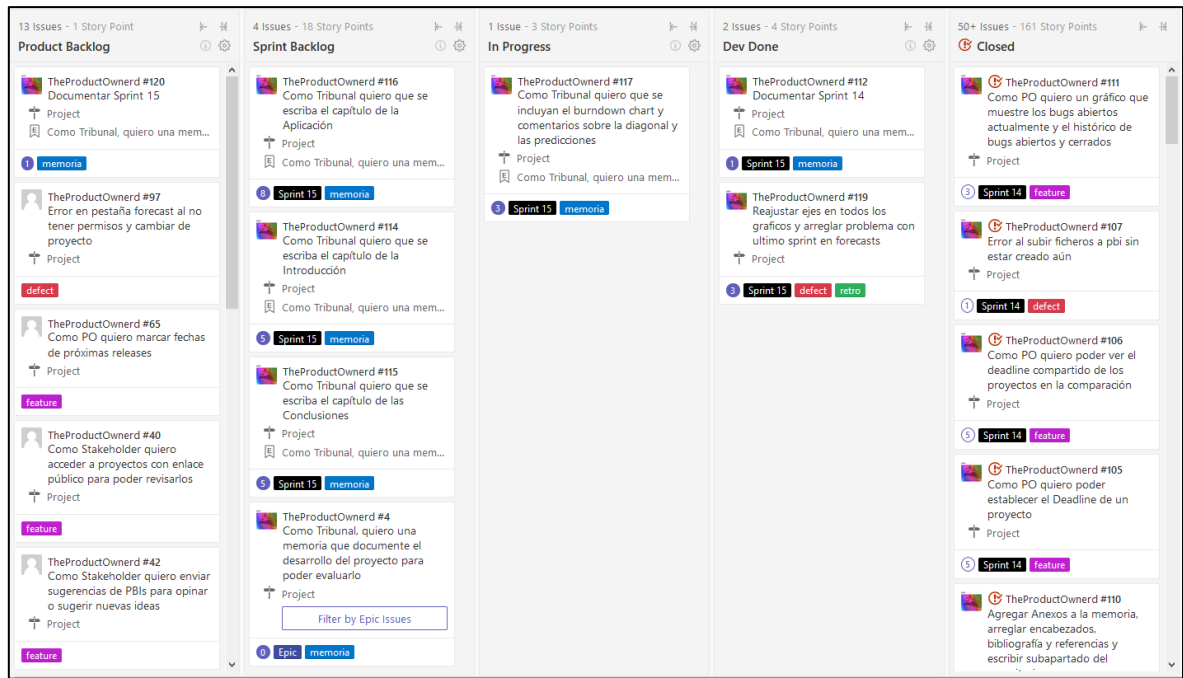


Figura 3.12: Kanban de ZenHub

Además de utilizar las herramientas proporcionadas por ZenHub, adicionalmente y cada varios sprints se generaron varias predicciones del avance del proyecto. En el sprint 4 se realizó una predicción por velocidad (Figura 4.5). Esta predice que manteniendo la velocidad media el proyecto estaría prácticamente acabado para el sprint 15, suponiendo que el backlog no tuviese cargas adicionales.

Unos sprints más adelante, en el sprint 13, se realizaron otras dos estimaciones, esta vez tanto por velocidad (Figura 4.18) como por regresión lineal (Figura 4.19). La predicción por velocidad estima que para el sprint 16 el trabajo estaría totalmente acabado. En cambio, la regresión indica que aún quedarían 42 puntos historia restantes, un valor algo más acertado en este caso que el anterior.

3.7. CONCLUSIONES

A lo largo de este capítulo se ha detallado el proceso seguido para el desarrollo del trabajo, explicando la metodología seguida, los roles y eventos utilizados. Se ha documentado la arquitectura general del proyecto justificando las tecnologías utilizadas, así como las herramientas que han facilitado la realización de este. Se ha descrito la configuración del repositorio, así como todos los enlaces con los servicios que sostienen la infraestructura del código. Por último, se han repasado los artefactos utilizados para la planificación y desarrollo del trabajo y para la estimación del esfuerzo.

CAPÍTULO 4. DOCUMENTACIÓN DE DESARROLLO Y PRUEBAS DE USABILIDAD

“If you don’t collect any metrics, you’re flying blind. If you collect and focus on too many, they may be obstructing your field of view” – Scott M. Graffius.

4.1. INTRODUCCIÓN

Este capítulo reúne la documentación generada en el proceso de desarrollo. En la sección 4.2 se adjuntan las reuniones asociadas a cada uno de los sprints, junto con un resumen del objetivo y temas tratados en cada uno. La sección 4.3 especifica la planificación, ejecución y resultados de las pruebas de usabilidad realizadas en la etapa final del desarrollo. Por último, se resumen las conclusiones del capítulo.

4.2. DOCUMENTACIÓN DE SPRINTS

Se documentan los informes de los sprints realizados incluyendo la siguiente estructura:

- Sprint Goal (Objetivo del sprint).
- Reunión de Planificación.
- Ejecución.
- Revisión.
- Retrospectiva.

SPRINT 1

Sprint Goal: configurar herramientas para la gestión del trabajo e iniciar el repositorio del proyecto.

Reunión de Planificación: En esta reunión se comenzó a analizar las distintas plataformas en las que era posible encontrar herramientas de gestión ágil, así como de repositorios. El equipo de desarrollo eligió las siguientes historias para su implementación: 8, 9 y 10¹.

Ejecución: En la ejecución del sprint se eligió la herramienta de gestión ágil y se inicializó el repositorio, así como el Product Backlog sin ningún problema durante el sprint. Tras la finalización del sprint se consiguieron finalizar con éxito todas las historias elegidas.

Revisión: En la revisión se aceptó y validó el trabajo realizado durante el sprint.

Retrospectiva: Se comentó que no era necesario nombrar todas las tareas del Backlog utilizando la plantilla de las historias de usuario si, por ejemplo, surgía alguna tarea como resolver errores o agregar alguna característica. También se planteó trabajar en un futuro con especificaciones por ejemplos, cuando el proyecto estuviese más avanzado.

¹ Los números indicados hacen referencia al identificador numérico de cada PBI. Si se desea examinar en detalle el listado de PBIs, se puede acceder a estos desde el siguiente enlace:
<https://app.zenhub.com/workspaces/theproductownerd-5da1ae2c8b2e250001841bdb/board?repos=214629397>

SPRINT 2

Sprint Goal: programar un front-end y un back-end simple que se interconecten y muestren un “hola mundo” en el cliente.

Reunión de Planificación: En esta reunión se planteó realizar un User Story Mapping, con el objetivo de recabar la gran mayoría de requisitos asociados a los principales usuarios del producto. Además, se planearon varias tareas relacionadas con la introducción a la Integración y Despliegue Continuos (CI/CD) . Se eligieron los siguientes PBIs: 1 y 11.

Ejecución: En la ejecución del sprint, el equipo de desarrollo realizó un User Story Mapping que permitió generar los PBIs 15 a 42. También se planificaron las entregas del producto y las distintas características que incluiría cada entrega (ver Figura 4.1 y Figura 4.2).

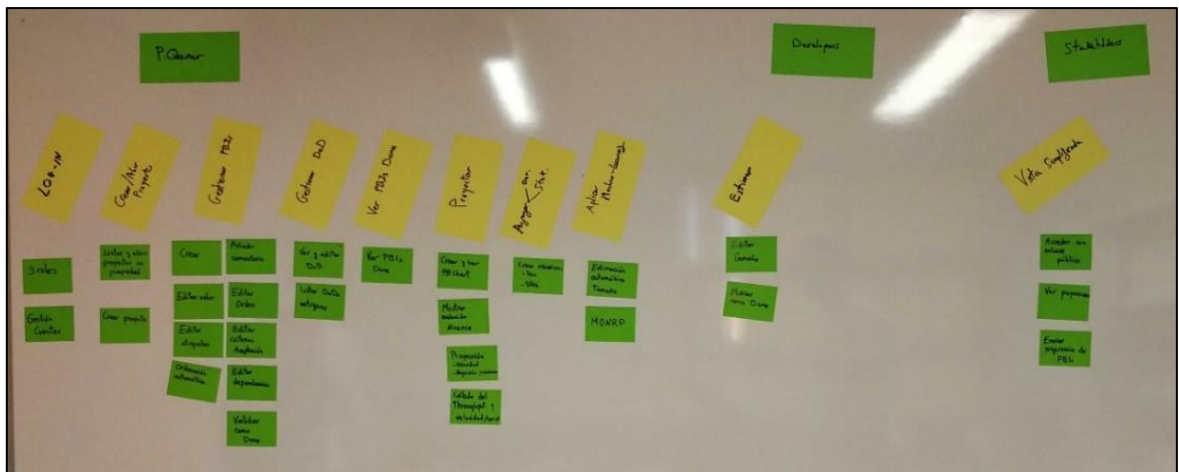


Figura 4.1: Sprint 2 - User Story Mapping

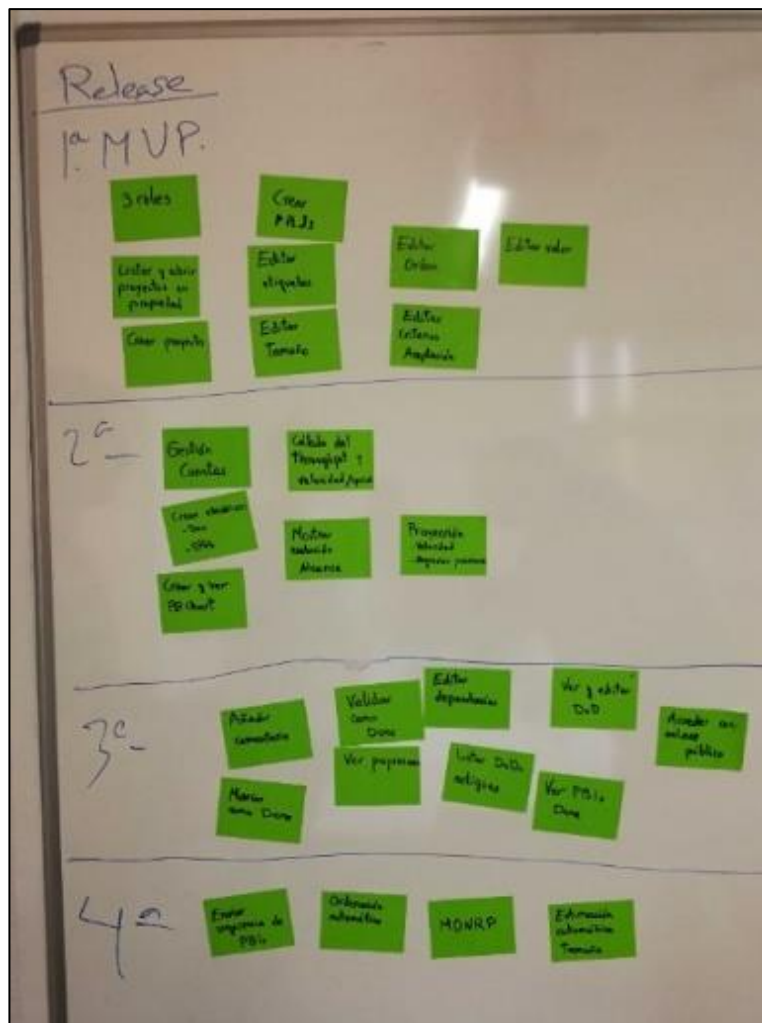


Figura 4.2: Sprint 2 - Entregas del User Story Mapping

También se codificaron un front-end sencillo en Angular y un back-end en Node.js, ambos simples y sin apenas funcionalidad.

Revisión: En la revisión se aceptó y validó el front y back sencillo, así como los PBIs introducidos en la herramienta de gestión que fueron generados durante el User Story Mapping.

Retrospectiva: Se indicó al equipo de desarrollo que era necesario enlazar los commits realizados en el repositorio con las historias a las que hacía referencia. También se propuso crear nuevas etiquetas para los siguientes tipos de PBI: feature, defect, risk, debt, memoria, flow.

SPRINT 3

Sprint Goal: El objetivo del sprint es configurar Integración Continua y pruebas continuas al proyecto.

Reunión de Planificación: En esta reunión se habló de manera breve de las posibilidades y dificultades de desplegar el proyecto en la nube y de las distintas herramientas que se podían utilizar para el testing. Debido a la falta de tiempo del equipo de desarrollo se estimó que sólo habría tiempo para realizar el PBI 12.

Ejecución: Se creó el hook entre GitHub y TravisCI para integrar las pruebas. La Figura 4.3 muestra la build con éxito en TravisCI del último commit realizado durante el sprint.

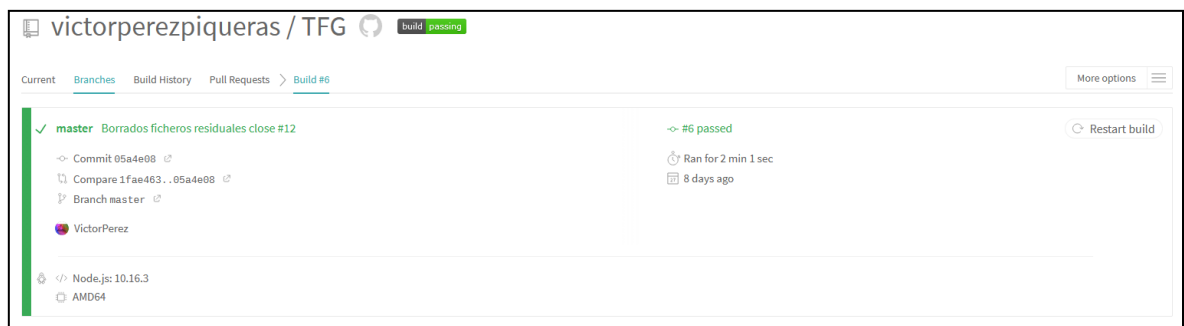


Figura 4.3: Sprint 3 - Build en Travis

Revisión: En la revisión se mostraron los commits realizados enlazados a TravisCI con las pruebas automáticas pasadas con éxito.

Retrospectiva: Se recomendó cambiar ligeramente el formato seguido hasta el momento para la documentación de los sprints.

SPRINT 4

Sprint Goal: Desplegar de forma automática en Heroku cualquier cambio en el repositorio.

Reunión de Planificación: Se planificó extender la duración del sprint una semana más, pasando a ser de 3 semanas, debido a la falta de tiempo del equipo de desarrollo. Se debatió sobre las distintas maneras de desplegar en Heroku, pudiendo desplegar desde el mismo script de TravisCI al pasar las pruebas de integración, o utilizar la opción que Heroku aporta que permite desde GitHub desplegar el proyecto tras aceptar los tests automáticos de Travis. Se eligieron los 2 PBIs referentes a Despliegue Continuo, ya que tienen mucho en común: despliegue manual y automático en Heroku, además de enlazar Travis con Slack para notificaciones, los PBIs son el 13, 14 y 43.

Ejecución: Se enlazó Travis con Slack para notificar los builds con éxito y se enlazó el despliegue en Heroku a partir de TravisCI.

Revisión: Se mostró el pipeline de integración y despliegue continuo y la aplicación totalmente funcional desplegada en Heroku. El código se almacena en GitHub y se envía a TravisCI para que lance las pruebas. Tras pasar las pruebas de forma satisfactoria, TravisCI inicia el despliegue en Heroku, que almacena la aplicación final desplegada y funcional.

TravisCI notifica por un canal de slack dedicado exclusivamente a las notificaciones de builds con éxito (ver Figura 4.4).

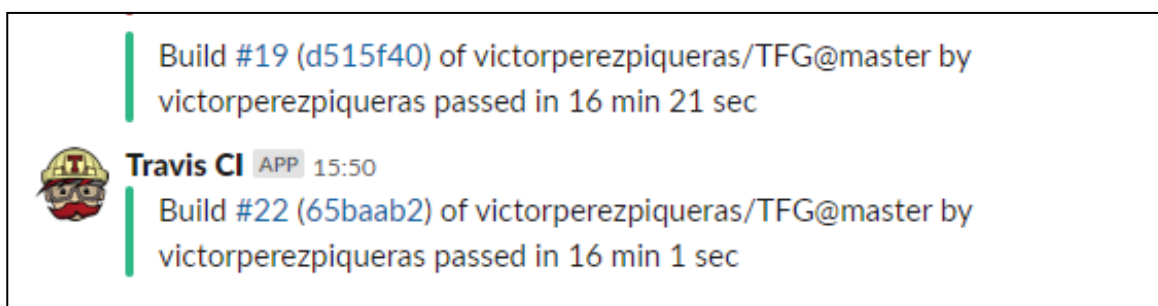


Figura 4.4: Sprint 4 - Mensaje automático de Slack

Teniendo en cuenta la velocidad actual hasta la fecha, se realizó una predicción por velocidad (ver Figura 4.5) para estimar los puntos historia restantes al final del proyecto, suponiendo

que el sprint 15 fuese el último realizado.

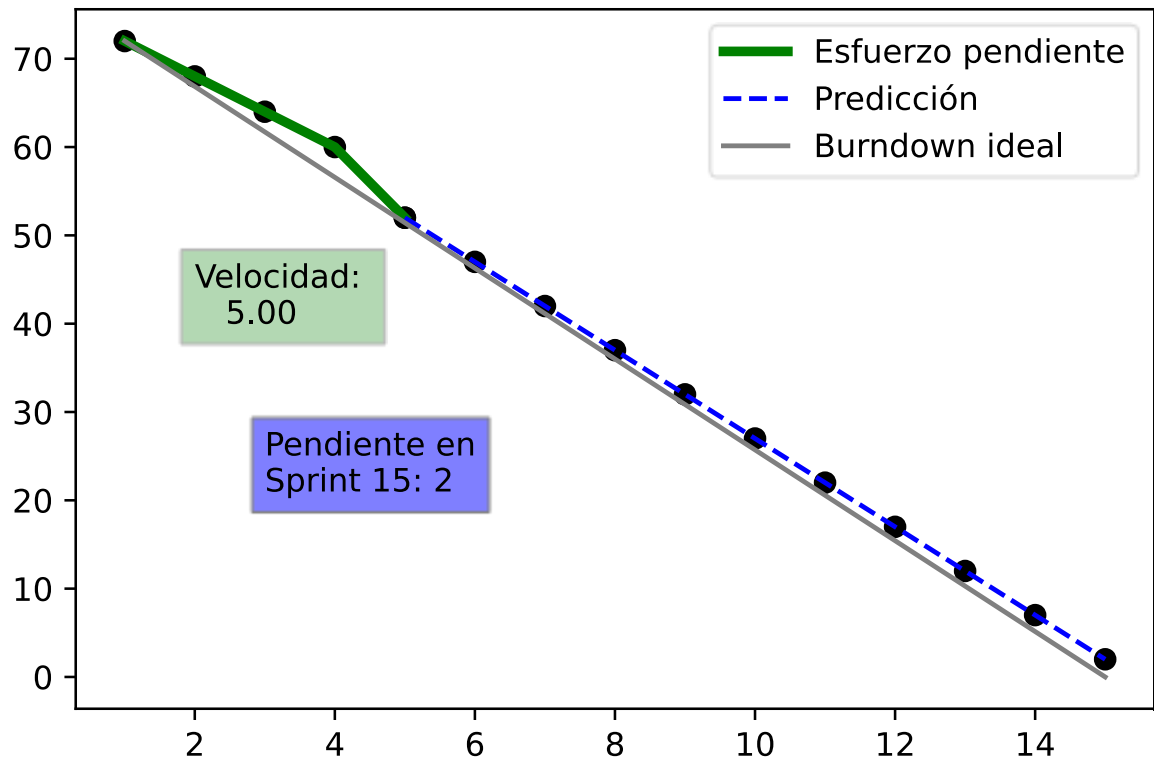


Figura 4.5: Sprint 4 - Predicción por velocidad

Retrospectiva: Al crear nuevos ítems a partir de una épica, hay que disminuir la estimación de la épica de forma acorde, para esto se creó el pbi 44. También se dio cuenta de la documentación de los sprints anteriores creando los PBIs 45, 46, 47 y 48.

SPRINT 5

Sprint Goal: Informarse y aprender más a fondo el contexto del TFG mediante la bibliografía proporcionada y gestionar documentos de sprints.

Reunión de Planificación: Se eligen los PBIs relacionados con lectura de información para adquirir una base que ayudará a documentar el estado del arte de la memoria, eligiendo los PBIs 2 y 7, además de crear otros PBIs referentes a la separación de documentación de cada sprint.

Ejecución: Se leyó y analizó la información de las 2 fuentes bibliográficas y se realizaron notas sobre esta. También se dividieron los documentos de scrum por sprints, agregando cada uno en GitHub al pbi correspondiente de documentación de sprint.

Revisión: Se comunicaron las conclusiones de la lectura.

Retrospectiva: Se recomendó recalcular la estimación de la Épica de la memoria, teniendo en cuenta los nuevos PBIs creados relacionados con esta.

SPRINT 6

Sprint Goal: Iniciar el diseño y esquema de la memoria y empezar a redactar el capítulo del estado del arte.

Reunión de Planificación: Se escoge para el desarrollo los PBIs 5 y 49 referentes a la memoria.

Ejecución: Se inició el desarrollo de la memoria y se empezó a escribir el Estado del Arte, dejándolo iniciado pero inconcluso.

Revisión: Se mostró el avance y se comentaron los fallos en el diseño y redacción, las carencias y las nuevas ideas que se podrían incluir.

Retrospectiva: Se anotó como interesante referenciar al inicio la bibliografía utilizando la herramienta Mendeley y crear las imágenes de forma manual si los diseños se podían realizar con herramientas de Word.

SPRINT 7

Sprint Goal: Crear los modelos de usuarios y proyectos, la arquitectura ReST y la base de datos y la interfaz básica de la aplicación.

Reunión de Planificación: Se escoge para el desarrollo los PBIs 15, 17 y 18, relacionados con la creación de modelos y diseño de la base de datos y la interfaz básica para conectarse con un usuario e inspeccionar los proyectos a los que se pertenece. Además, se escoge el PBI 5 de la memoria.

Ejecución: Se empezó diseñando el diagrama de clases para usuarios y proyectos y se pasó a diseñar la base de datos y el API-ReST. Tras tener el backend completamente funcional se pasó a diseñar la interfaz principal para que mostrase un listado de los proyectos a los que pertenece el usuario. Se crea además una opción que permite agregar proyectos a favoritos para tenerlos al alcance en la barra de tareas de la aplicación. Respecto a la memoria, se completa el Estado del Arte. La Figura 4.6 muestra como se presenta esta interfaz.

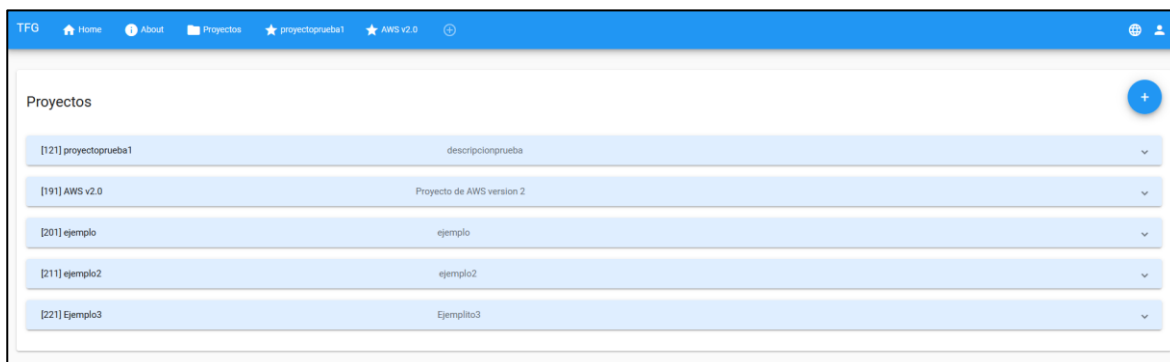


Figura 4.6: Sprint 7 - Interfaz de listado de proyectos

Revisión: Se mostró la aplicación y se comentaron las nuevas funcionalidades que podría traer para el siguiente Sprint. Respecto al estado del arte se comentaron varios cambios que sería conveniente aplicar para mejorar la calidad del mismo.

Retrospectiva: Se recomienda utilizar una herramienta de modelado de prototipos para diseñar los prototipos iniciales de las siguientes interfaces a realizar, ya que serán de mayor complejidad.

SPRINT 8

Sprint Goal: Diseñar la sección de gestión de Product Backlog Items.

Reunión de Planificación: Se eligen los PBIs 19, 20, 21, 22, 23, 27, 30, 38, 39, 50, 51 y 52 para realizarse durante el sprint. Todos relacionados con la gestión de PBIs, filtrado, creación y edición.

Ejecución: Antes de comenzar con la implementación se realizaron varios prototipos de pantallas que se validaron durante la primera semana del sprint con el Product Owner antes de comenzar el desarrollo de las vistas. El diseño de prototipos es el siguiente: en la Figura 4.7 se muestra el diseño de la vista de Overview de un proyecto, incluyendo el PBCh y el equipo. La Figura 4.8 muestra el diseño inicial del Product Backlog y la estructura visual de cada uno de los PBIs. La vista en detalle de cada PBI se muestra en la Figura 4.9.

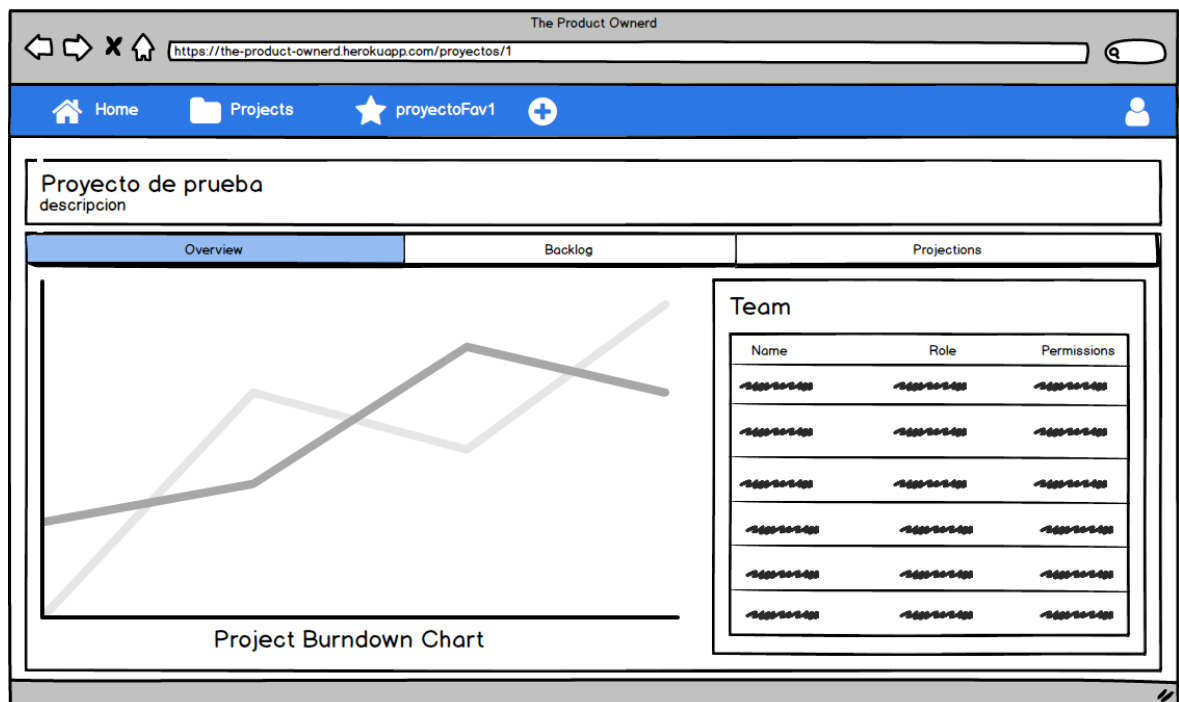


Figura 4.7: Sprint 8 - Vista principal del proyecto

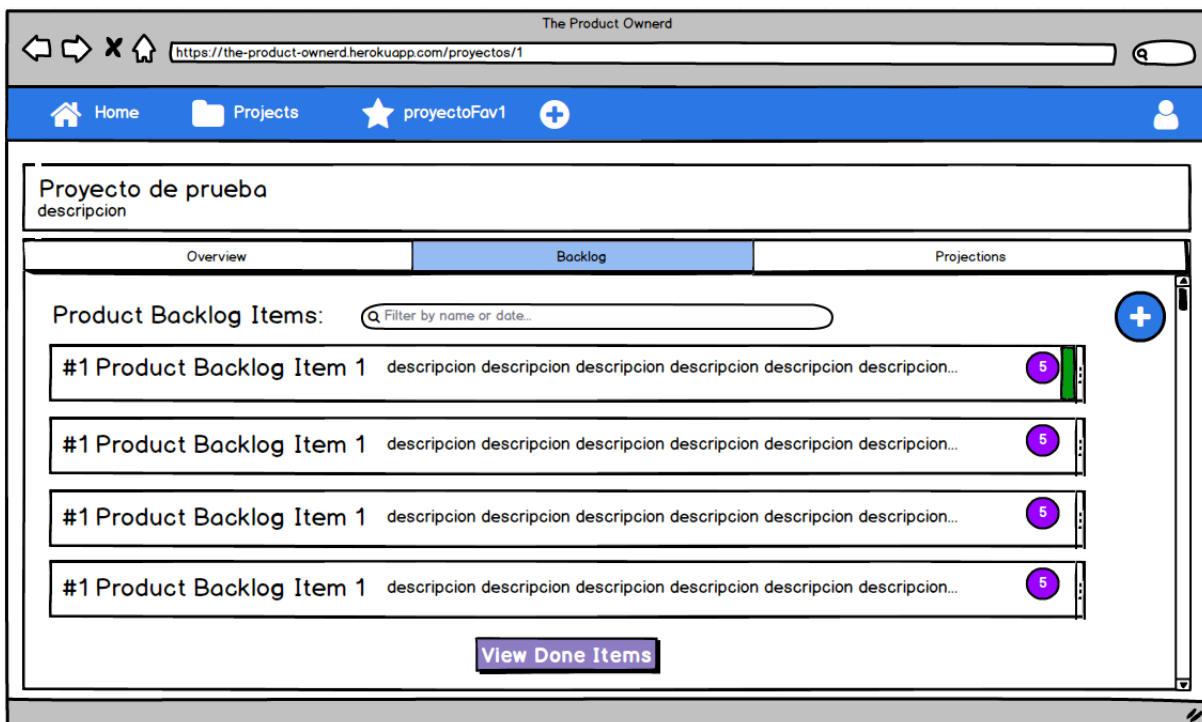


Figura 4.8: Sprint 8 - Vista del Product Backlog

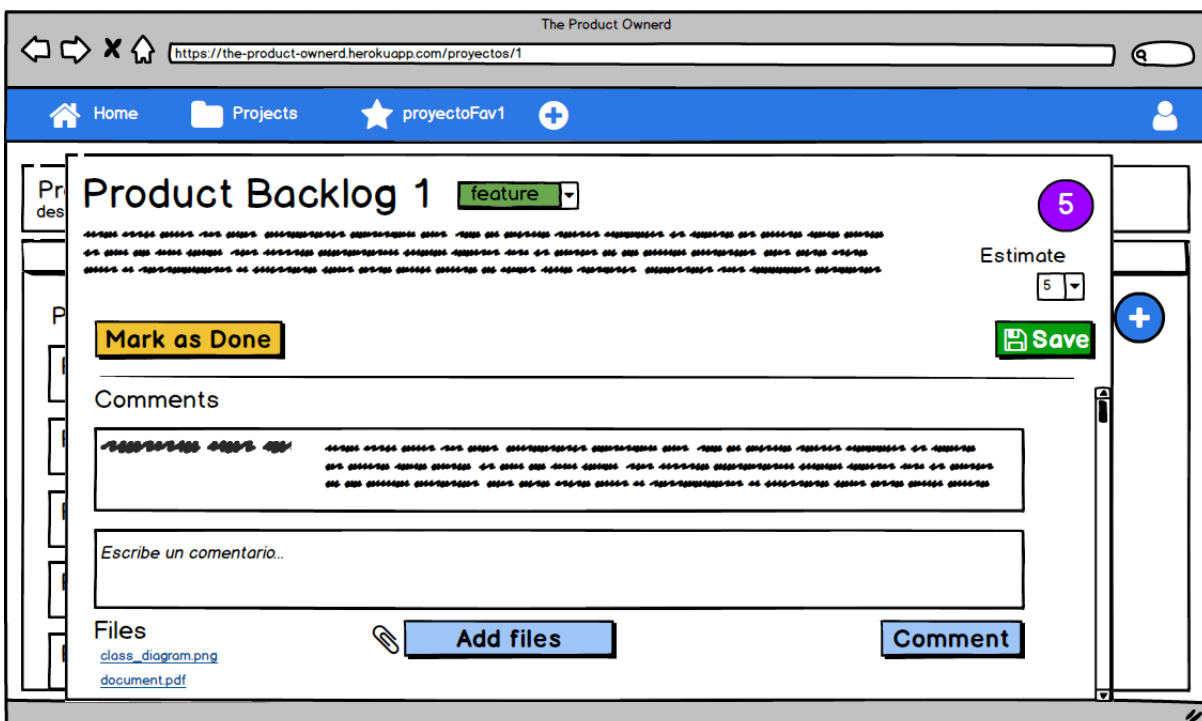


Figura 4.9: Sprint 8 - Vista detallada de un PBI

Tras validar con el Product Owner el diseño de prototipos, se diseñó el diagrama de clases y las relaciones entre las clases del sistema. Una vez se tuvo el modelo definido se pasó a diseñar las tablas de la base de datos y los controladores del backend y el api que daría acceso a los PBIs almacenados. A su vez, se diseñó la interfaz del cliente para mostrar los PBIs y permitir realizar operaciones de ordenado, creación y edición de estos (ver Figura 4.10).

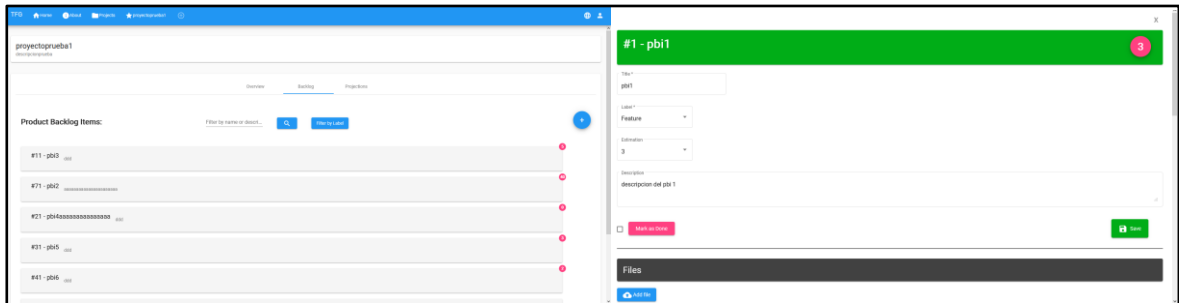


Figura 4.10: Sprint 8 - Interfaces de Listado y Detalles de PBIs

Revisión: Se probó la interfaz y se comentaron aspectos de la estética. También se discutió un problema del pipeline y las posibles soluciones que tendrían que tomarse para el siguiente sprint.

Retrospectiva: Para el futuro, se debe tener en mente que el entorno local y el de pruebas deben ser exactamente idénticos si queremos evitar fallos inesperados en la integración del sistema.

SPRINT 9

Sprint Goal: Mejorar la sección de PBIs agregando opciones de creación de criterios de aceptación, dependencias y ordenación por valor.

Reunión de Planificación: Se eligen los PBIs 20, 24, 25, 26, 53 y 54 para realizarse durante el sprint.

Ejecución: Partiendo del diseño anterior, se agregaron varias opciones nuevas a la vista de detalles del PBI. Se agregó la funcionalidad de crear y marcar como cumplidos criterios de aceptación. Se finalizó la opción de adjuntar ficheros, así como la opción de agregar dependencias con otros PBIs. También se añade la opción de dar un valor a cada PBI, opción disponible sólo para el PO. También permite ordenar PBIs por valor.

Revisión: Se mostró el nuevo funcionamiento para cada una de las nuevas características, todo correcto.

Retrospectiva: Arreglar el pipeline de CD pasa a ser una tarea menos prioritaria y se retomará en el futuro tras realizar otras historias de mayor interés.

SPRINT 10

Sprint Goal: Redactar metodología del proyecto y desarrollar gráficas de progreso y evolución del alcance.

Reunión de Planificación: Se eligen los PBIs 31, 32, 56, 57 y 58 para este sprint.

Ejecución: Por un lado, en la primera pestaña se sitúa el Project Burndown Chart y la tabla de miembros del equipo. En la pestaña de Forecasts se introduce primero un gráfico (Figura 4.11) que genera la velocidad media estimada y la mejor y peor velocidades medias. Todo esto pudiendo modificar varios parámetros para ajustar la predicción. Por otro lado, se crea otro gráfico que aporta información del PoC del proyecto.

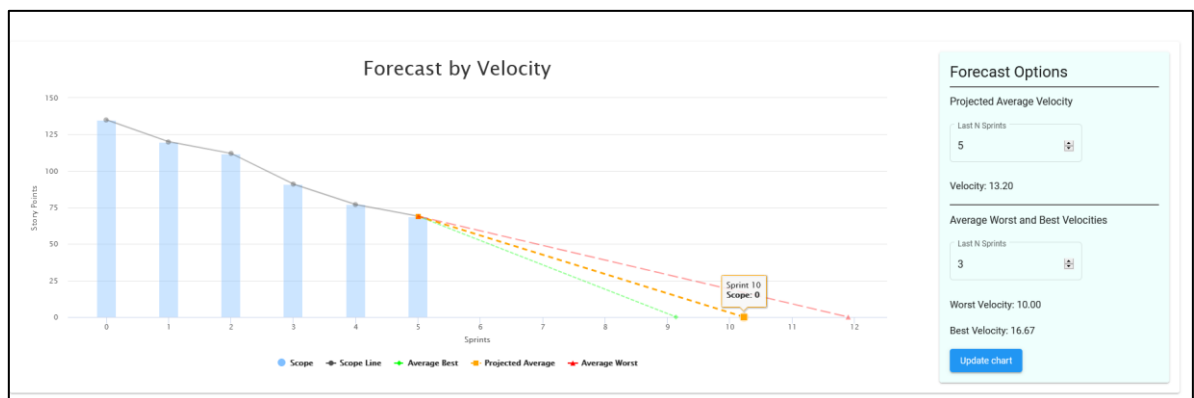


Figura 4.11: Sprint 10 - Vista de Forecast by Velocity

Revisión: Se muestran los avances y se decide retocar varios aspectos de las opciones de gráficos, validando las demás características como correctas. Se debatió el punto de corte de cada línea de predicción. Al final se decide dejarlo como está en vez de forzar a que la línea acabe en un punto entero fijo. Se incorpora la opción de elegir los N últimos sprints.

Retrospectiva: A partir de este sprint, el despliegue para a ser realizado por Heroku cuando recibe la orden de desplegarse desde la interfaz que proporciona (ver Figura 4.12), pudiendo desplegar la aplicación al pasar con éxito las pruebas de integración o mediante un trigger manual.

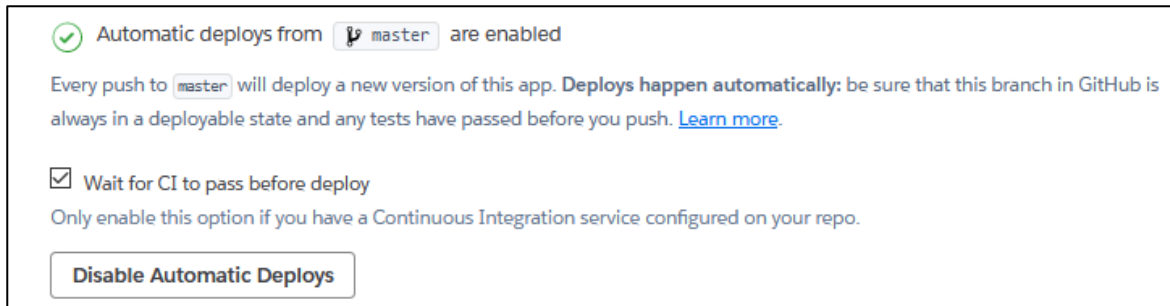


Figura 4.12: Sprint 11 - Configuración de Despliegue de Heroku

SPRINT 11

Sprint Goal: Mejorar gráficos, dar soporte a gestión de cuentas de usuario e invitaciones a nuevos usuarios mediante registro.

Reunión de Planificación: Se eligen los PBIs 16, 35, 59, 60, 63, 64, 69, 70 y 71 para realizarse durante el sprint.

Ejecución: Se distinguen en los gráficos de PoC y del PBCh los PBIs por etiqueta. Se agrega un control de fecha de último sprint al gráfico de la pestaña Forecasts que permite visualizar mediante puntos de corte los puntos historia sin finalizar tras dicho sprint. Tras esto, se desarrolla la interfaz de gestión de la cuenta del usuario, para permitirle cambiar sus datos. Después se mejoró la interfaz de inicio de sesión añadiendo un logo y opción de registro. Además, se creó un panel para invitar a nuevos usuarios mediante un enlace enviado automáticamente desde un correo electrónico que permite unir automáticamente al nuevo usuario al proyecto. Por último, se refactorizó código del backend y se incluyó el uso de tokens de sesión para autenticar a los usuarios al hacer llamadas al api, así como gestión de errores.

Revisión: En esta reunión se mostraron todos los avances y se comentaron varios puntos a retocar. Se moverá el gráfico de PoC a la pestaña de Reports. La sección de invitaciones se transformará en un desplegable para ahorrar espacio y se requerirá la contraseña del usuario para poder cambiar a una nueva contraseña. También se discutió la estética de los PBIs y la disposición de sus elementos visuales, quedando pendiente reajustarla para el próximo sprint.

Retrospectiva: Los despliegues volverán a ser automáticos utilizando la opción que proporciona Heroku en la web, que desplegará el servicio si la Integración Continua de Travis es satisfactoria.

SPRINT 12

Sprint Goal: Manejar gestión de sprints, sprint goals y agregar opción de visualización del scope creep. Mejorar varios aspectos técnicos del backend y frontend.

Reunión de Planificación: Se eligen los PBIs 34, 61, 62, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 87, 88, 89 y 90 para realizarse durante el sprint. Se busca darle al Product Owner mayor gestión de la información del proyecto, como: sprint actual, sprint goals, visión y detalles del proyecto.

Ejecución: Primeramente, se arreglaron fallos y se realizaron modificaciones detectadas en el anterior sprint review. Tras esto, se llevaron a cabo ajustes técnicos y visuales de varias interfaces. Tras tener todo organizado, se empezó con la sección de sprints. Se realizó una interfaz básica inicial que se fue mejorando hasta dar con el resultado final de esta (ver Figura 4.13). Se agregó al Project Burndown Chart la opción de visualizar el Scope Creep.

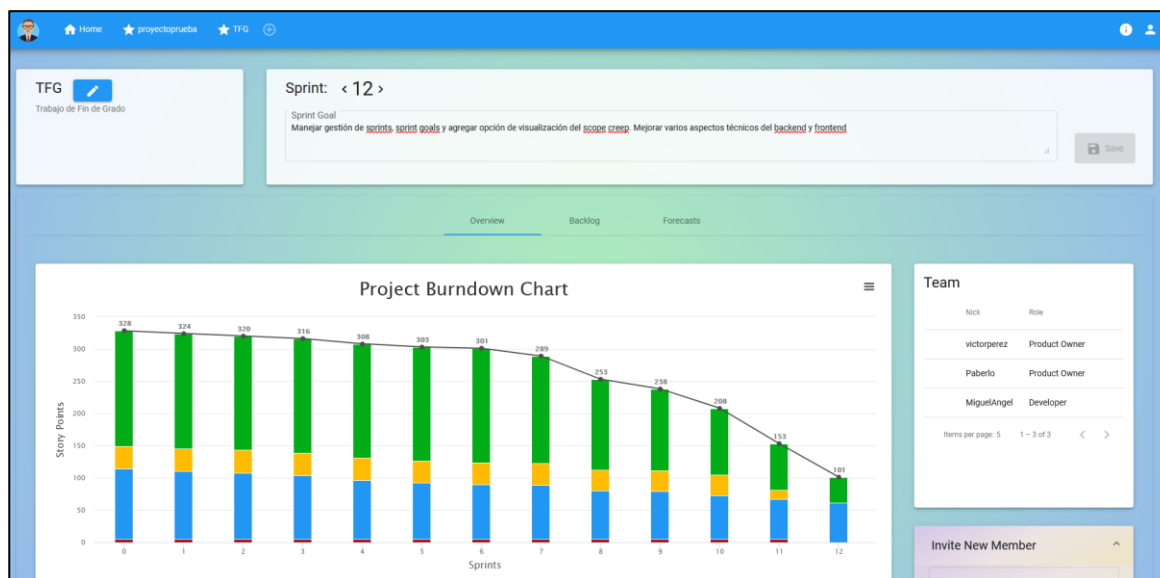


Figura 4.13: Sprint 12 - Interfaz de Overview

En la última etapa del sprint se pasó a realizar correcciones, actualizar y limpiar código, resolver inefficiencias, actualizar paquetes y documentar los métodos más importantes del código, entre otras actividades. Para finalizar el sprint, se le dio formato visual al correo de invitación automático (Figura 4.14) y se diseñó la pestaña de About (Figura 4.15).

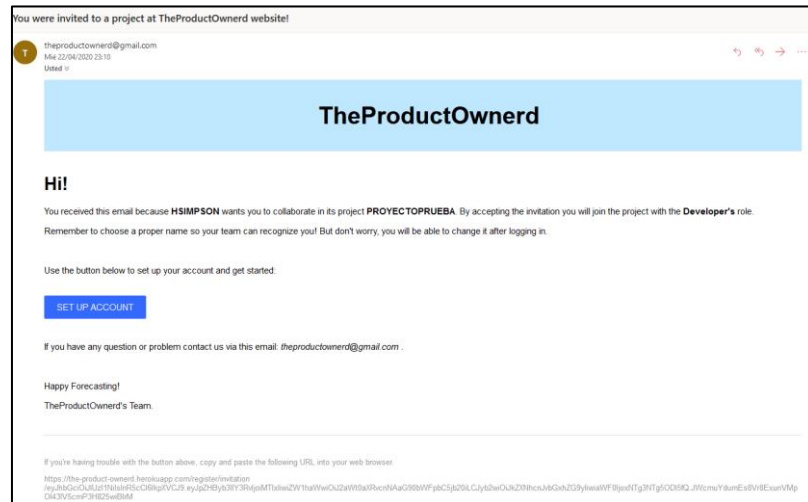


Figura 4.14: Sprint 12 - Email de invitación

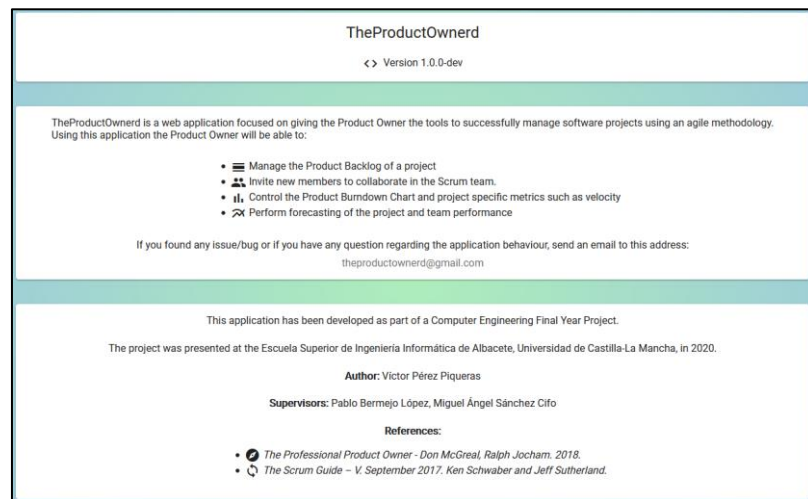


Figura 4.15: Sprint 12 - Pestaña de About

Revisión: Se revisó por encima el resultado de la web y se fueron comentando varios ajustes puntuales en ciertos sitios de la web.

Retrospectiva: Se observó que, con el tiempo, la velocidad ha aumentado en el desarrollo de manera considerable, debido a la mayor experiencia que el equipo de desarrollo está consiguiendo. También se hizo notar el gran número de PBIs que se generaron que no tenían relación con características de la aplicación, sino con deuda tecnológica o mejoras puntuales. Esto se debe al gran esfuerzo que se ha puesto en llevar a cabo todas las características importantes, que han relegado a un segundo plano tareas como depuración de código antiguo, limpieza, actualización de paquetes y características antiguas.

SPRINT 13

Sprint Goal: Ampliar los métodos de forecast.

Reunión de Planificación: Se eligen los PBIs 33, 66, 91, 93, 95, 96, 100, 101 y 102 para realizarse durante el sprint. Aparte de arreglar algunos fallos encontrados y mejoras, se busca por un lado hacer más modular el diseño de los gráficos del cliente, para facilitar su reutilización antes de ampliar los métodos de forecasting. Se planifica utilizar predicciones con regresión y poder comparar pendientes entre proyectos.

Para la finalización del sprint se diseñaron unas pruebas de usabilidad en las que varios profesionales del mundo tecnológico utilizarían las funcionalidades principales de la aplicación con el fin de detectar fallos y posibles mejoras y subsanarlas de cara a las últimas entregas del proyecto. Se planificó el esquema a seguir en la prueba, así como las tareas a realizar por los usuarios y el formato del informe a presentar. En el siguiente subapartado se documenta en profundidad el objetivo, procedimiento y resultados de estas pruebas de usabilidad.

Ejecución: Antes de comenzar, se refactorizó la estructura de los gráficos, separándolos en componentes. Con esto realizado, la tarea de crear los nuevos gráficos para las regresiones fue más sencilla. Se instaló una librería de regresiones para facilitar el cálculo de los modelos. Tras esto y después de algunos retoques a las interfaces se diseñó la vista de comparativa entre proyectos, dejándola en un prototipo simple y estable.

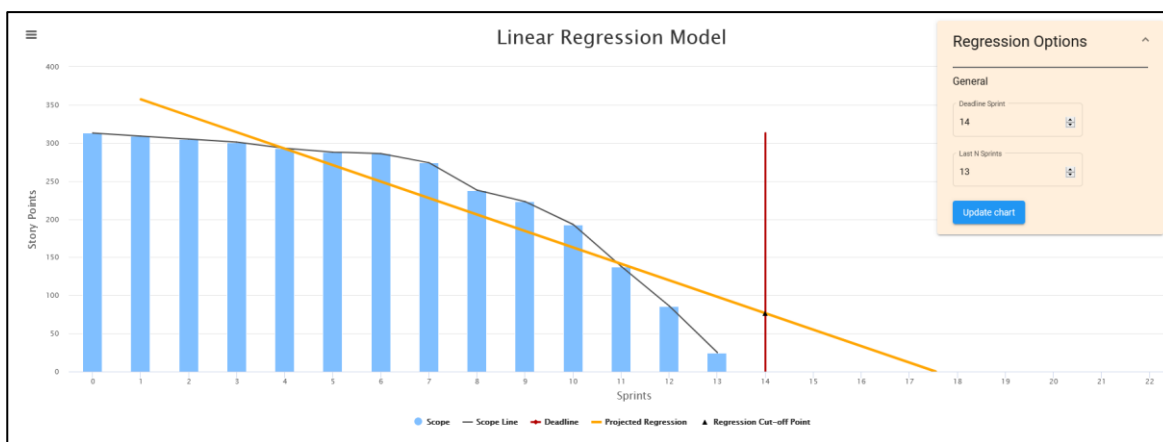


Figura 4.16: Sprint 13 - Modelo de Regresión Lineal

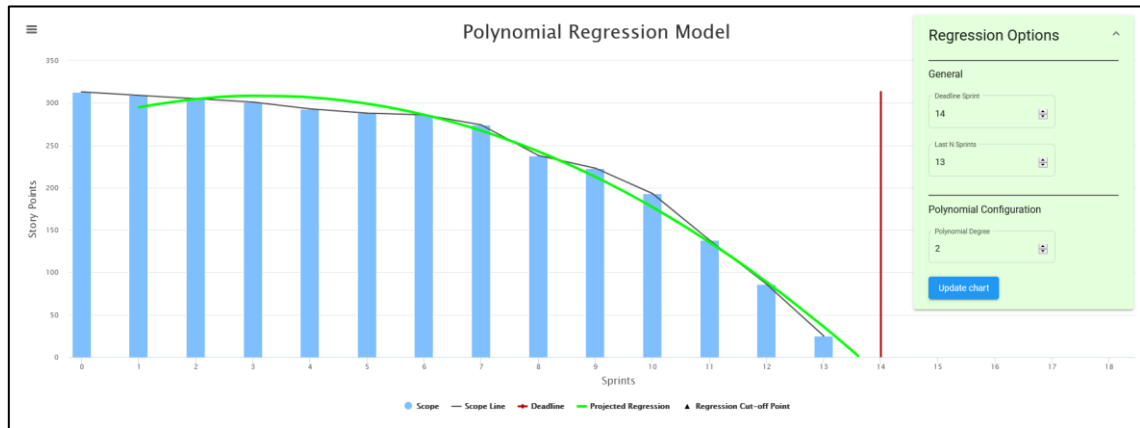


Figura 4.17: Sprint 13 - Modelo de Regresión Polinómica

Revisión: Se mostraron los nuevos gráficos introducidos (ver Figura 4.16 y Figura 4.17) y se valoraron las predicciones proporcionadas. Con respecto a las pruebas de usabilidad, se anotaron todos los defectos y sugerencias encontrados durante la reunión [ANEXO C], para tenerlos en cuenta de cara al próximo sprint. Para valorar el estado actual y futuro del proyecto suponiendo que el ultimo sprint fuese el 16 se generaron dos predicciones, una utilizando la velocidad media (Figura 4.18), y otra por regresión lineal (Figura 4.19).

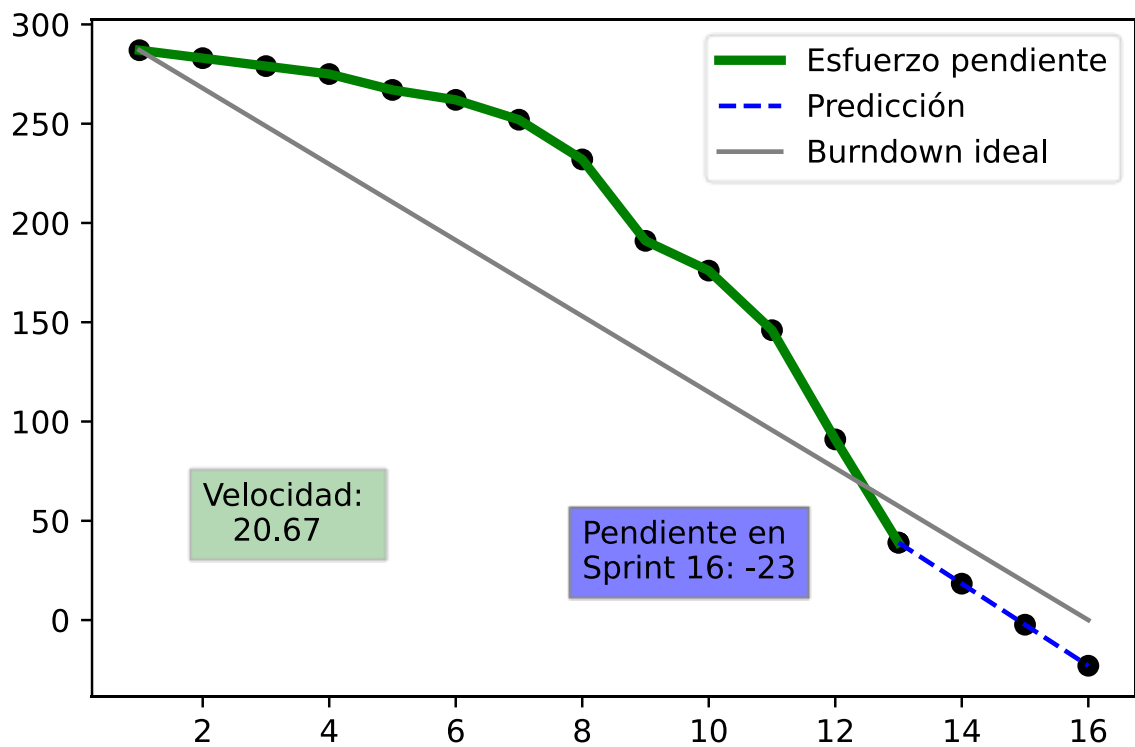


Figura 4.18: Sprint 13 - Predicción por velocidad

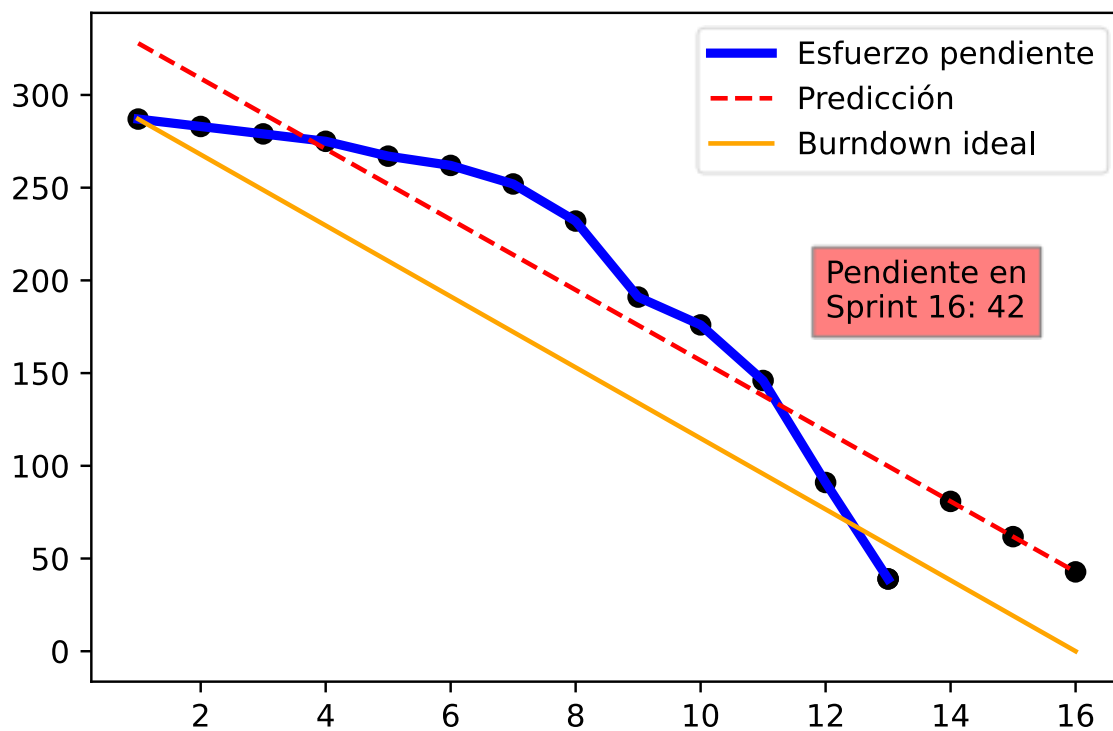


Figura 4.19: Sprint 13 - Predicción por regresión lineal

Retrospectiva: Teams necesita que se configuren permisos en una reunión para que los asistentes puedan descargar o adjuntar ficheros. Este problema surgió durante la prueba de usuarios, y se subsanó enviando el guion de pruebas por correo electrónico a los participantes.

SPRINT 14

Sprint Goal: Incluir fechas límite en los proyectos, documentar las pruebas de usabilidad y arreglar los problemas comentados por los participantes en el sprint anterior.

Reunión de Planificación: Se eligen los PBIs 94, 103, 104, 105, 106, 107, 108, 109, 110 y 111 para realizarse durante el sprint. Se persigue para este sprint redactar los informes de pruebas y agregar funcionalidad extra a la herramienta de comparativas, así como resolver varios de los problemas encontrados por los participantes de las pruebas.

Ejecución: Tras recoger el feedback de las pruebas de usabilidad en el sprint anterior, se arreglaron varios de los aspectos más sencillos de corregir y se redactó el informe de las pruebas, incluyendo los anexos correspondientes en la memoria y resumiendo los resultados. Tras esto, se procedió a incluir fechas límite en los proyectos, para poder realizar comparativas entre proyectos con fechas límite compartida (ver Figura 4.20). Se agregó un gráfico adicional en la pestaña de informes que muestra la cantidad de errores sin resolver del proyecto. Esta idea se obtuvo gracias a la retroalimentación de las pruebas de usabilidad. Por último, se arregló el funcionamiento de la regresión polinómica, pendiente desde el sprint anterior.

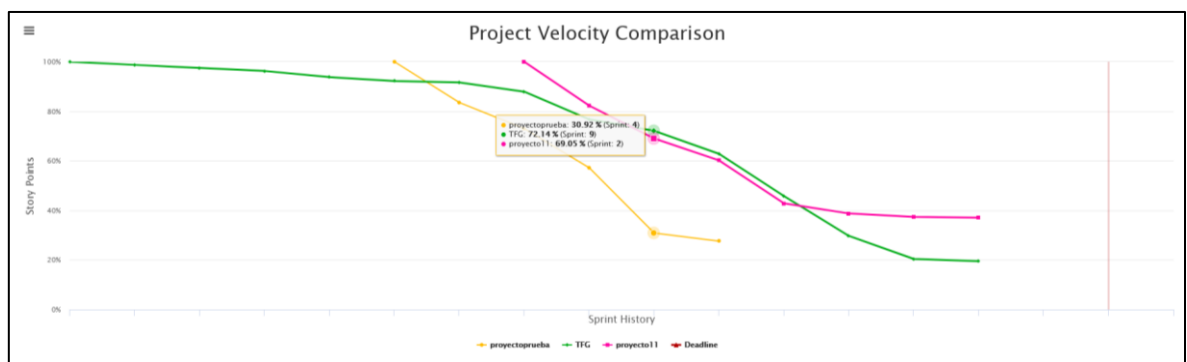


Figura 4.20: Sprint 14 - Comparativa de proyectos por velocidad relativa

Revisión: Se comprobaron las predicciones para ver que eran correctas y se llegó a la conclusión de que era necesario mostrar un aviso al usuario de que estas podrían ser imprecisas. También se anotó reajustar los ejes de los gráficos para mostrar el punto de inicio y clarificar la información que se muestra en los desplegados de los gráficos.

Retrospectiva: Se apuntó que la forma más común de representar gráficos Burn-down muestra en el punto X los valores del sprint X al final de este.

SPRINT 15

Sprint Goal: Desarrollar los capítulos de introducción y conclusiones y finalizar la memoria.

Reunión de Planificación: Se eligen los pbis 112, 114, 115, 116, 117, 119, 120, 121, 122 y 123 para realizarse durante el sprint. Se persigue desarrollar los capítulos restantes de la memoria, dando un margen a mitad del sprint para que se revisen los contenidos y se corrijan los fallos.

Ejecución: Se inició el sprint desarrollando el apartado descriptivo del producto. Tras esto, se desarrollaron las conclusiones y la introducción de la memoria. Una vez se tuvo completada la memoria, se mandó esta para ser revisada. En dos ocasiones se corrigieron los fallos y se modificaron varios aspectos de ortografía, conceptos, estructura y apariencia visual de la memoria.

Revisión: Se revisó la memoria destacando las secciones en las que surgió mayor controversia de opiniones. Tras llegar a un consenso en estas partes, se validó el trabajo restante realizado.

Retrospectiva: Se anotó como interesante el uso de la herramienta “mostrar todo” que proporciona MS Word, para poder visualizar los saltos de línea entre secciones y poder uniformar su apariencia de manera más sencilla.

4.3. PRUEBAS DE USABILIDAD

Durante el sprint 13 se planificó la realización de unas pruebas de usabilidad. Seguidamente, se resume la planificación seguida para las mismas, se describe su desarrollo y se presentan los resultados obtenidos.

4.3.1. PLANIFICACIÓN

El fin de esta actividad fue evaluar la usabilidad de la aplicación, la utilidad y aplicabilidad en un entorno de trabajo empresarial. Con este propósito se utilizó el conjunto de normas SQuaRE (System and Software Quality Requirements and Evaluation) [29] para estandarizar el proceso de especificación, diseño y evaluación de las pruebas. La planificación del proceso de evaluación a seguir se ajusta al estándar ISO/IEC 25040:2011 [30], el cual identifica los siguientes pasos a seguir para la evaluación:

1. Establecer los requisitos de la evaluación.
2. Especificar la evaluación.
3. Diseñar la evaluación.
4. Ejecutar la evaluación.
5. Concluir la evaluación.

Dado que el propósito de la prueba era evaluar la calidad en uso de la aplicación, se siguió la norma ISO/IEC 25022:2016 [31] de medición de la calidad en uso. Esta norma define las características que se deben evaluar y las posibles métricas que se pueden utilizar para alcanzar tal propósito. Para la evaluación se seleccionaron las siguientes características:

- **Efectividad:** precisión y completitud con la que los usuarios logran objetivos específicos.
- **Eficiencia:** recursos gastados en relación con la precisión y completitud con la que los usuarios logran objetivos específicos.
- **Satisfacción:** grado en el que se satisfacen las necesidades del usuario cuando se utiliza un producto o sistema en un contexto de uso específico.

Por último, se definió el modelo de informe a utilizar para documentar la realización de las pruebas de usabilidad. Para este fin se utilizó el estándar ISO/IEC 25062:2006 [32], que define el formato común de la industria (CIF) para los informes de pruebas de usabilidad.

Una vez se planificaron los objetivos, modelos y métricas a evaluar, se pasó a diseñar la evaluación. Se estableció contacto con varios profesionales del sector tecnológico para que participasen en el proceso, acordando una reunión grupal con estos. En los días previos a la reunión se redactaron las bases del informe de pruebas y se diseñó un guion de usuario, disponible en el ANEXO A. Este guion explica los pasos que cada usuario debe seguir durante la realización de las pruebas, describiendo todas las tareas que debe realizar y la información que debe anotar.

4.3.2. EJECUCIÓN

Durante la reunión se procedió a informarles del propósito de la prueba, se les proporcionó el guion y se comentaron los aspectos básicos de la actividad. Los participantes, sin experiencia alguna utilizando la aplicación, procedieron a desconectarse de la llamada en curso y empezaron a realizar sus tareas de forma individual. Mientras tanto, se hizo un resumen breve de la aplicación a varios de los invitados que no participaban en las pruebas.

Las tareas que siguieron los participantes se resumen de forma breve como las siguientes:

- a) *Tarea 1. Registro, login y cambio de datos.* El participante recibe un mail con una invitación a un proyecto. Siguiendo el enlace se registrará e iniciará sesión. Entonces se solicitará que cambie su nombre a uno más descriptivo desde el apartado de Cuenta.
- b) *Tarea 2. Análisis de datos en pestaña de Overview.* Tras la toma de contacto, se le pide que abra el proyecto al que ha sido invitado y comente el sprint con mayor incremento de PBIs (Scope Creep) y analice el gráfico de PoC.
- c) *Tarea 3. Gestión del equipo de un proyecto.* Se le pide al participante que expulse a un usuario de prueba e invite al mismo con el rol de Desarrollador.
- d) *Tarea 4. Gestión de sprints.* El participante debe pasar al siguiente sprint del proyecto y escribir un objetivo para el sprint.
- e) *Tarea 5. Creación y priorización de PBIs.* A la persona participante se le pide que introduzca en el sistema un PBI entregado mediante un documento, incluyendo todos los datos y atributos que se incluyan en dicho documento, y lo priorice el primero en la lista.

f) *Tarea 6. Gestión interna de PBIs y documentación.* Se pide también que edite otro PBI buscándolo y se adjunte el documento anterior y un comentario en este PBI. Además, se debe establecer un criterio de aceptación al PBI y marcarlo como realizado, así como una dependencia con el PBI anterior. Tras esto, se marca el PBI como hecho.

g) *Tarea 7. Manejo del sistema de predicciones.* Se pide al participante que analice en qué sprint es más probable que finalice el proyecto, teniendo en cuenta la velocidad de los últimos 5 sprints. También se pide que maneje las predicciones por regresión con la misma métrica.

4.3.3. RESULTADOS

Al finalizar todos los participantes, se agradeció su esfuerzo y se discutieron y anotaron todos los aspectos mejorables, problemas y funcionalidades que se podrían incorporar a la aplicación o que ellos vieran necesarios para un uso empresarial, accesibles en el ANEXO C. Para evaluar los resultados se utilizaron varias métricas:

- Para medir la efectividad y la eficiencia se evaluaron datos objetivos: tiempo medio de realización de las tareas, tasa de completitud de las tareas, número de errores y asistencias, entre otros.
- Para medir la satisfacción de los usuarios se evaluaron datos subjetivos. Se utilizó una encuesta con preguntas relacionadas con la percepción de cada participante sobre la aplicación: satisfacción general, eficiencia, facilidad de uso, apariencia visual, claridad y fluidez de las interfaces; además, se incluyeron varias cuestiones de respuesta breve, para obtener retroalimentación adicional sobre elementos que fueran necesarios y no estuvieran implementados, así como opiniones sobre la aplicación.

La Figura 4.21 muestra los resultados subjetivos globales que se obtuvieron como resultado de las pruebas. Se puede observar que, en general, son unas cifras muy positivas que aportan una hoja de ruta a seguir sobre las características implementadas acertadamente y las que deben ser mejoradas para asegurar el éxito del proyecto.

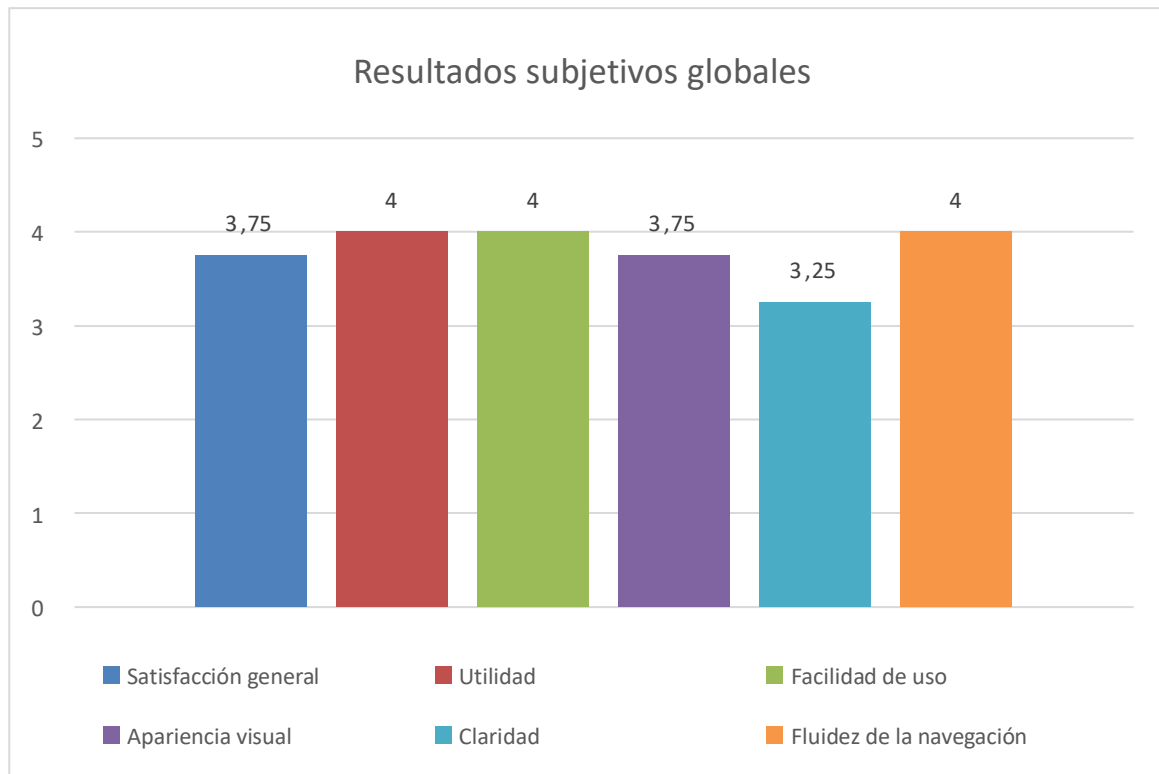


Figura 4.21: Resultados subjetivos globales de las pruebas de usabilidad

Los objetivos, procedimiento, métricas y resultados se pueden encontrar de forma detallada en el informe adjunto en el ANEXO B, el cual sigue el formato CIF del estándar ISO/IEC 25062:2006 [32] anteriormente mencionado.

4.4. CONCLUSIONES

En lo que respecta al trabajo realizado, en este capítulo se ha documentado toda la información generada de manera rigurosa y estricta, planificando todas y cada una de las actividades a realizar y almacenando toda la información generada. Una pieza clave del éxito del desarrollo ha sido hacer un buen uso de dicha información con dos finalidades distintas. La primera: tener siempre conocimiento del estado actual del proyecto, utilizando los artefactos y predicciones anteriormente documentados; y la segunda: generar un flujo constante de mejora tras cada sprint gracias a la retroalimentación realizada y, adicionalmente, gracias a la realización de pruebas de usabilidad. Todo en conjunto ha permitido mejorar en gran medida los resultados obtenidos en cada uno de los sprints y, por ende, mejorar la calidad final del producto software.

CAPÍTULO 5. THE PRODUCT OWNERD APP

*“The higher the price of information in a software team, the less effective the team is”
- Yegor Bugayenko.*

5.1. INTRODUCCIÓN

Tras haber documentado en la sección anterior los procesos ingenieriles seguidos durante el desarrollo de la aplicación, este capítulo se enfoca en presentar el resultado final de todo este esfuerzo anteriormente desglosado: TheProductOwnerd. Este es el nombre de la aplicación, que mezcla los conceptos de Product Owner y *Nerd* tanto en su título como en el logo de la web (ver Figura 5.1). El capítulo se divide en cuatro secciones, las cuales agrupan las funcionalidades principales de la aplicación final. La sección 5.2 aúna la gestión de los usuarios. En dicha sección se desarrollarán las vistas de inicio de sesión, registro y datos de usuarios. La sección 5.3 describe el “Overview” que la aplicación proporciona al usuario. La sección 5.4 desarrolla la vista del Product Backlog implementado en la herramienta. Por último, la sección 5.5 resume los métodos de predicción incorporados a la aplicación.

5.2. GESTIÓN DE USUARIOS

La primera imagen que un usuario tiene de la aplicación es su pantalla de inicio (ver Figura 5.1). En ella puede iniciar sesión con una cuenta o, si por el contrario no tuviera una, puede acceder al apartado de registro. Cada usuario tiene una vista “Home” en la que se listan todos los proyectos a los que pertenece dicho usuario y desde la que se pueden crear nuevos proyectos. Para agilizar el acceso a los mismos desde la barra de navegación, existe la opción de agregar los proyectos a favoritos, apareciendo estos en dicha barra de navegación. Junto a estas opciones se encuentra el acceso a los datos personales, en la pestaña de “Account”. Desde aquí el usuario puede modificar sus credenciales y contraseña.

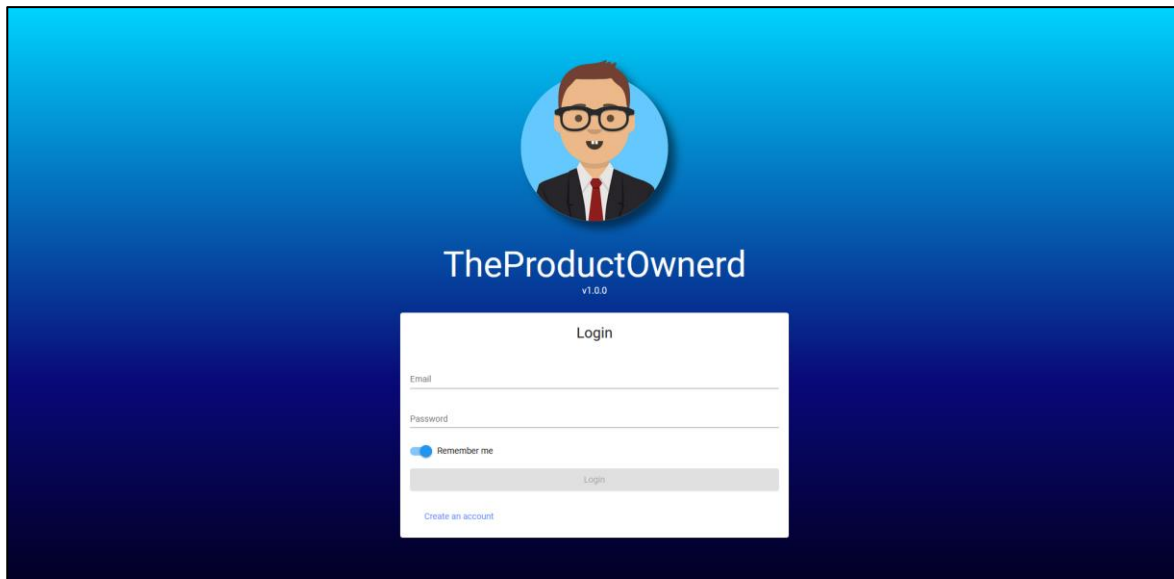


Figura 5.1: Login de usuarios

5.3. OVERVIEW

Al abrir un proyecto se carga una vista que incluye una cabecera con información sobre el proyecto: título, descripción del proyecto, sprint actual del desarrollo y sprint goal para este, así como la Deadline asignada, que equivale a la fecha de fin del proyecto (ver Figura 5.2). Además de estos datos, siempre visibles desde cualquiera de las vistas descritas a continuación, se muestra una barra de navegación que permite desplazarse entre tres menús: Overview, Backlog y Forecasts.

En la pestaña de Overview (Figura 5.3) se muestra la siguiente información descrita de izquierda a derecha: el Project Burndown Chart, que muestra el quemado de PBIs hasta el sprint actual. Este gráfico diferencia el quemado por etiquetado, es decir, desglosa los puntos historia quemados en función del tipo de PBI. Actualmente se incluyen solo cuatro etiquetas, correspondientes con los cuatro colores de la velocidad descritos en el apartado 2.3.6. Además de esta información, se puede activar el modo “Scope Creep”, que agrega al gráfico en forma de puntos negativos la carga de puntos historia que se ha creado en cada sprint. Esto permite detectar si algún sprint ha sufrido carga extra de nuevo trabajo, para justificar la tardanza en entregas. Junto a esta opción se muestran la velocidad media por sprint del proyecto y el throughput medio (número de PBIs quemados por sprint).

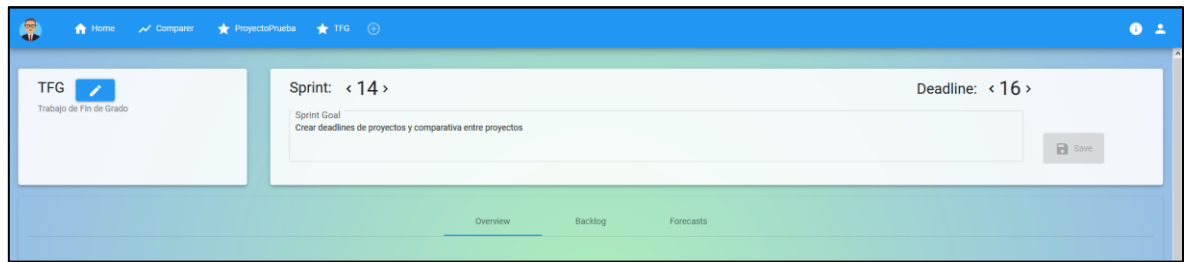


Figura 5.2: Cabecera de un proyecto

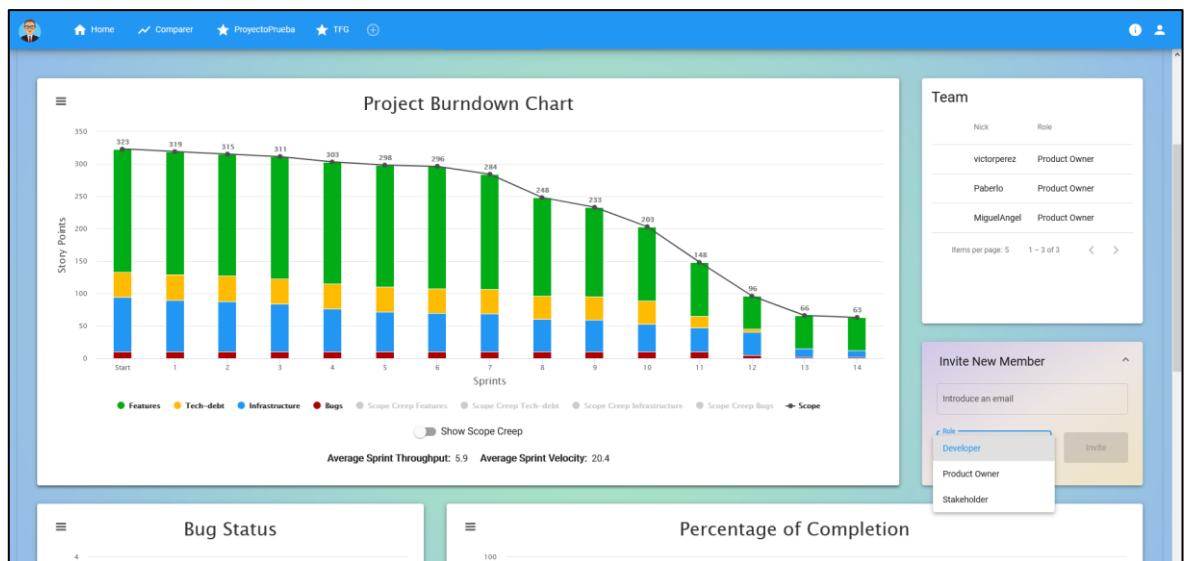


Figura 5.3: Project Burndown Chart y Equipo

Junto al gráfico se muestra en una tabla el equipo que conforma el proyecto. Se listan los usuarios utilizando el “Nick” o nombre de usuario de cada uno y el rol que desempeñan. El rol es un elemento importante en la aplicación, pues delimita los permisos que tiene el usuario para un proyecto. Actualmente existen tres roles:

- **Product Owner:** tiene acceso a todas las vistas de un proyecto y puede modificar todos los campos de un proyecto, así como reordenar el Product Backlog.
- **Developer:** tiene limitado el acceso a vistas y no puede editar un proyecto ni invitar a nuevos usuarios.
- **Stakeholder:** de forma similar al Developer, carece de permisos para modificar un proyecto, pero al contrario que el primero, tiene acceso a la pestaña de predicciones.

CAPÍTULO 5: The Product Ownerd App

Los permisos son los siguientes:

- **Ordenar PBIs:** permite ordenar los PBIs del Backlog por prioridad.
- **Editar PBIs:** da acceso a la edición de los campos de un PBI.
- **Estimar Tamaños:** permite asignar un valor número asociado al esfuerzo de realizar un PBI.
- **Mantener Usuarios:** permite invitar a nuevos usuarios a un proyecto o expulsar a los ya existentes.
- **Archivar Proyectos:** da acceso al usuario para archivar el proyecto (no implementado).
- **Set Done:** permite al usuario marcar un PBI como “Done” o Hecho.
- **Ver Proyecciones:** Da acceso a la vista de predicciones.
- **Estimar Valor:** permite asignar un valor número asociado al valor que este PBI aporta al cliente.
- **Editar Sprint Goals:** permite al usuario modificar el sprint actual, los sprint goals y la fecha límite del proyecto.
- **Editar Visión:** permite editar la visión del proyecto.

En la Tabla 5.1 se muestran los permisos detallados para cada tipo de usuario.

ROL	Ordenar PBIs	Editar PBIs	Estimar Tamaños	Mantener Usuarios	Archivar Proyectos	Set Done	Ver Proyecciones	Estiamr Valor	Editar Sprint Goals	Editar Visión
Product Owner										
Developer										
Stakeholder										

Tabla 5.1: Matriz de roles-permisos

Debido a los distintos permisos que existen, cada usuario visualizará los elementos de la web de diferente manera si posee un rol u otro. Por ejemplo, un usuario que forme parte del equipo de desarrollo no podrá ver los botones de Guardar que se encuentran en la cabecera del proyecto ni las flechas de cambio de sprint.

Cuando un usuario crea un proyecto se le asigna automáticamente el rol de Product Owner. Si desea invitar a nuevos usuarios al proyecto debe utilizar la caja de invitación colocada debajo de la tabla del equipo. Introduciendo un email y el rol que desea que desempeñe, la aplicación actuará de la siguiente forma: si el email tiene asociada una cuenta en la aplicación, se unirá automáticamente dicha cuenta al proyecto con el rol designado; si por el contrario no se encontrase ninguna cuenta asociada, la aplicación enviará un correo automatizado de invitación (ver Figura 5.4), el cual incluirá un enlace para que el usuario receptor acceda a la pestaña de registro utilizando un token de invitación; si el token de invitación es válido y no ha caducado, tras registrarse con éxito se unirá automáticamente al proyecto del cual recibió invitación.

Debajo de estos paneles (ver Figura 5.5) se encuentra la vista de “Bug Status”, que proporciona información sobre el número de errores abiertos en cada sprint y el número total de errores cerrados. A su lado se coloca el gráfico de PoC, que proporciona para cada sprint el porcentaje de PBIs completados sobre el total. Al igual que el PBCh, desglosa estos valores por etiqueta. Por último, se incluye un cuadro de texto con la Visión del proyecto.

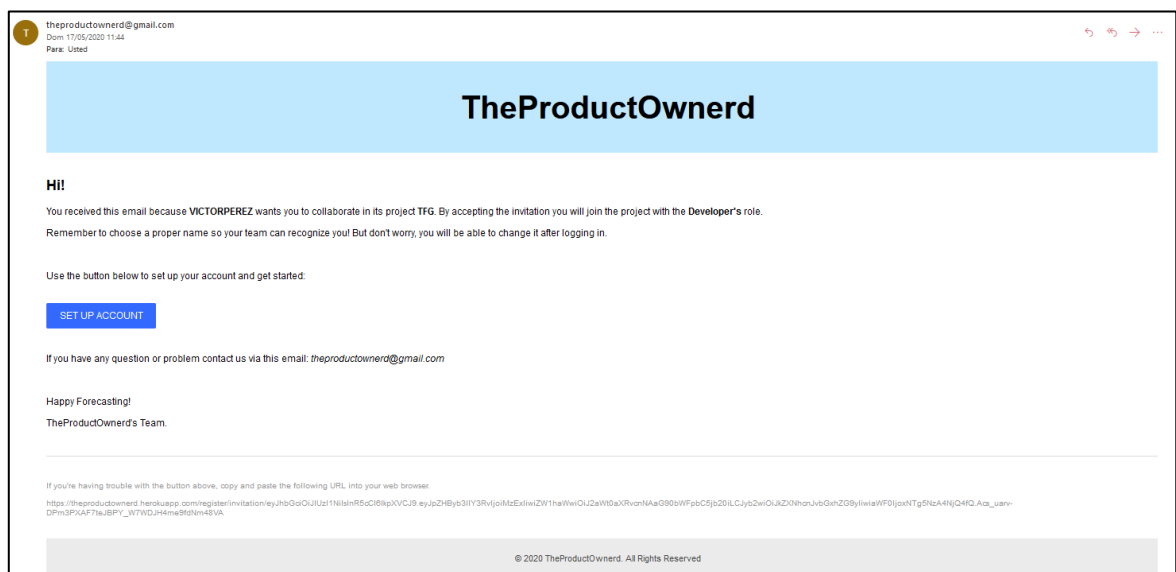


Figura 5.4: Email de invitación

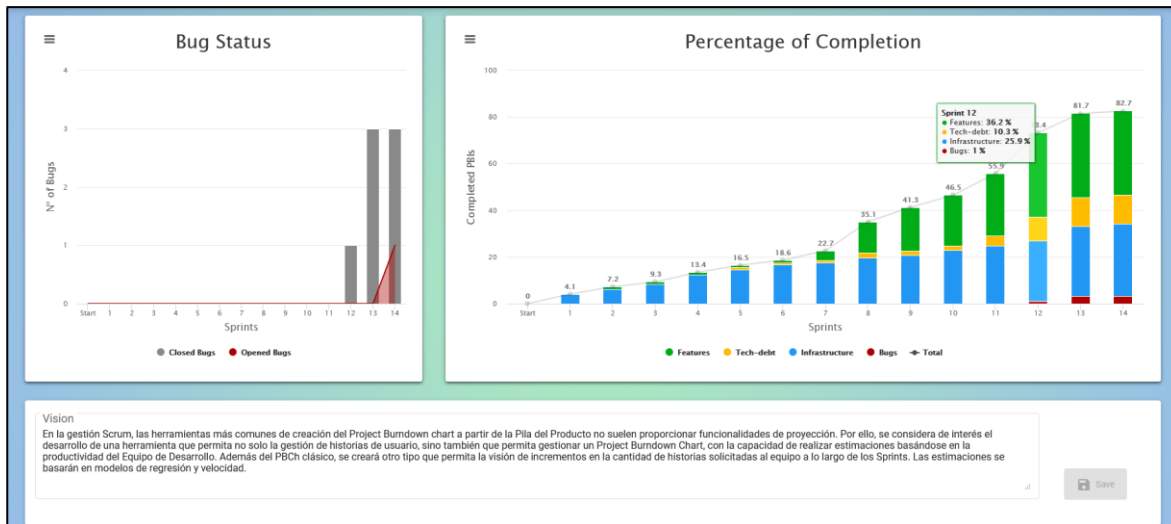


Figura 5.5: Bug Status, PoC y Visión

Es interesante remarcar que todos los gráficos pueden ser vistos en detalle colocando el puntero en el punto que se desee. Se pueden mostrar u ocultar las series simplemente pulsando en ellas en la leyenda. El icono de menú colocado en la esquina superior izquierda de cada gráfico permite descargar este en diversos formatos de imagen y PDF.

5.4. BACKLOG

La interfaz Backlog representa el Product Backlog de un proyecto en Scrum. Los PBIs de la pila se muestran en forma de tarjetas ordenadas verticalmente (la Figura 5.6 muestra el aspecto del Backlog cuando el usuario arrastra un PBI), incluyendo el título e ID del PBI y, si la hubiese, su descripción, además de la etiqueta asignada y la estimación, en caso de haber sido estimado previamente.

Para filtrar los PBIs se pueden utilizar varios de los elementos colocados sobre la pila. El buscador permite filtrar por título o descripción de PBI. También se pueden filtrar los PBIs por etiqueta u ordenar estos por priorización (Manual) o por Valor (ascendente o descendente). Un Product Owner puede cambiar la priorización de los PBIs arrastrando un PBI desde su icono de arrastrar situado en el lado izquierdo de la tarjeta y moviéndolo a la posición deseada. Mediante el botón de creación situado a la derecha se abre un diálogo que permite crear un PBI.

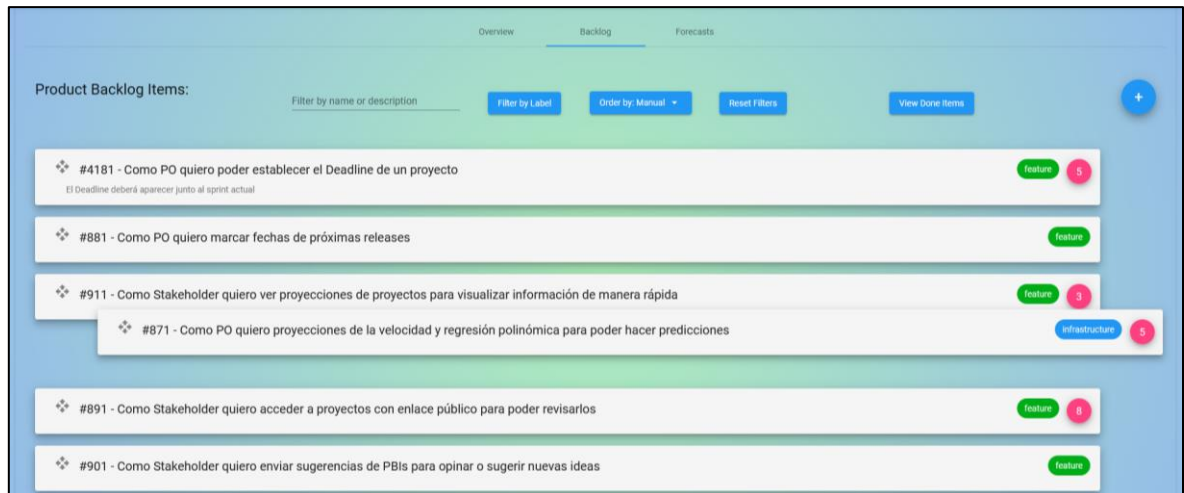


Figura 5.6: Vista del Backlog

Al seleccionar un PBI de la lista se abre un diálogo mostrando todos los datos del PBI: título, etiqueta, estimación, valor, sprint de creación, descripción, estado del PBI (hecho o no). Las figuras Figura 5.7 y Figura 5.8 muestran la interfaz de un PBI en detalle. Además de los campos básicos se incluyen varios adicionales:

- *Criterios de Aceptación:* se permiten crear y eliminar criterios, que pueden ser marcados y desmarcados como cumplidos.
- *Dependencias:* se pueden crear dependencias con otros PBIs.
- *Archivos:* se pueden adjuntar archivos a un PBI, pudiendo asignarles un nombre a estos. Los archivos adjuntados podrán ser descargados desde esta sección.
- *Comentarios:* Se pueden escribir comentarios al final del PBI, mostrando autor y fecha de creación del comentario.

Esta vista es importante para el usuario, pues debe ser capaz de reconocer con facilidad los elementos, ya que es altamente probable que esté familiarizado con aplicaciones similares del mercado. Opciones como la creación de comentarios, listas de verificación y campos para adjuntar documentos suelen ser algunos de los elementos más comunes. Por estas razones se ha buscado diseñar una interfaz que funcione de manera similar a otras herramientas, facilitando el rápido aprendizaje por parte de los nuevos usuarios.

The screenshot shows a form for creating or editing a Product Backlog Item (PBI). At the top, there is a green header bar with the title "#4181 - Como PO quiero poder establecer el Deadline de un proyecto" and a pink circle with the number "5". Below the header, the form contains several input fields: "Title *" with the value "Como PO quiero poder establecer el Deadline de un proyecto", "Label *" with a dropdown menu showing "Feature", "Estimation" with a dropdown menu showing "5", "Value" with a dropdown menu showing "4", and "Created at Sprint" with a dropdown menu showing "14". There is also a "Description" field with the text "El Deadline deberá aparecer junto al sprint actual". Below the description field, there is a checkbox labeled "Mark as Done" and a "Save" button. Further down, there is a section titled "Acceptance Criteria" with a checkbox labeled "Se deberá poder cambiar de Deadline pinchando en la flecha de próximo o anterior" and a "New Acceptance Criteria" input field with a "Create" button.

Figura 5.7: Vista de un PBI - 1

The screenshot shows the lower part of a Product Backlog Item (PBI) form. It features three main sections: "Dependencies" with a header bar, a dropdown menu showing "infra" and "debt", and an "Add Dependencies" button; "Files" with a header bar, an "Add file" button, and a file named "Sprint4.pdf" with a download icon; and "Comments" with a header bar, a "Write a comment..." input field, a "Comment" button, and a comment from "hsimpson" dated "16:47 - 27 Feb of 2020" with the text "Queda pendiente revisar con el cliente las características de la interfaz".

Figura 5.8: Vista de un PBI - 2

Desde la opción “View Done Items” se puede cambiar la vista del Backlog para mostrar los PBIs marcados como finalizados, ya que estos no se muestran en la vista general del Backlog. En esta vista los controles de priorización y edición están desactivados para evitar modificaciones en PBIs ya terminados.

5.5. FORECASTS

La sección de Forecasts es con toda seguridad la interfaz más interesante para un Product Owner, pues le permite visualizar varios modelos predictivos basados en los datos que se tienen del proyecto. Se han incorporado tres métodos de cálculo de predicciones: predicción por velocidad, regresión lineal y regresión parabólica.

5.5.1. PREDICCIÓN POR VELOCIDAD

Este modelo calcula la predicción basándose en la velocidad media por sprint calculada para el conjunto de sprints al que se le desee aplicar la predicción. Este modelo se ha diseñado de forma similar al modelo de velocidad descrito en la sección 2.3.3. El panel de configuración expandible permite cambiar los parámetros del modelo. Modificando el número de mejores y peores sprints se puede obtener una mayor o menor amplitud de apertura entre las rectas de mejor y peor velocidad (verde y rojo). La Figura 5.9 muestra la vista de la predicción por velocidad.

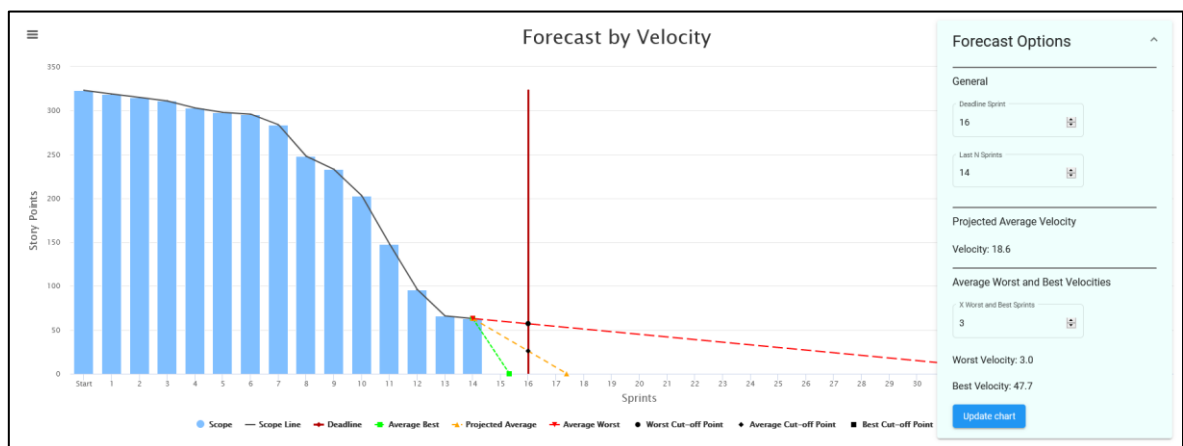


Figura 5.9: Predicción por velocidad

5.5.2. REGRESIÓN LINEAL

Se aplica un modelo de regresión lineal a partir del número de sprints indicado en las opciones. Una vez obtenida la fórmula, se calcula la recta en base a esta desde el punto inicial del cálculo de la predicción hasta el punto en el que se predice que finalice el proyecto. La Figura 5.10 muestra cómo se visualiza el modelo en la web.

5.5.3. REGRESIÓN PARABÓLICA

De forma similar a la regresión lineal, pero utilizando un modelo polinómico de segundo grado, este modelo reproduce la predicción en forma de parábola en el gráfico. Se advierte al usuario en la pestaña de configuración de que un bajo número de sprints puede resultar en una curva con la concavidad invertida, lo cual la hace incorrecta para una predicción de quemado. En la Figura 5.11 se puede observar el modelo generado contabilizando todos los sprints de un proyecto.

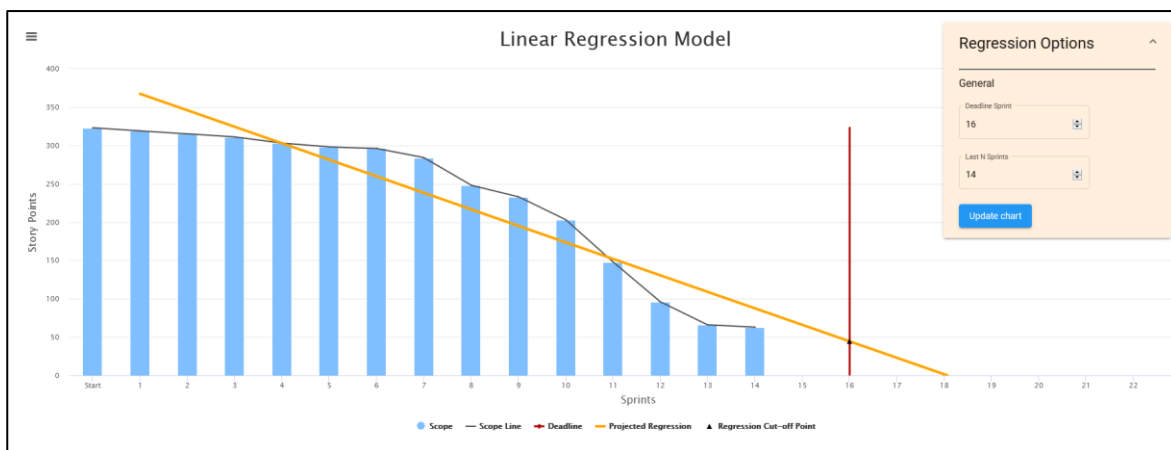


Figura 5.10: Modelo de regresión lineal

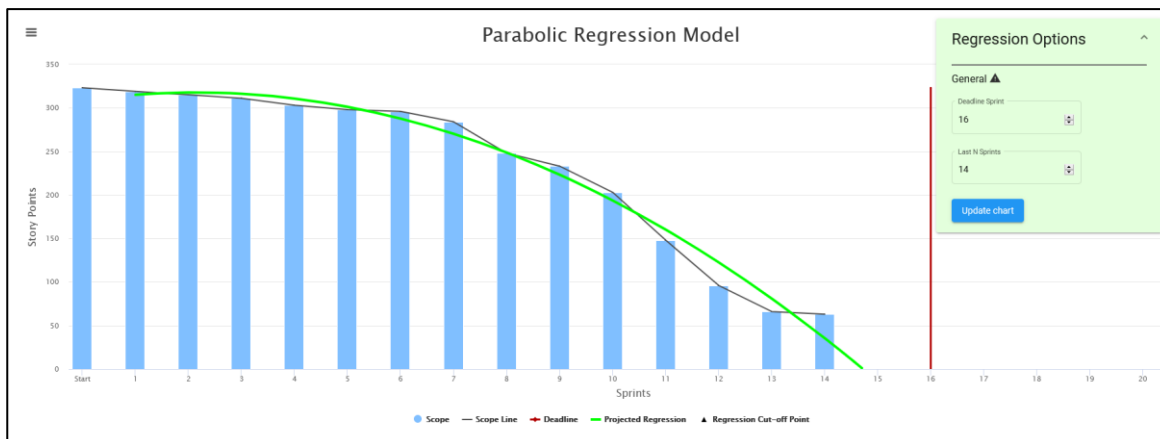


Figura 5.11: Modelo de regresión parabólica

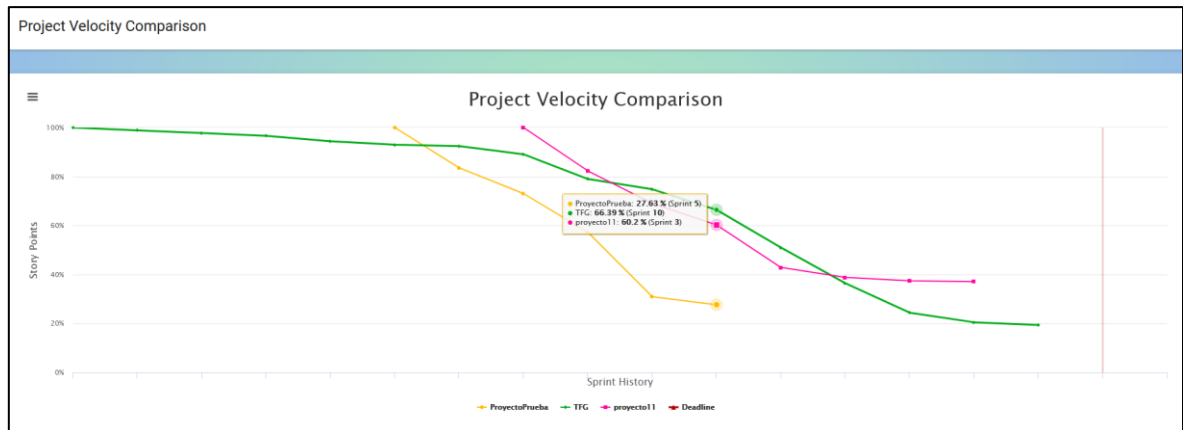


Figura 5.12: Comparativa entre proyectos

5.6. COMPARATIVA ENTRE PROYECTOS

Por último, en la barra de navegación se puede acceder a la vista de comparativa de proyectos (ver Figura 5.12). En esta vista se reúnen todos los proyectos de los que forma parte el usuario, y ordenándolos en el gráfico en función de su cercanía con su propia fecha de finalización. Esto permite detectar qué proyectos se encuentran en riesgo de no acabar a tiempo su trabajo y poder mover recursos entre ellos si así fuera necesario.

5.7. CONCLUSIONES

Durante el desarrollo de este capítulo se ha mostrado en profundidad la aplicación web TheProductOwnerd. Una aplicación de fácil uso y controles sencillos de utilizar que muestra una interfaz simple pero agradable a la vista. Recorriendo cada uno de los módulos de la aplicación se hace patente la gran cantidad de información sobre un proyecto que la web condensa en un vistazo, permitiendo administrar el equipo de un proyecto en unos pocos segundos. Su Product Backlog incorpora la gran mayoría de las opciones que se pueden encontrar en herramientas similares, dando gran flexibilidad al usuario y reduciendo su tiempo de aprendizaje. El factor determinante que hace de esta aplicación un servicio tan novedoso en el mercado es la introducción de modelos predictivos fácilmente configurables, que aportan al Product Owner la capacidad de anticiparse y prever riesgos en sus proyectos.

CAPÍTULO 6. CONCLUSIONES Y TRABAJO FUTURO

“Perfection is achieved not when there is nothing more to add, but rather when there is nothing more to take away” – Antoine de Saint-Exupéry.

6.1. INTRODUCCIÓN

En primer lugar, se resumen las conclusiones obtenidas tras la realización del Trabajo Fin de Grado en el apartado 6.2, haciendo hincapié en los resultados de este gran esfuerzo a nivel ingenieril pero también mostrando los efectos y beneficios a nivel personal. Seguidamente, en la sección 6.3 se menciona el trabajo pendiente de implementar en la aplicación, haciendo una breve observación del futuro próximo del mismo.

6.2. CONCLUSIONES

Como se menciona en la introducción, se desglosan las conclusiones obtenidas dividiendo estas en conclusiones de la herramienta, que resumen los logros de la aplicación; conclusiones a nivel académico, resaltando los factores e ideas que destaco como ingeniero; y conclusiones personales, en las que plasmo mi punto de vista tras la realización del trabajo.

6.2.1. CONCLUSIONES DE LA HERRAMIENTA

El producto desarrollado, TheProductOwnerd, ha acabado resultando en una herramienta de fácil acceso y gran utilidad para el usuario. Las interfaces son sencillas de utilizar y ofrecen controles intuitivos a los usuarios que estén familiarizados con otras herramientas ágiles. El diseño de datos de la aplicación permite introducir gran cantidad de información en esta y modificar o crear nuevos campos con gran rapidez y fluidez. Facilitar la gestión de equipos ha sido otro de los puntos fuertes en los que se ha enfocado la aplicación, poniendo especial

énfasis en acelerar el proceso de invitación de nuevos miembros. Por último, la funcionalidad más novedosa respecto del estado del arte y mejor lograda de la aplicación es el sistema de predicciones. Gracias a este, un Product Owner podrá utilizar estos modelos predictivos como apoyo para reforzar la toma de decisiones y asegurar el éxito de su proyecto.

6.2.2. CONCLUSIONES ACADÉMICAS

Realizar este proyecto ha supuesto para mí la culminación de mi formación como ingeniero informático. Llevar a cabo el trabajo necesario para cumplir con las expectativas de mis directores y, más importante aún, con las mías propias ha sido una ardua tarea en la que he tenido que aplicar todos los conocimientos aprendidos durante el grado. Por hacer un breve listado, he utilizado los conocimientos obtenidos en asignaturas como Sistemas de Información, Bases de Datos y Desarrollo de Bases de Datos para diseñar e implementar la base de datos que mantiene la información de la aplicación. En materia de procesos software y fundamentos, han sido de gran ayuda los conocimientos teóricos de Ingeniería del Software I-II, Gestión de Proyectos Software y Procesos de Ingeniería del Software. El diseño de interfaces de la web no habría sido posible llevarlo a cabo sin tener a mano los conceptos de Interacción Persona-Ordenador e Ingeniería Web y de Servicios. La programación modular, desacoplada y reusable debe su mérito al gran repaso que he tenido de esta en Diseño Software y varias asignaturas de programación anteriores. Y para finalizar, el esfuerzo en robustecer la calidad de la aplicación habría sido muy pobremente documentado y aplicado de no ser por Calidad de Sistemas Software. Haciendo este resumen, quiero hacer ver que todas las asignaturas aportan su granito de arena y que correctamente aplicadas y siguiendo un riguroso proceso ingenieril permiten dar forma a nuestras ideas.

Respecto a la metodología seguida en el proyecto, esta me ha aportado una nueva visión sobre el desarrollo ágil del software. La aplicación del pipeline de Integración y Entrega Continuos me ha permitido ver el gran potencial de DevOps. Esta forma de pensar me ha hecho renegar del concepto clásico de entrega en plazos mensuales o anuales, haciendo posible el despliegue automático de la aplicación en producción tras realizar cualquier cambio, por mínimo que sea. Reducir a un único click el número de acciones necesarias para compilar, ejecutar las pruebas y desplegar ha supuesto una gran reducción del tiempo que he tenido que dedicar a las entregas, lo cual ha desembocado en acelerar en gran medida el flujo

del desarrollo.

Más allá de lo anteriormente mencionado, he podido desarrollarme y descubrir nuevos conocimientos gracias a la labor que he llevado a cabo durante la práctica totalidad de el presente año académico. He podido formarme en mayor profundidad en los roles de Scrum y conocer distintos aspectos de este marco de trabajo que no habría llegado a conocer nunca si hubiera elegido una propuesta de trabajo diferente. Mi afán de mejora continua me ha conducido a mantenerme al tanto de las nuevas tecnologías y librerías que existen en Internet, buscando siempre una forma más eficiente, limpia y elegante de reordenar el código o reestructurar la arquitectura de la aplicación. Estos continuos quebraderos de cabeza los tomo como algo positivo, puesto que, debido a su efecto prácticamente superfluo desde un punto de vista externo, no los habría planteado si mi objetivo hubiera sido obtener un producto aceptable únicamente a ojos del usuario final.

6.2.3. CONCLUSIONES PERSONALES

La elaboración de este trabajo me ha hecho ver en su mayor magnitud algo que ya había percibido anteriormente: si algo te apasiona da igual si son las tres de la tarde o las cinco de la mañana, las horas vuelan mientras hay trabajo que hacer y este es gratificante. Huelga decir que no todo ha sido un camino de rosas. Me he topado con diversos problemas a lo largo del desarrollo que me han hecho cuestionar mi capacidad de sobrellevar este proyecto. Pero si algo he de destacar de este esfuerzo es que, tarde o temprano, los errores salen a la luz o se encuentran otras vías alternativas que permiten resolverlos.

A nivel de relaciones interpersonales creo ser un gran afortunado con mis directores (D. Pablo Bermejo López y D. Miguel Ángel Sánchez Cifo), pues la comunicación con ellos siempre ha sido rauda, clara y directa. Está claro que si te rodeas de un buen equipo todo se puede lograr. Respecto a la situación familiar y el entorno en el que he realizado el trabajo, he de decir que la coincidencia en fechas del trabajo con el cambio del modo de vida a nivel mundial ha desembocado en mi concentración casi total en este proyecto. También he de decir que si hubiese desarrollado el proyecto una situación normal el número de horas dedicadas al trabajo quizá hubiera debido ser algo inferior. Creo firmemente en que todo es bueno en su justa medida, y un buen trabajo solo es positivo si no afecta a otros aspectos de

tu vida cotidiana.

6.3. TRABAJO FUTURO

Tal y como se menciona en la Guía Scrum [1], “Un Product Backlog nunca está completo”. Puesto que se ha utilizado Scrum como metodología de desarrollo, presentar la aplicación en este trabajo no implica que sea el final de su camino. A lo largo del desarrollo se han incluido en el Backlog requisitos en forma de Product Backlog Items. Puesto que el alcance del Trabajo Fin de Grado es limitado, varios de estos PBIs no han llegado a ver la luz y siguen en el Product Backlog. Un breve resumen de las mejoras que no se han podido llevar a cabo es el siguiente:

- Dar la opción de crear Entregas para un proyecto.
- Crear enlaces públicos a proyectos para que los Stakeholders puedan acceder.
- Habilitar un buzón de sugerencias de PBIs para los Stakeholders.
- Crear predicciones utilizando el método de simulación de Monte Carlo.
- Estimar el tamaño de PBIs mediante Machine Learning.
- Agregar funcionalidad de importación y exportación de PBIs.
- Incluir un tablero Kanban conectado al Backlog para el equipo de desarrollo.
- Incluir un módulo que facilite la realización de técnicas de User Story Mapping.

Tras haber realizado este Trabajo Fin de Grado, es interesante remarcar mi intención de no dar por terminada la etapa de formación universitaria en el campo de la informática, ampliando esta mediante un Master Universitario en Ingeniería Informática. Con este propósito busco no solamente ampliar mis conocimientos en materia de gestión de proyectos y desarrollo software, sino aproximarme a otras áreas de la informática, como son la Inteligencia Artificial y la Ciencia de Datos. Los conocimientos y formación adquirida en el máster mencionado se aplicarán en su respectivo Trabajo Fin de Máster, extendiendo la funcionalidad de la aplicación TheProductOwnerd, siguiendo en esta ocasión un enfoque más próximo al Machine Learning y a los modelos predictivos aplicados a la gestión ágil del producto.

BIBLIOGRAFÍA

- [1] K. S. and J. Sutherland, *The Scrum Guide*. 2017.
- [2] J. W. G. Kim, J. Humble, P. Debois, *The DevOps Handbook: How to Create World-Class Agility, Reliability*, First Edit. IT Revolution, 2016.
- [3] J. Humble and D. Farley, *Continuous delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley, 2011.
- [4] Don McGreal and Ralph Jocham, *The Professional Product Owner: Leveraging Scrum as a Competitive Advantage*. Addison-Wesley Professional, 2018.
- [5] K. S. and J. Sutherland, “Changes to The Scrum Guide,” *Scrum.org*, 2016.
- [6] V. Duarte, *No Estimates. How to measure project progress without estimating*. .
- [7] “‘Least Squares Regression’ - Math is Fun.” [Online]. Disponible en: <https://www.mathsisfun.com/data/least-squares-regression.html>. [Accedido a: 20-Feb-2020].
- [8] “ZenHub.” [Online]. Disponible en: www.zenhub.com. [Accedido a: 04-Feb-2020].
- [9] “Jira Software.” [Online]. Disponible en: www.atlassian.com/software/jira. [Accedido a: 04-Feb-2020].
- [10] “Yodiz.” [Online]. Disponible en: yodiz.com. [Accedido a: 04-Feb-2020].
- [11] “NodeJS.” [Online]. Disponible en: <https://nodejs.org/en/>. [Accedido a: 10-Abr-2020].
- [12] “ExpressJS.” [Online]. Disponible en: <https://expressjs.com/es/>. [Accedido a: 10-Abr-2020].
- [13] “MySQL.” [Online]. Disponible en: <https://www.mysql.com/>. [Accedido a: 10-Abr-2020].
- [14] “Angular.” [Online]. Disponible en: <https://angular.io/>. [Accedido a: 10-Abr-2020].
- [15] “TypeScript.” [Online]. Disponible en: <https://www.typescriptlang.org/>. [Accedido a: 10-Abr-2020].
- [16] “REST Architectural Constraints.” [Online]. Disponible en: <https://restfulapi.net/rest-architectural-constraints/>. [Accedido a: 14-Abr-2020].
- [17] “GitHub.” [Online]. Disponible en: <https://github.com/>. [Accedido a: 28-Mar-2020].

Bibliografía

- [18] “TravisCI.” [Online]. Disponible en: <https://travis-ci.org/>. [Accedido a: 28-Mar-2020].
- [19] “Heroku.” [Online]. Disponible en: <https://www.heroku.com/>. [Accedido a: 28-Mar-2020].
- [20] “Slack.” [Online]. Disponible en: <https://slack.com>. [Accedido a: 28-Mar-2020].
- [21] Microsoft, “Microsoft Teams.” [Online]. Disponible en: <https://www.microsoft.com/es-es/microsoft-365/microsoft-teams/group-chat-software>. [Accedido a: 28-Mar-2020].
- [22] “Word.” [Online]. Disponible en: <https://products.office.com/en/word>. [Accedido a: 28-Mar-2020].
- [23] “Mendeley.” [Online]. Disponible en: <https://www.mendeley.com/>. [Accedido a: 28-Mar-2020].
- [24] “StarUML.” [Online]. Disponible en: <http://staruml.io/>. [Accedido a: 28-Mar-2020].
- [25] “Visual Studio Code.” [Online]. Disponible en: <https://code.visualstudio.com/>. [Accedido a: 28-Mar-2020].
- [26] “Postman.” [Online]. Disponible en: <https://www.postman.com/>. [Accedido a: 28-Mar-2020].
- [27] “DBeaver.” [Online]. Disponible en: <https://dbeaver.io/>. [Accedido a: 28-Mar-2020].
- [28] Ngx-rocket, “generator-ngx-rocket.” [Online]. Disponible en: <https://github.com/ngx-rocket/generator-ngx-rocket>. [Accedido a: 04-May-2020].
- [29] ISO/IEC, “ISO/IEC 25000:2005, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE,” 2005.
- [30] ISO/IEC, “ISO/IEC 25040:2011, Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Evaluation reference model and guide,” 2011.
- [31] ISO/IEC, “ISO/IEC 25022:2016, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of quality in use,” 2016.
- [32] ISO/IEC, “ISO/IEC 25062:2006, Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Common Industry Format (CIF) for usability test reports,” 2006.

ANEXOS

ANEXO A. GUION DE PRUEBAS DE USABILIDAD

GUION DE PRUEBAS - TheProductOwnerd

- Lo primero de todo, darte las gracias por participar en esta actividad.
- La prueba es muy sencilla, antes de empezar os pediré un correo electrónico al cual os mandaré una invitación desde la aplicación. Una vez os llegue el correo comienza la prueba. Esta está dividida en 9 actividades o tareas que queremos que probéis, para ver cómo de intuitivos son los controles y si os resulta fácil o difícil llevarlas a cabo.
- Comienza a contar el tiempo tras leer cada enunciado de tarea. Tras realizar una tarea debes apuntar debajo de cada tarea el tiempo que has tardado en llevarla a cabo. Para esto te dejo el enlace a un cronómetro online por si no tuvieses uno a mano: <https://reloj-alarma.es/cronometro/#>
- Las tareas intentan cubrir la mayor parte de las actividades que se llevarán a cabo en un uso real en la aplicación y se listan a continuación.

IMPORTANTE: si encuentras algún error durante la realización de las tareas o si necesitas ayuda porque no se entiende bien alguno de los enunciados o no encuentras algún menú de la aplicación, para el cronómetro y avísame para que te pueda ayudar a solucionarlo. Utiliza preferiblemente **Chrome** o **Firefox**, ya que no hemos probado a fondo la aplicación en otros navegadores.

- a) *Tarea 1. Registro, login y cambio de datos.* Comienza a contar el tiempo en cuanto recibas el correo y para el cronómetro cuando consigas registrarte, iniciar sesión y cambiar el Nick que has introducido en el registro a: <TuNick>+T1. (Simplemente agregarle T1 al final del Nick).

TIEMPO TRANSCURRIDO(s):

- b) *Tarea 2. Análisis de datos en pestaña de Overview.* Esta tarea busca ver si te resulta fácil analizar la información presentada en los gráficos del Project Burndown Chart y de Percentage of Completion. Comenta el sprint en el que has visto que se ha llevado a cabo más trabajo, en el que se han creado más puntos historia y qué tipos de PBIs están creciendo conforme los sprints avanzan.

COMENTARIO:

TIEMPO TRANSCURRIDO(s):

c) *Tarea 3. Gestión del equipo de un proyecto.* Verás que aparte de mí y de ti hay otro usuario llamado “expulsame” en el equipo. Este usuario ha sido movido de proyecto, así que debes expulsarlo. Tras esto te avisan de que debes invitar a otro trabajador al equipo, su cargo será Desarrollador y su correo es: “usuario@usuario.com”. Mide el tiempo transcurrido tras invitar a este usuario.

TIEMPO TRANSCURRIDO(s):

d) *Tarea 4. Gestión de sprints.* Cambia el sprint actual del proyecto, el objetivo del siguiente sprint es: “Arreglar los bugs surgidos en las pruebas”.

TIEMPO TRANSCURRIDO(s):

e) *Tarea 5. Creación y priorización de PBIs.* (Un PBI es un Product Backlog Item, puede ser una historia de usuario o tarea). Introduce en el sistema el siguiente PBI, y priorízalo **el primero** en el Product Backlog. Los campos del PBI son los siguientes:

Title: Realizar Tarea 5

Label: feature

Value: 10

Description: Tras la tarea se contará el tiempo

TIEMPO TRANSCURRIDO(s):

f) *Tarea 6. Gestión interna de PBIs y documentación.* Vamos a trabajar con otro PBI, su nombre es: PBI PARA CAMBIAR. Si no lo encuentras de un vistazo utiliza el buscador del backlog. Hay que cambiar algunos datos:

- Debes asignarle un criterio de aceptación nuevo que se llame: “encontrar PBI”. Marca el criterio de aceptación como cumplido.
- Debes crearle una dependencia con el pbi que creaste anteriormente (Realizar Tarea 5)
- Adjunta este guion como archivo al pbi PBI PARA CAMBIAR.
- Escribe en la sección de comentarios cualquier frase.

TIEMPO TRANSCURRIDO(s):

g) *Tarea 7. Manejo del sistema de predicciones.* Como Product Owner debes gestionar la información de que dispones, prueba a realizar predicciones para ver en qué estado del proyecto nos encontraríamos si se ha establecido que el Deadline del proyecto es en el sprint 8. Prueba a utilizar distintas métricas de las disponibles, cambia el número de sprints que se han tenido en cuenta, el grado polinomial, etc. Y comenta aquí abajo los resultados de forma breve. (Es posible que la predicción polinomial no funcione correctamente en algunos casos).

COMENTARIO:

TIEMPO TRANSCURRIDO(s):

Tras realizar las tareas te vuelvo a dar las gracias por tu ayuda. Por último, te adjunto un formulario anónimo con algunas preguntas de opinión personal sobre la aplicación: <https://forms.gle/onFHHP7s8aenBBia6>

Otra vez, muchas gracias por tu colaboración,
Víctor Pérez Piqueras.

**ANEXO B. ISO/IEC 25062 COMMON INDUSTRY FORMAT FOR USABILITY
TEST REPORTS**

TheProductOwnerd V1.0

Informe realizado por Víctor Pérez Piqueras

Fecha: 8/5/2020

Probado: 7/5/2020

Para:

Escuela Superior de Ingeniería Informática

Cualquier consulta sobre los contenidos del informe debe dirigirse a:

Víctor Pérez Piqueras

Escuela Superior de Ingeniería Informática
02071, Albacete
Universidad de Castilla-La Mancha

Victor.Perez6@alu.uclm.es

1 Introducción

1.1 Sumario ejecutivo

TheProductOwnerd es una aplicación web destinada a la ayuda y soporte en las actividades rutinarias y la gestión de información que el Product Owner (PO) realiza a lo largo del desarrollo de un proyecto ágil. Para demostrar su usabilidad se llevó a cabo una prueba de usabilidad realizando estas tareas en dicha aplicación.

Cuatro profesionales del sector tecnológico con amplio conocimiento en metodologías ágiles y experiencia en gestión de proyectos participaron en este estudio. Cada participante realizó las acciones básicas de la aplicación: registro de un nuevo usuario, inicio de sesión, gestión de la información del proyecto, gestión de Product Backlog Items (PBIs) y manejo de las predicciones que la aplicación provee. Se midieron en cada caso las tasas de error al realizar cada tarea, así como la tasa de finalización de tareas sin necesidad de ayuda externa. Tras finalizar la prueba se entregó un cuestionario con preguntas de carácter subjetivo referentes a la aplicación.

1.2 Descripción completa del producto

TheProductOwnerd provee al usuario de la capacidad de gestionar proyectos ágiles mediante una interfaz que facilita la incorporación de nuevos usuarios al proyecto. En el proyecto y en función del rol que desempeñe el usuario podrá: (I) visualizar el estado actual del proyecto y del Project Burndown Chart, (II) visualizar métricas como la velocidad y el Scope Creep, (III) gestionar el Product Backlog, (IV) crear y editar PBIs, pudiendo adjuntar ficheros, comentarios, dependencias y criterios de aceptación. También dispone de una herramienta generadora de predicciones que permite crear modelos visuales basados en la información ya conocida mediante diversos métodos estadísticos: velocidad media, regresión lineal y regresión polinómica. Por último, facilita la comparación del estado actual entre proyectos, para poder analizar y detectar posibles problemas en la distribución de recursos entre estos.

1.3 Objetivos de las pruebas

Las pruebas se realizaron con el objetivo de evaluar los siguientes aspectos de la herramienta: (I) utilidad, para comprobar si los usuarios a los que va dirigida la aplicación la encuentran útil, (II) facilidad de uso, pues la aplicación proporciona interfaces y menús que buscan facilitar su uso a los usuarios, (III) funcionalidad, pues se persigue que la aplicación contenga todas las funciones que necesiten los usuarios a los que esta va dirigida. La prueba estuvo enfocada al uso que un PO le dará a la herramienta para gestionar un proyecto y su información.

2 Método

2.1 Participantes

El conjunto de participantes estuvo formado en su totalidad por profesionales del sector tecnológico que tuvieran conocimientos sobre metodologías ágiles. Los requisitos para participar en la prueba fueron los siguientes:

- a) Conocimientos básicos de inglés.
- b) Haber tenido contacto con la gestión de proyectos.
- c) Conocer en profundidad tanto Scrum como el rol del PO.

Los participantes se seleccionaron invitando a varios de los contactos que se tenían en empresas del sector. La Tabla B.1 muestra la información básica de los participantes:

Participante	Nivel Académico	Ocupación/rol
P1	Ingeniería	Jefe de proyecto
P2	Doctorado	Responsable de I+D
P3	Doctorado	Responsable de desarrollo
P4	Ingeniería	Jefe de proyecto

Tabla B.1. Participantes en la prueba

Como se puede observar en la Tabla B.1, el 50% de los participantes poseen un alto nivel académico, y el 100% de ellos ocupa un rol de gestión dentro de su empresa.

2.2 Contexto de uso del producto en las pruebas

2.2.1 Tareas

Contexto de uso previsto: entrevistas con usuarios potenciales a los que se les pide que utilicen la herramienta para familiarizarse con su interfaz. Tras una primera toma de contacto, se les solicita que gestionen un proyecto simulado, creando y editando PBIs y gestionando los distintos gráficos e información que la aplicación genera.

Contexto usado para la prueba: las tareas seleccionadas para la evaluación fueron las siguientes:

- a) *Tarea 1. Registro, login y cambio de datos.* El participante recibe un mail con una invitación a un proyecto. Siguiendo el enlace se registrará e iniciará sesión. Entonces se le pedirá que cambie su nombre a uno más descriptivo desde el apartado de Cuenta.
- b) *Tarea 2. Análisis de datos en pestaña de Overview.* Tras la toma de contacto, se le pide que abra el proyecto al que ha sido invitado y comente el sprint con mayor incremento de PBIs (Scope Creep) y analice el gráfico de Percentage of Completion.
- c) *Tarea 3. Gestión del equipo de un proyecto.* Se le pide al participante que expulse a un usuario de prueba e invite al mismo con el rol de Desarrollador.
- d) *Tarea 4. Gestión de sprints.* El participante debe pasar al siguiente sprint del proyecto y escribir un sprint goal.
- e) *Tarea 5. Creación y priorización de PBIs.* A la persona participante se le pide que introduzca en el sistema un PBI entregado mediante un documento, incluyendo todos los datos y atributos que se incluyan en dicho documento, y lo priorice el primero en la lista.
- f) *Tarea 6. Gestión interna de PBIs y documentación.* Se pide al usuario que edite otro PBI buscándolo y se adjunte el documento anterior y un comentario en este PBI. Además, se debe establecer un criterio de aceptación al PBI y marcarlo como realizado, así como una dependencia con el PBI anterior. Tras esto, se marca el PBI como Done.

- g) *Tarea 7. Manejo del sistema de predicciones.* Se pide al participante que analice en qué sprint es más probable que finalice el proyecto, teniendo en cuenta la velocidad de los últimos 5 sprints. También se pide que maneje las predicciones por regresión con la misma métrica.

2.2.2 Entorno de pruebas

Contexto de uso pensado: dada la situación, el contexto de uso pensado fue el despacho o sala en el que cada participante estuviese más cómodo para realizar sus actividades de trabajo.

Contexto de uso en la prueba: la prueba se realizó de manera telemática mediante una reunión virtual. Cada participante recibió las instrucciones antes del comienzo de la prueba y se les informó del propósito de esta.

2.2.3 Entorno tecnológico de los participantes

Contexto de uso pensado: el ordenador o portátil en el que cada participante suele trabajar, con acceso a Internet y Google Chrome en la versión 83.0.4103.61 o Mozilla Firefox en la versión 76.0.1 como navegador web con JavaScript habilitado. Una herramienta de edición de documentos como Microsoft Word o Apache OpenOffice Writer.

Contexto de uso en la prueba: el portátil o computadora personal de cada participante, utilizando el navegador web Google Chrome en su versión 83.0.4103.61 y Microsoft Word como editor de documentos.

2.2.4 Herramientas de pruebas de administrador

La ejecución de la prueba fue medida mediante cronómetro por cada participante, para así obtener el tiempo que cada participante ha dedicado a cada tarea. Al final de la sesión, los participantes rellenaron un cuestionario online anónimo para valorar distintos aspectos de la aplicación.

2.3 Diseño de la prueba

Cuatro profesionales del sector tecnológico realizaron la prueba.

La tasa de cumplimiento sin ayuda, la tasa de cumplimiento con éxito, el tiempo de tarea y el número de errores surgidos durante la tarea se calcularon para las siguientes tareas:

- a) *Tarea 1.* Registro, login y cambio de datos.
- b) *Tarea 2.* Análisis de datos en pestaña de Overview.
- c) *Tarea 3.* Gestión del equipo de un proyecto.
- d) *Tarea 4.* Gestión de sprints.
- e) *Tarea 5.* Creación y priorización de PBIs.
- f) *Tarea 6.* Gestión interna de PBIs y documentación.
- g) *Tarea 7.* Manejo del sistema de predicciones.

2.3.1 Procedimiento

Tras iniciar la reunión virtual, a los participantes se les informó de los objetivos de la actividad referentes a probar distintas propiedades de la aplicación. Se les explicó que esta actividad no era una prueba de habilidad para ellos y que no había compromiso alguno. Se les informó a los participantes de los cuestionarios de evaluación. También se les pidió que rellenasen el formulario con algunos datos informativos sobre ellos mismos, sus habilidades y sus conocimientos técnicos.

Se les dieron unas instrucciones simples en un guion en formato Microsoft Word sobre cómo se iba a llevar a cabo la prueba y se les informó que debían anotar el tiempo que tardasen en realizar cada tarea, así como los comentarios requeridos asociados a cada tarea.

Al final del guion se proporcionó un cuestionario sobre los resultados e impresiones subjetivas de la aplicación.

También se les preguntó tras acabar las tareas qué aspectos de la actividad o de la aplicación habían encontrado más complejos o difíciles de realizar. Por último, se estableció una conversación para obtener sugerencias de los encuestados (los usuarios participantes más otros dos invitados a la sesión), a modo de última fase de Sprint Review.

Al acabar la prueba, se les agradeció el esfuerzo por participar.

2.4 Métricas

2.4.1 Efectividad

- *Tasa de finalización*. Porcentaje de finalización por tarea.
- *Errores*. Número de errores que aparecieron durante la tarea.
- *Asistencias*. Número de peticiones de ayuda por parte de los participantes para cada tarea.

2.4.2 Eficiencia

- *Tiempo de tarea*: tiempo medio necesario para completar cada tarea (para tareas completadas correctamente).
- *Eficiencia de la tasa de finalización*: tasa media de finalización / tiempo medio de tarea.

2.4.3 Satisfacción

La satisfacción se midió utilizando una escala de calificaciones subjetivas al final de la sesión, dando puntuaciones para la percepción de cada participante de satisfacción general, eficiencia, facilidad de uso, apariencia visual, claridad y fluidez de las interfaces.

3 Resultados

3.1 Tratamiento de datos

Media de éxito de objetivos: se calculó el grado medio con el que cada tarea fue completada de forma correcta, en forma de porcentaje.

El impacto de negocio de los distintos errores potenciales diarios en la aplicación se discutió con varios participantes, llevando a calcular la siguiente estructura de puntuación de objetivos:

- *Tarea 1.* Sobre 100%; por cada error cometido en el proceso se restan 33 puntos porcentuales.
- *Tarea 2.* Sobre 100%; por cada gráfico en el que no se entiendan los datos de manera rápida se restan 50 puntos porcentuales; por cada pregunta que no se formule correctamente se restan 33 puntos porcentuales.
- *Tarea 3.* Sobre 100%; por cada error cometido en el proceso de invitar a un miembro y expulsar a otro se restan 50 puntos porcentuales.
- *Tarea 4.* Sobre 100%; por cada error cometido en el proceso de cambio de sprint y creación de sprint goal se restan 50 puntos porcentuales.
- *Tarea 5.* Sobre 100%; por cada error cometido en el proceso de introducir los datos en un PBI se restan 20 puntos porcentuales; si no se consigue priorizar el PBI se restan 50 puntos porcentuales.
- *Tarea 6.* Sobre 100%; por cada error cometido en el proceso de adjuntar información a un PBI se restan 20 puntos porcentuales.
- *Tarea 7.* Sobre 100%; por cada error cometido en el proceso de calcular predicciones se restan 33 puntos porcentuales.

3.2 Presentación de resultados

Se presentan los resultados obtenidos:

3.2.1 Tarea 1

Participante	Tasa de finalización	Errores	Asistencias	Tiempo de tarea (s)	Eficiencia de la tasa de finalización
P1	100%	0,00	0,00	45,00	2,22
P2	100%	0,00	0,00	81,00	1,23
P3	100%	0,00	0,00	58,00	1,72
P4	100%	0,00	0,00	107,00	0,93
Media	100%	0,00	0,00	72,75	1,52
Error estándar	0,00	0,00	0,00	13,63	0,41
Desviación estándar	0,00	0,00	0,00	23,60	0,49
Mínimo	100%	0,00	0,00	45,00	0,93
Máximo	100%	0,00	0,00	107,00	2,22

Tabla B.2. Resultados de la Tarea 1

3.2.2 Tarea 2

Participante	Tasa de finalización	Errores	Asistencias	Tiempo de tarea (s)	Eficiencia de la tasa de finalización
P1	100%	0,00	0,00	290,00	0,34
P2	100%	0,00	0,00	360,00	0,27
P3	100%	0,00	0,00	392,00	0,25
P4	66,00%	0,00	0,00	279,00	0,23
Media	91,50%	0,00	0,00	330,25	0,27
Error estándar	8,50	0,00	0,00	27,30	0,00
Desviación estándar	14,72	0,00	0,00	47,28	0,04
Mínimo	66,00%	0,00	0,00	279,00	0,23
Máximo	100%	0,00	0,00	360,00	0,34

Tabla B.3. Resultados de la Tarea 2

3.2.3 Tarea 3

Participante	Tasa de finalización	Errores	Asistencias	Tiempo de tarea (s)	Eficiencia de la tasa de finalización
P1	100%	0,00	0,00	32,00	3,12
P2	100%	0,00	0,00	56,00	1,78
P3	100%	0,00	0,00	32,00	3,12
P4	100%	0,00	0,00	95,00	1,05
Media	100%	0,00	0,00	53,75	2,26
Error estándar	0,00	0,00	0,00	14,87	0,57
Desviación estándar	0,00	0,00	0,00	25,75	0,89
Mínimo	100%	0,00	0,00	32,00	1,05
Máximo	100%	0,00	0,00	95,00	3,12

Tabla B.4. Resultados de la Tarea 3

3.2.4 Tarea 4

Participante	Tasa de finalización	Errores	Asistencias	Tiempo de tarea (s)	Eficiencia de la tasa de finalización
P1	100%	0,00	0,00	47,00	2,13
P2	100%	0,00	0,00	30,00	3,33
P3	100%	0,00	0,00	81,00	1,23
P4	100%	0,00	0,00	37,00	2,70
Media	100%	0,00	0,00	48,75	2,35
Error estándar	0,00	0,00	0,00	11,30	0,41
Desviación estándar	0,00	0,00	0,00	19,57	0,77
Mínimo	100%	0,00	0,00	30,00	1,23
Máximo	100%	0,00	0,00	81,00	3,33

Tabla B.5. Resultados de la Tarea 4

3.2.5 Tarea 5

Participante	Tasa de finalización	Errores	Asistencias	Tiempo de tarea (s)	Eficiencia de la tasa de finalización
P1	100%	0,00	0,00	31,00	3,22
P2	100%	0,00	0,00	195,00	0,51
P3	100%	0,00	0,00	114,00	0,87
P4	50,00%	0,00	0,00	220,00	0,22
Media	87,50%	0,00	0,00	140,00	1,20
Error estándar	12,50	0,00	0,00	42,79	0,75
Desviación estándar	21,65	0,00	0,00	74,13	1,18
Mínimo	50,00%	0,00	0,00	31,00	0,22
Máximo	100%	0,00	0,00	220,00	3,22

Tabla B.6. Resultados de la Tarea 5

3.2.6 Tarea 6

Participante	Tasa de finalización	Errores	Asistencias	Tiempo de tarea (s)	Eficiencia de la tasa de finalización
P1	100%	0,00	0,00	112,00	0,89
P2	100%	0,00	0,00	122,00	0,82
P3	100%	0,00	0,00	110,00	0,91
P4	100%	0,00	0,00	104,00	0,96
Media	100%	0,00	0,00	112,00	0,89
Error estándar	0,00	0,00	0,00	3,74	0,00
Desviación estándar	0,00	0,00	0,00	6,48	0,05
Mínimo	100%	0,00	0,00	104,00	0,82
Máximo	100%	0,00	0,00	122,00	0,96

Tabla B.7. Resultados de la Tarea 7

3.2.7 Tarea 7

Participante	Tasa de finalización	Errores	Asistencias	Tiempo de tarea (s)	Eficiencia de la tasa de finalización
P1	100%	0,00	0,00	154,00	0,65
P2	100%	0,00	0,00	217,00	0,46
P3	100%	0,00	0,00	152,00	0,66
P4	100%	0,00	0,00	240,00	0,42
Media	100%	0,00	0,00	190,75	0,55
Error estándar	0,00	0,00	0,00	22,29	0,00
Desviación estándar	0,00	0,00	0,00	38,62	0,11
Mínimo	100%	0,00	0,00	152,00	0,42
Máximo	100%	0,00	0,00	240,00	0,66

Tabla B.8. Resultados de la Tarea 7

3.3 Resultados de rendimiento combinados

Participante	Tasa de finalización	Errores	Asistencias	Tiempo de tarea total (s)	Eficiencia de la tasa de finalización
P1	100%	0,00	0,00	711,00	0,14
P2	100%	0,00	0,00	1061,00	0,09
P3	100%	0,00	0,00	939,00	0,11
P4	88,00%	0,00	0,00	1103,00	0,08
Media	97,00%	0,00	0,00	951,00	0,10
Error estándar	3,00	0,00	0,00	87,00	0,00
Desviación estándar	5,19	0,00	0,00	150,70	0,02
Mínimo	88,00%	0,00	0,00	711,00	0,08
Máximo	100,00%	0,00	0,00	1103,00	0,14

Tabla B.9. Resultados de rendimiento combinados

3.3.1 Análisis de los datos objetivos

La información contenida en la Tabla B.9, que a su vez combina los resultados de las tablas B.1-B.8, permite analizar la efectividad y eficiencia de la aplicación.

Se observa que el 100% de los participantes pudo completar el guion proporcionado sin requerir de ayuda externa, cometer fallos ni interpretar los conceptos de forma errónea en el transcurso de las tareas.

La tasa de finalización de tareas es del 97%, lo cual indica que la mayoría de los participantes pudo completar todos los pasos de cada tarea.

El tiempo total para realizar todas las tareas se encuentra entre los 12 y 18 minutos, un tiempo asumible teniendo en cuenta que ningún usuario conocía la aplicación y pudo utilizar todas las funcionalidades que esta dispone.

Se desglosan las tareas:

- a) *Tarea 1.* Registro, login y cambio de datos. Tiempo medio (s): 72,75.

En el tiempo indicado cada participante pudo registrarse, entrar en su cuenta y cambiar sus datos. Es una medición positiva para esta tarea.

- b) *Tarea 2.* Análisis de datos en pestaña de Overview. Tiempo medio (s): 330,25.

Esta tarea requirió que el usuario analizara los datos presentados en el Project Burndown Chart y el Percentage of Completion. Se observa que tomó bastante tiempo, los motivos pueden ser varios: la entrada en frío a analizar datos por parte de los participantes, el desconocimiento de las interfaces, la descripción confusa del enunciado de la tarea, etc.

- c) *Tarea 3.* Gestión del equipo de un proyecto. Tiempo medio (s): 53,75.

Los participantes pudieron expulsar a un usuario e invitar a otro con relativa brevedad. Esta tarea quizá requiera de unos pocos segundos para un usuario experimentado. Un factor que pudiese afectar es la disposición del botón de expulsar, desactivado hasta que no se coloca el puntero encima de este.

d) *Tarea 4.* Gestión de sprints. Tiempo medio (s): 48,75.

En el tiempo indicado cada participante pudo cambiar el sprint y el sprint goal. Este tiempo se estima que puede bajar considerablemente una vez se haya utilizado.

e) *Tarea 5.* Creación y priorización de PBIs. Tiempo medio (s): 140,00.

Sin conocer cómo llevar a cabo esta tarea, los participantes encontraron la opción para crear un PBI con los datos aportados. Esta tarea de por sí es sencilla. El problema que ha generado un aumento del tiempo de finalización se debe a la dificultad de encontrar la forma de priorizar este PBI en la interfaz.

f) *Tarea 6.* Gestión interna de PBIs y documentación. Tiempo medio (s): 112,00.

Una vez han realizado la tarea anterior, esta tarea llevó menos tiempo realizarla, pese a tener más campos que rellenar.

g) *Tarea 7.* Manejo del sistema de predicciones. Tiempo medio (s): 190,75.

Al igual que en la Tarea 2, esta lleva más tiempo debido a que se pide al usuario que analice la información que se le presenta y emita un juicio de valor sobre esta. No obstante, en el tiempo dado se pudieron analizar todos los datos aportados por el sistema de predicciones, siendo esta la primera vez que lo utilizaban.

3.4 Resultados de satisfacción

3.4.1 Resultados subjetivos

Todos los valores utilizan una escala de 0 a 5, siendo 0 la peor calificación y 5 la mejor posible.

Participante	Satisfacción general	Utilidad	Facilidad de uso	Apariencia visual	Claridad	Fluidez de navegación
P1	3,00	4,00	4,00	3,00	3,00	4,00
P2	4,00	4,00	4,00	3,00	1,00	3,00
P3	4,00	4,00	4,00	5,00	4,00	4,00
P4	4,00	4,00	4,00	4,00	5,00	5,00
Media	3,75	4,00	4,00	3,75	3,25	4,00
Error estándar	0,25	0,00	0,00	0,48	0,85	0,41
Desviación estándar	0,43	0,00	0,00	0,82	1,47	0,71
Mínimo	3,00	4,00	4,00	3,00	1,00	3,00
Máximo	4,00	4,00	4,00	5,00	5,00	5,00

Tabla B.10. Resultados subjetivos

3.4.2 Preguntas formuladas

- ¿Has echado en falta alguna característica u opción?
 - *P1*: Pues por ejemplo no he visto cómo poder bloquear una tarea del sprint, ni cómo asignarlo a un sprint, creo que es el último recuadro, pero se supone que debería tener tooltip pero no hay y no sé lo que pone. Tampoco puedo ver las tareas que están hechas o las que están pendientes.
 - *P2*: Escalas diferenciadas para tareas e historias de usuario y Bugs/Hotfix en los gráficos.
 - *P3*: Integración con herramientas enfocadas al desarrollo.
 - *P4*: La movilidad de las pestañas de las tareas creo que se puede mejorar y el alta de usuarios, así como la libertad de meter más campos de forma libre al dar de alta una tarea.
- ¿Añadirías algún modelo de predicción que no estuviese presente en la aplicación?
 - *P1*: No.
 - *P2*: No, en todo caso los condensaría.
 - *P3*: No.
 - *P4*: Mas que añadir, es dejar un poco más claro y por número de tareas las que hay, las que se han hecho y las pendientes, por sprint y por total.
- ¿Usarías/usas en tu entorno de trabajo una herramienta similar para realizar predicciones y estimaciones?
 - *P1*: Actualmente estamos utilizando Jira, si se manejan bien los informes se pueden hacer predicciones, pero aparte claro.
 - *P2*: Sí.
 - *P3*: Sí.
 - *P4*: Me parece útil, pero si dejara poner más campos libres podría adaptarse más al pseudo agile que usamos algunos.

3.4.3 Análisis de los datos subjetivos

Partiendo de los resultados de la Tabla B.10, se pueden sintetizar los resultados de satisfacción de los usuarios, como se muestra en la Figura B.1.

Respecto a la satisfacción general, es claramente positiva, con un valor de **3,75** sobre 5 la valoración de los participantes.

En utilidad ha habido consenso en las valoraciones individuales, obteniéndose una media de **4,00**.

La facilidad de uso se ha valorado con un **4,00**, lo cual es bastante positivo siendo la primera vez que los usuarios interactuaban con la aplicación.

La apariencia visual ha tenido algunas diferencias entre calificaciones, pero la media se establece en **3,75**.

Respecto a la claridad que aporta la aplicación, se obtiene la peor calificación, con un **3,25** de media. Esto se debe a que los usuarios han tenido dificultades encontrando varias de las opciones en las interfaces. Sería importante mejorar este aspecto de cara a nuevos incrementos.

La fluidez de navegación ha sido valorada por todos los participantes con un **4,00**, y remarcada en comentarios como algo apreciado y agradable en su uso.

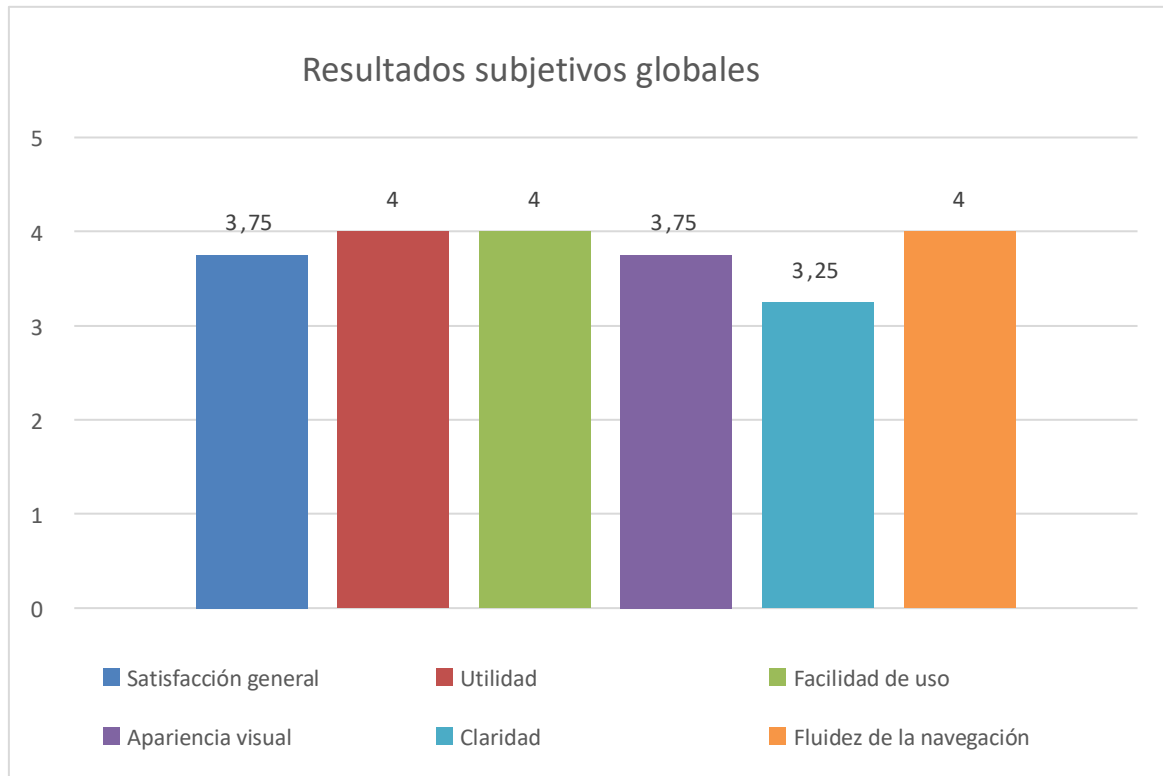


Figura B.1. Resultados subjetivos globales

Estos resultados aportan una visión global de los aspectos más visuales e interactivos de la aplicación, y marcan el camino a seguir para mejorar esta. Siendo los puntos más fuertes la utilidad encontrada por los participantes a las herramientas que se proporcionan, así como la fluidez con que se navega entre vistas.

Principalmente, los puntos a mejorar son el feedback y la claridad de los menús que utilizarán los usuarios, aspecto crucial sobre todo en su primer contacto con la aplicación.

Las funcionalidades que los participantes han sugerido en las preguntas indican que esperan de la aplicación que aporte flexibilidad al usuario. La aplicación debería permitir a cada organización gestionar su Product Backlog pudiendo personalizar los campos que se introduzcan en cada PBI a voluntad de cada usuario. También se marca como crucial la interoperabilidad con otras aplicaciones de gestión ágil, como Jira. Ya que, sin esta, el posible uso de la aplicación se vería severamente reducido. También se han esperado ciertas funcionalidades de alta importancia en el entorno empresarial, como es el bloqueo de PBIs cuando estos tienen dependencias con otros o la presencia de un Kanban genérico como el que incluyen la mayoría de las aplicaciones de gestión ágil.

Respecto a las predicciones no se han sugerido nuevos métodos, más allá de unificar los ya proporcionados en una sola vista.

3.5 Resultados globales

La aplicación tiene como puntos fuertes la utilidad aportada a los usuarios y su fluidez de uso, así como el sistema de predicciones que proporciona.

Las mejoras deberán centrarse en la comunicación con el usuario y facilidad de uso, así como la flexibilidad e interoperabilidad. No limitando al usuario ni restringiendo su forma de trabajo, sino facilitando sus tareas de gestión se podrá alcanzar un producto de alta calidad que no solo pueda ser utilizado en el entorno empresarial, sino que quiera ser utilizado.

ANEXO C. SUGERENCIAS DE USUARIOS

- Crear una escala diferente para los bugs separada del Product Burndown Chart para darle visibilidad.
- Mostrar cuantos puntos historia se queman por sprint además de los restantes ya mostrados.
- Arreglar un problema que surge al arrastrar PBIs mientras que se hace scroll en la pestaña de Backlog.
- Avisar de alguna forma al usuario de que se ha desactivado la opción de arrastrar PBIs al filtrarlos.
- Cambiar el icono de arrastrar PBIs.
- El filtro del Backlog no debería diferenciar entre mayúsculas y minúsculas.
- Agregar un buscador de PBIs a la opción de agregar dependencia dentro de los detalles de un PBI.
- Añadir dependencias Fs y FF a los PBIs, para que un PBI no pueda ser marcado como Done si existe alguna dependencia con otro PBI no finalizado.
- Dar la opción de crear campos adicionales del tipo que se elija dentro de un PBI, para flexibilizar la aplicación de cara a la inmensa diversidad de formas de trabajar que existen en metodologías ágiles.
- El sprint deadline N debería ser el sprint N+1 en los gráficos.
- Cambiar la numeración de los gráficos para que cada punto N del eje X represente el inicio del sprint N, y no el final.
- El primer sprint no debe llamarse Sprint 0.
- Sería muy útil desarrollar un plugin para integración con Jira.
- Poder marcar entregas de un proyecto y asignar PBIs a entregas, para que las predicciones filtren los PBIs y solo realicen predicciones dentro de la entrega que interesa.

ANEXO D. LISTADO DE PRODUCT BACKLOG ITEMS

Título de PBI	ID
Como Tribunal, quiero que se escriba en la memoria el capítulo del Estado del Arte para que la memoria introduzca y aporte un contexto al trabajo realizado	#5
Como PO_m, quiero un esquema inicial de la memoria del proyecto para poder aprobar la estructura de la misma	#6
Como PO_m, quiero que el equipo de desarrollo estudie a fondo la Guía Scrum para que entiendan mejor su funcionamiento	#7
Como PO_p, quiero inicializar un repositorio de GitHub para establecer un control de versiones al proyecto	#8
Como PO_p, quiero inicializar una herramienta de gestión de proyectos ágiles para controlar la actividad durante el proyecto	#9
Como PO_p, quiero inicializar el Product Backlog del proyecto para empezar a aplicar Scrum	#10
Crear un "hola mundo"	#11
Implementar CI en Travis	#12
Hacer un despliegue manual	#13
Implementar CD en Travis y Heroku	#14
Como PO quiero que existan 3 tipos de usuarios para poder gestionar los distintos permisos	#15
Como PO y Dev quiero gestionar mi cuenta para hacer modificaciones	#16
Como PO quiero listar y abrir proyectos en propiedad para poder trabajar en ellos	#17
Como PO quiero crear proyectos para poder iniciar el trabajo	#18
Como PO quiero crear PBIs para poder gestionar el trabajo a realizar	#19
Como PO quiero editar el valor de los PBIs para mostrar su importancia	#20
Como PO quiero comentar PBIs para aportar feedback	#21
Como PO quiero editar el orden de los PBIs para priorizarlos	#22
Como PO quiero editar etiquetas de PBIs para agruparlos por estas	#23
Como PO quiero editar criterios de aceptación de PBIs para controlar cuando aceptar los PBIs	#24
Como PO quiero que haya ordenación automática de PBIs para agilizar la priorización	#25
Como PO quiero editar dependencias de PBIs para poder controlar estas	#26
Como PO quiero poder validar como Done los PBIs para poder indicar cuando están realizados	#27
Como PO quiero ver y editar DoD para poder revisarlas y actualizarlas a voluntad	#28
Como PO quiero listar DoDs antiguas para controlar el cambio de estas	#29
Como PO quiero ver los PBIs que están Done para ver el trabajo finalizado	#30
Como PO quiero crear y ver PBCharts para visualizar el progreso	#31
Como PO quiero mostrar la evolución del alcance para poder gestionarla mejor	#32
Como PO quiero proyecciones regresión lineal y polinómica para poder hacer predicciones	#33
Como PO quiero calcular el throughput y la velocidad por sprint para poder hacer predicciones	#34
Como PO quiero crear usuarios de tipo Developer y Stakeholder para que puedan utilizar la herramienta	#35
Como Developer quiero editar el tamaño de los PBIs para poder estimarlos	#38
Como Developer quiero marcar como Done los PBIs para notificar que mi trabajo está hecho	#39

ANEXO D. Listado de Product Backlog Items

Como Stakeholder quiero ver proyecciones de proyectos para visualizar información de manera rápida	#41
Enlazar Travis con Slack para notificación constante de nuevas integraciones	#43
Recalcular peso de epica de memoria con los nuevos pbis creados	#44
Documentar Sprint 1	#45
Documentar Sprint 2	#46
Documentar Sprint 3	#47
Documentar Sprint 4	#48
Documentar Sprint 5	#49
Documentar Sprint 7	#50
Crear un prototipo de las pantallas de PBIs	#51
Como PO quiero filtrar PBIs por etiquetas o palabra clave	#52
Como PO quiero poder adjuntar ficheros a un PBI	#53
Documentar Sprint 8	#54
Arreglar pipeline de Travis-Heroku	#55
Documentar Sprint 9	#56
Escribir Capítulo 3 de la memoria	#57
Crear la opción de elegir número de sprint al marcar PBI como Done	#58
Documentar Sprint 10	#59
Como PO quiero visualizar en el PBC cuanto se ha quemado por label	#60
Como PO quiero ver e indicar el sprint actual del proyecto	#61
Como PO quiero visualizar el Scope Creep del proyecto	#62
Agregar opción de Deadline del proyecto y recolocar opciones de forecast by velocity	#63
Revisar la memoria para arreglar varios apartados	#64
Como PO quiero comparar pendientes de quemado entre distintos PBCharts	#66
Como Dev quiero poder diferenciar entre el informador de un nuevo PBI y el responsable de realizarla	#68
Mostrar la composición por labels en el gráfico de PoC	#69
Tokenizar ids de proyectos y crear token de sesión de usuarios	#70
Mejorar robustez del api mediante un mejor error handling y limpiar código	#71
Como PO quiero poder cambiar el sprint actual y escribir el sprint goal	#72
Pedir contraseña actual para cambiar contraseña del usuario	#74
Arreglar varios fallos y bugs encontrados	#75
Como PO quiero poder eliminar usuarios del proyecto	#76
Reajustar Product Backlog	#77
Cambiar campo de inicio de sesión a correo electrónico	#78
Guardar proyectos favoritos en la bbdd y comprobar que son correctos al iniciar sesión	#79
Como PO quiero poder escribir la Visión del proyecto	#80
Como PO quiero poder modificar la descripción del proyecto	#81
Desuscribir observables para reducir consumo de memoria	#82
Documentar métodos y limpiar código	#83
Transformar metodos de los controllers del backend a async	#84
Verificar que el usuario pertenece al proyecto en el api para poder modificar este	#85
Añadir text tips flotantes a opciones poco intuitivas de la UI	#87
Implementar pestaña de Home y About	#88
Arreglar lazyload y cambiar rutas a ingles	#89

ANEXO D. Listado de Product Backlog Items

Dar formato al email de invitacion de usuarios	#90
Documentar Sprint 12	#91
Arreglar UI y recolocar componentes detectados en la revisión	#92
Diseñar guión de user testing y empezar la estructura del CIF	#93
Escribir resultados en el Common Industry Format for Usability Test Reports	#94
Arreglar bug de fallo al mostrar scope creep	#95
Separar cada grafico en un componente aparte	#96
Arreglar feedback al guardar informacion en PBIs	#100
No aparece el boton de expulsar tras cambiar de proyecto	#101
Crear tests unitarios de servicios y modelos	#102
Los graficos de forecast deben calcular los pbis hasta el ultimo sprint del proyecto	#103
Documentar Sprint 13	#104
Como PO quiero poder establecer el Deadline de un proyecto	#105
Como PO quiero poder ver el deadline compartido de los proyectos en la comparación	#106
Error al subir ficheros a pbi sin estar creado aún	#107
Incorporar mejoras del user testing	#108
Agregar subcapítulo de análisis de resultados del user testing	#109
Agregar Anexos a la memoria, arreglar encabezados, bibliografía y referencias y escribir subapartado del repositorio	#110
Como PO quiero un gráfico que muestre los bugs abiertos actualmente y el histórico de bugs abiertos y cerrados	#111
Documentar Sprint 14	#112
Como Tribunal quiero que se escriba el capítulo de la Introducción	#114
Como Tribunal quiero que se escriba el capítulo de las Conclusiones	#115
Como Tribunal quiero que se escriba el capítulo de la Introducción	#116
Como Tribunal quiero que se incluyan el Burndown chart y comentarios sobre la diagonal y las predicciones	#117
Reajustar ejes en todos los gráficos y arreglar problema con ultimo sprint en forecasts	#119
Documentar Sprint 15	#120
Agregar línea de quemado ideal	#121
Escribir resumen, dedicatoria y agradecimientos	#122
Arreglar memoria revisada	#123

CONTENIDO DEL CD

En sustitución del CD adjunto al trabajo, se proporciona un enlace para acceder al repositorio del proyecto:

- <https://github.com/victorperezpiqueras/TheProductOwnerd>

En dicho repositorio se pueden encontrar los siguientes elementos:

- Memoria del Trabajo Fin de Grado en formato PDF.
- URL de la aplicación.
- Código fuente de la aplicación.
- Scripts de integración.
- Cuaderno de proyecciones de Jupyter.