Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Diseño y Desarrollo de Videojuegos

Proyecto Final

Herramienta de gestión y planificación de videojuegos

Autor del proyecto: Miguel Gracia Carballo

Director del proyecto: Jorge Echeverría Ochoa

Zaragoza, 9 de septiembre de 2021

universidad SANJORGE
GRUPO SANVALERO

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma:  Fecha: 09 de septiembre de 2021

Dedicatoria y Agradecimiento

En primer lugar, quiero dar las gracias a mis padres y a mi hermano por estar siempre apoyándome y animándome a conseguir mis metas y objetivos. Sin vuestro apoyo no hubiera sido capaz de ser quien soy y de llegar hasta aquí.

Quiero darles las gracias a mis compañeros, que durante todos estos años hemos estado trabajando juntos y todo lo que me habéis ayudado durante la carrera. Nunca olvidaré todos los buenos momentos que hemos pasado juntos.

No puedo olvidarme de todos los profesores, que han dado el cien por cien por enseñarnos durante años a ser mejores y todo lo que hemos aprendido de ellos. Muchas gracias por hacer de nosotros grandes profesionales.

Y por último quiero agradecer a Jorge Echeverría, por todo lo que me ha aportado estos años. Jorge fue la primera persona que me enseñó a programar hace seis años y ahora él me acompaña en mi último trabajo de la carrera.

Muchísimas gracias a todos.

Tabla de contenido

Tabla de ilustraciones

Tabla de tablas

Resumen

En las empresas y en las universidades, y no solo en el ámbito tecnológico, se está fomentando cada vez más el uso de metodologías agiles como método de trabajo. Esto se debe a la gran cantidad de beneficios que tiene implementar estas metodologías en los grupos de trabajo. Es por eso por lo que cada día hay más herramientas disponibles y con diferentes funcionalidades.

Como resultado del Trabajo de Fin de Grado de la carrera de Diseño y Desarrollo de Videojuegos se ha desarrollado una herramienta de para dar soporte a desarrollos basados en metodologías Agile. El objetivo de este proyecto es elaborar una herramienta de planificación y gestión de videojuegos que sea fácil de usar e intuitiva, sin perder las funcionalidades básicas necesarias de las metodologías agiles. Y que sea de utilidad tanto para estudiantes sin experiencia en este tipo de herramientas como para profesionales que usan este tipo de herramientas en su día a día.

Palabras clave

Metodologías agiles, Herramienta de planificación, *Sprint*, *Backend*, *Frontend*, Proyecto, Tareas.

Abstract

In companies and universities, and not only in the technological field, the use of agile methodologies as a working method is being increasingly promoted. This is due to the large number of benefits of implementing these methodologies in working groups. That is why every day there are more tools available and with different functionalities.

As a result of the Final Degree Project of the Videogame Design and Development career, a tool has been developed to support developments based on Agile methodologies. The objective of this project is to develop a video game planning and management tool that is easy to use and intuitive, without losing the basic functionalities required by agile methodologies. And being useful both for students without experience in these types of tools and for professionals who use these types of tools in their day to day.

Key words

Agile Methodologies, Planning Tool, Sprint, Backend, Frontend, Project, Tasks.

1. Introducción

Las metodologías agiles son uno de los recursos más empleados en los proyectos de las grandes compañías. Gracias a aplicar este tipo de metodologías, empresas como Spotify [1], Google o Amazon, han podido sacar adelante exitosamente sus productos. Originalmente surgieron para ayudar en proyectos de *software*, aunque hoy en día, las metodologías agiles siguen predominando en las empresas tecnológicas, cada vez se usan más en diferentes industrias, como en empresas financieras o hasta en los gobiernos.

Según la decimocuarta edición anual del informe State of Agile publicado en 2020 [2], que recoge los datos de 40000 participantes, el 95% de estos afirmaron que en sus empresas se utilizan metodologías agiles, aunque solo el 18% las usan en todos sus equipos. Algunos de los beneficios y ventajas de implantar estas metodologías en los equipos de trabajo son:

- Aumentan la calidad del producto
- Reducen el coste del proyecto
- Incrementan la flexibilidad frente a los cambios
- Mejoran la organización del proyecto
- Involucran al cliente en el proceso para satisfacer sus necesidades

Gracias a todas estas características cada vez más empresas están dispuestas a adoptar estas metodologías en sus equipos, y por lo tanto cada vez surgen más herramientas para poder llevar el seguimiento de los proyectos. Esto se puede observar en los cuadrantes de Gartner de las empresas que desarrollan herramientas de planificación agile. En los gráficos de 2017 y de 2021 que se pueden observar en la Ilustración 1, se puede ver las empresas que se han ido consolidando en el mercado, mientras que han surgido nuevas. Hay que destacar a Atlassian, como una de las empresas líder del sector según Gartner, y que, en el informe mencionado previamente, Jira, una de sus herramientas, aparece como la más utilizada entre los encuestados.

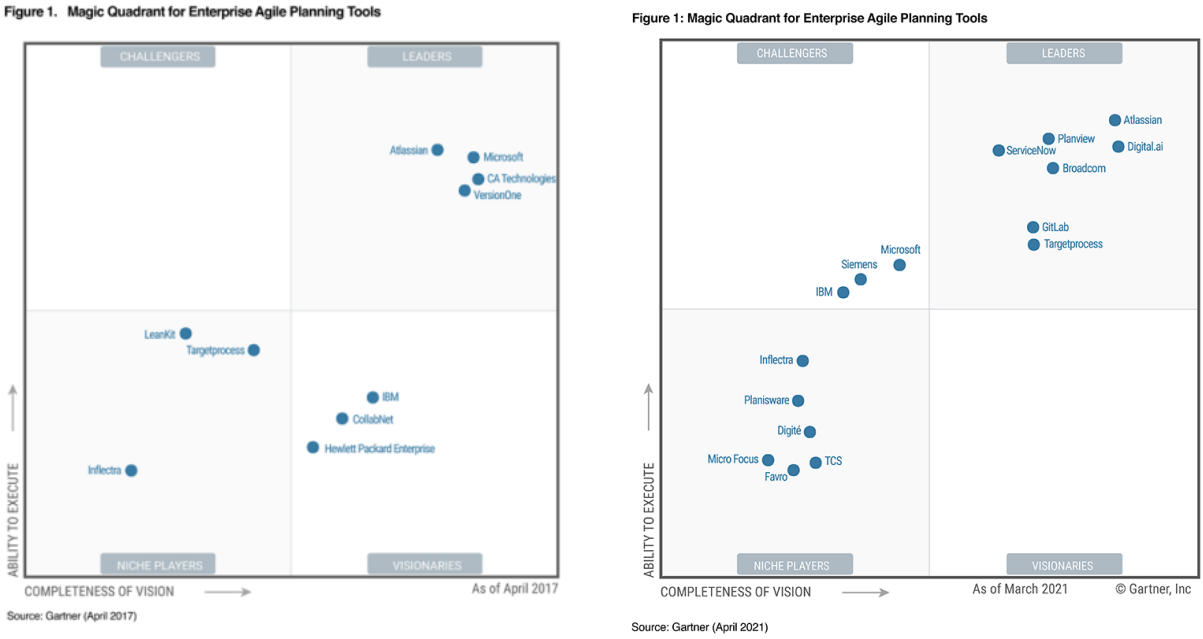

Ilustración 1 - Cuadrantes Gartner de 2017 y 2021 de las empresas de herramientas de planificación Agile. Source: Gartner

Con este aumento del número de soluciones que hay en el mercado, también han aumentado las funcionalidades y opciones que aportan estas herramientas. Ha llegado a un punto en el que se pueden encontrar plataformas especializadas tanto para los diferentes tipos de metodologías *Agile*, como para sus *frameworks*, por ejemplo, *Scrum*. Por otro lado, también hay herramientas de planificación que son capaces de aglutinar todas las funcionalidades posibles. Esto permite elegir entre una gran variedad de herramientas y utilizar las características que más convengan dependiendo del proyecto que se va a desarrollar.

Pero no todo es bueno, y es que también hace que las herramientas sean más complicadas de usar para los usuarios más nuevos en este tipo de herramientas. Además, al tener tantas funciones dentro de ellas se desperdician muchas de ellas porque no se llegan a utilizar. Por todo esto, el propósito de este Trabajo de Fin de Grado es desarrollar una herramienta que sea capaz de realizar las funciones más imprescindibles para la planificación y gestión de un proyecto. Y que, además, sea sencilla y útil para cualquier persona que no haya experimentado con este tipo de herramientas.

Para cumplir con estos objetivos, durante el desarrollo de esta herramienta, se realizó una encuesta para orientar el producto final a las necesidades de los usuarios. Esta encuesta ha sido muy importante, ya que ha permitido definir cuales son las funcionalidades que la herramienta

debe tener y cuales no. Más adelante, también se hicieron pruebas de usabilidad que permitieron mejorar aspectos de la herramienta para enriquecer la experiencia de los usuarios e incluir funcionalidades nuevas.

Al final del desarrollo de este *software*, se consiguió una herramienta capaz de organizar y planificar videojuegos y otro tipo de proyectos, a través del uso de metodologías *Agile*. La herramienta dispone de *Backlog* y de *Sprints* para la organización de las tareas, así como un tablero *Kanban*, y también un apartado de estadísticas que muestran el progreso realizado en cada proyecto.

Este Trabajo de Fin de Grado representa el trabajo y esfuerzo realizado estos meses, y de las capacidades técnicas y conocimientos adquiridos durante estos años en la universidad. Es una muestra de como la formación recibida nos ha preparado para solventar problemas y diseñar soluciones a problemas, así como ha adaptarnos a nuevas tecnologías.

2. Estado del arte

2.1. Introducción

Hoy en día trabajar en equipo es una práctica muy habitual y que se fomenta tanto en ambientes educativos como en el entorno de la empresa. Esta forma de trabajar ha servido como ayuda y ha sido determinante para el desarrollo de múltiples proyectos e ideas. A raíz de esto han surgido diferentes metodologías de trabajo, como las metodologías *Agile* y los *frameworks* y herramientas que se han desarrollado para poder aplicarlas.

Estas metodologías ayudan a gestionar el trabajo del equipo y llevan un seguimiento del trabajo realizado. Por eso el proceso de planificar y gestionar un proyecto es una de las fases más importantes dentro de la producción de este, ya que incluye los aspectos más relevantes. Se debe tener en cuenta las tareas necesarias para llevar a cabo el trabajo y su organización, una estimación de cuanto va a costar hacer cada una de las tareas que se han asignado, así como la metodología de trabajo que se va a emplear. Además, permite extraer estadísticas para optimizar los resultados y conseguir estimaciones más exactas para futuros proyectos basadas en los resultados obtenidos.

2.2. Herramientas existentes

Actualmente se dispone de múltiples herramientas para el desarrollo de *software* tanto para proyectos de estudiantes, como para proyectos de ámbito profesional. Alguna de las herramientas más conocidas son Trello, Jira, Asana y Hacknplan, siendo esta última una herramienta especializada para la planificación de videojuegos.

2.2.1. Trello

Trello es una herramienta de gestión de tareas desarrollado por Atalassian, una empresa australiana que se dedica al desarrollo de *software* para empresas, desde que lo adquirió en 2017 [3]. Trello permite la colaboración entre personas de un equipo para organizar el proyecto por grupos de tareas. Estas se rellenan en tarjetas en las que se puede añadir toda la información relativa a la tarea como el nombre, una descripción, el responsable de esta, entre otros. Cada una de estas tareas se muestran en un tablero *Kanban* como se puede ver en la Ilustración 2.

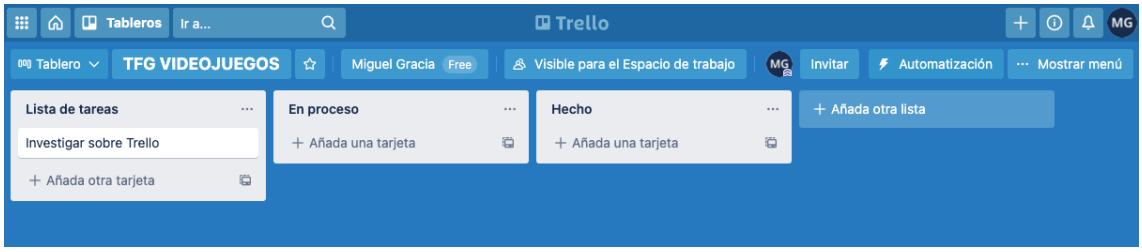

Ilustración 2 - Interfaz principal de un tablero de Trello

Trello es una herramienta gratuita, pero se puede ampliar el plan para obtener más funcionalidades pagando mensual o anualmente. Las dos versiones de pago son Trello Standard [4] y Trello Premiun [5], ambas opciones te dan más prestaciones como Power-Ups, mejoras dentro de cada proyecto para añadir más funciones.

2.2.2. Jira

Jira es una herramienta de gestión y planificación de proyectos, que como Trello pertenece a Atalassian. Esta aplicación nació con la idea de ser una aplicación para gestionar incidencias, pero con el tiempo se convirtió en una herramienta para manejar cualquier tipo de proyecto [6]. Actualmente es una de las herramientas más completas que existen con numerosas funcionalidades, tales como: tableros *Scrum,* informes y analíticas del proyecto, tablero *Kanban.* Además, cuenta con la integración con otras herramientas de Atalassian como Bitbucket, un repositorio de código. En la Ilustración 3 se puede ver como es la interfaz del *Backlog* de Jira.

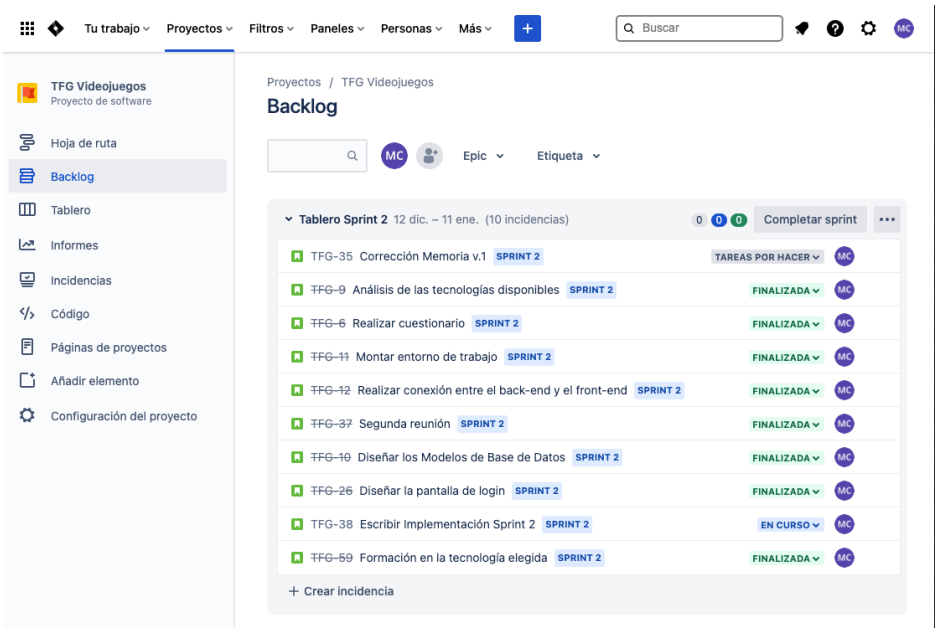

Ilustración 3 - Interfaz principal del Backlog de un proyecto en Jira

2.2.3. Asana

Asana un producto que se dedica a la planificación de proyectos, cuyo objetivo es conseguir cumplir con las fechas establecidas para las entregas o fechas límites para cumplir con las metas del proyecto, como se puede observar en la Ilustración 4. Asana dispone de diferentes plantillas dependiendo de las necesidades que necesite el equipo y el caso de uso del proyecto, algunos ejemplos son finanzas, recursos humanos o ingeniería entre muchos otros [7].

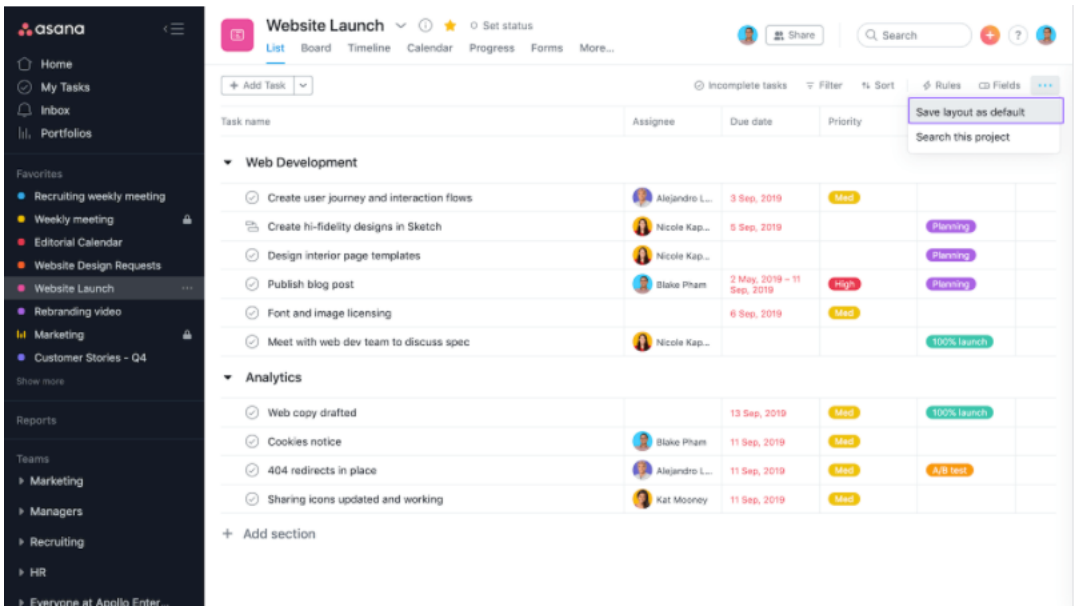

Ilustración 4 - Interfaz de Asana

2.2.4. Hacknplan

Es una herramienta de producción de *software* especializada en proyectos de videojuegos, fue desarrollada por Chris Estevez, un programador español, en 2015 [8]. La gran diferencia entre Hacknplan y las otras herramientas vistas, es su orientación a los videojuegos, incluyendo aspectos básicos de estos como, GDD o tener las categorías ya predefinidas como: programación, arte, sonido, etcétera [9]. Esto permite a los desarrolladores tener toda la información más clasificada y ordenada, además de poder integrar Hacknplan con otras aplicaciones como GitHub o Slack. La Ilustración 5 muestra la interfaz de un tablero *Kanban* de tareas con algunas de las categorías mencionadas previamente.

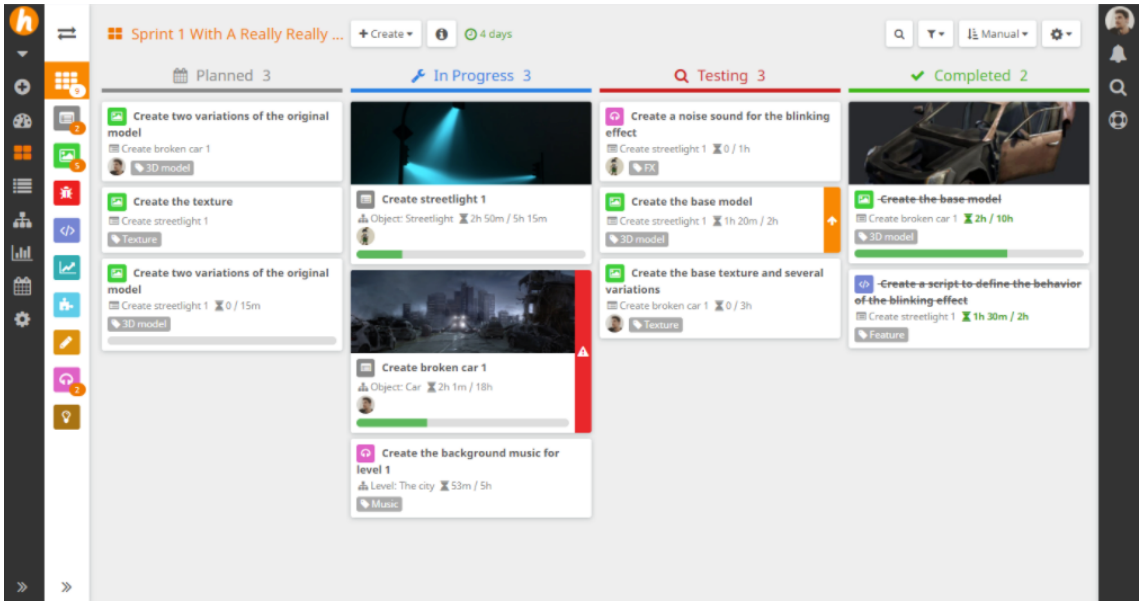

Ilustración 5 - Interfaz del tablero Kanban de Hacknplan

2.3. Tabla comparativa

En la siguiente tabla se pueden observar diferentes características y funcionalidades que tienen las plataformas analizadas y la herramienta que se ha creado para este Trabajo de Fin de Grado (TFG).

	Jira	**Trello**	**Asana**	**Hacknplan**	**TFG**
Totalmente gratuito	No	No	No	No	Si
Sencilla	No	Si	No	Si	Si
Especializada en videojuegos	No	No	No	Si	Si
Límite de colaboradores	10	No	15	No	No
Organización por Sprints	Si	No	Si	No	Si
Gestión del tiempo	Si	No	Si	No	Si
Backlog	Si	No	Si	Si	Si
Tablero Kanban	Si	Si	Si	Si	Si

Diagrama Gantt	Si	No	Si	Si	No
Estadísticas y reportes	Si	No	Si	Si	Si

Tabla 1 - Comparativa de las diferentes funcionalidades

3. Objetivos

Durante la realización de este proyecto no se han modificado los objetivos presentados en la propuesta del Trabajo Fin de Grado (TFG). En este capítulo de la memoria se van a desarrollar estos objetivos de forma más especifica y detallada. Los objetivos son:

- Estudio de las herramientas de gestión y planificación de videojuegos, ya existentes en el mercado y sus funcionalidades.

  Este objetivo está pensado en estudiar que herramientas se pueden emplear para llevar la gestión de un desarrollo de un videojuego. De este punto, se puede extraer información sobre las funciones que tienen las herramientas existentes. Esto sirve para comparar las herramientas que ya existen con el proyecto que se va a crear, además de buscar cualidades/funcionalidades que no existan para desarrollarlas en el *software*.

- Estudio de las necesidades y funcionalidades requeridas por los usuarios y su implementación en el *software*, además de las metodologías empleadas para el desarrollo de videojuegos.

  En relación con el objetivo anterior, este punto trata de especializarse en las herramientas que desea disponer un desarrollador. Para ello es necesario preguntar a posibles futuros usuarios de una herramienta de este estilo, sobre que necesidades o funciones tendría que cumplir para que ellos la usaran en sus proyectos.

- Creación de una herramienta *software* mediante tecnologías que están por determinar (aplicación web, aplicación móvil, etc).

  El último objetivo tiene como meta el desarrollo de la herramienta teniendo en cuenta todos los factores que se han estudiado en los dos objetivos anteriores. Además, en este objetivo se tienen en cuenta aspectos como la experiencia de los usuarios al usarla para corregir aspectos de esta.

4. Metodología

La metodología que se va a seguir en el proyecto se decidió tras la primera reunión con el tutor, y la idea principal es realizar una adaptación de *Scrum*. Se ha considerado que para este proyecto es ideal utilizar este *framework* en vez de seguir otras metodologías *Agile* u otros *frameworks* de trabajo, esto se debe a que el proyecto es muy escalable y se pueden usar los *Sprints* para cumplir con los objetivos e ir añadiendo más funcionalidades en la herramienta, de forma que el proyecto crecerá poco a poco en cada iteración.

4.1. Scrum

Scrum es un *framework* de trabajo que se usa para seguir metodologías *Agile* que tiene como objetivo fomentar el trabajo en equipo para maximizar la eficiencia del grupo y que se optimice el resultado final del producto que se va a desarrollar o del problema que se quiere resolver.

En este *framework* se pueden distinguir diferentes tipos de roles que se tienen que asignar para cada proyecto, en *Scrum* el equipo de trabajo tiene que estar formado por:

- *Product Owner*, es la persona responsable de que el resultado final sea lo más optimo, su objetivo es que el producto desarrollado obtenga el máximo valor posible. Sus principales funciones dentro de *Scrum* son: definir los objetivos del *Sprint*, crear y manejar las tareas en el *Product Backlog*, una lista con las tareas, ordenadas por prioridad, que se han de realizar en el Sprint, para poder alcanzar las metas propuestas en cada Sprint. Y también tiene que resolver las dudas que puedan surgir en las tareas planteadas.

- *Scrum Master*, es la persona encargada de dirigir la organización del equipo y de asegurar que se está cumpliendo con el seguimiento del *framework*, debe organizar los *daily meetings*, reuniones de unos quince minutos donde el equipo de trabajo se junta para comentar en que van a estar trabajando y si existe algún impedimento para continuar con las tareas, en caso de que exista algún problema el *Scrum Master* se debe encargar de solucionar esos bloqueos.

- *Developer Team,* es el grupo de desarrolladores que se va a encargar de realizar las tareas que existan en el *Product Backlog,* deben de cumplir con el objetivo del *Sprint* y adaptar su trabajo diario para alcanzar esa meta. Además, deben de comunicar, en los

daily meeting, al equipo si tienen algún impedimento y de que tareas se van a encargar durante ese día.

Además de roles, *Scrum* tiene establecido un conjunto de eventos para realizar cada iteración que va a tener el proyecto, estas etapas están formadas por:

- *Sprint Planning*, una reunión al inicio de cada *Sprint* entre el *Product Owner* y el *Scrum Master* dónde se establece el objetivo del *Sprint*, la planificación que éste va a tener y su duración, así como las tareas imprescindibles que se han de realizar.

- *Sprint*, es el periodo de tiempo, que suele ser de entre dos y cuatro semanas, donde el equipo de trabajo tiene que realizar las tareas y completar el objetivo, en este periodo no se puede cambiar la meta del *Sprint*. En este tiempo está por norma que no puede bajar la calidad del proyecto a desarrollar, ya que en cada *Sprint* se ha de mejorar y aumentar el producto. Además, y como se ha mencionado antes, cada día del *Sprint* se tiene una reunión para ver el estado actual del *Sprint* y las necesidades de cada uno de los integrantes del equipo.

- *Sprint Review*, es una reunión en la que el *Scrum Master* presenta al *Product Owner* el trabajo realizado y se revisan si se han cumplido los objetivos marcados al principio del *Sprint*. Además, se revisan las tareas que no se han podido acometer y los motivos por los que no se han podido realizar, si se ha dado el caso.

- *Sprint Retrospective*, una reunión que sucede al acabar el *Sprint* donde el equipo se reúne y se ven los resultados obtenidos y se identifican algunas mejoras para el próximo *Sprint*. También se analizan los problemas individuales que han surgido para buscar una solución y para que el equipo pueda reaccionar a tiempo en el próximo *Sprint*.

4.2. La implementación del framework

Como se ha mencionado al principio de este apartado de la memoria, se va a realizar una adaptación de *Scrum* dónde se van a seguir los aspectos fundamentales de este *framework*. Pero ciertas partes se van a modificar, principalmente porque el equipo de trabajo está formado por una única persona y no hay un cliente que actúe como *Product Owner*, por eso los roles se repartirán entre el alumno y el tutor.

En la ejecución del proyecto se han repartido los roles y las funciones de estos de la siguiente manera:

- Equipo de Desarrollo, las funciones de este rol dentro de *Scrum* las realizará el autor del TFG, de forma que será el único desarrollador del proyecto. Se encargará de programar el *software*, así como de realizar la investigación sobre el producto que se está desarrollando y de realizar las labores de *testing*.

- *Scrum Master*, en este caso el tutor será el encargado de realizar las funciones de está figura de *Scrum*. Las labores que realizará son las de revisar que el proyecto va avanzando cada *Sprint*, se encargará de resolver dudas que puedan surgir y revisará la memoria.

- *Product Owner*, la responsabilidad de este último rol será compartida. El autor del TFG se encargará de gestionar la herramienta que se usará para planificar el proyecto, en este caso Jira, y también de definir las tareas que se han de realizar. Por el otro lado, el tutor se encargará de definir la meta del *Sprint*, así como de las tareas que hay que realizar para la próxima reunión.

En nuestra implementación se eliminaron las reuniones diarias y se unificaron las reuniones de *Sprint Planning*, *Sprint Review* y *Sprint Retrospective*, en una reunión por *Sprint*. Esta reunión estaba planificada para tener una duración de entre treinta minutos y una hora.

En estas reuniones, a excepción de la primera donde solo se planificó el primer *Sprint*, se empezaba haciendo una revisión del trabajo realizado. Después de comprobar si se habían cumplido los objetivos del *Sprint*, se seguía con la *Retrospective* y se comentaban los aspectos a mejorar y los puntos fuertes. Una vez finalizado estas dos partes y dando el *Sprint* por finalizado, las reuniones trataban sobre el siguiente. Se marcaban los objetivos y la hoja de ruta del nuevo *Sprint*, además se asignaban las tareas que se tenían que acometer y se creaban nuevas en el *Backlog* en el caso de que no estuvieran contempladas. Estos *Sprints* se diseñaron para ser iteraciones de tres o cuatro semanas. La información sobre estas reuniones queda reflejada en el Anexo 10.2, Acta de reuniones.

4.3. Planificación inicial

Uno de los objetivos de la primera reunión fue hacer una planificación con las tareas imprescindibles que se han de realizar durante el desarrollo del proyecto, así como una estimación de cuanto podría costar realizar cada una de esas tareas. Esta planificación correspondía a los requisitos mínimos que debían de cumplir para completar el esqueleto del proyecto. Por otro lado, algunas de las funcionalidades no se tuvieron en cuenta en este análisis debido a que no estuvieron planteadas hasta que no se avanzó en el proyecto.

Para poder registrar todas las tareas y llevar un seguimiento de trabajo se utilizó Jira, una herramienta de planificación de proyectos. En donde las tareas se fueron añadiendo al *Backlog*, y en cada *Sprint Planning* se movían a la tabla de tareas del *Sprint* que se iba a realizar, como se muestra en las Ilustraciones 6 y 7. Se eligió esta herramienta por su estructura de organización de tareas por *Sprints*, que facilitaba la creación de tareas dentro de estos, así como una estimación de cada una de ellas.

Ilustración 6 - Backlog del TFG en Jira

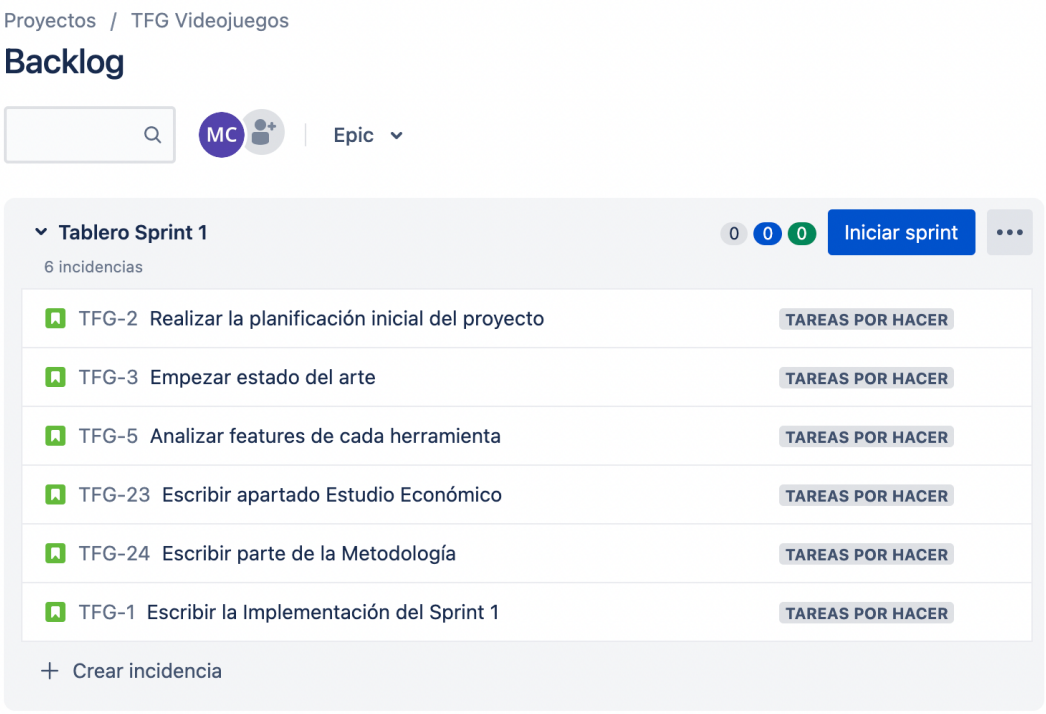

Ilustración 7 - Tablero con las tareas del Sprint 1

Además de tener una lista con las tareas, Jira dispone de un tablero *Kanban* para ver el estado en la que se encuentran las tareas del *Sprint*, como en el ejemplo de la Ilustración 8. Están divididas en tres grupos: por hacer, en progreso y, hecho. Aunque la herramienta te permite crear nuevos grupos para adaptarse a la necesidad del proyecto.

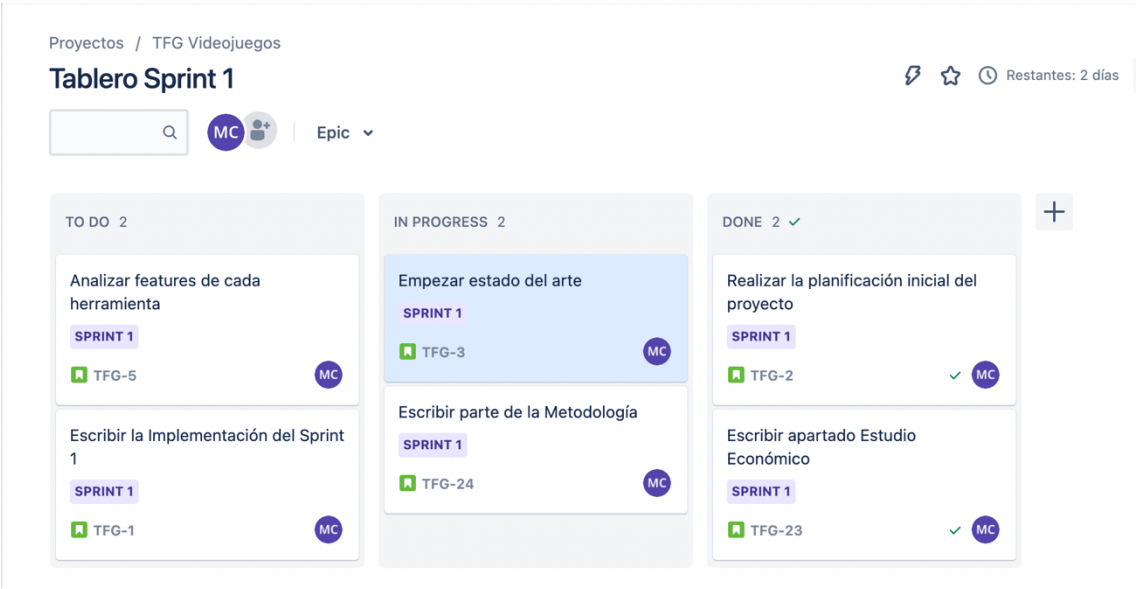

Ilustración 8 - Tablero Kanban correspondiente al primer Sprint

Cada tarea tiene diferentes campos con información que se puede añadir a estas, uno de estos es el del seguimiento del tiempo. Este apartado de la tarea permite llevar un control de cuanto se ha trabajado en cada tarea, además de indicar el tiempo que se había estimado.

La estimación inicial de las tareas que se consideraron fundamentales fue unas 274 horas, que se repartían en los siguientes apartados y en el gráfico de la Ilustración 9:

- Desarrollo del *software*: 155 horas y 30 minutos
- *Testing* y *User Experience*: 30 horas y 30 minutos
- Memoria: 59 horas y 30 minutos
- Reuniones: 4 horas y 30 minutos
- Formación: 24 horas

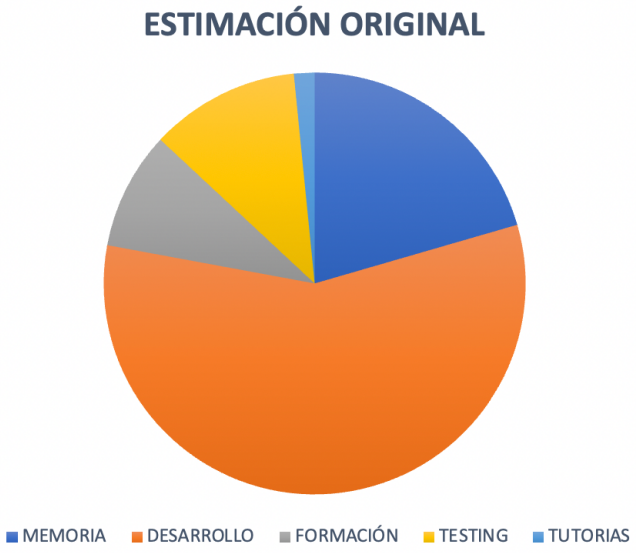

Ilustración 9 - Gráfico con la estimación original

Está planificación inicial se dividía en seis *Sprints* con una duración de cuatro semanas cada uno, donde el proyecto se iniciaba el 11 de noviembre de 2020 y se esperaba finalizar el 16 de mayo de 2021. Esta planificación se puede ver en el siguiente diagrama de Gantt, Ilustración 10, donde la carga de trabajo en horas entre los *Sprints* se dividía de la siguiente manera:

- *Sprint* 1: 21 horas
- *Sprint* 2: 72 horas
- *Sprint* 3: 36 horas
- *Sprint* 4: 49 horas
- *Sprint* 5: 49 horas
- *Sprint* 6: 47 horas

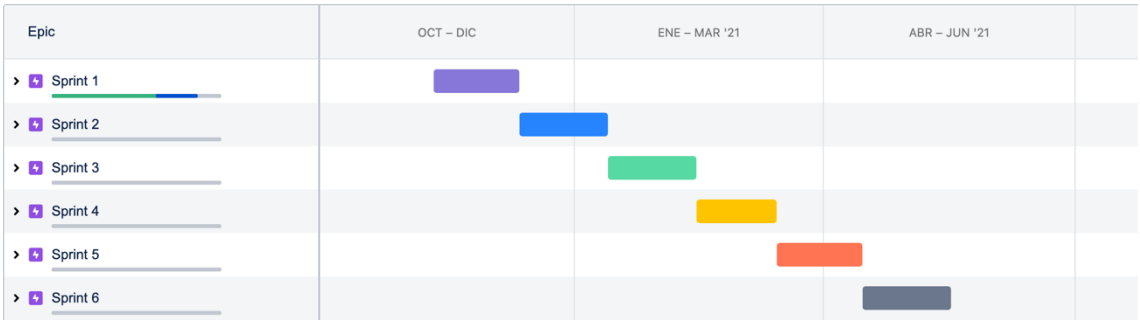

Ilustración 10 - Diagrama de Gantt de la planificación original

5. Implementación

En este capítulo de la memoria se desarrollará el proceso que se ha seguido en cada uno de los *Sprints*. En cada uno de ellos se mostrará la planificación inicial que se realizó al inicio, además se comparará la planificación con los resultados del *Sprint*.

5.1. Sprint 1

5.1.1. Planificación Inicial

En un principio este *Sprint* se planificó para empezar el 11 de noviembre, unos días después de la primera reunión con el tutor, y que acabase el 11 de diciembre. Para realizar todas las tareas se estimó que se necesitarían unas veintiuna horas. Durante este tiempo, el objetivo del *Sprint* era avanzar en la memoria y realizar una planificación inicial del proyecto.

5.1.2. Desarrollo del Sprint

Al principio del *Sprint* se realizó la estimación y la planificación inicial de las tareas que se debían hacer, para ello se usó Jira. Se eligió esta plataforma porque es un *software* que, de forma gratuita, incluye una gran cantidad de funcionalidades con respecto a otras herramientas. Además, es una herramienta con la que se estaba familiarizada de forma que no se necesitó aprender como usar las funciones más básicas. Primero se añadieron las tareas que se debían hacer para el primer *Sprint*, y el resto se fueron añadiendo al *Backlog*. Posteriormente se analizaron el resto de las tareas para ver en qué *Sprint* se podían realizar, y una vez se decidió se añadieron estas tareas a los tableros de los *Sprint* correspondientes.

Con toda la planificación hecha, se empezó con la memoria en concreto con los capítulos del Estado del Arte, Objetivos, Metodología y Estudio Económico. En el primer apartado se realizó una introducción a la actualidad de las herramientas de planificación y gestión de proyectos. Además, se presentaron algunas de las plataformas que existen, así como una comparación de sus funcionalidades. En el apartado de objetivos se expusieron los puntos presentados en la propuesta junto con una explicación más detallada de estos. El capítulo de la Metodología consistió en desarrollar *Scrum* junto con la planificación del proyecto y como se va a organizar este. Por último, se calcularon los costes de realizar este proyecto teniendo en cuenta las horas estimadas, los gastos materiales y el salario del desarrollador. Este presupuesto se incluyó en el apartado del Estudio Económico de la memoria.

5.1.3. Tareas Realizadas

En el transcurso de este Sprint se realizaron las siguientes tareas:

- Empezar estado del arte
- Escribir implementación del *Sprint* 1
- Analizar *features* de cada herramienta
- Escribir parte de la Metodología
- Realizar planificación inicial del proyecto
- Escribir objetivos del proyecto
- Escribir apartado del Estudio Económico

5.2. Sprint 2

5.2.1. Planificación Inicial

Los objetivos de este *Sprint* consistían en dos fases, la primera, analizar las diferentes tecnologías que se podían aplicar para desarrollar esta herramienta y empezar la formación necesaria para poder empezar a construir la base del proyecto. Una vez completado este primer proceso, los siguientes objetivos de este *Sprint* era elaborar la encuesta. Con la información obtenida se permitiría conocer las impresiones y necesidades que tiene la gente sobre las herramientas de planificación y gestión de *software*. Gracias a ello, se podría empezar a definir las características y funciones del proyecto.

Originalmente se planificó el completar estos objetivos en un periodo de tiempo de un mes, empezando el 12 de diciembre y finalizando el 12 de enero. Con una estimación de setenta y dos horas de trabajo.

5.2.2. Desarrollo del Sprint

En este *Sprint* se definió que tecnología se va a emplear para el desarrollo de este TFG, en este caso una aplicación MEAN, el acrónimo de Mongo-Express-Angular-Node.js. Para elegir está tecnología se tuvieron en cuenta todos los aspectos posibles, desde que tipo de base de datos usar, hasta que tecnologías emplear para *backend* y *frontend* por separado o una tecnología que los agrupe.

Lo primero que se tuvo en cuenta son las necesidades del proyecto, ya que existen diferentes soluciones, se buscó la que se adaptase mejor a las necesidades del proyecto. Al tener una idea tan escalable y que permite estar añadiendo funcionalidades, se prefirió utilizar un *framework*

basado en JavaScript. Ya que tienen una comunidad muy grande con la que se disponen de una enorme cantidad de preguntas y dudas resueltas, así como de múltiples librerías que pueden ser de gran utilidad para el desarrollo de esta herramienta.

Llegados a este punto se tuvo que elegir entre React y Angular, siendo esta última la elegida como se ha comentado antes. A pesar de que la curva de aprendizaje de React es menor que en el caso de Angular, se prefirió esta opción porque Angular es un *framework* de desarrollo de aplicaciones web más completo por defecto, mientras que React es una librería para el desarrollo de la interfaz de usuario que depende de otras librerías para desarrollar un proyecto como en Angular.

Para el *backend* se barajaron dos posibilidades, usar Node.js y Express o usar Java con Spring Boot. Esta decisión fue mucho más sencilla, puesto que Node.js y Express se complementan muy bien con *frameworks* como Angular, además de que tiene mejor rendimiento en tiempo de compilación. Aún así, se planteo la idea de usar Java debido a que se puede usar *Multi-Threads*, cosa que en Node.js no se puede.

Por último, se tenia que decidir en que tipo de base de datos se iba a guardar la información. En un principio la idea era utilizar una base de datos relacional, en especifico PostgreSQL. Pero cuando se empezó a diseñar como podría ser la base de datos, la cantidad de tablas que se tenían que crear era bastante grande debido a la normalización de estas. Esto suponía que, para realizar acciones básicas dentro del proyecto, como crear una tarea, había que realizar bastantes consultas anidadas. Analizando este proceso, se vio que se podía simplificar bastante empleando una base de datos no relacional como MongoDB, que son capaces de almacenar toda la información relativa a un proyecto en un mismo documento. Además, otra ventaja que se consiguió con este cambio es que MongoDB devuelve la información en formato JSON, por lo que ya se tiene estructurada para trabajar con ella.

Una vez se definió las tecnologías se necesito realizar una formación en ella, para poder completar esta formación se realizaron unos cursos de Udemy. En concreto fueron dos cursos: Angular & NodeJS – The MEAN Stack Guide [10] y, Angular – The Complete Guide [11]. Entre estos dos cursos se interiorizaron los conceptos más básicos de Angular y Node.js, además de introducir aspectos más complejos de estas tecnologías.

En este *Sprint*, al contrario que el anterior, no se cumplió con los objetivos marcados en la planificación. Esto fue debido a que el *Sprint* se planificó con bastante carga de trabajo teniendo en cuenta los días festivos de navidad, pero no se aprovecharon esos días.

Por ese motivo se añadió un *Sprint* entre el segundo y el tercero, según lo planificado originalmente, para todas esas tareas que no se alcanzaron a hacer. Además, las tareas que se realizaron se estimaron en veintiocho horas, cuando se necesitaron treinta y seis para completarlas.

5.2.3. Tareas Realizadas

En este *Sprint* se llegaron a hacer las siguientes tareas:

- Análisis de las tecnologías disponibles
- Formación en la tecnología elegida

5.3. Sprint 3

5.3.1. Planificación Inicial

Este *Sprint* se llevo a cabo entre los días 1 de abril y 30 de abril, con el objetivo de desarrollar las tareas que quedaron pendientes en el *Sprint* anterior. Estos objetivos consistían en la segunda parte que se había organizado para el *Sprint* anterior, que están mencionados en la subsección 5.3.2 de este capítulo, además de diseñar como serian las interfaces de la aplicación.

Además, a raíz del cambio con respecto a la planificación, se desajustó el tiempo previsto para la realización del TFG por lo que cambió el calendario y duración de los *Sprints*, así como se reorganizaron algunas de las tareas que se habían estimado para los próximos *Sprints*.

5.3.2. Desarrollo del Sprint

En este *Sprint* había dos objetivos fundamentales para el desarrollo de este TFG, el primero montar la estructura del proyecto y hacer funcionar la aplicación. Y el segundo, realizar una encuesta para obtener la información necesaria para definir que funcionalidades desarrollar.

Como se ha comentado en la sección anterior, el objetivo es programar una aplicación MEAN, para ello se empezó creando dos proyectos independientes. Uno de ellos se encargará de ser el *backend* de la aplicación, que estará compuesto por Node.js, Express y Mongo. Por el otro lado,

tendremos otro proyecto que hará las funciones del cliente, el *frontend* de la aplicación, que será en Angular.

Los primeros pasos consistían en levantar un servidor web con Node.js y Express, que fuese capaz de conectarse con la base de datos de Mongo. Para ello se estableció la siguiente estructura:

- *Models*: estos son los modelos de las colecciones que se van a necesitar en la base de datos, en este caso van a ser dos, *users* y *projects*. Para definir estos modelos se utilizó mongoose [12], una librería de Node.js que permite definir los objetos que se van a guardar en la base de datos, a través de *schemas*.

- *Controllers*: son las funciones que se van a encargar de hacer las *queries* en la base de datos y de obtener, modificar o borrar la información de esta.

- *Routes*: establecen las direcciones y los métodos que se van a aplicar en cada una de estas. Llamaran a los métodos necesarios de los *controllers*, así como a los *middlewares* en el caso de ser necesarios.

Toda la estructura está controlada en el fichero server.js que se encarga de realizar la conexión con la base de datos y desplegar el servidor que escuchará a un puerto en concreto en local.

Como se acaba de comentar, el sistema tendrá dos colecciones donde se almacenarán los datos del sistema. Una guardará la información relativa a los proyectos, y otra que registrará los datos de los usuarios registrados. En el caso de los proyectos, se han creado más modelos para representar alguno de los campos del objeto, como puede ser el *Backlog* o un *Sprint*, aunque estos no tengan su propia colección, si no que forman parte del mismo objeto. De este modo, se diseñó la estructura de la base de datos como se puede ver en la siguiente imagen.

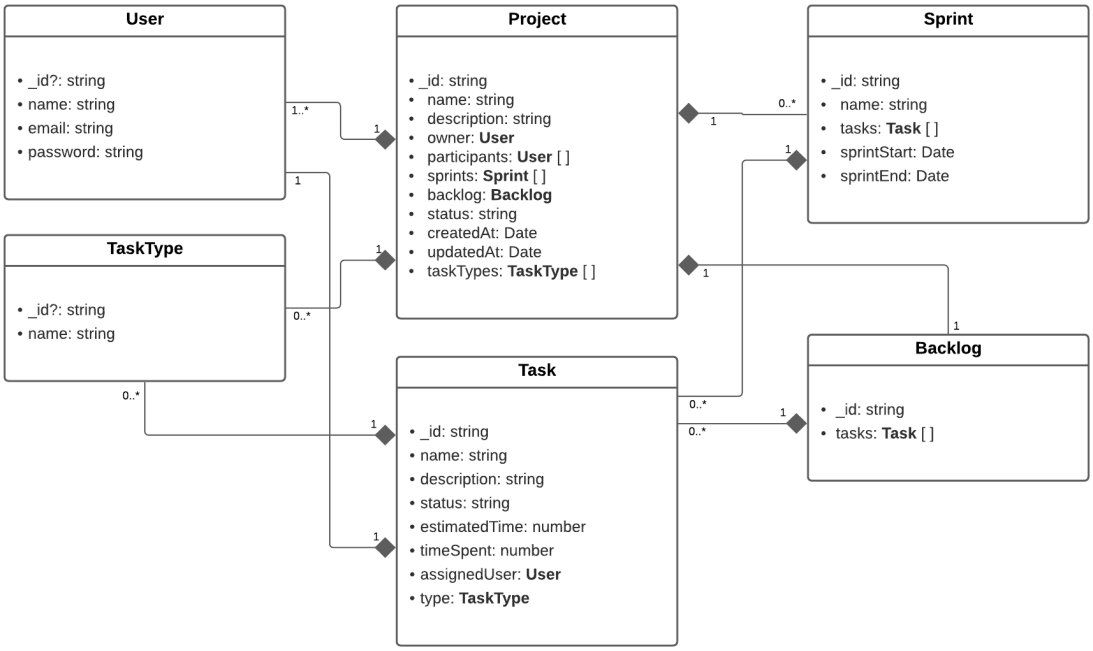

Ilustración 11 - Representación inicial de los modelos de la base de datos

Para poder comprobar el correcto funcionamiento del sistema se utilizó Postman, una herramienta que permite simular el funcionamiento de las peticiones que recibe el servidor. Con este programa se pudieron simular las operaciones CRUD necesarias para interactuar con nuestra base de datos, como pude ser el crear un proyecto o modificarlo. Para agilizar el proceso, no se utilizaron todos los datos del objeto, por ejemplo, solo se empezó trabajando con proyectos con nombre y descripción, en vez de con todos los campos.

Una vez completada la parte del servidor, se empezó a trabajar con el *frontend* de la aplicación, por lo que se creó un proyecto Angular. Este contendrá todos los componentes web de la aplicación y los servicios que se comunicarán con el backend de la aplicación, que serán peticiones HTTP. Además, también se importaron las librerías de Bootstrap [13], jQuery [14] y Popper.js [15], que ayudarán con los estilos y diseños de la web.

Antes de empezar a maquetar la web y montar los componentes, se hicieron unos diseños de como sería la herramienta. Se diseñaron las pantallas de *login* (Ilustración 12) y registro de usuarios (Ilustración 13), la pantalla de selección de proyectos del usuario (Ilustración 14), y las vistas principales de los componentes de la aplicación (Ilustraciones 15, 16, 17, 18 y 19).

Ilustración 12 - Diseño de la página de login

Ilustración 13 - Diseño de la página del registro de usuarios

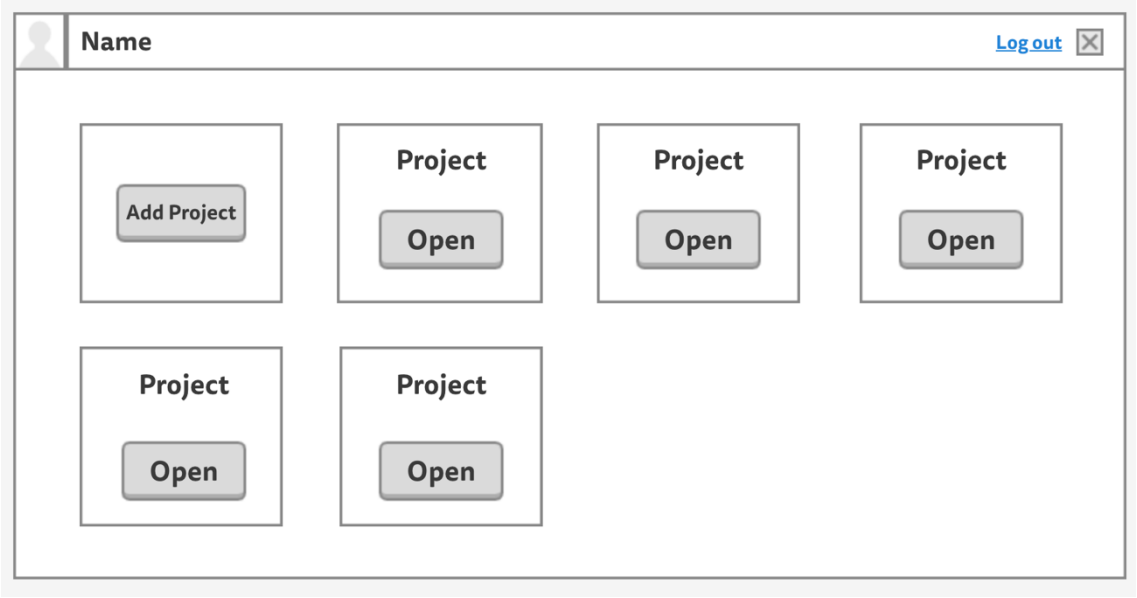

Ilustración 14 - Diseño de la pantalla de creación y selección de proyectos

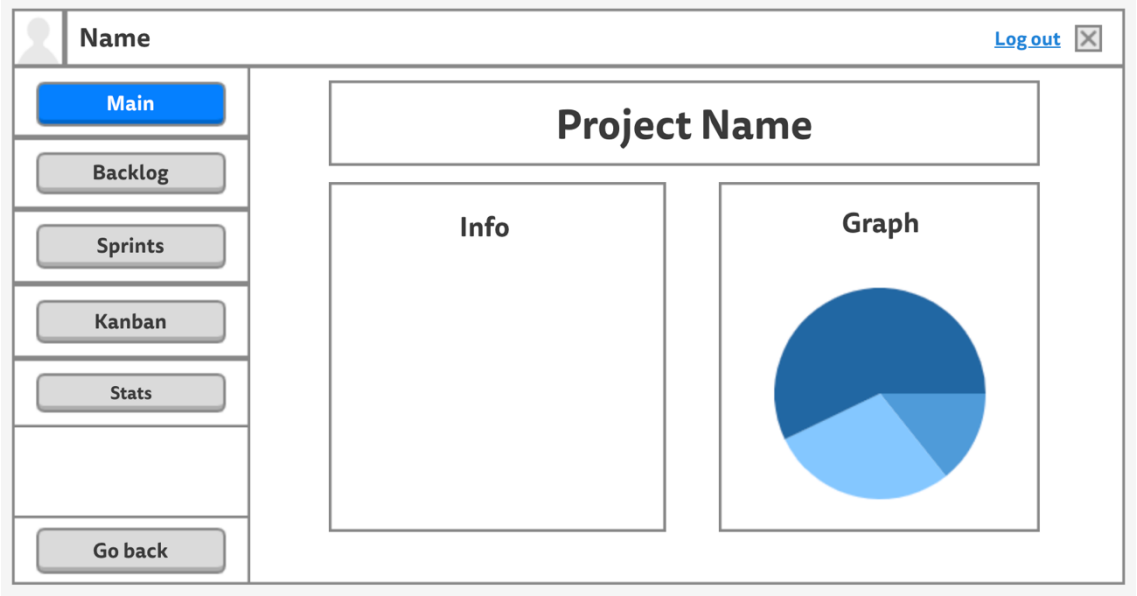

Ilustración 15 - Vista principal del proyecto

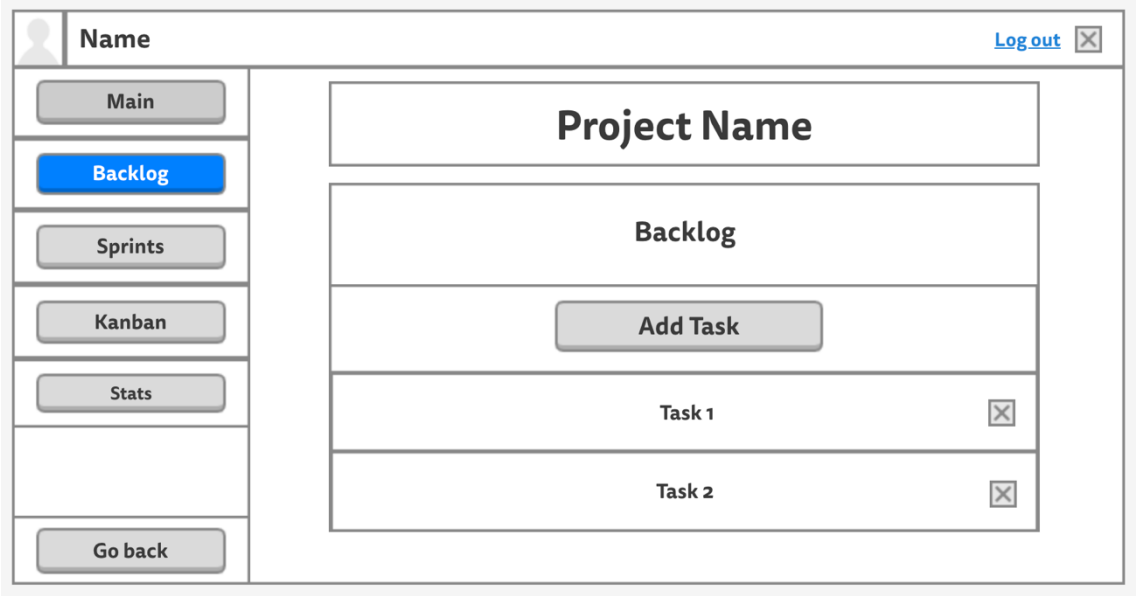

Ilustración 16 - Vista del Backlog del proyecto

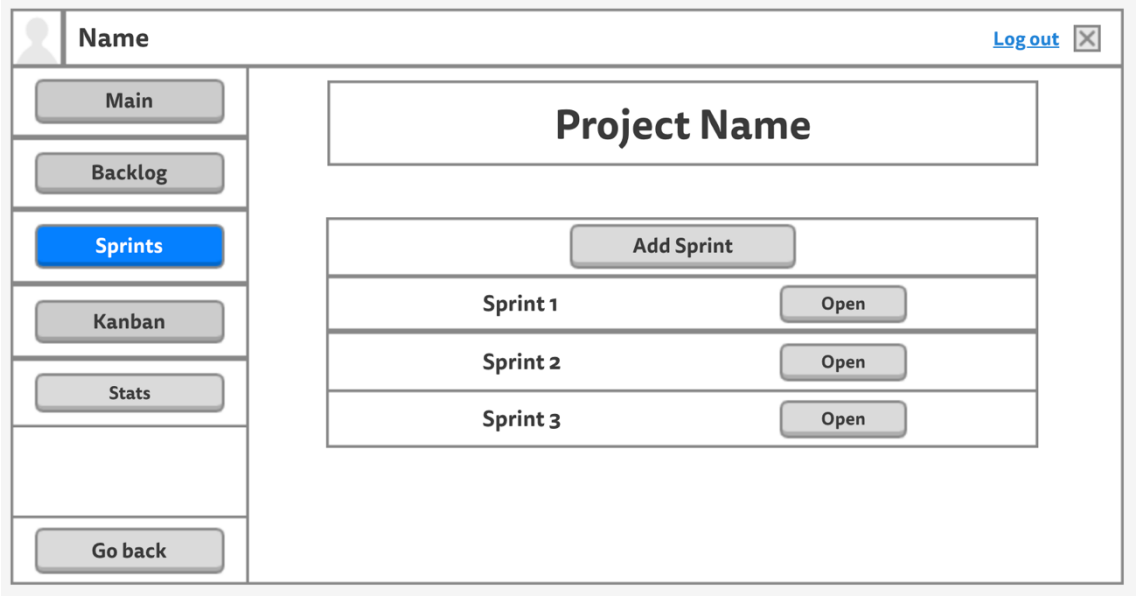

Ilustración 17 - Pantalla de selección de Sprints

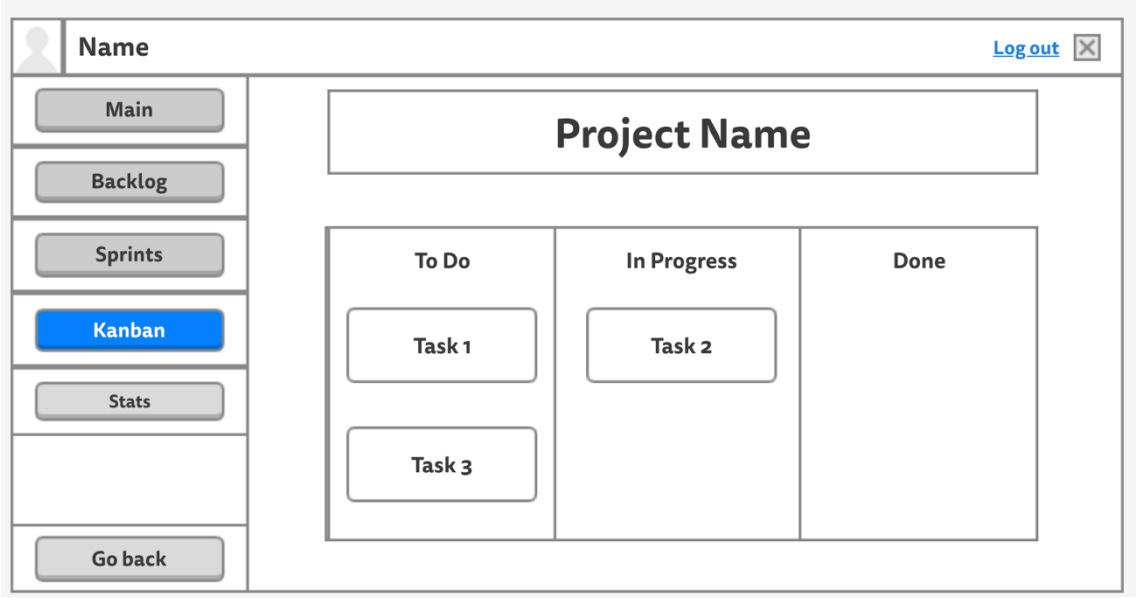

Ilustración 18 - Diseño del tablero Kanban

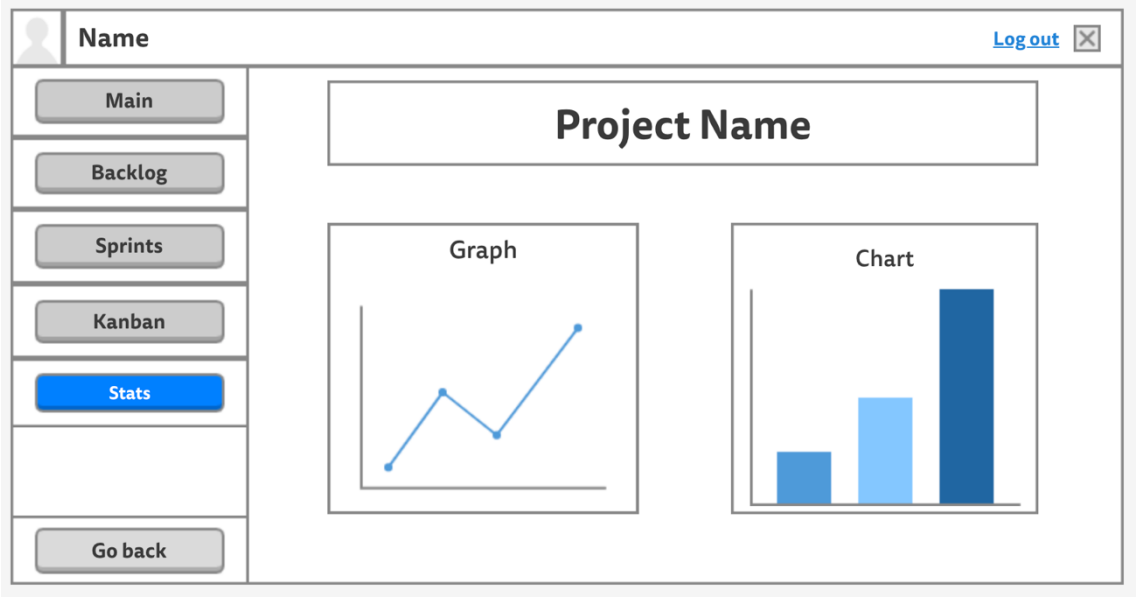

Ilustración 19 - Diseño del apartado de estadísticas

Al mismo tiempo que se estuvo diseñando las interfaces y vistas de como podría ser la aplicación, se realizó una encuesta entre alumnos y profesionales del mundo del desarrollo de videojuegos y de *software*, sobre metodologías *Agile* y herramientas de planificación y gestión de proyectos. El objetivo de la encuesta era obtener la mayor cantidad de información posible para orientar la herramienta a las necesidades de la gente. Gracias a esto, se pueden definir las características que el proyecto tendrá y cuales no incluirá.

La encuesta era muy importante para conocer las necesidades de los usuarios de estas herramientas. Por eso se pensó que las preguntas se podían dividir en tres bloques de dos preguntas. El primero analizará la forma y metodología de trabajar de los usuarios y las herramientas que emplean. El segundo bloque se centrará en lo que necesitan y buscan a la hora de elegir una herramienta. Y, por último, tratará de los aspectos que no consideran necesarios. Así que la encuesta se compone de las siguiente seis preguntas:

- ¿Utilizas metodologías *Agile* en la creación de vuestros proyectos/videojuegos?
- ¿Qué herramientas usas para la planificación de vuestros proyectos/videojuegos?
- ¿Qué características te parecen fundamentales para elegir una herramienta sobre otra?
- Si pudieras elegir una característica o una funcionalidad que te gustase tener en una herramienta de planificación, ¿cuál sería?
- ¿Qué funcionalidades te parecen irrelevantes para una herramienta de planificación?
- ¿Conoces alguna funcionalidad que disponga tu herramienta, pero no uses en tus proyectos?

Las dos primeras preguntas dan información sobre si usan metodologías *Agile*, y que plataformas emplean para sus proyectos. Conocer los métodos de planificación permite reconocer las opciones más elegidas no contempladas en el apartado, Estado del arte. Por otro lado, también se puede ver cuales son las herramientas más usadas entre los encuestados, mostrando cuales son más comunes, en este caso Trello y Jira, como se observa en la Ilustración 20.

Ilustración 20 - Resultados de las dos primeras preguntas de la encuesta

Las siguientes dos preguntas, la tercera y la cuarta, tienen el objetivo de buscar las características que más se adaptan a las necesidades de los encuestados. La tercera se componía de un listado de características que se consideraron determinantes tras estudiar las herramientas ya existentes. De las nueve opciones, cuatro destacaron sobre el resto en los resultados, como se muestra en la Ilustración 21. Estas fueron: ser una plataforma gratuita; disponer de un *Backlog* de tareas; poder realizar una organización por *Sprints*; y la más votada, que sea una herramienta sencilla. Además, otras dos opciones tuvieron una alta aceptación, aunque se encuentran por debajo del cincuenta por ciento, que fueron: incluir un tablero *Kanban*, y disponer de una función para gestionar el tiempo.

Ilustración 21 - Resultados de la tercera pregunta de la encuesta

Acto seguido, en la cuarta pregunta, se pedía especificar que tipo de funcionalidades le pedirían a una herramienta. Gracias a estas dos preguntas, se han identificado las funcionalidades que la gente incluirían en la nueva plataforma. Además, el cuestionario incluye dos últimas preguntas sobre que funciones no necesitan, y cuales disponen, pero no usan en sus proyectos.

Como se ve reflejado en los resultados, que se encuentran en el Anexo 10.3 – Resultados de la encuesta, se puede ver que hay diferentes enfoques pero que se coincide en las mismas características. Donde el objetivo es conseguir una herramienta sencilla, que visualmente sea intuitiva, y que esté muy centrada alrededor de las tareas. Uno de los aspectos a tener muy en cuenta, y que refuerza este objetivo, es que mucha gente no tiene la experiencia suficiente en estas herramientas. Por lo que se decidió desarrollar el *software* bajo esos objetivos, e ir ampliando el contenido centrándose en la opinión de los usuarios.

Aunque la encuesta ha reflejado que la herramienta no necesariamente tiene que estar especializada en videojuegos, se decidió mantener ese aspecto, y, además, con la idea de permitir a los usuarios poder llegar a personalizar el proyecto dada las necesidades de este. De esta forma se conseguiría una herramienta mucho más completa.

5.3.3. Tareas Realizadas

Durante este *Sprint* se acometieron estas tareas:

- Montar entorno de trabajo
- Realizar conexión entre el *backend* y el *frontend*
- Diseñar las interfaces de la herramienta
- Realizar encuesta
- Analizar resultados de la encuesta

5.4. Sprint 4

5.4.1. Planificación Inicial

En este *Sprint* se tomó la decisión de acelerar el ritmo de trabajo, con la idea de recuperar el tiempo que no se aprovechó en los meses anteriores. El objetivo de esta decisión fue el de intentar cumplir con los plazos de entrega del Trabajo de Fin de Grado (TFG) para la convocatoria de junio. A raíz de esto, los siguientes *Sprints* tendrían una duración de dos semanas, en vez de las cuatro semanas de los *Sprints* anteriores.

El cuarto Sprint comenzaría el 1 de mayo, finalizado el día 14 de ese mismo mes. Los objetivos de este *Sprint* son: realizar el sistema de *login* y registro de usuarios; hacer el sistema de creación y cargado de proyectos; y, maquetar la estructura de la web de estos componentes, tal y como se diseñó en el *Sprint* anterior.

5.4.2. Desarrollo del Sprint

Lo primero que se hizo fue generar los componentes y servicios que se iban a necesitar para cumplir con los objetivos de este *Sprint*. Además, se crearon las rutas para los componentes y así que se puedan conectar las diferentes pantallas. Por lo que la estructura del *frontend* se distribuyó de la siguiente forma:

- *Components*: son los encargados de almacenar los elementos visibles de la web, cada componente contiene tres archivos de este tipo: HTML, CSS y TypeScript.

- *Models*: donde se guardarán las interfaces necesarias para representar los objetos que se van a crear durante este TFG.
- *Services*: contendrán las llamadas y peticiones que se realizarán al servidor.
- *Guards*: su objetivo principal es la de proteger las rutas de dentro del proyecto, cuya función es comprobar si se pueden acceder a ellas o no.

Para poder trabajar con estos componentes, se crearon las interfaces de los objetos que se iban a necesitar, *projects* y *users*. En este paso, se cometió un error que se arrastró durante gran parte de la implementación, y es que se crearon estas interfaces con los datos necesarios en el momento y no como estaban originalmente diseñados. Esto provocó que cada vez que se desarrollaba una nueva funcionalidad, se tenía que añadir estos nuevos campos a los modelos, de forma que añadir un campo afectaba a varios archivos. Esto podría no haber sucedido si se hubiera trabajado directamente con el objeto completo desde un principio.

Para implementar el sistema de *login* y registro de usuarios se necesitaron dos servicios, uno para cada acción. Ambos servicios son dos peticiones *POST*, que enviaran la información necesaria al servidor en el cuerpo de estas, con la ruta que previamente se ha predefinido. En el caso del *login*, se mandará el email y contraseña, y en el proceso de registro se añadirá también el nombre del usuario.

Una vez la petición de registro de usuario haya llegado al servidor, lo primero que hará es comprobar a través de un *middleware*, *verifyEmail*, si el correo ya está registrado en la base de datos. Si esta dirección de correo está registrada, el servidor tratará de guardar este nuevo usuario en la colección. En caso de que se produzca un error, la respuesta del servidor será un código 500 con el mensaje de error. Por otro lado, si todo es correcto, el servidor devolverá un código 200 además de un mensaje confirmando el registro del usuario. Este comportamiento se puede ver reflejado en el diagrama de secuencia de la Ilustración 22.

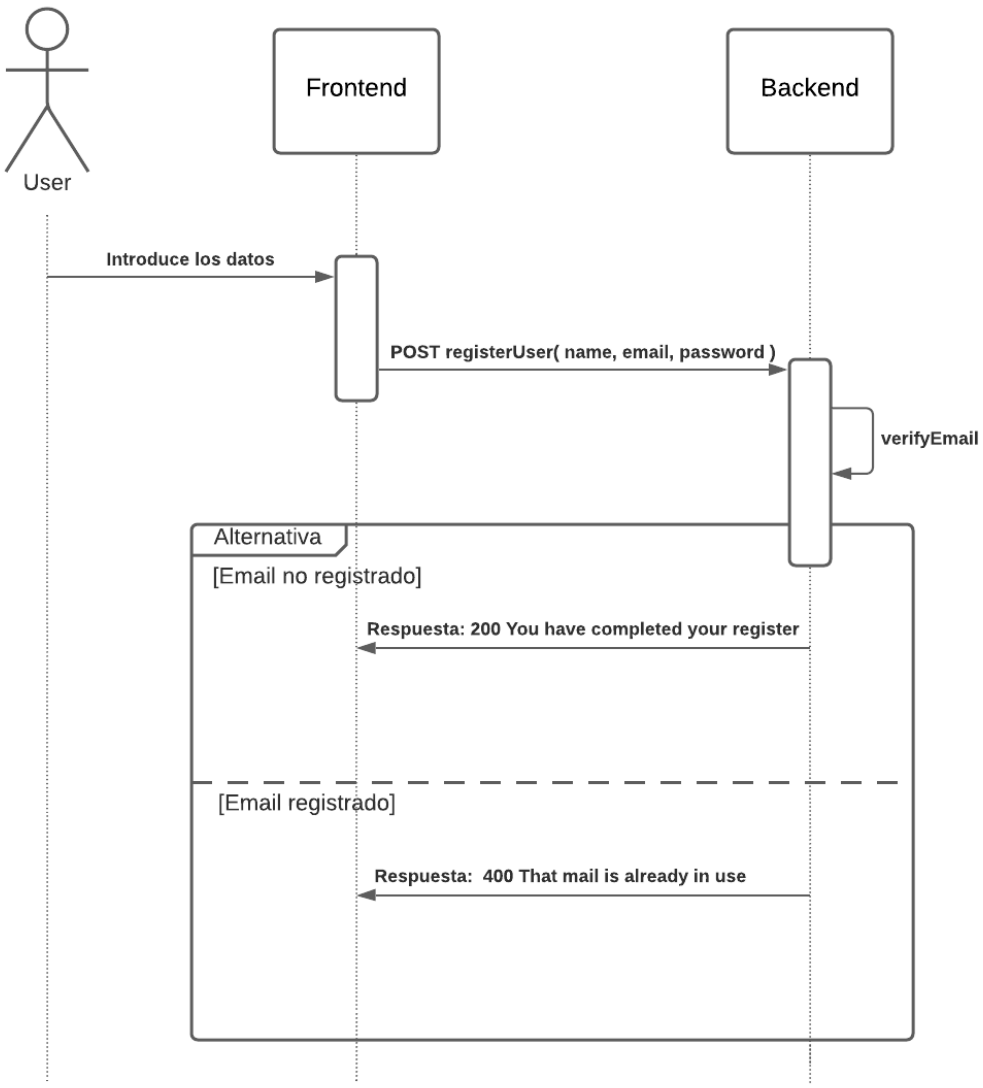

Ilustración 22 - Diagrama de secuencia con el funcionamiento del registro de usuarios

En el caso del *login*, como se observa en la Ilustración 23, lo primero que se hará es buscar un usuario en la base de datos con el email que se ha mandado con la petición. En caso de error, actuará de la misma forma que en el registro, devolviendo el código 500 con el mensaje. Y si no existe un usuario con ese correo, la respuesta será un código 404. En caso de que todo sea correcto, se comprobará si la contraseña guardada y la proporcionada en la petición de *login*, coinciden. Si no coinciden, se enviará un código 401, y por el caso de que todo fuese correcto, la respuesta sería un código 200, junto con la información del usuario sin la contraseña, y con un *token*. Estos datos se guardarán, a través de un servicio, en la Local Storage del navegador.

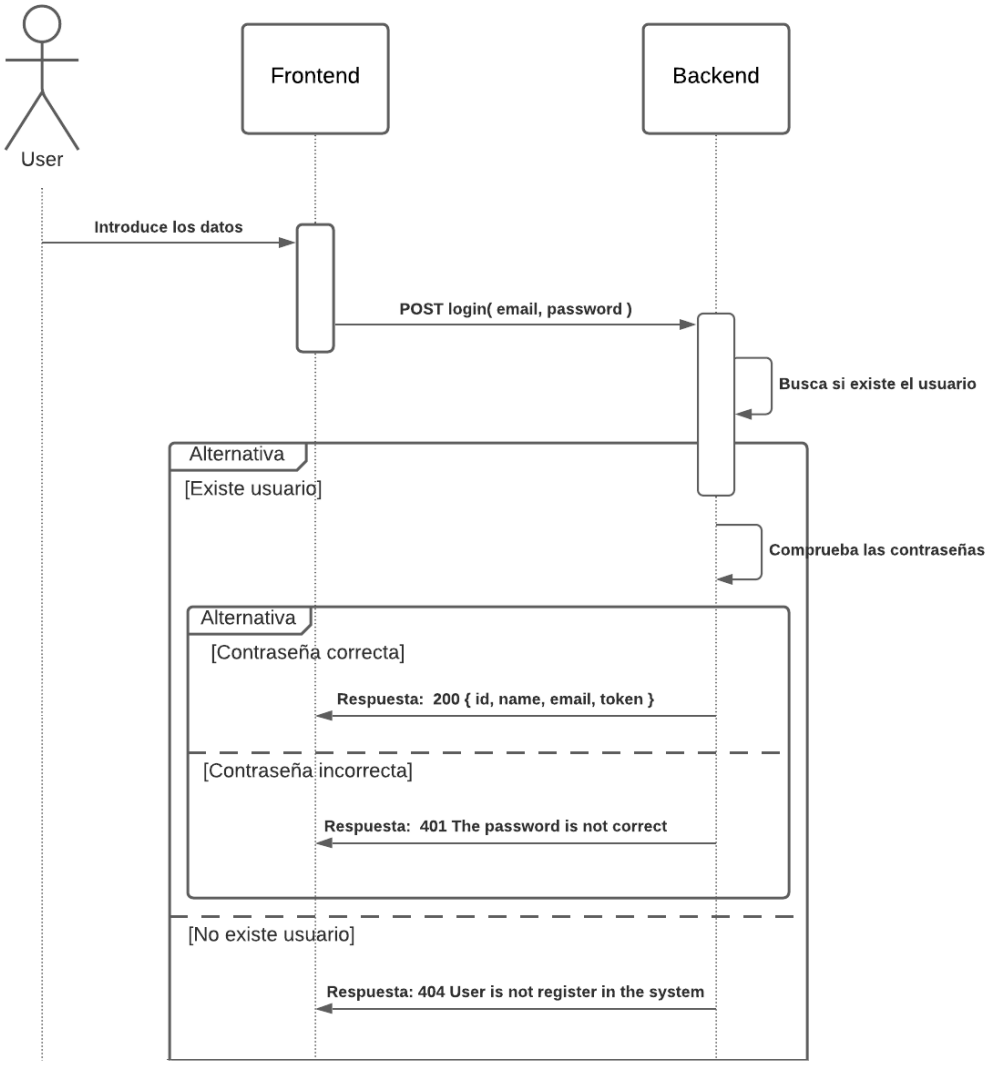

Ilustración 23 - Diagrama de secuencia con el funcionamiento de la identificación de usuarios

Este *token* se puede generar gracias a la librería de Node.js, que permite el uso del estándar *JSON Web Token* (JWT) [16]. El *token* se obtiene con el id del usuario y una *key* propia del servidor, "tfgvideojuegos", y tiene una validez de doce horas. Se decidió que expirará después de ese tiempo, ya que se consideró un tiempo razonable por el cuál, una persona no necesitaría identificarse más de una vez al día, sin que se interrumpa su sesión una sola vez.

La finalidad de este *token* es verificar todas las acciones que realicen los usuarios y de proteger la integridad de los datos del sistema, ya que este se enviará en el *x-access-token* del *header* de todas las *requests*. Esto sucederá gracias a un interceptor que se encargará de añadirlo

automáticamente cada vez que se haga una HTTP *request*. Y una vez en el servidor, se verificará que ese *token* pertenece al usuario que está realizando la petición.

Ya una vez con el sistema de *login* y de registro de usuarios creado, se crearon dos componentes nuevos. Una *navbar* en la que se muestra el nombre del usuario y un botón con la opción de cerrar la sesión y volver a la pantalla de *login*. El segundo componente, llamado *ProjectGrid*, se encarga de mostrar una cuadricula con los proyectos a los que pertenece el usuario, y además la opción de crear nuevos proyectos, como se puede ver en la Ilustración 24. Para poder cargar los proyectos en los que está en usuario, Angular dispone de un método que se llama *ngOnInit*, que se ejecuta cada vez que se inicia un componente. Dentro de este método, este componente tendrá una variable, que será un *BehaviorSubject*, que se suscribirá a otra del *ProjectService*, el servicio relativo a la gestión de los proyectos. En esta variable se almacenarán los proyectos de ese usuario. Una vez se está suscrito a esa variable, se realiza la llamada a la petición *GET*, que se encarga de buscar los proyectos con el id del usuario. Una vez el servidor devuelve los datos, al estar suscrita a esa variable, se actualizará automáticamente al guardar la información. Este proceso se repetirá también a la hora de abrir proyecto en concreto. Gracias a esto, no hará falta tener que actualizar la página para ver los datos cuando se realice algún cambio.

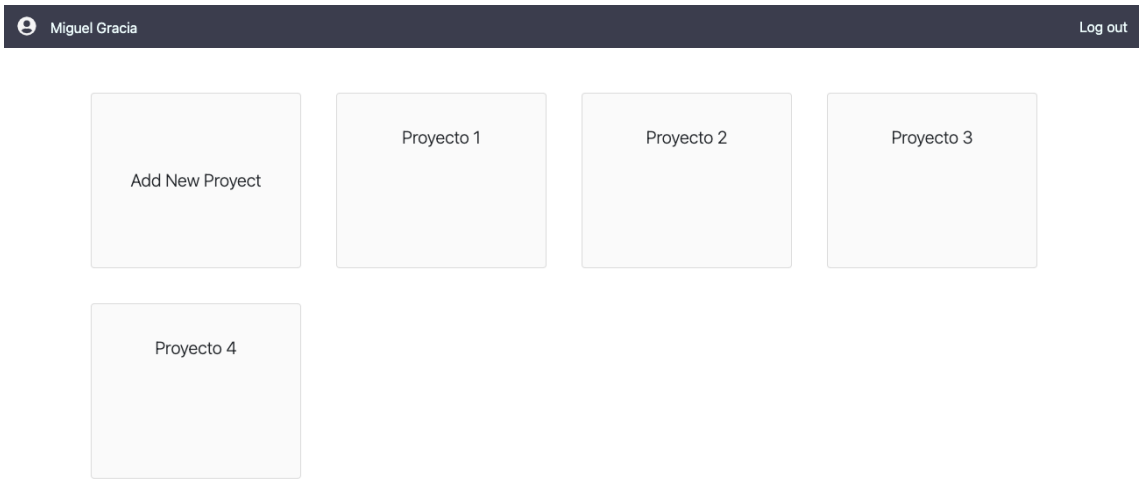

Ilustración 24 - Interfaz de selección de proyectos en el Sprint 4

A la hora de añadir un proyecto se mostrará el Modal que está reflejado en la Ilustración 25. Este modal, que se obtiene con la librería Ng-Bootstrap [17], incluye con un formulario en el que se tendrá que añadir el nombre de forma obligatoria, y una descripción, que es opcional.

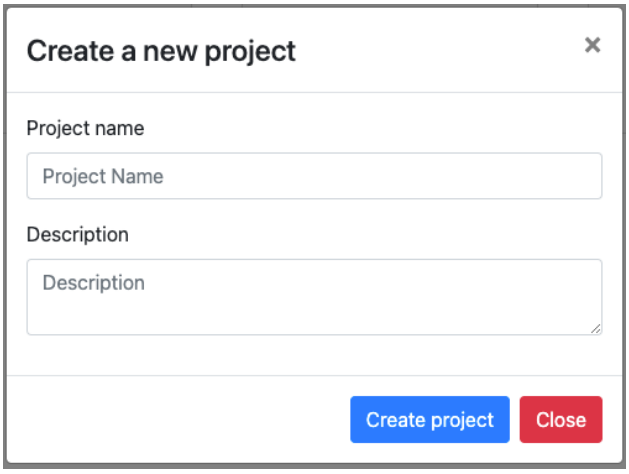

Ilustración 25 - Modal con el formulario para añadir proyectos

Una vez el usuario ha introducido los datos de forma correcta, un servicio se encargará de mandar una petición *POST* al servidor con los datos del nuevo proyecto y el id del usuario. El *backend* entonces verificará el *token* de la petición a través de un *middleware*, *verifyToken*, y se encargará de rellenar los datos en la colección de proyectos. Si todo es correcto, el servidor devolverá ese proyecto, que se añadirá a la variable donde están almacenados los proyectos del usuario.

Llegados a este punto, se necesitaba un componente que aglutinara todas la pantallas y opciones a realizar dentro de los proyectos de los usuarios, así como la *sidebar* para seleccionar y cambiar entre las diferentes opciones. Por lo que se creó el componente *ProjectHolder*, con la función de distribuir los elementos de la interfaz principal del programa, la *sidebar* y el contenido de los diferentes componentes de la aplicación. Este componente se puede ver en la Ilustración 26.

Ilustración 26 - Estructura del componente ProjectHolder

Dentro de este contenido, se encontrará el *router-outlet*, una directiva de Angular que permite mostrar los diferentes componentes asociados a las rutas hijas del componente en el que se encuentra. Junto con el *ProjectHolder*, también se desarrolló el componente para la *sidebar*, desde dónde se redijeran a las diferentes rutas. De este modo, cuando se vayan creando nuevas pantallas, solo habrá que añadirlas al *sidebar* y cuando sean seleccionadas, el sistema las mostrará sin refrescar el navegador. En este momento se creó el componente *ProjectScreen*, que será la vista principal del proyecto y la primera que se muestre al abrir uno. En ella se mostrará la información básica, como el número de participantes, la descripción, quien ha creado el proyecto, y más datos que se puedan añadir en un futuro.

5.4.3. Tareas Realizadas

Durante este *Sprint* se acometieron estas tareas:

- Crear los modelos de los objetos
- Implementar sistema de *login* y de registro de usuarios
- Desarrollar un sistema de creado y carga de proyectos
- Crear *sidebar* y *navbar*
- Montar estructura de la interfaz principal de los proyectos
- Crear la pantalla de inicio de un proyecto

5.5. Sprint 5

5.5.1. Planificación Inicial

El *Sprint* se planificó del 15 de mayo al 30 de mayo, con el objetivo principal de desarrollar un sistema capaz de crear tareas en la herramienta. Para lograr este objetivo también se ha implementar la creación de *Sprints* y añadir el *Backlog* a los proyectos.

5.5.2. Desarrollo del Sprint

Recapitulando un poco, el estado de la implementación en este punto es el siguiente: se dispone de una herramienta con un sistema de *login* y registro de usuarios. El programa puede crear proyectos y visualizar la pantalla principal de cualquier proyecto de un usuario. El servidor tiene un sistema de autentificación por *tokens* que permite verificar la identidad del usuario.

En este momento se necesita empezar a trabajar con las tareas del proyecto, por lo que se empezó a desarrollar el *Backlog* y los *Sprints* del proyecto. Como se ve en el diseño del modelo de la sección 5.3.2, un proyecto dispondrá de un *Backlog* y de un *array* de *Sprints*. La idea

principal es que el *Backlog* contenga un listado de tareas que no estén asignadas a un *Sprint*, y que cada *Sprint*, disponga únicamente de sus tareas. De esta forma, damos la libertad a los usuarios de poder trabajar de la forma que más se adapte a sus necesidades.

La estructura de los componentes del *Backlog* y de los *Sprint* será muy parecida, ya que solo se diferencian en un par de aspectos. En el desarrollo del *Sprint* anterior, se ha comentado como se tiene una variable que contiene el proyecto vinculado al servicio, a través de una suscripción. Esto nos permite actualizar el proyecto seleccionado sin recargar la página. Teniendo esto en cuenta, se pensó que la forma más cómoda y sencilla para realizar cambios en el proyecto, era crear un servicio al que se le pasase el objeto entero al servidor con la información ya añadida, en vez de mandar los cambios, y que se gestione en el servidor. Gracias a esta decisión se acortó el tiempo de desarrollo ya que solo se implementará una única petición *PUT* en el servicio.

El listado de tareas tendrá la opción de añadir tareas al principio de la lista. Al crearse una tarea se mostrará un modal que incluirá un formulario, como el de la Ilustración 27, con la información relativa a esta, nombre, descripción, tiempo estimado, tiempo gastado y de qué tipo de tarea es. Para completar esta última opción, se añadieron cuatro tipos de tareas por defecto en los proyectos. Estas cuatro categorías son: *Code, Art, Design* y *Testing*. Se eligieron pensando principalmente en el desarrollo de videojuegos, pero con la idea de que, durante el desarrollo del TFG, dar la posibilidad a los usuarios de personalizar estas categorías.

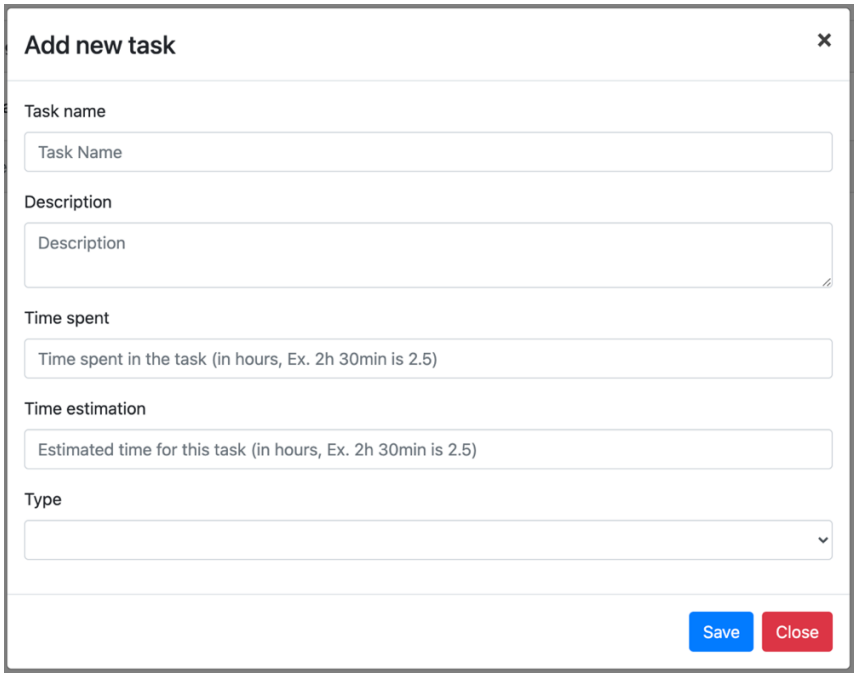

Ilustración 27 - Formulario con la creación de tareas

El resto de la lista se identificarán con nombre de las tareas, y cada una de ellas tendrá la opción de eliminarla, como se muestra Ilustración 28. Cada uno de estos elementos abrirán un modal, parecido al de añadir tareas, donde se mostrará la información de la tarea y se podrán guardar cambios. Además, en este modal se puede cambiar el estado de la tarea entre *To Do*, que es el estado por defecto de cuando se añaden una nueva, *In Progress* y *Completed*.

Ilustración 28 - Listado de tareas

Con el listado de tareas creado, y el componente *ProjectScreenBacklog* y *ProjectScreenSprintDetail*, que muestran este listado para el *Backlog* y cualquier *Sprint*, respectivamente. Entonces se creó un componente por el cuál, se pudieran añadir *Sprints* y poder seleccionarlo para acceder a estos. Para ello, se creo un componente, *ProjectScreenSprint*, que tendría un aspecto similar al componente dónde se pueden ver los proyectos y crearlos, *ProjectGrid*. Pero en este caso muestran los *Sprints* como un listado en vez de una cuadricula como se ve en la Ilustración 29. Al seleccionar *"Add New Sprint",* se creará un *Sprint* nuevo directamente.

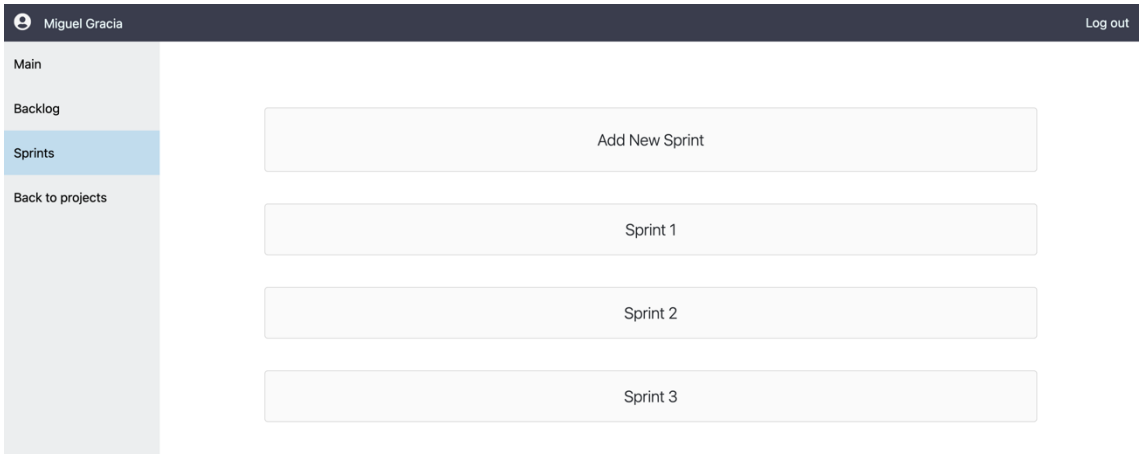

Ilustración 29 - Pantalla con los Sprints de un proyecto

5.5.3. Tareas Realizadas

Durante este *Sprint* se acometieron estas tareas:

- Desarrollar el componente del *Backlog*
- Crear un sistema de creación y edición de tareas
- Creación y visualización de *Sprints*

5.6. Sprint 6

5.6.1. Planificación Inicial

El sexto *Sprint* se llevo a cabo entre el 31 de mayo y el 17 de junio, este *Sprint* se planteó avanzar sabiendo que no se podría cumplir el objetivo de entregar el Trabajo de Fin de Grado en la convocatoria de junio. Se tomó esta decisión pensando en el potencial del proyecto y de las funcionalidades que quedarían sin hacerse. De todas formas, el avance realizado en los dos *Sprints* anteriores y lo planificado para este, permitirían preparar bastantes mejores para los próximos.

El objetivo principal de este *Sprint* es completar el apartado estadísticas de los proyectos y también, se incluirá la opción de añadir participantes a los proyectos. Por otro lado, se empezará a diseñar las mejoras en la aplicación y para ello, al finalizar los objetivos del *Sprint*, se realizarán pruebas con posibles usuarios finales. El *feedback* recibido por su parte permitirá marcar futuros objetivos y funcionalidades, así como corregir aspectos y detalles que no sean del agrado de este público objetivo.

5.6.2. Desarrollo del Sprint

Teniendo en cuenta los resultados de la encuesta que se realizó en el *Sprint* 3, uno de los comentarios recibidos fue que se consideraba indispensable disponer de los gráficos *Burn Up* y *Burn Down*, así como de tener una herramienta para la estimación del tiempo de las tareas. Es por eso, que se decidió tomar estos tres ejemplos para el apartado de estadísticas de los proyectos.

Para realizar las tres gráficas que acaban de mencionar, se han empleado las librerías ng2-charts [18] y chart.js [19]. Estas librerías cuentan con diferentes tipos de gráficas y diagramas, como pueden ser el gráfico de barras, de áreas, mapa de burbujas o gráficos circulares o en forma de donuts, entre otros. Esta librería se usó para representar gráficamente los datos que se disponen en el proyecto y que se pueden extraer de la información de las tareas. Para recoger toda está información, se creó una función que se encargaba de iterar entre todas las tareas de los *Sprints* y del *Backlog*. Gracias a este proceso, se pudieron extraer la cantidad total de horas estimada y gastadas por *Sprint*, las tareas pendientes y realizadas en cada *Sprint*.

Para hacer más fácil la implementación de los gráficos se creó la interfaz *Graph*, con los siguientes parámetros: opciones, colores, tipo de gráfico, si incluye legenda o no, los datos que van a mostrar y las etiquetas.

El gráfico *Burn Up* es un gráfico de líneas que muestra el avance de las tareas completadas a lo largo del proyecto. En la siguiente imagen, Ilustración 30, se puede observar la implementación de esta estadística dentro de la herramienta, en un proyecto de ejemplo con dos *Sprints*. En ella se puede ver a la izquierda el número de tareas y abajo la etiqueta con el nombre de los *Sprints*, dónde las líneas van creciendo en cada iteración. Se decidió hacer un cambio sobre el gráfico original, que consiste mostrar el número total de tareas, *scope*, que hay en el momento que finalizas una iteración. En este ejemplo se puede ver como en el *Initial Scope* están las dos tareas pendientes del primer *Sprint*, y en el *Sprint* 1 hay dos completas y faltaría por terminar la del *Sprint* 2.

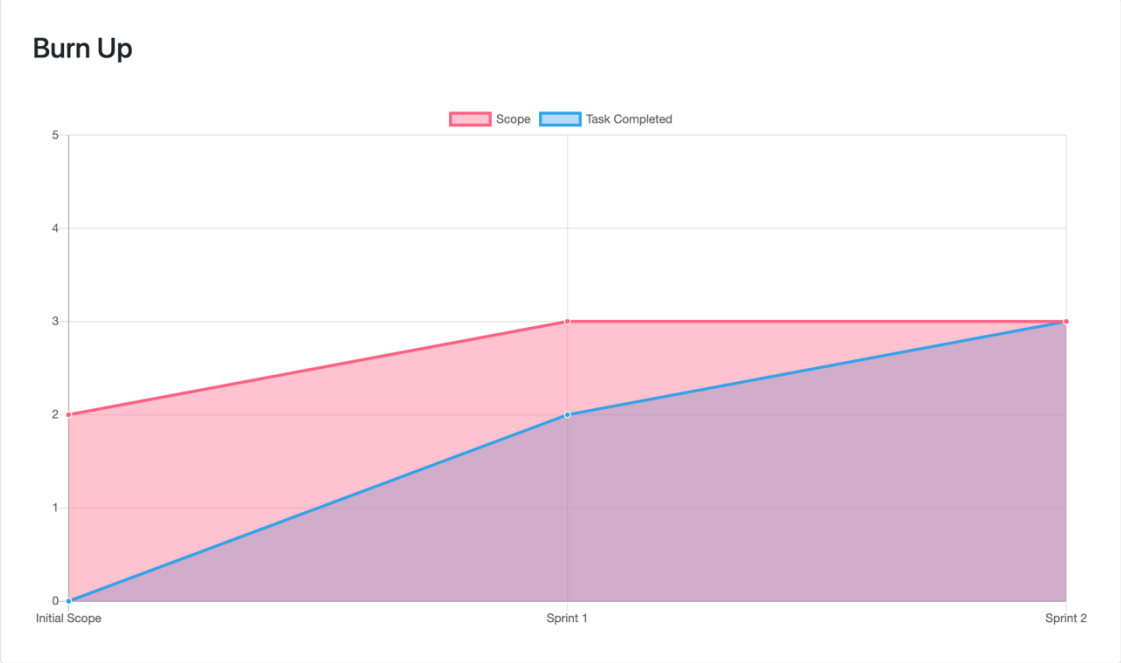

Ilustración 30 - Gráfico Burn Up

En el caso del gráfico Burn Down, se muestra un gráfico también de líneas, que tiene la misma información que el gráfico anterior. Pero en vez de mostrar las tareas completadas, se muestra el total de tareas del proyecto, y como las tareas pendientes por realizar van disminuyendo hasta llegar a cero. La Ilustración 31 muestra un ejemplo de este gráfico dentro de la herramienta.

Ilustración 31 - Gráfico Burn Down

A diferencia de los otros dos, el gráfico de tiempo estimado – gastado es un gráfico de barras en el que se compara esas dos estadísticas en cada iteración, así como en el *Backlog*. Gracias a este gráfico se puede ver si durante los *Sprints* se están estimando bien la duración de las tareas, o por el contrario si no se está cumpliendo con la estimación original. Este tipo de estadísticas pueden se de ayuda a la hora de distribuir la carga de trabajo. Haciendo que el alcance de un *Sprint* sea de la forma más equilibrada y optima posible. El ejemplo de la Ilustración 32 se muestra las tres posibles situaciones, una estimación perfecta, y dos desviaciones, una por encima y otra por debajo de lo estimado.

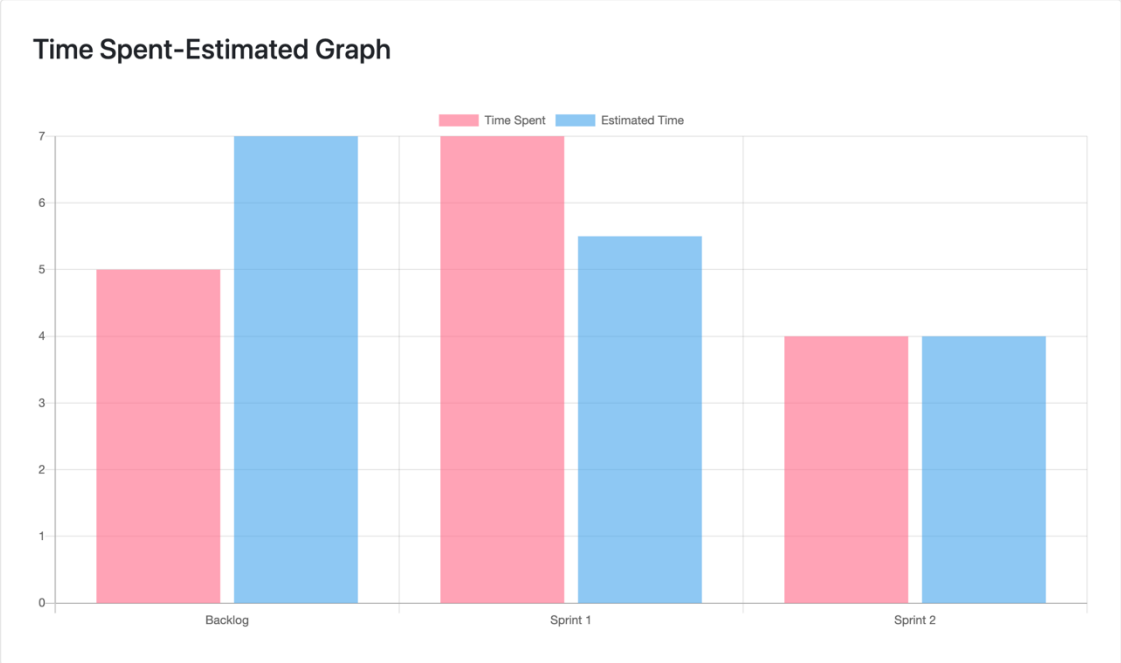

Ilustración 32 - Gráfico con el tiempo estimado y gastado en las tareas

Una vez se terminó de implementar el apartado de estadísticas, el siguiente objetivo era crear la funcionalidad de poder añadir participantes a los proyectos. Para completar este objetivo, se añadió un nuevo servicio con una petición *PUT*, a la que se le mandaba en el cuerpo de esta el email del nuevo usuario y el id del proyecto. En el servidor, lo primero que se comprueba es si está registrado un usuario con ese correo en el sistema. En caso de que no exista, el servidor manda el código 400 con un mensaje de que el usuario no existe. Por el otro lado, si el usuario existe, comprueba que no este en el proyecto, y si se cumple esa condición lo añade. Visualmente, esta funcionalidad se añadió en la página principal del proyecto al lado de la información básica. En ella se veía un recuadro con los participantes y un botón que abría un modal con un formulario para introducir el correo electrónico. Además, se añadió a las tareas la posibilidad de asignar usuarios a ellas, así como la información de quien creó la tarea.

Con todos estos avances en el proyecto, se realizaron unas pruebas de usabilidad con cuatro usuarios. La prueba consistía en que cada uno de ellos disponía de total libertad para probar la herramienta y dar su opinión sobre ella. Además, al finalizar la prueba se les hacían cinco preguntas que permitirían conocer el estado actual del proyecto, sus posibles puntos de mejora y que más le hace falta. Las preguntas eran las siguientes:

- ¿Qué puntuación le pones a la herramienta? (Del 0 al 10, dónde el 0 es muy mala y el 10 muy buena)
- ¿Podrías usarla en alguno de tus proyectos actuales?
- ¿Crees que es fácil de usar?
- ¿Qué cambiarías para mejorar la herramienta?
- ¿Añadirías alguna funcionalidad nueva?

El resultado de estas pruebas concluyó con los siguientes puntos:

- Tres de las cuatro personas dijeron que sí que podrían usar la herramienta para alguno de sus proyectos.
- Los cuatro coincidieron en que la herramienta era fácil de usar.
- Pidieron un mayor *feedback* visual de lo que es interactuable.
- Poder crear tipos nuevos de tareas.
- Hacer más visible el proyecto en el se está y poder seleccionar el *Sprint* actual.
- Separar las estadísticas en un apartado para cada una en vez de mostrarlas juntas.
- Poder mover tareas entre el *Backlog* y los *Sprints*.
- Cuando te registras, que aparezca en el *login* directamente el correo del usuario.
- Añadir un tablero *Kanban* con la funcionalidad del *Drag and Drop.*
- Que la información de una tarea se muestre en otra pantalla en vez de en un modal.
- Añadir una sección de comentarios en las tareas.
- Poder subir imágenes en las tareas.
- Poder finalizar *Sprints* y proyectos.
- Tener un apartado de *bugs*.
- Poder añadir y modificar las fechas de los *Sprints*.
- Tener un calendario con los eventos del proyecto.
- Tener una página personal con toda la información relativa al usuario de todos sus proyectos, como un listado de todas sus tareas.

5.6.3. Tareas Realizadas

Durante este *Sprint* se acometieron estas tareas:

- Crear el apartado estadísticas
- Opción de añadir participantes
- Realizar pruebas de usabilidad

5.7. Sprint 7

5.7.1. Planificación Inicial

Tras realizar las pruebas de usabilidad y definir el alcance final del proyecto, se planificaron el *Sprint* actual y el próximo, teniendo ambos una duración de tres semanas. Este *Sprint* comenzaría el 15 de julio y finalizaría el 7 de agosto. El objetivo propuesto es implementar un Tablero *Kanban* con la funcionalidad del *Drag and Drop* para mover las tareas entre los diferentes tableros. También se realizarán algunas mejoras entre las funcionalidades ya desarrolladas, con los datos obtenidos en las pruebas con los usuarios.

5.7.2. Desarrollo del Sprint

Los primeros pasos que se tuvieron en cuenta fueron como podía ser el *Kanban* de la herramienta, el diseño se realizó en el *Sprint* 3, pero no se analizó como distribuir las tareas en ese momento. Se llegó a la conclusión que el tablero debía mostrar las tareas del *Backlog* y también las del *Sprint* actual. De esta forma se realizaba también una de las peticiones que un usuario propuso cuando probó la herramienta.

Se añadió un nuevo parámetro, *status*, que definía la situación de un *Sprint*. Al crearse uno nuevo se definía como *Ready to Start*, y este iba variando durante su ciclo de vida, a *In Progress*, cuando es el *Sprint* en activo, y *Completed*, en el momento que se finalizaba el *Sprint*. Para modificar este estado se añadió un botón con el que se realizaban las transiciones. También se añadió un modal a la hora de crear los *Sprints*, en el que se debía de añadir las fechas de inicio y final de estos. Toda esta información se muestra en una nueva cabecera junto al titulo del *Sprint*, como se puede ver en la Ilustración 33.

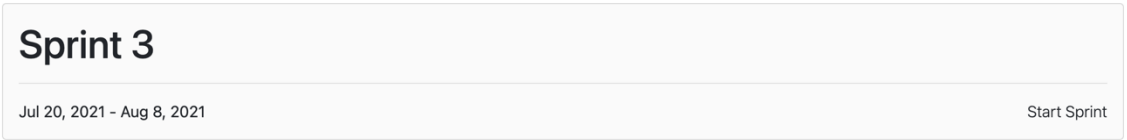

Ilustración 33 - Cabecera con la información relativa a un Sprint

Aprovechando estos cambios se mejoró la interfaz de selección de proyectos, donde ahora también muestra además el estado y las fechas. Ejemplo visible en la siguiente imagen, Ilustración 34.

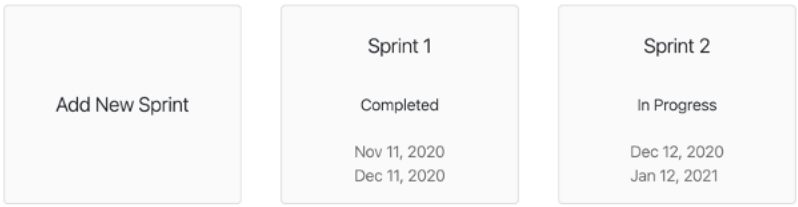

Ilustración 34 - Interfaz de selección de Sprints

Ahora que ya se tiene la opción de indicar que *Sprint* está activo, se empezó a montar el tablero *Kanban*. Para ello se creó la estructura que tendrían los recuadros de las tareas y las columnas, donde cada una representaría el estado de las tareas. Para implementar el sistema de *Drag and Drop* se utilizó la librería NgDragDrop [20], se eligió esta opción frente alternativas más completas, como el del Angular Material CDK [21], por su sencillez. Esta librería permite asignar las directivas *draggable*, para los elementos que se pueden arrastrar, y *droppable*, a aquellos elementos donde se van a mover. De esta forma solo habría que asignárselos a las cajas de las tareas y a las tres columnas, *To Do, In Progress y Completed*. Cada vez que una de estas tareas se mueve se cambia su estado a través de un evento que indica a que columna se ha desplazado. También se incluyó la opción de poder añadir tareas desde el *Kanban*. El resultado obtenido de esta implementación está reflejado en la Ilustración 35.

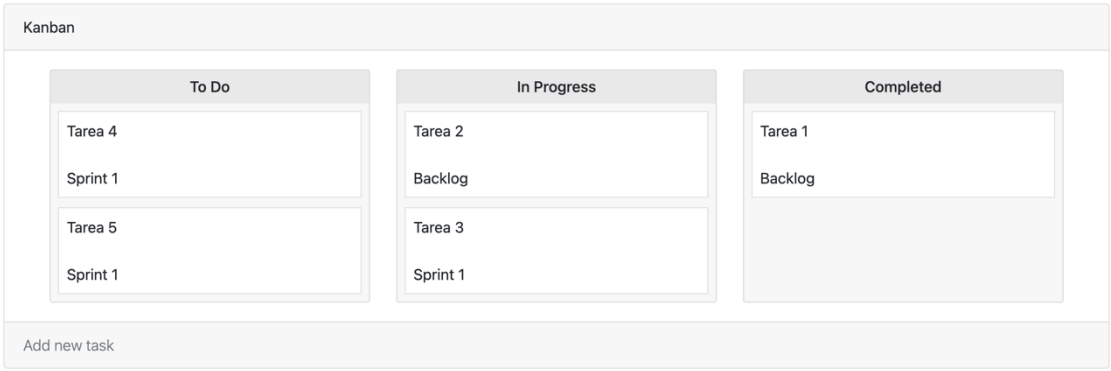

Ilustración 35 - Tablero Kanban

Con el objetivo principal del *Sprint* completado y, teniendo en cuenta que en el *Sprint* anterior se añadió la posibilidad de añadir nuevos participantes a los proyectos. Se incluyó una nueva

estadística que mostraba la cantidad de tareas que realizaba cada integrante del proyecto en cada iteración. Esta gráfica era de tipo radar, y en cada *Sprint* representaba un vértice. Además, se reestructuró los componentes de las estadísticas para mostrarlos por separado en vez de todas juntas. Esto supuso modificar la *sidebar* para que se desplegarán las diferentes opciones. Todos los cambios son visibles en la Ilustración 36, que se muestra a continuación.

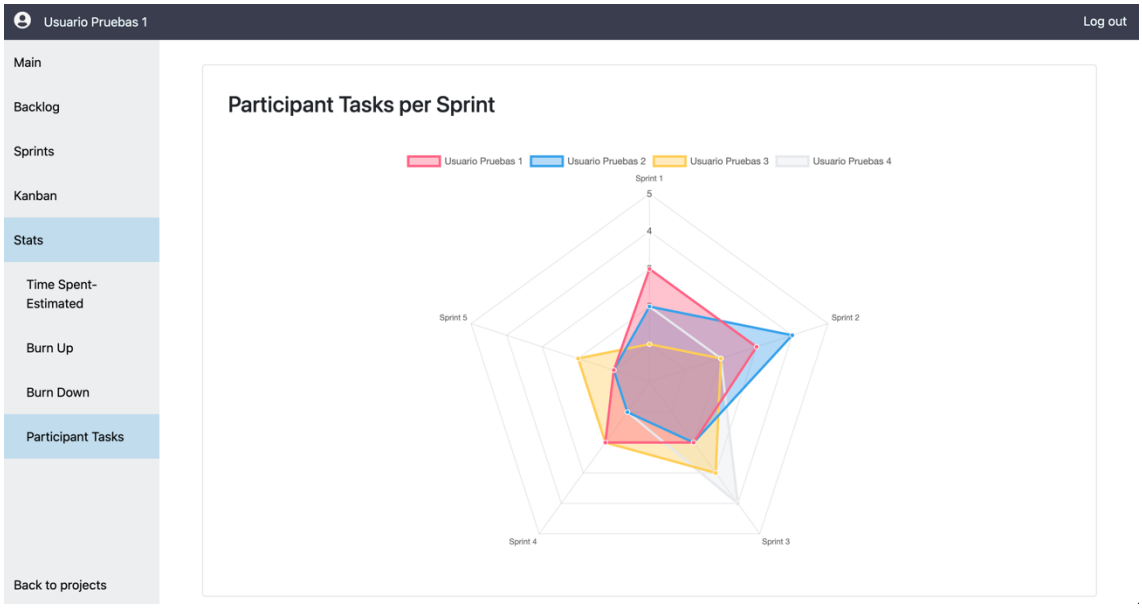

Ilustración 36 - Vista de la herramienta con la gráfica de tareas por participante

La última mejora que se planificó, y con la que se finalizó el *Sprint* fue añadir un método de personalización de los tipos de tareas que hay en el proyecto. Para ello se añadió una lista con los tipos de tareas predeterminados, a estas categorías se les asignó un color. En la lista se incluía la opción de añadir nuevos tipos de tareas, así como un color para representarlas, como se muestra en el modal de la Ilustración 37. También, el usuario será capaz de editar y eliminar los tipos predefinidos o lo que vaya creando. Tanto esta funcionalidad como la de añadir participantes se añadieron a una nueva pestaña, *Settings*, dentro de la *sidebar*.

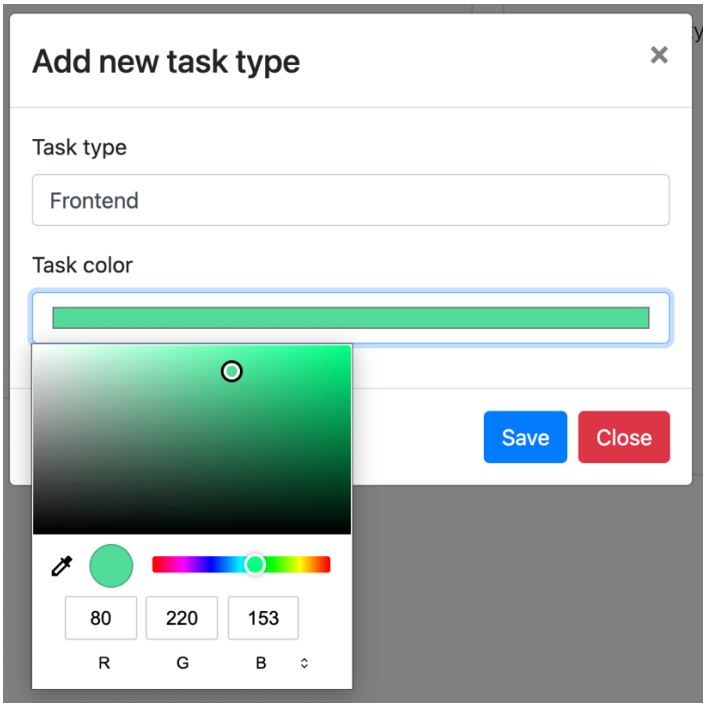

Ilustración 37 - Formulario para añadir tipos de tarea

5.7.3. Tareas Realizadas

Durante este *Sprint* se acometieron estas tareas:

- Completar apartado *Sprints*, añadir fechas y estado de los *Sprints*
- Implementar un tablero *Kanban*
- Creación de nuevos componentes para cada una de las diferentes estadísticas
- Incluir un gráfico que muestra las tareas realizadas por los participantes
- Añadir un método para la personalización de tipos de tareas

5.8. Sprint 8

5.8.1. Planificación Inicial

Como se ha comentado en la planificación inicial del *Sprint* anterior, este *Sprint* se ha diseñado para completarse en tres semanas. El inicio de este *Sprint* es 8 de agosto con la idea de finalizarlo el día 29 de ese mismo mes. En este *Sprint* se acabará el desarrollo de la aplicación y el objetivo es mejorar la forma en la que se muestran las tareas y mejorar visualmente la herramienta. Además, se añadirán nuevos detalles en la funcionalidad del sistema, como poder mover tareas, o añadir comentarios en estas.

5.8.2. Desarrollo del Sprint

Este *Sprint* ha involucrado a prácticamente todos los componentes de la aplicación, ya que el objetivo principal es el de mejorar visualmente la herramienta. Lo primero que se hizo fue añadir iconos para representar las acciones que pueden realizar los usuarios. Para ello se conto con FontAwesome [22], una librería que dispone de forma gratuita una gran cantidad de iconos. Con estos cambios se pretendió aumentar la visibilidad de las acciones, además se añadieron efectos *hover* en los ficheros css, para referenciar aún más las acciones de los usuarios en los elementos de la aplicación. También se añadieron *popovers*, mensajes que se muestran al interactuar con un elemento, para identificar los nuevos iconos.

Se mejoró la forma en la que se muestran las tareas en el *Backlog*, en los *Sprint*, como se puede ver en la Ilustración 38, y en el *Kanban*, como en la Ilustración 39. Al principio solo mostraban el nombre y un botón para eliminar tareas. Así que, para dar más información al usuario, se añadieron iconos con un *popover* que indicaba si esta asignada a un usuario o no. También se añadió el tipo de tareas con una etiqueta del color asignado a esa tarea, y en caso de no tener tipo, se muestra una etiqueta negra. Aprovechando que las tareas permiten estimaciones, y también se indican si están completadas o no, se añadió una barra que muestra el progreso de la tarea. Se añadió un botón que permite mover las tareas entre *Sprints*. Este botón abre un modal que permite al usuario seleccionar a donde mover la tarea.

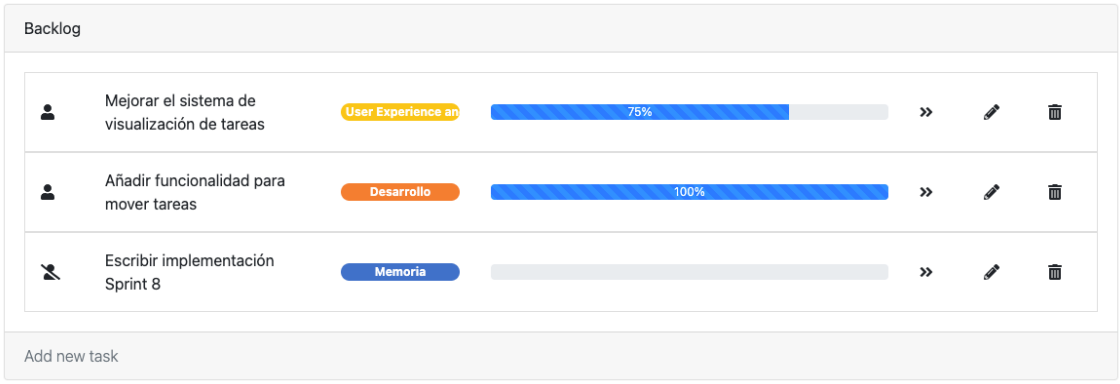

Ilustración 38 – Nueva representación del listado de tareas del Backlog en el Sprint 8

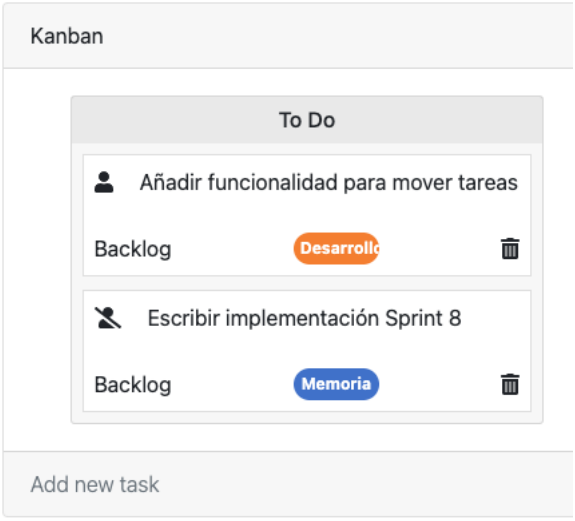

Ilustración 39 - Nueva representación del listado de tareas del Kanban en el Sprint 8

Se cambió la forma en la que se visualiza la información de las tareas, como se puede ver en la Ilustración 40. Ahora en vez de mostrarse en un modal, se muestra toda la información de la tarea al ser clicada. Si no hay un usuario asignado, se puede auto asignar la tarea directamente desde allí. Además, se incluyo una sección de comentarios donde se muestra el nombre, la fecha y el comentario de los usuarios que hayan realizado comentarios. Estos comentarios pueden ser editados por el dueño del comentario.

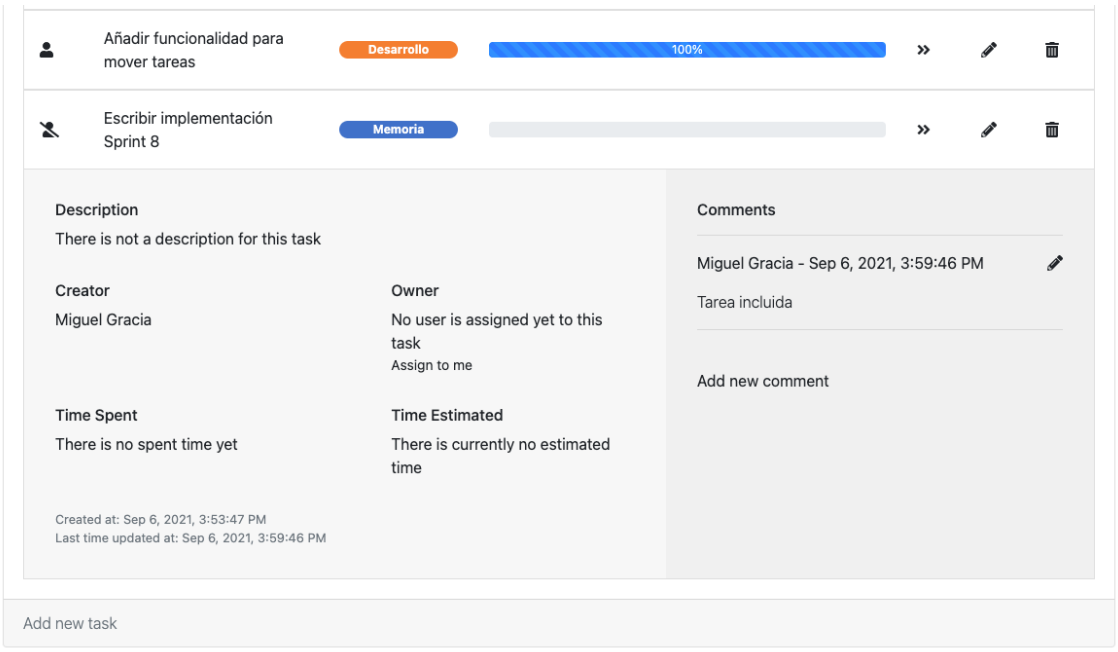

Ilustración 40 - Visualización de la información de las tareas

Relativo al estado del proyecto y a su visibilidad, se añadió el nombre del proyecto seleccionado en la *navbar* cuando se carga un proyecto. Y se añadió la opción de terminar proyectos, que solo la puede utilizar la persona que los ha creado. Esta opción no invalida que se pueda seguir modificando aspectos del proyecto, y eso queda como una posible mejora en un futuro.

Además, se ha incluido una gráfica en forma de pastel que indican como se han distribuido las tareas entre el *Backlog* y los *Sprints*. Toda esta información se encuentra en la pantalla principal del proyecto, dónde también, se contabilizan las horas totales empleadas en las tareas, estos cambios se pueden ver en la Ilustración 41.

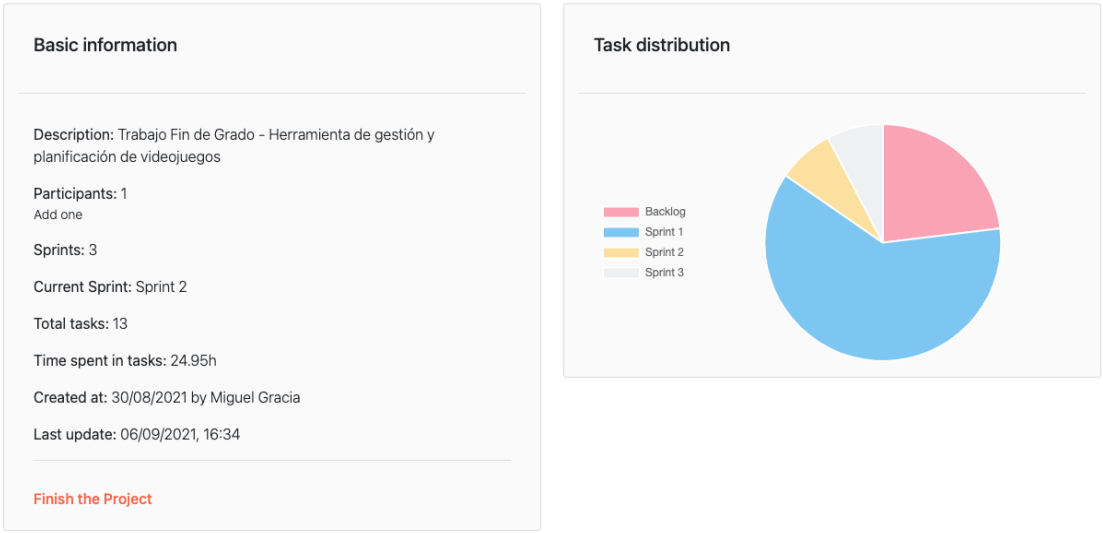

Ilustración 41- Información básica del proyecto con un gráfico de distribución de tareas

5.8.3. Tareas Realizadas

Durante este *Sprint* se acometieron estas tareas:

- Desarrollar un visor de tareas
- Implementar una sección de comentarios para cada tarea
- Añadir la funcionalidad para mover tareas
- Mejorar visualmente la aplicación
- Poder finalizar un proyecto

6. Resultados

6.1. Objetivos completados

Al inicio de este Trabajo de Fin de Grado se marcaron tres objetivos claros que se esperaban completar al final de este proyecto. Estos objetivos eran los siguientes:

6.1.1. Estudio de las herramientas de gestión y planificación de videojuegos, ya existentes en el mercado y sus funcionalidades

Este objetivo fue el primero en cumplirse, se revisaron algunas de las herramientas con más conocidas, como Jira o Trello. Se encontraron herramientas pensadas únicamente en el desarrollo de videojuegos, como HacknPlan.

De estas herramientas se extrajo una lista las características y funcionalidades que se consideraron fundamentales. Se hizo una comparación entre ellas, a la que se le añadió el *software* creado durante este proyecto.

6.1.2. Estudio de las necesidades y funcionalidades requeridas por los usuarios y su implementación en el software, además de las metodologías empleadas para el desarrollo de videojuegos.

Este objetivo se ha ido cumpliendo a lo largo de la producción del *software*. Al principio de este, se hizo una encuesta donde se valoraban aspectos importantes para el desarrollo de una herramienta de planificación y gestión de proyectos/videojuegos. Entre las preguntas se destacaban las necesidades que se buscan en una herramienta, así como las funcionalidades menos necesarias. Como parte de la metodología, se centró en las metodologías *Agile*, ya que la encuesta refleja que el cien por cien de los encuestados las empleaban en su día a día.

Además, se realizaron pruebas de usabilidad con usuarios familiarizados con este tipo de herramientas, para encontrar fallos y mejoras, así como nuevas ideas y funcionalidades que añadir. Como resultado de estas pruebas se añadieron nuevas funciones a la herramienta que no estaban contempladas en el diseño inicial.

6.1.3. Creación de una herramienta software mediante tecnologías que están por determinar (aplicación web, aplicación móvil, etc).

Este objetivo se cumplió con el desarrollo de una aplicación web con Angular, Node.js, Express y Mongo. Esta herramienta es el resultado final de la elaboración de este Trabajo de Fin de Grado, y refleja la importancia de los dos objetivos anteriores. Una aplicación que es sencilla de usar e intuitiva, y que contiene funcionalidades importantes para desarrollar proyectos con metodologías *Agile*. Como el *Backlog*, un tablero *Kanban*, un apartado de estadísticas, o, un método para ver el detalle de los *Sprints* pasados y futuros. Además, permite la personalización de elementos como el tipo de tareas que se van a desarrollar. Todos estos aspectos, permite la libertad a los usuarios de planificar sus proyectos de la forma que mejor se adapte a su estilo de trabajo.

6.2. Resultados obtenidos

En este apartado se van a mostrar las diferentes pantallas de la aplicación, que muestran el estado final de la herramienta al final de la última iteración, además de diagramas sobre el comportamiento de la aplicación y del modelo final de los objetos de la base de datos.

6.2.1. Pantalla final de Login y de Registro de usuarios

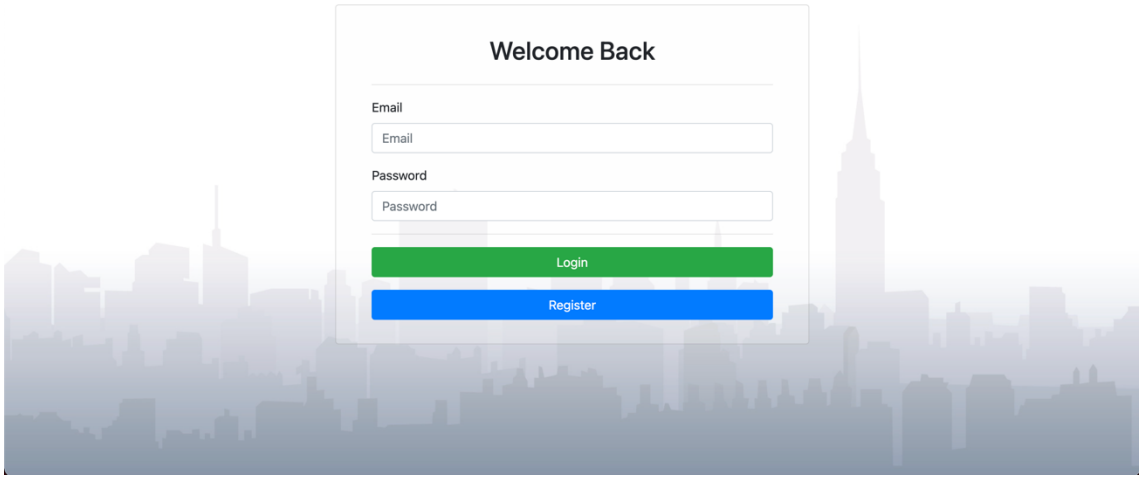

Ilustración 42 - Resultado final de la aplicación (Pantalla de Login)

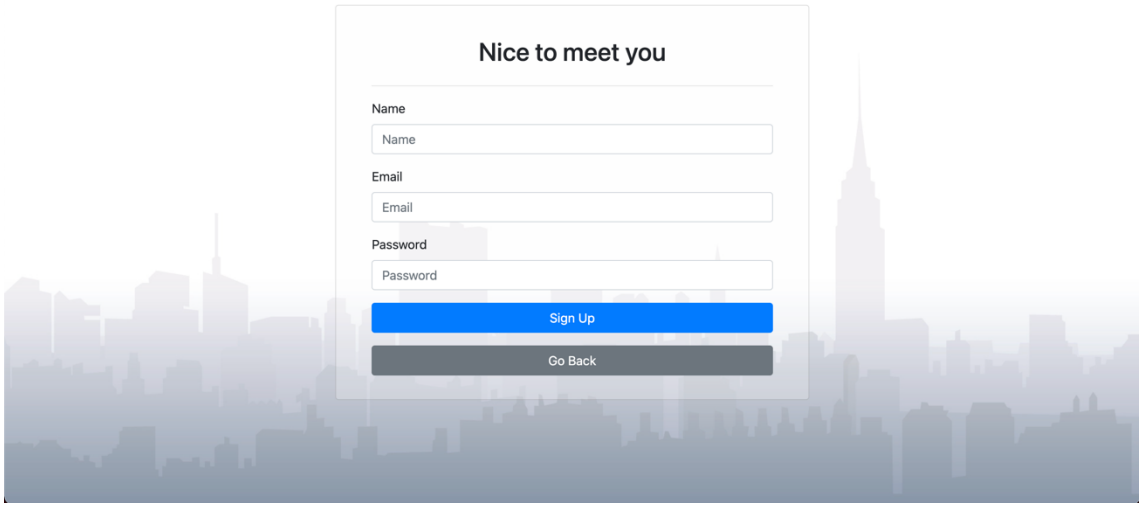

Ilustración 43 - Resultado final de la aplicación (Pantalla de registro de usuarios)

6.2.2. Pantalla final de la selección de proyectos

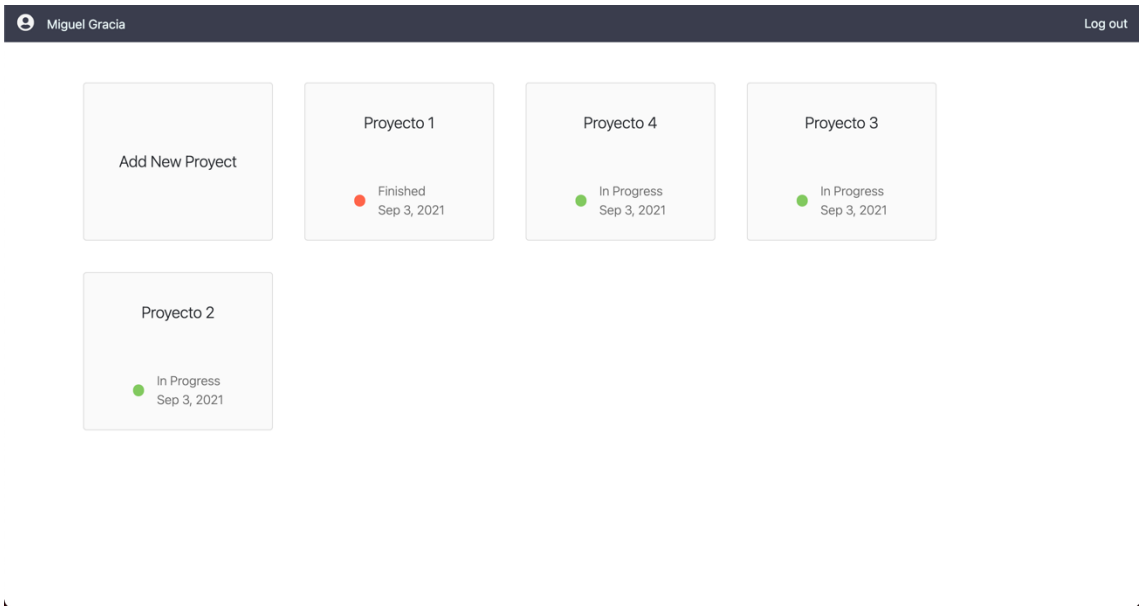

Ilustración 44 - Resultado final de la aplicación (Pantalla de selección de Proyectos)

6.2.3. Resultado de la pantalla principal de un proyecto

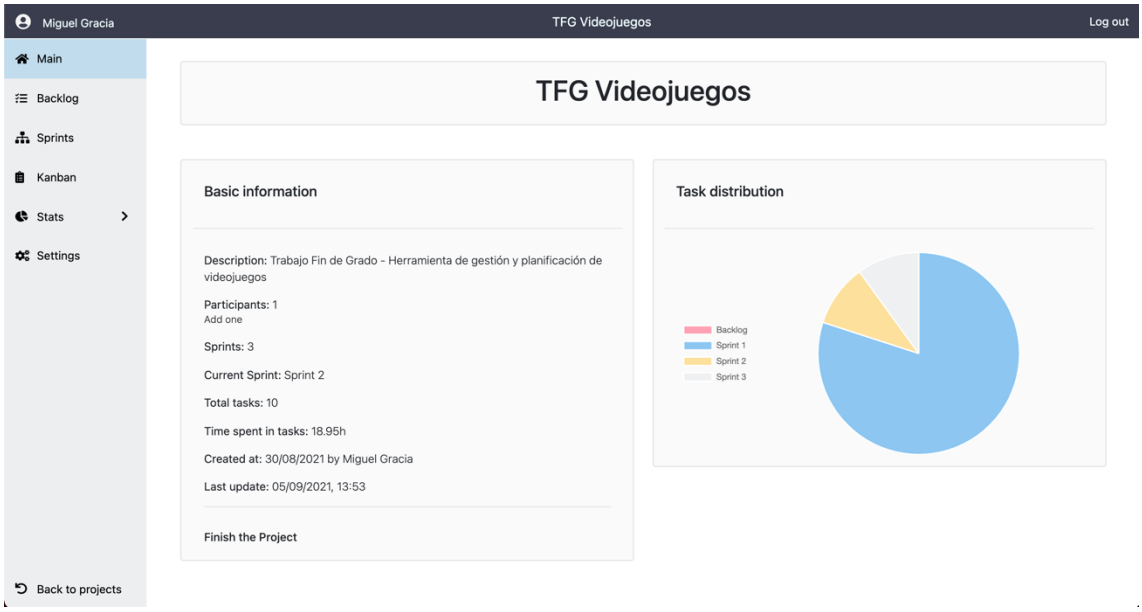

Ilustración 45 - Resultado final de la aplicación (Pantalla de Principal de un Proyecto)

6.2.4. Pantalla final del Backlog, con una tarea abierta

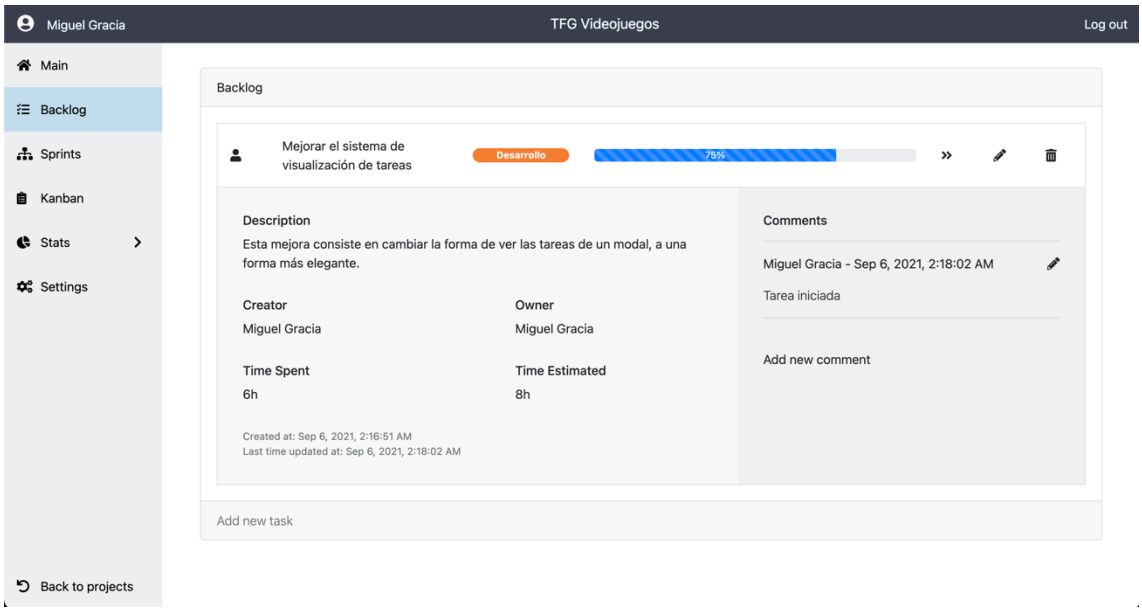

Ilustración 46 - Resultado final de la aplicación (Pantalla del Backlog)

6.2.5. Pantalla de selección de Sprints

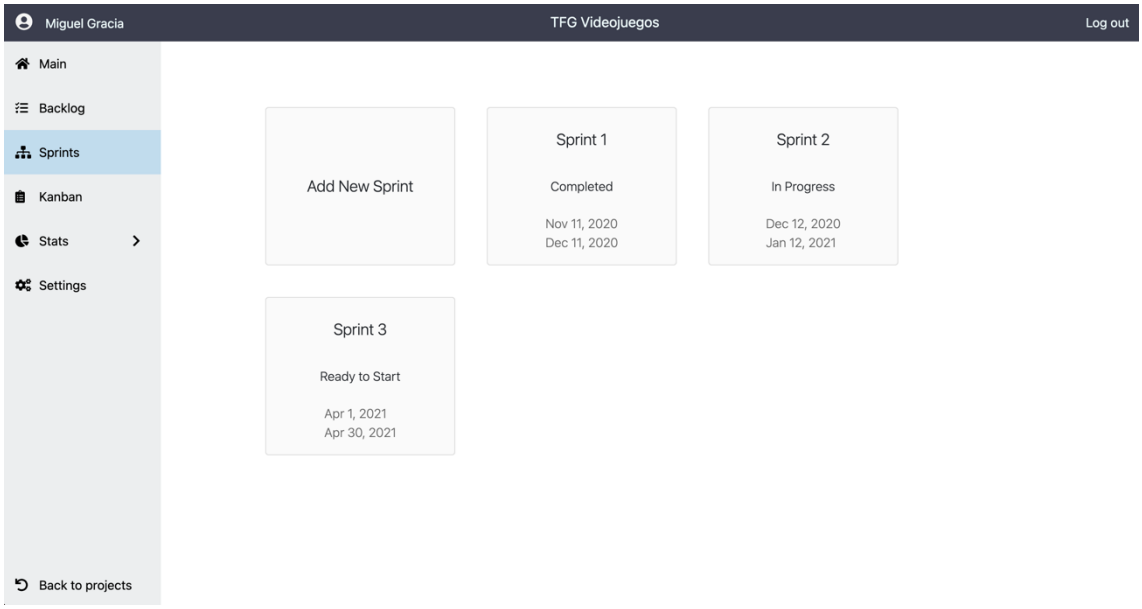

Ilustración 47 - Resultado final de la aplicación (Pantalla de selección de Sprints)

6.2.6. Pantalla final de un Sprint

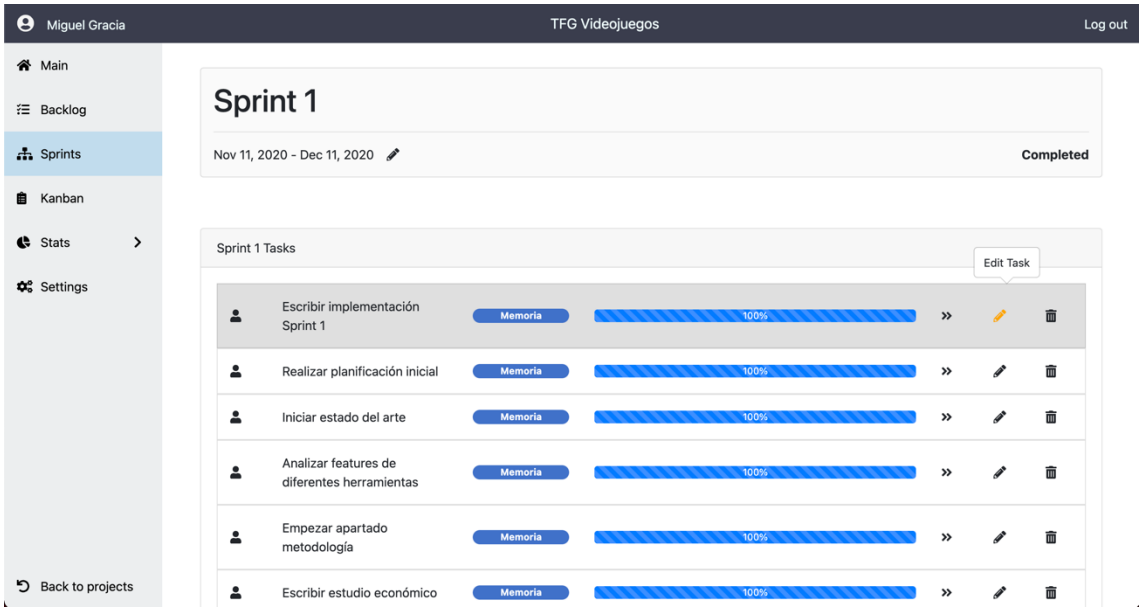

Ilustración 48 - Resultado final de la aplicación (Pantalla del detalle de un Sprint)

6.2.7. Pantalla de Estadísticas de los proyectos

Ilustración 49 - Resultado final de la aplicación (Pantalla de estadísticas)

6.2.8. Pantalla Settings de un proyecto

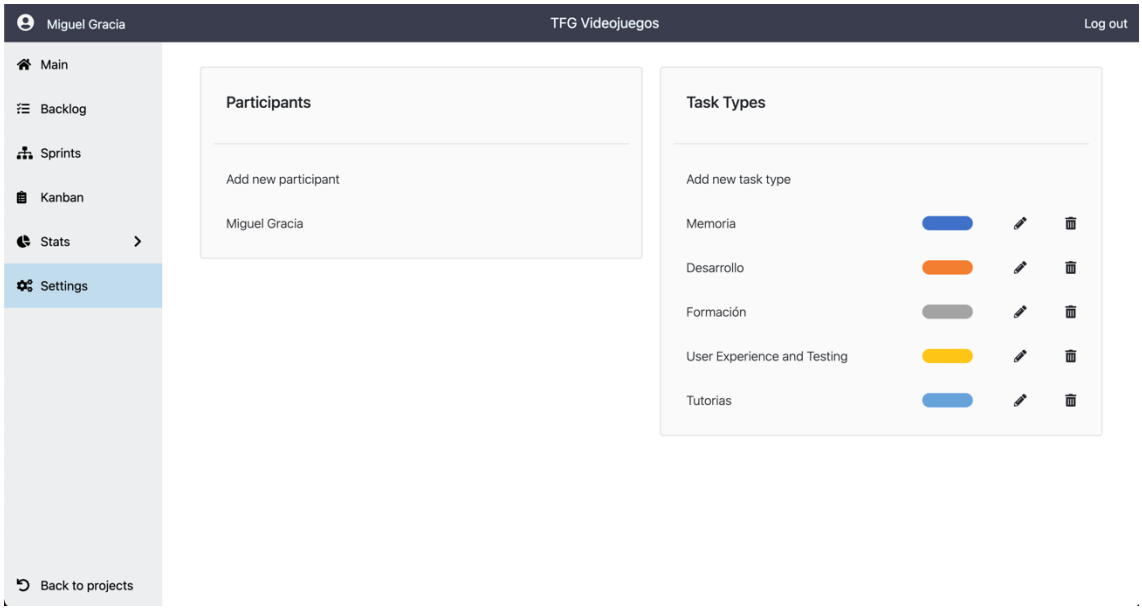

Ilustración 50 - Resultado final de la aplicación (Pantalla de Settings)

6.2.9. Modelo final de los modelos de la base de datos

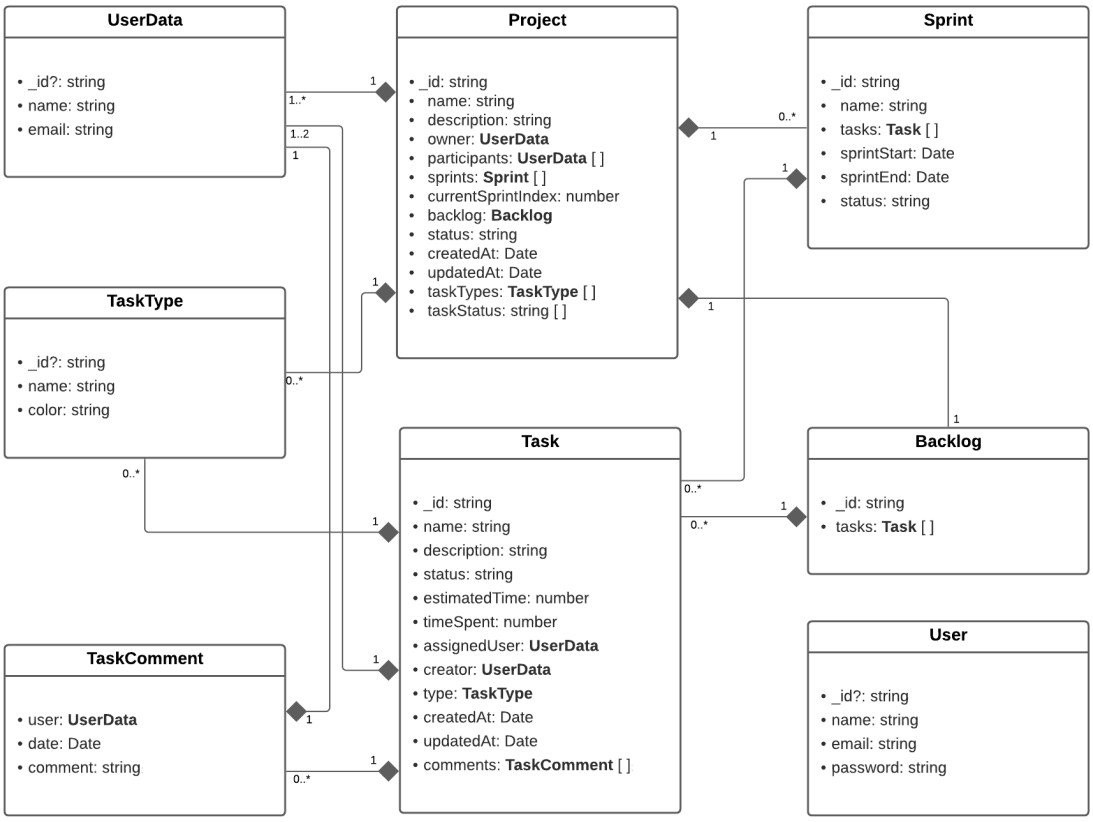

Ilustración 51 - Representación final de los modelos de la base de datos

6.2.10. Modelo final de los componentes de la aplicación

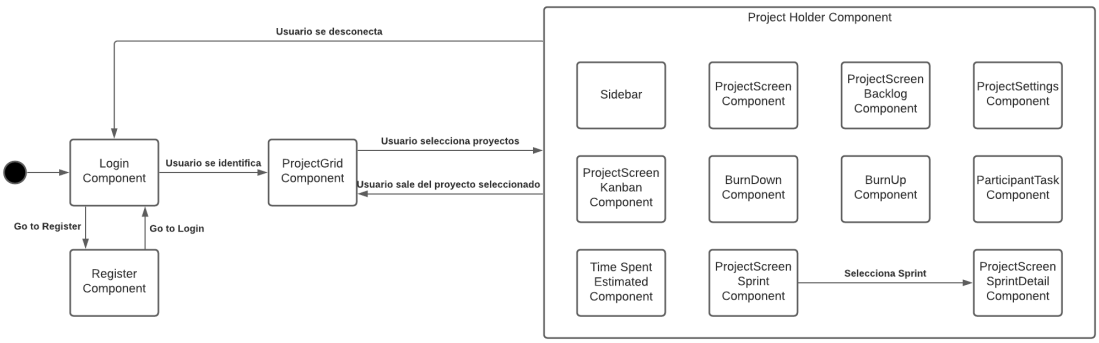

Ilustración 52 - Diagrama de componentes del Frontend

6.3. Resultado de la planificación

Como está detallado en el apartado Metodología sección 4.3 Planificación inicial, el proyecto se estimó en unas 270 horas sin tener en cuenta, funcionalidades que no estaban consideradas en ese momento. Y también se planificó el desarrollo de este Trabajo de Fin de Grado en seis *Sprints*.

La planificación original no se cumplió durante el desarrollo del Trabajo de Fin de Grado, esto se debió en parte a diferentes aspectos. El primero de los problemas fue una mala estimación en algunas de las tareas, ya que se planificaron en una cantidad de horas menor que las realmente necesarias para completar dichas tareas. Esto se agravó al principio en parte a la falta de experiencia trabajando en un *framework* como Angular, aunque al final las tareas de las últimas iteraciones se completaron en el tiempo planificado. Otro problema, por el cual se retrasó la fecha de finalización del proyecto, fue la discontinuidad del trabajo entre el final del segundo *Sprint* y el inicio del tercero. Esto supuso una reestructuración de los *Sprints* que atrasó la fecha de finalización del proyecto, como se puede ver en la Ilustración 53. A pesar de estos problemas, en ningún momento se cambio la metodología de trabajo, solo se modificaron la duración preestablecida en algunos de los *Sprints* pasando a ser de dos semanas, los *Sprints* 4, 5 y 6.

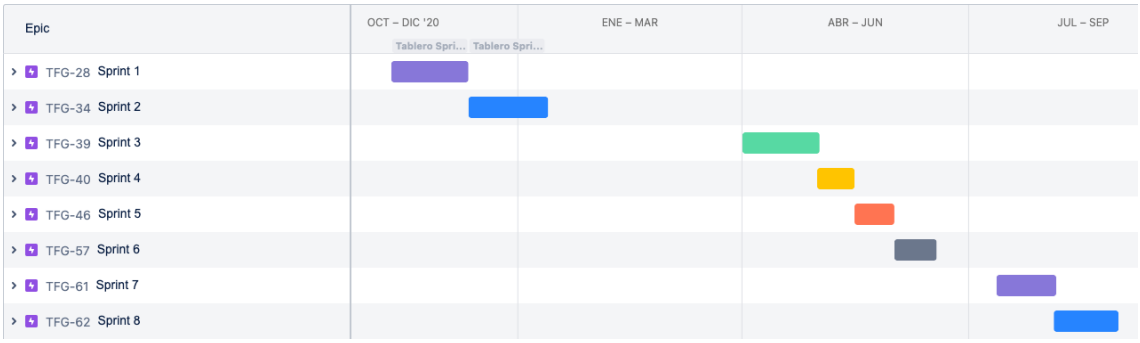

Ilustración 53 - Diagrama de Gantt del resultado de la planificación del proyecto

La estimación inicial de las tareas fue de 274 horas, esta estimación varió con respecto al resultado final. Es cierto que esa estimación no contemplaba algunas tareas que se realizaron durante el proyecto, que se contabilizaron en 15 horas. Teniendo esto en cuenta, hubo una desviación total de 38 horas sobre lo estimado originalmente, por lo que el total de las horas empleadas en el desarrollo del TFG ascendió a 327 horas.

La distribución final de las tareas, como se observa en la Ilustración 54, se repartía en los siguientes apartados:

- Desarrollo del *software*: 192 horas
- *Testing* y *User Experience*: 27 horas y 30 minutos
- Memoria: 70 horas
- Reuniones: 5 horas y 30 minutos
- Formación: 32 horas

Ilustración 54 - Gráfico con la distribución final de las horas empleadas por tipo de tarea

7. Estudio económico

En el estudio económico se va a reflejar todos los aspectos económicos que han intervenido en el proceso de desarrollar este proyecto, especialmente el apartado de costes del proyecto, puesto que la idea original es la de que fuera una herramienta de uso libre que ayude a alumnos y profesionales para planificar sus videojuegos, y no la de obtener un beneficio económico.

7.1. Costes directos

7.1.1. Costes en recursos humanos

En este apartado se tendrán en cuenta los gastos que se han necesitado para el desarrollo de este proyecto. Además de tener en cuenta el gasto que supondría el salario del trabajador que lo ha desarrollado.

En este caso, al solo contar con una única persona para desarrollar este *software*, se debería tener en cuenta el sueldo de un programador con un año de experiencia laboral. El salario medio de un trabajador, en Zaragoza, con estas características ronda los 18.000 euros al año [23]. También hay que contabilizar que a esta cantidad hay que sumarle la cuota de la Seguridad Social que tendría que abonar la empresa que es el 23% del salario.

	Sueldo Base	**Impuestos**	**Total**
Anualmente	18.000	4140	22.140
Mensualmente	1.500	345	1.845

Tabla 2 - Salario medio de un Programador Junior en Zaragoza

Considerando que una jornada laboral son ocho horas de trabajo y que un empleado trabaja veinte días al mes de media cada año, el coste diario por jornada trabajada de este programador sería de unos 92,25 euros para una empresa de este sector.

7.1.2. Costes materiales

En este apartado se va a valorar los gastos que se han requerido en concepto de hardware y *software*.

El material que se ha necesitado ha sido un ordenador portátil y algunos periféricos como teclado, ratón y una pantalla. El ordenador utilizado es un MacBook Pro de 13 pulgadas con 16 GB de

memoria unificada y un almacenamiento de 1 TB de SSD que salió a la venta en noviembre de 2020 con un valor de 2.139 euros.

Obviamente no se va a tener en cuenta el precio final del ordenador como un gasto, sino su amortización en el tiempo que se ha usado para la realización del proyecto. Se ha estimado que la vida útil de este MacBook puede ser de hasta diez años, pero su vida media, tiempo en el que el ordenador se puede utilizar sin que se hayan reducido sus prestaciones, podría llegar a los cinco años.

	Anual	**Mensual**	**Precio**
Amortización	427,80	35,65	2.139

Tabla 3 - Amortización del ordenador utilizado en la elaboración del proyecto

El *software* empleado no ha requerido ningún gasto, ya que todos los programas han sido utilizados con la licencia de estudiante o con la versión gratuita. Los programas que se han empleado han sido: Jira, Visual Studio Code, Microsoft Word, Microsoft Office, Postman, MongoDB Compass.

7.2. Costes totales

7.2.1. Estimación inicial

En un principio se estimó que el proyecto se realizaría en seis meses con un trabajo acumulado de 274 horas, unas 35 jornadas de trabajo. Teniendo esto en cuenta y los datos recogidos en las secciones anteriores podemos concluir que el presupuesto inicial del proyecto estaría entre 3.219,50 euros y 3.291,05 euros, dependiendo si se estima en horas trabajadas o en días.

	Precio por Hora	**Precio por Día**
Gasto en Recursos Humanos	11,53	92,25
Gasto en Materiales	0,22	1,78
Total	**11,75**	**94,03**

Tabla 4 - Relación de gastos por horas y días

	Estimación por Horas	**Estimación en Días**
Precio	11,75	94,03
Tiempo	274	35
Total	**3.219,50**	**3.291,05**

Tabla 5 - Estimación inicial del coste del proyecto

7.2.2. Presupuesto final

Como se ha indicado en el capítulo anterior, el desarrollo de este Trabajo de Fin de Grado, que incluye tanto el *software* como la memoria, se ha contabilizado en un total de 327 horas. A continuación, se mostrará una tabla con la relación del presupuesto final teniendo en cuenta si fuera un contrato en horas o por días.

	Presupuesto en Horas	**Presupuesto en Días**
Precio	11,75	94,03
Tiempo	327	41
Total	**3.842,25**	**3.855,23**

Tabla 6 - Presupuesto final basado en horas y en jornadas de trabajo

Como se puede observar, la desviación entre lo estimado y el gasto final es de entre 564,18 euros y 622,75 euros. Este es un detalle que destacar ya que, en este caso, esta estimación inicial hubiera conllevado unas pérdidas de cerca de un 15%. Esto es un ejemplo que demuestra la gran importancia de realizar buenas estimaciones y de contar con herramientas capaces de evitar estas situaciones.

A pesar de está desviación, el coste del proyecto es un precio razonable comparándolo con el coste de las herramientas estudiadas en el capítulo 2, Estado del arte. Por ejemplo, Jira tiene una tabla precios establecido dependiendo del número de usuarios, donde una organización debería de pagar 7000 dólares al año por la licencia *standard* para 100 usuarios [24]. Teniendo este dato en cuenta, una universidad que quisiera financiar esta herramienta le saldría bastante rentable sabiendo que un grado como Diseño y Desarrollo de Videojuegos, puede tener entre 80 y 100 alumnos entres sus cuatro cursos.

8. Conclusiones

8.1. Puntos de mejora

Como se ha comentado previamente, este tipo de herramientas son bastante escalables, por lo que en un futuro se pueden ir añadiendo nuevas funcionalidades. Pero desde el punto de vista actual y del estado de la aplicación, los pasos a seguir en los futuros *Sprints* para mejorar la herramienta son los siguientes:

8.1.1. Mejoras en la interfaz

Sería conveniente mejorar la interfaz gráfica de la aplicación, haciendo que los componentes sean visualmente más agradables. Uno de los aspectos que más pueden atraer es conseguir que la aplicación sea más atractiva para el usuario final, esto se podría conseguir añadiendo animaciones en los componentes.

Otro aspecto interesante que se añadiría en esta mejora es permitir más personalización dentro de la herramienta, como añadir un modo oscuro o disponer de plantillas predeterminadas, para que los usuarios puedan decidir como quieren que se visualice la herramienta.

También, pensando en las personas con alguna discapacidad visual, se podría adaptar la herramienta para facilitarles la interacción con ella. Para ello, sería necesario crear una configuración que permita adaptar la interfaz para personas con los diferentes tipos de daltonismo.

8.1.2. Montar la aplicación en un servidor

Ahora mismo la aplicación funciona solamente en local, ejecutándose en nuestros propios ordenadores. Por lo que uno de los pasos más importantes para el desarrollo de esta herramienta sería alojarla en un servidor externo, ya sea en plataformas como Heroku o en servidores *cloud*, como AWS o Google Cloud. Para poder llevar a cabo está mejora habría que realizar un estudio económico previo para elegir la mejor opción, ya que montar la aplicación en un servidor externo conlleva un gasto económico. De ahí la opción de Heroku, que permite subir las aplicaciones sin gasto alguno con su tarifa Free [25], aunque se tendrían algunas limitaciones.

8.1.3. Crear un sistema de envío de correos

Está opción permitiría ampliar los procesos de verificación en los registros a nuestra aplicación, así como dar más visibilidad a nuestros proyectos. Con este sistema podríamos añadir un sistema de reinicio de contraseñas, así como mejoraría el proceso de añadir usuarios a los proyectos, ya que ahora mismo tienes que estar registrado para poder acceder. Tras está mejora, este proceso mandaría un enlace al correo del usuario invitado para que acepte la invitación al proyecto, y en caso de no estar registrado que se pueda registrar en la plataforma.

Otra posibilidad que surgiría gracias a esta mejora sería un sistema de notificaciones por correo para avisar de los avances o modificaciones dentro del proyecto. La idea sería mandar correos a cada participante del proyecto cada vez que se haya creado una tarea nueva, o se haya finalizado alguna o para avisar del inicio o fin del *Sprint*. Además, se podría incluir un sistema que se controlase en cada proyecto que tipo de notificaciones se quieren recibir.

8.1.4. Visualizar cambios en tiempo real

Para conseguir está mejora se necesitaría implementar una comunicación bidireccional con sockets entre el *frontend* y el *backend*, esto permitiría ver los cambios en la plataforma en tiempo real. Actualmente hasta que no se llama al servicio no se pueden ver los cambios, esto quiere decir, por ejemplo, si dos usuarios de un proyecto están en la misma pestaña de tareas y uno de ellos añade una, el cambio solo es visible para quien la haya añadido, mientras que el otro usuario solo verá el cambio cuando se recargue la pagina. Con una comunicación con sockets, usando por ejemplo la librería Socket.io [26], se podría visualizar los cambios en tiempo real.

8.1.5. Crear un calendario para poder visualizar el proyecto

Esta sería una propuesta de una nueva funcionalidad del proyecto, esta mejora consistiría en añadir un calendario del proyecto. Se tendría que añadir un nuevo botón en el *sidebar* que mostrase este nuevo componente. En este calendario se mostrarían las fechas de los *Sprints* y permitiría crear nuevas funciones en esta herramienta como la de poder crear eventos. Estos eventos serían un listado con las etapas de un proyecto *Agile*, como pueden ser las *Daily Meetings*, los *Sprint Reviews* o cualquier otro tipo de evento. Además, se podría añadir la posibilidad de añadir eventos personalizados para cada proyecto fuera de los preestablecidos por el sistema.

8.1.6. Definir test cases y documentar la API

Para añadir más calidad al proyecto sería bueno incluir pruebas de uso para cada componente y servicio del *frontend*, así como para *backend*. Con esto se conseguiría crear una batería de pruebas que verificaran el correcto funcionamiento de nuestro proyecto, y aseguraría que los cambios que se realicen tanto para el mantenimiento de la aplicación, como para las nuevas funcionalidades que se añadan no cambien el comportamiento de la herramienta.

Con el mismo objetivo de hacer el proyecto más profesional, se debería de tener un documento de referencia para la API. Para completar este propósito, se necesitaría diseñar la API con Swagger [27], un *framework* que genera un documento para definir los servicios de un proyecto.

8.2. Opinión sobre el Trabajo de Fin de Grado

Realizar este Trabajo de Fin de Grado ha resultado ser una gran experiencia en muchos aspectos, ya que durante este tiempo he sido capaz de desarrollarme en una nueva tecnología. Este proyecto ha sido un reto que me ha encantado afrontar y que me encantaría retomar en un futuro.

Ahora, viendo todo proyecto en perspectiva, estoy muy orgulloso de ver todo lo que he aprendido y ver como las tareas que al principio me costaban más tiempo, ahora soy capaz de terminarlas con mejor calidad que al principio y en menos tiempo. Además, gracias a este proyecto he aprendido a manejar con mejor soltura la maquetación web, que era uno de mis puntos débiles.

Pero no todo ha sido fácil, compaginar un TFG y al mismo tiempo trabajar con jornada completa en una consultoría tecnológica ha sido realmente complicado, y aún más con las circunstancias que estamos viviendo por culpa de la pandemia. Esta situación de encontrarme todos los días en mi habitación trabajando todo el día, me ha generado mucho estrés. Este fue uno de los motivos por los que desde mediados de enero hasta abril no trabajé en el proyecto, y que me llevo a tomar la decisión de reducir mi jornada laboral.

A pesar de esta situación, siempre he confiado en que el proyecto saldría adelante. De este Trabajo de Fin de Grado no solo he aprendido una tecnología nueva, sino que también he aprendido a gestionar mejor mi tiempo y planificar mejor mi trabajo.

9. Bibliografía

[1] Mark Cruth, "Discover the Spotify model." https://www.atlassian.com/agile/agile-at-scale/spotify

[2] Digital.ai, "The 14th annual State of Agile report." [Online]. Available: https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494

[3] Trello, "Trello webpage." https://trello.com/about

[4] Trello, "Trello plan standard." https://trello.com/standard

[5] Trello, "Trello plan premium." https://trello.com/premium

[6] Atlassian, "What is Jira used for?" https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for#Jira-for-requirements-&-test-case-management

[7] Asana.com, "Asana." https://asana.com/es

[8] Hacknplan.com, "HacknPlan Press Kit." https://hacknplan.com/press/

[9] Hacknplan.com, "HacknPlan Features." https://hacknplan.com/#features

[10] Maximilian Schwarzmüller, "Angular & NodeJS - The MEAN Stack Guide [2021 Edition]." [Online]. Available: https://www.udemy.com/course/angular-2-and-nodejs-the-practical-guide/

[11] Maximilian Schwarzmüller, "Angular - The Complete Guide (2021 Edition)." [Online]. Available: https://www.udemy.com/course/the-complete-guide-to-angular-2/

[12] Mongoose, "Documentación de Mongoose ." https://mongoosejs.com/docs/index.html

[13] Bootstrap team, "Página web oficial de Bootstrap." https://getbootstrap.com/

[14] OpenJS Foundation, "Página oficial de jQuery." https://jquery.com/

[15] Popperjs, "Página web oficial de Popper.js", [Online]. Available: https://popper.js.org/

[16] Auth0, "Documentación JSON Web Token", [Online]. Available: https://jwt.io/introduction

[17] Ng-Bootstrap team, "Página oficial de Ng-Bootstrap." https://ng-bootstrap.github.io/#/home

[18] Valor-software.com, "Página oficial de la libreria ng2-charts." https://valor-software.com/ng2-charts/#/GeneralInfo

[19] Chart.js contributors, "Página oficial de la librería Chart.js." https://www.chartjs.org/

[20] ObaidUrRehman, "Angular Drag & Drop." https://www.npmjs.com/package/ng-drag-drop

[21] Google, "Drag and Drop Angular Material CDK." https://material.angular.io/cdk/drag-drop/overview

[22] Fontawesome, "FontAwesome.com." https://fontawesome.com

[23] Indeed.com, "¿Cuánto se gana en España de Programador/a junior?" https://es.indeed.com/career/programador-junior/salaries?from=top_sb

[24] Atlassian, "¿Cuánto cuesta una suscripción a Jira Software Cloud Standard?" https://www.atlassian.com/es/licensing/jira-software

[25] Heroku.com, "Heroku Pricing." https://www.heroku.com/pricing

[26] Socket.io, "Página oficial de Socket.io ." https://socket.io/

[27] SmartBear Software, "Página web oficial de Swagger.io." https://swagger.io/

10. Anexo

10.1. Propuesta

Nombre alumno: Miguel Gracia Carballo

Titulación: Grado en Diseño y desarrollo de videojuegos

Curso académico: 2020/2021

1. TÍTULO DEL PROYECTO

Herramienta de gestión y planificación de videojuegos

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

El proyecto consiste en un análisis de las diferentes funcionalidades que existen actualmente en herramientas de gestión de videojuegos, e implementar las características fundamentales en una herramienta de gestión de videojuegos.

Esta herramienta estará centrada en alumnos o profesionales del ámbito de los videojuegos y del ámbito profesional, con poca experiencia en el uso de este tipo de aplicaciones.

3. OBJETIVOS DEL PROYECTO

- Estudio de las herramientas de gestión y planificación de videojuegos, ya existentes en el mercado y sus funcionalidades.
- Estudio de las necesidades y funcionalidades requeridas por los usuarios y su implementación en el *software*, además de las metodologías empleadas para el desarrollo de videojuegos.
- Creación de una herramienta *software* mediante tecnologías que están por determinar (aplicación web, aplicación móvil, etc).

4. METODOLOGÍA

Se definirá en las primeras reuniones con el tutor.

5. PLANIFICACIÓN DE TAREAS

La planificación se definirá al inicio del proyecto en las primeras reuniones con el tutor.

6. OBSERVACIONES ADICIONALES

10.2. Acta de reuniones

10.2.1. Primera reunión

REUNIÓN:	**Primera Reunión**

Fecha:	21 de octubre del 2020		
Hora comienzo:	18:00	**Hora finalización:**	18:30
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	Asunto	Acuerdo
1	Presentación de la idea del proyecto y la metodología de trabajo	-
2	Establecer los primeros pasos a realizar	-
3	Explicación del estado del arte y recomendaciones a seguir	001
4	Recomendaciones sobre la planificación del proyecto	002
5	Preparar el acta de reunión	003

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Realizar el estado del arte del proyecto	Siguiente reunión	Miguel Gracia Carballo
002	Preparar una planificación inicial del proyecto	Siguiente reunión	Miguel Gracia Carballo
003	Rellenar la primera acta de reunión	Inmediato	Miguel Gracia Carballo

10.2.2. Segunda reunión

REUNIÓN:	**Segunda Reunión**

Fecha:	14 de diciembre del 2020		
Hora comienzo:	18:00	**Hora finalización:**	18:30
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Revisión de los avances realizados en el *Sprint* anterior	-
2	Comprobación de los objetivos propuestos en el *Sprint* anterior	-
3	Definición de los objetivos del nuevo *Sprint*	001
4	Preparar el acta de reunión	002

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Elegir la tecnología con la que se implementará la herramienta	Siguiente reunión	Miguel Gracia Carballo
001	Elaborar una encuesta para conocer las necesidades de los usuarios	Siguiente reunión	Miguel Gracia Carballo
001	Realizar la formación necesaria para poder empezar el desarrollo de la herramienta	Siguiente reunión	Miguel Gracia Carballo
002	Preparar el acta de reunión	Inmediato	Miguel Gracia Carballo

10.2.3. Tercera reunión

REUNIÓN:	**Tercera Reunión**

Fecha:	9 de febrero del 2021		
Hora comienzo:	18:00	**Hora finalización:**	18:35
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Revisión de los avances realizados en el *Sprint* anterior	-
2	Comprobación de los objetivos propuestos en el *Sprint* anterior	-
3	Análisis de los fallos cometidos	-
4	Definición de los objetivos del nuevo *Sprint*	001
5	Preparar el acta de reunión	002

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Enviar la memoria para su corrección	Siguiente reunión	Miguel Gracia Carballo
001	Preparar encuesta que no se realizó en el *Sprint* anterior	Próximas semanas	Miguel Gracia Carballo
001	Montar entorno de trabajo, y conectar *backend* y *frontend*	Siguiente reunión	Miguel Gracia Carballo
002	Preparar el acta de reunión	Inmediato	Miguel Gracia Carballo

10.2.4. Cuarta reunión

REUNIÓN:	**Cuarta Reunión**

Fecha:	30 de abril del 2021		
Hora comienzo:	18:00	**Hora finalización:**	18:40
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	Asunto	Acuerdo
1	Revisión de los avances realizados en el *Sprint* anterior	-
2	Comprobación de los objetivos propuestos en el *Sprint* anterior	-
3	Definición de los objetivos del nuevo *Sprint*	001
4	Preparar el acta de reunión	002

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Mandar análisis de los resultados obtenidos en la encuesta realizada	Dos días	Miguel Gracia Carballo
001	Crear un sistema de *login* y de registro de usuarios	Siguiente reunión	Miguel Gracia Carballo
001	Crear un sistema para crear, guardar y cargar proyectos	Siguiente reunión	Miguel Gracia Carballo
002	Preparar el acta de reunión	Inmediato	Miguel Gracia Carballo

10.2.5. Quinta reunión

REUNIÓN:	**Quinta Reunión**

Fecha:	18 de mayo del 2021		
Hora comienzo:	18:30	**Hora finalización:**	19:15
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Revisión de los avances realizados en el *Sprint* anterior	-
2	Comprobación de los objetivos propuestos en el *Sprint* anterior	-
3	Definición de los objetivos del nuevo *Sprint*	001
4	Preparar el acta de reunión	002

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Añadir la funcionalidad del *Backlog* al sistema	Siguiente reunión	Miguel Gracia Carballo
001	Crear un sistema para añadir *Sprints*	Siguiente reunión	Miguel Gracia Carballo
001	Montar un sistema de creado de tareas	Siguiente reunión	Miguel Gracia Carballo
002	Preparar el acta de reunión	Inmediato	Miguel Gracia Carballo

10.2.6. Sexta reunión

REUNIÓN:	**Sexta Reunión**

Fecha:	3 de junio del 2021		
Hora comienzo:	19:00	**Hora finalización:**	19:20
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Revisión de los avances realizados en el *Sprint* anterior	-
2	Comprobación de los objetivos propuestos en el *Sprint* anterior	-
3	Definición de los objetivos del nuevo *Sprint*	001
4	Preparar el acta de reunión	002

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Avanzar el desarrollo de la memoria	Siguiente reunión	Miguel Gracia Carballo
001	Añadir el apartado de estadísticas	Siguiente reunión	Miguel Gracia Carballo
002	Preparar el acta de reunión	Inmediato	Miguel Gracia Carballo

10.2.7. Séptima reunión

REUNIÓN:	**Séptima Reunión**

Fecha:	25 de junio de 2021		
Hora comienzo:	19:00	**Hora finalización:**	20:00
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Revisión de los avances realizados en el *Sprint* anterior	-
2	Comprobación de los objetivos propuestos en el *Sprint* anterior	-
3	Definición de los objetivos de los próximos dos *Sprints*	001
4	Preparar el acta de reunión	002

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Crear el tablero *Kanban*	Siguiente reunión	Miguel Gracia Carballo
001	Realizar mejoras visuales en la herramienta	Siguiente reunión	Miguel Gracia Carballo
001	Mejorar sistema de visualización de tareas	Siguiente reunión	Miguel Gracia Carballo
001	Escribir más apartados de la memoria	Siguiente reunión	Miguel Gracia Carballo
002	Preparar el acta de reunión	Inmediato	Miguel Gracia Carballo

10.2.8. Octava reunión

REUNIÓN:	**Octava Reunión**

Fecha:	30 de agosto de 2021		
Hora comienzo:	17:00	**Hora finalización:**	17:50
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Revisión final del proyecto	-
2	Acordar plazos para la memoria	001
3	Preparar el acta de reunión	002

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Terminar los últimos detalles de la memoria	Siguiente reunión	Miguel Gracia Carballo
002	Preparar el acta de reunión	Inmediato	Miguel Gracia Carballo

10.2.9. Novena reunión

REUNIÓN:	**Novena Reunión**

Fecha:	7 de septiembre de 2021		
Hora comienzo:	18:30	**Hora finalización:**	18:50
Lugar:	En remoto a través de Teams		
Elabora acta:	Miguel Gracia Carballo		
Convocados:	Jorge Echeverría Ochoa y Miguel Gracia Carballo		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Revisión de la memoria	001
2	Preparar el acta de reunión	002

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Realizar últimas correcciones y subir toda la información a la PDU	Siguiente reunión	Miguel Gracia Carballo
002	Preparar el acta de reunión	Inmediato	Miguel Gracia Carballo

10.3. Resultados de la encuesta

¿Utilizas metodologías Agile en la creación de vuestros proyectos/videojuegos?
Sí
Sí
Sí
Sí
Sí
Sí
Sí
Sí
Sí
Sí
Sí
Sí
Sí
Sí

Tabla 7 - Resultados de la primera pregunta de la encuesta

¿Qué herramientas usas para la planificación de vuestros proyectos/videojuegos?
Trello
Trello
Jira, Trello, Redmine, kanbanflow
Trello
Jira, Bitbucket, jenkins.
Jira, Trello, Hacknplan
Jira, No uso ningún *software*
Jira
Jira, Trello, Todoist
Trello

Jira, Trello
Jira, Trello
Jira, Trello, Miro, discord y mucho papel y post its
Hacknplan
Trello, No uso ningún *software*

Tabla 8 - Resultados de la segunda pregunta de la encuesta

¿Qué características te parecen fundamentales para elegir una herramienta sobre otra?
Es gratuita, Organización por *Sprints*, Sencilla
Tablero *Kanban*, Sencilla
Es gratuita, Organización por *Sprints*, Tablero *Kanban*, Estadísticas sobre el proyecto, Diagramas de Gantt, Gestión de tiempo, Sencilla
Estadísticas sobre el proyecto, Disponer de *Backlog*, Sencilla
Organización por *Sprints*, Tablero *Kanban*, Estadísticas sobre el proyecto, Disponer de *Backlog*, Sencilla
Es gratuita, Organización por *Sprints*, Disponer de *Backlog*, Estar especializada en videojuegos, Gestión de tiempo, Sencilla
Es gratuita, Gestión de tiempo, Sencilla
Es gratuita, Organización por *Sprints*, Tablero *Kanban*, Disponer de *Backlog*, Diagramas de Gantt, Gestión de tiempo, Sencilla
Es gratuita, Disponer de *Backlog*, Gestión de tiempo, Sencilla
Es gratuita, Organización por *Sprints*, Tablero *Kanban*, Disponer de *Backlog*
Es gratuita, Organización por *Sprints*, Tablero *Kanban*, Disponer de Backlog, Diagramas de Gantt, Gestión de tiempo, Sencilla
Es gratuita, Organización por *Sprints*, Disponer de *Backlog*, Sencilla
Es gratuita, Organización por *Sprints*, Estadísticas sobre el proyecto, Disponer de *Backlog*, Gestión de tiempo, Sencilla
Es gratuita, Organización por *Sprints*, Sencilla
Es gratuita, Organización por *Sprints*, Tablero *Kanban*, Diagramas de Gantt

Tabla 9 - Resultados de la tercera pregunta de la encuesta

Si pudieras elegir una característica o una funcionalidad que te gustase tener en una herramienta de planificación, ¿cuál sería?
Ordenar por tarjetas y *feedback* visual
Fusionar/vincular/deshacer vínculos entre tarjetas de forma visualmente sencilla
Poder poner colores como en los *possits* y *tags* para diferenciar el área.
Más inteligencia para explotar información de proyectos pasados
Con las que ya existen va bien la verdad
Versatilidad a la hora de cambiar de plan/ *drag and drop* de tareas
Mejor visualización de tareas completadas y por hacer (con grafica o croquis)
Planificación del *Sprint*
Sincronización con gitlab o servidor git
Votaciones en comun sobre los *user story points* que se le dan a una *user story.*
Facilidad de realizar varios proyectos simultáneos.
Que venga con herramientas integradas como, por ejemplo, *agile poker* para las sesiones de *Refinement*
Gráfico *Burn up, Backlog*, pila de *sprint*, tablero *kanban* y *burn down*, indispensables para mi. Añadiría una gran herramienta de estimación de duración de tareas
La organización por *sprints* (tablas en hack'n'plan a mi me resulta fundamental). El poder dividir las tareas en tipos ya definidos (arte, diseño, etc.) es un plus.
Que sea colaborativa, intuitiva y que puedas ver de un vistazo la información de como va el proyecto.

Tabla 10 - Resultados de la cuarta pregunta de la encuesta

¿Qué funcionalidades te parecen irrelevantes para una herramienta de planificación?
Obligar el uso de funcionalidades, como forzar a usar un diagrama de gantt o que todo deba de surgir de un *backlog* inicial.
Ninguna, todas son bienvenidas si es posible deshabilitarlas selectivamente.
No lo se
Integración con otras herramientas (siempre fallan conforme evolucionan las versiones)
Todas tienen su función, pero hay algunas que se adaptan a unos perfiles. No necesita la misma info un desarrollador, que un *tester* que un *scrum master.*

Creacion de perfiles personales
Pienso que en un futuro podré usarlos, pero tampoco se me ocurre ninguna.
Ninguna
Ninguna, todas son bienvenidas
No tengo exp suficiente como para saberlo.
No sé
Opción de videollamada o compartir pantalla
Lo que no sea lo de arriba...
N/A
Hay algunas que incorporan chat o videollamadas, me parece lo menos relevante ya que ahora mismo hay muchas opciones para comunicarse y no veo necesaria otra más. No veo mala idea que integren esas herramientas, pero si que las veo menos necesarias que otras cosas.

Tabla 11 - Resultados de la quinta pregunta de la encuesta

¿Conoces alguna funcionalidad que disponga tu herramienta, pero no uses en tus proyectos?
Actualmente usamos casi la totalidad de la herramienta, no nos sobra ninguna funcionalidad (trello).
Visualización de tableros en forma de hoja de cálculo.
Casi todas
Chat
Muchas, pero como no las conozco tampoco sabría decirte cuáles.
Inclusion de imagenes y audios
Unos cuantos, pero tampoco me suelo fijar
Graficas
Como no la uso no la conozco
No tengo exp suficiente.
No sé
Sinceramente, creo que lo uso todo 😅. Si hay algo que no uso, es porque no sé ni que existe.
medidor de tiempo

Seguro que hay, todo lo relacionado con el modelo de diseño lo estamos infrautilizando. Las métricas/estadísticas. no las he usado nunca.
Hago un uso muy básico de Trello (tablero para anotar tareas pendientes), así que no he indagado en conocer más funcionalidad.

Tabla 12 - Resultados de la sexta pregunta de la encuesta