UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de la Aplicación Móvil "Castles": Productividad y Gamificación

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Terol Martin, Lluis

Tutor/a: Molina Marco, Antonio

CURSO ACADÉMICO: 2023/2024

A mis amigos, mis padres y todos los que estuvieron ahí en el proceso.

Agradecimientos

Me gustaría agradecer a mi tutor de TFG Antonio Molina Marco por su paciencia y ayuda en todo este proceso.

Resumen

El objetivo de este Trabajo de Fin de Grado se centra en la creación de una aplicación móvil llamada "Castles", diseñada para potenciar la productividad mediante un sistema de gamificación efectiva. Esta aplicación permitirá a los usuarios gestionar tareas, registrar el tiempo dedicado a cada una, y ofrecerá herramientas como un calendario y una agenda para optimizar la gestión del tiempo. La gamificación se implementará a través de un sistema de recompensas en forma de puntos que los usuarios podrán canjear para construir su propio castillo. "Castles" tiene un amplio espectro de aplicabilidad, siendo una herramienta valiosa para estudiantes que buscan optimizar su preparación para exámenes, así como para cualquier persona que aspire a alcanzar objetivos específicos en su vida diaria.

En cuanto a la tecnología, el proyecto se basará en TypeScript junto con React Native y Expo, aprovechando las ventajas de este último, que simplifica el proceso de desarrollo al ofrecer componentes y bibliotecas preconstruidas. Para el almacenamiento de datos, se utilizará Firebase, una solución robusta y escalable que se adapta perfectamente a las necesidades del proyecto. El control de versiones se gestionará a través de GitHub, y la metodología ágil se aplicará en tres sprints, además del sprint inicial, para lograr un desarrollo eficiente.

Palabras clave: productividad, gamificación, aplicación móvil, "Castles", Desarrollo de software

Abstract

The objective of this Bachelor's Thesis focuses on the development of a mobile application named "Castles," designed to enhance productivity through an effective gamification system. This application will enable users to manage tasks, record the time dedicated to each one, and provide tools such as a calendar and an agenda to optimize time management. Gamification will be implemented through a reward system in the form of points that users can redeem to construct their own castle. "Castles" has a broad spectrum of applicability, serving as a valuable tool for students looking to optimize their exam preparations, as well as anyone aiming to achieve specific goals in their daily lives.

Regarding technology, the project will be based on TypeScript in conjunction with React Native and Expo, taking advantage of the benefits of the latter, which simplifies the development process by offering pre-built components and libraries. For data storage, Firebase will be used, a robust and scalable solution that aligns perfectly with the project's requirements. Version control will be managed through GitHub, and an agile methodology will be applied in three sprints, in addition to the initial sprint, to ensure efficient development.

Keywords : Productivity, gamification, mobile application, "Castles", software development

Tabla de contenidos

Índice de figuras

1. Introducción

En este primer apartado se expondrá la idea de negocio, sus objetivos, los motivos que me llevaron a realizar este proyecto y la estructura de la memoria.

1.1 Motivación

En la era digital, donde la tecnología juega un papel crucial en nuestro día a día, la gestión apropiada del tiempo se ha convertido en una habilidad esencial, especialmente para los jóvenes que enfrentan múltiples distracciones y responsabilidades académicas. Según un estudio realizado por Steel y Ferrari (2013), la procrastinación afecta a aproximadamente el 80-95% de los estudiantes universitarios [1] , lo que puede llegar a tener un impacto negativo notable en su rendimiento académico y bienestar general. En este contexto, una aplicación móvil de productividad que permita a los usuarios registrar el tiempo dedicado a diversas actividades puede ser una herramienta invaluable para ayudar a formar hábitos de estudio más efectivos y sostenibles.
La gamificación ha demostrado ser una estrategia eficaz para aumentar la motivación y el compromiso del usuario en diversas actividades. Un artículo del año 2014 publicado por Hamari, Koivisto y Sarsa concluyó que la gamificación puede mejorar el compromiso del usuario mediante la incorporación de sistemas de recompensas y logros [2]. Al implementar un sistema de juego que recompensa a los usuarios al registrar horas de trabajo, la aplicación no solo facilita el seguimiento del tiempo, sino que también incentiva a los jóvenes a mantenerse enfocados y productivos, convirtiendo el acto de estudiar en una experiencia más interactiva y gratificante.

Además, la necesidad de herramientas actuales como aplicaciones que ayuden a los jóvenes a desarrollar habilidades como la gestión del tiempo y autodisciplina es más importante que nunca. En un mundo lleno de información y distracciones digitales, las aplicaciones que usan técnicas de gamificación pueden ser una forma atractiva, accesible y efectiva de mejorar las habilidades de los usuarios así como su rendimiento académico. Al ofrecer recompensas por el tiempo dedicado a actividades productivas, esta aplicación puede ayudar a los usuarios a establecer y mantener hábitos positivos y saludables que les beneficiaran en numerosas situaciones.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es desarrollar una aplicación móvil de productividad que permita a los usuarios seguir y controlar el tiempo que dedican a diversas actividades de su elección, incorporando elementos de gamificación para incentivar el uso continuo y efectivo de la aplicación. Esta aplicación tiene como finalidad ayudar a los usuarios,

especialmente a los jóvenes, a mejorar sus habilidades de gestión del tiempo y a desarrollar hábitos de estudio más sólidos y sostenibles.

Para los usuarios jóvenes, se pretende ayudar a los estudiantes a mejorar sus habilidades de gestión del tiempo, permitiéndoles registrar de manera precisa y sencilla las horas dedicadas a actividades académicas, y fomentar el desarrollo de hábitos de estudio más sólidos y sostenibles mediante la gamificación, ofreciendo recompensas que motiven a seguir usando regularmente la aplicación.

En cuanto a las funcionalidades, se busca implementar un temporizador de enfoque que permita a los usuarios medir el tiempo dedicado a diferentes tareas, desarrollar un sistema de seguimiento de progreso que permita a los usuarios visualizar su rendimiento a lo largo del tiempo, e incorporar la gestión de tareas para que los usuarios puedan organizar sus actividades de manera eficiente.

Para alcanzar estos objetivos, se va a adoptadar la metodología ágil Scrum. Esta metodología se basa en la realización de sprints cortos y regulares, permitiendo una planificación flexible y una adaptación continua a los cambios y necesidades del proyecto. A lo largo de cada sprint, se llevan a cabo la especificación, desarrollo y testing de las unidades de trabajo (UT) correspondientes, garantizando que el producto evolucione de manera incremental y con calidad.

En términos de rendimiento, se garantizará que la aplicación funcione de manera fluida y eficiente, minimizando los tiempos de carga y el consumo de recursos del dispositivo, y asegurando que los datos se sincronizan correctamente en tiempo real, proporcionando una experiencia de usuario consistente y confiable. Respecto a la usabilidad, se diseñará una interfaz intuitiva y fácil de usar, asegurando que los usuarios puedan navegar y utilizar todas las funcionalidades de la aplicación sin dificultad. Además, se realizarán pruebas de usabilidad para identificar y corregir posibles problemas, mejorando continuamente la experiencia del usuario.

Con estos objetivos, se busca que la aplicación no solo sea una herramienta útil para la gestión del tiempo, sino también una plataforma atractiva y motivadora que fomente el uso continuo y ayude a los usuarios a alcanzar sus metas de manera más efectiva.

1.3 Estructura de la memoria

La memoria se ha estructurado de la siguiente forma:

El segundo capítulo desarrolla la evaluación de la idea de negocio, mostrando el estudio de mercado, el análisis DAFO, la proyección económica a 3 años, el Lean Canvas y las conclusiones de la evaluación.

El tercer capítulo comprende las herramientas utilizadas durante el desarrollo del proyecto.

En el cuarto capítulo se define el alcance del producto, mostrando las prioridades y el público objetivo, la metodología de trabajo y la cronología del desarrollo.

El quinto capítulo ilustra la especificación de requisitos, incluyendo el análisis y la especificación detallada de los mismos.

El sexto capítulo se centra en el diseño de la aplicación, describiendo el uso del patrón MVC, el patrón Repositorio, la arquitectura general y la base de datos.

El séptimo capítulo aborda la implementación, explicando la conexión con la base de datos y la implementación de los componentes.

El octavo capítulo describe el experimento de validación, presentando los resultados obtenidos de la encuesta con los usuarios.

Finalmente, el noveno capítulo abarca las conclusiones y el trabajo futuro para la aplicación, relacionando los objetivos iniciales, las lecciones aprendidas y las sugerencias obtenidas en la validación.

2. Evaluación de la idea de negocio

En este segundo capítulo se evaluará la viabilidad de una aplicación móvil de productividad mediante un estudio de mercado. Se determinará la posición dentro del sector de aplicaciones de productividad y gestión del tiempo del proyecto. Se presentará un análisis DAFO para obtener una vista global del proyecto, identificando sus fortalezas, debilidades, oportunidades y amenazas. La viabilidad de la idea se estudiará mediante una proyección económica detallada, y se analizará el modelo de negocio propuesto utilizando el método Lean Canvas, para certificar que la estructura y estrategia de la aplicación sean sólidas y efectivas para alcanzar los objetivos establecidos.

2.1 Estudio de mercado

Una de las partes más importantes a la hora de lanzar un producto es conocer el mercado al que va a salir y los competidores y productos similares que hay disponibles con el fin de encontrar un hueco que llenar e identificar las características básicas que debería proporcionar el producto que queremos desarrollar.

Con ese fin, se han identificado cuatro aplicaciones destacadas en el mercado: Flora, Timeular, Todoist y Forest. Cada una de estas aplicaciones ofrece distintas funcionalidades y enfoques para ayudar a los usuarios a mejorar su gestión del tiempo. Flora es conocida por su integración de la gamificación con el objetivo de fomentar la concentración y el enfoque mediante el crecimiento virtual de plantas. Timeular, por otro lado, proporciona una solución física y digital para el seguimiento del tiempo, ofreciendo un enfoque único al permitir a los usuarios rastrear el tiempo con un dispositivo tangible. Todoist se destaca por su capacidad de gestión de tareas y proyectos, siendo una herramienta robusta para organizar y priorizar tareas diarias. Forest, similar a Flora, utiliza la gamificación para motivar a los usuarios a desconectarse de sus dispositivos y concentrarse en sus tareas, promoviendo hábitos de trabajo saludables. La selección de estos competidores permite una comparación integral y relevante, considerando la diversidad de métodos y características que abordan la gestión del tiempo desde diferentes ángulos.

Flora es una aplicación móvil diseñada para ayudar a los usuarios a mantenerse enfocados y productivos. Funciona principalmente como un temporizador basado en la técnica Pomodoro. Flora es una app con una gamificación que consiste en que por cada set de tiempo que completes (el usuario puede elegir el tiempo) la app te recompensa con un árbol. Estos árboles constituyen un jardín que el usuario puede ir viendo crecer. El usuario “cuida” su jardín digital completando temporizadores y siendo disciplinado ya que los árboles obtenidos desaparecen cada 7 días.

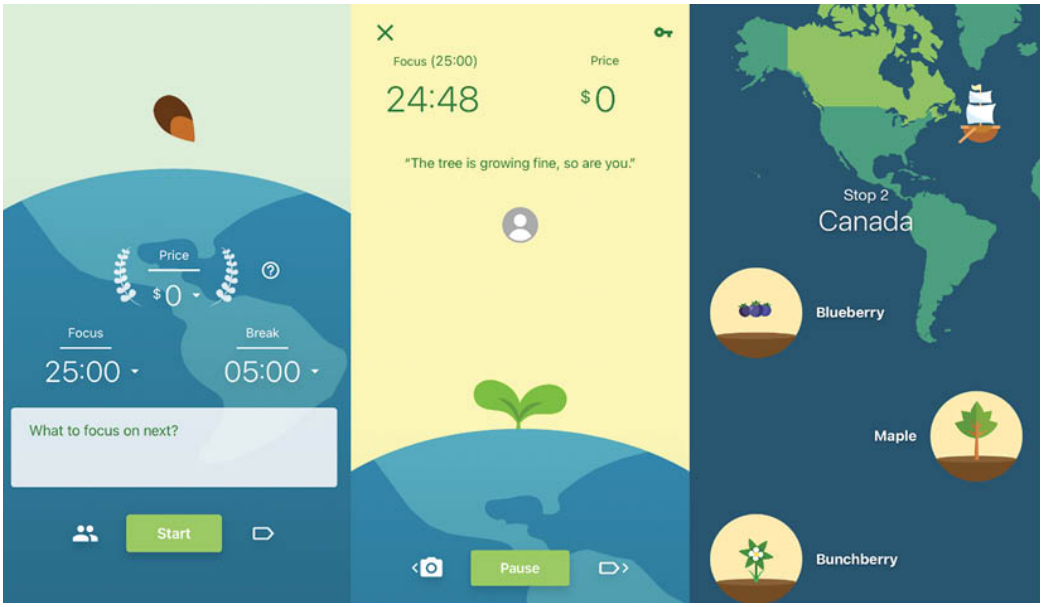


Figura 1. Capturas de pantalla de Flora

La app cuenta con un perfil que comparte espacio con las estadísticas del usuario. Las stats que se muestran son las horas completadas en los últimos 7 días y los árboles reales plantados por el usuario. Más abajo se muestran unas pestañas la primera de las cuales muestra pequeños carteles acerca de los últimos sets de tiempo completados por el usuario incluyendo el árbol que ha desbloqueado así como el tiempo que estuvo concentrado. La segunda pestaña muestra las estadísticas del usuario. Hay un apartado para stats diarias, semanales y mensuales. La última pestaña es una to-do list un poco complicada y nada intuitiva en la que puedes ponerle un recordatorio para que te notifique a una hora en particular y puedes ponerle objetivo de tiempo pero no en horas si no en sets de concentración pero al poder cambiar el tiempo de estos sets esto puede ser un poco confuso y llevar a no saber cuánto tiempo le quieres asignar a una tarea

La funcionalidad "Habit Tracker" en la aplicación Flora ofrece una forma innovadora y atractiva de fomentar y mantener hábitos saludables. Esta característica permite a los usuarios establecer y personalizar hábitos específicos que desean desarrollar, como hacer ejercicio, meditar o estudiar. Flora envía recordatorios en forma de notificaciones para ayudar a los usuarios a mantenerse en el camino.

La app es gratuita y ofrece la posibilidad de conseguir árboles diferentes comprando paquetes de expansión. El precio de estos es de 1.99€. La app también incluye una penalización monetaria opcional si el usuario no es capaz de completar el set de tiempo de concentración que se puso. Esta penalización económica va destinada a plantar árboles reales en Asia y en África. La empresa no menciona en ningún sitio si de esta penalización se lleva una pequeña parte con lo cual aunque no sea algo ético no se puede descartar como posible fuente de ingreso suplementaria.

Timeular es una aplicación innovadora diseñada para mejorar la gestión del tiempo y el seguimiento de las actividades. Su enfoque distintivo combina una aplicación de software con un dispositivo físico, el Timeular Tracker, que es un poliedro de ocho caras. Cada cara del Tracker puede ser asignada a una actividad o tarea específica, y al voltear el dispositivo y colocar la cara correspondiente hacia arriba, la aplicación comienza a registrar automáticamente el tiempo dedicado a esa actividad.

La aplicación Timeular proporciona una interfaz visual clara que muestra el desglose del tiempo dedicado a diferentes tareas, facilitando a los usuarios la comprensión de cómo utilizan su tiempo. Los usuarios pueden personalizar las etiquetas de cada cara del Tracker para adaptarse a sus necesidades específicas, ya sea para trabajo, estudio, o actividades personales. Además de su función de seguimiento del tiempo, Timeular también permite establecer objetivos de tiempo, recibir recordatorios y generar informes detallados, lo que lo hace ideal para autónomos, profesionales y estudiantes interesados en una gestión del tiempo más eficiente.

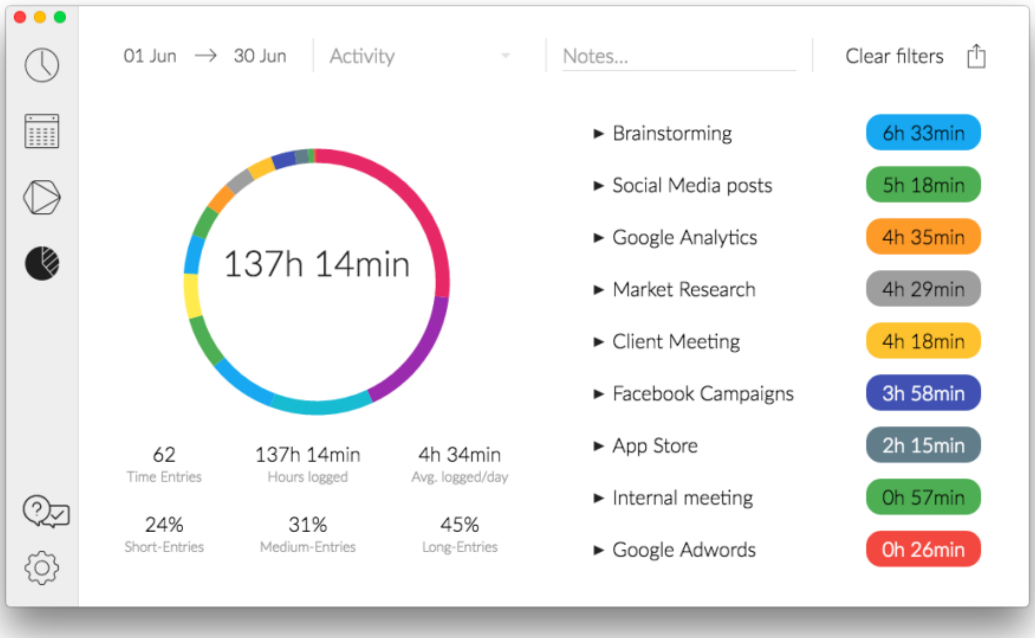


Figura 2. Dashboard de Timeular

Timeular opera con un modelo de negocio híbrido que combina la venta del dispositivo físico Tracker (57.50€) con opciones de suscripción para características avanzadas en la aplicación. El dispositivo Timeular Tracker se compra por un precio único, proporcionando acceso a las funciones básicas de seguimiento del tiempo. Sin embargo, para acceder a funcionalidades más avanzadas como informes detallados, integraciones con otras aplicaciones, y capacidades de personalización más profundas, los usuarios pueden optar por una suscripción premium.

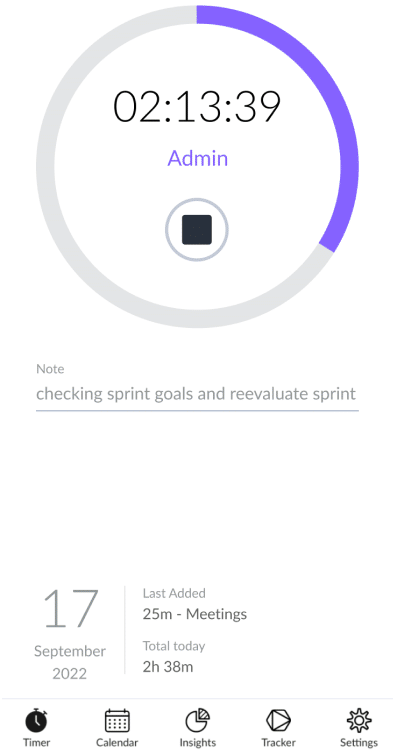


Figura 3. Temporizador Timeular

Esta estrategia de "hardware como entrada" seguida de un modelo de suscripción para servicios adicionales permite a Timeular atender tanto a usuarios casuales interesados solo en el seguimiento básico del tiempo como a profesionales y empresas que buscan una solución más robusta y detallada para la gestión del tiempo y la productividad. La combinación de un dispositivo tangible con una aplicación sofisticada distingue a Timeular en el mercado de aplicaciones de productividad y gestión del tiempo aunque supone una barrera de entrada para los usuarios no dispuestos a comprar el dispositivo hardware.

Todoist es una aplicación de gestión de tareas y organización personal que ayuda a los usuarios a mantener el control de sus actividades diarias y proyectos. Su interfaz intuitiva permite a los usuarios crear tareas, establecer fechas límite y recordatorios, y organizarlas en proyectos. Una de las principales características de Todoist es su flexibilidad para adaptarse a diferentes metodologías de trabajo, como GTD (Getting Things Done) o la técnica Pomodoro. Además, la aplicación ofrece etiquetas y filtros para categorizar y priorizar tareas, facilitando la gestión y el seguimiento del trabajo.

Todoist también se destaca por sus capacidades de colaboración, permitiendo a los equipos compartir proyectos, asignar tareas y comunicarse dentro de la aplicación. Otra funcionalidad destacada es la integración con otras aplicaciones y servicios, como Google Calendar, Dropbox, y Slack, lo que amplía su utilidad en distintos entornos de trabajo.

La aplicación cuenta con una vista de "Hoy" y "Próximas" para ayudar a los usuarios a enfocarse en lo inmediato mientras mantienen la visibilidad de tareas futuras. Las estadísticas de productividad de Todoist, conocidas como

"Karma", ofrecen una visión gamificada del rendimiento personal, incentivando a los usuarios a ser consistentes en la gestión de sus tareas.

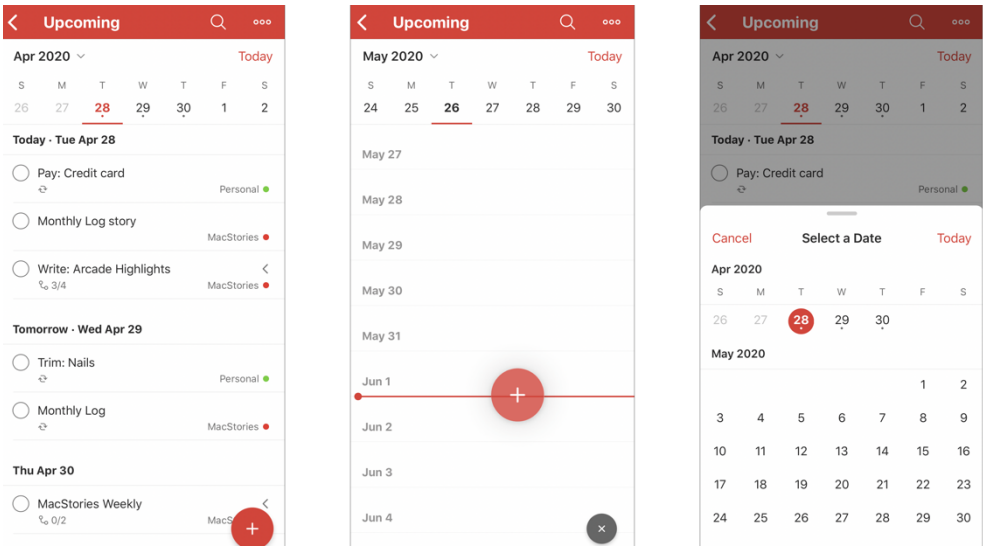


Figura 4. Pantallas principales Todoist

Todoist opera con un modelo freemium. La versión gratuita proporciona funciones básicas de gestión de tareas, mientras que la versión premium, con un costo de suscripción, ofrece características adicionales como recordatorios basados en la ubicación, archivos adjuntos más grandes, y vistas personalizadas. La suscripción premium está dirigida tanto a usuarios individuales como a equipos que buscan una mayor funcionalidad y personalización.

Forest es una aplicación móvil muy similar en concepto a Flora diseñada para ayudar a sus usuarios a mantenerse enfocados y alejados de las distracciones de sus teléfonos. La aplicación utiliza una técnica única y visual para fomentar la concentración: los usuarios plantan una semilla virtual en Forest, y esta semilla crece gradualmente hasta convertirse en un árbol mientras el usuario se mantiene enfocado. Si el usuario deja la app para usar otras aplicaciones en su teléfono, el árbol plantado dejará de crecer y eventualmente morirá.

La aplicación permite a los usuarios seleccionar diferentes especies de árboles para crear un sentido de variedad y logro a medida que se construye un bosque virtual con el tiempo. Cada árbol representa un período de concentración exitoso, y el bosque entero es un reflejo visual del tiempo total que el usuario ha pasado enfocado.

Forest también incluye un componente social, donde los usuarios pueden competir con amigos o trabajar juntos para plantar árboles virtuales, fomentando la motivación y la responsabilidad. Además, la aplicación ofrece la posibilidad de plantar árboles reales, colaborando con organizaciones de reforestación, lo que añade un aspecto de responsabilidad social y ambiental.

Figura 5. Pantallas principales Forest

Forest es una aplicación de pago (3.99€ en la App Store) con opciones de compra dentro de la app. Los usuarios pueden comprar monedas virtuales o "créditos del bosque" para desbloquear diferentes especies de árboles o sonidos ambientales que mejoran la experiencia de la aplicación. Aunque la funcionalidad básica es gratuita, las características premium proporcionan una experiencia más personalizada y diversa.

Como podemos ver cada una de estas aplicaciones ofrece distintas funcionalidades y enfoques para ayudar a los usuarios a mejorar su gestión del tiempo, formación de hábitos y concentración. A modo de resumen he creado la siguiente tabla (ver Tabla 1):

Aplicación	Descripción	Características destacables	Modelo de Negocio	Precio
Flora	Aplicación móvil que ayuda a mantenerse enfocado mediante la técnica Pomodoro, recompensando con árboles virtuales.	-Temporizador Pomodoro - Gamificación con árboles virtuales -Estadísticas de tiempo y árboles plantados - Habit Tracker con recordatorios	Gratuita con opciones de compra dentro de la app (expansiones de árboles) y penalización monetaria opcional.	Gratis, paquetes de expansión a 1.99€

Timeular	Combina software con un dispositivo físico de ocho caras para rastrear el tiempo dedicado a actividades específicas.	- Dispositivo físico para rastreo de tiempo - Interfaz visual clara - Personalización de etiquetas - Objetivos de tiempo y recordatorios - Informes detallados	Venta del dispositivo físico y suscripciones para funciones avanzadas.	57.50€ por el dispositivo, suscripción para características avanzadas
Todoist	Aplicación de gestión de tareas y organización personal que permite crear, organizar y seguir tareas y proyectos.	- Gestión de tareas y proyectos - Fechas límite y recordatorios - Colaboración en equipo - Integración con otras aplicaciones - Estadísticas de productividad ("Karma")	Freemium con versión gratuita y suscripción premium para características adicionales.	Gratis, suscripción premium disponible
Forest	Similar a Flora, fomenta la concentración plantando árboles virtuales que crecen mientras el usuario se mantiene enfocado.	- Temporizador para evitar distracciones - Gamificación con árboles virtuales - Componentes sociales y competiciones - Plantación de árboles reales mediante colaboraciones	Aplicación de pago con compras dentro de la app para desbloquear características adicionales.	3.99€ en la App Store, compras dentro de la app

Tabla 1. Tabla comparativa entre aplicaciones competidoras

Cada una de estas aplicaciones ofrece un enfoque diferente para mejorar la gestión del tiempo y la formación de hábitos. Flora y Forest utilizan la gamificación para incentivar la concentración mediante la creación de jardines

virtuales, mientras que Timeular combina un dispositivo físico con una aplicación para rastrear el tiempo de manera precisa. Por su parte, Todoist se destaca por su flexibilidad y capacidad de integración con otras herramientas de productividad.

Por lo tanto, podemos ver que las características clave que deben ser consideradas para "Castles" incluyen un sistema avanzado de gamificación con recompensas, una sección de estadísticas precisas, apartado social para conectar con la comunidad, alta personalización de tareas y recordatorios. Estas características no solo atraerán a usuarios interesados en mejorar su productividad, sino que también ofrecerán una experiencia única y atractiva.

"Castles" se diferenciará en el mercado de aplicaciones de productividad mediante una gamificación más compleja y atractiva, donde los usuarios podrán construir y personalizar castillos virtuales. Este enfoque proporciona un incentivo visual y tangible para mantener hábitos positivos, haciendo que la experiencia sea más envolvente y motivadora. El sistema de puntos y recompensas permitirá a los usuarios canjear puntos por mejoras en su castillo virtual o en un futuro beneficios tangibles o alguna acción social similar a la idea de Forest y Flora de plantar arboles, aumentando su compromiso y motivación.

2.2 Estudio de Técnicas de Gamificación

La gamificación se ha convertido en una estrategia poderosa para aumentar la participación y el compromiso de los usuarios en diversas plataformas y contextos. Esta técnica implica la aplicación de elementos de juego con el objetivo de motivar y recompensar comportamientos deseados. Esto es conocido como refuerzo positivo [3]. En este apartado, expondré varias técnicas de gamificación, incluyendo sistemas de puntos y recompensas, tableros de líderes, desafíos y misiones, narrativa y elementos de historia, así como niveles y progresión. Todas estas técnicas se basan en principios psicológicos como el refuerzo positivo que fomentan la participación activa y el compromiso a largo plazo de los usuarios.

La técnica de los sistemas de puntos y recompensas se basa en la psicología de la recompensa, donde los usuarios son incentivados a participar y completar actividades mediante la promesa de puntos acumulativos o recompensas tangibles. Los puntos pueden ser otorgados por diversas acciones, como completar tareas, alcanzar hitos o participar de manera regular. Las recompensas pueden variar desde descuentos en productos o servicios, acceso a contenido exclusivo, reconocimiento social a través de insignias virtuales o incluso premios físicos. Este sistema motiva a los usuarios a seguir participando y a comprometerse con la plataforma, ya que buscan obtener más puntos y alcanzar recompensas cada vez mayores.

Los tableros de líderes (leaderboards) son una herramienta poderosa para fomentar la competencia amistosa y el compromiso entre los usuarios. Al mostrar las puntuaciones o clasificaciones de los participantes en tiempo real,

se crea un ambiente de desafío donde los usuarios se esfuerzan por superar a sus compañeros y alcanzar los primeros puestos. Esto puede aumentar la motivación intrínseca de los usuarios, ya que buscan mejorar constantemente y demostrar su habilidad en relación con los demás. Además, los líderes pueden servir como modelos a seguir para otros usuarios, inspirándolos a esforzarse más y mantenerse activos en la plataforma.

La implementación de desafíos y misiones dentro de una experiencia gamificada proporciona a los usuarios objetivos claros y específicos para alcanzar. Estos desafíos pueden variar en dificultad y complejidad, y pueden estar diseñados para promover el aprendizaje, la colaboración o la exploración dentro de la plataforma. Al completar con éxito un desafío o una misión, los usuarios pueden recibir recompensas adicionales, como puntos extra, insignias especiales o acceso a contenido exclusivo. Esto crea un ciclo de retroalimentación positiva donde los usuarios se sienten motivados a participar activamente y a superar obstáculos para obtener recompensas.

La incorporación de una narrativa o una historia dentro de la experiencia gamificada añade profundidad y significado a la actividad. Al desarrollar un contexto narrativo para la plataforma, los usuarios se ven inmersos en un mundo ficticio donde sus acciones tienen consecuencias y su progreso contribuye a la trama general. Esto crea un sentido de propósito y compromiso emocional, ya que los usuarios se identifican con los personajes o situaciones presentadas en la historia. Además, una narrativa bien desarrollada puede aumentar la retención de información y el compromiso a largo plazo, ya que los usuarios están más involucrados en la experiencia en su conjunto.

Dividir la experiencia gamificada en niveles o etapas progresivas proporciona a los usuarios una sensación clara de logro y avance a medida que completan tareas o acumulan puntos. Cada nivel puede representar un nuevo conjunto de desafíos o habilidades para dominar, lo que mantiene el interés y la motivación a lo largo del tiempo. A medida que los usuarios avanzan de nivel, pueden desbloquear nuevas funciones, recompensas o áreas de la plataforma, lo que proporciona un incentivo adicional para seguir participando y explorando. Además, la sensación de progresión constante puede aumentar la autoeficacia de los usuarios, haciéndolos sentir competentes y capaces de alcanzar metas cada vez mayores.

Técnicas a Aplicar en Castles:

La gamificación de la app está basada en el sistema de puntos y recompensas. La funcionalidad consiste en que el usuario reciba puntos por completar horas concentrado en una actividad que el usuario designe. De esta manera, el usuario siente una atracción hacia la aplicación y consigue una motivación externa para reforzar la posible motivación intrínseca que le ha llevado a descargar la aplicación en primer lugar.

El sistema de puntos y recompensas es una estrategia de gamificación profundamente arraigada en los principios de la psicología del comportamiento

humano. Explicado desde un punto de vista psicológico, consiste en aprovechar el poder del refuerzo positivo, un concepto bien establecido en la teoría del aprendizaje. Cuando los usuarios participan en una actividad o completan una tarea dentro de la plataforma, son gratificados con puntos. Estos puntos actúan como recompensas, activando centros de placer en el cerebro y fortaleciendo la conexión entre la acción realizada y la gratificación recibida. Este proceso refuerza el comportamiento deseado, incentivando a los usuarios a seguir participando y alcanzando objetivos dentro del sistema.

Además del refuerzo positivo, el sistema de puntos y recompensas se basa en la teoría del refuerzo, donde los puntos y las recompensas se convierten en estímulos secundarios asociados con una experiencia positiva. Con el tiempo, estos puntos adquieren un valor simbólico y se vuelven intrínsecamente gratificantes para los usuarios. La anticipación de ganar puntos y alcanzar recompensas crea un ciclo de retroalimentación positiva, motivando a los usuarios a participar activamente y a perseguir objetivos dentro de la plataforma.

Además, el sistema de puntos y recompensas capitaliza la teoría de la motivación extrínseca al ofrecer incentivos tangibles para la participación. Las recompensas satisfacen las necesidades básicas de los usuarios y refuerzan su comportamiento deseado. Esta motivación extrínseca impulsa a los usuarios a comprometerse con la actividad, especialmente cuando las recompensas son personalizadas y alineadas con sus intereses y objetivos. En resumen, el sistema de puntos y recompensas no solo motiva a los usuarios a participar y a alcanzar metas dentro de la plataforma, sino que también fortalece su conexión emocional y su compromiso con la experiencia gamificada.

Otra de las técnicas de gamificación que se podría implementar es la de las misiones y desafíos. La idea se basa en proporcionar a los usuarios objetivos claros y específicos para alcanzar, enriqueciendo así su experiencia. Estas misiones y desafíos representan una extensión natural del sistema de puntos y recompensas, ya que ofrecen oportunidades adicionales para que los usuarios obtengan gratificaciones al completar tareas específicas o superar obstáculos designados. Al igual que con el sistema de puntos y recompensas, las misiones y desafíos aprovechan los principios de la psicología del comportamiento, ofreciendo incentivos claros para motivar a los usuarios a participar activamente y a perseguir objetivos dentro de la plataforma. La anticipación de completar una misión o superar un desafío proporciona una sensación de logro y satisfacción, fortaleciendo la conexión emocional de los usuarios con la experiencia gamificada y aumentando su compromiso a largo plazo.

Existen diversas técnicas de gamificación que he comentado, como la utilización de tableros de líderes o la implementación de sistemas de niveles y progresión, que serían pasos lógicos a dar para seguir con el crecimiento y desarrollo de la aplicación. Sin embargo, dado el alcance del proyecto, no serán incorporadas en esta fase de desarrollo, a pesar de que representan propuestas intrigantes. Estas funcionalidades podrían ser consideradas y desarrolladas en el futuro si el proyecto acabara saliendo al mercado y esté consolidado y se busque mejorar la experiencia del usuario.

2.3 Validación ideas

Para confirmar las ideas analizadas y las funcionalidades sugeridas en la sección anterior, se propuso realizar una encuesta con el fin de consolidar las conclusiones previamente mencionadas.

Con esa idea elaboré una serie de preguntas y me puse en contacto con diferentes personas que podrían ser usuarios en potencia. Entre ellos se encontraban estudiantes de grado, de posgrado y de instituto. Finalmente participaron 10 potenciales early adopters y aquí estan sus respuestas.

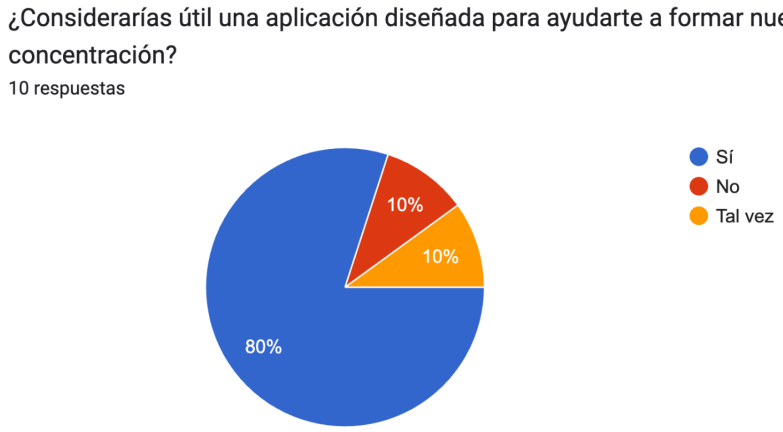


Figura 6. Encuesta validación ideas Pregunta 1

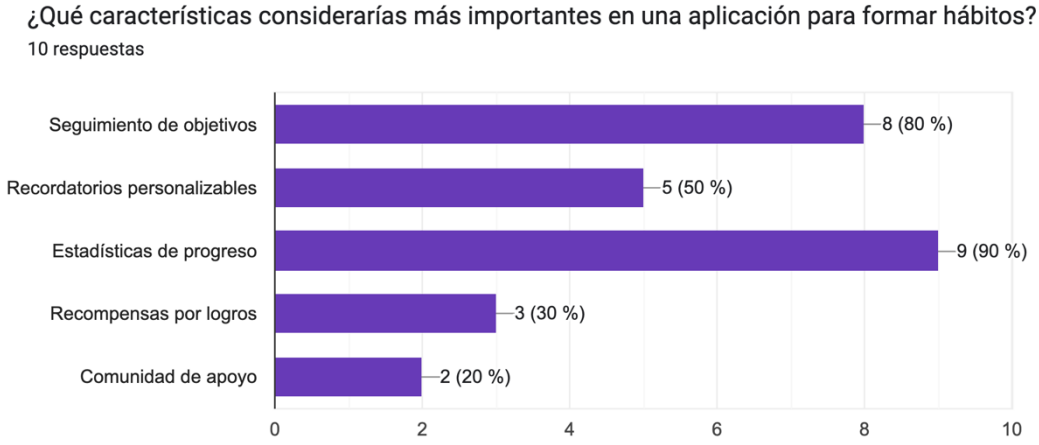


Figura 7. Encuesta validación ideas Pregunta 2

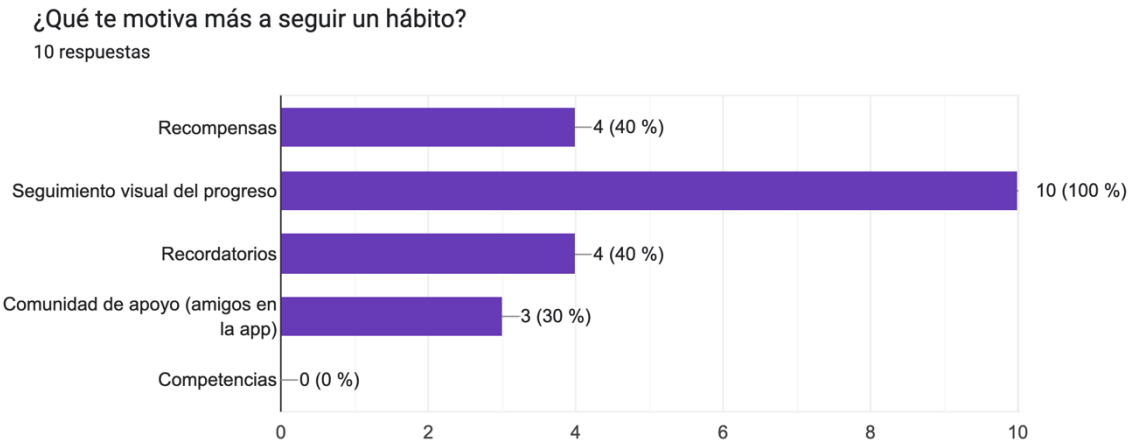


Figura 8. Encuesta validación ideas Pregunta 3

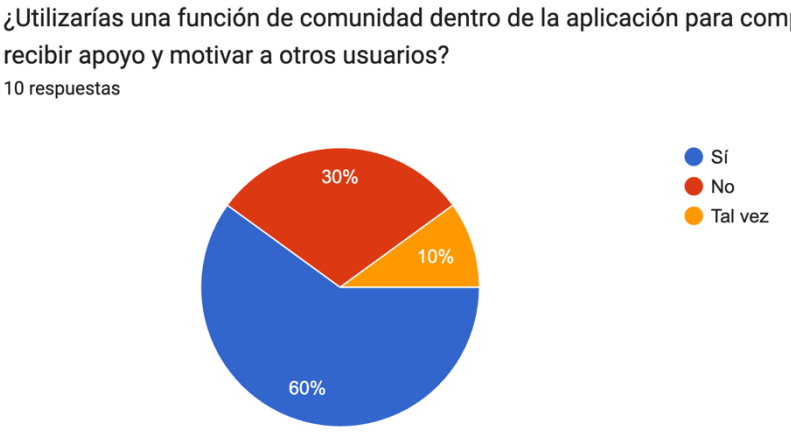


Figura 9. Encuesta validación ideas Pregunta 4

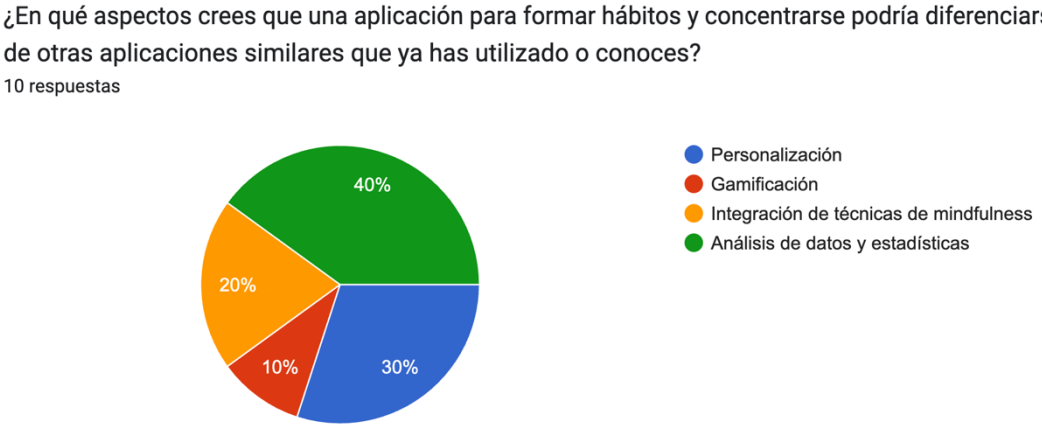


Figura 10. Encuesta validación ideas Pregunta 5

¿Crees que una sección dentro de la app donde se pueden canjear los puntos y se desarrolla un minijuego sería necesario o consideras que puede ser una distracción para el usuario?

10 respuestas

Figura 11. Encuesta validación ideas Pregunta 6

¿Qué tipo de seguimiento te resultaría más útil para evaluar tu progreso hacia la formación de hábitos?

10 respuestas

Figura 12. Encuesta validación ideas Pregunta 7

¿Estarías dispuesto/a a pagar por una aplicación premium que ofrezca características adicionales para ayudarte a formar hábitos y mejorar la concentración?

10 respuestas

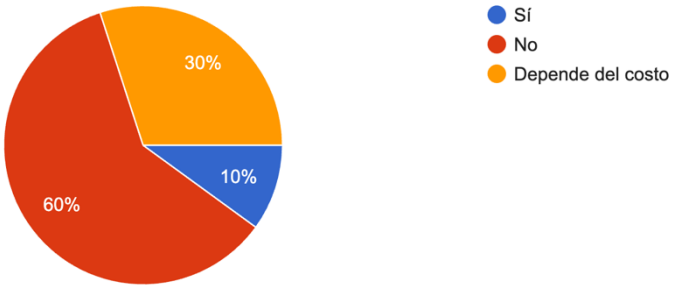


Figura 13. Encuesta validación ideas Pregunta 8

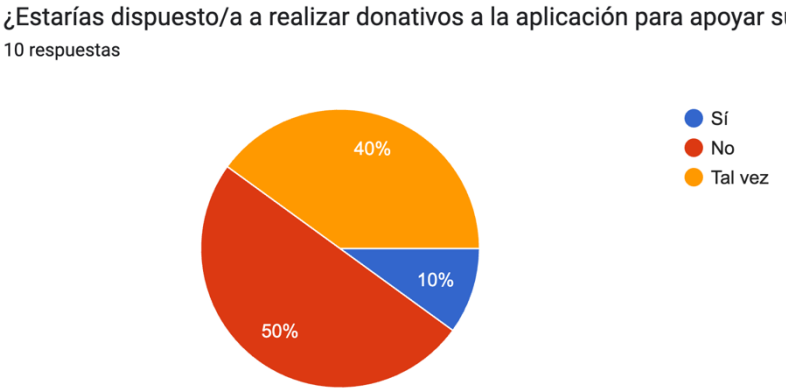


Figura 14. Encuesta validación ideas Pregunta 9

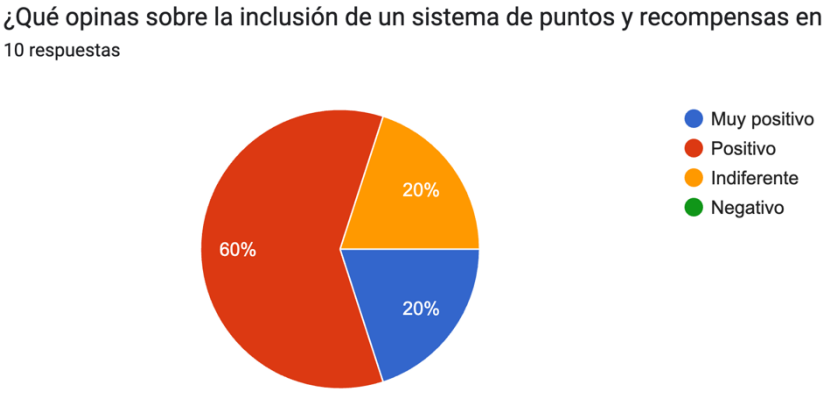


Figura 15. Encuesta validación ideas Pregunta 10

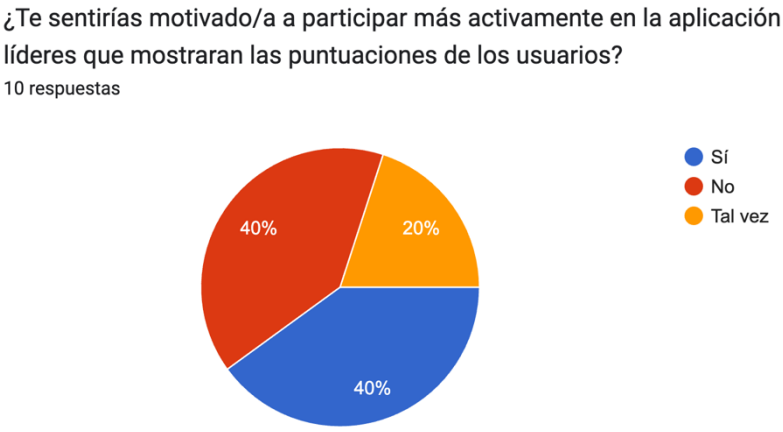


Figura 16. Encuesta validación ideas Pregunta 11

¿Cómo valorarías la idea de tener desafíos y misiones dentro de la aplicación para alcanzar objetivos específicos?

10 respuestas

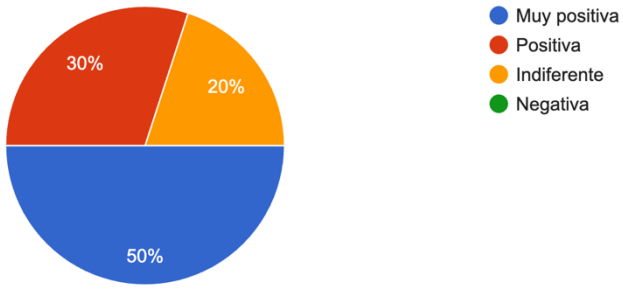


Figura 17. Encuesta validación ideas Pregunta 12

¿Crees que la incorporación de una narrativa o elementos de historia mejorarían tu experiencia con la aplicación?

10 respuestas

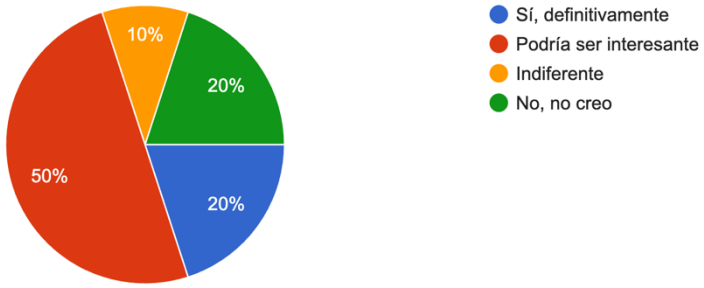


Figura 18. Encuesta validación ideas Pregunta 13

¿Qué frecuencia de uso consideras que le darías a una aplicación de este tipo?

10 respuestas

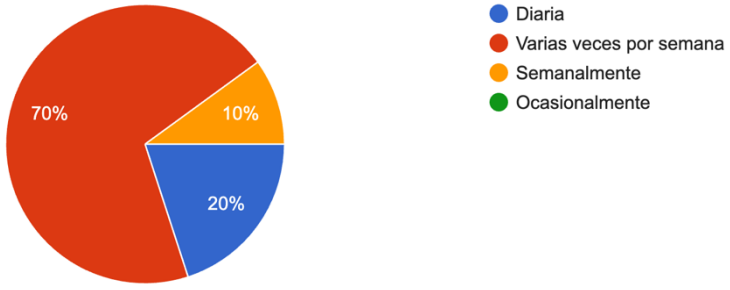


Figura 19. Encuesta validación ideas Pregunta 14

Figura 20. Encuesta validación ideas Pregunta 15

Los resultados de la encuesta (ver figuras 6-20) muestran un alto nivel de interés y aceptación hacia una aplicación confirmando así la viabilidad del proyecto. Confirmación de este hecho es que el 80% de los encuestados consideraron que una aplicación de este tipo les sería útil y tendría un impacto positivo en su vida y rendimiento académico.

Las características más valoradas por los encuestados para una aplicación de formación de hábitos incluyen el seguimiento de objetivos, estadísticas de progreso y recompensas por tiempo invertido. En cuanto a la diferenciación, los participantes sugirieron que la personalización, la gamificación y el análisis de datos detallados son aspectos que podrían hacer destacar a la aplicación frente a otras similares. Sin embargo, hay opiniones divididas sobre la inclusión de minijuegos; algunos creen que serían motivadores, mientras que otros consideran que podrían ser una distracción.

Por lo tanto Castles deberá centrarse en desarrollar las características más valoradas por los usuarios encuestados. El producto final se comparará con las demás aplicaciones como se muestra en la tabla (ver Tabla 2):

Funcionalidad	**Forest**	**Flora**	**Todoist**	**Timeular**	**Castles (Características planeadas)**
Temporizador de Enfoque	✓	✓	✓	✓	✓
Gamificación	✓	✓	✓	?	✓
Seguimiento del Progreso	✓	✓	?	✓	✓

Seguimiento de Hábitos					
Gestión de Tareas					
Informes y Estadísticas					
Modelo de Negocio	Aplicación de pago	Freemium	Freemium	Híbrido	Freemium

 - Característica incluida
 - Característica no incluida
 - Característica parcialmente incluida

Tabla 2. Tabla comparativa de funcionalidades entre aplicaciones competidoras y la aplicación propuesta.

Una de las áreas donde Castles se distingue es en el seguimiento de hábitos. A diferencia de Forest y Timeular, que no ofrecen esta funcionalidad, Castles planea implementar un sistema de seguimiento de hábitos, similar a Flora y Todoist, pero integrando recompensas y mejoras en el castillo como incentivos.

En la gestión de tareas, Castles planea ofrecer capacidades completas, al igual que Todoist y Timeular, superando las capacidades limitadas de Forest y Flora. Esto permitirá a los usuarios gestionar sus tareas de manera más eficiente dentro de una única aplicación.

Castles también ofrecerá informes y estadísticas, una funcionalidad que está presente en Flora, Todoist y Timeular, pero no en Forest. Esto ayudará a los usuarios a obtener una visión detallada de su rendimiento y a tomar decisiones informadas para mejorar su productividad.

En resumen, Castles se diferencia de las otras aplicaciones en su enfoque integral y motivacional, combinando la gestión de tareas, el seguimiento de hábitos y el progreso con una experiencia de gamificación envolvente y recompensas visuales, lo que la convierte en una opción atractiva y única para los usuarios que buscan mejorar su productividad de manera entretenida y efectiva.

2.3 Análisis DAFO

El análisis DAFO es una herramienta utilizada para evaluar la situación de un proyecto, examinando tanto sus características internas (Fortalezas y Debilidades) como su contexto externo (Oportunidades y Amenazas). Esta

evaluación permite planificar estrategias futuras de manera efectiva. En la figura 1 se presenta el análisis realizado sobre mi proyecto:

Debilidades	**Amenazas**
• Complejidad Potencial • Desarrollo y Mantenimiento	• Competencia Intensa • Cambios en las Preferencias de los Usuarios • Desafíos Técnico • Dependencia del Compromiso del Usuario
Fortalezas	**Oportunidades**
• Enfoque en la Productividad • Gamificación • Funcionalidades Versátiles	• Tendencia del Teletrabajo y Estudio en Casa • Integración con Otras Aplicaciones • Expansión de Funcionalidades • Tema de la productividad es tendencia

Debilidades:

- Complejidad Potencial: La variedad de funciones podría resultar abrumadora para algunos usuarios, especialmente para aquellos que buscan una solución más simple.

- Desarrollo y Mantenimiento: Implementar y mantener todas estas características puede requerir recursos significativos.

Amenazas:

- Competencia Intensa: El mercado de aplicaciones de productividad es altamente competitivo, con muchas alternativas disponibles.

- Cambios en las Preferencias de los Usuarios: Los usuarios pueden

preferir soluciones más simples o diferentes enfoques para la gestión del tiempo.

- Desafíos Técnicos: Mantener la aplicación actualizada y compatible con múltiples plataformas (iOS y Android) puede ser un desafío.

- Dependencia del Compromiso del Usuario: El éxito de la aplicación depende de que los usuarios registren activamente su tiempo, lo cual puede no ser consistente.

Fortalezas:

- Enfoque en la Productividad: La aplicación ayuda a los usuarios a rastrear y gestionar su tiempo, lo cual es fundamental para mejorar la productividad.

- Gamificación: El sistema de recompensas y la construcción de un castillo/fortaleza/palacio añaden un elemento lúdico que puede aumentar la motivación y el compromiso del usuario.

- Funcionalidades Versátiles: Incluye la creación de actividades, registro de tiempo, calendario, agenda y estadísticas, lo que la hace una herramienta integral para la gestión del tiempo.

Oportunidades:

- Tendencia del Teletrabajo y Estudio en Casa: Con más personas trabajando y estudiando desde casa, hay una creciente demanda de herramientas de productividad.

- Integración con Otras Aplicaciones: Posibilidad de integrar con otras herramientas y plataformas para aumentar la utilidad.

- Expansión de Funcionalidades: Oportunidad de agregar nuevas características basadas en la retroalimentación de los usuarios, como integración con métodos de productividad populares (p. ej., Pomodoro, GTD).

- El tema de la productividad está de moda [4] y esa popularidad puede servir para impulsar la app

2.4 Modelo de negocio y proyección económica

Como se ha comentado anteriormente la aplicación va a seguir un modelo Freemium que consiste en ofrecer unos servicios básicos gratuitos con la opción de ofrecer contenido o funciones más avanzadas a cambio de un pago [5]. Para una aplicación de este tipo, tendría más sentido económico implementar un modelo Freemium basado en suscripciones ya que garantizaría la sostenibilidad económica del proyecto mejor que un modelo basado en pagos únicos.

Sin embargo, como se ha visto reflejado en los resultados de la encuesta, no todos los usuarios se mostraron dispuestos a pagar por usar la app por lo tanto se podría optar por un modelo mixto donde el usuario pueda usar las funcionalidades principales de la aplicación de forma gratuita, pueda suscribirse por un bajo coste mensual para poder utilizar funcionalidades avanzadas y pueda realizar pagos únicos por paquetes de ampliación (p.e. cosméticos, temas para la interfaz etc.). Todo esto se podría implementar a futuro una vez se tenga una buena base de usuarios.

La encuesta también ha reflejado una predisposición a realizar donativos (50% de los usuarios estarían dispuestos a pensar si realizar un donativo). Por lo tanto, puede ser otra vía de ingresos que se debería explorar y explotar. Otra vía alternativa para generar ingresos podría ser incluir anuncios en la app. Un informe de Business of Apps indica que una proporción significativa de aplicaciones en la Google Play Store utiliza anuncios como modelo de monetización. En 2021, se estimó que aproximadamente el 82% de todas las aplicaciones gratuitas en la Google Play Store incluían anuncios [6]. Esta tendencia debe de ser similar en la Apple App Store, aunque las cifras podrían variar.

Con esto en mente, se ha llevado a cabo una proyección económica a cinco años con la que podemos ver la viabilidad y sostenibilidad del producto (ver Tabla 3).

Número de Año	**1**	**2**	**3**	**4**	**5**	**Coste o coste medio**
Tipos ingresos	**Año 1**	**Año 2**	**Año 3**	**Año 4**	**Año 5**	0.99€
Suscripciones	100	400	700	1200	1320	0.99€
Suscripciones renovadas después de un año	0	20	200	500	800	10€
Publicidad / Ads (visualizaciones)	986	3942	6899	11826	13009	3€
Donaciones	120	200	300	500	600	1.99€
Microtransacciones	40	150	275	450	700	
Total Ingresos	11 495 €	40 319 €	70 432 €	120 656 €	133 279 €	

Gastos Anuales						
Marketing	1 000 €	1 500 €	2 000 €	3 000 €	2 000 €	
Desarrolladores Frontend	3 000 €	3 100 €	3 200 €	3 500 €	3 000 €	
CMO - Director de Marketing	1 700 €	1 700 €	1 800 €	1 900 €	2 000 €	
Desarrolladores Backend	3 200 €	3 300 €	3 400 €	4 000 €	6 000 €	
Técnicos de soporte	1 200 €	1 200 €	1 200 €	1 200 €	2 400 €	
Administrativo	600 €	900 €	1 200 €	1 200 €	1 200 €	
Equipo creativo	0 €	0 €	600 €	1 200 €	1 200 €	
Licencia desarrollador registrado Apple	99 €	99 €	99 €	99 €	99 €	
Total Gastos	10 799 €	11 799 €	13 499 €	16 099 €	17 899 €	
Resultado Anual	696€	28 520 €	56 933 €	104 557 €	115 380 €	
Resultado Anual Acumulado	696 €	29 215 €	86 148 €	190 705 €	306 085 €	
Resultado Anual Acumulado	1 896 €	31 615 €	89 748 €	195 505 €	313 135 €	

Tabla 3. Proyección económica

La cantidad que un desarrollador puede ganar por la visualización de un anuncio en una aplicación móvil puede variar dependiendo de factores, como la plataforma de anuncios utilizada, la región del usuario, el tipo de anuncio y la demanda de los anunciantes.

Hay diferentes tipos de pagos dependiendo del tipo de anuncios. Por ejemplo, existe el pago por clic, donde el desarrollador gana dinero cada vez que un usuario hace clic en un anuncio, y el pago por acción, donde se paga al desarrollador cuando el usuario realiza una acción específica después de hacer clic en el anuncio, como registrarse en un servicio o hacer una compra. Sin embargo, el modelo que vamos a seguir, y el más simple, es el pago por cada mil impresiones (CPM), que paga al desarrollador por cada mil visualizaciones del anuncio

Las tarifas de CPM pueden variar, generalmente oscilando entre $1 y $10. En ciertos casos, como anuncios de video recompensados, pueden alcanzar hasta $12 en iOS y $10 en Android [7]. Por simplicidad en la tabla del presupuesto he usado 10€ como valor de referencia

Las microtransacciones [8] son pequeñas compras dentro de una aplicación que permiten a los usuarios adquirir bienes virtuales o desbloquear funcionalidades adicionales. Las microtransacciones se han convertido en una estrategia popular para monetizar aplicaciones y juegos, proporcionando a los desarrolladores una fuente continua de ingresos más allá de las ventas iniciales o suscripciones. A futuro, se podría considerar la implementación de microtransacciones en la aplicación para ofrecer artículos cosméticos, como personalizaciones del castillo y su entorno, así como expansiones de funcionalidades que permitan a los usuarios acceder a características

avanzadas. Estas microtransacciones no solo mejorarían la experiencia del usuario al ofrecer más opciones de personalización y funcionalidad, sino que también contribuirían a la sostenibilidad económica del proyecto.

Se puede observar que al estar desarrollando una aplicación puramente de software no hay gastos excesivos ya que, por la naturaleza intangible del software, no es necesario invertir en materia prima. También apostaríamos por el teletrabajo como forma de trabajo suprimiendo así costes de oficina, electricidad y demas. Por lo que la inversión más grande a realizar sería en el talento necesario para desarrollar la app. El coste de personal iría ascendiendo a lo largo de los años según la app vaya creciendo.

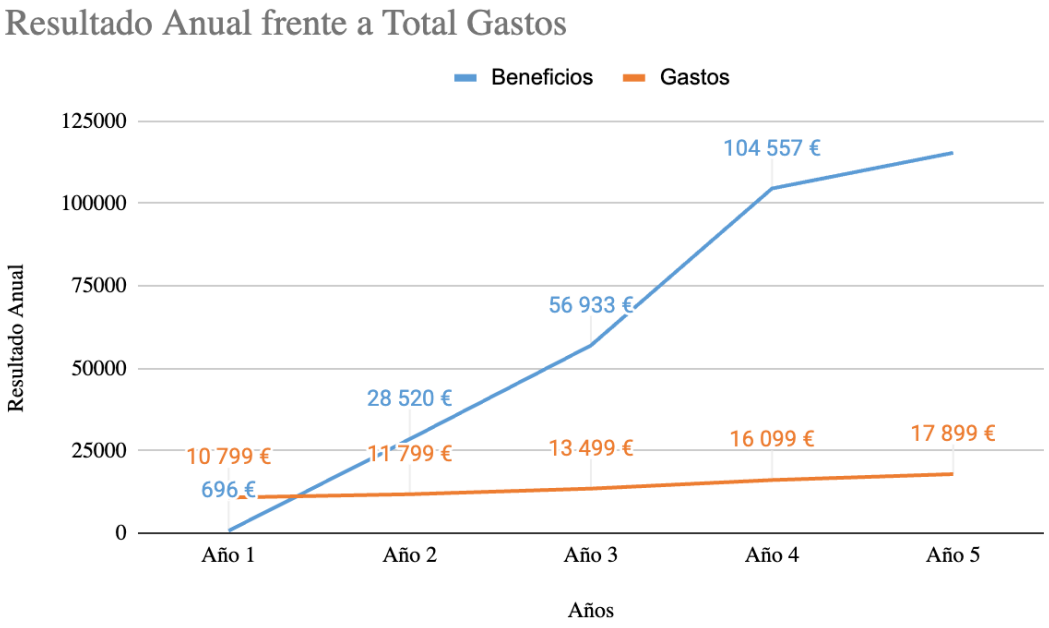


Figura 21. Gráfico ingresos frente a gastos totales anuales

En la tabla se puede apreciar lo comentado con anterioridad. Al plantear el desarrollo de una aplicación no se requeriría una gran inversión inicial ya que lo único realmente necesario serían los equipos de los desarrolladores que al estar en un entorno de startup podrían utilizar sus propios equipos por eso ese gasto se ha desestimado.

En definitiva, el estudio económico demuestra la viabilidad del proyecto así como su potencial económico resultando en un proyecto altamente rentable y sostenible en el tiempo que requeriría una mínima inversión que en ningún momento reportaría perdidas.

2.5 Lean Canvas

<table>
<tr>
<td rowspan="2">Problema

No hay una buena manera de registrar y trackear el tiempo que una persona dedica a cierta actividad sea estudiar, estar en el gimnasio, leer etc.

Las diferentes apps que existen (Flora, Forest, Todoist…) permiten al usuario realizar estas funciones pero de una manera muy tediosa y contraintuitiva.

La parte de gamificación (las que tienen) no suele estar demasiado elaborada y explotan el potencial de sus aplicaciones.</td>
<td>Solución

Ofrecer un “hub” que permita al usuario registrar tiempo para la actividad que desea y consultar el esfuerzo que ha invertido de una manera simple, didáctica y útil.</td>
<td rowspan="2">Proposición de valor
Castles es una app diseñada para ayudarte a mejorar tu productividad mientras disfrutas de una experiencia de juego única. Con nuestro innovador sistema de gamificación, te motivaremos a seguir tus actividades diarias y a alcanzar tus objetivos de tiempo de manera divertida y gratificante.

Crear Actividades Personalizadas: Crea actividades con nombres, descripciones y establece tus objetivos de tiempo para cada una

Calendario de Productividad: Visualiza de manera retrospectiva tu esfuerzo

Gamificación Divertida: Gana puntos por el tiempo que dedicas a tus</td>
<td>Ventaja competitiva
Enfoque innovador en la gamificación de la productividad, ofreciendo a los usuarios la oportunidad de registrar y mejorar sus hábitos de manera entretenida. Con una combinación única de registro de tiempo en tiempo real o a posteriori, una agenda integrada, y la posibilidad de personalizar actividades. La recompensa constante a través de la expansión de un castillo virtual motiva a los usuarios a mantenerse productivos.</td>
<td rowspan="2">Clientes
Profesionales Independientes: Freelancers, consultores y autónomos que desean llevar un registro efectivo de su tiempo y aumentar su productividad.

Estudiantes: Estudiantes de instituto, universidad o posgrado que buscan gestionar su tiempo de estudio y tareas de manera más efectiva.

Personas con Metas Personales: Cualquier persona que tenga objetivos personales, como ponerse en forma, aprender un nuevo idioma y necesite un medio para rastrear su progreso.

Early adopters:</td>
</tr>
<tr>
<td>Métricas

Nº de usuarios medio por mes, Nº de horas registradas semanales, Nº horas registradas semanales/usuarios activos</td>
<td>Canales

Anuncios e internet, por chats o por conocidos. Ámbitos personal y educativo. Colaboraciones con youtubers e influencers con temática de productividad.</td>
</tr>
</table>

		fortaleza o palacio virtual.		Productividad: Aquellas personas que ya utilizan aplicaciones de productividad y estén dispuestas a probar
Costos Costos para mantener el servidor(es)/bases de datos -> escalan según el uso, sueldos de empleados, campaña de marketing inicial /publicidad (?)		**Ingresos** La aplicación seguirá un modelo Freemium ofreciendo una versión gratuita de la aplicación con funcionalidades básicas, y luego proporciona una versión Premium de pago mensual que incluye características avanzadas. Compras en la Aplicación: Permite a los usuarios comprar elementos virtuales para personalizar su experiencia de gamificación, como accesorios para su castillo virtual		

2.6 Conclusiones

Como ha quedado reflejado en los apartados anteriores, la idea de negocio sería viable y debería de generar beneficios desde el primer año que el producto saliera al mercado.

La mayor complicación sería la competición con otras aplicaciones más asentadas en el mercado como se ha podido ver en el DAFO ya que arrebatarles cuota de mercado podría resultar una incognita. Asi como una vez conseguida la descarga inicial, mantener al usuario ligado a la app y convencerlo de realizar pagos dentro de la misma podría suponer un desafío.

Aun con esas se ha podido ver que hay un interés en lo que pueda ofrecer la app y una oportunidad de mercado que debe ser aprovechada.

3. Alcance del producto

En este apartado se detallará el alcance del producto a desarrollar. Se describirán las funcionalidades principales, los objetivos específicos y las limitaciones del sistema. Además, se abordarán las necesidades y requisitos que el producto pretende satisfacer, el público objetivo al que está dirigido, la metodología que se seguirá en el desarrollo y la cronología del mismo.

3.1 Prioridades y Público Objetivo

Al tratarse de una aplicación de productividad orientada al ayudar a contabilizar y hacer un seguimiento del estudio y la formación de hábitos la app el público objetivo sería logicamente estudiantes a todos los niveles sea instituto, grado, posgrado, opositores etc. y gente que requiera un seguimiento de sus hábitos para reforzar e impulsar su creación.

Además, el público estudiantil ofrece un entorno propicio para la experimentación y la retroalimentación. Este tipo de usuarios suelen estar abiertos a probar nuevas tecnologías y aportar feedback constructivos que puede contribuir a mejorar el producto en incrementos posteriores. Al involucrar a estudiantes en el desarrollo y la validación de la aplicación, se abre la posibilidad de recibir una retroalimentación valiosa que permita adaptar el producto a las necesidades reales de los usuarios, así como también proporcionar una experiencia de aprendizaje en el proceso.

Este proceso de realizar hipótesis o incrementos sobre la idea de negocio y verificarlas con el público objetivo según se van implementando se alinea estrechamente con la metodología Lean Startup [9]. En el enfoque Lean Startup, se apuesta por un ciclo de desarrollo iterativo, continuo e incremental, donde se proponen hipótesis sobre el mercado, el producto y el modelo de negocio, y luego se validan mediante experimentación y retroalimentación del público objetivo. Al adoptar este enfoque, se reduce el riesgo de desarrollar un producto que no sea viable o no tenga demanda real en el mercado, ya que se prioriza la validación temprana de las ideas y se ajusta el enfoque y dirección del producto en función de los comentarios recopilados.

Con esto en mente, se ha optado por realizar tres sprints que producirán un MVP (Minimum Viable Product producto minimo viable) el cual será validado por estudiantes a través de una prueba de la aplicación al final del tercer sprint donde los interesados podrán probar la app y rellenarán una encuesta para proporcionar sus criticas, comentarios y sugerencias.

Con los resultados de la encuesta presentada en el capítulo 2 en la mano, se han identificado las funcionalidades principales a desarrollarse, se han ordenado por prioridad y se ha elaborado el siguiente mapa de caracteristicas (ver Figura 22):

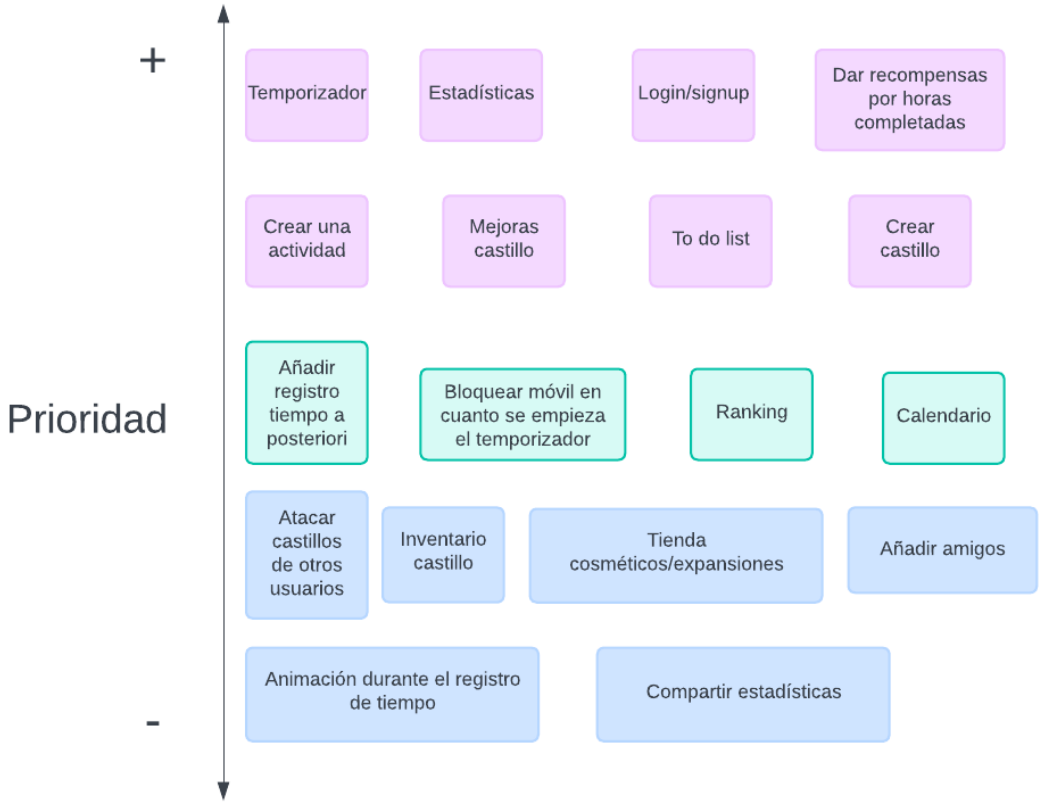


Figura 22. Mapa características

En la gráfica se puede obsevar las características que se han identificado ordenadas por prioridad. En los primeros sprints se va a optar por desarrollar las más características más prioritarias dejando para los sprints posteriores las menos prioritarias o incluso dejandolas fuera del alcance del proyecto. Estas características se ampliarán y se definiran para crear unidades de trabajo que se añadirán al backlog que es una especie de repositorio o fondo de armario donde se contienen las tareas a desarrollar en los diferentes sprints. Se verá más en detalle en el siguiente apartado.

3.2 Metodología

Como se ha mencionado en secciones anteriores, para el desarrollo del proyecto se han utilizado diversas técnicas y herramientas de Scrum[10] para implementar una metodología ágil de desarrollo. La metodología ágil [11] se centra en la entrega incremental y repetitiva del trabajo, promoviendo la flexibilidad y la rápida adaptación a los cambios. Para gestionar estas prácticas,

se ha utilizado Jira, una herramienta de gestión de proyectos que facilita la planificación, el seguimiento y la gestión del trabajo.

Figura 23. Tablero Kanban empleado

Se ha configurado un proyecto en Jira con el template Scrum, estableciendo cuatro "epics" (sprints) desde cero hasta tres. Se han explorado los diferentes apartados disponibles en la plataforma, incluyendo el backlog, el tablero Kanban y el cronograma. En el tablero por defecto se incluyen tres estados: por hacer, en curso y listo. Para alinear mejor la metodología con lo aprendido durante la carrera en diferentes asignaturas, se han agregado dos estados adicionales: "especificar requisitos" y "pasar pruebas de aceptación". El nuevo orden de los estados en el tablero es especificar requisitos, por hacer, en curso, pasar pruebas de aceptación y listo. Además, se ha actualizado el flujo de trabajo en Jira para reflejar estas modificaciones, asegurando que las transiciones entre estados estén correctamente configuradas (ver Figura 23).

Al inicio de un sprint, las actividades a desarrollar durante el mismo deberían estar ya deberían estar seleccionadas e incluidas en el tablero listas para empezar a trabajar en ellas. Esto quiere decir que para cada tarea en al iniciar el sprint se debe de haber incluido una descripción detallada de la tarea, un mock up o boceto de como debería de visualizarse al finalizar la implementación y una serie de pruebas de aceptación que la tarea deberá de satisfacer antes de que se finalice. Una vez una tarea ha sido dotada de esta información pasará al estado “Por hacer” de donde pasarán a “En curso” una vez se empiece a implementar la tarea.

Cuando el desarrollo de una tarea ha finalizado, la tarea pasa al estado “Pasar pruebas aceptación” donde se verifica que la implementación cumple con las pruebas de aceptación definidas con anterioridad. En el caso de que alguna de las pruebas de aceptación no fuera satisfecha, la tarea volvería la fase de desarrollo donde se trataría de solucionar el problema. En el caso contrario, si todas las pruebas se satisfacen, la tarea pasaria al estado de “Listo” lo que significaría que la tarea ha sido completada con éxito.

En la imagen (ver Figura 24) se pude observar una tarea con su breve explicación, su boceto y sus pruebas de aceptación:

Figura 24. Unidad de trabajo definida en Jira

Los sprints tendrían una duración máxima de un mes con la opción de que finalizasen antes si todas las tareas se implementaran y pasaran todas las pruebas de aceptación antes de la fecha de finalización del sprint.

3.3 Cronología del desarrollo

En la figura 25 se puede observar como se dividieron los sprints incluyendo su fecha de inicio y su fecha de fin asi como la validación del MVP desarrollado a la conclusión de los sprints.

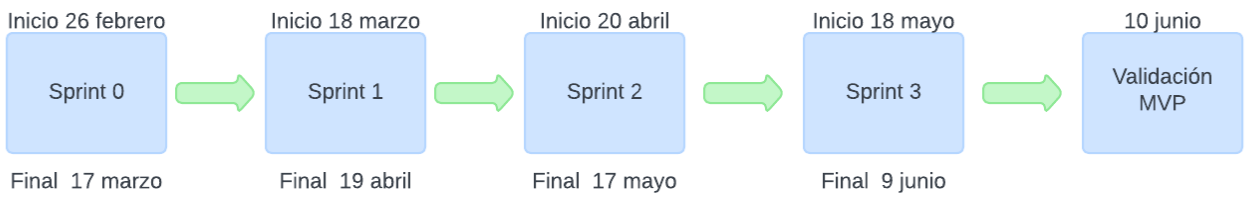


Figura 25. Cronología de desarrollo

En el sprint 0, se incluyeron las siguientes tareas: estudio de la competencia, análisis DAFO, formación en tecnología, inicialización del proyecto, conexión de la base de datos y creación de un conjunto de datos para pruebas, así como el poblamiento inicial del backlog. Fue un periodo de aprendizaje rápido en el que me tuve que familiarizar con las tecnologías en particular con React Native que supuso una curva de aprendizaje pronunciada en el inicio pero finalmente acabe haciendome a la tecnología.

En el sprint 1, se realizaron las siguientes tareas: implementación del temporizador (UT2), desarrollo del sistema de login/signup (UT1), creación de una funcionalidad para crear actividades (UT9), desarrollo del sistema de ranking (UT8) y la integración de una lista de tareas pendientes (UT10).

En el sprint 2, se llevaron a cabo las siguientes actividades: implementación de recompensas por horas completadas (UT11), configuración de estadísticas (UT6), desarrollo de la funcionalidad para crear un castillo (UT12), mejoras adicionales al castillo (UT13) y resolución de un bug que apareció después del primer sprint relacionado con la visualización de actividades creadas en la base de datos en la lista de tareas pendientes (BUG1).

En el sprint 3, se enfocaron en la mejora gráfica del sistema (UT14), implementación de animación durante el registro del tiempo (UT4), desarrollo de un sistema de ranking (UT7), creación de gráficos para visualizar los últimos 7 días y la implementación de una funcionalidad para bloquear el móvil o cancelar el temporizador si se abandona la aplicación (UT5).

En cada sprint se llevó a cabo la especificación, desarrollo y testing de las unidades de trabajo (UT) correspondientes. Sin embargo, para mejorar la exposición de estas actividades en la memoria, se han agrupado en capítulos separados para cada una de estas actividades. Esto no significa que se haya seguido un proceso secuencial, sino que se ha organizado de esta manera para una mejor comprensión y presentación del trabajo realizado. En la tabla se puede apreciar con claridad lo comentado.

Sprint 1	**Sprint 2**	**Sprint 3**
UT1: Login/signup	UT6: Estadísticas	UT4: Animación durante el registro de tiempo
UT2: Temporizador	UT10: To do list	UT5: Cancelar temporizador al salir de la pantalla
UT3: Asociar temporizador a actividad	UT11: Dar recompensas por horas completadas	UT7: Gráfico últimos 7 días
UT8: Ranking	UT12: Crear castillo	UT13: Mejoras castillo
UT9: Crear una actividad	BUG1: Tareas creadas no se muestran en la To do List	UT14: Mejora gráfica

Tabla 4: Organización UTs por sprint

4. Herramientas

A continuación realizaré una breve descripción de las herramientas utilizadas para el desarrollo de la aplicación.

TypeScript 

"TypeScript[1] es un lenguaje de programación fuertemente tipado que se basa en JavaScript, ofreciéndole mejores herramientas a cualquier escala." [12]

Aquí hay un debate que parece dividir a los programadores. Hay muchos que prefieren trabajar con JavaScript sin embargo yo he optado por TypeScript. Al ser un lenguaje con tipado estático, typescript permite detetectar errores en tiempo de compilación haciendo que el código se escriba sea más mantenible. Typescript tambien da un soporte más estruturado y robusto para la programación orientada a objetos [13]. Por estos motivos he optado por utilizar TypeScript en lugar de JavaScript.

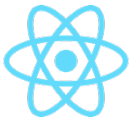
Node.js 

"Node.js[2] es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos"[14]

Se ha usado node para la ejecucion de la aplicación, la instalación de paquetes y las llamadas a base de datos.

[1] Pagina de TypeScript: https://www.typescriptlang.org/[1]

[2] Página de Node JS: https://nodejs.org/

React Native 

"React Native[3], es un framework de código abierto creado por Meta Platforms, Inc. Se utiliza para desarrollar aplicaciones para Android, Android TV, iOS, macOS, tvOS, Web, Windows y UWP al permitir que los desarrolladores usen React con las características nativas de estas plataformas." [15]

React Native permite desarrollar aplicaciones para móviles utilizando JavaScript o TypeScript lo cual facilita el proceso de desarrollo y elimina la necesidad de aprender un nuevo lenguaje a proposito para desarrollar en móvil como podría ser el caso con Kotlin, Swift o Flutter. A diferencia de los dos primeros lenguajes mencionados, con React Native podemos usar una misma base de código tanto para aplicaciones Android como iOS.

Expo 

"Expo[4] es un framework que facilita el desarrollo de aplicaciones para Android e iOS. Nuestro framework proporciona enrutamiento basado en archivos, una biblioteca estándar de módulos nativos y mucho más. Expo es de código abierto y cuenta con una comunidad activa en GitHub y Discord." [16]

Expo ayuda a agilizar el desarrollo y facilita el ver cambios lo más rapido posible con la funcionalidad de "hot reaload" la cual permite ver cambios guardados al momento. Esto hace que la creacion de prototipos sea fluida y la modificación de componentes mucho más eficiente. En combinación con su app Expo Go de móvil que permite ver tu aplicación en un entorno real y hacer cambios en el código sobre la marcha, Expo se convierte en una herraienta casi esencial para el desarrollo de aplicaciones móvil con React Native.

[3] Página de React Native: https://reactnative.dev/

[4] Página de Expo: https://expo.dev/

Firebase

Firebase[5] es una plataforma de desarrollo en la nube que ofrece una amplia gama de herramientas y servicios para gestionar y utilizar en aplicaciones móviles y web. Entre otros muchos servicios permite gestionar bases de datos, autenticar usuarios, gestionar almacenamiento, enviar notificaciones, analizar datos de tu aplicación y mucho más, todo ello desde una única plataforma.

Se ha optado por Firebase en lugar de otras plataformas que ofrecen funcionalidades parecidas debido a que ya estoy familiarizado con diferentes servicios que proporcionan.

Se ha usado el servicio Cloud Firestore que proporciona una base de datos NoSQL en la nube que te permite almacenar, sincronizar y consultar datos de forma sencilla. Esta organizada como se observa en la Figura 26.

Figura 26. Organización NoSql de Firebase

Una colección contiene documentos y estos contienen pares clave-valor como si de una estructura de datos hashmap se tratara. A nivel más tecnico los datos se guardan en archivos JSON que tienen una estructura flexible y adaptable a las necesidades de los datos de la aplicación. Otra de las funcionalidades más interesantes de la Cloud Firestore es la sincronización en tiempo real gracias a la cual mantiene los datos sincronizados en todos

[5] Página de Firebase: https://firebase.google.com/

los dispositivos conectados, incluso sin conexión a internet.

Otro servicio de Firebase que se ha utilizado es Firebase Authentication el cual permite autenticar a los usuarios de tu aplicación de forma segura y sencilla.

Github 

Github[6] es una plataforma gratuita para alojar el código de tus proyectos que utiliza el sistema de control de versionePas Git. Es una herramienta de gran ayuda a la hora de organizar y mantener un registro de los cambios realizados en el código permitiendo al usuario revertir cambios volviendo a versiones anteriores asi como revisar el codigo lo que mejora la calidad general del software y fomenta buenas prácticas. Github es uno de los estandares de la industria y el producto de su estilo más populares.

Visual Studio Code 

Visual Studio Code[7] es un entorno de desarrollo de código abierto altamente personalizable, diseñado para satisfacer las necesidades de programadores de diferentes lenguajes. Ofrece una interfaz intuitiva y eficiente, con características básicas y flexibles adaptandose a cualquier leguaje. Visual Studio Code es uno de los editores de texto más usados en la industria. He optado por usarlo debido a que estoy familiarizado con el y ofrece un alto nivel de compatibilidad tanto con TypeScript como con React Native.

Jira 

Jira[8] es una herramienta de gestión de proyectos ampliamente utilizada en la industria del desarrollo de software. Desarrollada por Atlassian, Jira permite a los equipos planificar y gestionar el trabajo de forma ágil. He utilizado Jira para implementar metodologías Scrum, creando sprints para gestionar ciclos de trabajo iterativos y controlados. Además, he empleado el tablero Kanban que proporciona Jira para personalizar y visualizar el flujo de trabajo para obtener una visión clara de las tareas en progreso, completadas y pendientes.

---

[6] Página de Github: https://github.com/

[7] Página de Visual Studio Code: https://code.visualstudio.com/

[8] Página de Jira: https://www.atlassian.com/es/software/jira

5. Especificación requisitos

En esta sección, se presenta un análisis detallado de los requisitos del sistema. Se incluirá tanto el análisis como la especificación de estos requisitos, proporcionando una visión clara de las necesidades y expectativas del proyecto.

5.1 Análisis de requisitos

Esta subsección se centra en el análisis detallado de los requisitos del sistema, identificando y documentando las necesidades específicas de los usuarios y otros actores involucrados. Se incluye un diagrama de casos de uso para representar gráficamente las interacciones entre los usuarios y el sistema. Este diagrama facilita la comprensión de los distintos escenarios de uso, asegurando que se consideren todas las funcionalidades necesarias durante el desarrollo del proyecto.

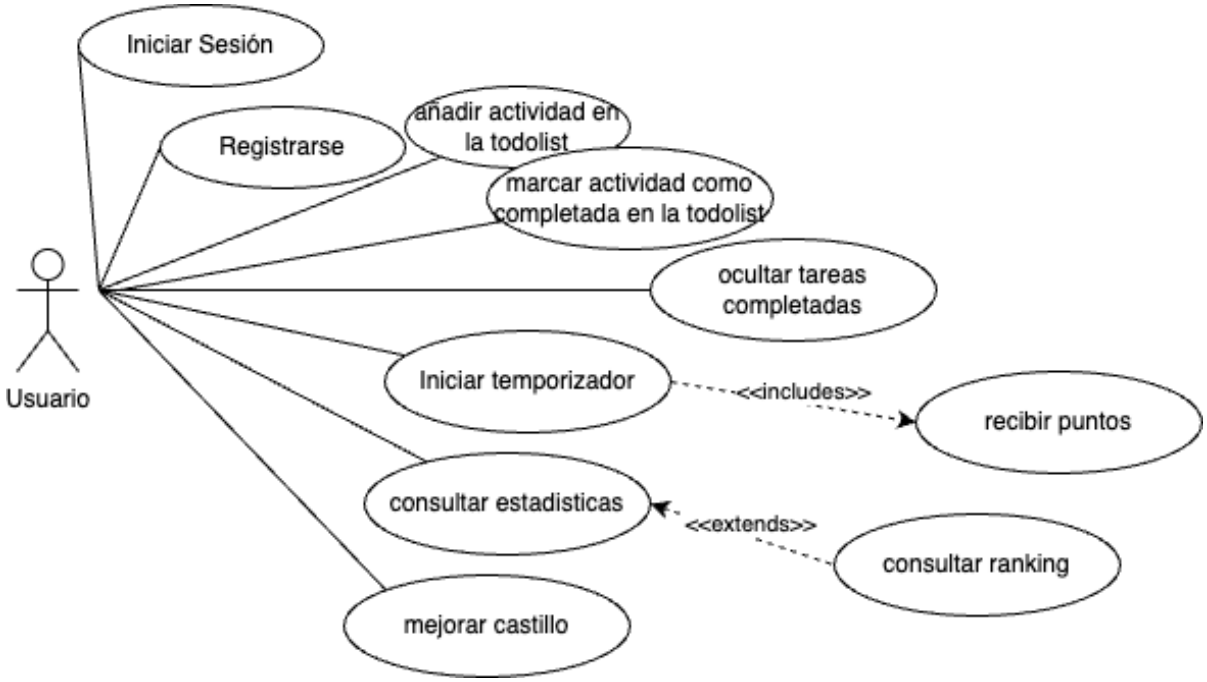


Figura 27. Diagrama de casos de uso

En el diagrama de casos de uso (ver Figura 27) se han identificado las principales interacciones del usuario con el sistema. Estos casos de uso incluyen acciones esenciales como iniciar sesión, registrarse, añadir actividades a la lista de tareas, marcar actividades como completadas, ocultar tareas completadas, iniciar el temporizador, consultar estadísticas, mejorar el

castillo y consultar el ranking. Además, se muestra la relación de inclusión entre iniciar el temporizador y recibir puntos, así como la extensión en consultar estadísticas para incluir la funcionalidad de consultar el ranking que se alojará en la misma página.

De estos casos de uso se extraerán las unidades de trabajo que serán añadidas al backlog y subsecuentemente a los correspondientes sprints. Este análisis permite estructurar y priorizar las tareas de desarrollo de manera efectiva, asegurando que todas las funcionalidades necesarias sean cubiertas y que la experiencia del usuario sea coherente y fluida. El diagrama también ayuda a visualizar la interdependencia entre diferentes funcionalidades, lo que facilita la planificación y coordinación de las actividades de desarrollo.

Una vez identificadas las funciones principales emepecé a plantearme la forma de estructurar los datos que iban a ser necesarios para desarrollar los diferentes casos de uso. Con lo que se elaboró el siguiente modelo de datos:

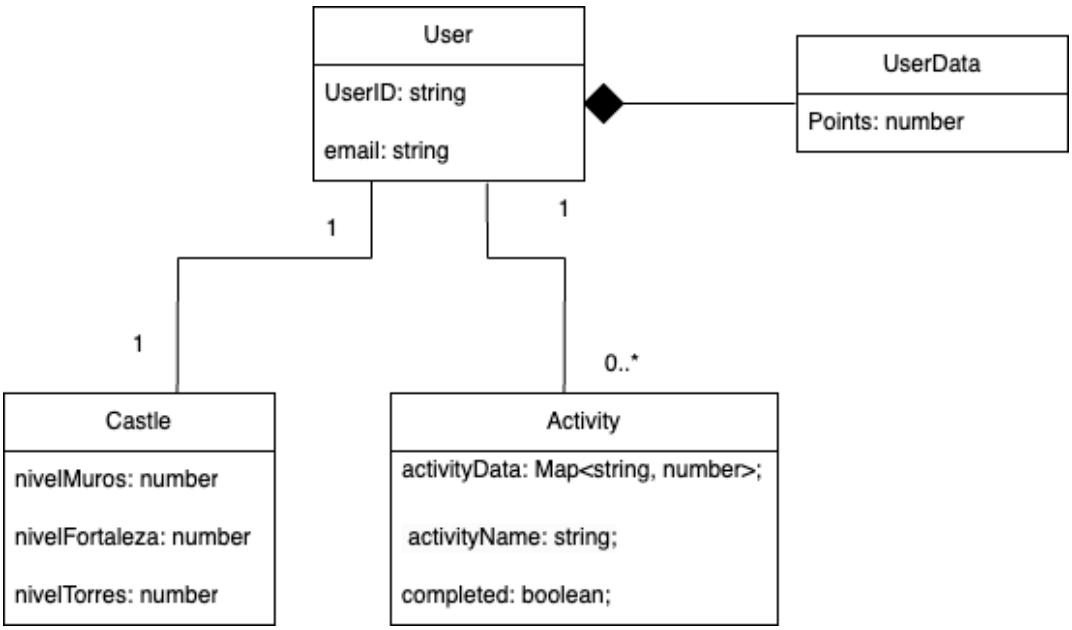


Figura 28. Modelo de datos

Como se puede ver en la figura 28, se ha tomado alguna decisión un tanto extraña a la hora de modelar los datos, pero esto se debe a la naturaleza no SQL de la base de datos Firestore de Firebase. En este modelo, un usuario (User) está compuesto por un único userData que almacena información adicional, tiene una relación de uno a uno con un Castle, y puede tener de cero a muchas actividades (Activity). También se ha optado por una aproximación ligera a la hora de guardar datos en la base de datos. Se ha optado por almacenar datos básicos y esenciales y calcular datos a raíz de estos. Un ejemplo de esto es el Ranking, el cual recoge estadisticas de las actividades de los usuarios para calcular el tiempo usado para calcular los puestos. Este diseño refleja la flexibilidad y la estructura jerárquica de

Firestore, permitiendo almacenar datos de manera más intuitiva y escalable para aplicaciones con necesidades dinámicas y cambiantes.

5.2 Especificación de requisitos

En esta sección se van a detallar las unidades de trabajo que se identificaron tras el analisis y que fueron incluidas en alguno de los sprints.

Sprint 1	Sprint 2	Sprint 3
UT1	UT6	UT4
UT2	UT10	UT5
UT3	UT11	UT7
UT8	UT12	UT13
UT9	BUG1	UT14

Tabla 5. Distribución de las unidades de trabajo en sprints

En la tabla se puede observar que tareas fueron incluidas en cada sprint. Cabe aclarar que la unidad de trabajo identificada como “BUG1” se creo a raíz de un errror que surgió en el código después del primer sprint.

UT1: Login/signup

Descripción: Implementar el sistema de registro y login, permitiendo al usuario registrarse y acceder a la aplicación utilizando un correo electrónico y contraseña propios, además de la opción de iniciar sesión con Google.

Pruebas de Aceptación:

- El usuario puede registrarse correctamente utilizando un correo electrónico y contraseña.
- El usuario puede iniciar sesión con las credenciales registradas.
- El usuario puede iniciar sesión utilizando su cuenta de Google.

Mock up:

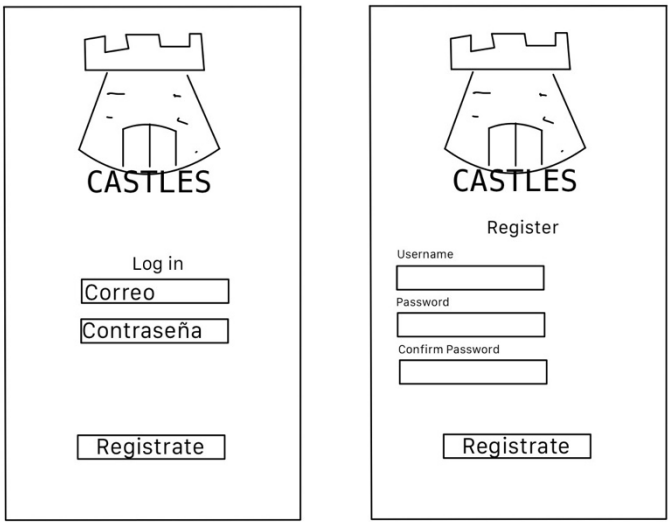


Figura 29. Boceto UT1

UT2: Temporizador

Descripción: Sección para registrar tiempo mediante un temporizador. El usuario podrá elegir la duración del mismo.

Pruebas de Aceptación:

- El usuario puede iniciar el temporizador con una duración específica.
- El usuario puede detener el temporizador y registrar el tiempo transcurrido.

Mock up:

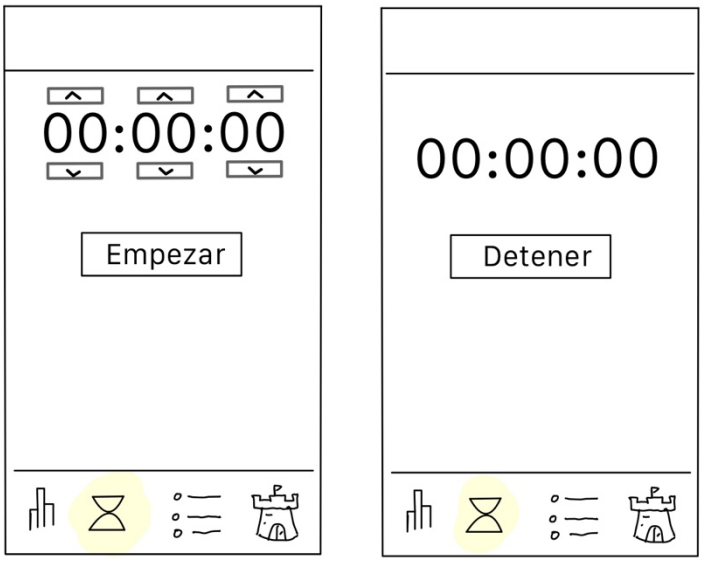


Figura 30. Boceto UT2

UT3: Asociar temporizador a actividad

Descripción: Asociar una actividad al temporizador para que se registre el tiempo.

Pruebas de Aceptación:

- El usuario puede asociar un temporizador a una actividad seleccionada.
- El sistema registra automáticamente el tiempo transcurrido para la actividad asociada.

Mock up:

Figura 31. Boceto UT3

UT4: Animación durante el registro de tiempo

Descripción: Animación de construcción de un castillo mientras se ejecuta el temporizador.

Pruebas de Aceptación:

- El temporizador muestra una animación visual durante su ejecución.
- La animación no ralentiza la aplicación.

Mock up:

Figura 32. Boceto UT4

UT5: Cancelar temporizador al salir de la pantalla

Descripción: Si el usuario sale de la pantalla del temporizador, éste se detendrá. Incluir un texto en el temporizador cuando esté en marcha avisando.

Pruebas de Aceptación:

- Al salir, se mostrará un pop-up con el mensaje anunciando que se ha detenido el temporizador.

Mock up:

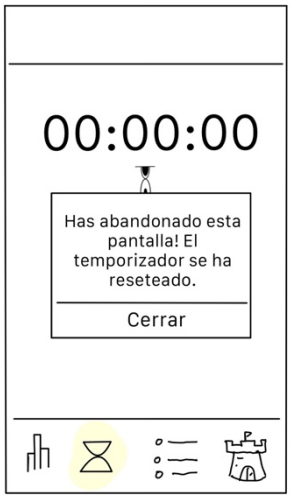


Figura 33. Boceto UT5

UT6: Estadísticas

Descripción: Página donde ver estadísticas generales como horas totales, horas del último año, día, etc.

Pruebas de Aceptación:

- El usuario puede ver claramente las estadísticas definidas.
- Las estadísticas se actualizan una vez se ha registrado tiempo.
- El tiempo de carga es inferior a 1 segundo.

Mock up:

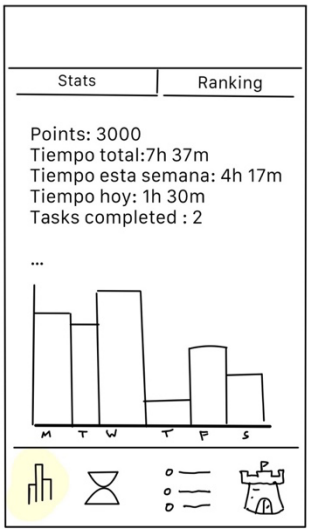


Figura 34. Boceto UT6

UT7: Gráfico últimos 7 días:

Descripción: Mostrar un grafico que ilustre la actividad en horas realizada por el usuario en los ultimos 7 días.

Pruebas de Aceptación:

- El gráfico muestra correctamente las horas registradas por el usuario en cada uno de los últimos siete días.
- Los datos del gráfico se actualizan en tiempo real cuando el usuario registra nuevas horas.

Mock up:

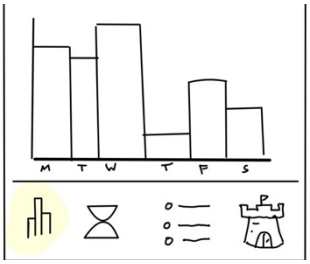


Figura 35. Boceto UT7

UT8: Ranking

Descripción: Ranking en función de las horas que se ha concentrado el usuario en comparación con sus amigos.

Pruebas de Aceptación:

- El ranking muestra de manera precisa y actualizada la posición del usuario en comparación con sus amigos basado en las horas dedicadas a las actividades registradas.
- Las posiciones en el ranking se actualizan en tiempo real o después de cada sesión registrada.

Mock up:

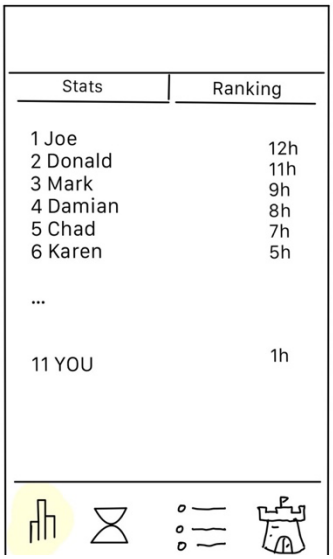


Figura 36. Boceto UT8

UT9: Crear una actividad

Descripción: Crear una actividad con nombre a la que poder registrar tiempo.

Pruebas de Aceptación:
El usuario puede crear una nueva actividad con un nombre descriptivo.
La actividad creada se añade correctamente a la lista de tareas pendientes.

Mock up:

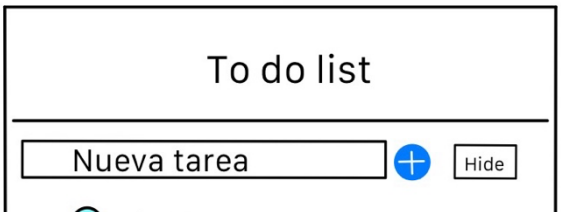


Figura 37. Boceto UT9

UT10: To do list

Descripción: To-do list donde se muestran todas las tareas tanto completadas como en curso, y las completadas se muestran tachadas o marcadas de alguna manera.

Pruebas de Aceptación:
- Todas las tareas completadas se muestran tachadas o con una marca distintiva.
- Las tareas en curso se visualizan sin ninguna modificación visual.

Mock up:

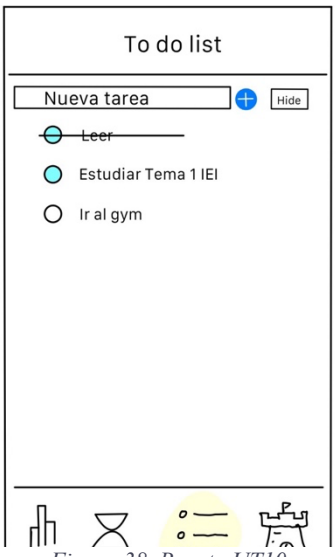


Figura 38. Boceto UT10

UT11: Dar recompensas por horas completadas

Descripción: Dar x número de monedas o puntos por cada hora completada.

Pruebas de Aceptación:

- El usuario recibe una cantidad específica de monedas por cada hora de actividad completada.
- Las recompensas por horas completadas se asignan correctamente según el registro de tiempo.

Mock up:

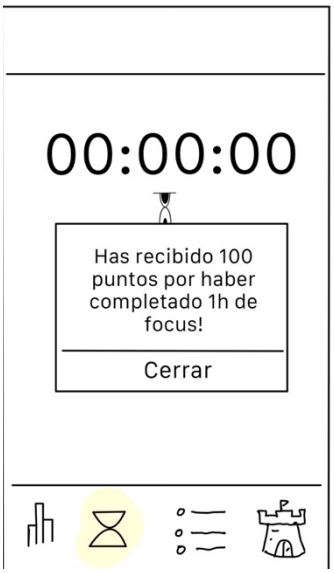


Figura 39. Boceto UT11

UT12: Crear castillo

Descripción: Crear la sección del castillo y mostrar el mismo junto a las torres, fortaleza y muros.

Pruebas de Aceptación:
El usuario podrá ver en la pantalla su castillo y los componentes del mismo.
Los elementos del castillo (torres, fortaleza, muros) se visualizan correctamente y de manera cohesiva.

Mock up:

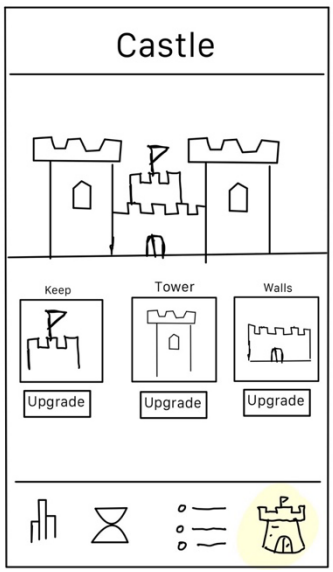


Figura 40. Boceto UT12

UT13: Mejoras castillo

Descripción: Añadir funcionalidad para mejorar las partes del castillo a cambio de puntos.

Pruebas de Aceptación:

- El usuario puede canjear sus puntos por mejoras de nivel para las características del castillo.
- Las mejoras adquiridas se reflejan adecuadamente en el castillo del usuario.

Mock up:

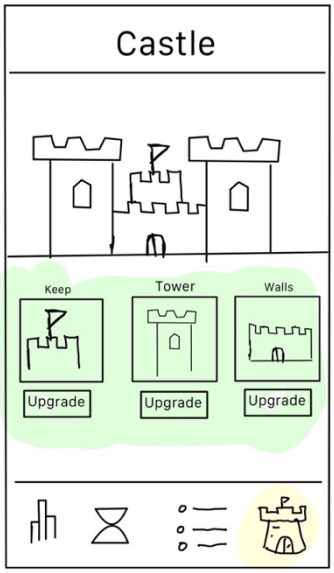


Figura 41. Boceto UT13

UT14: Mejora gráfica

Descripción: Actualizar y mejorar la interfaz gráfica de todas las pantallas de la aplicación para asegurar una experiencia de usuario más atractiva y moderna.

Pruebas de Aceptación:

- Todas las pantallas de la aplicación presentan un diseño gráfico mejorado y consistente.
- Las mejoras gráficas no afectan negativamente el rendimiento de la aplicación.
- Los usuarios pueden navegar por la aplicación de manera intuitiva y sin dificultades.

BUG 1: Tareas creadas no se muestran en la To do List

Severidad: Mayor

Prioridad: Alta

Descripción: Tras el desarrollo de la funcionalidad de la Lista de Tareas (To-Do List), durante una reunión con el tutor, se encontró un bug en el que las actividades creadas se reflejan en la base de datos pero no se muestran en la lista de tareas. Este problema se rastreó hasta una implementación incorrecta del método en el patrón de repositorio, que es responsable de obtener las actividades de la base de datos.

Pasos para Reproducir:
1. Iniciar sesión en la aplicación.
2. Navegar a la sección de la Lista de Tareas.
3. Crear una nueva actividad y guardarla.
4. Observar que la nueva actividad creada no aparece en la lista de tareas.

Resultados Actuales: Las nuevas actividades creadas no se muestran en la lista de tareas a pesar de haberse guardado correctamente en la base de datos.

Resultados Esperados: Todas las actividades creadas y guardadas en la base de datos deberían mostrarse en la lista de tareas.

Cabe comentar las dos últimas unidades de trabajo: la mejora gráfica y el bug identificado. La unidad de trabajo de la mejora gráfica no implementa ninguna funcionalidad nueva, se enfoca en actualizar y mejorar la interfaz de usuario para asegurar una experiencia más atractiva y moderna. Para abordar el bug, se aplicó el procedimiento de documentación aprendido durante la carrera, que incluyó la identificación del problema, la documentación exhaustiva del bug, la replicación de los pasos necesarios para reproducirlo y la implementación de una solución efectiva.

6. Diseño

En esta sección se describen las técnicas y el diseño de la aplicación, destacando la arquitectura utilizada y los patrones de diseño aplicados.

6.1 Modelo Vista Controlador

El modelo Vista Controlador (MVC) [17] es un patrón de diseño de software que separa la aplicación en tres componentes principales:

- Modelo: Representa la lógica de datos de la aplicación y las reglas de negocio. Maneja directamente los datos y la lógica, incluyendo la recuperación y actualización de datos en la base de datos.

- Vista: Es la representación visual de los datos que se encuentran en el modelo. La vista se encarga de presentar la información al usuario y de recoger las interacciones del usuario, como clics o entradas de datos.

- Controlador: Actúa como intermediario entre el modelo y la vista. Recibe las entradas del usuario desde la vista, procesa la entrada mediante la lógica del modelo, y devuelve la salida a la vista para su presentación.

Este patrón facilita la gestión y el mantenimiento del código, ya que permite separar claramente las responsabilidades de cada componente, mejorando la modularidad y la escalabilidad de la aplicación. Como curiosidad el patrón MVC antecede a la web y fue originalmente conceptualizado a finales de la década de 1970 por Trygve Reenskaug. Su propósito inicial era crear un marco para desarrollar interfaces gráficas de usuario que pudiera separar la lógica de la interfaz de usuario de la lógica de negocio, un concepto que ha demostrado ser invaluable y sigue siendo ampliamente utilizado en el desarrollo moderno de software y web. En la figura 42 se puede observar la implementación.

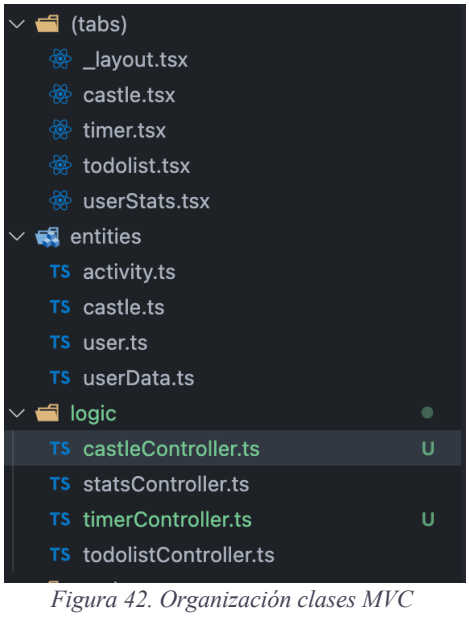


Figura 42. Organización clases MVC

En la carpeta “tabs” estan definidas las diferentes pantallas de la aplicación ya que el router de Expo, que permite la navgación sencilla entre ventanas en la aplicación, en la carpeta logic se encuentra toda la logica de negocio de los diferentes componentes de la app y en la carpeta entities se encuentran los modelos de objetos. Aquí un pequeño ejemplo de la lógica implementada y parte de una de las vistas (ver Figuras 43 y 44):

```
async getTotalTimePerActivity(uid: string) {
        const activities = await this.activityRepo.getAll(uid);
        const totalTimePerActivity = new Map<string, number>();
        activities.forEach((activity) => {
            const totalTime = activity.getTotalTime();
            const activityName = activity.getActivityName();
            if (totalTimePerActivity.has(activityName)) {
                const currentTotalTime = totalTimePerActivity.get(activityName);
                if (currentTotalTime) {
                    totalTimePerActivity.set(activityName, currentTotalTime + totalTime);
                }
            } else {
                totalTimePerActivity.set(activityName, totalTime);
            }
        });
        return totalTimePerActivity;
    }
        You, 6 days ago • a little button on the todolist :)
```

Figura 43. Ejemplo logica

```
const filteredTasks = showCompleted ? tasks : tasks.filter(task => !task.getCompleted());

return (
    <View style={styles.container}>
        <StatusBar barStyle="light-content" />

        <View style={styles.inputContainer}>
            <TextInput
                style={styles.input}
                placeholder="Enter task"
                value={newTaskTitle}
                onChangeText={setNewTaskTitle}
                placeholderTextColor="#6200EE"
            />
            <TouchableOpacity style={styles.addButton} onPress={addTaskToDB}>
                <Text style={styles.addButtonText}>Add</Text>
            </TouchableOpacity>
        </View>
        <TouchableOpacity style={styles.toggleButton} onPress={() => setShowCompleted(!showCompleted)}>
            <Text style={styles.toggleButtonText}>{showCompleted ? "Hide Completed Tasks" : "Show Completed Tasks"}</Text>
        </TouchableOpacity>
        <FlatList
            style={styles.taskList}
            data={filteredTasks}
            renderItem={renderTask}
            keyExtractor={(item) => item.getActivityName()}
        />
    </View>
);
```

Figura 44. Ejemplo presentación

6.2 Patrón Repositorio

El patrón Repositorio es un diseño que sirve como intermediario entre la lógica de negocio y la capa de acceso a datos, ofreciendo una abstracción sobre las operaciones de almacenamiento, recuperación y gestión de datos. Este patrón permite encapsular la lógica necesaria para acceder a la fuente de datos, manteniendo el código de la lógica de negocio enfocado y limpio.

Principales Componentes del Patrón Repositorio:

- Repositorio: Define la interfaz y la implementación para interactuar con los datos, proporcionando métodos para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre la fuente de datos.
- Unidad de Trabajo: Gestiona las transacciones, asegurando que todas las operaciones de datos se completen con éxito o se reviertan en caso de error, manteniendo la integridad de los datos.

Este patrón mejora la organización y la mantenibilidad del código, permitiendo realizar cambios en la capa de datos sin afectar a la lógica de negocio. Además, facilita las pruebas unitarias al permitir el uso de repositorios simulados.

En mi implementación del patrón, empecé por definir una interfaz genérica (ver Figura 45) para las operaciones del repositorio en el archivo 'iRepo.ts'. De esta manera se proporciona un diseño desacoplado y modular.

app > persistance > TS iRepo.ts > iRepo

You, last month | 1 author (You)

```
1  interface iRepo<T> {      You, last month • volver a empezar
2      getById(id: string): Promise<T | null>;
3      getAll(uid: string): Promise<T[]>;
4      add(item: T): Promise<void>;
5      update(item: T): Promise<void>;
6      remove(id: string): Promise<void>;
7  }
```

Figura 45. Interfaz genérica del patrón repositorio

A continuación para cada objeto se definió una interfaz específica para cada para cada objeto. Estas extienden la interfaz generica añadiendo métodos específicos para cada objeto. En la figura 46 esta 'iRepoActivity.ts' .

```
1  import { Activity } from "../entities/activity";
2
   You, last month | 1 author (You)
3  export interface iRepoActivity extends iRepo<Activity>{
4      getById(uid: string, activityName?: string): Promise<Activity | null>;
5      getAll(uid: string): Promise<Activity[]>;
6      add(item: Activity, uid?:string): any;
7      update(item: Activity): Promise<void>;
8      remove(id: string): Promise<void>;
```

Figura 46. Interfaz específica para objeto Actividad

Ahora solo faltaría la implementación de las diferentes interfaces específicas y sus respectivos métodos CRUD para que interactuaran con la base de datos.

6.3 Arquitectura

El patrón MVC predispone a organizar la arquitectura de la aplicación por capas siendo estas la capa presentación, la capa de lógica de negocio y la capa de persistencia.

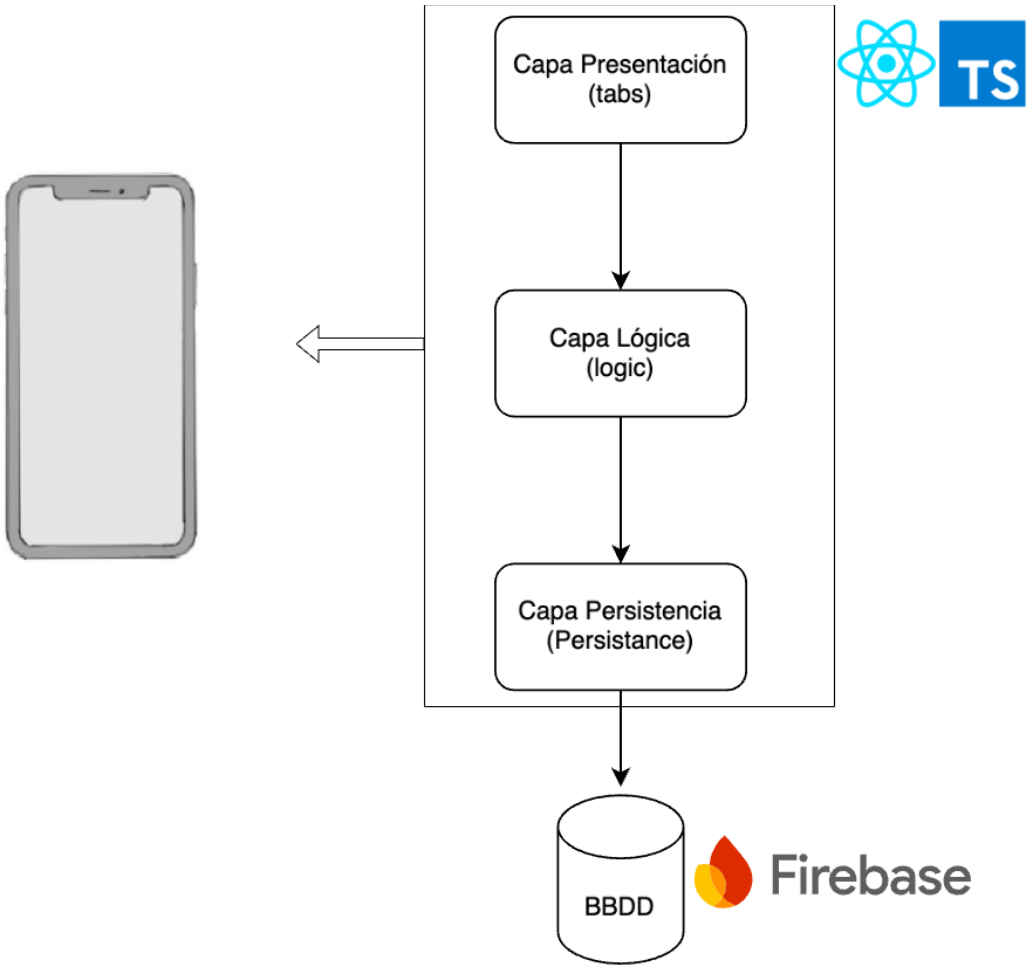


Figura 47. Gráfico arquitectura de la aplicación

Como se puede apreciar en la Figura 47 la aplicación muestra una clara separación de capas y responsabilidades, organizando los componentes de la interfaz de usuario en la carpeta tabs, la lógica de la aplicación en la carpeta logic y la persistencia de datos en persistence. Cada entidad como activity, castle, user y userData tiene sus propios repositorios y controladores, lo que

dota la aplicación con modlaridad como se ha comentado previamente. Conviene tambien explicar brevemente la estructura de las carpetas del proyecto (ver Figura 48).

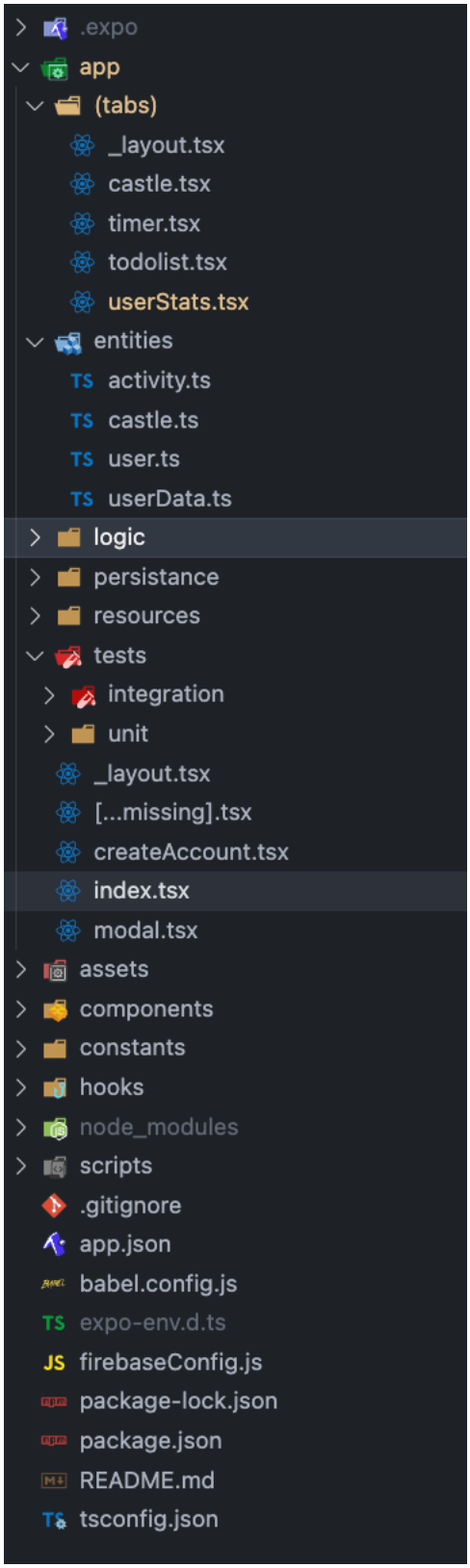


Figura 48. Organización carpetas del proyecto

La estructura de carpetas del proyecto está organizada de la siguiente manera para mantener una separación clara entre los diferentes componentes y funcionalidades de la aplicación:

- .expo: Carpeta que contiene archivos de configuración específica para Expo.

- app:
  - (tabs): Contiene los componentes principales de las diferentes pestañas de la aplicación.

  - _layout.tsx: Componente de layout para las pestañas.

  - entities: Contiene las modelos de datos utilizadas en la

  - logic: Carpeta destinada a la lógica de negocio de la aplicación. Aquí se manejan las operaciones y reglas de negocio.

  - persistance: Carpeta que contiene el código relacionado con la persistencia de datos, incluyendo la configuración y funciones para interactuar con Firebase Firestore.

  - resources: Carpeta destinada a almacenar recursos adicionales necesarios para la aplicación.

  - tests:
    - integration: Contiene las pruebas de integración.
    - unit: Contiene las pruebas unitarias.

- components: Carpeta que contiene componentes reutilizables dentro de la aplicación.

- constants: Carpeta que contiene archivos con constantes utilizadas en toda la aplicación.

- hooks: Carpeta destinada a los custom hooks utilizados en la aplicación.

- node_modules: Carpeta generada automáticamente que contiene los módulos de Node.js instalados.

- scripts: Carpeta que contiene scripts adicionales necesarios para el proyecto.

- .gitignore: Archivo que especifica qué archivos y directorios deben ser ignorados por Git.

- app.json: Archivo de configuración de la aplicación.

- babel.config.js: Configuración para Babel, el compilador de JavaScript.
- expo-env.d.ts: Archivo de declaración de tipos para Expo.
- firebaseConfig.js: Archivo de configuración para Firebase.
- package-lock.json: Archivo que guarda las versiones exactas de las dependencias instaladas.
- package.json: Archivo que contiene metadatos del proyecto y las dependencias.
- README.md: Archivo de documentación del proyecto.
- tsconfig.json: Archivo de configuración para TypeScript.

6.4 Base de datos

Como se comentó en el apartado de Herramientas, se va autilizar Firebase Firestore como base de datos. Esta tecnología se ha elegido debido a que ya estoy familiarizado con la utilización de esta plataforma. Firebase Firestore proporciona una base de datos NoSQL. NoSQL es un enfoque utilizado en el diseño de bases de datos que permite el almacenamiento y consulta de datos fuera de las estructuras tradicionales que se encuentran en las bases de datos relacionales [18].

A diferencia de las bases de datos relacionales, que organizan datos en tablas y utilizan SQL para las consultas, Firestore almacena datos en documentos (ver Figura 49), que son colecciones de pares clave-valor, anidados dentro de colecciones. Esto permite una mayor flexibilidad en la estructura de los datos y facilita la escalabilidad horizontal.

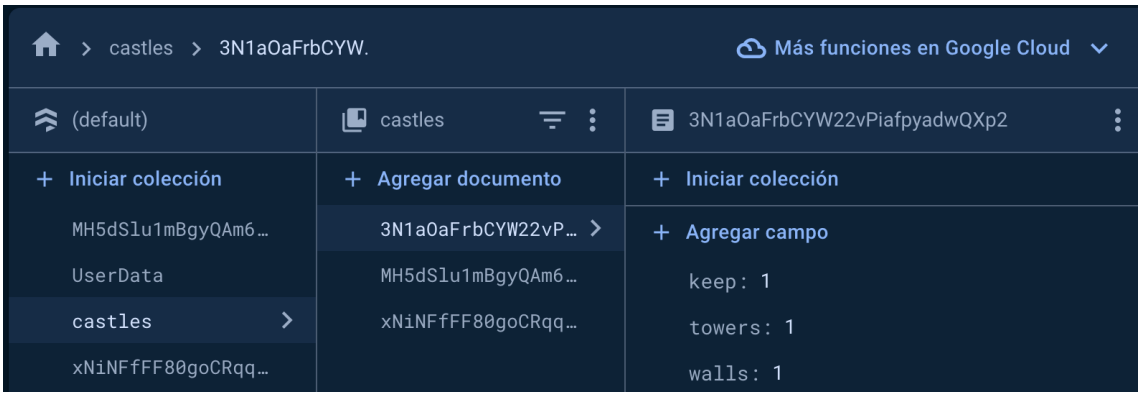


Figura 49. Organización datos en Firestore

Además de Firestore, se ha utilizado Firebase Authentication para autenticar y crear usuarios en la aplicación. Firebase Authentication ofrece

varios métodos de inicio de sesión pero se ha utilizado el clásico acceso mediante correo y contraseña. Se han coordinado Firestore y Authentication para que una vez se haya registrado un usuario Authentication cree un User ID (UID de ahora en adelante) que se almacenara como una colección que dentro contendrá las distintas actividades que el usuario vaya creando como documentos y dentro de esos documentos se guardara como par clave-valor la fecha y la cantidad de segundos registrados.

Dentro de las colecciones “castles” y “UserData” los documentos tienen por nombre el UID y dentro de ellos se almacenan los campos específicos relacionados con cada usuario. En la colección “castles”, los documentos contienen datos sobre las diferentes partes del castillo, como se muestra en la imagen adjunta:

Cada documento dentro de “castles” tiene campos como:

- keep: Representa el estado o nivel de la torre principal del castillo.
- towers: Representa el número de torres adicionales.
- walls: Indica el estado o nivel de los muros defensivos.

De manera similar, en la colección “UserData”, se almacenan otros datos relevantes sobre el usuario, organizados de manera que faciliten la gestión y consulta eficiente de la información.

Esta integración entre Firestore y Authentication no solo asegura que los datos del usuario estén bien estructurados y accesibles, sino que también permite una gestión segura y eficiente de la autenticación y los permisos de usuario, mejorando la experiencia general de la aplicación.

7. Implementación

En esta sección se describen los pasos y procesos llevados a cabo para la implementación de la aplicación, centrándonos en la conexión con la base de datos, la transferencia de datos al front-end y el desarrollo de las funcionalidades en el front-end.

7.1 Capa Persistencia

Para conectar la aplicación con Firebase Firestore, se utilizó la biblioteca oficial de Firebase, que permite una integración directa y eficiente con la base de datos. La conexión se estableció mediante la configuración del SDK de Firebase en el proyecto, utilizando las credenciales de la aplicación. Se inicializó Firebase en la aplicación con las credenciales proporcionadas en el panel de control de Firebase y se configuraron las reglas de seguridad para controlar el acceso a los datos, asegurando que solo los usuarios autenticados pudieran leer y escribir en la base de datos.

Se implementaron funciones para consultar y recuperar datos desde Firestore, utilizando promesas y async/await para manejar las operaciones asíncronas. Estas funciones fueron utilizadas en los patrones repositorio para acceder a las colecciones de datos. Los datos recuperados se estructuraron adecuadamente para ser utilizados en el front-end. En la figura de abajo se pueden observar algunas de las funciones implementadas siguiendo un modelo CRUD para interactuar con las colecciones de datos (ver Figura 50).

```
class dbfuncts {
    static async writeData(collection: string, docu: string, data: Record<string, any>) {
        try {
            await setDoc(doc(db, collection, docu), data);

            console.log("Document written");
        } catch (e) {
            console.error("Error adding document: ", e);
        }
    }      You, last month • volver a empezar

    static async readData(collection: string, docu: string) {
        try {
            const docRef = doc(db, collection, docu);
            const docSnap = await getDoc(docRef);

            if (docSnap.exists()) {
                console.log("Document data:", docSnap.data());
            } else {
                // docSnap.data() will be undefined in this case
                console.error("No such document!");
            }
            return docSnap.data()
        } catch (e) {
            console.error("Error reading data: ", e);
        }
    }

    static async deleteData(collection: string, docu: string) {…
    }

    static async updateData(collectionName: string, documentId: string, data: Map<string, any>): Promise<void> {…
    }
```

Figura 50. Funciones CRUD

La figura a continuación ilustra la comunicación entre las capas de la aplicación, destacando cómo interactúan la capa lógica, el repositorio y Firebase:

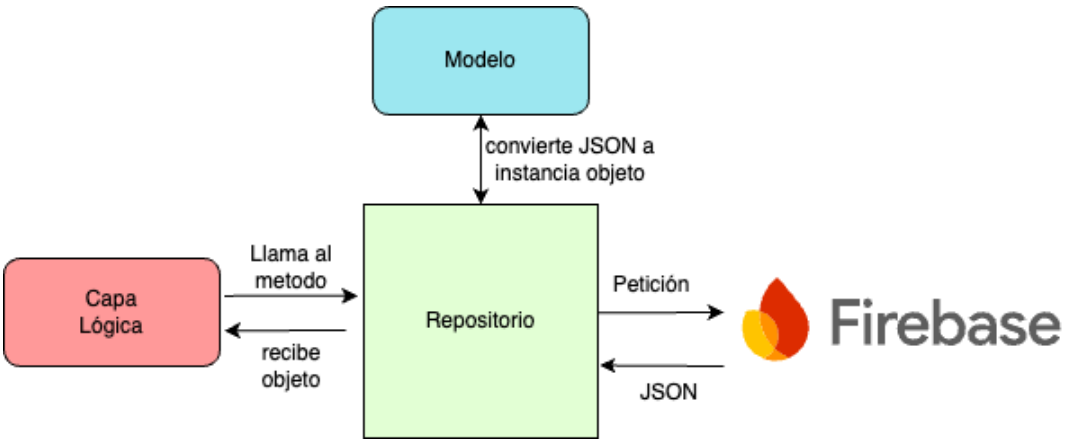


Figura 51. Interacción capas del sistema con base de datos

El funcionamiento de una llamada a un método del repositorio sería el siguiente:

1. **Capa Lógica a Repositorio**: La capa lógica llama a un método del repositorio, por ejemplo, para consultar un elemento buscándolo en base a su clave primaria.
2. **Repositorio a Firebase**: El método en el repositorio envía una petición a Firebase Firestore para obtener la información de dicho elemento.
3. **Firebase a Repositorio**: Firebase Firestore devuelve la información solicitada en formato JSON al repositorio.
4. **Repositorio a Modelo**: El repositorio construye un objeto basandose en la clase modelo usando pasando la información obtenida de Firebase.
5. **Repositorio a Capa Lógica**: El repositorio obtiene el objeto de la clase modelo y lo devuelve a la capa lógica.

El uso de estas clases modelo y su mapeo mediante los repositorios en el proyecto resulta sumamente útil. Gracias a TypeScript, que permite el uso de tipos, se ha facilitado el proceso de depuración al conocer las propiedades de cada uno de los objetos con los que se opera. Este tipado estricto no solo ha prevenido multitud de errores, sino que también ha hecho la solución más mantenible.

7.2 Capa Lógica

La lógica de negocio es una parte fundamental en la arquitectura de software. Esta capa es crucial porque separa las funciones dentro de la aplicación, actuando como intermediaria entre la capa de presentación y la capa de persistencia facilitando la escalabilidad y la modularidad en la aplicación

En el contexto de nuestra aplicación, la lógica de negocio está

implementada a través de diversas clases controladoras. A continuación, se explica el funcionamiento de la lógica detrás de las principales pantallas de la aplicación.

- CastleController: se encarga de manejar la gestión del castillo en la aplicación, una funcionalidad que forma parte de la gamificación del producto. Mediante el repositorio correspondiente, se recuperan los datos del castillo de la base de datos y se monta un objeto para despues visualizarlo en la capa presentación. Esta clase también contiene la lógica responsible de gestionar la interacción del usuario con el castillo con funciones para subir de nivel las partes del castillo.

- TimerController: la gran mayoría de la complexión de este componente es visual como puede ser el mostrar el temporizador empezarlo y pausarlo asi como la animación y el finalizar el temporizador si se abandona la ventana del temporizador. En este controlador encontramos funciones para calcular los puntos que recibirá el usuario y las interacciones con el repositorio para recuperar y persistir los datos generados en esta ventana como el tiempo registrado y los puntos.

- ToDoListController: en esta se aborda la lógica necesaria para alterar el estado de las actividades en base de datos. Destacan 3 funciones principales todas ellas asíncronas. La primera de ellas ‘getTasks’ recupera la lista de tareas usando los metodos del repositorio. La función ‘addTask’ permite añadir nuevas tareas a la base de datos. La función ‘toggleTaskCompletion’ permite marcar las tareas como completadas o incompletas.

- StatsController: es responsable de gestionar y procesar las estadísticas de los usuarios. Esta clase maneja varias funcionalidades clave, como contar las actividades completadas, sumar el tiempo total dedicado a actividades y calcular el tiempo invertido en actividades durante la última semana. Esta última funcionalidad también se utiliza para calcular la posiciones en el ránking que tambien se visualiza en el correspondiente componente de la capa lógica.

7.3 Capa Presentación

A continuación, se detallan las clases que he programado para el proyecto, explicando cómo se implementaron y su propósito dentro de la aplicación. Las clases revisadas son castle.tsx, timer.tsx, todolist.tsx, userStats.tsx, index.tsx y createAccount.tsx.

El componente Castle se encarga de la visualización del castillo en la aplicación. Se muestra a los usuarios ver y mejorar su castillo. En su implementación, primero se define el estado inicial del castillo usando useState, estableciendo valores predeterminados para las diferentes partes del castillo como la torre principal, las torres y los muros. Luego, mediante useEffect, se recuperan los datos del castillo desde la capa inferior cuando el componente se monta. Esta función asíncrona obtiene los datos del documento correspondiente al usuario en la colección "castles" y actualiza el estado del castillo con los datos recuperados. Finalmente, se renderiza la información del castillo mostrando los diferentes componentes del mismo según los datos obtenidos.

La clase Timer proporciona una funcionalidad de temporizador que permite a los usuarios registrar el tiempo dedicado a diferentes actividades. Se define el estado inicial del temporizador con useState, estableciendo el contador de segundos y el estado de actividad del temporizador. Utilizando useEffect, se actualiza el tiempo cada segundo cuando el temporizador está activo. Este efecto se gestiona mediante un intervalo que se establece cuando el temporizador está activo y se limpia cuando se detiene. Las funciones handleStart y handleStop controlan el inicio y la detención del temporizador, respectivamente. El componente se renderiza mostrando el tiempo transcurrido y botones para iniciar y detener el temporizador.

La clase Todolist permite a los usuarios gestionar una lista de tareas, incluyendo la adición, visualización y marcación de tareas como completadas. Se define el estado inicial de las tareas usando useState. Mediante useEffect, se recuperan las tareas cuando el componente se monta. Esta función asíncrona obtiene los documentos correspondientes al usuario en la colección "tasks" y actualiza el estado de las tareas con los datos recuperados.

El componente UserStats muestra estadísticas del usuario, como el tiempo total registrado, el progreso en diversas actividades y un ránking de usuarios. Se define el estado inicial de las estadísticas usando useState. Utilizando useEffect, se recuperan las estadísticas cuando el componente se monta. Esta función asíncrona obtiene los datos del documento correspondiente al usuario en la colección "userStats" y actualiza el estado de las estadísticas con los datos recuperados. El componente se renderiza mostrando las estadísticas del usuario, incluyendo las horas totales y las horas registradas en los últimos siete días, proporcionando una visión clara del progreso del usuario.

La clase createAccount.tsx se encarga de manejar la funcionalidad de creación de cuentas en la aplicación, permitiendo a los usuarios registrarse mediante correo electrónico y contraseña. En la implementación, se utiliza el hook useState para manejar los estados del formulario de registro, como el correo electrónico, la contraseña y los posibles errores. Se define una función handleSubmit que se ejecuta cuando el usuario envía el formulario. Esta función utiliza el método createUserWithEmailAndPassword de Firebase Authentication para registrar al usuario. Si el registro es exitoso, el usuario es

redirigido a la página principal de la aplicación. En caso de error, se captura y se actualiza el estado de error para mostrar un mensaje al usuario. El componente se renderiza mostrando un formulario con campos para el correo electrónico y la contraseña, y un botón para enviar el formulario.

La clase index.tsx a parte de ser el punto de entrada de la app, se encarga de manejar la funcionalidad de inicio de sesión en la aplicación, permitiendo a los usuarios autenticarse mediante correo electrónico y contraseña. Esta funcionalidad es crucial para la seguridad y la gestión de sesiones de usuario.En la implementación, se utiliza el hook useState para manejar los estados del formulario de inicio de sesión, como el correo electrónico, la contraseña y los posibles errores. Se define una función handleSubmit que se ejecuta cuando el usuario envía el formulario. Esta función utiliza el método signInWithEmailAndPassword de Firebase Authentication para autenticar al usuario. Si la autenticación es exitosa, el usuario es redirigido a la página principal de la aplicación. En caso de error, se captura y se actualiza el estado de error para mostrar un mensaje al usuario. El componente se renderiza mostrando un formulario con campos para el correo electrónico y la contraseña, y un botón para enviar el formulario.

7.4 Resultado

Tras haber comentado la implementación completa de la aplicación, se procederá a mostrar las interfaces finales de la aplicación mediante capturas de pantalla de un dispoitivo iOS.

Cuando el usuario abre la aplicación, se le presenta el formulario de login asi como la opción de crear una cuenta con un boton que cuando es presionado redirige al usuario a la pantalla correspondiente. Una vez el usuario se ha registrado, se vuelve a llevar al ususario a la pantalla de inicio de sesión (ver Figura 52). Cuando el usuario inicia la sesión se le lleva a la pantalla del temporizador.

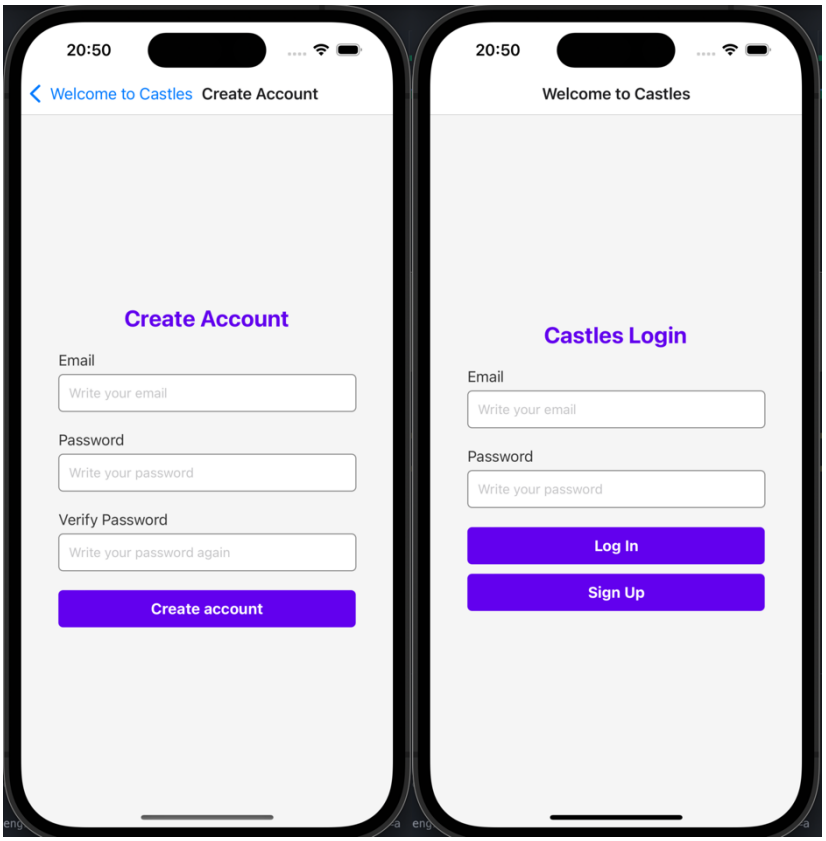


Figura 52. Pantallas crear cuenta e iniciar sesión

En el temporizador el usuario puede seleccionar una actividad donde registrará tiempo y la duración del temporizador. Cuando el usuario presione sobre 'start' el temporizador dara comienzo y se mostrará una cuenta atrás con la duración que ha seleccionado el usuario, una animación mostrando que el tiempo esta transcurriendo y un botón de 'stop'. Si en algún momento el usuario el temporizador se ha detenido en consecuencia. Al finalizar el temporizador el usuario verá una ventana informándole de los puntos que ha obtenido (ver Figura 53).

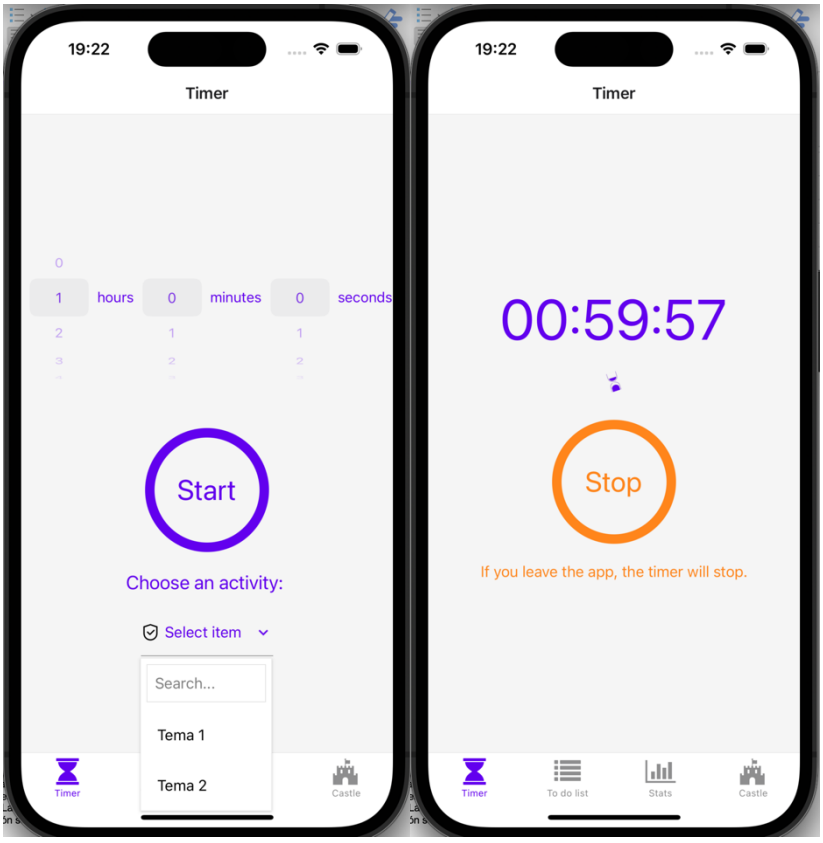


Figura 53. Pantallas del temporizador

En la pantalla de la lista de tareas, el usuario puede visualizar todas las actividades pendientes y completadas, así como el tiempo dedicado a cada una de ellas. Las actividades completadas aparecen tachadas, mientras que las pendientes se muestran de manera normal. El usuario puede añadir nuevas actividades presionando un botón específico y marcarlas como completadas o eliminarlas según sea necesario. El usuario también cuenta con la opción de esconder las tareas ya completadas (ver Figuro 54).

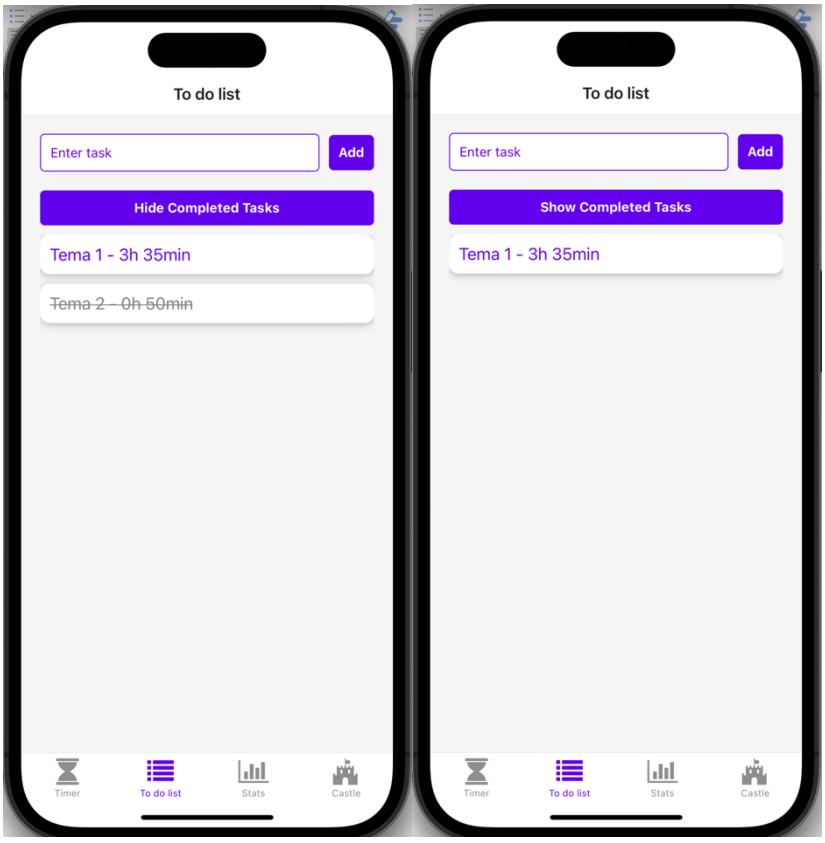


Figura 54. Pantalla to do list

La pantalla de estadísticas (ver Figura 55) permite a los usuarios ver un resumen detallado de su progreso y tiempo dedicado a las actividades. En esta pantalla, se muestran gráficos que reflejan las horas totales registradas, el tiempo dedicado en la última semana y un ranking basado en las horas invertidas por usuarios en la última semana.

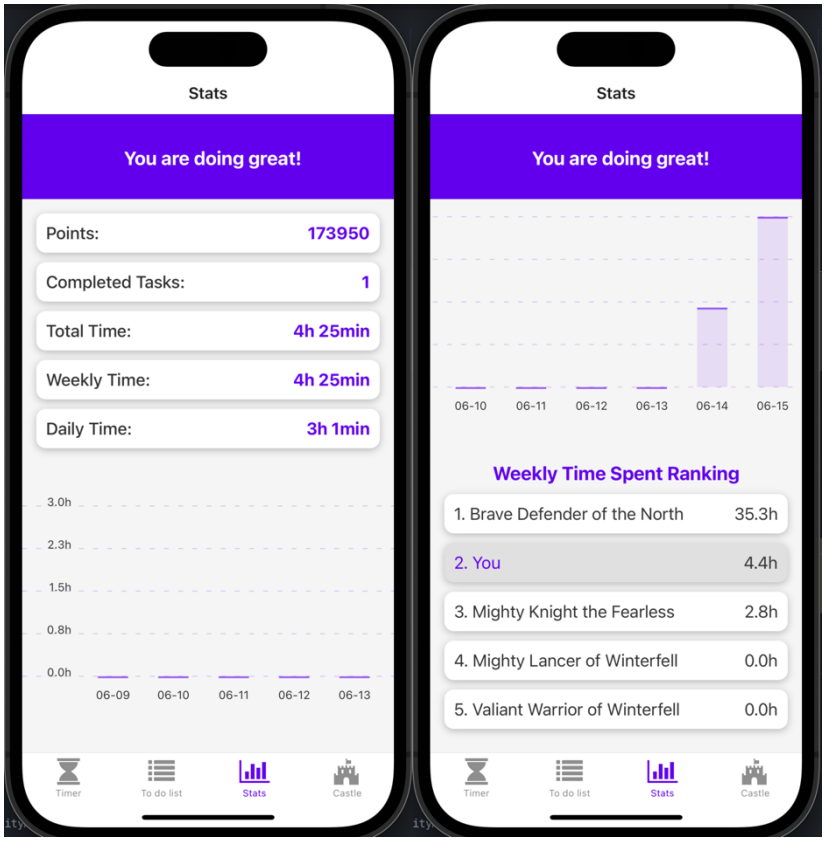


Figura 55. Pantalla estadísticas

En la pantalla de gamificación (ver Figura 56), los usuarios pueden interactuar con su castillo virtual. Esta pantalla muestra una representación gráfica del castillo y permite a los usuarios mejorar diferentes partes del mismo utilizando los puntos obtenidos por completar actividades.

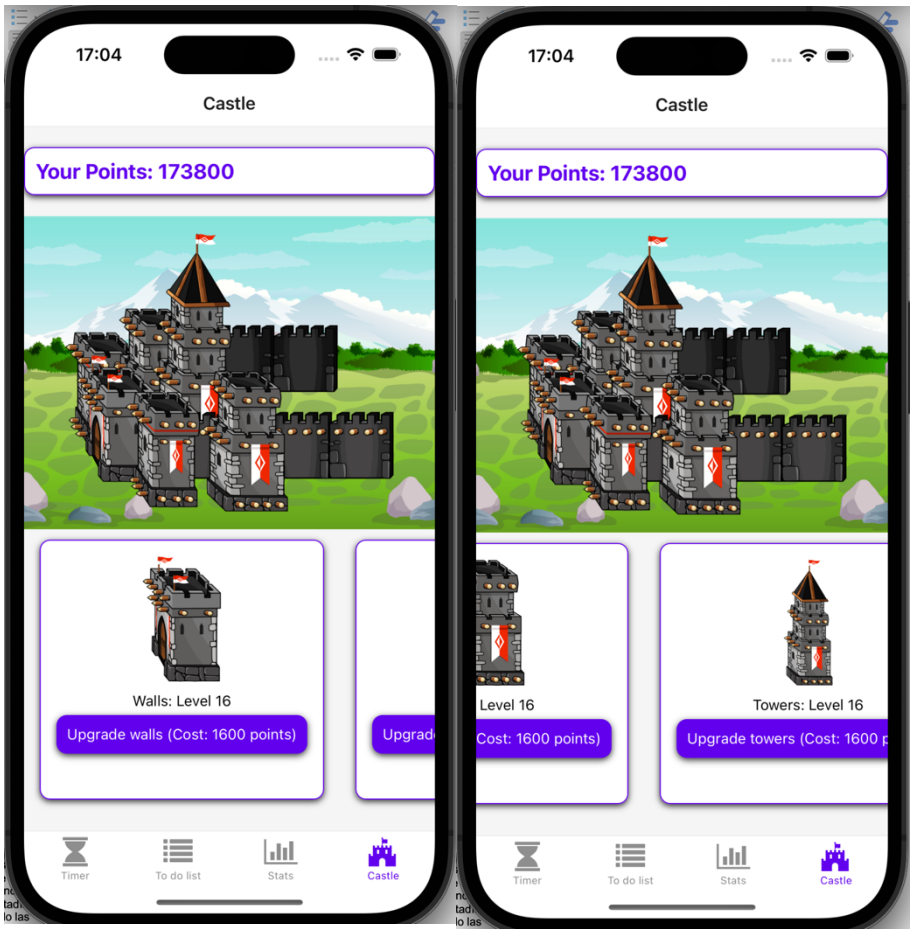


Figura 56. Pantalla castillo

8. Pruebas

Realizar pruebas de software es fundamental en el desarrollo de una aplicación, ya que asegura la calidad, fiabilidad y seguridad del producto final. Incluso los errores menores pueden tener consecuencias significativas. Por ejemplo, en 2018, un error en el software de navegación de los trenes en Japón causó múltiples retrasos y pérdidas económicas significativas[19]. Otra instancia es el fallo del software en el sistema de administración de vuelos de British Airways en 2017, que resultó en la cancelación de cientos de vuelos y pérdidas millonarias [20].

8.1 Herramientas utilizadas para las pruebas

Para llevar a cabo las pruebas de la aplicación, que fue desarrollada en React Native con TypeScript, se seleccionaron las siguientes herramientas:

- Jest: Este framework nos permitió crear pruebas unitarias y de integración, gracias a su compatibilidad con TypeScript.
- React Native Testing Library: Esta librería nos ayudó a simular interacciones de usuario en la aplicación y verificar que los componentes reaccionan correctamente a estas interacciones.

8.2 Pruebas unitarias

Las pruebas unitarias están diseñadas para verificar el funcionamiento de componentes específicos de la aplicación de forma aislada. Esto asegura que cada parte del código funcione correctamente por sí misma.

- Las pruebas unitarias realizadas se centraron en:
- Validar el funcionamiento de las clases entidad: Se crearon pruebas unitarias para cada clase entidad, verificando sus constructores, atributos y métodos getters y setters.
- Validar el funcionamiento de componentes aislados: Utilizando React Native Testing Library, se crearon pruebas para verificar la correcta creación y comportamiento de componentes visuales aislados.

8.3 Pruebas de integración

Las pruebas de integración verifican cómo interactúan entre sí varios componentes de código, asegurando que funcionen correctamente cuando se combinan.

Para estas pruebas, se crearon test que validan el funcionamiento de los elementos de la lógica de la aplicación. Cada clase de la capa lógica fue sometida a pruebas para comprobar su correcto funcionamiento, ya que muchas de estas clases colaboran con repositorios para realizar operaciones en el Backend o proporcionar información a la interfaz de usuario.

9. Experimento validación

Para validar el MVP de la aplicación, se presentó a un grupo de usuarios objetivos, en su mayoría universitarios. Se les permitió experimentar con la aplicación sin intervención alguna. Tras este periodo de interacción, se les pidió que completaran una encuesta. A continuación, se comentan los resultados obtenidos de la encuesta.

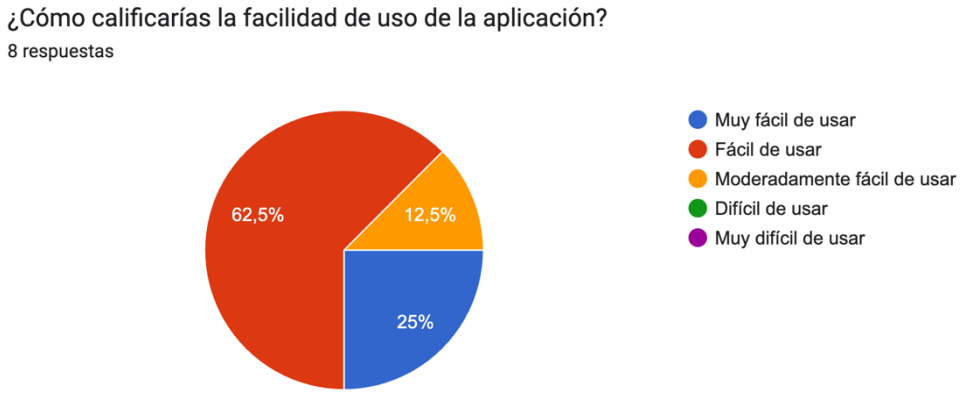


Figura 57. Encuesta validación Pregunta 1

Figura 58. Encuesta validación Pregunta 2

¿Qué tan útil encuentras la funcionalidad del temporizador para gestionar tu tiempo?

8 respuestas

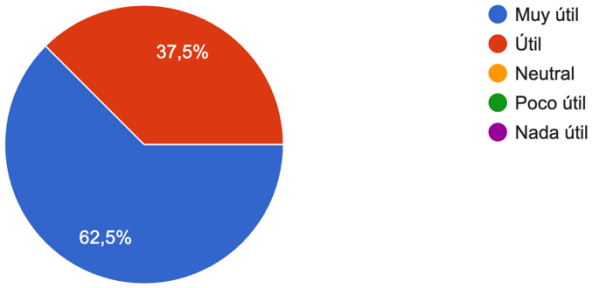


Figura 59. Encuesta validación Pregunta 3

¿Consideras útil la función de estadísticas para seguir tu progreso?

8 respuestas

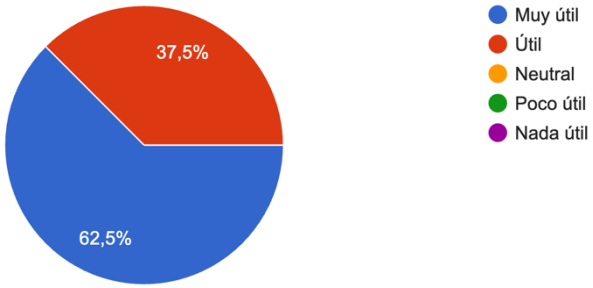


Figura 60. Encuesta validación Pregunta 4

¿Qué tan fácil te resulta crear y gestionar actividades en la To-Do List?

8 respuestas

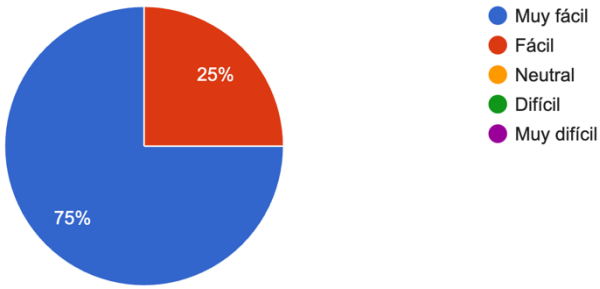


Figura 61. Encuesta validación Pregunta 5

Figura 62. Encuesta validación Pregunta 6

Figura 63. Encuesta validación Pregunta 7

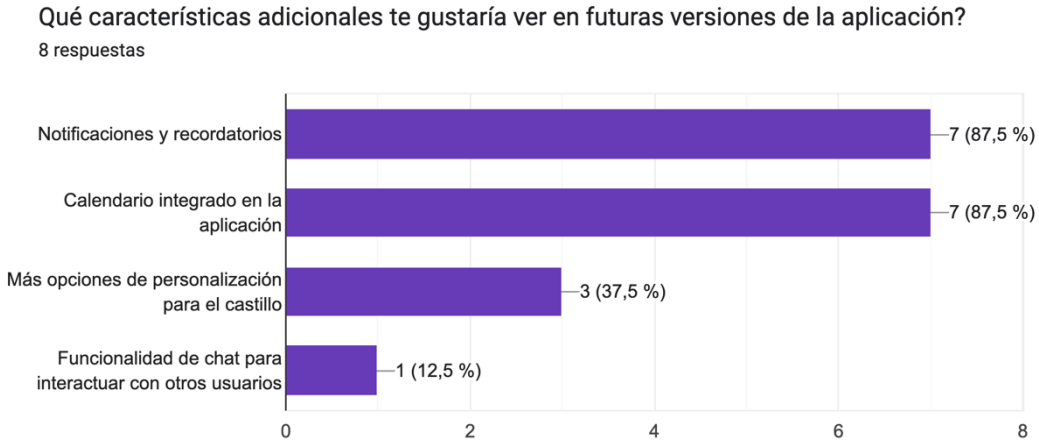


Figura 64. Encuesta validación Pregunta 8

Los resultados de la encuesta reflejan una recepción mayormente positiva hacia la aplicación. En cuanto a la facilidad de uso, la mayoría de los participantes consideraron la aplicación fácil o muy fácil de usar, con una sola mención de "moderadamente fácil de usar". Esto indica que la interfaz y la navegación de la aplicación son generalmente intuitivas y accesibles para los

usuarios.

En relación a la satisfacción con las funcionalidades actuales, todos los participantes se mostraron satisfechos, y un usuario incluso indicó estar "muy satisfecho". Esto sugiere que las funcionalidades presentes en la versión MVP cumplen con las expectativas y necesidades básicas de los usuarios.

Respecto a la utilidad del temporizador para gestionar el tiempo, la mayoría de los usuarios lo encontraron muy útil, con solo un par de menciones de "útil". Este feedback positivo destaca la importancia de esta funcionalidad para el público objetivo y su eficacia en la gestión del tiempo.

La función de estadísticas también fue bien recibida, siendo considerada muy útil por la mayoría de los encuestados. Solo unas pocas respuestas la calificaron como "útil", lo que reafirma el valor que los usuarios encuentran en el seguimiento de su progreso.

La creación y gestión de actividades en la To-Do List fue calificada mayoritariamente como muy fácil, con solo una mención de "fácil". Esto indica que los usuarios encuentran esta funcionalidad sencilla y eficiente de usar.

En cuanto a la motivación que proporciona la gamificación (recompensas, creación y mejora de castillos), las respuestas fueron variadas. Mientras que algunos usuarios se sintieron bastante motivados, otros mostraron neutralidad o incluso falta de motivación. Este es un área donde se podría explorar más para aumentar el compromiso de los usuarios.

La interfaz gráfica fue generalmente considerada muy útil e intuitiva, con una sola excepción que la calificó como "útil". Esto indica que el diseño visual de la aplicación es atractivo y funcional para los usuarios.

Finalmente, se pidió a los usuarios que sugirieran características adicionales para futuras versiones de la aplicación. Las sugerencias más comunes incluyeron la integración de un calendario, notificaciones y recordatorios, más opciones de personalización para el castillo, y la funcionalidad de chat para interactuar con otros usuarios. Estas sugerencias proporcionan una valiosa guía para futuras mejoras y nuevas funcionalidades que podrían aumentar la satisfacción y el compromiso de los usuarios.

En general, la mayoría de los usuarios indicaron que probablemente recomendarían la aplicación a sus amigos, lo cual es un fuerte indicador de aceptación y potencial de crecimiento de la base de usuarios. Este experimento de validación muestra que, aunque hay áreas para mejorar, la aplicación en su estado actual es bien recibida y cumple con las necesidades básicas de los usuarios.

10. Conclusiones

El objetivo principal de este Trabajo de Fin de Grado era desarrollar una aplicación móvil de productividad que permitiera a los usuarios seguir y controlar el tiempo que dedican a diversas actividades de su elección, incorporando elementos de gamificación para incentivar el uso continuo de la aplicación. Este objetivo se logró con éxito, como lo demuestran los resultados positivos obtenidos en el experimento de validación con los usuarios.

Durante el desarrollo de este proyecto, se aplicaron conocimientos adquiridos en varias asignaturas del plan de estudios. En la asignatura de Proyecto de Ingeniería del Software, se aprendió la metodología ágil, que fue fundamental para la organización y gestión del desarrollo del proyecto. Esta metodología permitió una planificación flexible y una adaptación continua a los cambios y necesidades del proyecto. En la asignatura de Mantenimiento y Evolución del Software, se adquirieron habilidades para la documentación y resolución de errores. Un ejemplo de esto fue el bug encontrado durante el desarrollo, relacionado con la visualización de actividades en la lista de tareas. La metodología aprendida en esta asignatura facilitó la identificación, documentación y solución eficiente del problema. La asignatura de Análisis y Especificación de Requisitos fue crucial para las secciones de análisis y especificación de requisitos del proyecto. Gracias a esta formación, se pudieron definir claramente las necesidades del usuario y los requisitos funcionales de la aplicación, lo que aseguró que el producto final cumpliera con las expectativas de los usuarios.

Durante el desarrollo, se encontraron varias dificultades, como la gestión eficiente de los errores y la adaptación a cambios inesperados, pero estas se abordaron con éxito gracias a la formación recibida y a la aplicación de metodologías ágiles.

A futuro, las sugerencias obtenidas en la encuesta de validación proporcionan una hoja de ruta clara para futuras mejoras y nuevas funcionalidades. La incorporación de características como notificaciones y recordatorios, integración de un calendario, más opciones de personalización para el castillo y funcionalidades de chat para la interacción entre usuarios, son prioridades para próximas versiones de la aplicación. Estas mejoras no solo aumentarían la satisfacción del usuario, sino que también incrementarían el uso y la recomendación de la aplicación, alineándose con el objetivo inicial de crear una herramienta efectiva y atractiva para la gestión del tiempo.

En resumen, el proyecto no solo cumplió con los objetivos establecidos, sino que también proporcionó valiosas lecciones y oportunidades de aprendizaje que serán útiles en futuros desarrollos y proyectos profesionales. La validación del MVP y las sugerencias de los usuarios servirán de guía en las próximas fases de desarrollo de la aplicación. Cada uno de los objetivos específicos se logró, demostrando que la aplicación es una herramienta

eficaz para mejorar la gestión del tiempo y fomentar hábitos de estudio sostenibles entre los usuarios.

11. Referencias

[1] Steel, P. y Ferrari, J. (2013). Sex, education and procrastination: An epidemiological study of procrastinators' characteristics from a global sample. European Journal of Personality, 27(1), 51-58. doi:10.1002/per.1851
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjT1MTTupqGAxWAVfEDHRD3ANYQFnoECBEQAw&url=https%3A%2F%2Frevistas.um.es%2Frie%2Farticle%2Fdownload%2F344781%2F277071%2F1370731%23%3A~%3Atext%3DPor%2520encima%2520de%2520esas%2520cifras%2Clos%2520cuales%2520reconocen%2520ser%2520procrastinadores.&usg=AOvVaw1dDKvCJz_jzpxkz4JwAa8d&opi=89978449

[2] J. Hamari, J. Koivisto and H. Sarsa, "Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification," 2014 47th Hawaii International Conference on System Sciences, Waikoloa, HI, USA, 2014, pp. 3025-3034, doi: 10.1109/HICSS.2014.377
https://ieeexplore.ieee.org/document/6758978/citations#citations

[3] Refuerzo positivo
https://psicologiaymente.com/psicologia/refuerzo-positivo-negativo

[4] la productividad está de moda
https://trends.google.es/trends/explore?date=today%205-y&geo=ES&q=%2Fm%2F026y05&hl=es

[5] Modelo de negocio Freemium
https://es.wikipedia.org/wiki/Freemium

[6] el 82% de todas las aplicaciones gratuitas en la Google Play Store incluían anuncios
https://www.businessofapps.com/data/google-play-statistics/

[7] Las tarifas de CPM
https://www.businessofapps.com/insights/how-inflation-affects-ad-revenues-all-the-mobile-ecpm-insights-to-boost-your-ad-strategy-in-2023/

[8]Microtransacciones: https://es.wikipedia.org/wiki/Microtransacci%C3%B3n

[9] Eric Ries. El método Lean Startup. Crown Business Publishing, S.A., 2011.

[10] Scrum
https://www.scrum.org/resources/what-scrum-module

[11] Metodología ágil
https://www.atlassian.com/agile

[12] Typescript
https://www.typescriptlang.org/

[13] Programación orientada a objetos
https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

[14] node.js
https://es.wikipedia.org/wiki/Node.js

[15] React Native
https://es.wikipedia.org/wiki/React_Native

[16] Expo
https://docs.expo.dev/get-started/introduction/

[17] Modelo vista controlador
https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[18] NoSQL es un enfoque utilizado en el diseño de bases de datos que permite el almacenamiento y consulta de datos fuera de las estructuras tradicionales que se encuentran en las bases de datos relacionales.

https://www.ibm.com/es-es/topics/nosql-databases

[19] un error en el software de navegación de los trenes en Japón causó múltiples retrasos y pérdidas económicas significativas
https://www.theguardian.com/world/2018/oct/22/japan-rail-software-error-causes-massive-delays

[20] fallo del software en el sistema de administración de vuelos de British Airways en 2017 https://www.bbc.com/news/business-40080224

Anexo A: Objetivos de Desarrollo Sostenible

Los Objetivos de Desarrollo Sostenible (ODS) son un conjunto de 17 metas globales adoptadas por las Naciones Unidas en 2015 como parte de la Agenda 2030 para el Desarrollo Sostenible. Estos objetivos buscan abordar los desafíos globales, incluidos la pobreza, la desigualdad, el cambio climático, la degradación ambiental, la paz y la justicia. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos años, con el propósito de mejorar la vida de las personas en todo el mundo y proteger el planeta.

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**x**
ODS 2. **Hambre cero.**				**x**
ODS 3. **Salud y bienestar.**	**x**			
ODS 4. **Educación de calidad.**	**x**			
ODS 5. **Igualdad de género.**				**x**
ODS 6. **Agua limpia y saneamiento.**				**x**
ODS 7. **Energía asequible y no contaminante.**		**x**		
ODS 8. **Trabajo decente y crecimiento económico.**		**x**		
ODS 9. **Industria, innovación e infraestructuras.**		**x**		
ODS 10. **Reducción de las desigualdades**.				**x**
ODS 11. **Ciudades y comunidades sostenibles**.				**x**
ODS 12. **Producción y consumo responsables.**				**x**
ODS 13. **Acción por el clima.**				**x**
ODS 14. **Vida submarina.**				**x**
ODS 15. **Vida de ecosistemas terrestres.**				**x**
ODS 16. **Paz, justicia e instituciones sólidas.**				**x**
ODS 17. **Alianzas para lograr objetivos.**		**x**		

Tabla 1-A

Como se puede observar en la Tabla 1-A, la aplicación desarrollada con el objetivo de formar hábitos esta relacionada con varios ODS:

Alta Relación:

- **ODS 3. Salud y bienestar**: La aplicación promueve una mejor gestión del tiempo, lo cual puede reducir el estrés y mejorar la salud mental de los usuarios. Al organizar mejor su tiempo, los usuarios pueden equilibrar sus actividades diarias, lo que contribuye a su bienestar general.

- **ODS 4. Educación de calidad:** La aplicación está diseñada para ayudar a los jóvenes a mejorar sus hábitos de estudio y gestión del tiempo. Al proporcionar herramientas para registrar y analizar el tiempo dedicado a las actividades académicas, la aplicación fomenta una educación más estructurada y eficiente, contribuyendo a una educación de calidad.

Relación Media:

- **ODS 8. Trabajo decente y crecimiento económico**: Aunque la relación es media, la mejora en la gestión del tiempo y la productividad personal puede llevar a un mejor desempeño laboral. Esto puede contribuir al crecimiento económico personal y, en un sentido más amplio, fomentar prácticas de trabajo más eficientes y productivas.

- **ODS 9. Industria, innovación e infraestructuras**: La aplicación fomenta la innovación al utilizar tecnología para mejorar la productividad y la gestión del tiempo. Al integrar métodos innovadores como la gamificación para incentivar el uso de la aplicación, se promueve el desarrollo de soluciones tecnológicas avanzadas.

- **ODS 17. Alianzas para lograr objetivos:** La aplicación tiene el potencial de formar parte de iniciativas colaborativas para mejorar la educación y la productividad. Al integrarse en programas más amplios y alianzas, puede contribuir a alcanzar los objetivos comunes de desarrollo sostenible.

Anexo B: Código

B.1 Código de la clase de lógica StatsController

A continuación, se muestra el código de la clase `StatsController` (ver Fragmento 1-B), perteneciente a la capa de lógica de negocio. Esta clase se encarga de gestionar y procesar las estadísticas del usuario, realizando las llamadas correspondientes a la base de datos para recuperar y actualizar los datos necesarios.

```
1. import { Request, Response } from 'express';
2. import { UserStats } from '../models/userStats';
3. import { UserDataService } from '../services/userDataService';
4.
5. export class StatsController {
6.   static async getUserStats(req: Request, res: Response) {
7.     try {
8.       const userId = req.params.userId;
9.       const stats = await UserDataService.getUserStats(userId);
10.      res.status(200).json(stats);
11.    } catch (error) {
12.      res.status(500).json({ error: 'Error retrieving user stats' });
13.    }
14.  }
15.
16.  static async updateUserStats(req: Request, res: Response) {
17.    try {
18.      const userId = req.params.userId;
19.      const statsData: UserStats = req.body;
20.      const updatedStats = await UserDataService.updateUserStats(userId, statsData);
21.      res.status(200).json(updatedStats);
22.    } catch (error) {
23.      res.status(500).json({ error: 'Error updating user stats' });
24.    }
25.  }
26. }
27.
```

Fragmento de código 1-B: Código de la clase `StatsController`.

B.2 Código de la clase entidad UserData

El siguiente fragmento de código (ver Fragmento 2-B) corresponde a la clase de la entidad `UserData` creada dentro de nuestro proyecto. Esta clase define los atributos del usuario y proporciona métodos para transformar el JSON obtenido de la base de datos en un objeto de la clase.

```
1. export class UserData {
2.   constructor(
3.     public userId: string,
4.     public totalHours: number,
5.     public last7DaysHours: number[],
6.     public activities: string[]
7.   ) {}
```

```
 8.
 9.   static fromJSON(json: any): UserData {
10.     return new UserData(
11.       json.userId,
12.       json.totalHours,
13.       json.last7DaysHours || [],
14.       json.activities || []
15.     );
16.   }
17. }
18.
```

Fragmento de código 2-B: Código de la clase `UserData`.

B.3 Código de la clase de prueba StatsControllerTest

En el siguiente código (ver Fragmento 3-B) se puede observar el funcionamiento de una de las pruebas creadas para la aplicación. En este caso, es una prueba que valida el funcionamiento de la lógica de las estadísticas del usuario, comprobando que se recuperan y actualizan correctamente.

```
1. import { StatsController } from "../../controllers/statsController";
 2. import { UserDataService } from "../../services/userDataService";
 3. import { Request, Response } from "express";
 4.
 5. jest.mock("../../services/userDataService");
 6.
 7. const mockRequest = (params: any, body: any) => ({
 8.   params,
 9.   body,
10. }) as Request;
11.
12. const mockResponse = () => {
13.   const res = {} as Response;
14.   res.status = jest.fn().mockReturnValue(res);
15.   res.json = jest.fn().mockReturnValue(res);
16.   return res;
17. };
18.
19. describe("StatsController", () => {
20.   beforeEach(() => {
21.     jest.clearAllMocks();
22.   });
23.
24.   describe("getUserStats", () => {
25.     it("should return user stats successfully", async () => {
26.       const req = mockRequest({ userId: "user-1" }, {});
27.       const res = mockResponse();
28.       const mockStats = { totalHours: 50, last7DaysHours: [5, 7, 8, 6, 6, 9, 9] };
29.
30.       (UserDataService.getUserStats as jest.Mock).mockResolvedValue(mockStats);
31.
32.       await StatsController.getUserStats(req, res);
33.
34.       expect(res.status).toHaveBeenCalledWith(200);
```

```
35.       expect(res.json).toHaveBeenCalledWith(mockStats);
36.     });
37.
38.     it("should handle errors", async () => {
39.       const req = mockRequest({ userId: "user-1" }, {});
40.       const res = mockResponse();
41.
42.       (UserDataService.getUserStats as jest.Mock).mockRejectedValue(new Error("Error
retrieving user stats"));
43.
44.       await StatsController.getUserStats(req, res);
45.
46.       expect(res.status).toHaveBeenCalledWith(500);
47.       expect(res.json).toHaveBeenCalledWith({ error: "Error retrieving user stats" });
48.     });
49.   });
50.
51.   describe("updateUserStats", () => {
52.     it("should update user stats successfully", async () => {
53.       const req = mockRequest({ userId: "user-1" }, { totalHours: 55, last7DaysHours:
[6, 8, 9, 7, 7, 10, 8] });
54.       const res = mockResponse();
55.       const mockUpdatedStats = { totalHours: 55, last7DaysHours: [6, 8, 9, 7, 7, 10,
8] };
56.
57.       (UserDataService.updateUserStats as
jest.Mock).mockResolvedValue(mockUpdatedStats);
58.
59.       await StatsController.updateUserStats(req, res);
60.
61.       expect(res.status).toHaveBeenCalledWith(200);
62.       expect(res.json).toHaveBeenCalledWith(mockUpdatedStats);
63.     });
64.
65.     it("should handle errors", async () => {
66.       const req = mockRequest({ userId: "user-1" }, { totalHours: 55, last7DaysHours:
[6, 8, 9, 7, 7, 10, 8] });
67.       const res = mockResponse();
68.
69.       (UserDataService.updateUserStats as jest.Mock).mockRejectedValue(new
Error("Error updating user stats"));
70.
71.       await StatsController.updateUserStats(req, res);
72.
73.       expect(res.status).toHaveBeenCalledWith(500);
74.       expect(res.json).toHaveBeenCalledWith({ error: "Error updating user stats" });
75.     });
76.   });
77. });
78. ```
79.
```

Fragmento de código 3-B: Código de la clase `StatsControllerTest`.