Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN DE INGENIERÍA DE SOFTWARE

WEB APP PARA LA RESOLUCCIÓN DEL PASATIEMPO DE LA MESA DEL RELOJERO

Alumno: Ángel César Pérez Blanco

Tutor: Alfonso Jesús Población Sáez

A mi familia, que siempre me ha apoyado a lo largo del camino.

Agradecimientos

A mi familia, que me ha apoyado en los momentos mas complicados, y por ende, he podido lidiar con los problemas para poder seguir adelante durante estos años academicos.

A mis amigos, que ellos han sido mi principal apoyo y me han ayudado a lidiar tanto en lo academico como en lo personal, y han sido un gran apoyo en muchos momentos complicados durante estos años.

Gracias a todos.

Resumen

El objetivo de este proyecto es crear una página web para la resolución del pasatiempo de la mesa del relojero.

En esta página web se pretende que los usuarios puedan resolver la mesa del relojero de su elección y guardar la marca de tiempo en la que completan el desafío, y asi fomentar la competitividad entre los usuarios, invitándolos a superarse y compararse con otros.

El desarrollo de esta página web se ha llevado a cabo con el framework React JS, HTML5, CSS y Javascript, y para la gestión de base de datos se ha optado por utilizar Firebase.

Abstract

The objective of this project is to create a website for solving the quiz "Mesa del relojero".

In this website, the goal is to allow users to solve "Mesa del relojero.ºf their choice and save the timestamp when they complete the challenge, thus fostering competitiveness among users, encouraging them to push their limits and compare themselves with others.

The development of this website has been carried out using the React JS framework, HTML5, CSS, and Javascript. Firebase has been chosen for managing the database.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Introducción

En el campo de los rompecabezas y los acertijos, la resolución de la mesa del relojero ha capturado la atención de los aficionados y ha permitido que estos puedan poner a prueba su capacidad de razonamiento lógico y su agudeza espacial. Este pasatiempo, llamado la mesa del relojero, muestra un casillero con letras, donde originalmente mostraba el fragmento de una obra literaria de un autor que se ha dividido en una serie piezas, que se han colocado debajo del casillero. El objetivo principal es colocar todas las piezas en el casillero para que al final se pueda leer el fragmento literario.

1.2. Motivación

Aunque existan revistas o publicaciones impresas donde puedas encontrar este tipo de rompecabezas, no existe ninguna página web donde puedas entretenerte online con este tipo de pasatiempo en concreto. Este hecho, me ha motivado a desarrollar esta página web, además de aplicar los conocimientos que he ido adquiriendo durante estos años en el grado.

Desde mi infancia, este pasatiempo, junto con otros rompecabezas y acertijos que suelen acompañar las revistas en las que se publican, ha sido uno de mis hobbies favoritos, lo cual me ha motivado a embarcarme en el desarrollo de este proyecto.

También me ha llegado a motivar el hecho de aprender más de cerca el desarrollo web tanto el funcionamiento del Back-End, como del Front-End, ya que ambas me llaman la atención, y desearía orientar mi futuro laboral en esa dirección.

1.3. Objetivos de proyecto

1.3.1. Objetivos de desarrollo

El objetivo principal de este proyecto es desarrollar un sistema interactivo que permita a los usuarios resolver el pasatiempo de la mesa del relojero en el menor tiempo posible, y luego poder comparar sus marcas de tiempo con el resto de usuarios. A través de esta experiencia, buscamos combinar el desafío intelectual de la resolución del rompecabezas con la competitividad que motive a los usuarios a superarse entre sí.

Además, este proyecto tiene como objetivo fomentar el desarrollo de habilidades cognitivas, como el razonamiento lógico, la percepción espacial y la resolución de problemas, al tiempo que se brinda una experiencia entretenida para los usuarios.

La importancia de este tipo de pasatiempo radica en su potencial para promover el desarrollo cognitivo y creativo de los usuarios, así como en su capacidad para combinar los elementos de los rompecabezas y la literatura.

1.3.2. Objetivos personales

Ya que este proyecto se enmarca en un contexto educativo y ha sido desarrollado como Trabajo de Fin de Grado, también se han marcado unos objetivos de formación personal que se enumerarán a continuación:

- Conocer en profundidad el framework React.
- Ampliar mis conocimientos en el diseño gráfico de páginas web utilizando CSS.
- Obtener mayor experiencia con el uso de base de datos, utilizando para ello Firebase.
- Mejorar mi resolución y destreza a la hora de desarrollar y diseñar la estructura de una página web.
- Tener un mejor entendimiento sobre la metodología Scrum, una herramienta muy utilizada a día de hoy por las empresas debido a su fácil entendimiento y su versatilidad a la hora de aplicar diferentes procesos y técnicas.

1.4. Nicho de mercado y usuarios objetivo

Tras haber realizado una búsqueda exhaustiva por internet respecto a este tipo de pasatiempo, puede señalar que no existe ninguna página web que permita resolver de manera interactiva el pasatiempo de la mesa del relojero, por lo que puede ser novedoso y llamar la atención a los usuarios que les guste resolver pasatiempos de forma online.

Cabe destacar que cualquier usuario con acceso a internet podrá acceder y utilizar la página web. El público objetivo de esta web abarca un amplio espectro de usuarios, aunque sobre todo está dirigido a los amantes de los rompecabezas y los acertijos.

Se proporciona a la aplicación una interfaz intuitiva para que todos los usuarios, incluidos aquellos con poca experiencia tecnológica o de edad avanzada, puedan hacer uso de su funcionalidad con facilidad.

1.5. Estructura de la memoria

La memoria está estructurada de la siguiente manera:

- En el **Capítulo 2** veremos la planificación inicial del proyecto y su desarrollo utilizando la metodología Scrum.
- En el **Capítulo 3** veremos en detalle las tecnologías y herramientas usadas para el desarrollo del proyecto.
- En el **Capítulo 4** realizaremos un análisis del proyecto y los requisitos que conlleva.
- En el **Capítulo 5** veremos y explicaremos como es el diseño de la aplicación.
- En el **Capítulo 6** realizaremos una estimación de costes y un plan de riesgos que pueda suceder durante el desarrollo de la página web.
- En el **Capítulo 7** veremos como hemos estructurado el código y los cambios que han ido surgiendo en su desarrollo.
- En el **Capítulo 8** veremos tras realizar el proyecto, las conclusiones y las futuras mejoras de la página.
- Finalmente veremos el **"Manual de usuario"** y el **"Manual de despliegue"**.

Capítulo 2

Desarrollo y Planificación del Proyecto

2.1. Scrum

2.1.1. Introducción

Scrum [4] es un marco de trabajo ágil utilizado para la gestión de proyectos, especialmente en el desarrollo software, y que promueve y simplifica el trabajo colaborativo entre equipos. Su objetivo principal es reducir la complejidad a la hora de embarcarse en un nuevo proyecto. Además, proporciona un enfoque iterativo e incremental, lo que permite a los equipos de trabajo adaptarse a los cambios que surjan y entregar objetivos del proyecto de manera continua.

Se le pueden asociar 3 pilares fundamentales:

- **Transparencia:** Los aspectos importantes deben ser claros y visibles para todos los involucrados que quieran alcanzar un resultado optimo con el proyecto. Este enfoque transparente ayuda a alinear a todos los involucrados, lo que significa que todos tengan una comprensión clara del desarrollo, objetivos y su progreso, y a su vez, fomentar la colaboración y evitar malentendidos o confusiones que podrían retrasar el proyecto.
- **Inspección:** Es importante que todas las personas involucradas en el proyecto realicen inspecciones periódicas del progreso, de manera que se pueda evitar que los problemas se acumulen y puedan convertirse en obstáculos más difíciles de afrontar.
- **Adaptación:** Cuando se detecten variaciones en el producto durante su desarrollo, es importante realizar ajustes adecuados para abordar las variaciones y mantener el proyecto en el camino correcto.

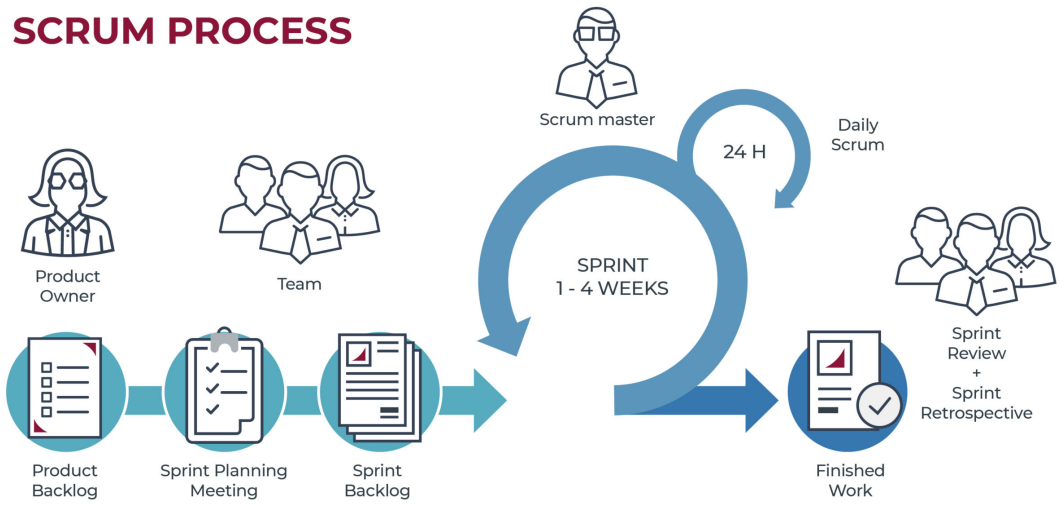

Figura 2.1: Proceso de la metodología SCRUM

2.1.2. Equipo Scrum

El equipo Scrum está compuesto por un grupo de personas que desempeñan diferentes roles, los cuales tienen una responsabilidad especifica. Cada miembro del equipo tiene la responsabilidad informarse entre sí, como ante el resto de la organización. Los roles clave en Scrum son los siguientes:

- **Product Owner:** es el responsable de representar los intereses de los stakeholders y definir las prioridades del producto. Su rol principal es tomar decisiones sobre las características que debe tener el producto y obtener el máximo valor al mínimo coste.
- **Scrum Master:** es el líder del equipo Scrum y facilita su progreso. Su principal funcion es asegurarse que se sigan los principios y prácticas de Scrum, así como que se alcancen los objetivos, eliminando los obstáculos que se van afrontando durante su desarrollo y fomentando la colaboración entre los miembros del equipo.
- **Equipo de desarrollo:** son los profesionales que realizan el trabajo de desarrollo real del producto, autoorganizándose y autogestionandose para entregar incrementos de producto funcionales.

2.1.3. Eventos Scrum

Scrum dispone de cinco eventos [23] para mantener los mínimos necesarios para facilitar el control empírico de procesos funcionales y por tanto no dinamitar ninguno de los pilares

fundamentales de Scrum.

- **Sprint:** es el periodo de tiempo que tarda un equipo Scrum en finalizar un incremento, y es el evento fundamental del marco Scrum [5]. La duración ideal de un sprint es de aproximadamente 2 semanas, pero esto puede variar debido al equipo o al proyecto (lo que vea adecuado el *Scrum Master*). Todos los eventos, desde la planificación hasta la retrospectiva, tienen lugar durante el sprint.
- **Planificación del Sprint (*Sprint Planning*):** es una reunión que da inicio a cada sprint. En esta reunión, el equipo se reúne para decidir qué elementos del backlog del producto se van a elegir para trabajar en el próximo sprint, y que objetivos alcanzar. Además, se crea un plan detallado para que se ejecute a lo largo del sprint.
- **Reunion diaria (*Daily Scrum*):** es una reunión corta de unos 15 minutos que se realiza diariamente durante el sprint. Durante la reunión, cada miembro del equipo revisa el progreso que realizó el dia anterior o que problemas se encontró que le impiden avanzar, además de planificar el trabajo para las próximas 24 horas. En resumen, durante la reunión se comenta los siguientes puntos:
  - Tareas realizadas después de la Daily del día anterior.
  - Tareas asignadas para el día de hoy.
  - Problema u obstáculo que se han ido encontrando.
- **Revisión del Sprint (*Sprint Review*):** es una reunión que se realiza al final de cada sprint, en la cual se muestra el trabajo realizado por el equipo Scrum a los Stakeholders. Además, se revisa si todas las tareas asignadas al Backlog están completadas y se discute las funcionalidades desarrolladas y se evalúa si cumplen con los criterios de aceptación.
- **Retrospectiva del Sprint (*Sprint Retrospective*):** es una reunión después de la revisión del sprint, en la cual el equipo Scrum reflexiona sobre el sprint y analiza que ha funcionado y que no. La idea de esta reunión es hacer que el equipo analice los posibles errores del sprint anterior y que se puede mejorar para el próximo sprint.

  Es importante indicar que estos eventos son flexibles y pueden adaptarse según las necesidades del equipo y según lo indique el *Scrum Master*.

2.1.4. Artefactos Scrum

Los artefactos [28] son elementos físicos que ayudan a proporcionar transparencia y visibilidad en el proceso de desarrollo del producto. Entre ellos se incluyen:

- **Backlog del producto (*Product Backlog*):** es una lista ordenada de todas las requisitos, funcionalidades, mejoras y correcciones que se desean para el producto. Es gestionada por el Product Owner y representa el trabajo que queda por finalizar por parte del equipo Scrum. El backlog del producto se actualiza en función de las necesidades y el feedback de los stakeholders.

- **Backlog del Sprint (*Sprint Backlog*):** es una lista de elementos seleccionados del backlog del producto que el equipo Scrum se ha comprometido a finalizar durante el ciclo actual del sprint. El equipo Scrum colabora para descomponer los elementos del backlog del producto en tareas más pequeñas y estimadas para el sprint. El backlog del sprint es dinámico y puede actualizarse a medida que surgen cambios, pero sin comprometer el objetivo inicial del sprint.
- **Incremento del producto (*Product Increment*):** es el producto final que se ha conseguido tras finalizar un sprint. Es un incremento funcional y potencialmente entregable del producto que agrega valor al mismo. Esto incluye todas las tareas, historias de usuario y casos de uso, que posteriormente se pondrá a disposición del usuario final. Cada incremento debe cumplir con los criterios de aceptación definidos y estar listo para ser liberado o demostrado al final del sprint. Esta metodología ágil se basa en realizar estos incrementos o evolutivos de manera iterativa e incremental, de ahí el nombre.

En algunos proyecto podemos encontrar más artefactos como 'Definition of Done(DoD)', 'Definition of Ready(DoR)', o Burndown Chart, pero estos son menos importantes y no son necesarios en muchos proyectos.

Estos artefactos proporcionan una base sólida para la planificación, seguimiento y transparencia en Scrum. Permiten al equipo y a los stakeholders tener una comprensión clara del trabajo que se está realizando, las prioridades y el progreso general del producto.

2.2. Planificación

Debido a que en este proyecto no puede haber todos los roles ya que al ser un Trabajo de Fin de Grado tiene que ser individual, se ha tenido que adaptar los roles a las personas involucradas en dicho proyecto.

Ante estas circunstancias, la organización del equipo Scrum estará formada por el alumno, que desempeñará el rol de 'Equipo de Desarrollo' y también de 'Product Owner', y el tutor será el encargado de desempeñar el rol de 'Scrum Master', que se encargará de brindar orientación para garantizar el cumplimiento de esta metodología.

En cuanto a la organización del proyecto, se ha establecido que cada sprint tenga una duración de 2 semanas, finalizando cada sprint los viernes. En este día también se llevarán a cabo la 'Revisión del sprint', la 'Retrospectiva' y la 'Planificación del sprint' para las siguientes 2 semanas.

Hemos utilizado la herramienta de gestión de proyectos 'Trello' para administrar tanto el 'Sprint Backlog' como el 'Product Backlog'. Esta herramienta nos ha facilitado el seguimiento de las tareas realizadas en los sprints.

El proyecto se inició en marzo de 2023 con el primer sprint. Durante este sprint inicial, se enfocó en la planificación, análisis y estructura del proyecto. Esto implicó comprender los

requisitos y las historias de usuarios que se debían implementar, así como evaluar el cómo abordar esas tareas utilizando las tecnologías seleccionadas.

Durante el desarrollo de este proyecto, nos encontramos con el desafío de equilibrar las responsabilidades laborales y el trabajo de fin de grado. Como resultado, la presentación del TFG se retrasó un año en comparación con la planificación inicial. Esto significa que existen diferencias entre los sprints originalmente planeados y los sprints que finalmente se llevaron a cabo.

Consultando con la correspondiente guía de Trabajo de Fin de Grado de Ingeniería Informática de Valladolid [27], se ha podido observar que dicho trabajo consta de 12 créditos, por lo que su equivalencia en horas será un total de 300. Dada la disponibilidad del alumno, correspondería a 35 horas por iteración y la estimación para la finalización del proyecto sería de un total de 8 sprints, dejando un sprint de margen para la corrección de errores o posibles imprevistos que puedan surgir.

En la siguiente tabla se puede observar la planificación inicial de los sprints del proyecto:

Sprint	**Inicio**	**Fin**	**Comentarios**
Sprint 0	10/02/2021	24/02/2021	Planificación inicial
Sprint 1	24/02/2021	10/03/2021	
Sprint 2	10/03/2021	24/03/2021	
Sprint 3	24/03/2021	07/04/2021	
Sprint 4	07/04/2021	21/04/2021	
Sprint 5	21/04/2021	05/05/2021	
Sprint 6	05/05/2021	19/05/2021	
Sprint 7	19/05/2021	02/06/2021	
Sprint 8	02/06/2021	16/06/2021	Sprint de refuerzo. Opcional

Cuadro 2.1: Planificación inicial de sprints

2.3. Historias de usuario

En esta tabla, vamos a enumerar todas las historias de usuarios que vamos a implementar en el proyecto.

ID	Título	Descripción
1	Ver niveles disponibles	Como usuario quiero ver todos los niveles disponibles que se pueden jugar.
2	Ver clasificacion de cada nivel	Como usuario quiero ver la clasificación que disponen los usuarios en cada nivel.
3	Jugar un nivel	Como usuario quiero poder arrastrar las piezas disponibles y completar el tablero en el menor tiempo posible.
4	Ver tiempo disponible	Como usuario quiero saber del tiempo del que se dispone para completar el tablero.
5	Guardar mi clasificación	Como usuario quiero poder guardar mi clasificación una vez haya terminado el nivel.
6	Generar una pistas	Como usuario quiero disponer de alguna ayuda para poder completar el tablero, si se requiere.

Cuadro 2.2: Historias de usuario

Para la realización de todos los sprints, vamos a utilizar los siguientes códigos que indicarán de que tipo es cada tarea que se va a realizar en cada sprint:

- **DOC**: se refiere a las tareas relacionadas con la documentación del proyecto. Estas tareas implican la creación o actualización de cualquier forma de documentación necesaria.
- **DES**: se refiere a las tareas relacionadas con el desarrollo de la aplicación. Estas tareas están enfocadas en la creación o modificación del código y la implementación de nuevas características.
- **ERR**: se refiere a las tareas relacionadas con la búsqueda y resolución de errores en la página. Estas tareas implican identificar y solucionar los problemas o defectos que puedan afectar al funcionamiento o la experiencia del usuario.
- **EVOL**: se refiere a las tareas relacionadas con la mejora o evolución de las funcionalidades existentes en la aplicación. Estas tareas buscan optimizar las características existentes, agregar nuevas funcionalidades o mejorar la experiencia del usuario.

2.3.1. Sprint 0

Este sprint inicial será enfocado en la preparación de la documentación inicial de la aplicación y la creación de un diseño lógico.

Durante este periodo, hemos realizado diversas tareas como definir todas las historias de usuario y la elaboración del documento inicial, que abarca la planificación del proyecto. También hemos dedicado tiempo a investigar la tecnología que utilizaremos para desarrollar

el proyecto, con el objetivo de establecer una base sólida y verificar si podemos cumplir nuestros objetivos iniciales.

En total, hemos invertido 33 horas en este sprint. Sin embargo, hay dos tareas que quedaron incompletas: la redacción de la introducción y la formación en React.js y Firebase. Estas tareas se trasladarán al siguiente sprint para que puedan ser completadas.

Es importante destacar que en aquellos casos en los que no podamos finalizar todas las tareas planificadas, las transferiremos al siguiente sprint como tareas incompletas. Esta práctica se repetirá en todos los sprints en los que no logremos finalizar una tarea dentro del tiempo asignado.

De esta manera, mantenemos una gestión dinámica de nuestro trabajo, adaptándonos a las circunstancias y asegurándonos de que todas las tareas sean abordadas adecuadamente, incluso si requieren más tiempo del esperado.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 001	DOC	Búsqueda de sitios web similares	1h	Completada
T - 002	DOC	Investigación funcionamiento del Drag & Drop	1h	Incompleta
T - 003	DOC	Definición inicial de historias de usuario	3h	Completada
T - 004	DOC	Diseño inicial del proyecto	5h	Completada
T - 005	DOC	Redactar la Introducción	3h	Incompleta
T - 006	DOC	Creación repositorio GitLab	1h	Completada
T - 007	DES	Investigación de Firebase en React.js	7h	Incompleta
T - 008	DES	Aprendizaje de React.js	12h	Incompleta
Total			**33h**	

Cuadro 2.3: Tareas del sprint 0

2.3.2. Sprint 1

Este sprint se inició el día 24 de Febrero. Comenzaremos por completar las tareas pendientes del sprint anterior y luego nos enfocaremos en las tareas del nuevo sprint.

En este sprint lo que haremos será definir la metodología que vamos a utilizar, que va a ser Scrum, y vamos a empezar a documentar este capítulo, explicando en que consiste el método Scrum y como tiene que ser el proyecto para adecuarse a esta metodología.

Además, comenzaremos con la parte de desarrollo, implementando las primeras secciones de la página web y trabajando en los componentes más simples para adquirir experiencia práctica con React, incluso sin realizar ninguna integración con Firebase. También empezaremos a crear bocetos de algunos de los componentes de la página.

Respecto a la parte de React, nos centraremos en el enrutamiento entre las vistas de la página web y el diseño gráfico de las mismas.

Durante este sprint, hemos invertido un total de 27 horas, principalmente dedicadas al desarrollo de los primeros componentes de la página.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 009	DOC	Redactar la Introducción	2h	Completada
T - 010	DOC	Aprendizaje React.js	7h	Incompleta
T - 011	DOC	Añadir tecnologías usadas a la documentación	1h	Completada
T - 012	DES	Realizar bocetos	3h	Incompleta
T - 013	DES	Creación de las vistas de inicio y selección de nivel	7h	Completa
T - 014	DES	Aprendizaje del enrutamiento en React	2h	Completada
T - 015	DES	Diseño de componentes (CSS)	5h	Incompleta
Total			**27h**	

Cuadro 2.4: Tareas del sprint 1

2.3.3. Sprint 2

En este sprint, se resolveran las tareas pendientes de sprints anteriores, ya que uno de los objetivos es evitar acumular tareas sin terminar y evitar posibles problemas al iniciar nuevas tareas.

Durante el sprint, nos hemos enfocado en investigar y buscar información sobre cómo implementar la funcionalidad de *Drag & Drop* en React. El objetivo es permitir que los usuarios finales puedan interactuar con la aplicación mediante esta técnica.

Además, hemos empezado a dar forma a la estructuración de los datos que van a ser almacenados en la base de datos para facilitar su uso en la aplicacion.

Durante el transcurso de este sprint, se presentó un error de sincronicidad entre componentes a la hora de pasar información entre ellos, lo cual requirió un tiempo adicional para solucionarlo.

Hemos invertido un total de 28 horas de trabajo hasta el inicio del tercer sprint.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 015	DOC	Creación de bocetos	4h	Completada
T - 016	DES	Aprendizaje del uso de Drag & Drop	3h	Incompleta
T - 017	DES	Estructurar la base de datos	6h	Completada
T - 018	DES	Investigar Firebase en React	4h	Completada
T - 019	DES	Aprendizaje React	4h	Incompleta
T - 020	ERR	Solucionar el estilo de un componente	1h	Completada
T - 021	ERR	Solucionar problema de sincronicidad entre componentes	5h	Completada
Total			**28h**	

Cuadro 2.5: Tareas del sprint 2

2.3.4. Sprint 3

En este sprint, vamos a crear la vista principal de la pagina web donde se hará uso de la mayoria de los componentes de la aplicación. Además, se empezará la creación del componente Tablero donde se ubicarán las piezas a colocar.

Tambien se ha añadido a la base de datos un ejemplo de nivel sobre el cual se van a ir realizando las pruebas de desarrollo de la aplicación.

Además hemos agregado una nueva página a la aplicación. Esta página se podrá acceder desde la página de inicio, y tiene como objetivo proporcionar instrucciones sobre cómo jugar y utilizar la aplicación que estamos desarrollando en este proyecto.

Durante el transcurso de este sprint, se va a redactar en la memoria lo realizado durante los sprints anteriores y acabar las tareas pendientes del anterior sprint.

El tiempo empleado en este sprint es de aproximadamente unas 25 horas de trabajo.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 022	DOC	Redactar sprints anteriores	4h	Completada
T - 023	DES	Creacion de prueba de nivel	1h	Completada
T - 024	DES	Creación de la vista principal de la aplicación	9h	Incompleta
T - 025	DES	Creacion de la vista de como jugar	4h	Completada
T - 026	DES	Desarrollo componente Tablero	7h	Incompleta
Total			**25h**	

Cuadro 2.6: Tareas del sprint 3

2.3.5. Sprint 4

En este sprint, vamos a crear tanto la funcionalidad como el diseño grafico del componente Pieza, el cual se va a arrastrar hasta el componente Tablero para poder completar el juego.

Durante la creación del componente Pieza, se vio necesario crear un componente adicional que abarcase todos los componentes Pieza que completan el tablero, para poder optimizar su funcionalidad y para un mejor manejo de los handler del Drag & Drop. Además, durante el diseño grafico de las piezas surgieron complicaciones debido al numero de piezas y el tamaño en pantalla, por lo que se tuvo que redimensionar el tamaño de las piezas para que pudiesen caber todas.

Tambien se tuvo que reestructurar como estaban almacenada las piezas en la base de datos, ya que de la forma anterior era mas complicado y confuso de interpretar la información.

Durante este sprint hemos invertido un total de 32 horas de trabajo.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 027	DOC	Documentar el propio Sprint y la sección de Scrum	4h	Completada
T - 028	DES	Creación componente Pieza	13h	Incompleta
T - 029	DES	Creación componente Piezas	3h	Completada
T - 030	EVOL	Reestructurar la base de datos	2h	Completada
T - 031	EVOL	Redimensionar el tamaño de las piezas y el tablero	4h	Completada
T - 032	ERR	Arreglar el handler del Drag & Drop de los componentes	6h	Completada
Total			**32h**	

Cuadro 2.7: Tareas del sprint 4

2.3.6. Sprint 5

Durante este sprint, se ha puesto como objetivo poder completar el tablero arrastrando todas las piezas al tablero en su posición correspondiente.

A la hora de arrastrar la pieza sobre el tablero, se encontró con el error de que al sujetar la pieza no se reconocía la posicion del tablero donde se estaba soltando la pieza debido a la superposición de elementos. Solucionar este problema llevo su tiempo, aunque fuese tan facil como recolocar el tablero en el eje Z a través del diseño grafico.

Una vez se consigó poder arrastrar la pieza y colocarla en el tablero, se vio necesario incorporar un elemento de retroalimentación para saber exactamente donde se colocaba la pieza en el tablero, por ende, se empezó a trabajar en generar una sombra en el tablero cuando se estuviese agarrando una pieza.

Dentro del manejador de eventos del *Drop*, se implementó una verificación para asegurar que la pieza se coloca en la posición correcta del tablero. Para lograr esto, se utilizó un algoritmo de fuerza bruta, ya que el número máximo de comprobaciones en la pieza es de 16, por lo que se consideró que no era necesario emplear un algoritmo más sofisticado y complejo. En su lugar, se optó por utilizar un enfoque más simple y directo.

En este sprint, el tiempo invertido ha sido un total de 36 horas de trabajo.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 033	DOC	Redactar las tecnologias empleadas	6h	Completada
T - 034	DOC	Documentar el sprint realizado	1.5h	Completada
T - 035	DES	Verificación de la pieza en el tablero	8h	Completada
T - 036	DES	Redimensionar la pieza durante el evento Drag	1h	Completada
T - 037	EVOL	Generar la sombra de la pieza en el tablero	9h	Completada
T - 038	ERR	Limpiar la sombra generada por la pieza	3h	Completada
T - 039	ERR	Solucionar la superposición de componentes	5h	Completada
Total			**33h**	

Cuadro 2.8: Tareas del sprint 5

2.3.7. Sprint 6

En el comienzo de este sprint se ha propuesto como objetivo crear, guardar y mostrar el tiempo que tarda el usuario en completar el juego.

De primeras se ha empezado reestructurando la base de datos, incorporando una nueva colección llamada *clasificacion* donde se guardarán por niveles los tiempos de cada usuario.

Para poder saber el tiempo transcurrido durante el juego, se ha añadido un componente llamado *Contador*, el cual mostrará por pantalla el tiempo transcurrido.

Una vez finalizado el juego, es decir, completado el tablero con las piezas disponibles, se ha creado un componente de tipo Pop-Up en el que se muestra los cinco primeros usuarios con los mejores tiempos en ese nivel y además se pedirá al usuario que introduzca un nombre para poder guardar su clasificación en la base de datos. El nombre que se introduzca a la base de datos estará controlado por una expresión regular para evitar la inyección de codigo malicioso, y por lo tanto, que puedan manipular nuestro sistema.

También se ha implementado una nueva vista en la que se pueden ver las clasificaciones de los usuarios en cada nivel del juego. Para mostrar estas clasificaciones en forma de ranking, hemos utilizado el algoritmo de ordenación por defecto de JavaScript, conocido como *QuickSort*.

El tiempo empleado en este sprint es de aproximadamente unas 32 horas de trabajo.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 040	DOC	Documentar el sprint realizado	1,5h	Completada
T - 041	DOC	Busqueda de algoritmos de ordenación	2h	Completada
T - 042	DES	Creación vista Ranking	11h	Incompleta
T - 043	DES	Creación de componente Clasificacion	10h	Completada
T - 044	DES	Validación de datos	2h	Completada
T - 045	DES	Creación componente Contador	4h	Completada
T - 046	DES	Almacenar datos en Firebase	1h	Completada
T - 047	EVOL	Reestructurar base de datos	1h	Completada
Total			**32h**	

Cuadro 2.9: Tareas del sprint 6

2.3.8. Sprint 7

Durante el transcurso de este sprint, hemos finalizado las tareas pendientes del sprint anterior, y a su vez hemos dado mas funcionalidad a la aplicación.

En la pantalla de principal, se ha añadido un nuevo botón que ayudará al usuario a resolver el juego dandole una pista. Dicha pista consta de mostrar en el tablero la forma de la pieza que corresponde a esa posición. Además, al usar una pista se añadirá al contador de tiempo 20 segundos, y dará al usuario unos 6 segundos para buscar la pieza que corresponda al hueco señalado.

Tambien se ha añadido una dificultad a los niveles haciendo que para completar un nivel haya un tiempo determinado para finalizarlo, es decir, que el contador irá hacia atras hasta que llegue a cero. Si no se ha completado el juego en el tiempo dado se cerrará el nivel y devolverá al usuario a la vista donde se muestran todos los niveles.

En este sprint, el tiempo invertido ha sido un total de 31 horas de trabajo.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 048	DOC	Documentar el sprint realizado	1.5h	Completada
T - 049	DOC	Redactar las tecnologías empleadas	13h	Completada
T - 050	DES	Crear la funcionalidad de las pistas	8h	Completada
T - 051	DES	Añadir dificultad a los niveles	5h	Completada
T - 052	DES	Terminar la vista Ranking	3h	Completada
T - 053	ERR	Modificar componente Contador	0.5h	Completada
Total			**31h**	

Cuadro 2.10: Tareas del sprint 7

2.3.9. Sprint 8

En este ultimo sprint nos enfocaremos en realizar pruebas exhaustivas de la aplicación con el objetivo de identificar posibles fallos o errores. Si encontramos algún bug durante las pruebas, lo agregaremos como tarea al sprint y trabajaremos en su resolución.

El enfoque principal de este sprint será completar la sección de la memoria que nos quedó pendiente del sprint anterior. Hay varias secciones que aún faltan por implementar, como el plan de riesgos y costes, así como la parte de análisis. Aunque ya hemos realizado los diagramas y hemos estado analizándolos, aún no los hemos incorporado a la memoria.

Después de considerarlo detenidamente, se ha llegado a la decisión de ajustar el tiempo requerido para resolver los niveles del juego. Segun vayan avanzando los niveles, el usuario tendrá menos tiempo disponible para completarlos, lo que aumentará el desafío, e incluso en los ultimos niveles se ha decidido deshabilitar las pistas para que se aun más complicado.

Además, hemos estado revisando la aplicación y hemos notado que en algunas áreas y botones es necesario proporcionar retroalimentación visual al usuario para mejorar la experiencia de uso.

En esta ultima iteracción se han consumido un total de 32 horas de trabajo para completar todo.

Tarea	Tipo	Descripción	Tiempo	Estado
T - 054	DOC	Redactar Plan de riesgos	6h	Completada
T - 055	DOC	Redactar Análisis del proyecto	5h	Completada
T - 056	DOC	Redactar Implementación	3h	Completada
T - 057	DOC	Redactar Conclusiones	1h	Completada
T - 058	DOC	Redactar Manual de usuario	3h	Completada
T - 059	ERR	Búsqueda de bugs en la aplicación	4h	Completada
T - 060	ERR	Mostrar retroalimentación	7h	Completada
T - 061	EVOL	Añadir más niveles en la base de datos	3h	Completada
Total			**32h**	

Cuadro 2.11: Tareas del sprint 8

Capítulo 3

Tecnologías utilizadas

3.1. React.js

React.js [21] es una biblioteca de JavaScript de código abierto y de alto rendimiento utilizada para construir interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Fue desarrollada por Facebook, y a día de hoy, se ha convertido en una de las tecnologías más populares para desarrollar aplicaciones web.

Hoy en día muchas empresas de primer nivel utilizan ReactJS para el desarrollo de sus aplicaciones, y es que entre ellas podemos encontrar Facebook, Instagram y el cliente web de WhatsApp, y otras como AirBnb, Uber, Netflix, Twitter, Reddit o Paypal.

La principal característica de React.js es su enfoque en la construcción de componentes reutilizables. En lugar de trabajar directamente con DOM [29] (*Document Object Model*) como en las habituales aplicaciones web, React.js utiliza un modelo de programación basado en componentes, donde cada componente representa una parte específica de la interfaz de usuario. Estos componentes pueden ser reutilizados en diferentes partes de la aplicación, lo que facilita el mantenimiento y la organización del código.

A continuación se resumirán las funcionalidades y caracteristicas más importantes que dispone React:

- **Modularidad:** el uso de componentes ofrece una gran ventaja para el desarrollo de grandes aplicaciones, ya que creando componentes independientes y reutilizables se obtiene una mayor modularidad y legibilidad del código, y permite trabajar en paralelo en varios componentes sin ningun problema.
- **Estados y reactividad:** los estados de React nos permiten realizar muchas acciones, como por ejemplo volver a renderizar un componente tras el cambio de un estado sin necesidad de ninguna acción del usuario .

- **Flujo de datos unidireccional:** este patrón de funcionamiento hace que los componentes superiores propaguen los datos a los que están en un orden inferior. Estos trabajarán con esos datos y cuando cambian su estado envíaran los eventos hacia los componentes de orden superior para actualizarse. Mantenindo un flujo unidireccional, el mantenimiento de la aplicación será mucho más sencillo.
- **Virtual Dom:** en React se utiliza un concepto llamado Virtual DOM para mejorar la eficiencia en la actualización de la interfaz de usuario. El Virtual DOM [20] es una representación en memoria del DOM real, que es la estructura que representa la página web y sus elementos. Actúa como un intermediario entre el estado de la aplicación y el DOM visible para el usuario. Cuando se realiza una actualización en un componente, se actualiza automáticamente los componentes padres necesarios para reflejar los cambios. Sin embargo, los componentes hermanos no se ven afectados y no se vuelven a renderizar, lo que mejora la eficiencia de la aplicación.

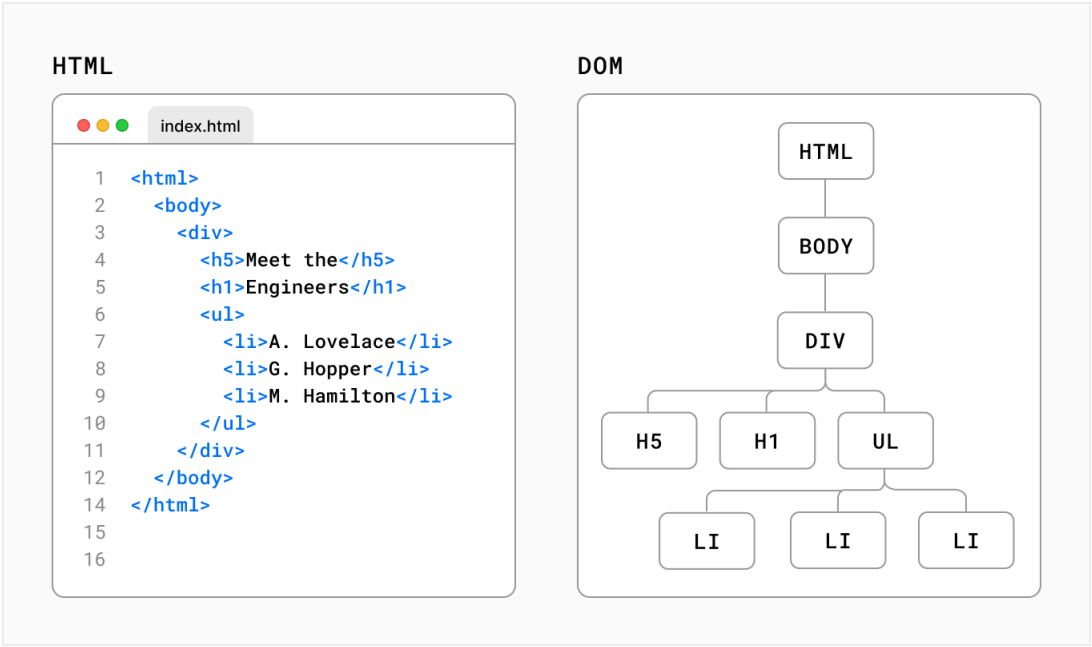

Figura 3.1: Arbol DOM

- **Isomorfismo:** es la capacidad de ejecutar el código tanto en el cliente como el servidor. También se conoce como "Javascript Universal". Sirve principalmente para solucionar problemas de posicionamiento tradicionales de las aplicaciones Javascript.
- **Elementos y JSX:** React no retorna HTML. El código embebido dentro de Javascript, parece HTML pero realmente es JSX. Son como funciones Javascript, pero expresadas mediante una sintaxis propia de React llamada JSX. Lo que produce son elementos en memoria y no elementos del DOM tradicional, con lo cual las funciones no ocupan tiempo en producir pesados objetos del navegador sino simplemente elementos de un DOM virtual. Todo esto, como hemos dicho, es mucho más rápido.

3.1.1. Ciclo de vida de un componente

Las aplicaciones desarrolladas con React, están formadas por componentes. Estos componentes tienen distintos estados desde que se crean, hasta que el componente se destruye [15]. En esta imagen vamos a ver los estados por los que pasa y vamos a explicar los más importantes.

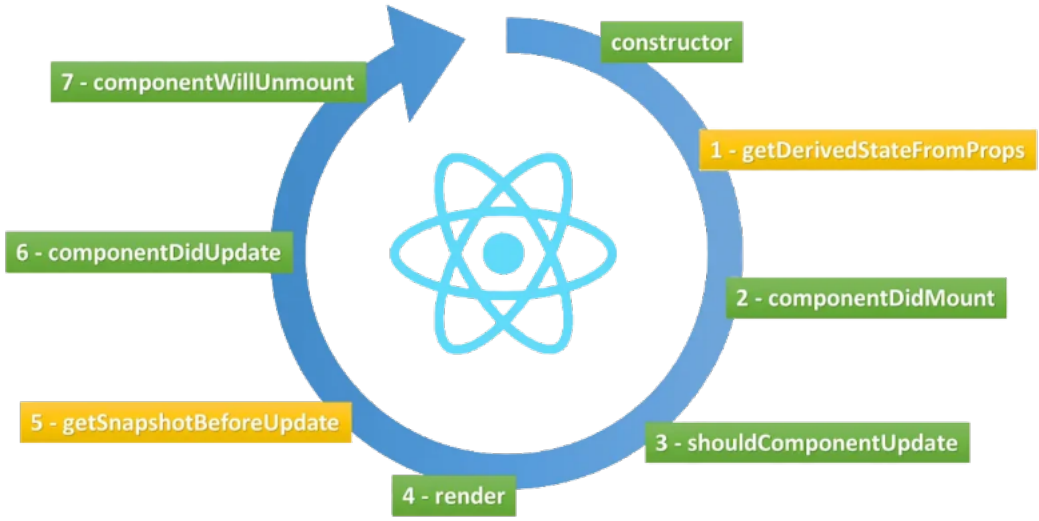

Figura 3.2: Componente React

- **componentDidMount():** se invoca inmediatamente después de que un componente se monte (se inserte en el árbol). Se utiliza cuando se va a cargar datos desde una API o si se necesita cargar datos desde un punto final remoto. Este es un buen lugar para instanciar la solicitud de red.
- **shouldComponentUpdate():** permite al desarrollador prevenir el re-renderizado innecesario de un componente, devolviendo falso si no es necesario.
- **render():** es el unico metodo requerido en un componente de clase. Este método es llamado cada vez que se necesita actualizar el componente, y además, no modifica el estado del componente, devuelve el mismo resultado cada vez que se invoca y no interactúa directamente con el navegador.
- **componentDidUpdate():** este método se invoca inmediatamente después de que la actualización del render ocurra. Este método no es llamado para el renderizador inicial.
- **componentWillUnmount():** se invoca inmediatamente antes de desmontar y destruir un componente. Realiza las tareas de limpieza necesarias en este método, como la invalidación de temporizadores, la cancelación de solicitudes de red o la eliminación de las suscripciones que se crearon en componentDidMount(). Basicamente elimina cualquier dependencia que tenga el componente.

3.1.2. Hooks

Los Hooks [22] son una nueva incorporación a partir de React 16.8.0, que nos permiten "enganchar" el estado y el ciclo de vida en componentes basados en funciones. Los dos hooks más utilizados son:

- **useState:** permite agregar y manipular el estado en componentes funcionales. Proporciona una forma de declarar variables de estado y actualizarlas. Contiene un par de valores: el valor del estado y una función para cambiar su valor. Es similar a this.setState en una clase.
- **useEffect:** se utiliza para ejecutar código en respuesta a cambios en el componente, como hacer una llamada a una API o suscribirse a eventos. Además, se ejecuta cada vez que se renderiza el componente o segun se cambie la variable que especifiques. Tiene un comportamiento similar a *componentDidMount()* y *componentDidUpdate()*.

3.2. NPM

Node Packager Manager (NPM) [13] es el gestor de paquetes por excelencia de Javascript y Node.js. Es una herramienta que permite descargar, instalar y administrar paquetes de código reutilizable en proyectos de JavaScript y Node.js. NPM viene incluido con la instalación de Node.js y se ejecuta en la línea de comandos.

NPM permite acceder a un amplio repositorio de paquetes de software públicos y privados que pueden ser utilizados en proyectos. Estos paquetes pueden contener librerías, frameworks, módulos y otras dependencias que pueden ser utilizadas para añadir más funcionalidad a una aplicación o facilitar el desarrollo de software.

3.2.1. Router

Es un plugin que nos permite cambiar la navegación de la página entre las distintas vistas y componentes. Tenemos también varios componentes que podemos utilizar dependiendo de las características de la página.

3.2.2. React Bootstrap

React Bootstrap [17] es una biblioteca que combina las funcionalidades de React.js y Bootstrap, un popular framework de diseño web. Proporciona componentes predefinidos de interfaz de usuario listos para usar, que están construidos con React y siguen las convenciones y estilos de diseño de Bootstrap, lo que facilita el desarrollo de aplicaciones complejas.

React Bootstrap simplifica el desarrollo de aplicaciones web al ofrecer una amplia variedad de componentes reutilizables, como botones, formularios, barras de navegación, tarjetas,

modales y muchos más. Estos componentes se pueden integrar fácilmente en una aplicación React.js sin necesidad de escribir código adicional complejo.

3.3. Firebase

Firebase [26] es una plataforma de desarrollo de aplicaciones móviles y páginas web creada por Google. Proporciona un conjunto de servicios y herramientas que permiten a los desarrolladores construir aplicaciones de gran calidad de una forma más rápida y sencilla. En nuestra aplicación final solo se ha usado la siguiente funcionalidad:

- **Firestore Database:** es una base de datos de documentos NoSQL que permite almacenar, sincronizar y consultar fácilmente datos desde la nube de forma sencilla y segura.

3.4. Git

Git [6] es un sistema de control de versiones que se ha convertido en una herramienta fundamental para la gestión de proyectos y el seguimiento de cambios en el código fuente. Además, permite a los desarrolladores trabajar de manera colaborativa y controlar el historial de cambios en un repositorio de código. Permite realizar un seguimiento de las modificaciones, crear ramas para trabajar en paralelo, fusionar cambios, revertir modificaciones anteriores y gestionar conflictos.

3.4.1. Github

Github [31] es una plataforma de alojamiento de repositorios de control de versiones basada en Git. Proporciona un entorno en línea donde los desarrolladores pueden gestionar, almacenar y colaborar en proyectos de software utilizando Git. No requiere de pago, por lo que se utiliza para muchos proyectos personales e incluso empresariales.

3.5. Trello

Trello [22] es una herramienta de gestión de proyectos en línea que permite a los equipos organizar y realizar un seguimiento de sus tareas de manera colaborativa. Proporciona una interfaz intuitiva y visual basada en tableros, listas y tarjetas, que facilita la gestión de proyectos y el seguimiento del progreso.

En esta herramienta, los usuarios pueden crear tableros para representar proyectos o áreas de trabajo. Dentro de cada tablero, pueden crear listas para organizar las tareas en etapas o

categorías, y luego añadir tarjetas dentro de esas listas para representar tareas o elementos individuales.

En este proyecto, lo hemos utilizado para llevar un seguimiento de las tareas e historias de usuario que teniamos pendientes, y poder organizar y aclarar las tareas a realizar.

3.6. LaTeX

LaTeX [19] es un sistema de composición de textos ampliamente utilizado para la creación de documentos científicos y técnicos de alta calidad.

En lugar de centrarse en el formato visual, LaTeX se enfoca en la estructura lógica y el contenido del documento. Los usuarios escriben el texto utilizando comandos y macros de LaTeX para indicar la estructura del documento, como títulos, secciones, párrafos, ecuaciones matemáticas, citas bibliográficas, etc. En resumen, LaTeX se encarga de la composición tipográfica y genera un documento final con un diseño profesional y coherente.

3.6.1. Overleaf

Overleaf [2] es una plataforma en línea que permite la edición, colaboración y compilación de documentos LaTeX. Es ampliamente utilizada por la comunidad académica y científica para crear y compartir documentos científicos y técnicos. Se asocia con una amplia gama de editoriales científicas para proporcionar plantillas oficiales de LaTeX. Es la herramienta empleada para desarrollar esta memoria.

3.7. Astah

Astah [7] es una herramienta de modelado y diseño de software utilizada para visualizar y representar gráficamente diferentes aspectos de un sistema, como diagramas UML (*Unified Modeling Language*), diagramas de flujo, diagramas de clases, diagramas de secuencia, etc.

Astah es la herramienta que hemos empleado en nuestro proyecto para realizar el diagrama de modelo de dominio, el diagrama de casos de uso, el diagrama de secuencia y el de componentes.

3.8. Netlify

Netlify [14] es una plataforma de alojamiento web y servicios de desarrollo centrada en la implementación y la entrega continua de sitios web estáticos. Proporciona una forma sencilla y eficiente de hospedar y distribuir sitios web estáticos, así como ofrecer funcionalidades adicionales como despliegue automatizado, integración con control de versiones y servicios de redireccionamiento y formularios. Gracias a esta web se ha podido desplegar nuestra aplicación web desde un repositorio Git.

Capítulo 4

Análisis del proyecto

4.1. Introducción

En este capítulo, nos enfocaremos en mostrar los requisitos funcionales y requisitos no funcionales [24] que debe satisfacer la aplicación. Estos requisitos nos ayudarán a definir qué funcionalidades y características debe tener nuestra aplicación. También vamos a ver las acciones que puede realizar cada tipo de usuario con la aplicación, nombrándolas en el diagrama de casos de uso. Además, vamos a mostrar en el modelo de dominio todas las entidades que participan, con los atributos y relaciones que tiene con otras entidades.

4.2. Requisitos funcionales (RF)

Los requisitos funcionales son las especificaciones de los servicios y funcionalidades que nuestra aplicación debe proporcionar. Son las declaraciones que describen qué acciones o tareas debe ser capaz de realizar la aplicación. En nuestro proyecto, estos requisitos funcionales están estrechamente relacionados con las historias de usuario que hemos identificado en el marco del desarrollo Scrum.

En la tabla 4.1 enumeramos los requisitos funcionales.

ID	Descripción
RF-1	El sistema deberá permitir que el usuario seleccione un nivel para jugar.
RF-2	El sistema deberá permitir mostrar a los usuarios la clasificación del resto de usuarios en cada nivel.
RF-3	El sistema deberá permitir que el usuario pueda arrastrar las piezas para completar el juego.
RF-4	El sistema deberá permitir mostrar al usuario una pista en el juego, si asi lo requiere.
RF-5	El sistema deberá permitir que el usuario pueda guardar su clasificación al final del juego.

Cuadro 4.1: Requisitos funcionales

4.3. Requisitos no funcionales (RNF)

Los requisitos no funcionales se centran en las cualidades, características y restricciones que deben tener nuestra aplicación, en lugar de describir funcionalidades específicas. A diferencia de los requisitos funcionales, que se enfocan en qué puede hacer la aplicación, los requisitos no funcionales se centran en cómo debe ser la aplicación. Podemos verlos los requisitos no funcionales de la aplicación en la tabla 4.2.

ID	Descripción
RNF-1	El sistema deberá funcionar con React
RNF-2	El sistema deberá acceder a Firebase para obtener los datos de la aplicación.
RNF-3	El sistema deberá permitir el acceso desde cualquier ordenador.
RNF-4	El sistema deberá disponer de conexión a internet.

Cuadro 4.2: Requisitos no funcionales

4.4. Casos de uso

Los casos de uso son una técnica utilizada en el análisis y diseño de sistemas para describir las interacciones entre los usuarios y el sistema. Representan las diferentes formas en que los usuarios interactúan con el sistema.

En esta sección veremos los casos de uso que se pueden presentar en nuestra aplicación, los cuales podemos observar en la figura 4.1.

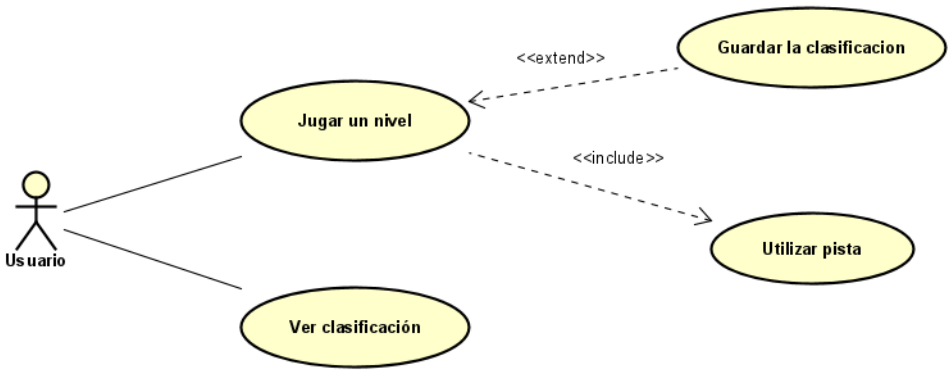

Figura 4.1: Diagrama de casos de uso

Principalmente, el usuario tendrá dos objetivos a realizar en nuestro sistema, los cuales serían poder jugar al nivel y ver las clasificaciones de los usuarios por niveles.

Durante el juego, el usuario tendrá acceso a un botón que le ofrecerá una pista para ayudarlo a resolver el nivel de manera más rápida. Una vez que el usuario haya completado el nivel, se le pedirá que guarde su tiempo de resolución junto con un nombre que elija libremente.

Cada caso de uso proporciona una descripción detallada de su funcionamiento, incluyendo las condiciones iniciales (precondiciones) y las condiciones resultantes (postcondiciones) que deben cumplirse. Además, se enumeran las secuencias de acciones que pueden ocurrir simultáneamente y se identifican las posibles excepciones que podrían surgir durante la ejecución del caso de uso.

En la figura 4.2, podemos observar como hemos detallado el caso de uso principal de nuestra aplicación.

ITEM	VALUE
UseCase	Jugar un nivel
Summary	El usuario debe arrastrar todas piezas al tablero para completar el juego.
Actor	Usuario
Precondition	1. El usuario debe haber seleccionado un nivel
Postcondition	1. El usuario debe poder ver su puntutación en la pantalla de clasificación.
Base Sequence	1- El usuario selecciona el nivel a jugar. 2- El sistema hace una petición a la base de datos para disponer de los datos del nivel. 3- El sistema le devuelve al usuario la interfaz del nivel. 4- El usuario arrastra las piezas al tablero para completar el juego. 5- El sistema muestra un Pop-up con las 5 mejores clasificaciones. 6- El usuario introduce el nombre con el que quiera registrar su tiempo. 7- El sistema guarda los datos en la base de datos. 8- El sistema enseña una retroalimentación al usuario antes de cambiar la pa ntalla a la de inicio.
Branch Sequence	4.1.1- El usuario pulsa el boton para disponer de una pista. 4.1.2- El sistema selecciona una pieza al azar de las disponibles. 4.1.3- El sistema muestra la sombra de la pieza en el tablero durante 6 segun dos.
Exception Sequence	4.1.1- El usuario coloca una pieza donde no corresponde. 4.1.2- El sistema vuelve a colocar la pieza donde estaba. 4.2- El sistema muestra que se ha acabado el tiempo y cambia de pantalla a l a de inicio.
Sub UseCase	Utilizar pista

Figura 4.2: CU de jugar un nivel

4.5. Modelo de dominio

En el desarrollo de software, un modelo de dominio [8] se utiliza para comprender y representar la estructura, entidades principales, las relaciones y las reglas del dominio en el que se desarrolla una aplicación.

Nuestra aplicación dispone de un modelo de dominio que podemos ver representado en la figura 4.3, y a continuación explicaremos brevemente el contenido de cada una de las tablas.

Cada nivel dispone de una serie de atributos, los cuales la mayoria estan almacenados en la base de datos. Un nivel dispone de un tablero con 270 celdas, en las cuales estarán almacenadas las letras que formaran la frase a completar. Los niveles dispondrán de aproximadamente de unas 40 piezas, las cuales constan solo de 16 celdas cada una. Si las celdas de las piezas estan vacias (representadas con un cero si esta vacias), no se mostrarán en pantalla, dando asi lugar a cada pieza que el usuario podrá mover para jugar el nivel.

Cabe destacar que cada nivel tendrá unos atributos necesarios para el uso del Drag & Drop, como puede ser almacenar momentaneamente la pieza mientras el usuario la esta

agarrando o la posición sobre la que se encuentra el cursor del usuario.

Si el nivel se ha resulto por un usuario al menos una vez, dispondrá de una clasificación representado con un array, donde cada elemento del array consta de una dupla del nombre del usuario y del tiempo que tardó en resolver el nivel.

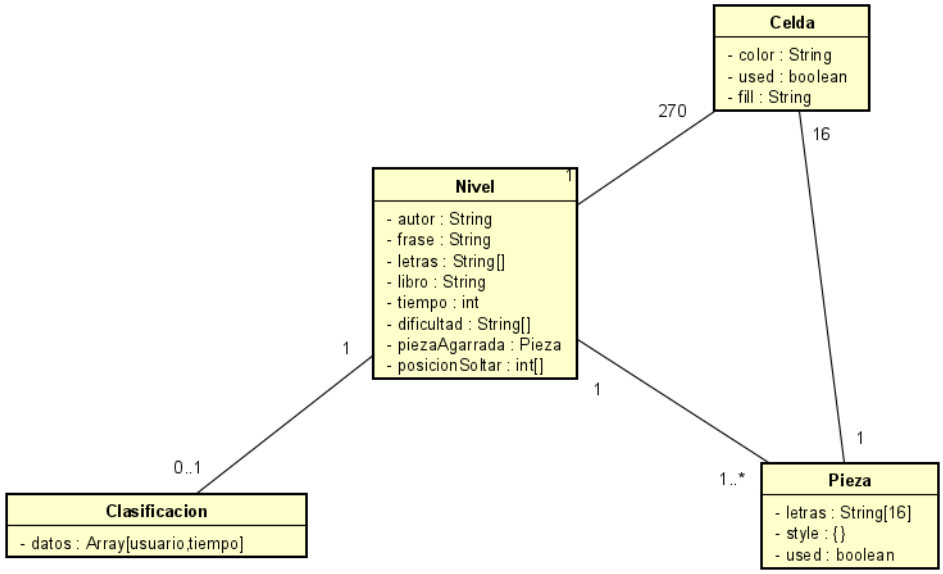

Figura 4.3: Diagrama de clases

4.6. Diagramas de secuencia

Un diagrama de secuencia [12] es una representación visual de cómo interactúan los diferentes objetos y componentes en un sistema a lo largo del tiempo. Se utiliza para mostrar la secuencia de acciones y mensajes que ocurren entre estos objetos durante la ejecución de un escenario específico. Son una solución de modelado dinámico que centran específicamente en líneas de vida o en los procesos y objetos que coexisten simultáneamente, y los mensajes intercambiados entre ellos para ejecutar una función antes de que la línea de vida termine.

A pesar de que hay muchos diagramas posibles en nuestra aplicación, vamos a describir 3 de ellos para ver como opera la aplicación

En el primer diagrama vamos a representar las acciones que puede realizar el usuario en el sistema una vez iniciado la pantalla de juego y como trabaja el sistema debido a ello. El

usuario tendrá la opción de hacer dos cursos de acción en el sistema mientras este presenta la pantalla de juego. El primero será de mover una pieza al tablero, y para ello realizará 3 acciones.

A la hora de agarrar la pieza, el sistema hará uso del manejador de eventos del Drag, el cual guardará momentaneamente la pieza en una variable y la modificará visualmente para dar retroalimentación al usuario de que se esta agarrando y coincida con el tamaño de las celdas del tablero.

Una vez agarrada, el usuario podrá mover la pieza al tablero, el cual dispondrá de otro manejador de eventos que detectará sobre que celda esta el cursor y cuando cambia de celda. Si la pieza cupiese donde el usuario la esta arrastrando, actuará otra función que coloreara solamente las celdas del tablero correspondientes al espacio donde se ubicara la pieza.

Cuando el usuario decide soltar la pieza, el sistema actuará haciendo uso del manejador de eventos del Drop, el cual comprobará si la sombra se ha dibujado en el tablero (lo cual indica que la pieza cabe en el tablero). Tambien se comprobará si la pieza corresponde a esa posición en el tablero, y si todo es correcto, se eliminará esa pieza y se verá reflejado en el tablero la pieza.

El otro curso de acción que puede realizar el usuario en el juego es pulsar un botón para que el sistema proporcione una pista al usuario. Una vez pulsado el botón, se seleccionará una pieza al azar de las disponibles, y se verá reflejada como una sombra en el tablero. Está acción durará solamente 6 segundos, durante los cuales el usuario no podrá volver a pulsar el botón para pedir una nueva pista. Además, acelerará el contador unos 20 segundos de tiempo como “pago” por la pista.

La Figura 4.4 representa visualmente toda la información y conceptos mencionados anteriormente.

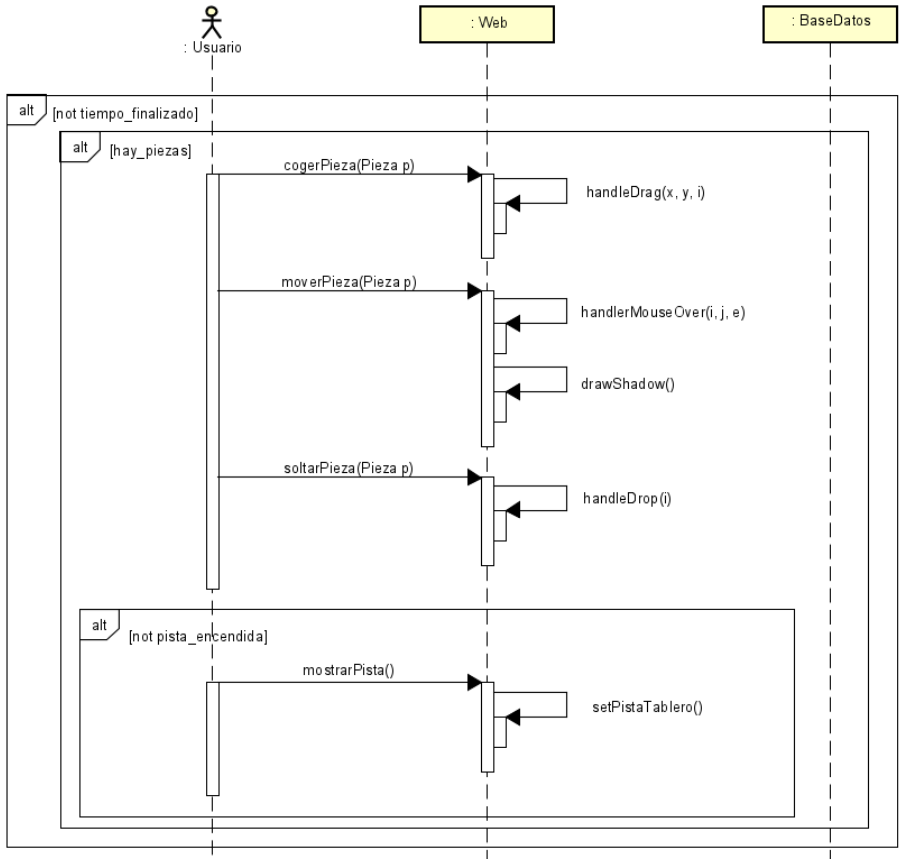

Figura 4.4: Secuencia durante el juego

En el segundo diagrama podemos ver como actua el sistema una vez el usuario ha completado el juego. El juego finalizará cuando el sistema detecte que no hay más piezas disponibles, por lo que detendrá el contador para saber el tiempo exacto que tardó el usuario en completar el juego y abrirá un Pop-Up para mostrar al usuario los 5 mejores tiempos en completar ese nivel por el resto de usuarios. En esta nueva ventana, el usuario podrá guardar el tiempo registrado junto con un nombre que el desee, aunque ese nombre solo podrá tener entre 6 y 15 caracteres, y sin ningun caracter extraño a excepción de "-", "_" y ".". Hemos controlado esto con una expresión regular para evitar así inyecciones de código malicioso en la base de datos.

Una vez comprobado que el nombre a guardar sea correcto, el sistema guardará la tupla del nombre y tiempo en la base de datos. Después, cerrará el Pop-Up y retroalimentará al usuario con una nueva ventana que durará solamente 1,2 segundos para indicarle que los datos han sido guardados correctamente, antes de cambiar a la ventana de inicio.

En la figura 4.5 se puede observar una representación visual de todos los conceptos e información discutidos anteriormente.

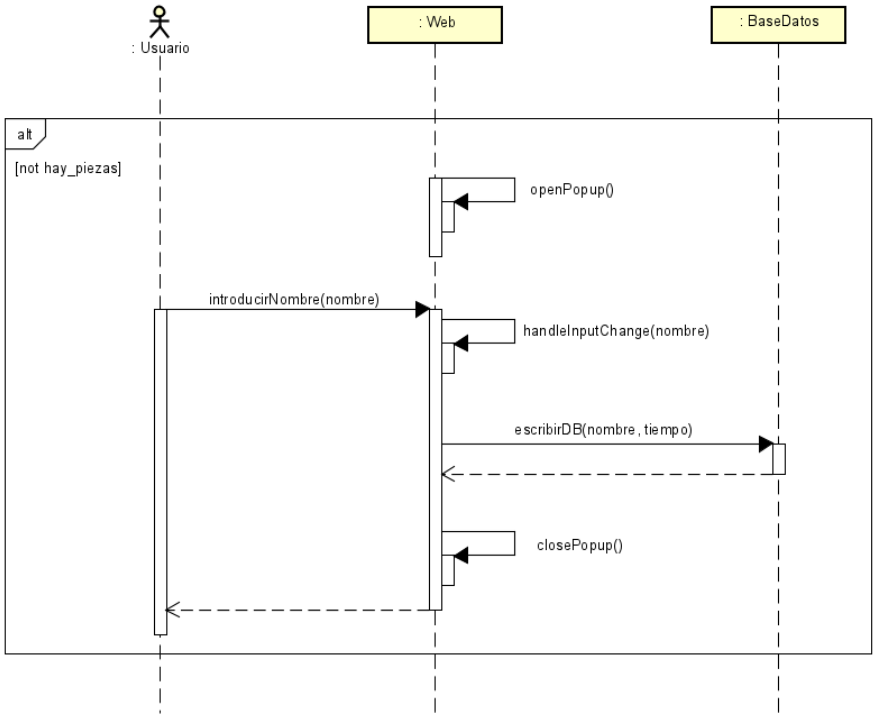

Figura 4.5: Secuencia al finalizar el juego

En este tercer diagrama de secuencia vamos a representar como actua el sistema una vez el usuario seleccione el nivel del cual desea ver las clasificaciones de los usuarios en dicho nivel. Una vez seleccionado el nivel, el sistema cambiará de pantalla donde se mostrará los nombres de los usuarios con su tiempo. Para especificar que datos coger de la base de datos, se especifica el nivel en la ultima parte del path de la pagina web, y a partir de esa información el componente hace una petición a la base de datos devolviendo todos los datos de ese nivel. Una vez tengamos los datos, se ordenan segun el tiempo para tener un ranking enumerado.

La Figura 4.6 representa visualmente de manera sencilla lo discutido anteriormente.

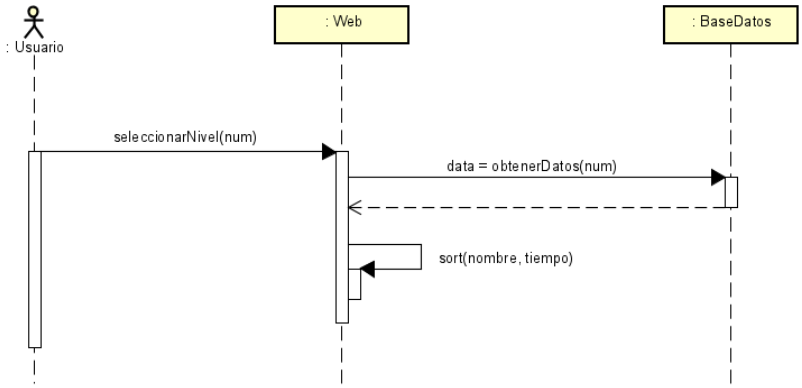

Figura 4.6: Secuencia al mostrar la clasificación

Capítulo 5

Diseño

En este capítulo se va a explicar los modelos y patrones utilizados. En lo referente al diseño arquitectónico se ha utilizado el patrón MVVM combinado con el diseño guiado por componentes, ya que se ha utilizado React.js. También vamos a explicar brevemente los modelos BaaS y SaaS junto con algunas ventajas y desventajas que nos proporcionan. Además explicaremos los bocetos de la interfaz que hemos realizado en nuestra aplicación.

5.1. SaaS

SaaS (*Software as a Service*) [25] es un modelo de distribución y de licencias usado para entregar aplicaciones de software a través de Internet, es decir, como un servicio. En lugar de adquirir y mantener el software localmente en sus propios servidores, los usuarios acceden a las aplicaciones a través de un navegador web o una interfaz de usuario dedicada.

En el modelo SaaS, el proveedor de servicios se encarga de alojar, mantener y actualizar el software, así como de administrar la infraestructura subyacente. Los usuarios solo necesitan de una conexión a internet y un dispositivo compatible para acceder al software y utilizarlo según sus necesidades.

En el proyecto, se ha utilizado el modelo SaaS al emplear Firebase como la base de datos para la aplicación, ya que asi evitamos encargarnos de la administración y mantenimiento de la infraestructura subyacente.

A continuación veremos algunas de las ventajas y desventajas de utilizar el modelo SaaS.

5.1.1. Ventajas de SaaS

- **Facilidad de implementación:** en nuestro caso se proporciona una interfaz y documentación sencillas que facilitan la integración de la base de datos en la aplicación.

- **Acceso desde cualquier lugar:** al utilizar una base de datos alojada en la nube, los datos son accesibles desde cualquier lugar y en cualquier momento, lo que permite a los usuarios utilizar la aplicación sin restricciones geográficas. Esto brinda agilidad, practicidad y usabilidad.
- **Adaptabilidad a las necesidades del cliente:** se adapta perfectamente a lo que necesites en tu empresa, pudiendo ampliar o reducir los paquetes, aumentando su eficiencia y logrando una personalización adecuada.
- **Ahorro en costes:** en nuestro caso, no es necesario que comprar el software, solo necesitas conexión a Internet.
- **Actualizaciones automáticas:** el proveedor del servicio se encarga de aplicar mejoras y actualizaciones a la plataforma, asegurándose de que la base de datos esté siempre actualizada y con nuevas características.
- **Seguridad:** los proveedores de servicios implementa medidas de seguridad para proteger los datos almacenados en la base de datos, como autenticación de usuarios, reglas de acceso y encriptación de datos en tránsito.

5.1.2. Desventajas de SaaS

- **Dependencia de red:** este servicio depende de tener conexión a internet.
- **Personalización limitada:** suelen ser estándar y diseñadas para satisfacer las necesidades generales de un amplio grupo de usuarios. Además, no se puede modificar el sistema, ya que es el proveedor el que tiene la potestad de modificar las funcionalidades.
- **Control de datos:** al utilizar un servicio SaaS, se almacenan los datos de la aplicación en servidores controlados por el proveedor. Esto plantea preocupaciones en términos de privacidad, seguridad y cumplimiento normativo.
- **Centralización:** al ser un servicio centralizado, un fallo deja sin servicio a todos los usuarios, y por ende, la aplicación dejaria de cumplir su función.

5.2. BaaS

BaaS (*Backend as a Service*) [16] es un modelo de servicio en la nube que proporciona a los desarrolladores las herramientas y la infraestructura necesarias para crear, ejecutar y gestionar la parte del servidor de una aplicación. En lugar de tener que desarrollar y mantener su propio backend desde cero, los desarrolladores pueden aprovechar un conjunto de servicios predefinidos y listos para usar que ofrecen proveedores de BaaS.

Este servicio automatiza el desarrollo del lado del backend y se encarga de la infraestructura en la nube, en nuestro caso del almacenamiento de datos con Firebase. Con BaaS, se subcontrata las responsabilidades de ejecución y mantenimiento de servidores a un tercero y se centra en el desarrollo del lado del cliente o de la interfaz.

En la siguiente figura 5.1 podemos ver un resumen de como funciona dicha arquitectura:

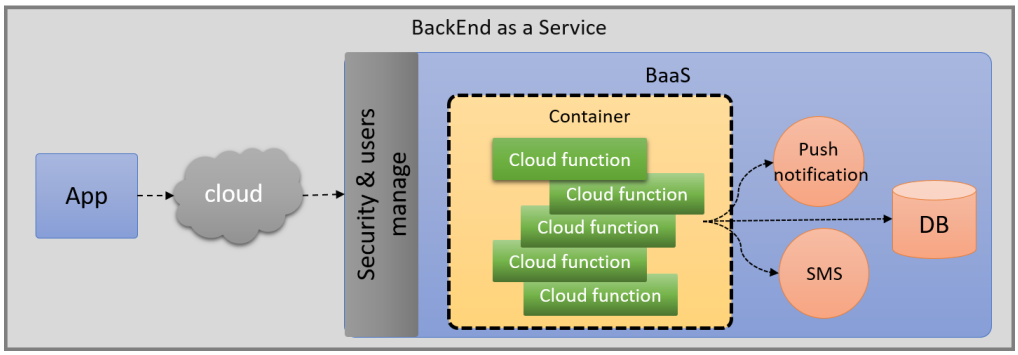

Figura 5.1: BaaS

A continuación, veremos un listado de ventajas y desventajas [10] que tiene utilizar este tipo de servicios.

5.2.1. Ventajas

El uso de BaaS ofrece una serie de ventajas para los desarrolladores y las empresas. Algunas de las ventajas más destacadas son:

- **Escalabilidad:** los proveedores de BaaS gestionan la infraestructura subyacente, lo que permite a las aplicaciones escalar fácilmente según las necesidades de uso y carga. Pueden crecer en función de la demanda del producto.
- **Rapidez en el desarrollo:** este tipo de servicio proporciona una infraestructura lista para usar, lo que permite centrarse en la lógica del cliente y acelerar el proceso de desarrollo. Al aprovechar los servicios predefinidos, se evita la necesidad de desarrollar y mantener un backend personalizado desde cero.
- **Seguridad:** la información almacenada se encuentra respaldada por copias de seguridad utilizando este tipo de servicio, evitando que nuestro sistema este expuesto a amenazas de hackers, desastres naturales y errores humanos. Además, los datos almacenados en BaaS están encriptados.
- **Reducción de costos:** se elimina la necesidad de invertir en hardware, infraestructura y personal especializado para gestionar el backend de la aplicación. Esto puede resultar en ahorros significativos en costos de desarrollo y mantenimiento, lo que puede permitir al personal de las empresas que utilizan este modelo disponer su tiempo en otras tareas.
- **Actualizaciones y mantenimiento simplificados:** los proveedores de este servicio se encargan de realizar actualizaciones y mantenimiento de la infraestructura subyacente, lo que libera a los desarrolladores de estas tareas.

5.2.2. Desventajas

Aunque el uso de BaaS ofrece muchas ventajas, también hay algunas desventajas que se deben tener en cuenta. Algunas de las desventajas más comunes son las siguientes:

- **Limitaciones de personalización:** este servicio ofrece menos opciones de personalización al ser un servicio externo, ya que se esta utilizando una infraestructura compartida que está diseñada para adaptarse a una amplia variedad de casos de uso
- **Dependencia de red:** este servicio depende de tener conexión a internet.
- **Privacidad:** al utilizar un servicio de BaaS, estás confiando en el proveedor para gestionar y proteger los datos de tu aplicación, lo cual siempre existe un riesgo de violaciones de seguridad o falta de control sobre la privacidad de los datos.

5.3. Patrón Model-View-ViewModel (MVVM)

En este proyecto hemos empleado el patrón MVVM [11] (*Model-View-ViewModel*). Es un patrón de arquitectura de software que se utiliza comúnmente en el desarrollo de aplicaciones con interfaz gráfica. Proporciona una separación clara de responsabilidades entre la lógica de negocio, la presentación de datos y la interfaz de usuario. Este patrón consta principalmente de 3 componentes:

- **View:** es la capa de presentación de la aplicación. Se encarga de mostrar la interfaz de usuario. La vista está vinculada al ViewModel para obtener los datos y actualizar la interfaz en consecuencia. En nuestro proyecto, está formado por los archivos JSX y CSS.
- **Model:** es la capa de datos de la aplicación. Aquí es donde se definen las estructuras de datos, las operaciones de lectura/escritura de la base de datos y cualquier lógica relacionada con los datos. Representa el modelo de dominio.
- **ViewModel:** actúa como un intermediario entre el modelo y la vista. Se encarga de proporcionar los datos necesarios para la vista y manejar las acciones y eventos del usuario. El ViewModel también puede contener la lógica de presentación, como el formateo de datos y la validación. Las propiedades y comandos que proporciona el modelo de vista definen la funcionalidad que ofrece la interfaz de usuario, pero la vista determina cómo se va a mostrar esa funcionalidad.

En la descripción de este patrón, podemos ver que es muy similar al patrón MVC (*Model-View-Controller*), pero se diferencian en que mientras el patrón MVC se centra en la separación de responsabilidades entre el modelo, la vista y el controlador, el patrón MVVM se centra en la vinculación de datos entre la vista y el ViewModel, lo que facilita la actualización de la interfaz de usuario y las pruebas unitarias.

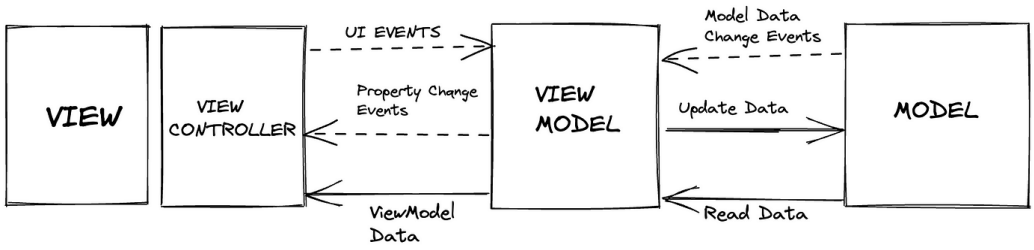

Figura 5.2: Patrón de diseño MVVM

Este modelo arquitectónico se adapta muy bien a la tecnología que empleamos en nuestro proyecto, React.js. A continuación hablaremos de las ventajas y desventajas que se pueden dar al usar este patrón MVVM.

5.3.1. Ventajas

- **Vinculación de datos bidireccional:** los cambios en el ViewModel se reflejan automáticamente en la vista, y viceversa.
- **Separación clara de responsabilidades:** MVVM permite una separación clara de las responsabilidades en el desarrollo de aplicaciones. La lógica de negocio está desacoplada de la interfaz de usuario.
- **Reutilización de código:** la reutilización de componentes nos permite ahorrar tiempo y esfuerzo, ya que no tenemos que escribir código desde cero cada vez que necesitamos una funcionalidad similar.

5.3.2. Desventajas

- **Curva de aprendizaje:** MVVM puede tener una curva de aprendizaje superior respecto a otros modelos, especialmente para aquellos desarrolladores que no están familiarizados con el patrón.
- **Complejidad adicional:** los desarrolladores deben de adaptarse a una estructura predefinida y eso incrementa la complejidad del sistema.
- **Mayor mantenimiento:** la distribución de componentes nos obliga a la creación y mantenimiento de un mayor número de ficheros.
- **Sobrecarga de implementación:** implementarlo correctamente puede requerir una planificación y una estructura adecuadas. Esto implica crear los modelos, vistas y ViewModels correspondientes, establecer las vinculaciones de datos adecuadas y garantizar que haya una comunicación fluida entre las capas.

5.4. Diseño guiado por componentes

El diseño guiado por componentes [1] es un enfoque de diseño de software en el que se construye una aplicación dividiéndola en componentes reutilizables e independientes. Cada componente representa una parte específica de la interfaz o una funcionalidad del sistema.

Basicamente al utilizar React.js, se esta empleando este tipo de diseño ya que React.js no puede funcionar sin él. Hoy en día es un diseño muy utilizado para el desarrollo de aplicaciones web.

A continuación vamos explicar algunas de las ventajas y desventajas de utilizar este tipo de diseño.

5.4.1. Ventajas

- **Reutilización de código:** este diseño permite crear componentes independientes y reutilizables en diferentes partes de la aplicación o incluso en otros proyectos, reduciendo el esfuerzo de desarrollo, mejorando la productividad y facilitando la mantenibilidad del código.
- **Consistencia en la interfaz de usuario:** al utilizar componentes reutilizables, se promueve la consistencia en la apariencia y el comportamiento de la interfaz de usuario. Esto crea una experiencia de usuario más coherente, reduciendo la posibilidad de errores.
- **Costos reducidos:** el uso de componentes de terceros permite compartir la carga económica y técnica del desarrollo y mantenimiento de una aplicación.
- **Facilidad de desarrollo:** los componentes ofrecen una interfaz clara y definida que encapsula la funcionalidad específica que proporcionan. Esto significa que se pueden desarrollar y modificar los componentes de forma independiente, sin afectar otras partes del sistema.
- **Mitigación de complejidad:** cuando nos encontramos con un problema complejo y requiere demasiado esfuerzo, la solución más sencilla es descomponer el problema en componentes más simples y analizar cada componente de manera individual.

5.4.2. Desventajas

- **Mayor complejidad inicial:** aumenta la complejidad para los diseñadores que tienen que desgranar la solución. Se requerir de un mayor esfuerzo y tiempo en comparación con otros enfoques.
- **Comprobar los test de cada componente:** requiere un mayor esfuerzo de testeo, ya que hay que comprobar la funcionalidad y compatibilidad de cada componente.

- **Dependencia de herramientas y frameworks:** en nuestro caso, los componentes creados estan estrechamente vinculado a nuestra herramienta de trabajo, React.js. Esto puede limitar la flexibilidad y la portabilidad del proyecto, ya que los componentes pueden estar acoplados a un entorno específico.

5.4.3. Componentes del proyecto

En esta sección vamos a explicar brevemente la funcionalidad de cada uno de los componentes creados en este proyecto. Como se puede ver en la figura 5.3 nos hemos centrado en crear los componentes necesarios para que la interfaz de la aplicación sea sencilla y el código desarrollado sea facilmente interpretable.

- **App:** componente inicial que contiene toda la aplicación y al cual estan asociados el resto de componentes.
- **Home:** componente que se muestra inicialmente una vez se inicie la aplicación.
- **ComoJugar:** componente que se compone unicamente de una imagen para explicar al usuario como se juega con la aplicación.
- **Ranking:** componente que muestra todos los niveles que hay en nuestra aplicación de los cuales puedes seleccionar para ver la clasificación de ellos.
- **RankingNivel:** componente que muestra la clasificación de un nivel en especifico.
- **Niveles:** componente que muestra todos los niveles que hay en nuestra aplicación de los cuales puedes seleccionar para poder jugar a ellos.
- **InfoJuego:** componente que recoge la información de la base de datos del contenido de cierto nivel.
- **Juego:** componente principal de la aplicación ya que mayormente recoge todo el funcionamiento de la mecanica del juego.
- **Tablero:** componente que muestra el tablero de juego.
- **Contador:** componente que muestra el tiempo del que dispone el usuario.
- **Piezas:** componente que almacena las piezas del juego.
- **Pieza:** componente que se arrastra al tablero para poder completar el juego.
- **Clasificación:** componente que se muestra al finalizar el juego y permite al usuario guarda su clasificación.

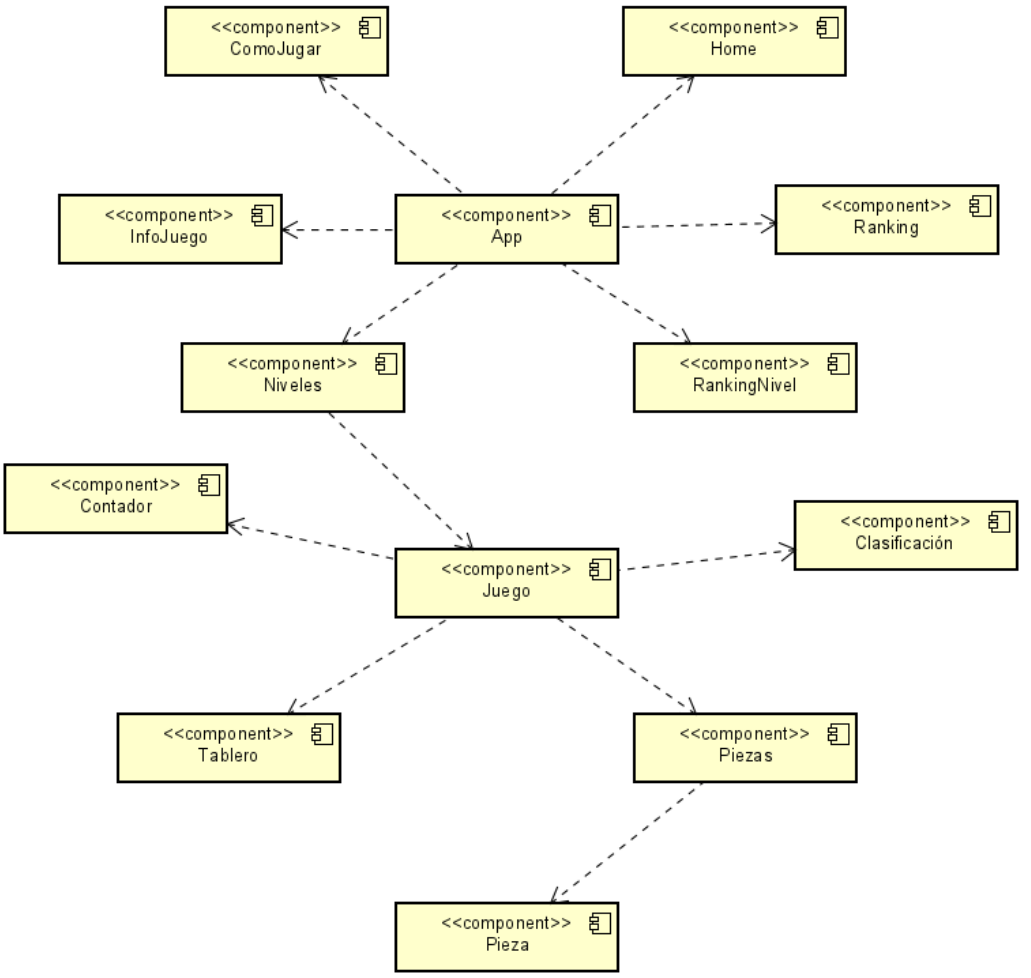

Figura 5.3: Diagrama de componentes

5.5. Modelo de despliegue

En este apartado vamos a explicar brevemente el modelo de despliegue [18] de nuestra web. Se ha utilizado un servicio de web hosting llamado Netlify, el cual unifica el código de nuestra aplicación que se encuentra ubicado en un repositorio Github, con los datos de nuestra base de datos de Firebase. Para que los usuarios puedan acceder a la página web, lo podrán hacer desde cualquier navegador web.

La estructura del modelo de despligue la podemos observar en la figura 5.4.

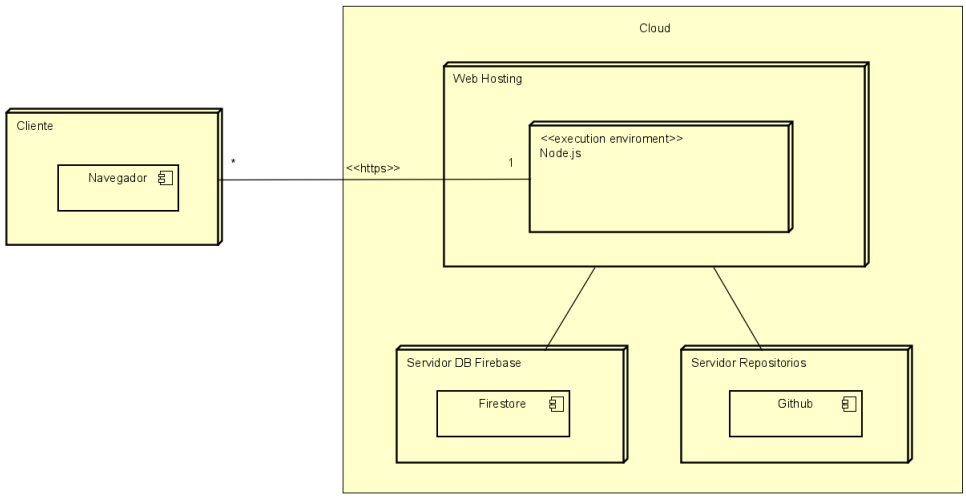

Figura 5.4: Diagrama de despliegue de nuestra aplicación

5.6. Base de Datos

En este apartado, vamos a hablar de la base de datos que hemos empleado en este proyecto, la cual está almacenada en la nube gracias a que hemos empleado un producto de Firebase llamado Firestore Database. Tambien explicaremos brevemente como hemos gestionado la estructura de los datos.

5.6.1. Firestore Database

Firestore Database es una base de datos NoSQL, lo que significa que no utiliza el modelo tradicional de tablas y filas, sino que almacena los datos en documentos JSON estructurados en colecciones.

Ha sido una herramienta fundamental en nuestra aplicación, ya que nos ha permitido crear una base de datos en la nube con soporte para diversos tipos de datos complejos, como arrays y mapas.

Se han creado solo dos colecciones de datos ya que no se vio necesario crear más, ya que de esta manera era más facil de entender y tratar los datos. Las colecciones creadas son las siguientes:

- **Niveles:** en esta tabla guardamos todos los datos que se utilizan en el componente Juego. Se han guardado como una variable de tipo string: el autor, la frase y el libro. Las letras iniciales del tablero se han guardado como un array de strings, donde cada string representa la posicion del tablero donde se ubica las letras iniciales.

  Respecto a las piezas, tambien se han almacenado en un array de strings donde cada elemento del array corresponde a una pieza, la cual esta estructurada como un string de 16 letras separadas por comas (la pieza tiene una estructura de 4x4).

  Por ultimo, la dificultad esta representado por un array de dos posiciones, en el que la primera posición corresponde al tiempo limite para resolver el nivel y la segunda posición corresponde a un booleano que indica si se permiten pistas en el nivel o no.

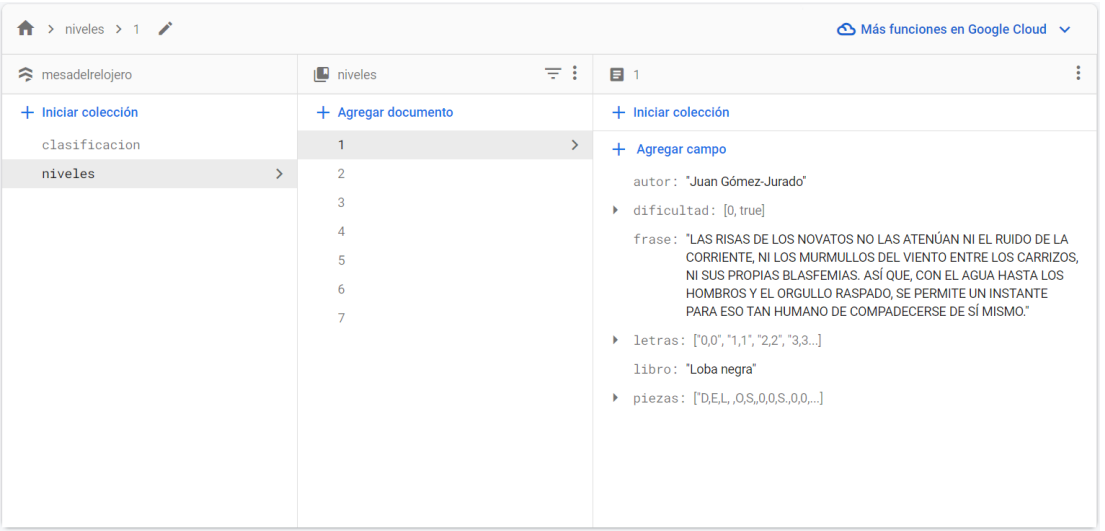

Figura 5.5: Estructura de la colección Niveles

- **Clasificacion:** en esta tabla guardamos todos las clasificaciones de los usuarios por niveles. Las clasificaciones de cada usuario se almacenaran como una tupla de campo/valor, donde el campo corresponde al nombre del usuario introducido y el valor corresponde a la marca de tiempo realizada por el usuario.

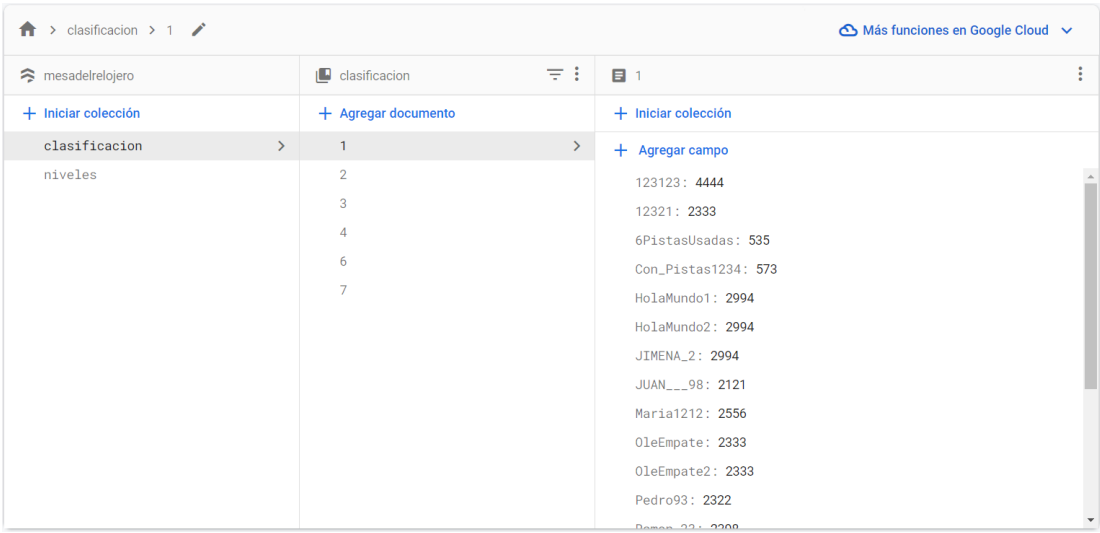

Figura 5.6: Estructura de la colección Clasificacion

Capítulo 6

Plan de riesgos y estimación de costes

6.1. Plan de riesgos

El riesgo es un suceso cuya probabilidad de que ocurra es incierta, y en caso de ocurrir, puede poner en peligro los objetivos de un proyecto o alterar el desarrollo previsto del proyecto.

En cualquier proyecto, siempre existen riesgos que pueden afectar el alcance de los objetivos establecidos dentro de los plazos previstos, así como los costos, recursos y funcionalidades planificadas. Estos riesgos pueden tener diferentes naturalezas y dependen del tipo de proyecto en cuestión.

Pueden darse varios tipos de riesgos durante el desarrollo de una aplicación:

- **Riesgos del proyecto:** afectan al desarrollo del proyecto pero en concreto al coste, la planificación y a la calidad. Suelen ser problemas de presupuesto, personal, de tiempo, etc.
- **Riesgos técnicos:** amenazan a la calidad del producto que estamos desarrollando. Suelen ser problemas de diseño, estructuración, obsolescencia técnica, implementación, etc.
- **Riesgos de negocio:** son riesgos que suponen una amenaza para el proyecto tanto el desarrollo como en la entrega y distribución del producto. Estos riesgos de negocio se suelen deber a riesgos de mercado, de presupuesto, de ventas, etc.

A lo largo del desarrollo de un proyecto es conveniente tener una adecuada planificació y gestión de riesgos. Estos riesgos se pueden clasificar según las posibilidades de predicción:

- **Predecibles y conocidos:** se pueden identificar analizando el proyecto y su entorno. Se puede estimar probabilidad de ocurrencia.
- **Predecibles:** Se pueden intuir a partir de la experiencia en otros proyectos, pero no se puede estimar la probabilidad de ocurrencia.
- **Impredecibles:** Pueden ocurrir, pero son difíciles de identificar con anticipación.

La procedencia de los riesgos es variada. Puede ser: técnica, administrativa, contractual, financiera, política, derivada de recursos inadecuados o insuficientes, de relaciones laborales, del entorno, de impacto social, por fuerza mayor, de mercado, etc.

Para realizar un plan de riesgos adecuado es recomendable seguir una serie de pasos:

1. Identificación de riesgos.
2. Análisis y establecimiento de prioridades.
3. Planificación de riesgos.
4. Monitorización de riesgos.

6.1.1. Riesgos encontrados en este proyecto

A continuación vamos a exponer un análisis de los riesgos de mayor importancia que se han encontrado durante el ciclo de vida del proyecto. Expondremos estos riesgos en formato de tablas, una para cada riesgo, en las que se ofrece una breve descripción del riesgo, la probabilidad de que pueda surgir (muy baja, baja, media, alta y muy alta), su impacto (muy baja, baja, media, alta y muy alta), un plan de mitigación que reduzca su probabilidad y un plan de contingencia.

Título	Riesgo-01
Descripción	Un miembro del equipo de desarrollo no dispone de la suficiente formación en las tecnologías empleadas para realizar el proyecto
Probabilidad	Media
Impacto	Medio
Plan de mitigación	Se habilita un curso de formación relacionado con esta tecnología
Plan de contingencia	Ayudar al miembro del equipo en el sprint inicial en sus primeras tareas y con su formación

Cuadro 6.1: Riesgo de falta de formación

Título	Riesgo-02
Descripción	Un miembro del equipo de desarrollo enferma y se retrasan las tareas que tenía asignadas.
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	Realizar estimaciones con un tiempo de margen para la entrega final del proyecto, creando un sprint de refuerzo final para evitar este tipo de infortunios
Plan de contingencia	Intentar retrasar algunas tareas para el proximo sprint

Cuadro 6.2: Riesgo de enfermedad

Título	Riesgo-03
Descripción	Mala planificación del tiempo de un algunas tareas o evolutivo
Probabilidad	Media
Impacto	Medio
Plan de mitigación	En la estimación del tiempo, disponer de un tiempo extra al final del proyecto, como un sprint de refuerzo final
Plan de contingencia	Intentar retrasar algunas tareas para el proximo sprint

Cuadro 6.3: Riesgo de falta de tiempo

Título	Riesgo-04
Descripción	El Scrum Master no se encuentra disponible en las reuniones importantes para el desarrollo del proyecto
Probabilidad	Media
Impacto	Medio
Plan de mitigación	Acordar reuniones con antelación para evitar este tipo de imprevistos
Plan de contingencia	Se le asigna una mayor responsabilidad a los desarrolladores durante esa iteración del sprint

Cuadro 6.4: Riesgo de ausencia del Scrum Master

Título	Riesgo-05
Descripción	Cambian los requisitos del sistema, añadiendo o quitando funcionalidad según se ha ido desarrollando la aplicación
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	Desarrollar de una manera modular, que permitirá facilitar los cambios de última hora
Plan de contingencia	Modificar los componentes anteriores

Cuadro 6.5: Riesgo de cambio de requisitos

Título	Riesgo-06
Descripción	Cambios o limitaciones en la tecnología para determinados requisitos
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	Realizar un análisis de los requisitos respecto a la tecnologia a emplear para comprobar que se puede realizar todos.
Plan de contingencia	Disponer de un tiempo extra en el proyecto creando un sprint de refuerzo final para evitar este tipo de problemas

Cuadro 6.6: Riesgo de limitaciones tecnológicas

Título	Riesgo-07
Descripción	Cortes en el servicio de base de datos debido a agentes externos.
Probabilidad	Muy baja
Impacto	Alto
Plan de mitigación	Disponer de una copia de los datos en archivos locales para cambiar a otro tipo de servicio
Plan de contingencia	Tener en cuenta otro tipo de servicio que pueda sustituir al servicio que no funcione

Cuadro 6.7: Riesgo de averías

6.2. Presupuesto

Este proyecto se ha creado con una única persona como desarrollador. Se ha realizado una consulta al salario base de una desarrollador junior [9] y el valor es de unos 21.205 euros al año, lo cual equivale a un salario mensual de 1.514 euros. Con este salario, calculamos que el ingreso por hora es de entorno a 9,47 euros. Se han invertido aproximadamente unas 300 horas en el proyecto, por lo que 300*9,47 supone un coste final de 2.841 euros en salarios.

Además, hay que añadir el valor del hardware empleado en el desarrollo, así como la amortización a lo largo del proyecto. Se ha utilizado un ordenador con una valoración aproximada de 900 euros. Revisando la tabla de coeficientes de amortización lineal [3], se puede comprobar que el coeficiente lineal máximo tiene un valor de un 20 %. Como el tiempo empleado ha sido alrededor de 5 meses, el coste del hardware es de 900*(0.2)*(5/12) que equivale a 75 euros.

Por último, el coste de herramientas utilizadas (software, licencias, etc) es de 0 euros, ya que las herramientas y servicios empleados son gratuitas y la licencia del Astah para desarrollar los diagramas es proporcionada gratuitamente por parte de la universidad.

El valor total del presupuesto final del proyecto sería de 2916 euros, ya que es lo que equivale la suma del salario de los programadores más el valor del hardware empleado.

Capítulo 7

Implementación

En este capítulo se va a explicar detalladamente la estructura del código y los cambios realizados a lo largo del desarrollo del proyecto.

7.1. Estructura del código del proyecto

Los ficheros del proyecto tienen la siguiente estructura:

```
|   .gitignore
|   package-lock.json
|   package.json
|   .env.local
|   README.md
|
+---public
|   index.html
|   logo192.png
|   logo512.png
|   favicon.ico
|   manifest.json
|   icono.png
|
+---node_modules (se omite su contenido por brevedad)
|
+---src
|   |   App.css
|   |   App.jsx
|   |   logo.svg
|   |   index.css
|   |   index.jsx
```

```
|   |
|   +---bd
|   |    +---Firebase.js
|   |
|   +---components
|   |    +---Botones.css
|   |    +---Botones.jsx
|   |    +---Clasificacion.css
|   |    +---Clasificacion.jsx
|   |    +---Contador.css
|   |    +---Contador.jsx
|   |    +---Pieza.css
|   |    +---Pieza.jsx
|   |    +---Piezas.css
|   |    +---Piezas.jsx
|   |    +---Tablero.css
|   |    +---Tablero.jsx
|   |
|   +---images
|   |   |    como_jugar.png
|   |   |    fondopantalla.png
|   |   |    icon.png
|   |   |    icono.png
|   |
|   +---view
|   |   +---ComoJugar.css
|   |   +---ComoJugar.jsx
|   |   +---Home.css
|   |   +---Home.jsx
|   |   +---InfoJuego.jsx
|   |   +---Juego.css
|   |   +---Juego.jsx
|   |   +---Niveles.css
|   |   +---Niveles.jsx
|   |   +---Ranking.css
|   |   +---Ranking.jsx
|   |   +---Rankingniveles.css
|   |   +---Rankingniveles.jsx
|
```

Los primeros archivos del arbol de directorios de nuestro proyecto (package-lock.json, package.json) corresponden a la configuración de la página web, donde especifican las dependencias y librerias utilizadas. Además dos archivos (.gitignore, README.md) que corresponden a un proyecto que a utilizado Git. El archivo .gitignore ignora los archivos que se encuentran en él a la hora de subir los cambios al repositorio. El archivo .env.local tiene almacenado toda la información sensible de como acceder a la base de datos, por lo que esta incluido en el gitignore.

Los archivos ubicados en el directorio *public* incluyen imagenes o archivos poco relevantes para el proyecto.

En cuanto al directorio *node-modules* tiene todas las dependencias y librerias del proyecto, las cuales no especificamos por brevedad.

En la carpeta *src*, a parte el archivo de arranque del proyecto (index.jsx), podemos encontrar otros 4 directorios (bd, components, images, view).

- **bd:** solo se encuentra el fichero Firebase.js, el cual nos permite traer una instancia de la base de datos de Firebase.
- **components:** directorio tiene almacenado todos los componentes necesarios para que funcione el juego.
- **images:** carpeta se encuentran todas las imagenes que se han utilizado para mostrar en la aplicación.
- **view:** directorio que posee todas las interfaces que llaman a los componentes de nuestra aplicación y le dan un formato y estructura a la vista.

7.2. Decisiones a lo largo del proyecto

Tanto antes de comenzar como durante todo el proceso del proyecto, se han tomado decisiones cruciales para la implementación de la página web. A continuación, destacaremos algunas de estas decisiones relevantes.

En primer lugar, se eligió utilizar React.js como el framework principal para el desarrollo en el Frontend. Esta elección se basó en la popularidad y amplio uso de React.js en la comunidad de desarrollo de JavaScript. A pesar de su curva de aprendizaje, se decidió optar por este framework ya que de cara al mundo laboral es muy beneficioso manejar este tipo framework tan utilizado.

La elección de Firebase como servicio de base de datos se basó en su facilidad de uso y en las funcionalidades que ofrecía. Firebase simplificó en gran medida la creación y gestión de las bases de datos, permitiendo ahorrar tiempo y esfuerzo en esta área.

7.3. Cambios realizados a lo largo del proyecto

A lo largo del ciclo de vida del proyecto, las tareas objetivo de algún sprint fueron modificadas debido a problemas que fueron surgiendo en el desarrollo de la aplicación.

Entre los cambios más importantes respecto a la planificación inicial del proyecto, se encuentra el despliegue de la página web. Inicialmente se pensó realizar con Azure Devops, pero debido a problemas de alguna de las variables de ambiente autentificación de Firebase, se decidió utilizar Netlify como herramienta de despliegue de la aplicación.

7.4. Licencias

Al ser un estudiante de la universidad informática de Valladolid, se ha podido pedir a la empresa Astah una licencia gratuita de una duración de 6 meses, el tiempo necesario para poder crear y explicar de manera gráfica la estructura de la aplicación y su funcionamiento.

Capítulo 8

Conclusiones y futuro

8.1. Conclusiones

Ya finalizando la memoria vamos a exponer una breves conclusiones obtenidas tras la realización de este proyecto y que posible trabajo se podría realizar en un futuro.

Una de las principales conclusiones obtenidas tras el desarrollo del proyecto, es que un trabajo de este tipo puede ser dificil de realizar por una sola persona. Se echa en falta el trabajo en equipo, tener un compañero para debatir las decisiones a tomar o para pedir ayuda.

También es importante tener una buena planificación para sacar adelante el proyecto, ya que han ido surgiendo imprevistos durante los meses de desarrollo que han complicado que algunas tareas marcadas para un sprint se hayan tenido que aplazar.

Desarrollar esta aplicación web en React.js ha sido una buena ida, ya que la mayoria de empresas trabajan con este tipo de framework o una estructura parecida, por lo que ha sido una apuesta a futuro que va a ser bastante beneficiosa personalmente.

Consideramos la posibilidad de adaptar el juego para que sea accesible en dispositivos electrónicos distintos a los ordenadores, sin embargo, se llegó a la conclusión de que no sería una buena idea debido a las limitaciones de pantalla y las dificultades de interacción que podrían surgir en dispositivos móviles. La visualización de todas las piezas y el tablero de manera adecuada en una pantalla pequeña, así como la precisión requerida para arrastrar las piezas con el dedo, podrían resultar complicadas para los usuarios y entorpecer la experiencia de los usuarios.

8.2. Trabajo futuro

Durante el transcurso de este proyecto hemos podido encontrar algunas tareas adicionales que podrían mejorar la aplicación, pero debido a la limitación de tiempo en los sprint, se ha decidido dejar como tareas a realizar en un futuro. Las tareas en las que se han pensado son las siguientes:

- Poder subir una foto de un nivel en concreto de la mesa del relojero, y que la aplicación reconociera los datos en la foto y guardar el nivel en la base de datos.
- Poder crear un nivel a partir de una frase dada por el usuario, donde la aplicación pueda crear todas las piezas y el tablero con las letras de ayuda necesarias para poder resolverlo.
- Cambiar la plataforma en la que esta alojada la web, ya que en ocasiones, al actualizar la página desde un ruta que no es la inicial, puede fallar.
- Poder cambiar la forma del tablero y disponer de nuevos niveles donde la forma del tablero no sea rectangular.

Apéndice A

Manual de usuario

En esta sección vamos a explicar las interacciones que puede realizar un usuario con la aplicación. Este proyecto se ha desarrollado con el objetivo de crear una página web intuitiva y facil de usar, para que al usuario le cueste poco esfuerzo aprender a utilizar la página web.

En esta primera vista A.1 podemos ver la página inicial una vez abierta la aplicación. A simple vista podemos ver que hay 3 botones, los cual accederan a distintas vista. El primer botón accederá a la pantalla de selección de niveles para poder jugar, en el segundo botón accederá a la pantalla de selección de niveles para ver la clasificación de cada nivel, y el tercer botón accederá a una pantalla donde se explicará como se jugar los niveles para completarlos

Figura A.1: Pantalla inicial

Si hemos seleccionado el botón de “Como Jugar” en la pantalla de inicio, aparecerá la

pantalla que veremos en la figura A.2 que consta unicamente de una imagen que explica brevemente como jugar los niveles para poder completarlos. En esta vista el usuario no puede interactuar, lo unico que puede hacer es retroceder para volver a la pantalla de inicio.

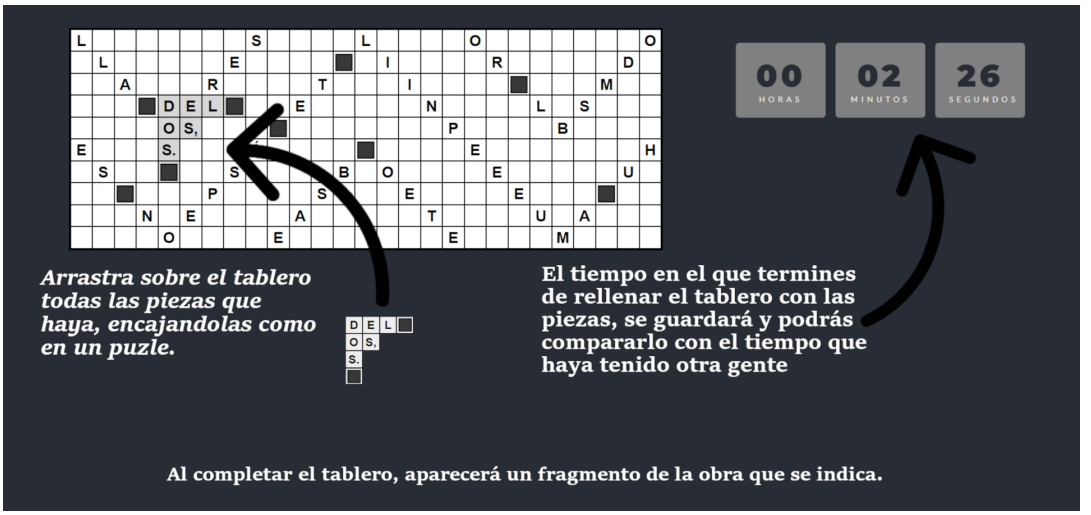

Figura A.2: Pantalla de cómo jugar

Si hemos seleccionado el botón de "Ranking" en la pantalla de inicio, se mostrará la vista que veremos en la figura A.3 donde se podrá visualizar todos los niveles disponibles en la aplicación que dispongan de al menos una clasificación en cada uno de los niveles.

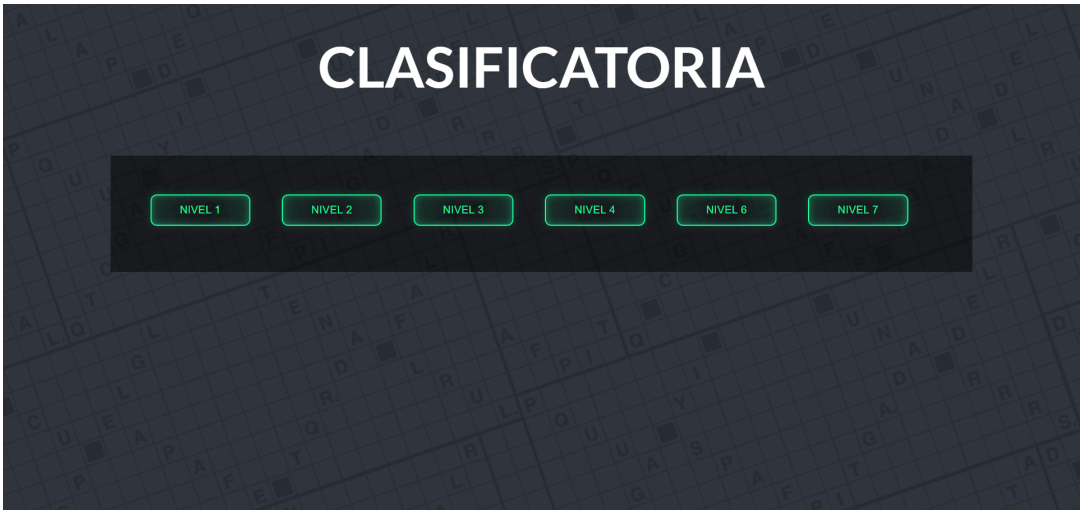

Figura A.3: Pantalla previa a la clasificación del nivel

Una vez seleccionado un nivel en la vista anterior, se presentará una vista como podemos ver en la figura A.4. En esta vista, se podrá visualizar todos los resultados de los usuarios que en algún momento han logrado resolver el nivel seleccionado. Estos resultados estan ordenados en formato de Ranking, indicando la posición que ocupa dependiendo del tiempo en que tardo en resolver el nivel. En caso de empate, como se da el caso en este nivel, tendrán la misma posición en el ranking, pero el que lo haya resuelto primero, será considerado el líder del empate y ocupará la posición más alta dentro de ese grupo de usuarios.

Figura A.4: Pantalla de clasificación del nivel

Si hemos seleccionado el botón de “Jugar” en la pantalla de inicio, aparecerá la pantalla que veremos en la figura A.5, en la cual podemos visualizar todos los niveles disponibles para jugar.

Figura A.5: Pantalla de selección de nivel a jugar

Una vez seleccionado el nivel a jugar, se mostrará una pantalla como en la fígura A.6. En esta vista, el usuario deberá arrastrar todas las piezas al tablero para poder completar la frase de una obra literaria y así completar el juego. Si el usuario lo requiere, tambien dispone de un botón para dar una pista al usuario para facilitar la resolución del nivel.

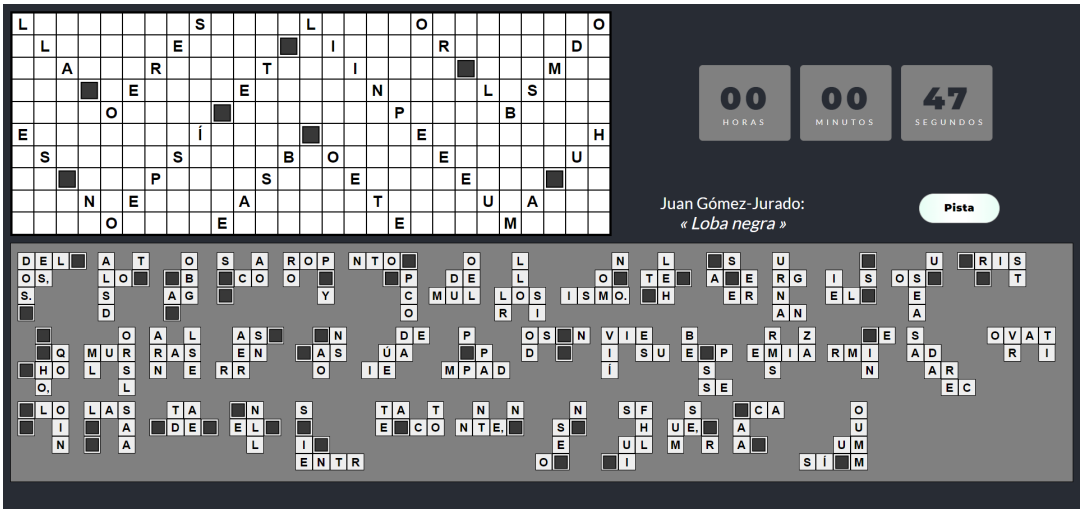

Figura A.6: Pantalla para jugar el nivel

Si durante el juego se presiona el botón para que la aplicación de al usuario una pista, se verá reflejado en el tablero la sombra de una de las piezas que aún esta disponible para arrastrar al tablero, como podemos observar en la figura A.7. Como “coste” por utilizar una

pista, se agregará al contar 20 segundos.

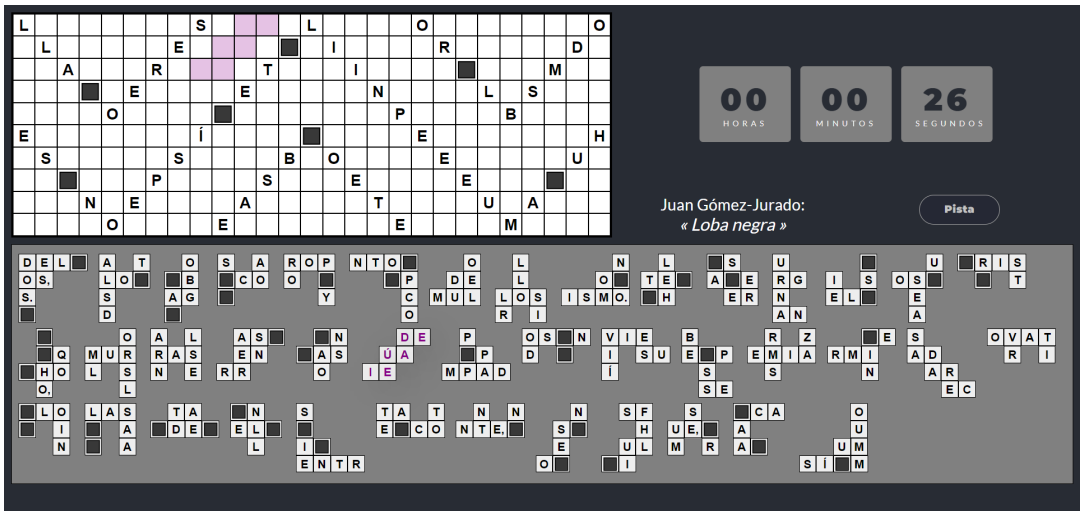

Figura A.7: Pantalla del juego con pista

Dependiendo de la dificultad que haya en cada nivel habrá limite de tiempo o no. Si el nivel seleccionado tiene limite de tiempo, y el usuario ha consumido todo el tiempo que se permitía en el nivel, la aplicación avisará al usuario de que no dispone de más tiempo y le devolverá a la pantalla de inicio. En la figura A.8, podemos ver el Pop-Up que se muestra cuando se queda sin tiempo el nivel.

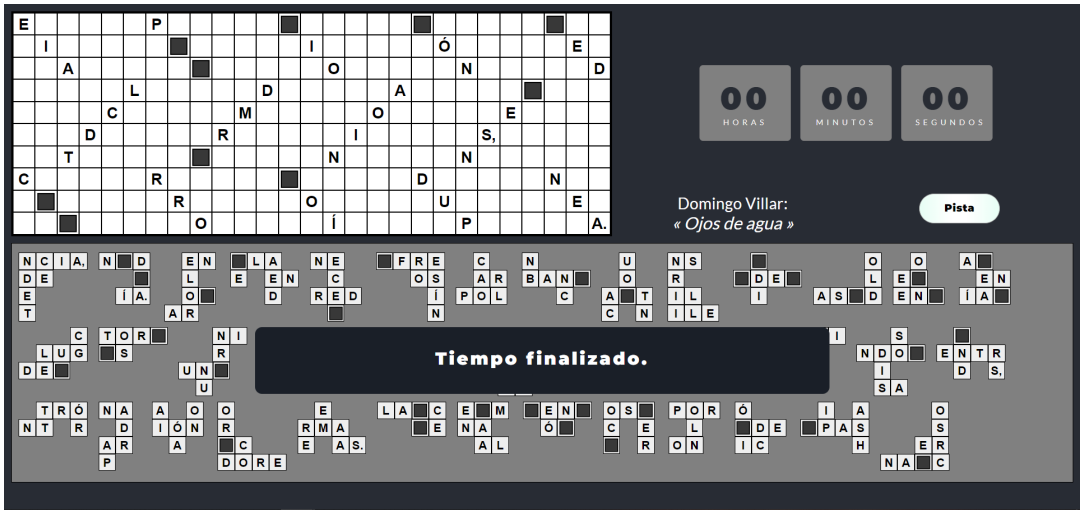

Figura A.8: Pantalla de juego sin tiempo

Si hemos terminado de rellenar el tablero con las piezas disponibles, y por ende, completado el nivel, se mostrara un Pop-Up como se muestra en la figura A.9. En esta nueva ventana, podemos ver las cinco mejores clasificaciones de este nivel, y podemos guardar nuestra marca de tiempo en la clasificación.

Figura A.9: Pantalla de juego completado

Si al guardar los datos utilizamos un nombre que no cumple con los requisitos necesarios por la aplicación, se mostrará una pequeña ventana de información que nos indicará los criterios de como guardar el nombre, como podemos observar en la figura A.10.

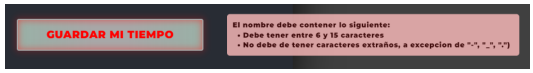

Figura A.10: Aviso de que no se cumple los criterios.

Si al guardar los datos utilizamos un nombre que ya esta en la clasificación, se mostrará un pequeña ventana avisando al usuario de ello, lo cual podemos ver reflejado en la figura A.11.

Figura A.11: Aviso de nombre en uso

Si guardamos el nombre y la marca de tiempo de forma correcta, se mostrará un Pop-Up que durará solamente 1,2 segundos para retroalimentar al usuario e indicarle que los datos

se han guardado de manera correcta en la base de datos. Dicha ventana la podemos observar en la figura A.12. Una vez transcurrido ese tiempo, se devolverá a la pantalla inicial.

Figura A.12: Mensaje de juego completado

Apéndice B

Manual de despliegue

B.1. Herramientas necesarias

Las herramientas utilizadas para crear nuestro entorno de trabajo y así poder desplegar la aplicación son las siguientes:

- Firebase CLI versión 9.6.11
- NPM versión 6.14.8
- Node.js versión 14.15.1

B.2. Preparación del entorno y despliegue

En primer lugar es necesario tener el código de nuestra aplicación en un repositorio online, el cuál se localiza en `https://github.com/angpere/MesaRelojero`. Hay dos formas de desplegar la aplicación: en local o en red.

Hay que tener en cuenta que no hemos subido al repositorio de Git las variables de entorno que permiten acceso a nuestra base de datos, ya que sino cualquiera podría modificarla. Para ello, hemos subido un fichero llamado *.env.temporal*, al cual habría que cambiarle de nombre a *.env.local* y rellenar las variables que contiene con las claves de la base de datos que se quiera utilizar.

B.2.1. Despliegue en local

Si deseamos visualizar la página web de forma local, lo primero que debemos hacer es instalar las dependencias, ya que no se subieron al repositorio Git debido al extenso tamaño de las librerias, y para ello ejecutamos el comando **npm install**.

Una vez instaladas las dependencias, ponemos el comando **npm start**, lo que nos permitirá tener nuestra aplicación desplegada en un servidor web local en el puerto 3000.

B.2.2. Despliegue en red

En nuestro caso, hemos utilizado Netlify para desplegar la aplicación a la red desde un repositorio de Github, ya que esta plataforma web es facil de utilizar y no requiere de subir ningun archivo a la plataforma de despligue [30].

Una vez registrados en la página web de Netlify, vamos a la pestaña "Sites" y seleccionamos "Añadir un proyecto existente". A continuación, sincronizamos nuestra cuenta de Github con Netlify y seleccionamos el proyecto que queremos desplegar. Luego escribimos el comando de compilación de nuestro proyecto, que en nuestro caso es **npm run build**, y finalmente escribimos nuestras variables de entorno con sus claves correspondientes para tener acceso a la base de datos.

Una vez hecho esto, ya tendríamos la aplicación desplegada en la red. En nuestro caso la tenemos desplegada con el dominio `https://mesa-del-relojero.netlify.app/`.

Bibliografía

[1] Adrian Alonso, *Atomic Web Design o Diseño Guíado por Componentes,* Ultima consulta el 07/07/23. URL: https://adrianalonsodev.medium.com/atomic-web-design-o-dise%C3%B1o-gu%C3%ADado-por-componentes-fa2fa152807f.

[2] Adrian Neja, *Qué es Overleaf,* Ultima consulta el 07/07/23. URL: https://www.adrinerja.com/que-es-overleaf/.

[3] Agencia Tributaria, *Coeficientes de amortización lineal,* Ultima consulta el 07/07/23. URL: https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/irpf-2020/capitulo-7-rendimientos-actividades-economicas-directa/fase-1-determinacion-rendimiento-neto/amortizaciones-dotaciones-ejercicio-fiscalmente-deducibles/requisitos-generales/coeficientes-amortizacion-lineal.html.

[4] Atlassian, *¿Qué es SCRUM?,* Ultima consulta el 07/07/23. URL: https://www.atlassian.com/es/agile/scrum.

[5] Atlassian, *¿Qué son los Sprints?,* Ultima consulta el 07/07/23. URL: https://www.atlassian.com/es/agile/scrum/sprints#:~:text=son%20los%20sprints%3F-,Un%20sprint%20es%20un%20per%C3%ADodo%20breve%20de%20tiempo%20fijo%20en,con%20menos%20quebraderos%20de%20cabeza..

[6] Atlassian, *Qué es Git,* Ultima consulta el 07/07/23. URL: https://www.atlassian.com/es/git/tutorials/what-is-git.

[7] COMPONENTSOURCE, *Acerca de Astah UML,* Ultima consulta el 07/07/23. URL: https://www.componentsource.com/es/product/astah-uml/about.

[8] IBM, *Modelos de dominio,* Ultima consulta el 07/07/23. URL: https://www.ibm.com/docs/es/ida/9.1.2?topic=types-domain-models.

[9] INDEED, *¿Cuánto se gana como uno Programador/a junior en España?,* Ultima consulta el 03/07/23. URL: https://es.indeed.com/career/programador-junior/salaries.

[10] Jessica Clark, *Plataformas de backend para su aplicación de React Native,* Ultima consulta el 07/07/23. URL: https://blog.back4app.com/es/las-mejores-plataformas-de-backend-para-su-aplicacion-de-react-native/.

[11] Kushal Agrawal, *Apply MVVM in React Native App,* Ultima consulta el 07/07/23. URL: https://tech.groww.in/apply-mvvm-in-react-native-app-ad77fa0f851b.

[12] LUCIDCHART, *Tutorial de diagrama de secuencia UML,* Ultima consulta el 07/07/23. URL: https://www.lucidchart.com/pages/es/diagrama-de-secuencia.

[13] Matias Hernandez , *¿Qué es NPM?,* Ultima consulta el 07/07/23. URL: https://www.freecodecamp.org/espanol/news/que-es-npm/.

[14] NETLIFY, *Qué es Netlify Connect,* Ultima consulta el 07/07/23. URL: https://www.netlify.com/products/connect/.

[15] Oscar Blancarte, *Ciclo de vida de los componentes,* Ultima consulta el 07/07/23. URL: https://reactiveprogramming.io/blog/es/react/ciclo-de-vida-de-los-componentes.

[16] Oscar Blancarte, *Que es el BackEnd as a Service (BaaS),* Ultima consulta el 07/07/23. URL: https://www.oscarblancarteblog.com/2018/06/18/backend-as-service-baas/.

[17] Página oficial de Bootstrap , *React-Bootstrap,* Ultima consulta el 07/07/23. URL: https://react-bootstrap.netlify.app/.

[18] Página oficial de Creately, *Guía Fácil de los Diagramas de Despliegue UML,* Ultima consulta el 07/07/23. URL: https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/.

[19] Página oficial de Latex, *LaTeX – A document preparation system,* Ultima consulta el 07/07/23. URL: https://www.latex-project.org/.

[20] Página oficial de Next.js, *From JavaScript to React,* Ultima consulta el 07/07/23. URL: https://nextjs.org/learn/foundations/from-javascript-to-react.

[21] Página oficial de React.js, *Inicio rápido con React,* Ultima consulta el 07/07/23. URL: https://es.react.dev/learn.

[22] Página oficial de React.js, *Un vistazo a los Hooks,* Ultima consulta el 07/07/23. URL: https://es.legacy.reactjs.org/docs/hooks-overview.html#:~:text=%C2%BFPero%20qu%C3%A9%20es%20un%20Hook,permiten%20usar%20React%20sin%20clases..

[23] Página oficial de ScrumManager, *Eventos de Scrum: el sprint y las reuniones,* Ultima consulta el 07/07/23. URL: https://www.scrummanager.com/blog/2023/02/cuales-son-los-eventos-reuniones-scrum/.

[24] PMOINFORMATICA, *¿Qué es un requerimiento funcional?,* Ultima consulta el 07/07/23. URL: http://www.pmoinformatica.com/2018/05/que-es-requerimiento-funcional.html.

[25] SALESFORCE, *¿Qué es SaaS?,* Ultima consulta el 07/07/23. URL: https://www.salesforce.com/es/learning-centre/tech/saas/.

[26] Sara Lopez Mora, *Firebase: qué es, para qué sirve, funcionalidades y ventajas,* Ultima consulta el 07/07/23. URL: https://digital55.com/blog/que-es-firebase-funcionalidades-ventajas-conclusiones/.

[27] Universidad de Valladolid, *Guia docente del Trabajo Fin de Grado, Grado en Ingenieria Informatica, Mencion en Ingenieria de Software, Universidad de Valladolid,* Ultima consulta el 03/07/23. URL: https://www.inf.uva.es/wp-content/uploads/2016/06/G46976.pdf.

[28] VIEWNEXT, *Artefactos Scrum ¿Qué son y para qué sirven?,* Ultima consulta el 07/07/23. URL: `https://www.viewnext.com/artefactos-scrum/`.

[29] Wikipedia, *Document Object Model,* Ultima consulta el 07/07/23. URL: `https://es.wikipedia.org/wiki/Document_Object_Model`.

[30] Yoelvis Mulen, *Cómo desplegar APP de React con GitHub y Netlify,* Ultima consulta el 03/07/23. URL: `https://www.youtube.com/watch?v=aCEn6_BHZyI`.

[31] Yubal Fernandez, *Qué es Github y qué es lo que le ofrece a los desarrolladores,* Ultima consulta el 07/07/23. URL: `https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores`.