UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

Aplicación web con sistema de recomendación de videojuegos integrado

Trabajo Fin de Grado

Grado en Tecnologías Interactivas

AUTOR/A: Torres Cruz, Claudia

Tutor/a: Palanca Cámara, Javier

CURSO ACADÉMICO: 2023/2024

Resumen

Los sistemas de recomendación son esenciales y cada vez están más presentes en las aplicaciones web que se visitan a diario. Ayudan al usuario a mejorar su experiencia haciendo que los resultados que se muestran coincidan con sus intereses. Por esta razón, nace esta web, para servir como herramienta a todas aquellas personas que disfrutan de los videojuegos y quieran descubrir algo diferente de forma personalizada. Además, los jugadores se enfrentan a un desafío a la hora de gestionar sus colecciones ya que existe una gran cantidad de plataformas con títulos exclusivos. Esta aplicación web pretende ofrecer una solución para resolver esta cuestión. El objetivo es mejorar la experiencia de los "gamers" sumergiéndolos en un universo de posibilidades a medida que descubren títulos nuevos, optimizando no sólo la interacción con los videojuegos, sino también facilitando su organización de forma cómoda. Para lograr su implementación, se ha decidido utilizar una arquitectura Full-Stack que comprende el uso de MySQL, Express, React y Node. Esta arquitectura pretende dar un marco sólido en el que desarrollar aplicaciones dinámicas, interactivas y avanzadas, destacando por su capacidad de escalabilidad.

Palabras clave: videojuegos, recomendadores, web, MySQL, react, nodeJS.

Resum

Els sistemes de recomanació són essencials i cada vegada estan més presents en les aplicacions web que es visiten diàriament. Ajuden l'usuari a millorar la seua experiència fent que els resultats que es mostren coincidisquen amb els seus interessos. Per aquesta raó, naix aquesta web, per a servir com a eina a totes aquelles persones que gaudeixen dels videojocs i vulguen descobrir alguna cosa diferent de manera personalitzada. A més, els jugadors s'enfronten a un desafiament a l'hora de gestionar les seues col·leccions ja que hi ha una gran quantitat de plataformas amb títols exclusius. Aquesta aplicació web pretén oferir una solució per a resoldre aquesta qüestió. L'objectiu és millorar l'experiència dels "gamers" submergintlos en un univers de possibilitats a mesura que descobreixen nous títols, optimitzant no només la interacció amb els videojocs, sinó també facilitant la seua organització de manera còmoda. Per a aconseguir la seua implementació, s'ha decidit utilitzar una arquitectura Full-Stack que comprén l'ús de MySQL, Express, React i Node. Aquesta arquitectura pretén donar un marc sòlid en què desenvolupar aplicacions dinàmiques, interactives i avançades, destacant per la seua capacitat d'escalabilitat.

Paraules clau: videojocs, recomanadors, web, MySQL, react, nodeJS.

Abstract

Recommendation systems are essential and are increasingly present in the web applications visited daily. They help users enhance their experience by matching displayed results with their interests. For this reason, this website was created to serve as a tool for all those who enjoy video games and want to discover something different in a personalized way. Additionally, gamers face a challenge in managing their collections due to the numerous platforms with exclusive titles. This web application aims to offer a solution to this issue. The goal is to improve the experience of gamers by immersing them in a universe of possibilities as they discover new titles, optimizing not only their interaction with video games but also making their organization more convenient. To achieve its implementation, a Full-Stack architecture comprising MySQL, Express, React, and Node has been chosen. This architecture aims to provide a solid framework for developing dynamic, interactive, and advanced applications, standing out for its scalability.

Keywords: video games, recommenders, web, MySQL, React, Node.js

Índice

Índice de tablas

Índice de figuras

Capítulo 1. Introducción

1.1 Presentación

La industria de los videojuegos ha crecido exponencialmente en los últimos años, convirtiéndose en una de las principales formas de entretenimiento a nivel global. Dicho crecimiento ha dado lugar a que cada vez haya más empresas y desarrolladores en solitario que deciden dar el paso en este mundo, generando más títulos, lo que plantea un desafío para los jugadores que les gusta mantener su colección organizada. El incremento de plataformas digitales y la persistencia del formato físico han hecho que muchos usuarios busquen herramientas que les ayuden a mantener un control de sus colecciones.

1.2 Motivación

Este proyecto surge de la posibilidad de mezclar uno de mis mayores hobbies con todo lo que he aprendido a lo largo de estos años y de la necesidad personal de descubrir nuevos títulos basados en todos los que he jugado con anterioridad. Además de la motivación personal, este proyecto representa una oportunidad de profundizar en el desarrollo web y expandir mis habilidades con tecnologías que ya conocía como NodeJS y MySQL y aprender a utilizar React, un framework muy conocido. La integración de algoritmos de recomendación también me brinda la posibilidad de explorar áreas más avanzadas de la programación y análisis de datos.

Otra motivación significativa ha sido el deseo de contribuir a la comunidad "gamer". Existe una clara necesidad de que los jugadores cuenten con una plataforma que les ayude a mantener organizadas sus colecciones y a encontrar juegos que se ajusten a sus gustos y preferencias. Este proyecto busca ofrecer una solución que no solo mejore la experiencia de usuario, sino también la experiencia de juego de los jugadores.

1.3 Objetivos

El **objetivo general** de este proyecto es desarrollar una aplicación web que permita a los usuarios gestionar sus colecciones de forma simple y personalizada, integrando algoritmos de recomendación que mejoren la experiencia de descubrimiento de nuevo títulos. Los objetivos específicos son:

- Objetivo 1: Investigar las aplicaciones existentes y soluciones actuales para identificar las mejores prácticas y posibles mejoras.
- Objetivo 2: Desarrollar funcionalidades que permitan a los usuarios acceder y gestionar la información relacionada con sus colecciones de videojuegos (añadir a una lista, eliminarlo, puntuarlo…)
- Objetivo 3: Diseñar e integrar sistemas de recomendación que utilice técnicas de filtrado colaborativo, basado en contenido e híbrido para sugerir juegos desconocidos por el usuario.
- Objetivo 4: Desarrollar una aplicación web que sea compatible con dispositivos móviles y de escritorio, asegurando una experiencia consistente.
- Objetivo 5: Proporcionar un método alternativo al habitual de usuario y contraseña, para recoger los juegos que el usuario ya ha jugado y mejorar su experiencia desde el primer momento que ingresa en la web.

- Objetivo 6: Realizar pruebas exhaustivas de la aplicación para asegurar su usabilidad, recogiendo retroalimentación de los usuarios para mejorar continuamente.
- Objetivo 7: Documentar el proceso de desarrollo y proporcionar una guía para el uso de la aplicación

Por último, cabe resaltar la importancia de los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030 de las Naciones Unidas, en particular, la aplicación que se desarrolla contribuye al objetivo ODS 9: Industria, Innovación e Infraestructura, ya que el proyecto incorpora tecnologías avanzadas y promueve la innovación a través del desarrollo de una web moderna y eficiente, apoyando a la infraestructura digital y tecnológica.

1.4 Metodología

Se ha utilizado la metodología ágil SCRUM, que divide el trabajo en ciclos cortos denominados sprints, durante los que se planifican, desarrollan y verifican tareas específicas. Aunque se suele aplicar para trabajos en equipo, se puede adaptar a proyectos individuales como este, estableciendo objetivos claros y facilitando la capacidad de adaptar las nuevas funcionalidades a cambios que surjan durante las reuniones.

La duración de los sprints ha sido entre 2-4 semanas, durante cada fase se han ido realizando funcionalidades para, finalmente, cuando el periodo de tiempo termina, enseñar los avances al “producto owner”, que en este caso ha sido el tutor, y recoger sus comentarios y sugerencias, para aplicarlos en el siguiente sprint.

1.5 Estructura de la memoria

La memoria está organizada en capítulos en los que se detallan el estado del arte, el análisis de requisitos, el diseño de la web y su implementación, los manuales de uso, la evaluación de la aplicación web y las conclusiones.

- Capítulo 2: Estado del arte. Este capítulo describe las tecnologías disponibles en el mercado para el desarrollo web y un análisis de la competencia, identificando problemas y comparando las aplicaciones más conocidas para la gestión de colecciones de videojuegos.
- Capítulo 3: Análisis de requisitos. Se definen las características, los requisitos y los casos de uso necesarios para el desarrollo de la aplicación.
- Capítulo 4: Diseño de la aplicación. Se centra en el diseño de la aplicación, incluyendo la lógica de negocio, ayudándose de diagramas.
- Capítulo 5: Implementación. Se detalla el desarrollo y los problemas de implementación que se han resuelto.
- Capítulo 6: Evaluación de la Aplicación. Se describe la forma en la que se ha evaluado el sistema, detallando métodos utilizados para conocer la satisfacción de los usuarios.
- Capítulo 7: Manuales de uso. Este capítulo incluye guías de uso y se proporcionan instrucciones para la instalación de la aplicación.
- Capítulo 8: Conclusiones y Mejoras a futuro. Este capítulo resume los hitos logrados durante el desarrollo de la aplicación. Además, se proporcionan funcionalidades para mejorar en el futuro.

Capítulo 2. Estado del arte:

En este apartado, para entender mejor el propósito de este proyecto se estudiará la importancia que tienen este tipo de aplicaciones en la sociedad, así como su evolución en los últimos años, además de investigar algunas opciones que ya se encuentran en el mercado.

Del mismo modo, se indagará en las arquitecturas full-stack y algoritmos de recomendación más utilizados, junto con sus ventajas e inconvenientes, para así determinar la que mejor se alinea con las características de este trabajo.

2.1 Aplicaciones de organización y gestión

Vivimos en una realidad repleta de servicios que tratan de ayudar a los usuarios a ordenar y organizar tanto videojuegos, como películas, series, libros, discos…etc. Esto se debe a que cada vez existen más plataformas digitales y, aunque poco a poco va predominando el formato digital, el formato físico sigue siendo una prioridad para muchos, lo que acaba desembocando en tener colecciones en casa de la que terminamos sin saber cuáles de ellos aún no hemos utilizado. Aquí este tipo de aplicaciones tienen un papel crucial para que nunca vuelva a ser un problema seguir aumentando la colección.

Las características que comparten la mayoría son una cuenta personal en la que se van guardando todos los datos y la capacidad de crear listas para añadir los títulos y la posibilidad de puntuarlos. En resumen, proporcionan una interfaz centralizada para organizar y gestionar el contenido.

La implementación de la Inteligencia Artificial en este tipo de aplicaciones ha provocado una mejora ya que algunas son capaces de ofrecer recomendaciones personalizadas. Analizan el comportamiento del usuario a través de estadísticas y de este modo consiguen predecir contenido que le pueda resultar interesante, mejorando la experiencia de descubrimiento.

Existe una clara tendencia a que este tipo de aplicaciones estén disponibles tanto en web como en aplicaciones móviles, permitiendo al usuario acceder desde diferentes dispositivos para gestionar el contenido tanto desde la comodidad de su hogar como desde fuera, adaptándose a un estilo de vida más dinámico.

2.2 Estudio de mercado

Este proyecto se centra en el desarrollo de una aplicación web para la gestión de colecciones digitales de videojuegos. Como primer paso, se ha realizado un estudio de mercado de diversas aplicaciones que incorporan y proporcionan características y funcionalidades similares a las que se describen en este proyecto. A continuación, se presenta una revisión crítica de las más destacables.

2.2.1 Stash: Video Game Manager

Stash es la aplicación líder en el mercado de la organización y la gestión de videojuegos, convirtiéndose además en una red social en la que poder seguir a otros jugadores y dejar reseñas. Una de sus actualizaciones más recientes añade la posibilidad de conectarse con tu cuenta de distintas plataformas, entre ellas Steam, pudiendo importar los títulos de su biblioteca a la aplicación. Es gratuita, aunque posee una versión de pago para ofrecer estadísticas personales acerca de la experiencia de juego, eliminar anuncios…etc. Algo que distingue esta aplicación del resto es que tiene una sección de juegos que pueden ser de interés para el usuario.

La parte negativa es que sólo está disponible para móvil y el buscador muestra los títulos de forma horizontal dejando mucho espacio sin utilizar, haciendo que sea incómodo realizar búsquedas.

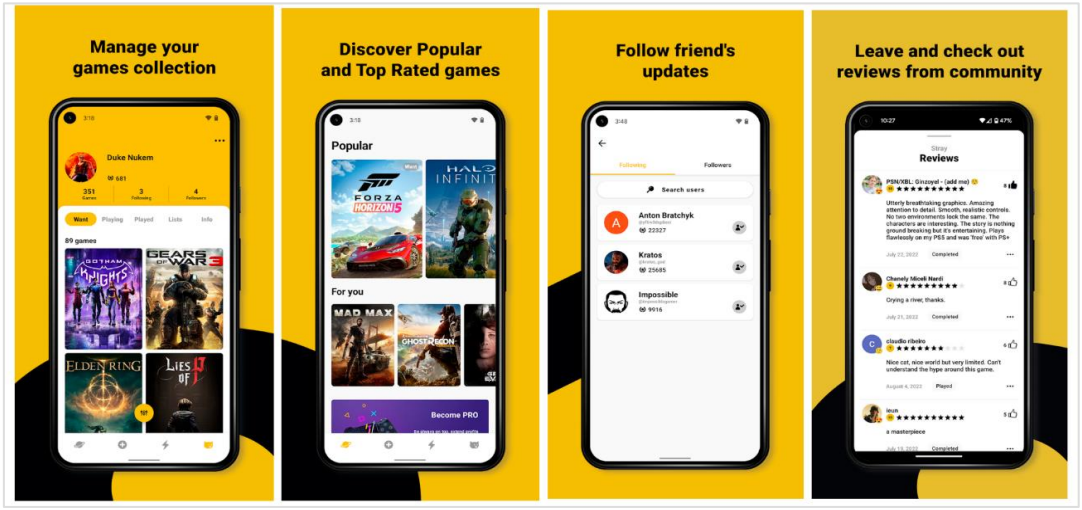

Figura 1. Imágenes Play Store Slash

2.2.2 GG App: Video Game Collection Tracker

Esta web está desarrollada por una única persona, tiene funcionalidades centradas principalmente en la gestión de videojuegos, teniendo la posibilidad de crear listas y añadir etiquetas a los juegos por ejemplo “abandonado”, “completado” y “jugando”. Además, cuenta con una parte social en su pantalla de inicio donde aparecen reseñas y listas de otros usuarios que forman parte de la aplicación.

Una desventaja destacable es que se necesita de una membresía para tener un mejor control de las listas, por ejemplo, para fusionar listas o clonar las de la comunidad. Al igual que la barra de búsqueda no cuenta con ningún tipo de filtrado por plataforma por lo que tampoco se puede especificar en cuál se tiene el juego.

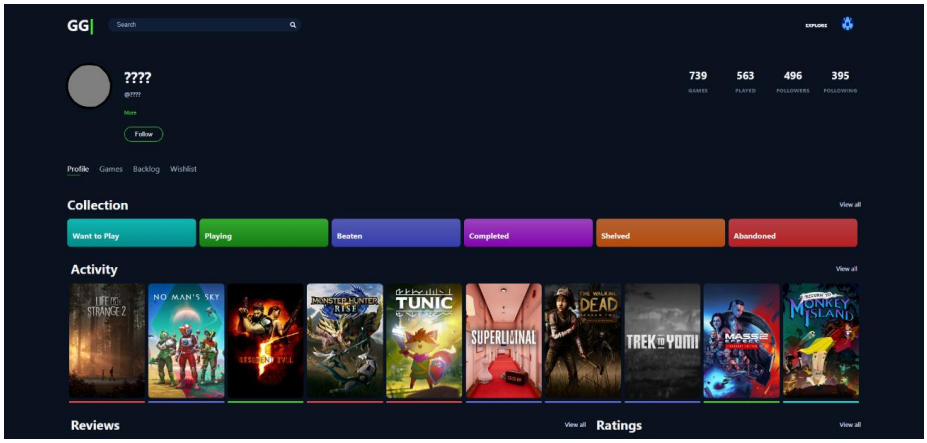

Figura 2. Captura Perfil GG|

2.2.3 HowLongToBeat

Inicialmente esta aplicación fue diseñada para saber de forma aproximada la duración de los videojuegos, sin embargo, también funciona como un rastreador, permitiendo

registrarlos y añadirlos tanto a listas personalizadas como a listas por defecto de la aplicación (“rejugados”, “completados” y “abandonados”). Además, tiene funcionalidades interesantes como añadir si se ha abordado solo la historia principal o también contenido extra. Toda esta información se puede descargar y mostrar estadísticas interesantes sobre el tiempo promedio que tarda en completar los títulos y sus tendencias acerca de los géneros que predominan en su biblioteca.

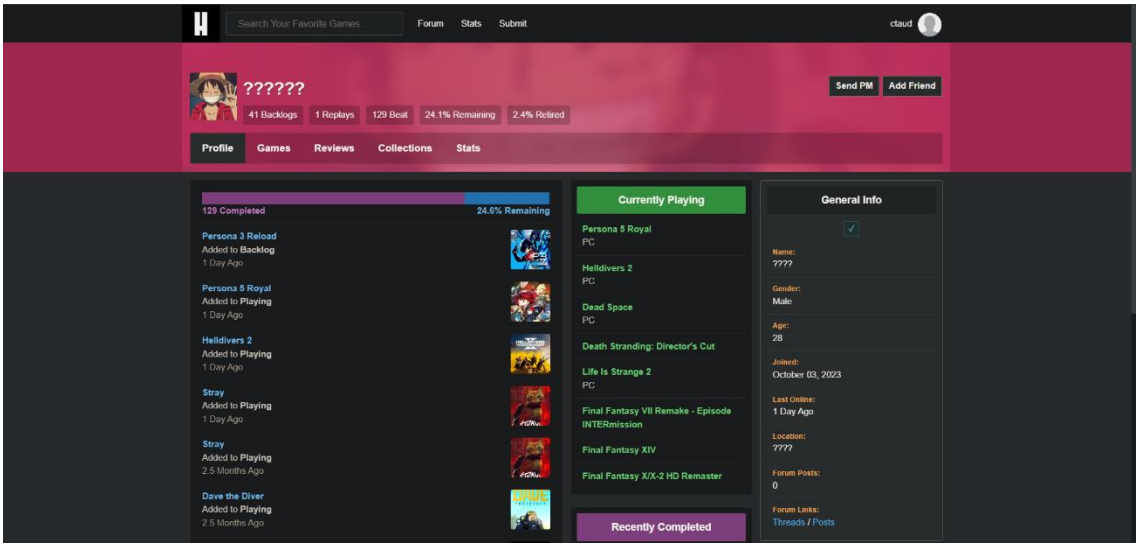

Figura 3. Captura Perfil HowLongToBeat

2.2.4 Resultados

Ahora que ya se tiene una idea general de las aplicaciones existentes en el mercado, procederemos a realizar una comparación visual, que abarque también la aplicación que se contempla en este trabajo “GamerWise”.

Característica	Stash	GG	HowLongToBeat	GamerWise
Capacidad de crear listas personalizadas	X	X	X	X
Posibilidad de conectarse usando Steam u otra aplicación existente	X			X
Buscador con opciones de filtrado	X		X	X
Algoritmo de recomendación	X			X
Valoración de los videojuegos a través de puntuaciones	X	X	X	X
Disponible para PC		X	X	X
Aprovecha el espacio de la pantalla		X	X	X

Tabla 1. Comparativa de aplicaciones

2.3 Herramientas de desarrollo Web

Una arquitectura full-stack es aquella que comprende tanto las tecnologías usadas en el lado del servidor como en el lado del cliente, permitiendo que ambas se integren.

En este apartado, se estudiarán las herramientas disponibles para el desarrollo de una aplicación Web con una arquitectura de este tipo, el objetivo final será determinar aquellas que encajan mejor con las características de este proyecto.

2.3.1 Herramientas front-end

Aquí se encuentran las bibliotecas y frameworks que se han evaluado, que corresponden a los más demandados según un estudio realizado desde el 1 de noviembre de 2022 hasta el 31 de diciembre de 2023 realizada por devjobsscanner.

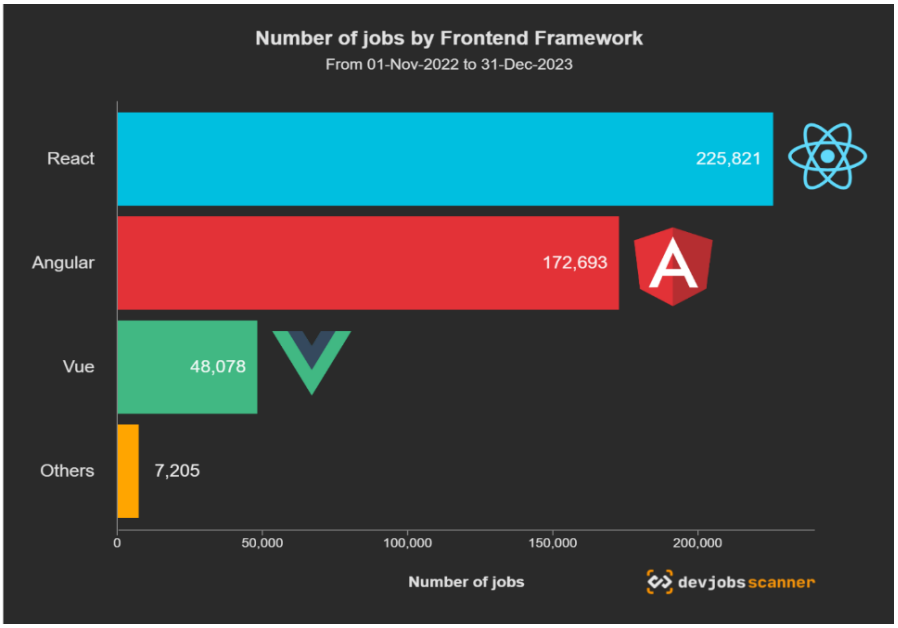

Figura 4. Gráfica perteneciente a DevJobsScanner, 2023

2.3.1.1 ReactJS

React es una biblioteca JavaScript que facilita la creación de interfaces de usuario, lo consigue a través del uso de componentes que se pueden usar en varias partes del sitio web sin necesidad de escribir de nuevo el código, ahorrando así mucho tiempo a los programadores. Otra característica esencial es que la sintaxis que utiliza es una combinación entre HTML y JavaSctipt, haciendo que adaptarse al lenguaje no signifique un problema. Además, su lógica es clara y fácil de entender.

Otra ventaja que presenta es el uso de DOM virtual. En lugar de cargar toda la página cuando hay cambios en el estado de la aplicación, React compara el nuevo árbol de elemento (llamado Fiber Tree, resultante de un re-rende) con el anterior (denominado render inicial) haciendo así que solo se renderice aquel componente que presente cambios. Esto lo hace a través de React Fiber, que es el motor que utiliza ReactJS para comprobar qué componentes han cambiado en su estructura interna. De esta manera, solo renderiza el componente que presenta cambios optimizando el rendimiento al reducir la necesidad de recargar la página completa en cada actualización.

2.3.1.2 Angular

Angular es un framework escrito en TypeScript, lo que facilita la detección de errores durante el desarrollo al tratarse un lenguaje tipado. Al igual que React, está basado en componentes y se puede desarrollar tanto webs como aplicaciones móviles. Es una plataforma que tiene un nivel de complejidad que puede resultar difícil para desarrolladores principiantes, sin embargo, esto se traduce en una estructura que beneficia sobre todo a aplicaciones grandes y complejas. Entre sus principales ventajas, la más destacable es la capacidad para sincronizar automáticamente los datos entre el modelo y la vista, y viceversa, permitiendo una mejor gestión del estado de la aplicación. También, permite un desarrollo de aplicaciones web más ligeras, ya que permite hacer paquetes más pequeños que ayudan a acelerar la aplicación.

Por otro lado, hace uso de DOM real, que relacionado con su complejidad hacen que, si se pierde el flujo, pueda ocasionar problemas de rendimiento. Es crucial implementar estrategias para minimizar el impacto que puede generar.

2.3.1.3 VueJS

Vue es un framework JavaScript progresivo, es decir es ideal para migrar y adaptar proyectos existentes que se han desarrollado con otras tecnologías. Posee algunas de las mejores características de otros framekworks, tales como el DOM virtual y el enlace de datos, mencionados anteriormente en React y Angular. Además, su curva de aprendizaje es suave, haciendo que crear aplicaciones se vuelva una tarea sencilla con esta tecnología.

2.3.2 Herramientas Back-end

2.3.2.1 NodeJS

Node es un entorno de ejecución de JavaScript multiplataforma, es asíncrono y su arquitectura está orientada a eventos. Cuando se usa una herramienta en el lado del cliente que también utiliza JavaScript, facilita el intercambio de información entre el frontend y el backend.

Es una de las herramientas más utilizadas en todo el mundo para el desarrollo web (StackOverflow, 2023). Además, contiene muchísimos frameworks que hacen que Node sea una tecnología muy completa.

2.3.2.2 Django

Es un framework de desarrollo web creado en Python, que destaca por su estructura diferenciando el proyecto en ‘apps’, que son funcionalidades independientes. De esta manera Django se convierte en una herramienta útil para desarrolladores principiantes que buscan una guía clara en la organización de sus proyectos, y al mismo tiempo, facilita el mantenimiento de grandes proyectos. También, tiene características integradas, la más llamativa es un panel de administrador desde el cual podemos controlar nuestra web, tanto de forma visual similar a WordPress, como mediante código.

Sin embargo, puede resultar complicado sacar el máximo provecho de esta tecnología al contar con un conjunto amplio de características.

2.3.2.3 Laravel

Laravel es el framework más famoso y utilizado de PHP. Su filosofía es desarrollar código de forma simple y elegante, evitando código mal estructurado y difícil de mantener. Es similar a Django, ya que ofrece funciones integradas muy similares, entre ellas ORM, autenticación y routing, al igual que ambos siguen una estructura MVC (Model View Controller).

Destaca con respecto a otras herramientas en su facilidad de uso, gracias a tener una sintaxis elegante. Es más apropiado para proyectos pequeños, aunque también se puede adaptar para aplicaciones grandes.

2.3.3 Selección de herramientas

Tras conocer las herramientas líderes en el ámbito del desarrollo web, estudiando no solo sus características si no también su popularidad, la elección para la construcción de la aplicación se centra en hacer uso de NodeJS, con su framework más popular Express para construir la API REST que se comunique con la base de datos relacional (MySQL). Para el desarrollo en el lado del cliente, se optará por React, debido a que es una librería muy completa que proporciona flexibilidad y rapidez y dada su popularidad y creciente uso es una buena oportunidad para aprender a utilizar esta valiosa herramienta.

2.4 Sistemas de recomendación

Los sistemas de recomendación van a ser un elemento clave, es por ello importante conocer bien tanto sus características como sus las posibilidades para elegir el o los métodos que mejor se adapten.

Definiremos un sistema de recomendación como una herramienta que, a través de un conjunto de valoraciones e interacciones previas del usuario con los datos, es capaz de realizar predicciones sobre recomendaciones de artículos que puedan ser de interés para el usuario.

2.4.1 Filtrado colaborativo

Un sistema basado en filtrado colaborativo se basa en utilizar los gustos y preferencias de un grupo de personas que son similares entre ellos, de esta forma construye una lista clasificada de sugerencias que mostrar al usuario. A modo de ejemplo, si tenemos un usuario 1 que le gustan los videojuegos de terror, aventura y acción, y un usuario 2 que le gustan los de terror y acción, este tipo de sistema recomendará videojuegos de aventura al usuario 1.

2.4.2 Filtrado basado en contenido

Este tipo de sistema utiliza el historial del usuario para realizar predicciones similares a las que ya ha visitado. Su desventaja principal es que, utilizando este método, los usuarios no podrán descubrir nada diferente a lo que suelen visitar y se podrán perder experiencias diferentes. Para entenderlo mejor, si tenemos un usuario que suele jugar juegos de terror, el sistema recomendará otro juego de terror.

2.4.3 Filtrado híbrido

Este método combina el filtrado colaborativo y basado en contenido, de este modo al aprovechar tanto los elementos que le suelen gustar al usuario como el comportamiento de otras personas que son afines, son capaces de ofrecer recomendaciones precisas. Por ejemplo, si un usuario tiene interés por juegos de terror y competitivos, el sistema puede recomendar uno que combine ambas.

Capítulo 3. Análisis de requisitos

El objetivo de este capítulo es determinar los requisitos necesarios para desarrollar una aplicación web que permita a los usuarios gestionar sus colecciones de videojuegos y recibir recomendaciones personalizadas. Para realizar este análisis, se llevaron a cabo dos actividades principales:

1. **Investigación de mercado**: Como se ha visto en el capítulo anterior, se analizaron las funcionalidades y limitaciones de aplicaciones similares.
2. **Entrevistas con potenciales usuarios**: Se recogió información sobre las necesidades de los jugadores y sus experiencias con las recomendaciones de nuevos juegos.

3.1 Personas UX

Para asegurar que la aplicación satisfaga las necesidades de los usuarios finales, se desarrollaron dos personas UX (Figuras 5 y 6) basadas en la información recogida en las entrevistas. Estas personas representan diferentes tipos de usuarios con sus respectivas características, necesidades y objetivos.

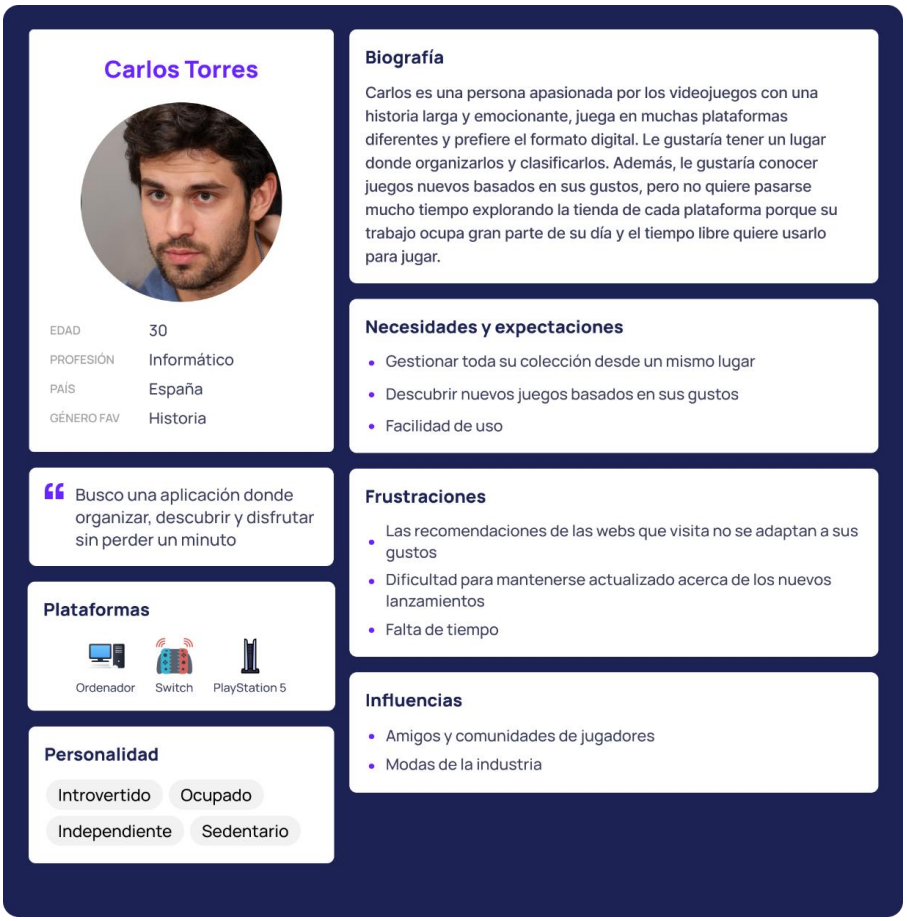

Figura 5. Persona UX (Carlos)

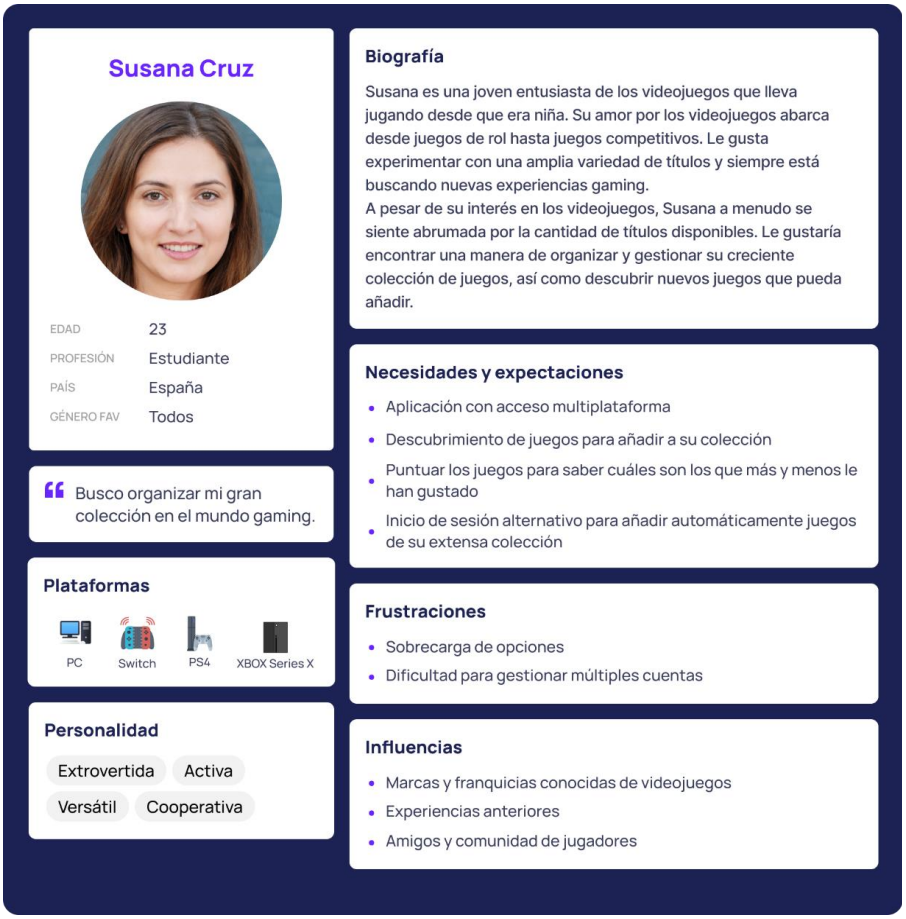

Figura 6. Persona UX (Susana)

3.2 Casos de uso

A continuación, se detallan los casos de uso extraídos del estudio realizado, que representan las funcionalidades de las que dispondrá la web:

Caso de uso	**Registro de usuario**
Descripción	Permite a los usuarios registrarse en la aplicación
Precondición	El usuario no debe estar registrado
Flujo principal	1. El usuario selecciona el botón de registro 2. El usuario introduce sus datos (nombre, correo y contraseña) 3. El sistema valida los datos y crea una nueva cuenta
Flujo alternativo	Si el usuario introduce un email existente o las contraseñas no coinciden, el sistema muestra un mensaje de error alertando al usuario.
Postcondición	El sistema redirige al usuario a la pantalla de iniciar sesión

Tabla 2. Caso de uso: Registro de usuario

Caso de uso	**Inicio de sesión**
Descripción	Permite a los usuarios acceder a la aplicación
Precondición	El usuario no debe estar autenticado anteriormente.
Flujo principal	1. El usuario selecciona el botón de iniciar sesión 2. El usuario introduce sus credenciales (correo y contraseña) o inicia sesión con Steam autorizando la aplicación a acceder a sus datos. 3. El sistema valida los datos
Flujo alternativo	Si el usuario introduce sus credenciales mal, el sistema muestra un mensaje de error alertando al usuario.
Postcondición	- El sistema guarda el token de autenticación y redirige al usuario a la página de usuarios registrados. - El usuario puede empezar a utilizar la aplicación

Tabla 3. Caso de uso: Inicio de sesión

Caso de uso	**Vincular cuenta de Steam**
Descripción	Permite a los usuarios vincular sus juegos de Steam con la aplicación.
Precondición	El usuario debe tener la sesión iniciada sin cuenta de Steam vinculada.
Flujo principal	1. El usuario accede a Ajustes y pulsa el enlace “Vincular cuenta de Steam”. 2. El usuario introduce sus datos de Steam y autoriza a la aplicación. 3. El sistema obtiene el token ID y los juegos.
Flujo alternativo	Si el usuario tiene vinculada su cuenta de Steam aparecerá la opción de desvincularla.
Postcondición	Se añaden los juegos a la lista “Liked”

Tabla 4. Caso de uso: Vincular cuenta de Steam

Caso de uso	**Añadir un juego a una lista**
Descripción	Permite a los usuarios añadir juegos a sus listas
Precondición	El usuario debe estar autenticado
Flujo principal	1. El usuario accede al juego que desea añadir 2. El usuario selecciona el botón "Añadir a una lista" y elige a la/s que quiera que pertenezca. 3. El sistema guarda la información y actualiza la/s lista/s del usuario
Flujo alternativo	Si el juego ya pertenece a una de las listas del usuario, no podrá seleccionarla.
Postcondición	El juego aparece en la/s lista/s seleccionada/s

Tabla 5. Caso de uso: Añadir un juego a una lista

Caso de uso	**Crear una lista**
Descripción	Permite a los usuarios crear listas personales
Precondición	El usuario debe estar autenticado
Flujo principal	1. El usuario accede a su perfil, aparado "Listas" o al panel que aparece al pulsar en el botón "Añadir a una lista". 2. El usuario selecciona el botón "+" o "Nueva Lista". 3. El sistema despliega un panel que solicita un nombre y una descripción opcional. 4. El usuario rellena los campos necesarios y selecciona el botón "Crear" 5. El sistema añade la lista
Flujo alternativo	Si no se proporciona nombre, el sistema alertará al usuario indicando que el campo es obligatorio.
Postcondición	El usuario puede interactuar con la lista creada

Tabla 6. Caso de uso: Crear una lista

Caso de uso	**Editar una lista**
Descripción	Permite a los usuarios actualizar las listas creadas por ellos.
Precondición	El usuario debe estar autenticado
Flujo principal	1. El usuario accede a su perfil, aparado “Listas” 2. El usuario pulsa la lista que quiere editar y selecciona el icono de un lápiz 3. El sistema despliega un panel. 4. El usuario hace las modificaciones que considere y pulsa el botón “Actualizar” 5. El sistema actualiza la información de la lista.
Flujo alternativo	Si se elimina el nombre, se alertará al usuario del error.
Postcondición	El usuario visualiza los cambios que ha realizado

Tabla 7. Caso de uso: Editar una lista.

Caso de uso	**Eliminar un juego de una lista**
Descripción	Permite a los usuarios eliminar un juego de una lista.
Precondición	El usuario debe estar autenticado
Flujo principal	1. El usuario accede a la lista que contiene el juego a eliminar. 2. El usuario pulsa el icono de una X. 3. El sistema actualiza los juegos contenidos en esa lista.
Postcondición	El juego se elimina de la lista

Tabla 8. Caso de uso: Eliminar un juego de una lista.

Caso de uso	**Eliminar una lista**
Descripción	Permite a los usuarios eliminar una lista.
Precondición	El usuario debe estar autenticado
Flujo principal	1. El usuario accede a la lista que desea eliminar y selecciona el icono de un contenedor de basura. 2. El sistema despliega un panel para asegurarse de que el usuario desea eliminar la lista 3. El usuario selecciona el botón “Eliminar”

	4. El sistema elimina la lista y la relación con todos los juegos que contenía desaparece.
Flujo alternativo	Si el usuario selecciona el botón “Cancelar” no se elimina la lista.
Postcondición	La lista entera desaparece.

Tabla 9. Caso de uso: Eliminar una lista.

Caso de uso	**Modificar la información del usuario**
Descripción	Permite a los usuarios modificar su nombre, email y contraseña.
Precondición	El usuario debe estar autenticado
Flujo principal	1. El usuario accede al apartado de “Opciones” y selecciona la opción que quiera: cambiar nombre, email o contraseña. 2. En caso de cambiar email, el sistema despliega un panel donde debe introducir el email 2 veces. 3. En caso de cambiar la contraseña, el sistema despliega un panel donde debe introducir la contraseña actual y la nueva 2 veces. 4. El usuario rellena los campos y hace click en el botón “Cambiar”. 5. El sistema actualiza la información del usuario.
Flujo alternativo	Si el formato de los campos introducidos es erróneo se alertará al usuario con mensajes de error.
Postcondición	El usuario cambia sus datos

Tabla 10. Caso de uso: Modificar la información del usuario.

Caso de uso	**Eliminar cuenta de usuario**
Descripción	Permite a los usuarios eliminar su cuenta
Precondición	El usuario debe estar autenticado
Flujo principal	1. El usuario accede al apartado de “Opciones” y selecciona la opción “Eliminar cuenta”. 2. El sistema despliega un panel para asegurarse de que el usuario desea borrar toda su información de la aplicación. 3. El usuario pulsa el botón “Eliminar”.

	4. El sistema elimina todos los registros del usuario y se redirige a la Landing Page.
Flujo alternativo	Si el usuario selecciona el botón "Cancelar" no se eliminará la cuenta.
Postcondición	El usuario elimina su cuenta y todos sus datos

Tabla 11. Caso de uso: Eliminar cuenta de usuario.

Caso de uso	**Buscar juegos**
Descripción	Permite a los usuarios buscar cualquier juego
Precondición	El usuario puede no estar autenticado
Flujo principal	1. El usuario desde la Landing Page, en el apartado "Nuevos Lanzamientos" o desde la página de usuarios registrados encontrarán un buscador. 2. El usuario introduce el nombre y/o alguna de las opciones de filtrado que se ofrece. 3. El sistema proporciona juegos que encajen con las características que el usuario ha proporcionado.
Flujo alternativo	Si no hay juegos que mostrar aparecerá un mensaje indicándolo.
Postcondición	El usuario visualiza los juegos que coinciden con su criterio de búsqueda y puede interactuar con ellos.

Tabla 12. Caso de uso: Buscar juegos

Caso de uso	**Puntuar juegos**
Descripción	Permite a los usuarios darles una puntuación del 1 al 5 a los juegos.
Precondición	El usuario debe estar autenticado
Flujo principal	1. En la pantalla de información de los juegos, aparecen unas estrellas que el usuario puede pulsar para proporcionar una puntuación. 2. El usuario selecciona la cantidad de estrellas que quiera. 3. El sistema actualiza la puntuación del juego para ese usuario.
Flujo alternativo	Si se vuelve a pulsar en la puntuación elegida se elimina y si se pulsa en otra se actualiza.

Postcondición	- El sistema podrá recomendar al usuario juegos con más exactitud. - El usuario tendrá una mejor experiencia al recibir mejores recomendaciones.

Tabla 13. Caso de uso: Puntuar juegos

Caso de uso	**Visualizar juegos de una lista**
Descripción	Permite a los usuarios ver los juegos que contiene una lista, así como sus puntuaciones.
Precondición	El usuario debe estar autenticado
Flujo principal	1. El usuario accede a una de sus listas. 2. El sistema carga los juegos que contiene, mostrando también la puntuación que le ha dado el usuario.
Flujo alternativo	Si la lista está vacía aparecerá un mensaje indicándolo.
Postcondición	El usuario visualiza los juegos y tiene mayor control sobre su colección.

Tabla 14. Caso de uso: Visualizar juegos de una lista.

Caso de uso	**Cerrar sesión**
Descripción	Permite al usuario cerrar sesión en su cuenta
Precondición	El usuario debe estar autenticado
Flujo principal	El usuario elige la opción cerrar sesión de la barra de navegación.
Postcondición	- El sistema elimina el token de sesión contenido en una cookie. - El usuario es redirigido a la Landing Page.

Tabla 15. Caso de uso: Cerrar sesión.

Caso de uso	**Obtener recomendaciones de juegos**
Descripción	El sistema genera y muestra recomendaciones de juegos que cree que le pueden gustar al usuario.
Precondición	El usuario debe estar autenticado y debe tener juegos añadidos a su colección, si están puntuados será más preciso.
Flujo principal	1. Si el usuario accede a la página de usuarios registrados.

	- El sistema analiza la similitud de gustos entre el usuario y los demás usuarios. - El sistema presenta una lista de juegos jugados por los usuarios más similares, que pueden gustarle 2. Si el usuario accede a la página donde se muestran los detalles de un juego. - El sistema analiza la similitud de gustos entre el usuario y los demás usuarios y la similitud entre el juego visitado y los juegos contenidos en la base de datos - El sistema presenta una lista de juegos tanto de similares al juego visitado como de juegos jugados por usuarios similares 3. Si el usuario accede a una lista específica. - El sistema analiza la similitud entre los juegos contenidos en la lista con los guardados en la base de datos. - El sistema presenta una lista de juegos que el usuario podría agregar a esa lista.
Flujo alternativo	Si la lista de recomendaciones que devuelve el sistema está vacía porque el usuario no tiene juegos puntuados/añadidos a listas, no aparecerán recomendaciones.
Postcondición	El usuario visualiza las recomendaciones y puede explorar detalles de los juegos sugeridos.

Tabla 16. Caso de uso: Obtener recomendaciones de juegos.

Capítulo 4. Diseño de la aplicación

El propósito de esta fase de diseño es describir la arquitectura de la aplicación web. Dado que se trata de un proyecto de desarrollo web, se utilizará una arquitectura de tres niveles (presentación, aplicación y datos) ya que según IBM (2024) ha sido la arquitectura predominante durante décadas para las aplicaciones cliente-servidor. Además, se seguirá el patrón de diseño Modelo-Vista-Controlador (MVC), aunque es importante señalar que React no se adhiere estrictamente a este patrón. En su lugar, utiliza una arquitectura basada en componentes donde el flujo de datos es unidireccional (Figura 7). Esto significa, que los datos se pasan de componentes principales a secundarios utilizando "props" (atributos que se definen en el componente padre y se pasan al componente hijo para configurarlo y personalizar su comportamiento). Los cambios en los datos se manejan a través de actualizaciones de estado que fluyen hacia arriba en la jerarquía de componentes.

Esta estructura hace que la aplicación sea más predecible y fácil de entender, ya que cada componente tiene un propósito claro y las actualizaciones de estado son manejadas de manera controlada.

Para la comunicación con el backend, se utilizan peticiones HTTP con Axios, aprovechando así la capacidad de React para gestionar la vista de manera eficiente sin adherirse estrictamente al patrón MVC.

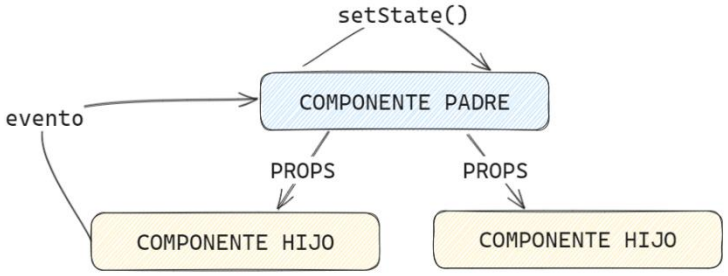

Figura 7. Diagrama de flujos de datos unidireccional (React).

La principal ventaja de seguir una arquitectura de tres niveles es que cada nivel se puede desarrollar, modificar y escalar de forma independiente, sin afectar a los demás. Estos niveles son:

- **Nivel de presentación**: El usuario final interactúa con la aplicación, mostrando información y recopilando datos, este nivel se ejecuta en el navegador web y está desarrollado haciendo uso de React.
- **Nivel de aplicación o lógico**: La información recopilada en el nivel anterior se procesa utilizando la lógica de negocio que ejecuta métodos CRUD, encargados de crear, leer, eliminar y actualizar datos en el nivel de datos. Desarrollado utilizando NodeJS.
- **Nivel de datos**: Se almacena y gestiona la información procesada por la aplicación, en este caso se utiliza MySQL, una base de datos relacional. La comunicación con la base de datos se realiza a través de Sequelize, un ORM (Object-Relational Mapping) que simplifica las operaciones CRUD y la gestión de esquemas de base de datos, proporcionando una capa de abstracción sobre las consultas SQL.

En la Figura 8 se puede observar la arquitectura que sigue la aplicación con los niveles diferenciados.

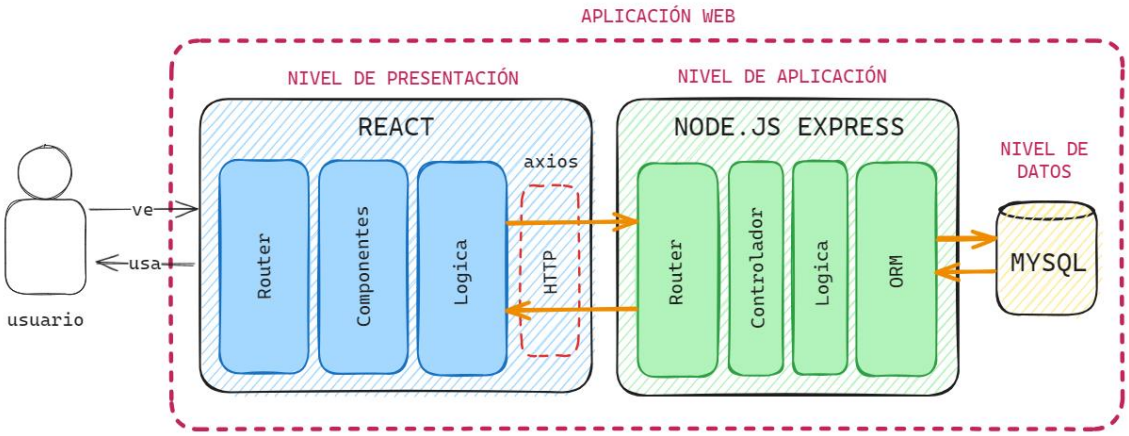

Figura 8. Diagrama de arquitectura de la aplicación.

El patrón Modelo-Vista-Controlador (MVC) es una arquitectura de software ampliamente utilizada para estructurar aplicaciones web, ya que proporciona una separación clara de responsabilidades, lo que facilita el desarrollo, mantenimiento y escalabilidad de las aplicaciones.

- **Modelo**: Se encarga de la gestión de los datos consultando la base de datos. Como vemos en la Figura 8, en este caso, se encarga el ORM. Para ello, se han definido los modelos que representan las entidades de la base de datos (tablas) y sus relaciones, las cuales se utilizan para realizar consultas y manipular los datos.
- **Vista**: Es la representación visual de los datos para el usuario final. En este caso, el back-end proporciona los datos necesarios al front-end a través de la API REST. El front-end solicita estos datos al back-end, los procesa y los utiliza para generar la vista que se mostrará al usuario.
- **Controlador**: Recibe las peticiones del usuario y gestiona la lógica para solicitar datos al modelo y comunicarlos a la vista. En este escenario, los controladores se corresponden con las rutas y los métodos HTTP del back-end, los cuales manejan las solicitudes entrantes, llaman al método de la lógica que interactúa con el modelo para obtener los datos y luego los pasan a la vista para su presentación al usuario.

4.1 Base de datos

El diseño de las tablas de la base de datos es una fase crucial en el desarrollo, define cómo se almacenará, organizará y gestionará la información. Este proyecto utiliza MySQL como sistema de gestión de bases de datos relacional, ya que proporciona robustez, eficiencia y capacidad de manejar grandes volúmenes de datos. A continuación, se detallan las estructuras de las principales tablas de la base de datos, así como las relaciones entre ellas (Figura 9).

La tabla “lista”, contiene información propia de la lista como el id que la identifica, el nombre, la descripción y el usuario al que le pertenece. La tabla “juego” contiene información detallada sobre cada juego, además de su identificador y el identificador de la API de IGDB. La tabla “usuario” contiene los datos del usuario tales como su identificador, email, nombre,

contraseña y token de steam. La tabla “preferencias” contiene las puntuaciones que los usuarios han proporcionado a cada juego.

Las tablas “lista” y “juego”, tienen una relación muchos a muchos, una lista puede tener muchos juegos y un juego puede pertenecer a muchas listas, se relacionan a través de la tabla “lista_juego”.

Las tablas “usuario” y “lista”, tienen una relación 1 a muchos, debido a que un usuario puede tener muchas listas, pero una lista no puede pertenecer a muchos usuarios.

Las tablas “usuario” y “preferencias”, tienen una relación 1 a muchos, un usuario puede puntuar muchos juegos, pero una puntuación no puede pertenecer a muchos usuarios. Pasa lo mismo con las tablas “juego” y “preferencias”.

Figura 9. Diseño de la base de datos.

4.2 Mockups

Los mockups son esenciales para visualizar la apariencia y la funcionalidad de la aplicación antes de su desarrollo completo, tal y como se ven en las Figuras 11-21. Estos elementos ayudan a entender cómo los usuarios interactuarán con la interfaz y proporcionan una guía visual para entender mejor el desarrollo.

Se ha seguido un enfoque **mobile first**. Este enfoque se centra en diseñar primero para dispositivos móviles antes de escalar el diseño hacia dispositivos de mayor tamaño, como tabletas y computadoras de escritorio. La razón principal detrás de esta estrategia es que cada vez son más los usuarios que acceden a internet a través de sus dispositivos móviles, además proporciona una base sólida para un diseño adaptable, asegurando que todos los usuarios tengan una experiencia agradable, independientemente del dispositivo que utilicen.

Como parte de nuestro diseño, se ha incorporado un logo (Figura 10) distintivo que se adapta a la estética del sitio web y que refleja la identidad de la marca, el búho simboliza las recomendaciones inteligentes y personalizadas, los símbolos gaming dentro de los ojos del búho indican la temática de la aplicación, de este modo se intenta lograr que la experiencia del usuario sea aún más coherente y memorable.

Figura 10. Logo

Figura 11. Mockup inicio de sesión.

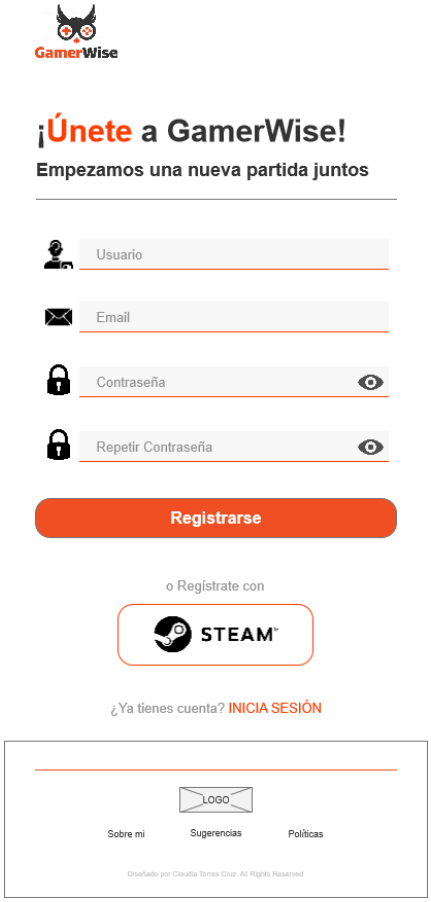

Figura 12. Mockup Landing Page

Figura 13. Mockup perfil "Listas"

Figura 14. Mockup perfil "Puntuaciones"

Figura 15. Mockup perfil "Opciones"

Figura 16. Mockup panel "Crear Lista"

Figura 17. Mockup página de lista

Figura 18. Mockup página de detalles de un juego

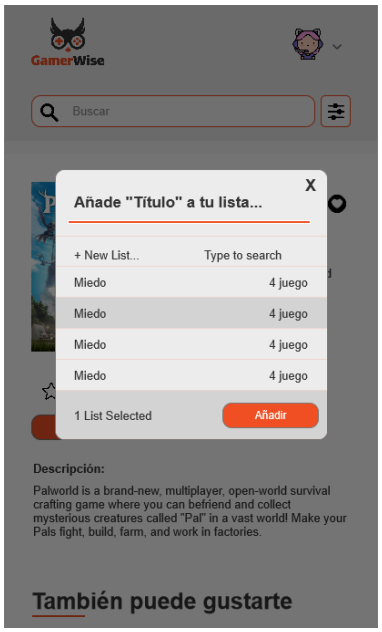

Figura 19. Mockup panel “Añadir a lista”

Figura 20. Mockup Landing Page

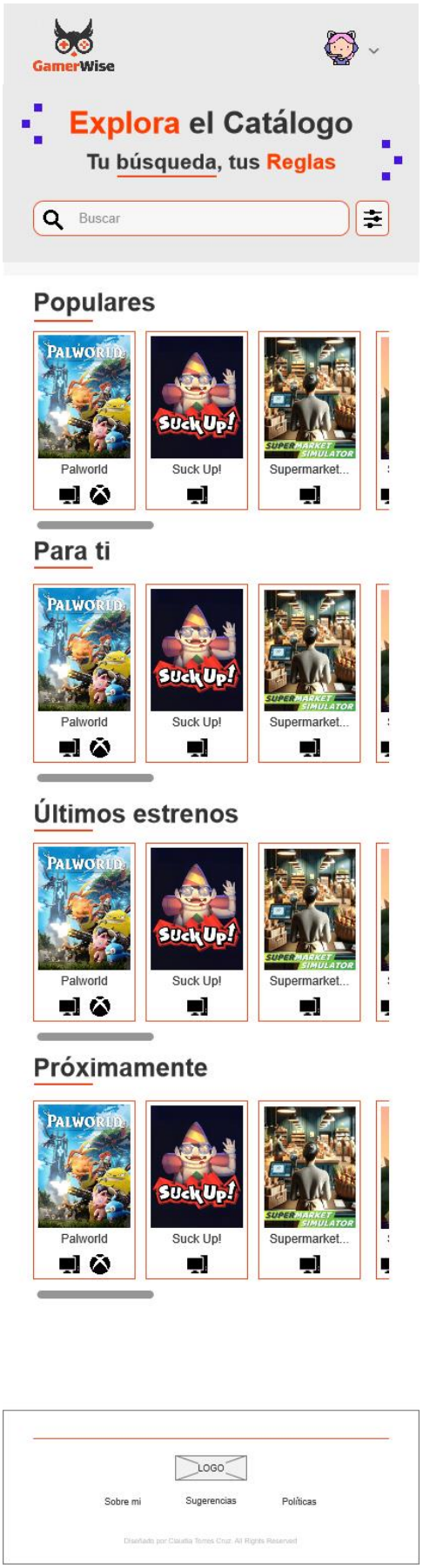

Figura 21. Mockup Landing Loggeados

4.3 Backend y Endpoints

Para la comunicación entre el frontend y el backend, se utiliza una API REST. Es una arquitectura que permite la comunicación entre sistemas mediante el protocolo HTTP, utilizando métodos estándar como GET, POST, PUT y DELETE. Las API RESTful son conocidas por su simplicidad, escalabilidad y compatibilidad con múltiples lenguajes y plataformas, convirtiéndolas en una opción ideal para aplicaciones web modernas.

Cuando el backend recibe una petición HTTP, esta solicitud activa un proceso dentro del servidor. El enrutador actúa como el gestor principal de las peticiones entrantes, recibe y dirige esta solicitud HTTP al lugar adecuado dentro de la aplicación. El controlador, extrae datos cruciales de la petición, como parámetros de la URL o información del cuerpo. Después, invoca métodos dentro de la capa de lógica de negocio, encargada de comprobar que los datos son aptos para ejecutar operaciones CRUD y devolver la respuesta de las operaciones al controlador.

El controlador, una vez que ha recibido la respuesta de la lógica de negocio, se encarga de formatearla adecuadamente para una respuesta HTTP. A continuación, en la Figura 22 se muestra un diagrama de este proceso a nivel reducido:

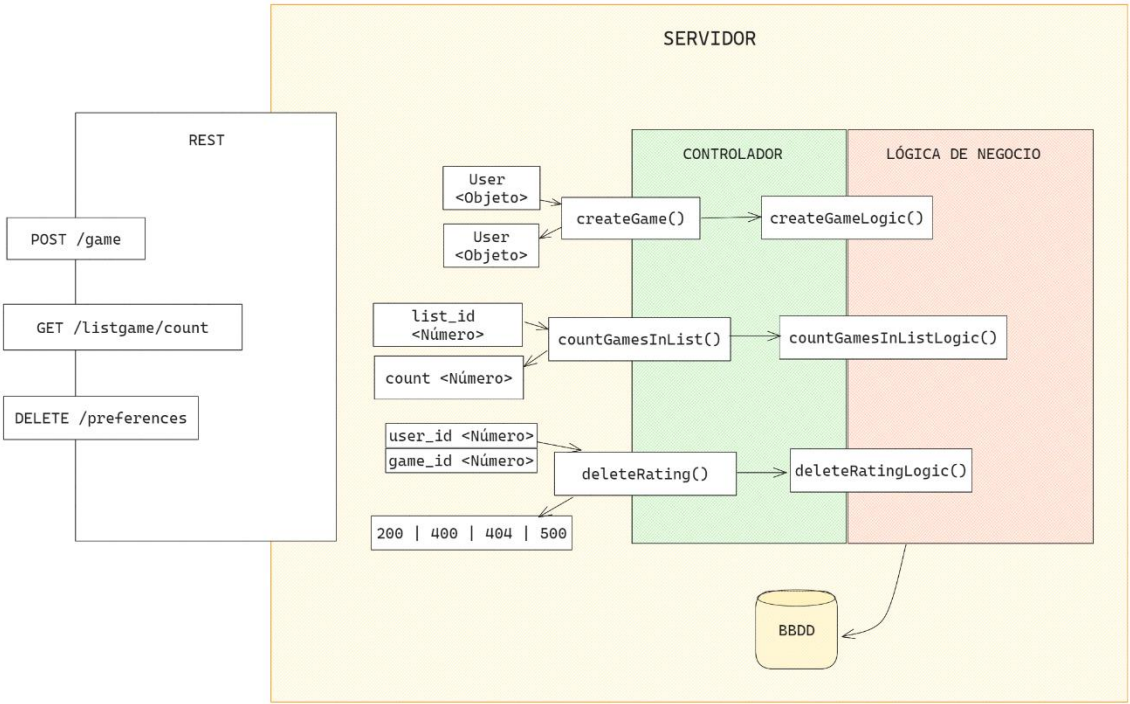

Figura 22. Diseño backend

La aplicación utiliza endpoints como puntos de entrada fundamentales para la interacción entre el cliente y el servidor. Cada endpoint está asociado a una función dentro de la aplicación, permitiendo a los usuarios interactuar con la web. Aquí se describen los principales conjuntos:

- **Gestión de usuarios:** Manejan todas las operaciones relacionadas con los usuarios, incluyendo inicio de sesión, registro, actualización de datos, y obtención de información personal.

- **Gestión de juegos:** Se encuentran los endpoints encargados de todas las operaciones relacionadas con los juegos, como la creación y obtención de juegos.
- **Interacción con IGDB**: Se encargan de realizar peticiones a la base de datos pública de videojuegos de IGDB (Internet Games Database). IGDB es una base de datos que proporciona información detallada sobre videojuegos, incluyendo datos como títulos, descripciones, plataformas, fechas de lanzamiento, etc. Este conjunto de endpoints se utiliza para realizar búsquedas con filtros específicos, obtener los últimos lanzamientos, entre otras operaciones.
- **Gestión de listas:** Permiten crear, editar y eliminar listas de juegos.
- **Gestión de puntuaciones:** Gestionan las puntuaciones que los usuarios añaden a los videojuegos.
- **Gestión de juegos en listas:** Permiten, entre otros, añadir juegos a una lista, eliminarlos, obtener todos los juegos de una lista específica, facilitando la organización de las colecciones de juegos de los usuarios.
- **Integración con Steam:** Están dedicados a la comunicación con la API de Steam. Permiten iniciar sesión utilizando Steam como método alternativo y vincular la cuenta.
- **Gestión de recomendaciones:** Se encarga de gestionar las recomendaciones de juegos, proporcionando a los usuarios sugerencias basadas en sus preferencias y actividad dentro de la aplicación.

4.4 Diagrama de flujo

El diagrama de flujo (Figura 23) detalla las rutas que los usuarios pueden seguir al interactuar con los diferentes botones de la aplicación y los endpoints que se invocan en cada caso. Para acceder a la página de autenticados, a las diferentes secciones del perfil, a las listas y para interactuar con las opciones de la página de detalles del juego, se requiere estar autenticado, ya que se tratan de rutas protegidas.

Estas rutas aseguran que sólo los usuarios autenticados puedan acceder a áreas específicas de la aplicación, proporcionando una capa adicional de seguridad. Para implementarlas, se ha utilizado el token JWT generado en el backend que se envía como cabecera en las solicitudes a los endpoints protegidos. Este token, se genera durante el proceso de inicio de sesión e incluye el ID del usuario, de modo que se verifica su identidad.

Cuando un usuario intenta acceder a una ruta protegida, la aplicación verifica si existe un token válido. Si el token es válido, el usuario puede acceder a la página solicitada. En cambio, si no lo es o no está presente, el usuario es redirigido a la página de inicio de sesión.

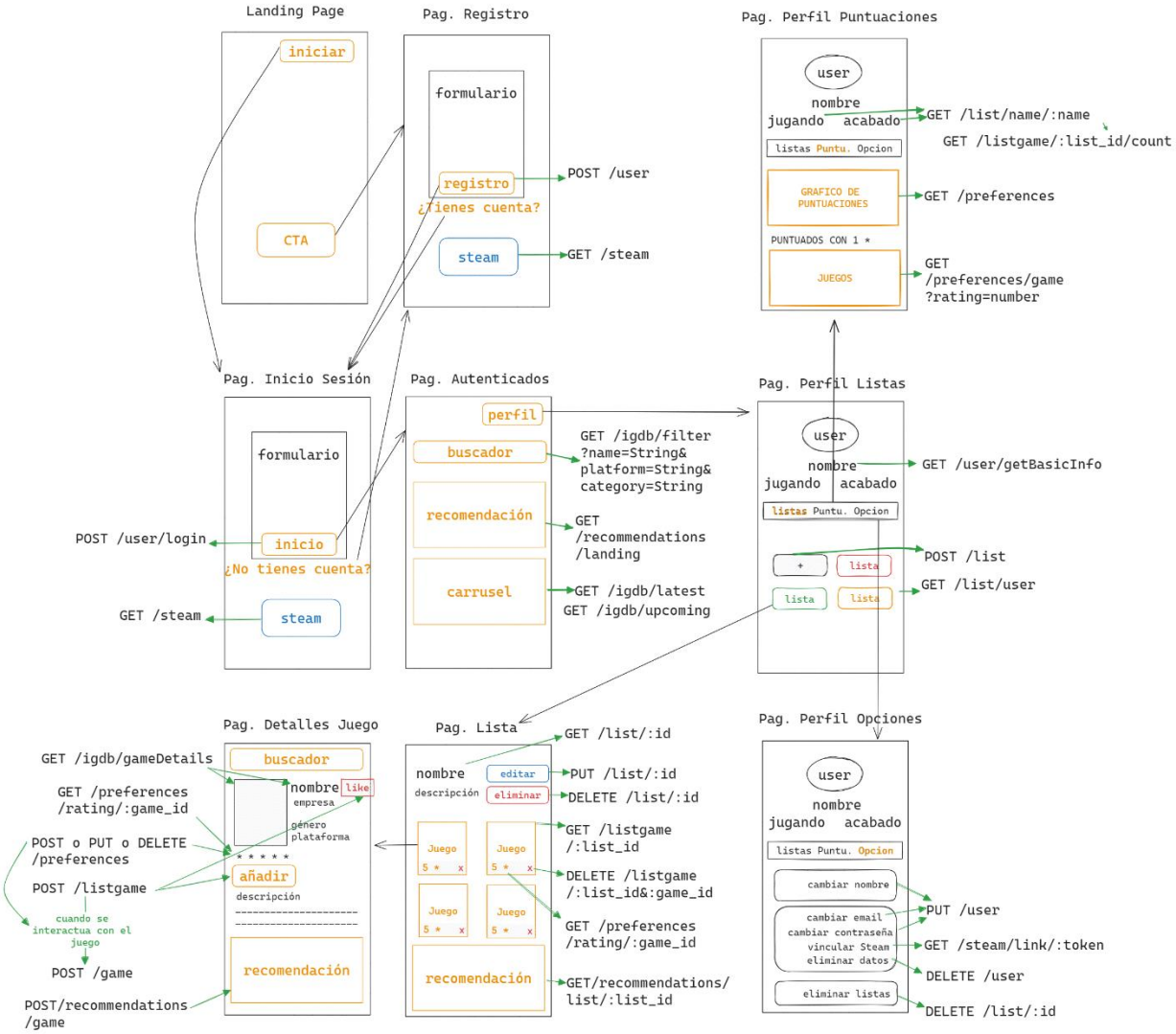

Figura 23. Diagrama de flujo

Capítulo 5. Implementación

En este capítulo se cubrirá el desarrollo del front-end con React, la implementación del back-end con NodeJS, las medidas de seguridad seguidas y la implementación de los algoritmos de recomendación. Además, se incluirán ejemplos de código y se abordarán los principales desafíos y soluciones encontrados durante el desarrollo. Esta fase se realiza teniendo en cuenta la fase de análisis de requisitos y la fase de diseño.

5.1 Backend con Node.js

El backend se encarga de gestionar la lógica del servidor y la comunicación con la base de datos. En esta sección se describe cómo ha sido el proceso de desarrollo para configurarlo y gestionarlo.

En primer lugar, hay que conectar nuestro ORM con la base de datos MySQL. Se utiliza el constructor de Sequelize para crear una instancia, además como vemos en la Figura 24, se define también la base de datos que se utilizará para comprobar que nuestros endpoints funcionan correctamente:

```
import { Sequelize } from "sequelize";

const dbConfig =
  production: { {
    database: "gamerwise_db",
    username: "root",
    password: "",
    host: "localhost",
    dialect: "mysql",
  },
  test: {
    database: "gamerwise_db_test",
    username: "root",
    password: "",
    host: "localhost",
    dialect: "mysql",
  },
};

const getDBInstance = () ⇒
    const env = process.env{NODE_ENV || 'production'
;
    const config = dbConfig[env];
    return new Sequelize(config.database, config.
username, config.password, {
      host: config.host,
      dialect: config.dialect,
      logging: false
    });
  };

  const db = getDBInstance();

  export default db;
```

Figura 24. Instancia base de datos

Luego, se crean los modelos Sequelize para manejar la base de datos. Estos modelos son clases de JavaScript que encapsulan la estructura y el comportamiento de una tabla en la

base de datos. Se definen utilizando la instancia declarada en la Figura 24, y se especifican los atributos de la tabla, sus tipos de datos, las relaciones con otras tablas y las restricciones (Figura 25). Las relaciones entre los modelos se definen en un archivo a parte para facilitar futuras modificaciones (Figura 26).

```
//connection db
import db from '../database/db.js'
import DataType from 'sequelize'

const ListModel = db.define('list', {
    //Atributes
    name:{
        type: DataType.STRING,
        allowNull: false
    },
    description:{
        type: DataType.STRING,
        // allow null true
        defaultValue: 'null'
    }
},{
    //Other model options
    freezeTableName: true,
// stip the auto-pluralization
    timestamps: false
})
export default ListModel
```

Figura 25. Ejemplo modelo de Sequelize

```
// relation 1-n Game*Preferences
GameModel.hasMany(PreferencesModel, {
    foreignKey: {
        name: 'game_id',
        allowNull: false
    }
})
PreferencesModel.belongsTo(GameModel,{
    foreignKey: {
        name: 'game_id',
        allowNull: false
    }
})

// relation n-n List*Game
ListModel.belongsToMany(GameModel, {
    through: { model: ListGameModel,
unique: false },
    foreignKey: {
        name: 'list_id',
        allowNull: false,
    },
});
```

Figura 26. Ejemplo relaciones de Sequelize

Con el ORM configurado, está listo para conectarse a la base de datos y establecer el puerto desde el que nuestro servidor empezará a recibir peticiones HTTP (Figura 27). El siguiente paso es empezar a configurar los enrutadores, que son manejadores de rutas modulares usados para soportar diferentes métodos HTTP (POST, PUT, DELETE) dentro de la aplicación (Figura 28).

```
try
  await db.authenticate();
  console.log(
"Conexion exitoa a la base de datos");
} catch  error) {
  console.log(`El error de conexion es: ${
error}`);
}

app.listen(8000, () =>
  console.log(        {
"Server UP running in http://localhost:800
0/"
);
});
```

Figura 27. Conexión base de datos

```
import express from "express";
import { createGame, getGame, getAllGames
 } from "../controllers/GameController.js"
import { verifyToken } from
'../utils/auth.js'
const router = express.Router();

router.post('/', verifyToken, createGame)
router.get('/:igdb_id', verifyToken,
getGame)
router.get('/', getAllGames)

export default  app) =>  app.use("/game",
router)}      (        {
```

Figura 28. Ejemplo enrutador Express

Los enrutadores deben tener asignados métodos que se ejecutarán cuando se reciba una petición que coincida con la ruta. Estos métodos son los controladores, que se encargan de recibir los parámetros y cabeceras de las peticiones, comunicarse con la lógica y preparar las respuestas HTTP (Figura 29).

```
// create a new Game
export const createGame = async  req, res) ⇒
    try                        (             {
       {const { name, company, platforms, max_players, gender, igdb_id, release_date, cover } = req.body;
        const { success, newGame, error } = await createGameLogic(name, company, platforms, max_players, gender
, Number(igdb_id), cover, Number(release_date));
        if  success) {
          ( res.status(201).json({ message: "Game created successfully", game: newGame
        } else                                                                    });
            re{.status(400).json({ message: error
        }                                        });
    } catch  error) {
        res(status(500).json({ message: error.message
    }                                                 });
};
```

Figura 29. Ejemplo de controlador

La lógica de la aplicación (Figura 30) es la encargada de comprobar que los datos recibidos son correctos antes de interactuar con los modelos y devolver a los controladores información de la consulta a los controladores.

```
// create a new Game
export async function createGameLogic(name, company, platforms, max_players, gender, igdb_id, cover,
release_date) {
    console.log(name, company, platforms, max_players, gender, igdb_id, release_date)
    try
       {// Validar campos requeridos
        if  !validateRequiredFields({ name, company, platforms, max_players, gender, igdb_id, release_date})) {
          ( throw new Error("Required fields are not provided");
        }
        // Validar tipos de datos
        if  !validateDataTypes({ name, company, platforms, max_players, gender, igdb_id, release_date })) {
          ( throw new Error("Invalid data type");
        }
        // Crear nuevo juego
        const newGame = await GameModel.create({ name, company, platforms, max_players, cover, release_date,
gender, igdb_id
        return }){success: true, newGame };
    } catch  error) {
        retúrn { success: false, error: error.message };
    }
}
```

Figura 30. Ejemplo de lógica

Esta estructura modular permite una mejor gestión del código, facilitando la ampliación de funcionalidades, la modificación y la detección de errores.

5.2 Frontend con React

El frontend de la aplicación web está desarrollado utilizando React, que permite crear aplicaciones web rápidas, escalables y fáciles de mantener gracias a su arquitectura basada en componentes.

Lo primero que se hizo fue establecer en base a los mockups, los componentes que tendría la web. Cada componente en React es una pieza reutilizable de la interfaz del usuario que pueden combinarse para construir una aplicación completa. Los componentes pueden ser simples como un elemento de contenido (Figura 31) o complejos como una página completa.

```jsx
import "../styles/ListSection.css";

export function ListPreview({ name, color, newlist, list_id }) {
  const handleListClick = () =>
    if  newlist) {             {
      console.log("crear nueva lista");
    } else
      console.log("entrar a", list_id);
    }
  };
  return
    <div(
      style={{
        border:`2px solid ${color}`,
        color: "black",
        backgroundColor: newlist ? color : "rgb(249 249 249)"
      }}
      className="list-container"
      onClick={handleListClick}
    >
      {name}
    </div>
  );
}
```

Figura 31. Ejemplo de componente de React

Para gestionar el estado de la aplicación, se utilizan los contextos de React y los hooks. Los hooks son funciones de javascript que permiten crear/acceder al estado y a los ciclos de vida de React. Se pueden crear hooks personalizados, que encapsulan lógica de estado reutilizable y ayudan a mantener el código limpio y organizado. Un hook destacable de la aplicación sería el que se encarga de la funcionalidad de las búsquedas (Figura 32).

```jsx
export function useGames({ query, sort, category }) {
  const  games, setGames] = useState([]);
  const[ loading, setLoading] = useState(false);
  const[ error, setError] = useState(null);
  const[prevSearch = useRef();
  const prevCategory = useRef();
  const prevPlatform = useRef();


  const getGames = useCallback(async ({ query, category, platform }) =>
    //inyectado para que solo se ejecute 1 vez                    {
    if  query === prevSearch.current & category === prevCategory.current & platform === prevPlatform.current) return;
    try
      prevSearch.current = query;
      prevCategory.current = category;
      prevPlatform.current = platform;
      setLoading(true);
      setError(null);
      const data = await searchGame(query, category, platform);
      setGames(data);
    } catch  error) {
      setError(error.message);
    } finally
      // esto{es lo que entra tanto en el try y en catch
      setLoading(false);
    }
  }, []);

  return { games, getGames, loading, setGames };
}
```

Figura 32. Ejemplo de hook de React

Los contextos permiten compartir el estado global de la aplicación entre múltiples componentes sin necesidad de pasar props manualmente a través de su jerarquía y así simplificar la gestión del estado y mejora la escalabilidad del código. Como contexto importante tenemos el de autenticación, que se encarga de todo lo que tiene que ver con registrarse, iniciar sesión y su labor a su vez es guardar y recuperar el token de sesión.

La comunicación con el backend se realiza a través de solicitudes HTTP a la API gestionadas mediante la biblioteca Axios (Figura 33), que facilita la realización de peticiones y el manejo de respuestas.

```
export const registerRequest = async  user) ⇒
    try                              (          {
       {const res = await axios.post(`${API}/user`, user);
        return res.data;
    } catch  error) {
        thr0w new Error(error.response.data.message);
    }
}
```

Figura 33. Ejemplo de solicitud HTTP con Axios

La navegación entre las diferentes páginas de la aplicación se gestiona utilizando React Router (Figura 34). React Router permite definir rutas dinámicas y anidadas, facilitando una navegación fluida. Esto es muy importante para crear aplicaciones de una sola página (SPA) donde el contenido cambia sin necesidad de recargar la página.

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<AuthRedirect />} />
    <Route path="/login" element={<LoginPage />} />
    <Route path="/signup" element={<RegisterPage />} />
    <Route element={<ProtectedRoute />}>
      <Route path="/logged" element={<LandingLogged />} />
      <Route path="/profile" element={<ProfilePage />} />
      <Route path="/list/:id" element={<ListPage />} />
    </Route>
    <Route path="/game/:id" element={<GameDetailPage />} />
    <Route path="/steam/:token" element={<SteamPage />} />
  </Routes>
</BrowserRouter>
```

Figura 34. React Router

5.3 Seguridad

La seguridad es un aspecto fundamental en el desarrollo de aplicaciones web, ya que protege tanto los datos de los usuarios como la integridad de la aplicación.

JSON Web Tokens (JWT) es un estándar abierto que define una forma de transmitir información de forma segura entre un cliente y un servidor utilizando objetos JSON. En esta aplicación, los tokens JWT se utilizan para autenticar a los usuarios y asegurar que sólo las solicitudes válidas puedan acceder a recursos protegidos. Durante el proceso de inicio de sesión, el backend genera un token JWT que incluye el ID del usuario. Este token se firma con una clave secreta y se envía al cliente (Figura 35). El token contiene información codificada en base64 que puede ser decodificada y verificada por el servidor para autenticar al usuario (Figura 36).

Los endpoints de la API están protegidos mediante el uso de tokens JWT. La mayoría de las rutas requieren que el cliente envíe un token válido en la cabecera de autenticación para poder acceder. Esto asegura que solo los usuarios autenticados puedan realizar ciertas operaciones, como acceder a su perfil o realizar acciones que afectan a la base de datos.

```
export const generateAuthToken =  userId) ⇒
    return jwt.sign({ userId }, (${process.env.JWT_token_secret}`, {
expiresIn: '24h'
};              });
```

Figura 35. Generar token JWT

```
export function verifyToken(req, res, next) {
    const header = req.header("Authorization") || "";
    const token = header.split(" ")[1];
    if  !token) {
      return res.status(401).json({ message: "Token not provied"
    }                                                          });
    try
      const payload = jwt.verify(token, token_secret);
      req.id = payload.id;
      next();
    } catch  error) {
      return res.status(403).json({ message: "Token not valid"
    }                                                        });
  }

export const getUserIdFromToken =  req) ⇒
  const token = req.headers.authorization.$plit(' ')[1];
  const decoded = jwt.verify(token, process.env.JWT_token_secret);
  return decoded.userId;
};
```

Figura 36. Verificar y obtener id del token JWT

Para proporcionar mayor seguridad a los datos del usuario, se utiliza bcrypt, una biblioteca para encriptar contraseñas utilizando un algoritmo de hashing adaptativo que es una técnica de seguridad que aplica múltiples rondas de hashing a la contraseña, haciendo que el proceso de creación del hash sea intencionadamente lento, dificultando los ataques de fuerza bruta, ya que cada intento de adivinar la contraseña requiere una cantidad significativa de tiempo computacional. Este método asegura que las contraseñas almacenadas en la base de datos sean seguras y resistentes a este tipo de ataques.

Antes de almacenar una contraseña en la base de datos, se encripta (Figura 37) y durante el proceso de autenticación, la contraseña ingresada por el usuario se compara con el hash almacenado.

```
export const hashPassword = async  password) ⇒
    try                          (             {
       {const hash = await bcrypt.hash(password, saltRounds);
        return hash;
    } catch  error) {
        throw error; // Rechazar la promesa en caso de error
    }
};
```

Figura 37. Hash contraseña

Por otro lado, se implementan rutas protegidas utilizando React Router y un contexto de autenticación. Estas rutas garantizan que sólo los usuarios autenticados puedan acceder a ciertas páginas de la aplicación, proporcionando una capa adicional de seguridad. Esto no solo protege la información de nuestros usuarios, sino que también previene accesos no autorizados que podrían comprometer la seguridad de la aplicación.

El componente ProtectedRoute (Figura 38) verifica si el usuario está autenticado antes de permitir el acceso a la ruta solicitada. Si el usuario no lo está, se redirige a la página de inicio de sesión. Durante el proceso, se muestra un componente de carga (Loading) mientras se verifica el estado de autenticación.

```
import { useAuth } from "../hooks/useAuth"
import  Navigate, Outlet} from 'react-router-dom'
import{{ Loading } from "./Loading"
export function ProtectedRoute() {

    const  loading, isAuthenticated} = useAuth()
    if(loading) return  <Loading/>)
    if(!loading && !isAuthenticated) return <Navigate to='/login' replace/>

    return <Outlet/>
}
```

Figura 38. Componente ruta protegida

5.4 Algoritmos de recomendación

En esta aplicación se han implementado tres tipos de algoritmos de recomendación con el objetivo de mejorar la experiencia del usuario, estos son: filtrado colaborativo, filtrado basado en contenido y filtrado híbrido.

Se integra en el backend a través de un proceso de Node.js que ejecuta los scripts de Python encargados de realizar las predicciones y devolver una lista de recomendaciones.

5.4.1 Filtrado Colaborativo

El filtrado colaborativo se basa en las interacciones y calificaciones de los usuarios para sugerir nuevos juegos que podrían ser de su interés. Se utiliza en la página que se ve nada más iniciar sesión en la aplicación, a fin de que los usuarios empiecen a experimentar sugerencias cuanto antes.

Se ha implementado este tipo de filtrado utilizando la biblioteca “Surprise”, con el algoritmo de descomposición en valores singulares (SVD). Este enfoque permite identificar patrones en las preferencias de los usuarios y generar recomendaciones personalizadas de juegos que podrían interesarles.

En primer lugar, los datos de calificaciones se obtienen de un método del backend que proporciona información detallada sobre los juegos puntuados por parte de los usuarios. Estos datos se transforman en un DataFrame y se codifican utilizando LabelEncoder, que sirve para transformar etiquetas categóricas en números enteros, y MultiLabelBinarizer empleado para transformar un conjunto de datos, en este caso los géneros de los juegos, en una matriz binaria (Tabla 17).

	Aventura	Terror	Familiar	Etiquetas
0	1	0	1	[“Aventura”, “Familiar”]
1	0	0	1	[“Familiar”]
2	1	1	1	[“Aventura”, “Terror”, “Familiar”]

Tabla 17. Tabla ejemplo de MultiLabelBinarizer

Después, el algoritmo SVD entrena el modelo de recomendación y se optimiza su rendimiento realizando validación cruzada y una búsqueda de hiperparámetros que nos permite encontrar la mejor combinación para minimizar el error cuadrático medio (RMSE) (Figura 39).

```
Mejores hiperparámetros:  {'n_factors': 20, 'lr_all': 0.002, 'reg_all': 0.1, 'n_epochs': 100}
RMSE: 0.3674
```

Figura 39. Hiperparámetros y RMSE

Una vez el modelo está entrenado, se realizan recomendaciones proporcionándole el identificador del usuario y los juegos con los que ya ha interactuado, bien sea porque están incluidos en alguna lista o porque se han puntuado, así se logra evitar sugerencias redundantes y se asegura una buena experiencia de usuario.

5.4.2 Filtrado Basado en Contenido

El filtrado basado en contenido se enfoca en las características de los juegos para hacer recomendaciones. Utiliza técnicas de procesamiento de lenguaje natural (NLP) y similitud del coseno (Figura 40) para encontrar juegos similares basados en sus características textuales.

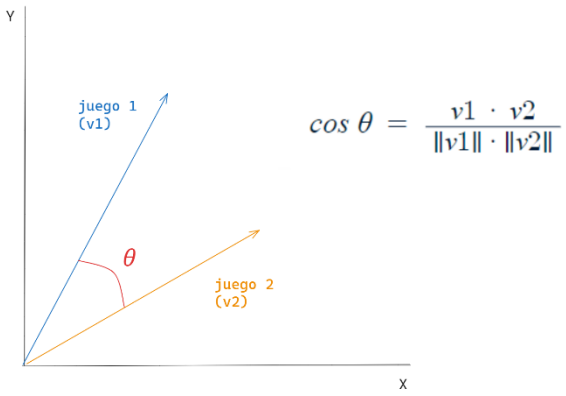

Figura 40. Similitud de coseno

Al igual que en el filtrado colaborativo, los datos de los juegos se obtienen a través de una llamada al backend, se transforman en un Dataframe y luego se combinan las características textuales relevantes de los juegos (nombre, género, compañía y plataformas) en una sola columna. Para asegurar una mejor comprensión y análisis de estas características, se utilizan técnicas de vectorización de texto. En particular, se emplea la técnica de TF-IDF (Term Frequency-Inverse Document Frequency), que se convierte el texto en una matriz de características numéricas, donde cada término se pondera por su frecuencia en el documento y su rareza en el corpus, resultando eficaz para resaltar términos importantes y reducir el peso de

los que pueden no ser informativos. Con la matriz TF-IDF obtenida, se calcula la similitud del coseno entre los juegos.

Se ha incluido este tipo de filtrado en la página donde se visualizan los juegos que contiene una lista, de este modo los usuarios podrán añadir aquellos que se asemejen a los que ya hay incluidos.

5.4.3 Filtrado Híbrido

El filtrado híbrido combina técnicas de filtrado colaborativo y filtrado basado en contenido para ofrecer recomendaciones más precisas y personalizadas. Sugiere títulos que le puedan gustar al usuario cuando está mirando los detalles de un juego.

En primer lugar, se generan recomendaciones basadas en contenido enfocadas en las características del juego visitado y recomendaciones colaborativas basadas en los usuarios y sus puntuaciones. Posteriormente, se combinan las listas de ambos métodos eliminando duplicados, y finalmente se devuelve una lista de recomendaciones equilibrada.

Este método garantiza que los usuarios reciban sugerencias relevantes y variadas, mejorando la experiencia de uso en la plataforma.

5.5 Desafíos y soluciones

Durante el desarrollo de la aplicación, se enfrentaron varios desafíos técnicos. En esta sección se describen los problemas más significativos y las soluciones implementadas.

Uno de los desafíos principales fue la autenticación con la API de Steam. A diferencia de una llamada a un endpoint estándar, la API de Steam no permite una comunicación directa desde cualquier cliente, sino que autoriza a una dirección específica, en este caso fue la del backend. Esto implica que el flujo de autenticación no podía manejarse desde el frontend. Para resolverlo, se implementó una solución que involucra la apertura de una ventana emergente que se dirige a la dirección IP de nuestro backend. Desde esta ventana, se inicia el proceso de autenticación con Steam, una vez que es exitosa, el backend recibe el token de autenticación y redirige al usuario de vuelta al frontend con el token incluido en la URL del callback. Este flujo asegura que el proceso sea seguro.

Otro desafío significativo fue la integración de la base de datos de IGDB con la de los juegos obtenidos tras iniciar o vincular Steam. No siempre hay coincidencias directas entre los nombres, fechas o sitios web de los juegos en ambas bases de datos, lo que complica la fusión de información. Para solucionar este problema, se implementó una estrategia que consistió en obtener el nombre que recibía el juego en Steam, este se transformaba a minúsculas y se eliminaban los símbolos para estandarizar el formato. Luego, se realizaba una búsqueda en IGDB para obtener hasta 10 juegos que coincidieran con el nombre transformado. De estos, se seleccionaba aquel que se ajustaba, puesto que en algunos casos no era hasta el 5º resultado, el juego que se esperaba. De este modo las discrepancias entre los nombres en ambas bases de datos se minimizan.

Un tercer desafío fue considerar los juegos que los usuarios han incluido en una lista creada por defecto, determinada como positiva (“Jugando”, “Completados” y “Me gusta”) pero que no han puntuado. Estos pueden ser importantes para el sistema de recomendaciones, ya

que la inclusión en una de las listas mencionadas indica un interés por parte del usuario. Para abordar este problema, se decidió asignar una puntuación intermedia a los juegos que no tienen una puntuación explícita. Específicamente, se asigna una puntuación de 3 (en una escala de 1 a 5). Así aquellos títulos sin puntuación son considerados en las recomendaciones sin sesgar excesivamente los resultados.

Capítulo 6. Evaluación de la aplicación.

En este capítulo se aborda la evaluación de la aplicación desarrollada, describiendo los métodos utilizados para medir la satisfacción de los usuarios y la efectividad del sistema. Permite identificar áreas de mejora, validar las funcionalidades implementadas y asegurar que la aplicación cumpla con las expectativas y necesidades de los usuarios.

6.1 Encuesta de usabilidad

La evaluación del sistema de recomendaciones y la experiencia general de la aplicación se realizó mediante una encuesta dirigida a los usuarios, siendo 12 los participantes. El objetivo fue recopilar opiniones sobre la usabilidad de la aplicación, en la que se incluyeron preguntas tanto cerradas como abiertas para tener una visión completa de la experiencia del usuario.

A continuación, se muestran los resultados obtenidos de las preguntas relacionadas con la autenticación con Steam y la sincronización de los juegos.

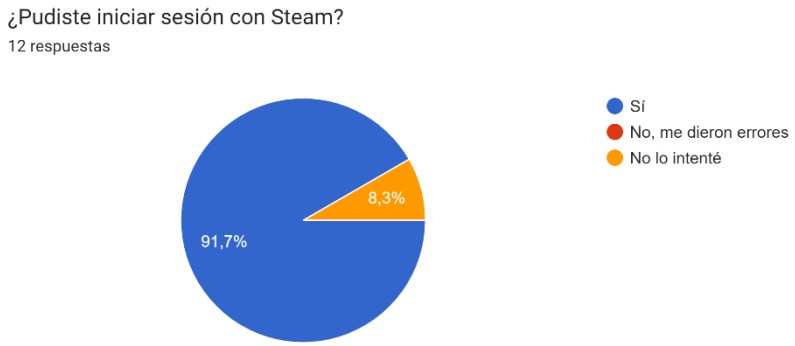

Figura 41. Gráfica de la encuesta de inicio de sesión con Steam

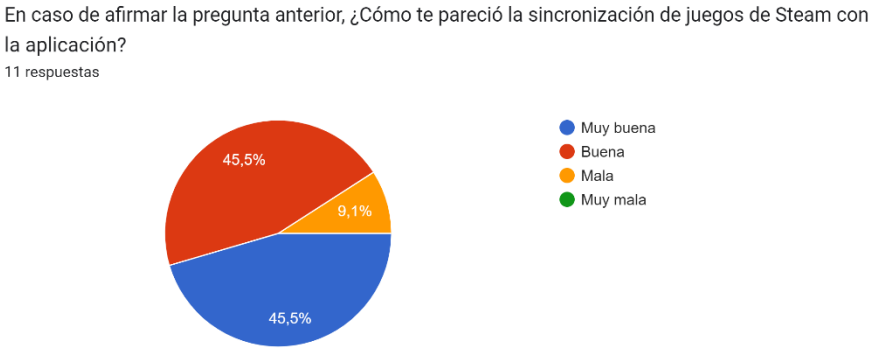

Figura 42. Gráfica de la encuesta de la sincronización de juegos

La mayoría de los usuarios, 91,7%, aprovecharon la opción de inicio de sesión con sus cuentas de Steam sin problemas (Figura 41), indicando que es una funcionalidad beneficiosa para la aplicación. Además, más de la mitad de los usuarios evaluaron positivamente la sincronización de juegos (Figura 42), con el 45,5% calificándola como "Muy buena" y otro 45,5% como "Buena". Sin embargo, un 9,1% que equivale a 1 usuario de los 11 que respondieron esta pregunta experimentó una mala experiencia. Debido a este resultado, se

realizaron modificaciones en la forma de sincronizar la información de las bases de datos de Steam e IGDB y se descubrió que cuando se normalizaba el nombre faltaba eliminar el símbolo "- ", haciendo que no se encontraran juegos, de este modo se solucionó el problema para mejorar la experiencia de futuros usuarios.

De igual modo, se quiso medir la facilidad de uso de las siguientes funciones principales: crear y eliminar listas, añadir y eliminar juegos de una lista, y puntuarlos.

Figura 43. Gráfica de la encuesta sobre la usabilidad de las funcionalidades

Respecto a la usabilidad de las funcionalidades mencionadas (Figura 43), los resultados fueron totalmente positivos con un 100% de respuestas que indican que el sistema es intuitivo, lo que sugiere que la interfaz de usuario está bien diseñada y es fácil de usar.

Al estar este proyecto enfocado no solo en la gestión de videojuegos sino también en la importancia de las recomendaciones, se evaluó la perspectiva del usuario acerca de la precisión y relevancia de las recomendaciones de juegos.

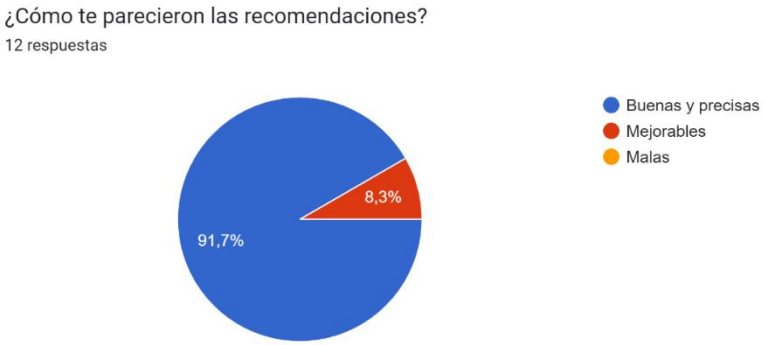

Figura 44. Gráfica de la encuesta acerca la calidad de las recomendaciones

La mayoría de los usuarios (11 de 12) encontraron que las recomendaciones eran buenas y precisas (Figura 44). Solo uno sugirió que podrían mejorarse, indicando un alto nivel de satisfacción con la calidad de las recomendaciones. La forma en la que se mejorarían sería distribuyendo la web a más personas, ampliando así los usuarios y sus interacciones, para que el modelo pudiera generar más relaciones.

También, se recopilaron opiniones sobre el diseño general de la aplicación.

Figura 45. Gráfica de la encuesta acerca del diseño de la aplicación

La mayoría de los usuarios encontró el diseño coherente y adecuado (Figura 45), aunque tres usuarios mencionaron que le faltaban detalles o que es mejorable, sugiriendo un área que se puede mejorar a futuro e incrementar la experiencia visual.

Por último, se recogió la satisfacción general de la aplicación.

Figura 46. Gráfica de la encuesta acerca la satisfacción general con la aplicación

Todos los usuarios encuestados recomendarían la aplicación a sus contactos (Figura 46), por lo que, en general, los usuarios están satisfechos con las funcionalidades y la usabilidad del sistema. Estos resultados positivos reflejan el éxito de las soluciones implementadas y proporcionan una base para futuras mejoras y optimizaciones.

6.2 Pruebas de los endpoints

Para garantizar que los endpoints de la aplicación funcionan adecuadamente, se han implementado pruebas automatizadas utilizando el framework Jest. Este framework es popular por su simplicidad y eficacia en la realización de pruebas tanto unitarias como de integración. Las pruebas se han estructurado siguiendo el modelo "Arrange (Preparar), Act (Actuar) y Assert (Afirmar)", que ayuda a organizar y comprender el proceso de prueba (Figura 47).

Durante la fase “Arrange (Preparar)”, se configura el entorno necesario para la prueba. Esto incluye la creación de datos de prueba y la inicialización de cualquier objeto o servicio necesario para la ejecución de la prueba. Después, en la fase “Act (Actuar)”, se ejecuta el código que se desea probar. Generalmente implica llamar al endpoint con los datos de prueba

preparados en la fase anterior. Finalmente, la fase “Assert (Afirmar)”, verifica los resultados de la prueba, comparando los resultados obtenidos con los resultados esperados para determinar si la prueba pasa o falla.

```
const request = testServer(userRoutes);

describe("UserRoutes", () =>
  describe("[ routes / user ]", () =>
    it("should create a new user", async () =>
      // Arrage                                {
      const user =
        email: "te$t@test.com",
        name: "test",
        password: "Test12345678!",
      };
      // Act
      const { status, body } = await request.post("/user").send(user);
      // Assert
      expect(status).toEqual(201);
      expect(body.message).toEqual("User created successfully");
    });
```

Figura 47. Ejemplo de prueba de endpoint

Además, para asegurar confiabilidad y robustez en el sistema, se configura un servidor de prueba implementado con “supertest” y “Express” (Figura 48), que facilita la ejecución de pruebas de integración, ya que proporciona un entorno aislado para validar las rutas.

```
import express from 'express';
import supertest from 'supertest';

function testServer(route){
    const app = express();
    app.use(express.json());
    route(app)
    return supertest(app)
}

export default testServer
```

Figura 48. Código del servidor de prueba

Capítulo 7. Manuales de uso.

En este capítulo se proporcionan las instrucciones para instalar y configurar el proyecto en un ordenador.

7.1 Compilación.

En primer lugar, para la puesta en marcha de la aplicación se necesita disponer de XAMPP o un programa similar y se pondrán en marcha los servicios “Apache” y “MySQL”.

Tras clonar el repositorio https://github.com/ClaudiaTC02/TFG_GamerWise se deben realizar los siguientes pasos:

1. Acceder desde la terminal a la carpeta donde se aloja el backend “node” y ejecutar el comando “npm install” para instalar las dependencias necesarias de JavaScript.
2. Desde la misma carpeta acceder a la que se llama “recommender” y ejecutar "pip install -r requirements.txt”, de esta forma se instalan los paquetes de Python necesarios.
3. Se importarán las bases de datos contenidas por la carpeta “database”, de este modo tendremos las bases de datos listas para su funcionamiento.
4. Para ejecutar los tests se ejecutará “npm test”, mientras que para ejecutar el servidor de producción el comando será “npm start”, si todo ha ido bien, le aparecerá en la terminal un mensaje que indica que el servidor está activo en el puerto 8000.
5. Para poner en marcha el frontend, tan solo deberá abrir un nuevo terminal, dirigirse al directorio “react” y ejecutar el comando “npm run dev”.

Es importante disponer de un archivo con las siguientes variables de entorno:

- **JWT_token_secret**: esta es la clave que se utilizará para firmar el token JWT.
- **IGDB_client_id** y **IGBD_authorization**: proporcionados por la API de IGDB.
- **JWT_test**: token JWT que se utilizará para la realización de las pruebas.
- **STEAM_key**: proporcionado por la API de Steam.
- **EXPRESS_sesion_secret**: es una clave secreta para firmar y verificar la identificación de sesión en una cookie, protegiendo los datos del usuario.

7.2 Claves APIs: IGDB y Steam

Para poder integrar y utilizar las funcionalidades de IGDB y Steam, es necesario obtener claves de API para ambos servicios.

7.2.1 IGDB (Internet Game Database):

1. Visite el sitio web de IGDB.
2. Diríjase a la sección de API y siga los pasos que se muestran para solicitar acceso.
3. Tras completar todos los pasos, obtendrá un client_id y un authorization token que será importante a la hora de hacer peticiones a la API.

7.2.2 Steam:

1. Visite el sitio web dedicado https://steamcommunity.com/dev/apikey .
2. El campo “Nombre” de dominio está destinado a la dirección de la página donde planeas usar una clave, en este caso “GamerWise”.
3. Confirma la solicitud y la clave aparecerá en la página.

Capítulo 8. Conclusiones.

En este capítulo se mencionarán los resultados del proyecto, teniendo en cuenta los objetivos específicos que se han completado durante el proceso de desarrollo para lograr el objetivo general.

A lo largo del documento se muestra que todos los objetivos enumerados en la introducción se han cumplido. Para lograr un desarrollo efectivo, ha sido importante realizar un estudio exhaustivo de las tecnologías actuales de las que dispone el mercado, evaluando sus ventajas y desventajas, para decidir las más adecuadas para completar el proyecto en el tiempo establecido. A su vez, analizar las aplicaciones existentes que ofrecen un servicio similar al propuesto, permitió identificar las oportunidades de mejora que, junto a las personas UX, sirvieron para descubrir las necesidades de los “gamers”.

Los diagramas y mockups facilitaron la visualización del flujo de trabajo y aseguraron que se cumplieran todos los requisitos funcionales. La aplicación fue implementada siguiendo buenas prácticas de desarrollo y abordando y resolviendo problemas técnicos que surgieron durante el proceso. La seguridad y la eficiencia fueron priorizadas, para garantizar un producto lo más robusto posible.

Durante la implementación se realizaron pruebas para comprobar que el desarrollo avanzaba adecuadamente y además prevenir futuros errores. Una vez todos los objetivos se cumplieron, se llevó a cabo una evaluación para medir la satisfacción del usuario. La retroalimentación fue fundamental para realizar ajustes y mejoras.

Finalmente se redactó una documentación detallada del proceso, incluyendo manuales de uso, lo que facilita a los usuarios y desarrolladores comprender y utilizar la aplicación.

En conclusión, el desarrollo de esta aplicación ha sido un proceso enriquecedor y desafiante, que ha resultado en un producto funcional y útil para apoyar la gestión de colecciones de videojuegos en algoritmos de recomendación, cumpliendo así los objetivos establecidos.

8.1 Trabajo a futuro

Aunque la aplicación cumple con los requisitos iniciales y proporciona una experiencia satisfactoria, se pretende continuar el desarrollo de este proyecto mejorando y ampliando las funcionalidades de las que dispone actualmente la web. Algunas de ellas son:

- **Mejoras en la interfaz de usuario.** Aunque la interfaz actual es funcional, la adición de características como mejoras en el estilo de los elementos, ayudará a crear una experiencia aún más atractiva y fácil de usar.
- **Integración con otras plataformas externas**. Actualmente sólo se puede vincular la cuenta con Steam, sin embargo, es interesante contemplar otras plataformas, sobre todo de consola, para mejorar la experiencia de usuario.
- **Funcionalidades sociales**. Añadir elementos sociales, como la posibilidad de seguir a otros usuarios, compartir y ver las colecciones de amigos, puede incrementar el compromiso y la satisfacción del usuario.
- **Soporte multilingüe.** Implementar soporte para múltiples idiomas permitirá a la aplicación llegar a una audiencia más amplia y diversa.

8.2 Relación con las asignaturas del grado

Programación 1 y Programación 2: Los fundamentos de programación obtenidos han sido esenciales para la implementación técnica de la aplicación, proporcionando las bases necesarias para el desarrollo de la lógica de negocio y para abordar la estructura del código.

Diseño de interfaces y experiencia de usuario: Esta asignatura ha permitido crear una interfaz de usuario intuitiva y atractiva, mejorando la experiencia del usuario final.

Metodología CDIO: Ha sido fundamental para la planificación del proyecto, se enfoca en seguir los siguientes pasos: Concebir, Diseñar, Implementar y Operar. De este modo se obtuvieron las herramientas para tener una idea del producto final más precisa y garantizar una entrega de calidad.

Inteligencia artificial: Los conocimientos en esta asignatura han sido aplicados en la implementación de algoritmos de recomendación, ya que ofrece una visión general de cómo funcionan los sistemas IA y, sobre todo, garantiza una buena base para poder entender la forma de actuar de los parámetros modificables en los modelos.

Big data: En esta asignatura optativa, aparte de aprender sobre la gestión de grandes volúmenes de datos (conforme aumente el alcance de la aplicación va a ser interesante conocer la forma de tratarlos) también sirve para familiarizarse con los entrenamientos de los modelos, ya que se utiliza MLlib de Apache Spark, que, aunque Spark y Surprise sirven para propósitos distintos dentro del análisis y manejo de datos, ambos tienen una estructura del código similar.

Proyectos que han tenido relación con el desarrollo web, sobre todo el del primer cuatrimestre de tercero, en el que se debía desarrollar una web muy sencilla que mostraba los datos de un sensor de CO2, capturados a través de beacons por una aplicación móvil y posteriormente subidos a una base de datos, todo testeado y documentado correctamente. La combinación de todas las tecnologías y la importancia que empezó a tomar el testing, ayudaron a conocer buenas prácticas que se han incorporado en este proyecto.

Seguridad en redes y sistemas: Durante el estudio de esta asignatura, se exploraron conceptos fundamentales como el hashing y la encriptación, los cuales resultaron importantes para comprender la tecnología subyacente en la protección de datos sensibles. Estos conocimientos no sólo permitieron una mejor comprensión teórica, sino que también facilitaron la implementación efectiva de bibliotecas de seguridad en el proyecto.

Bibliografía

- *The Most Demanded Frontend Frameworks in 2023*. Consultada el 17 de febrero de 2024. Devjobsscanner. https://www.devjobsscanner.com/blog/the-most-demanded-frontend-frameworks/
- StackOverflow. (2023). *Stack Overflow Developer Survey 2023: Most popular technologies - Web frameworks and libraries*. https://survey.stackoverflow.co/2023/#most-popular-technologies-webframe-prof
- *Stash - Games tracker*. Consultada el 15 de febrero de 2024. https://stash.games/
- *GG*|. Consultada el 15 de febrero de 2024. https://ggapp.io/
- *HowLongToBeat.com* | *Game Lengths, Backlogs and more!* Consultada el 15 de febrero de 2024. https://howlongtobeat.com/
- *React*. Consultada el 19 de febrero de 2024. https://es.react.dev/
- Saks, E. (2019). *JavaScript Frameworks: Angular vs React vs Vue.*
- Challapalli, S. S. N., Kaushik, P., Suman, S., Shivahare, B. D., Bibhu, V., & Gupta, A. D. (2021, November). Web Development and performance comparison of Web Development Technologies in Node.js and Python. In 2021 *International Conference on Technological Advancements and Innovations* (ICTAI) (pp. 303-307). IEEE.
- Dimassi, M. (2023, 6 julio). Choosing the Right Web Framework: Laravel vs. Node.js vs. Django. *Medium*. https://medium.com/@dimassimehdi58/choosing-the-right-web-framework-laravel-vs-node-js-vs-django-240bdc24a07d
- Selvaraj, N. (2023, abril 12). How to Build a Recommendation System in Python. 365 Data Science. https://365datascience.com/tutorials/how-to-build-recommendation-system-in-python/
- Çoğalan, A. (2023, abril 11). Practical Guide to Building Scalable Recommender Systems in Python. *Medium*. https://medium.com/@anilcogalan/practical-guide-to-building-scalable-recommender-systems-in-python-b175547e6fce
- Valencia, S. (2018, junio 19). An Introductory Recommender Systems Tutorial - AI Society - Medium. *Medium*. https://medium.com/ai-society/a-concise-recommender-systems-tutorial-fa40d5a9c0fa
- Komperla, V., Deenadhayalan, P., Ghuli, P., & Pattar, R. (2022). React: A detailed survey. *Indonesian Journal of Electrical Engineering and Computer Science*, 26(3), 1710-1717.
- MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN. (2023, 13 noviembre). MDN Web Docs. https://developer.mozilla.org/es/docs/Glossary/MVC
- *Empezando* | Axios Docs. Consultado el 5 de abril de 2024. https://axios-http.com/es/docs/intro
- Lopez, J. O. (2022, 23 octubre). MVC Pattern in NodeJS and express, old but gold - Jon Oyanguren Lopez - Medium. *Medium*. https://medium.com/@jonoyanguren/mvc-pattern-in-nodejs-and-express-old-but-gold-46c21bee365a
- ¿Qué es la arquitectura de tres niveles? | IBM. Consultado el 15 de junio de 2024. https://www.ibm.com/es-es/topics/three-tier-architecture

- *Guía de relaciones de tablas - Soporte técnico de Microsoft*. Consultado el 17 de junio de 2024. https://support.microsoft.com/es-es/topic/gu%C3%ADa-de-relaciones-de-tablas-30446197-4fbe-457b-b992-2f6fb812b58f
- Mardan, A. (2018). Security and Auth in Node.js. En Apress eBooks (pp. 205-237). https://doi.org/10.1007/978-1-4842-3039-8_6
- *Prácticas recomendadas para la indexación centrada en los móviles | Centro de la Búsqueda de Google*. Consultado el 20 de junio de 2024. Google For Developers. https://developers.google.com/search/docs/crawling-indexing/mobile/mobile-sites-mobile-first-indexing?hl=es
- *Mobile First Design | What It is, Why & How to Perform?* (2024, 27 febrero). Testsigma Blog. https://testsigma.com/blog/mobile-first-design/
- L. Li and W. Chou, "Design and Describe REST API without Violating REST: A Petri Net Based Approach," 2011 *IEEE International Conference on Web Services*, Washington, DC, USA, 2011, pp. 508-515.
- Sundarbadagala. (2023, 6 noviembre). *Protected Routes in React*. DEV Community. https://dev.to/sundarbadagala081/protected-routes-in-react-47b1
- *¿Qué es un endpoint (punto final) API? | Mailchimp*. Consultado el 20 de junio de 2024. Mailchimp. https://mailchimp.com/es/resources/what-is-an-api-endpoint/
- Diego.Coder. (2024, 10 marzo). Autenticación en Node.js con JSON Web tokens y Express. *Medium*. https://medium.com/@diego.coder/autenticaci%C3%B3n-en-node-js-con-json-web-tokens-y-express-ed9d90c5b579
- *Sequelize*. Consultado el 1 de marzo de 2024. Feature-rich ORM For Modern Type-Script & JavaScript. https://sequelize.org/
- *Express 5.X - API Reference*. Consultado el 1 de marzo de 2024. https://expressjs.com/en/5x/api.html
- *Welcome to Surprise' documentation! — Surprise 1 documentation*. Consultado el 15 de mayo de 2024. https://surprise.readthedocs.io/en/stable/
- *Empezando · jest*. Consultado el 8 de marzo de 2024. https://jestjs.io/es-ES/docs/getting-started
- *npm: supertest*. Consultado el 8 de marzo de 2024. Npm. https://www.npmjs.com/package/supertest
- *IGDB – API*. Consultado el 12 de marzo de 2024. IGDB. https://api-docs.igdb.com/#getting-started
- *Steam Community :: Steam Web API Documentation*. Consultado el 5 de mayo de 2024. https://steamcommunity.com/dev
- Leitch, J. (2021, 23 diciembre). Building a Recipe Recommendation API using Scikit-Learn, NLTK, Docker, Flask, and Heroku. *Medium*. https://towardsdatascience.com/building-a-recipe-recommendation-api-using-scikit-learn-nltk-docker-flask-and-heroku-bfc6c4bdd2d4

Anexo A. Objetivos de Desarrollo Sostenible

En este apartado, se evalúa el grado de relación del proyecto con los Objetivos de Desarrollo Sostenible (ODS) propuestos por la ONU, categorizándolos en niveles de alto, medio, bajo o no aplica.

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

Tabla 18. Relación del proyecto con los ODS