PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

PÁGINA WEB DE RESEÑAS DE DISCOS MUSICALES EN FORMATO FISICO

Proyecto de titulación previo a la obtención del título de: Tecnólogo Superior en Desarrollo de Software

Autores: Kevin Marcillo Gabriel Suntaxi - Cristopher Alexander Puma Guañuna

Tutor: Patricio Omar Alvear Grande

Quito, Ecuador

2024

Tabla de contenidos

Lista de tablas

Lista de figuras

DECLARACIÓN y AUTORIZACIÓN

Yo, **Kevin Gabriel Marcillo Suntaxi** con C.I. 1753927555 autor(a) del trabajo de titulación intitulado: **"PÁGINA WEB DE RESEÑAS DE DISCOS MUSICALES EN FORMATO FISICO"**, previa a la obtención del título de **Tecnólogo De Desarrollo De Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, febrero 2025

KEVIN GABRIEL MARCILLO SUNTAXI

C.I. 1753927555

DECLARACIÓN y AUTORIZACIÓN

Yo, **Cristopher Alexander Puma Guañuna** con C.I. 1726692161 autor(a) del trabajo de titulación intitulado: **"PÁGINA WEB DE RESEÑAS DE DISCOS MUSICALES EN FORMATO FISICO"**, previa a la obtención del título de **Tecnólogo De Desarrollo De Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, febrero 2025

CRISTOPHER ALEXANDER PUMA GUAÑUNA

C.I. 1726692161

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que han sido fundamentales en el desarrollo de este proyecto. Agradecemos, en primer lugar, a nuestras familias y amigos por su constante apoyo emocional y motivación a lo largo de este proceso. Su confianza y comprensión han sido una gran fuente de inspiración para ambos.

Agradecemos también a nuestros profesores y mentores, cuyas enseñanzas y orientaciones nos han permitido adquirir los conocimientos técnicos necesarios para llevar a cabo este proyecto. Su disposición para resolver dudas y brindar consejos siempre fue invaluable para nosotros.

Queremos extender nuestro agradecimiento a nuestros trabajos, los cuales nos brindaron el tiempo libre necesario para poder dedicarle a esta tesis. Sin su flexibilidad, no habría sido posible avanzar en este proyecto de manera tan efectiva.

Finalmente, deseamos agradecer a la comunidad de desarrolladores y las herramientas de código abierto como React, Node.js y PostgreSQL, quienes proporcionaron los recursos y tecnologías que hicieron posible el éxito de esta plataforma. También valoramos las contribuciones de la documentación oficial y las comunidades en línea que han facilitado la resolución de problemas y la implementación de mejores prácticas.

Gracias a todos los que, de alguna manera, contribuyeron al éxito de este proyecto.

Introducción

En el presente proyecto se desarrollará una aplicación web que permitirá a los usuarios dejar reseñas sobre productos físicos de una tienda de música, como CDs y discos de vinilo. Las reseñas pueden ser personales en relación con el producto o a la calidad del sonido, el empaque y, si el producto incluye algún extra (como cancioneros o imágenes promocionales del álbum) que ayuden.

El objetivo principal de esta aplicación es brindar información valiosa al administrador de la tienda sobre los productos que llegan. Aunque existen páginas como Bertus que ofrecen bastante información, estas suelen ser limitadas cuando se trata de productos latinoamericanos (especialmente de Argentina o Chile). Además, la información suele estar en otros idiomas, lo que puede dificultar su comprensión. Las traducciones automáticas, como las de Google Translate, no suelen capturan bien el sentido original, lo que puede generar malentendidos importantes.

Al permitir que usuarios hispanohablantes compartan sus opiniones, se logrará reunir datos clave que ayudarán a la empresa a la toma de decisiones importantes. Por ejemplo, si varios clientes mencionan un defecto de fábrica inesperado, el administrador podrá frenar futuros pedidos de ese artículo y proveedor.

Para construir esta página de reseñas, se utilizará React en el frontend, una biblioteca que facilitará la creación de componentes reutilizables y mejorará la experiencia del usuario con una carga dinámica de contenidos, haciendo que la página sea más rápida y fluida. Además, se empleará JavaScript para gestionar la lógica en el lado del cliente, manejar eventos y realizar solicitudes al servidor de manera asíncrona.

El diseño y la apariencia de la aplicación estarán a cargo de SCSS, asegurando un estilo visual atractivo y adaptado a distintos dispositivos (responsive design).

En cuanto al backend, Node.js será el framework elegido para desarrollar el servidor y las APIs que conectarán la interfaz con la base de datos. A través de una API RESTful, se gestionarán las solicitudes del frontend, como la búsqueda de discos, la creación de reseñas y la calificación de productos. Esta API garantizará una comunicación segura y estandarizada entre el cliente y el servidor.

Los datos se almacenarán en una base de datos relacional, PostgreSQL, ideal para manejar relaciones complejas y consultas estructuradas, como las de discos, reseñas, usuarios y calificaciones.

Finalmente, se implementará un sistema de autenticación para gestionar el inicio de sesión y los permisos de los usuarios. El uso de LocalStorage permitirá mantener la sesión activa durante las pruebas y asegurar una mejor experiencia de usuario.

Con este conjunto de tecnologías, se busca desarrollar una página de reseñas robusta, eficiente y fácil de usar, que no solo facilite la evaluación de los productos musicales, sino que también sirva como una herramienta clave para la toma de decisiones en la tienda.

Capítulo I

1.1 Levantamiento de Requisitos y Diseño del Sistema

1.1.1 Pantalla de Login

<table>
<tr><th>No.</th><th>Nombre Actividad</th><th>Fecha Inicio</th><th>Fecha Fin</th><th>Descripción</th><th>Prioridad</th></tr>
<tr><td rowspan="3">F1</td><td>PANTALLA DE LOGIN</td><td>18-11-2024</td><td>22-11-2024</td><td colspan="2">La pantalla de inicio mostrará un botón de Login en el menú superior donde el usuario llenará los campos respectivos y pueda entrar a la plataforma web.</td><td>Alta.</td></tr>
<tr><td>Entrada</td><td colspan="2">Proceso</td><td colspan="2">Salida</td></tr>
<tr><td>• El usuario ingresara Usuario y contraseña.</td><td colspan="2">• El usuario selecciona el botón de Login.
• Se le desplegará los campos Usuario y Contraseña.
• Una vez llenado los campos, el usuario debe dar al botón de ingresar para que el sistema verifique los datos.</td><td colspan="2">• El usuario podrá acceder al Aplicativo Web y realizar funciones de acuerdo con su Rol.</td></tr>
</table>

Tabla 1 - Pantalla de Login

1.1.2 Registro de Usuarios

<table>
<tr><th>No.</th><th>Nombre Actividad</th><th>Fecha Inicio</th><th>Fecha Fin</th><th>Descripción</th><th>Prioridad</th></tr>
<tr><td rowspan="3">F2</td><td>REGISTRO DE USUARIOS</td><td>23-11-2024</td><td>30-11-2024</td><td>El registro contendrá 4 campos para registrarse y sus datos se guardarán en la base de datos de PostgreSQL.
Los usuarios registrados serán clientes de forma automática.
El nombre de usuario debe ser único, sin posibilidad de repetirse con otro que ya este registrado en el sistema.</td><td>Alta.</td></tr>
<tr><td>Entrada</td><td colspan="2">Proceso</td><td colspan="2">Salida</td></tr>
<tr><td>• El usuario ingresará su nombre de usuario, correo, contraseña y confirmación de contraseña.</td><td colspan="2">• El usuario selecciona el botón de Registro.
• Se le desplegará los campos nombre de usuario, correo, contraseña y confirmación de contraseña.
• Una vez llenado los campos, el usuario debe dar al botón de registrar para que el sistema verifique los datos</td><td colspan="2">• El usuario podrá obtener su Nombre de Usuario y contraseña para ingresar al sistema.</td></tr>
</table>

Tabla 2 - Registro de Usuarios

1.1.3 Registro de Discos

<table>
<tr><th>No.</th><th>Nombre Actividad</th><th>Fecha Inicio</th><th>Fecha Fin</th><th>Descripción</th><th>Prioridad</th></tr>
<tr><td rowspan="3">F3</td><td>REGISTRO DE DISCOS</td><td>02-12-2024</td><td>07-12-2024</td><td colspan="2">El administrador podrá ingresar los discos que desee ser reseñados con la información detallada.
La foto del disco debe ser opcional, debido a que ciertas portadas son nuevas o clasificadas, siendo difícil encontrar buenas imágenes de la portada en cuestión, sino hasta que pasen unos días después del lanzamiento.</td><td>Alta.</td></tr>
<tr><td>Entrada</td><td colspan="2">Proceso</td><td colspan="2">Salida</td></tr>
<tr><td>• El administrador ingresara Título, Artista, Género Musical, Detalles.
• El ingreso de la foto es opcional.</td><td colspan="2">• El administrador al momento de Ingresar al sistema estará en la pantalla de registro de discos.
• El administrador deberá llenar los campos respectivos para poder subir el disco.
• La imagen del disco por defecto estará con una imagen genérica hasta que el administrador pueda subir una mejor imagen.
• Una vez llenos los campos, debe seleccionar el botón “Guardar” para subir el disco al sistema.</td><td colspan="2">• El disco es registrado en el sistema, listo para ser reseñado.</td></tr>
</table>

Tabla 3 - Registro de Discos

1.1.4 Vista de Disco

<table>
<tr><th>No.</th><th>Nombre Actividad</th><th>Fecha Inicio</th><th>Fecha Fin</th><th>Descripción</th><th>Prioridad</th></tr>
<tr><td rowspan="3">F6</td><td>VISTA DE DISCO</td><td>18 - 12- 2024</td><td>23 - 12- 2024</td><td colspan="2">Una vez el usuario seleccione un disco, se le desplegará las reseñas que ya le hayan dado al disco otros usuarios.
En caso de no haber ninguna, se mostrará una frase que alenté al usuario a subir la suya.</td><td>Alta.</td></tr>
<tr><td>Entrada</td><td colspan="2">Proceso</td><td colspan="2">Salida</td></tr>
<tr><td>• El usuario ingresa el nombre del título del disco o el artista.
• El usuario selecciona un disco.</td><td colspan="2">• El usuario selecciona el botón “reseña” en el menú superior.
• El usuario entrará a la pantalla de búsqueda de disco, deberá elegir el disco seleccionado
• Al seleccionar un disco, podrá ver una descripción más completa del disco y las reseñas que otros usuarios hayan hecho con anterioridad.</td><td colspan="2">• El usuario entra a la pantalla de disco, listo para poder reseñarlo.</td></tr>
</table>

Tabla 4 - Vista de Disco

1.1.5 Registro de Reseñas

<table>
<tr><th>No.</th><th>Nombre Actividad</th><th>Fecha Inicio</th><th>Fecha Fin</th><th>Descripción</th><th>Prioridad</th></tr>
<tr><td rowspan="3">F7</td><td>REGISTRO DE RESEÑAS</td><td>03 - 01- 2025</td><td>18 - 01- 2025</td><td>El usuario podrá subir su reseña de un disco especifico al sistema.
Si el usuario desea subir una foto del disco, ya sea para reportar algún daño o verificar que compro el disco, lo podrá hacer, es opcional en todo caso para el resto de los usuarios.</td><td>Alta.</td></tr>
<tr><td>Entrada</td><td colspan="2">Proceso</td><td colspan="2">Salida</td></tr>
<tr><td>• El usuario ingresará su reseña y calificación en base a estrellas en el sistema.
• El ingreso de la foto es opcional</td><td colspan="2">• El usuario selecciona el botón de “reseña”.
• El usuario entra a la pantalla de vista de discos.
• Una vez seleccionado el disco, se le mostrará los discos registrados que coincidan con su búsqueda, el usuario debe seleccionar un disco.
• Al seleccionar un disco se le mostrará una descripción del disco, reseñas anteriores y el apartado para realizar reseñas.
• El usuario de llenar el un apartado para calificar, escribir la reseña.
• El usuario debe seleccionar el botón de “guardar” para que su reseña se suba al sistema.</td><td colspan="2">• La reseña del disco estará subida al sistema junto al disco.</td></tr>
</table>

Tabla 5 - Registro de Reseñas

1.2 Diseño del Sistema

1.2.1 Diagrama E/R

El diagrama nos ayuda a ver el tipo de relación que tienen las tablas entre sí para que funcione el sistema web.

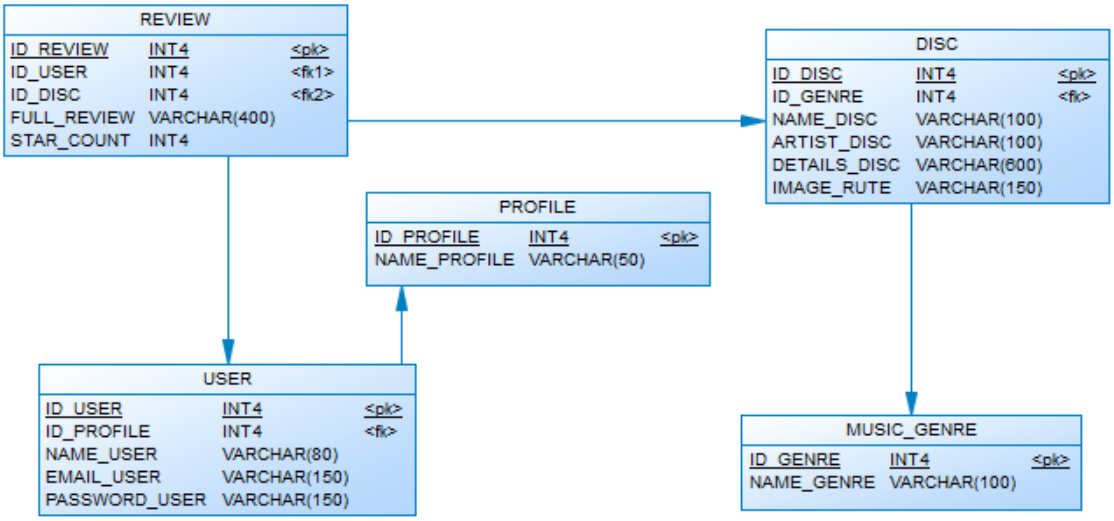

Ilustración 1- Diagrama E/R del sistema Web

1.2.2 Diseño de Mockups

El mockup de registro indica los principales datos que serán recogidos de los Usuarios y que serán guardados en la base de datos

Ilustración 2 - Mockup Pantalla de registro

El mockup de login indica los principales campos que serán verificados en la base de datos para poder dar inicio de sesión en la cuenta

Ilustración 3 - Mockup de login

El mockup de registro de discos muestra los campos que se requieren a la hora de registrar un disco dentro de la página web.

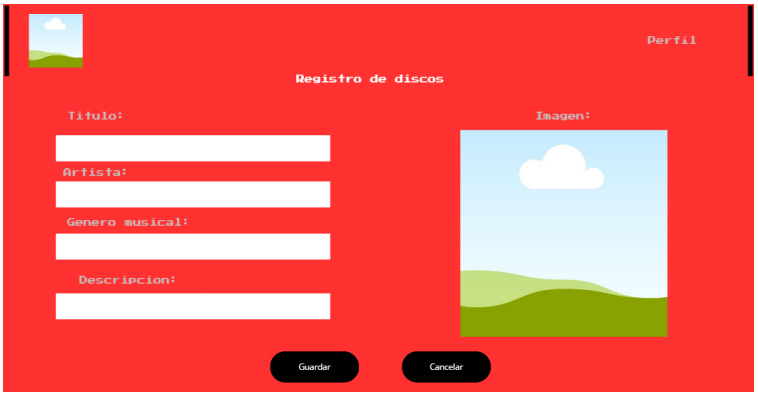

Ilustración 4 - Mockup Registro de discos

El mockup de registro de reseñas y ranking será el apartado dónde le usuario podrá escribir sus comentarios del disco de música al igual que dejar su reseña con un formato de estrellas al igual que podrá subir una fotografía del producto a la página.

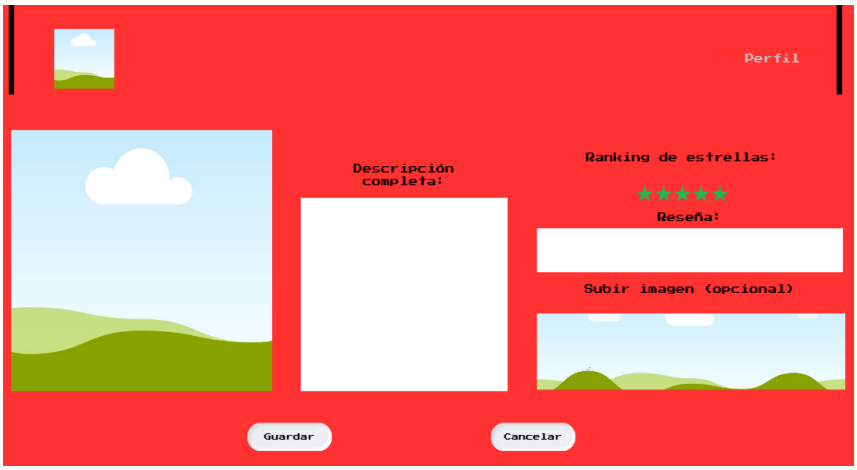

Ilustración 5 - Mockup Registro de reseñas y ranking

El mockup de vista de discos y reseñas será un apartado para observar el disco y reseñas anteriores en el caso que existan.

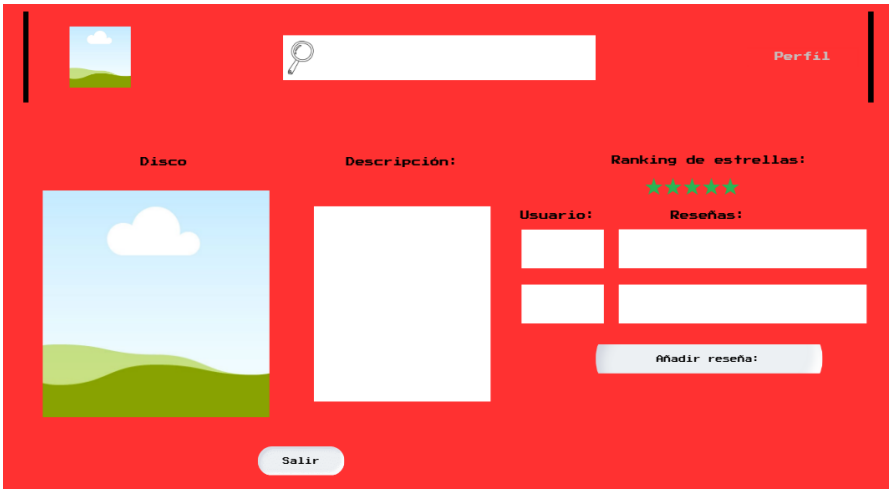

Ilustración 6 - Mockup vista de disco y reseñas

1.2.3 Diagrama de Casos de Usos

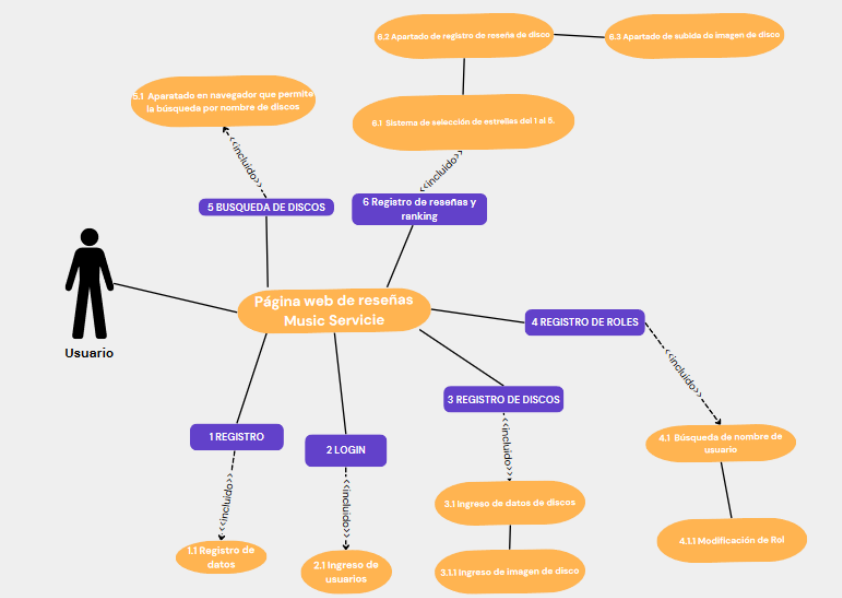

Ilustración 7 - Casos de uso Generales

Capítulo II

2.1 Construcción del Sistema FrontEnd

2.1.1 Estándares de Construcción

Se ha adoptado el uso de React.js para el desarrollo del frontend, empleando la estructura de componentes reutilizables y la gestión del estado mediante Context API. La arquitectura sigue las buenas prácticas de modularidad para separar la lógica y la presentación.

2.1.2 Estructura del Proyecto

El frontend del sistema se compone de los siguientes componentes:

1. **Contexto**:
    - UserContext.jsx: Administra el contexto de autenticación y la información del usuario dentro de la aplicación, permitiendo compartir el estado de inicio de sesión entre múltiples componentes sin necesidad de pasar props.
2. **Componentes de la Aplicación**:
    - App.jsx: Punto de entrada principal de la aplicación donde se configuran las rutas para la navegación entre las páginas.
    - main.jsx: Archivo donde se monta la aplicación React y se integra con la raíz del documento HTML.
3. **Componentes de Interfaz de Usuario**:
    - Login_page.jsx: Gestiona la interfaz de inicio de sesión, capturando las credenciales de usuario y autentica con el servidor.

- Register_page.jsx: Permite a los usuarios crear una nueva cuenta proporcionando sus datos personales.
- Navbar_main.jsx: Barra de navegación que muestra enlaces a las secciones principales de la aplicación, incluyendo el nombre del usuario autenticado.
- Main_page.jsx: Página de inicio que proporciona una vista general de las funcionalidades disponibles.
- cards_preview.jsx: Muestra una vista previa de los discos y permite acceder a más detalles.
- Register_disc.jsx: Componente para registrar nuevos discos en la base de datos, incluyendo sus detalles y la imagen asociada.
- Review.jsx: Permite ver y escribir reseñas de discos, incluyendo la asignación de calificaciones.

4. **Estilos**:
    - Los estilos personalizados se implementan utilizando archivos. scss para mantener una separación clara entre la lógica de los componentes y su apariencia.

2.1.3 Gestión de Estado

La gestión del estado global se maneja con Context API, lo que permite que la información del usuario autenticado esté disponible en cualquier parte de la aplicación sin necesidad de props drilling.

2.1.4 Uso de Rutas

Uso de Rutas

Se emplea react-router-dom para gestionar la navegación entre las páginas, garantizando una experiencia fluida para el usuario.

Esta estructura modularizada facilita la extensibilidad y el mantenimiento del frontend, asegurando un desarrollo sostenible y escalable.

2.1.5 Interacción de Bibliotecas y Herramientas Front-end (React)

El sistema front-end se basa en la biblioteca **React** para construir una interfaz de usuario dinámica e interactiva. La estructura incluye componentes reutilizables, que permiten mantener el código modular y eficiente.

1. **React**:
    - Responsable de crear interfaces de usuario declarativas.
    - Facilita la renderización de componentes dinámicos en función del estado.
2. **react-router-dom**:
    - Permite la navegación entre páginas sin recargar la aplicación.
    - Implementa rutas como /login o /review para una experiencia fluida de usuario.
3. **react-icons**:
    - Proporciona íconos predefinidos para enriquecer la interfaz gráfica.

- Usado para mejorar la experiencia visual con iconos de menú, cierre, etc.

4. **axios**:
    - Utilizado para hacer solicitudes HTTP hacia el servidor Node.js.
    - Maneja las operaciones de POST y GET para el login, registro de usuarios y obtención de datos de discos.
5. **bootstrap** y **react-bootstrap**:
    - Simplifican la creación de componentes estilizados y responsivos.
    - Garantizan un diseño visual coherente y adaptado a múltiples resoluciones.
6. **sass-embedded** y **sass**:
    - Sass permite la creación de estilos con variables, mixins y anidaciones.
    - Facilita el mantenimiento de los estilos mediante un código más limpio y organizado.

2.2 Construcción del Sistema Backend

2.2.1 Estándares de Construcción

Para el desarrollo del sistema se implementó la metodología **MVC** (Modelo-Vista-Controlador)

Es una arquitectura de software que organiza la aplicación en tres componentes principales: el **Modelo**, encargado de gestionar los datos y la lógica de la aplicación; la **Vista**, que se ocupa de la presentación e interfaz de usuario; y el **Controlador**, que actúa como intermediario entre el Modelo y la Vista, manejando las interacciones del usuario y actualizando los demás componentes según sea necesario. Esta separación facilita el mantenimiento del sistema, mejora su organización y promueve una mayor estabilidad a lo largo del tiempo.

2.2.2 Definición de Modelos

En el sistema se implementan los siguientes modelos:

- **USER**
- **DISC**
- **REVIEW**
- **MUSIC_GENRE**
- **PROFILE**

Modelo USER

Este modelo representa a los usuarios registrados en el sistema y contiene los siguientes atributos:

- **ID_USER:** Identificador de Usuario como clave primaria.
- **ID_PROFILE:** Relación con la tabla de perfiles de usuario.
- **NAME_USER:** Nombre del usuario.
- **EMAIL_USER:** Correo electrónico del usuario.
- **PASSWORD_USER:** Contraseña del usuario (almacenada de manera segura).

Modelo DISC

Este modelo almacena la información sobre los discos:

- **ID_DISC:** Identificador único del disco.
- **ID_GENRE:** Relación con la tabla de géneros musicales.
- **NAME_DISC:** Nombre del disco.
- **ARTIST_DISC:** Nombre del artista o banda.
- **DETAILS_DISC:** Descripción o detalles del disco.
- **IMAGE_RUTE:** Ruta de la imagen de la carátula del disco.

Modelo REVIEW

Este modelo contiene las reseñas de los discos:

- **ID_REVIEW:** Identificador único de la reseña.

- **ID_USER:** Relación con la tabla de usuarios.
- **ID_DISC:** Relación con el disco reseñado.
- **FULL_REVIEW:** Texto completo de la reseña.
- **STAR_COUNT:** Calificación en forma de estrellas.

Modelo MUSIC_GENRE

Este modelo define los géneros musicales disponibles:

- **ID_GENRE:** Identificador del género musical.
- **NAME_GENRE:** Nombre del género.

Modelo PROFILE

Este modelo define los tipos de perfiles de usuario:

- **ID_PROFILE:** Identificador del perfil.
- **NAME_PROFILE:** Nombre del perfil (Administrador o Usuario normal).

2.2.3 Definición de Controladores

Los controladores son responsables de manejar las solicitudes entrantes, procesar los datos y devolver respuestas adecuadas. Los principales controladores utilizados son:

- **userController.js:** Maneja las operaciones relacionadas con los usuarios, como registro, inicio de sesión y cierre de sesión.
- **discController.js:** Gestiona las operaciones sobre los discos, incluyendo listar, crear y buscar discos.

- **reviewController.js:** Procesa las solicitudes relacionadas con las reseñas de discos.
- **authMiddleware.js:** es un middleware que verifica la autenticación de los usuarios en el backend. Valida el token de autorización enviado en las solicitudes, permitiendo el acceso solo a usuarios autenticados y devolviendo un error en caso de token inválido o ausente.

2.2.4 Vistas

El sistema backend está diseñado para integrarse con un frontend desarrollado en React. Las vistas son representaciones de las respuestas enviadas desde los controladores para ser mostradas en la interfaz de usuario, utilizando JSON como formato de intercambio de datos.

La estructura MVC permite que cada componente sea desarrollado y mantenido de forma independiente, promoviendo una mejor organización y extensibilidad del sistema.

2.2.5 Interacción de Bibliotecas y Herramientas

2.2.5.1 Back-end (Node.js y Express)

El servidor back-end se desarrolla con Node.js y Express para manejar solicitudes del cliente, gestionar la lógica del negocio y conectar con la base de datos PostgreSQL.

1. **express:**
    - Framework minimalista para construir aplicaciones web.
    - Define rutas para manejar las operaciones CRUD relacionadas con usuarios, discos y reseñas.

2. **pg (node-postgres):**
    - Biblioteca para interactuar con la base de datos PostgreSQL.
    - Ejecuta consultas SQL para manejar registros de usuarios, discos, géneros y reseñas.
3. **bcrypt:**
    - Utilizado para el hashing de contraseñas.
    - Protege las credenciales de usuario antes de almacenarlas en la base de datos.
4. **jwt:**
    - El token se almacena generalmente en el almacenamiento local
    - Genera y verifica tokens de acceso que permiten a los usuarios autenticarse y acceder a recursos protegidos en la aplicación.
5. **multer:**
    - Permite la carga de archivos, como imágenes de portada para los discos.
6. **cors:**
    - Permite el intercambio de recursos entre dominios (CORS).
    - Habilita la comunicación segura entre el cliente (React) y el servidor (Express).
7. **dotenv:**

- Maneja variables de entorno para almacenar datos confidenciales como claves de la base de datos.

2.2.6 Flujo de Interacción

1. **Autenticación y Sesiones:**
   - El cliente envía credenciales mediante un formulario (manejado con axios).
   - El servidor valida las credenciales, inicia una sesión con express-session y devuelve un archivo guardado en el LocalStorage.
   - LocalStorage se almacena en el navegador para mantener la sesión activa.
2. **Gestión de Datos:**
   - Los datos de discos, usuarios y reseñas se consultan y actualizan mediante axios en el cliente y pg en el servidor.
   - Las consultas SQL están protegidas para prevenir inyecciones de código.
3. **Carga de Imágenes:**
   - Multer procesa las imágenes cargadas y las guarda en el servidor, actualizando la base de datos con las rutas.

Capítulo III

3. Pruebas y Despliegues

3.1 Pruebas Funcionales FrontEnd

N° 1 Caso de prueba Registro de Usuario

DESCRIPCIÓN DEL CASO DE PRUEBA **Se prueba la funcionalidad del Login.**	
Precondiciones	• **Deben existir usuarios en la base de datos.** • **Ingresar datos correctos en los campos de Usuario, Correo y Contraseña.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**Hacer click en el botón de Registro en el menú principal. Llenar los datos solicitados de "Usuario", "Correo" y "Contraseña" y "Confirmar contraseña". Hacer click en el botón "Registrarse"**
Datos de Entrada	• **Usuario** • **Correo** • **Contraseña**
Resultados Esperados	**Autenticación e ingreso al sistema.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: La autenticación de datos fue correcta y se registro el usuario al sistema** **RECHAZO: Mensaje de error en Registro de datos.**
Desarrollador Asignado	**Christopher Puma.**

Tabla 6 - N° 1 Caso de prueba Registro de Usuario

N° 2 Caso de prueba Login

DESCRIPCIÓN DEL CASO DE PRUEBA **Se prueba la funcionalidad del Login.**	
Precondiciones	• **Deben existir usuarios en la base de datos.** • **Ingresar datos correctos en los campos de Correo y Contraseña.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**Hacer click en el botón de Login en el menú principal. Ingresar el correo y contraseña correctos. Posteriormente se debe dar click al botón "Iniciar Sesión", este validará los datos e ingresará al usuario a la pantalla de inicio, con el nombre de usuario reemplazando al botón de Login/Registro.**
Datos de Entrada	• **Correo** • **Contraseña**
Resultados Esperados	**Autenticación e ingreso al sistema.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: La autenticación de datos fue correcta y se ingresó al sistema** **RECHAZO: Mensaje de error en ingreso de Login.**
Desarrollador Asignado	**Christopher Puma.**

Tabla 7 - N° 2 Caso de prueba Login

N° 3 Caso de prueba Función Especial Barra de Menú

DESCRIPCIÓN DEL CASO DE PRUEBA **Se prueba la funcionalidad de la barra de Menú.**	
Precondiciones	• **Deben existir usuarios en la base de datos.** • **Estar Logeado en el sistema.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**La barra de menú mostrará tres opciones. Inicio, Reseña y botón de Registro/Logeo. Una vez el usuario complete el proceso de Logeo, la barra mostrará un botón adicional llamado Discos.**
Datos de Entrada	• **No aplica**
Resultados Esperados	**Mostrar el botón de Discos una vez el usuario complete su Logeo.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El sistema detecta que el usuario ingreso al sistema y muestra el botón.** **RECHAZO: El sistema no detecto al usuario y el botón no se muestra.**
Desarrollador Asignado	**Christopher Puma.**

Tabla 8 - N° 3 Caso de prueba Función Especial Barra de Menú

N° 4 Caso de prueba Registrar Discos

DESCRIPCIÓN DEL CASO DE PRUEBA Se prueba la funcionalidad de la Reseña.	
Precondiciones	• **Ingresar datos correctos en los campos de Título del disco, Artista, Género Musical, Descripción e Imagen del Disco.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**Hacer Click en el botón Discos que se encuentra en el Navbar, luego llegar los campos correspondientes con datos correctos y luego aplastar el botón guardar para enviar todo a la base de datos o cancelar si ya no desea continuar.**
Datos de Entrada	• **Título del disco** • **Artista** • **Género Musical** • **Descripción** • **Imagen del producto**
Resultados Esperados	**Disco guardado y aparición dentro del área de reseñas.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El disco hizo su aparición de manera correcta.** **RECHAZO: Aparición de mansaje de error y no hace su aparición el disco.**
Desarrollador Asignado	**Cristopher Puma.**

Tabla 9 - N° 4 Caso de prueba Registrar Discos

N° 5 Caso de prueba Envío de reseña

DESCRIPCIÓN DEL CASO DE PRUEBA **Se prueba la funcionalidad de la Reseña.**	
Precondiciones	• **Deben existir discos en la base de datos.** • **Ingresar datos correctos en los campos de Reseña y Elegir Cantidad de estrellas.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**Hacer click al botón de reseñas. Elegir el disco a reseñar. Después se debe Rellenar la información de Reseña y Cantidad de Estrellas. Posteriormente se debe dar click al botón "guardar", este validará los datos e ingresará la reseña. Se reenvía al usuario a la pantalla de inicio con la reseña mostrándose en un apartado específico.**
Datos de Entrada	• **Reseña** • **Cantidad de Estrellas**
Resultados Esperados	**Reseña subida y desplegada en el sistema.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: La reseña fue validada y subida al sistema.** **RECHAZO: En caso de no llenar datos o no haber ingresado al sistema, se mostrará un mensaje emergente impidiendo el registro vacio.**
Desarrollador Asignado	**Christopher Puma.**

Tabla 10 - N° 5 Caso de prueba Envío de reseña

N° 6 Caso de prueba Verificación de roles

DESCRIPCIÓN DEL CASO DE PRUEBA **Se prueba la funcionalidad de los roles en él sistema.**	
Precondiciones	• **Deben existir usuarios en la base de datos.** • **Estar Logeado en el sistema.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**Hacer click en el botón de Login en el menú principal. Ingresar el correo y contraseña correctos. Posteriormente se debe dar click al botón "Iniciar Sesión", este validará los datos e ingresará al usuario a la pantalla de inicio. Debe mostrarse el botón de Discos dependiendo de si el usuario es Administrador o No.**
Datos de Entrada	• **Correo** • **Contraseña**
Resultados Esperados	**El sistema debe reconocer el rol del usuario registrado.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El sistema reconoce el rol del Usuario y muestra o no el botón de Discos.** **RECHAZO: El sistema no muestra el botón y el nombre del usuario no aparece en el botón.**
Desarrollador Asignado	**Cristopher Puma.**

Tabla 11 - N° 6 Caso de prueba Verificación de roles

N° 7 Caso de prueba Sin Reseñas en el sistema

DESCRIPCIÓN DEL CASO DE PRUEBA **Se prueba la funcionalidad de detectar que no hay reseñas en él sistema.**	
Precondiciones	• **No deben existir registros de reseñas.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**El usuario debe entrar al menú de inicio.** **Otra opción es elegir un disco registrado y ver si hay reseñas anteriores.**
Datos de Entrada	• **No aplica**
Resultados Esperados	**El sistema debe reconocer que no hay reseñas y mostrar un mensaje que lo confirme**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El sistema no detecta ningún registro y muestra un mensaje que lo confirme.** **RECHAZO: El sistema muestra un mensaje de cargando reseñas de manera interminable. Debe haber un problema con la conexión de la base de datos.**
Desarrollador Asignado	**Cristopher Puma.**

Tabla 12 - N°7 Caso de prueba Sin Reseñas en el sistema

N° 8 Caso de prueba Sin Discos en el sistema

DESCRIPCIÓN DEL CASO DE PRUEBA **Se prueba la funcionalidad de detectar que no hay discos en él sistema.**	
Precondiciones	• **No deben existir registros de discos.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**El usuario debe entrar a la sección de reseñas**
Datos de Entrada	• **No aplica**
Resultados Esperados	**El sistema debe reconocer que no hay discos y mostrar un mensaje que lo confirme**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El sistema no detecta ningún disco y muestra un mensaje que lo confirme.** **RECHAZO: El sistema muestra un mensaje de cargando discos de manera interminable. Debe haber un problema con la conexión de la base de datos.**
Desarrollador Asignado	**Cristopher Puma.**

Tabla 13 - N° 8 Caso de prueba Sin Discos en el sistema

N° 9 Caso de prueba Disco Subido sin imagen

DESCRIPCIÓN DEL CASO DE PRUEBA **Se prueba la funcionalidad de usar la imagen predeterminada en caso de que no se suba una imagen al crear un disco**	
Precondiciones	• **Ingresar datos correctos en los campos de Título del disco, Artista, Género Musical, Descripción e Imagen del Disco.** • **Debe estar desplegado Front y Backend.**
Pasos para seguir	**Hacer Click en el botón Discos que se encuentra en el Navbar, luego llegar los campos correspondientes con datos correctos y luego aplastar el botón guardar para enviar todo a la base de datos o cancelar si ya no desea continuar.**
Datos de Entrada	• **Título del disco** • **Artista** • **Género Musical** • **Descripción**
Resultados Esperados	**Disco guardado y aparición dentro del área de reseñas.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El disco hizo su aparición de manera correcta con la portada por defecto.** **RECHAZO: Aparición de mansaje de error y no hace aparición el disco.**
Desarrollador Asignado	**Cristopher Puma.**

Tabla 14 - N° 9 Caso de prueba Disco Subido sin imagen

3.2 Despliegue FrontEnd

Pasos para desplegar el Frontend de manera Local:

1. Descargar o verificar que **Node.js** esté instalado en la máquina (requisito esencial para manejar paquetes npm).
2. Descargar o verificar que Visual Studio Code este instalado en la máquina.
3. Clonar o descargar la carpeta del proyecto desde Github.
4. Abrir la carpeta con Visual Studio Code.
5. Abrir la terminal integrada o la consola de PowerShell y verificar que esté ubicada en la carpeta raíz del proyecto.
6. Instalar las bibliotecas y dependencias mediante la consola usando "*npm install"* siguiendo los pasos del README.md, en la sección de FrontEnd.
7. Resolver cualquier conflicto de instalación de bibliotecas según sea necesario.
8. Ejecutar "npm run dev" para desplegar el frontEnd Localmente

3.3 Pruebas Unitarias BackEnd.

N° 1 Caso de prueba API User/ Método GET Usuarios.

DESCRIPCIÓN DEL CASO DE PRUEBA **El método para esta API retoma la información del usuario (Rol, Usuario, Correo y Contraseña)**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Realizar una petición GET a la API de User (/api/user/login).**
Datos de Entrada	• **ID del Usuario.**
Resultados Esperados	**Método GET debe devolver la información del usuario en base al ID.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El método devolvió la información solicitada** **RECHAZO: No devuelve información.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 15- N° 1 Caso de prueba API User/ Método GET Usuarios.

N° 2 Caso de prueba API Registro/ Método POST Usuarios.

DESCRIPCIÓN DEL CASO DE PRUEBA **El método para esta API envía información valida del usuario (Rol, Usuario, Correo y Contraseña)**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Realizar una petición POST a la API de User (/api/user/register).**
Datos de Entrada	• **Rol (Por defecto el sistema lo tiene en 2)** • **Usuario** • **Correo** • **Contraseña**
Resultados Esperados	**Método POST debe registrar al usuario con los datos validados.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El método envió la información validada del usuario a la base de datos, y se inserta en la tabla de "User"** **RECHAZO: No se inserta la información, devuelve error 404.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 16 - N° 2 Caso de prueba API Registro/ Método POST Usuarios.

N° 3 Caso de prueba API Discos/ Método GET Discos.

DESCRIPCIÓN DEL CASO DE PRUEBA **El método para esta API retoma la información del Disco (id del género musical, Nombre del Disco, Nombre del Artista, Detalles del Disco y Ruta de imagen)**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Realizar una petición GET a la API de Discos (/api/discs).**
Datos de Entrada	• **ID del Disco.**
Resultados Esperados	**Método GET debe devolver la información del Disco en base al ID.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El método devolvió la información solicitada.** **RECHAZO: No Devuelve 41nformación.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 17 - N° 3 Caso de prueba API Discos/ Método GET Discos.

N° 4 Caso de prueba API Discos/ Método GET Reseñas.

DESCRIPCIÓN DEL CASO DE PRUEBA **El método para esta API retoma información de la reseña (Id del disco, reseña, Contador de estrellas)**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Realizar una petición GET a la API de Reseñas (/api/review).**
Datos de Entrada	• **ID de la reseña.**
Resultados Esperados	**Método GET debe devolver la información de la reseña en base al ID.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El método devolvió la información solicitada** **RECHAZO: No Devuelve informacion.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 18 - N° 4 Caso de prueba API Discos/ Método GET Reseñas.

N° 5 Caso de prueba API Reseñas/ Método POST Reseñas.

DESCRIPCIÓN DEL CASO DE PRUEBA **El método para esta API envía información valida de la reseña (Id del disco, reseña, Contador de estrellas)**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Realizar una petición POST a la API de Reseña (/api/review).**
Datos de Entrada	• **Reseña** • **Contador de Estrellas**
Resultados Esperados	**Método POST debe enviar la información de la reseña a la base de datos.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: El método envió la información validada de la reseña a la base de datos y la inserta a la tabla de "Review".** **RECHAZO: No ingresa la información, recibe Error 404.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 19 - N° 5 Caso de prueba API Reseñas/ Método POST Reseñas.

N° 6 Caso de prueba API Registro/ Método POST Usuarios duplicado con correo.

DESCRIPCIÓN DEL CASO DE PRUEBA **El método para esta API envía información duplicada del usuario para verificar si el usuario ya está registrado (Correo)**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Realizar una petición GET a la API de Reseñas (/api/review).**
Datos de Entrada	• **Rol (Por defecto el sistema lo tiene en 2)** • **Usuario** • **Correo (Enviar uno ya registrado)** • **Contraseña**
Resultados Esperados	**Método POST debe reconocer que el correo ya está registrado en él sistema.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: No aplica.** **RECHAZO: No ingresa la información al reconocer el correo en la base de datos. Muestra mensaje de enviar otro correo.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 20 -N° 6 Caso de prueba API Registro/ Método POST Usuarios duplicado con correo.

N° 7 Caso de prueba API Registro/ Método POST Usuarios duplicado con usuario.

DESCRIPCIÓN DEL CASO DE PRUEBA **El método para esta API envía información duplicada del usuario para verificar si el usuario ya está registrado (Usuario)**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Realizar una petición GET a la API de Reseñas (/api/review).**
Datos de Entrada	• **Rol (Por defecto el sistema lo tiene en 2)** • **Usuario (Enviar uno ya registrado)** • **Correo** • **Contraseña**
Resultados Esperados	**Método POST debe reconocer que el usuario ya está registrado en él sistema.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: No aplica.** **RECHAZO: No ingresa la información al reconocer el usuario en la base de datos. Muestra mensaje de usar otro usuario.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 21 - N° 7 Caso de prueba API Registro/ Método POST Usuarios duplicado con usuario.

N° 8 Caso de prueba API Login/ Método GET Usuarios, no existe usuario.

DESCRIPCIÓN DEL CASO DE PRUEBA **El método para esta API no retorna información si el usuario no está registrado en el sistema, verificado por el email.**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Realizar una petición GET a la API de User (/api/user/login).**
Datos de Entrada	• **Correo del Usuario.**
Resultados Esperados	**Método GET debe reconocer que el usuario no está registrado en él sistema.**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: No aplica.** **RECHAZO: No retorna información. Muestra mensaje de correo no encontrado.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 22- N° 8 Caso de prueba API Login/ Método GET Usuarios, no existe usuario.

N° 9 Caso de prueba API Limite Métodos GET

DESCRIPCIÓN DEL CASO DE PRUEBA **Se realizará una prueba de carga con 10,000 solicitudes GET en 1 segundo utilizando Apache JMeter para evaluar la capacidad de respuesta del servidor en la consulta de reseñas.**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Configurar Apache JMeter con:** • **Número de hilos (usuarios virtuales): 10,000** • **Período de subida (Ramp-Up Period): 1 segundo** • **Sampler: HTTP Request → Método GET** • **Ruta: /api/review** **Ejecutar la prueba y monitorear la respuesta del servidor.**
Datos de Entrada	**Ninguno**
Resultados Esperados	**El método GET debe devolver la lista de reseñas almacenadas en la base de datos.** **La API no debe colapsar ni generar errores de servidor (5xx).**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: No aplica.** **RECHAZO: No retorna información.** **Muestra errores a partir de 6.000 solicitudes.** **El servidor se bloquea o deja de responder.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 23 - N° 9 Caso de prueba API Limite Métodos GET

N° 10 Caso de prueba API Limite Métodos POST

DESCRIPCIÓN DEL CASO DE PRUEBA **Se realizará una prueba de carga con 10,000 solicitudes POST en 1 segundo utilizando Apache JMeter para evaluar la capacidad de respuesta del servidor en la subida de reseñas.**	
Precondiciones	• **La base de datos debe estar disponible y accesible.** • **El servidor de la API debe estar en ejecución.**
Pasos para seguir	**Configurar Apache JMeter con:** • **Número de hilos (usuarios virtuales): 10,000** • **Período de subida (Ramp-Up Period): 1 segundo** • **Sampler: HTTP Request → Método POST** • **Ruta: /api/review** **Ejecutar la prueba y monitorear la respuesta del servidor.**
Datos de Entrada	**Ninguno**
Resultados Esperados	**El método POST debe subir los datos de reseñas** **La API no debe colapsar ni generar errores de servidor (5xx).**
Criterios de Aceptación / Rechazo	**ACEPTACIÓN: No aplica.** **RECHAZO: No retorna información.** **Muestra errores a partir de 2.500 solicitudes.** **El servidor se bloquea o deja de responder temporalmente.**
Desarrollador Asignado	**Kevin Marcillo.**

Tabla 24 - N° 10 Caso de prueba API Limite Métodos POST

3.4 Despliegue Backend

Pasos para desplegar el Backend de manera Local:

1. Descargar o verificar que **Node.js** esté instalado en la máquina (requisito esencial para manejar paquetes npm).
2. Descargar o verificar que Visual Studio Code este instalado en la máquina.
3. Descargar o verificar que PostgreSQL y pgadmin 4 este instalado en la máquina (Requisito para la funcionalidad de la base de datos)
4. Clonar o descargar la carpeta del proyecto desde Github.
5. Abrir pgadmin 4 y crear la base de datos: MUSICSERVICE_REVIEW_SITE
6. Hacer click derecho en la base y abrir un query
7. Copiar los comandos de creación de la base desde el archivo Readme.md del proyecto
8. Abrir la carpeta con Visual Studio Code.
9. Abrir la terminal integrada o la consola de PowerShell y verificar que esté ubicada en la carpeta **server** del proyecto.
10. Instalar las bibliotecas y dependencias mediante la consola usando ”*npm install”* siguiendo los pasos del README.md, en la sección de Backend.
11. Resolver cualquier conflicto de instalación de bibliotecas según sea necesario.
12. Ejecutar “npm run” devpara desplegar el Backend Localmente

Recomendaciones:

Asegurar que las credenciales de la base de datos (usuario, contraseña y nombre) en la configuración de tu backend (archivo db.js dentro la carpeta config) coincidan con las que usaste en PostgreSQL.

Conclusiones

- El sitio web dedicado a reseñas de discos musicales no solo ofrece un inventario bien estructurado de los discos, sino que también prioriza la seguridad en el manejo de la información de los usuarios que desean dejar su reseña. La implementación de JWT (JSON Web Tokens) para la autenticación de usuarios garantiza una experiencia segura y confiable, permitiendo que los usuarios puedan iniciar sesión y naveguen por la plataforma sin preocupaciones por el robo de algún dato personal.

- Al usar JWT, se puede almacenar de forma segura la sesión del usuario sin la necesidad de mantener información sensible en el servidor, lo que reduce en su gran mayoría las vulnerabilidades del sistema.

- Además, existe la integración de tecnologías como React en el área del Front-end y Node.js en el área de Back-end permitiendo construir una plataforma modular y eficiente, capaz de manejar grandes volúmenes de datos. El uso de arquitectura MVC en el Back-end asegura una separación más organizada y clara, lo que facilita la escalabilidad del sistema y el mantenimiento a largo plazo. La interacción con la base de datos PostgreSQL permitirá el almacenamiento ya sea del inventario de discos como las reseñas de los usuarios, brindado una mejor experiencia de fluidez y organización.

- Como última instancia, se incorporaría la herramienta React Router Dom para el proceso de navegación por la plataforma, Axios para la recepción de solicitudes HTTP y Context API para la gestión del estado global, la plataforma será altamente interactiva y fácil de usar. Los usuarios tendrían la posibilidad de

consultar discos, impregnar sus reseñas y calificar productos de manera más ordenada y simple, mientras que los administradores poseerán la opción de gestionar inventario y obtener estadísticas sobre la satisfacción de los clientes. Esta combinación de seguridad, tecnología moderna y experiencia de usuarios robusta asegura que la plataforma no solo sea efectiva, sino también segura, escalable y sencilla.

Recomendaciones

- **Optimización del Rendimiento**: A medida que el sitio web continua con el debido proceso de crecimiento, es importante optimizar la carga de datos para garantizar una experiencia fluida para los usuarios. Utilizar técnicas como la paginación en las listas de discos o lazy loading para el proceso de carga de reseñas solo cuando el usuario desee realizar Scroll down puede ayudar a mejorar el rendimiento de la plataforma.
- **Fomentar la interacción Social**: Además de las reseñas y calificaciones, se podría agregar una funcionalidad para que los usuarios puedan compartir señas en redes sociales o recomendar discos a otros posibles usuarios - clientes. Esto no solo aumentaría la interacción usuario – plataforma, sino que también podría ayudar a generar tráfico hacia el sitio web.
- **Monitoreo y Análisis de Datos**: Implementación de un sistema de análisis de datos para monitorear las reseñas y la satisfacción del cliente puede favorecer de manera valiosa la información sobre dichos discos que sean más populares o géneros musicales que se encuentren en tendencia. Esto permitiría ajustar el inventario y las estrategias de marketing en base a las preferencias de los usuarios.

Referencias bibliográficas

- Banks, A. &. (2017). *Learning React: Functional Web Development with React and Redux.* Sebastopol, CA.: O'Reilly Media.
- Haverbeke, M. (2018). *Eloquent JavaScript: A Modern Introduction to Programming (3rd ed.).* San Francisco, CA.: No Starch Press.
- Jin, S. (2019). *API Security in Action.* New York, EE.UU.: Manning Publications. .
- Mariuxi Paola Zea Ordóñez, J. R. (2017). *Administración de bases de datos con PostgreSQL.* Alcoy, España: 3Ciencias.
- Sierra, K. &. (2020). *Head First Programación Orientada a Objetos.* Madrid, España.: Anaya/O'Reilly Media.
- Simmons, C. (2018). *Sass and Compass in Action.* New York, USA.: Manning Publications.

Anexos

Repositorio del proyecto:
https://github.com/kgmarcillo/proyect-reviewpage-musicservice