



Desarrollo full stack

Avance de Proyecto Final

Nombre: Frida Sarahi Garza Gálvez Pedro Francisco De León Salazar Paola Guadalupe Urdiales Luna Abraham Isaí Garza Sánchez Esteban Eliud Arguijo Ramírez	Matricula: al07002961 al03103352 al03107338 al03109910 al07021734
Profesor: Erik Bethuel Estrada Andrade	
Fecha: 03 de Febrero de 2025	

DESCRIPCION

En la primera fase del reto, los aprendedores deberán crear la base de una aplicación *full stack* con un enfoque general y de interés amplio (puede estar orientada a sectores como el educativo, automotriz, financiero, industrial, etc.), y centrarse en el desarrollo del *frontend* y *backend* utilizando Node.js y Express.js. Esta fase incluye la configuración del servidor, la implementación de rutas y *middleware*, y la gestión de operaciones CRUD en una base de datos. Además, se espera la creación de un diseño básico de interfaz utilizando HTML, CSS y JavaScript.

OBJETIVO

Configurar el entorno de desarrollo y los módulos iniciales. Desarrollar una estructura básica del frontend con HTML, CSS y JavaScript, así como un servidor backend en Node.js y Express que permita realizar operaciones CRUD en una base de datos e implemente autenticación básica mediante JWT. Finalmente, desplegar una versión preliminar de la aplicación en un entorno local o en la nube.

INTRODUCCION

En este trabajo esperamos ofrecer al usuario una versión mejorada de la pagina realizada en la actividad 2, esto con el fin de mejorar la experiencia, realizamos cambios enfocados en la accesibilidad y en la privacidad, dandonos como resultado apartados para cambiar de usuario, junto con un login, para mantener su cuenta, al mismo tiempo se podrá editar los datos ingresados por dicho usuario. Esperamos utilizar HTML, JavaScript, CSS junto con un servidor básico en Express

REPORTE FINAL

Esteban Eliud Arguijo Ramirez

Cargo: LEAD

introducción

Ser un LEAD implica liderar y coordinar un equipo, siendo responsable de dirigir el desarrollo de un proyecto o tarea, el LEAD no solo toma decisiones importantes, sino que también actúa como intermediario entre los miembros del equipo, asegurándose de que todos trabajen con un mismo propósito y que se optimicen los recursos, de igual manera, debe gestionar los tiempos de manera efectiva, mantener la motivación alta y adaptarse a cambios en los requerimientos, asegurando que los entregables sean de calidad y se entreguen en el tiempo previsto

Requerimientos:

- LEAD-Esteban-Realizar votacion para elegir el proyecto a realizar en node.js
- LEAD-Esteban-Seleccionar que plataforma utilizar para la base de datos (MongoDB, MySQL o PostgreSQL)
- LEAD-Esteban-Aplicar un formato para los reportes individuales
- LEAD-Esteban-Gestionar el avance de los requerimientos
- LEAD-Esteban-Possible actualización de requerimientos
- LEAD-Esteban-Gestionar entregable de reportes y reporte final
- LEAD-Esteban-Comunicación en el equipo

Realización

Una vez definidos los requerimientos a mi cargo y definidos los requerimientos de mis compañeros empecé a trabajar en darle seguimiento a cada uno, realizando así un arduo análisis sobre los avances realizados

1er Requerimiento

Para el primer requerimiento realice un a votación para elegir que proyecto realizaremos en node.js, todos concluimos que la mejor opción seria darle seguimiento a la actividad 2 realizada anteriormente y mejorar ese diseño, así que nos pusimos manos a la obra y empezamos a trabajar en ello

Para ello realice un Excel en donde le asignaría a cada uno de mis compañeros unos requerimientos a seguir, estos basandonos en la mejora del diseño y funcionalidad de la actividad 2, en dicho Excel me base en los roles de cada uno para asignarles tareas diferentes, el formato utilizado fue basado en el anterior Excel realizado en la actividad anterior teniendo así divisiones en donde se agregaría cada comentario como se muestra a continuación:

ID_Requerimiento	Descripción REQ		
LEAD-Esteban-Realizar votacion para elegir el proyecto a realizar en node.js	Generar una votación para la elección del proyecto		
LEAD_Realización	LEAD_Notas	QA_Resultados	QA_Notas
Listo	Se llevó a cabo la votación y ya tenemos el proyecto definido		

Este Excel se subió al main en la carpeta documentos para que mis compañeros puedan ver los requerimientos y al momento de ir realizandolos me notificarían para ir anotando sus avances, así mismo hay varias versiones del Excel en donde se van modificando los requerimientos y se van notificando los avances

Avance-De-Proyecto / Documentos /		
Name	Last commit message	Last commit date
..		
QA-MatrizDeRequerimientos.xlsx	QA-Pedro-Agregar-se agrego la matriz de requerimientos	1 hour ago
Req_DEV.xlsx	Dev-Frida-DocumentosRequeridos-Se añaden los requerimientos	2 hours ago
Req_Primera actualizacion.xlsx	LEAD-Esteban-Se notifica el avance obtenido hasta el momento por mis ...	3 days ago
Req_SegundaActualizacion.xlsx	LEAD-Esteban-Se notifica los avances realizados por mis compañeros	1 minute ago

2do Requerimiento

Evaluamos las opciones existentes y una vez propuestas al equipo decidimos que la opción mas factible seria realizar la base de datos en MySQL ya que el DEV tenia antiguo conocimiento sobre este sistema y se le facilitaba la realización de esta, además que ya había realizado anteriormente tareas donde necesitaba conectar la base de datos además de eso, nos basamos en la facilidad, la accesibilidad y el rendimiento que nos podría proporcionar

3er requerimiento

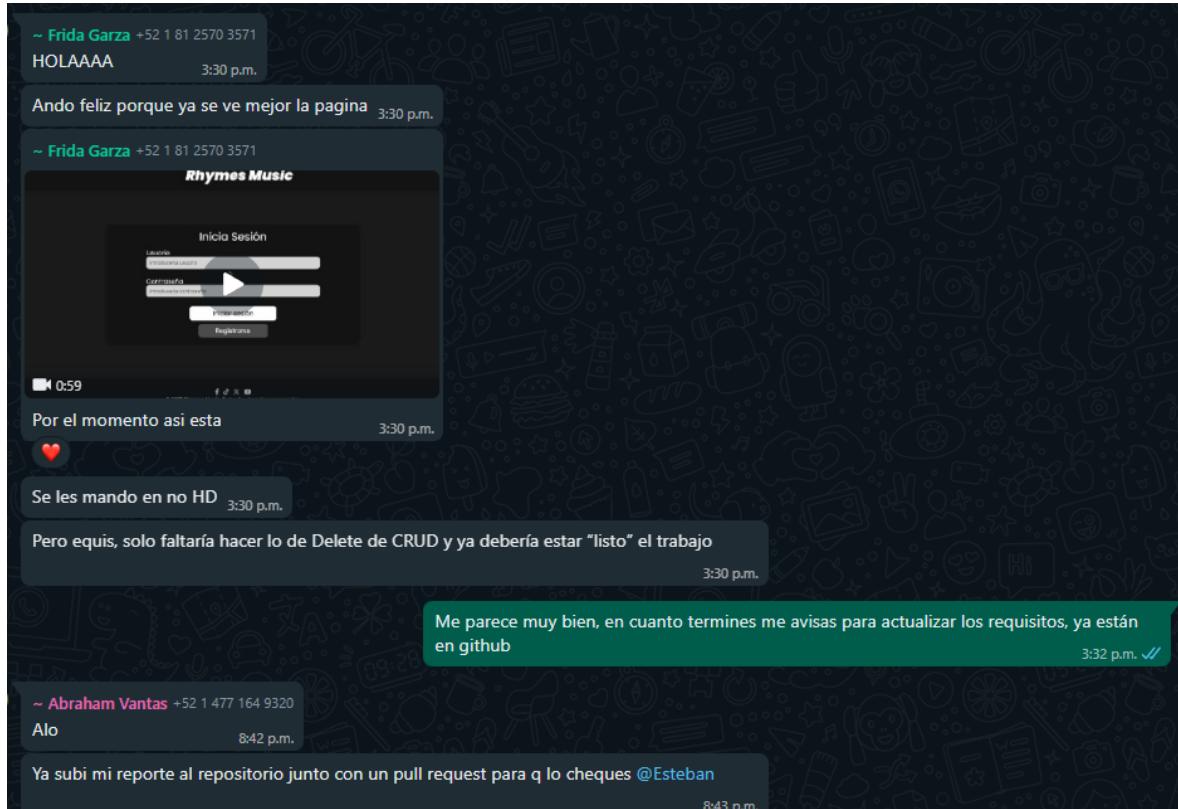
Establecimos un formato para cada reporte individual, esto para ayudarles con la optimización y el orden a la hora de realizarlos, el formato utilizado fue el mismo de la actividad 2 donde el formato es el siguiente:

Al igual que establecimos un formato para el Reporte utilizamos una nomenclatura igual Para la creación de branches, siguiendo de esta Manera “ROL-NOMBRE-DETALLE DE LO QUE SE REALIZÓ”

Reporte individual	
Nombre	Puesto
<i>Información básica de lo que se realizó.</i>	
Requerimientos	
Dar una muestra ya sea en lista o tabla los requerimientos que se tienen en su puesto.	
Realización	
Dar una explicación detallada de lo que se realizó, para cada uno de los requerimientos; se deben mostrar capturas de muestra donde se confirma que se realizó el requerimiento.	
Inconvenientes/Problemas y Solución	
En caso de que haya ocurrido un problema con alguno de los requerimientos, detalla lo que se tuvo que hacer ya sea solo o en conjunto de un compañero para solucionarlo.	

4to requerimiento

Al momento de que mis compañeros iban realizando sus requerimientos, mi tarea es ir dando seguimiento a cada uno de ellos, en dado caso de que alguno tuviera alguna complicación se le notificaría al responsable sobre el error, al mismo tiempo se iba actualizando el Excel, con las notas del LEAD, de igual manera se le notificaba al QA para ir testeando los avances conforme se vayan realizando, esto lo hicimos para que el QA no se quedara sin que hacer hasta el ultimo momento que la pagina este realizada por completo, esto se baso mas que nada por la comunicación que mantuvimos durante la realización de dicho proyecto, establecimos la comunicación por el medio de WhatsApp en donde se notificaba cada avance, al momento de la notificación se checaba en GitHub donde se había subido el archivo para poder aprobarlo mediante los PULLREQUEST una vez aprobados se actualizaría el Excel



Como pueden observar aquí se notifica el avance del DEV, al momento de notificarse se actualiza el Excel usando este formato

ID_Requerimiento	Descripción REQ	LEAD_Realización	LEAD_Notas
LEAD-Esteban-Realizar votacion para elegir el proyecto a realizar en node.js	Generar una votación para la elección del proyecto	Listo	Se llevó a cabo la votación y ya tenemos el proyecto definido
LEAD-Esteban-Seleccionar que plataforma utilizar para la base de datos (MongoDB, MySQL o PostgreSQL)	Hacer una reunion con el equipo para acordar que plataforma utilizar para la realización de la base de datos	Listo	Despues de evaluar las opciones optamos por MySQL
LEAD-Esteban-Aplicar un formato para los reportes individuales	Reutilizar el formato que utilizamos en la actividad 2 creado por Frida	Listo	Formato definido y compartido con el equipo
LEAD-Esteban-Gestionar el avance de los requerimientos	Gestionar el avance de cada uno de mis compañeros a la vez que voy dandole notas sobre la realización de estos mismos	Listo	Se creó un seguimiento del avance
LEAD-Esteban-Possible actualización de requerimientos	Aquí analizare cada uno de los avances que se tenga sobre los requerimientos de cada uno de los integrantes del equipo	Listo	Se revisaron los requerimientos y se actualizaron según lo necesario

5to Requerimiento

Al momento de obtener complicaciones con el avance del proyecto en dadas ocasiones optamos por cambiar los requerimientos ya que había alguna complicación de por medio, simplemente se actualizaban para facilitar al compañero dueño del requerimiento, esto para agilizar el trabajo y su funcionalidad como se muestra a continuación

A	B	C	D	E	F	G
1	ID	Categoría	Descripción de la prueba	Prioridad	Responsable	Estatus
2	QA-1	Frontend	El sitio web es responsive en móvil.	Alta	Dev	
3	QA-2	Frontend	El sitio web es responsive en computadora.	Alta	Dev	
4	QA-3	Frontend	El sitio web es responsive de manera general y tiene un correcto funcionamiento.	Alta	Dev	
5	QA-4	Frontend	Los botones funcionan correctamente.	Alta	Dev	
6	QA-5	Frontend	No permite enviar campos vacíos en ninguna de las páginas.	Alta	Dev	
7	QA-6	Frontend	Redirección adecuada.	Alta	Dev	
8	QA-7	Backend	Encripta la contraseña.	Media	Dev	
9	QA-8	Frontend	No permite campos negativos en ninguna de las páginas.	Alta	Dev	
10	QA-9	Frontend	Las validaciones (como la de email) están validadas en todas las páginas.	Media	Dev	
11	QA-10	Frontend + Backend	Los datos se guardan en la base de datos sin problemas.	Alta	Dev	
12	QA-11	Frontend	El texto no sobresale de los cuadros en ninguna de las páginas.	Media	Dev	
13	QA-12	Frontend + Backend	La página de perfil muestra la información correcta.	Alta	Dev	
14	QA-13	Frontend	Todo funciona a la perfección en perfil.	Media	Dev	
15	QA-14	Frontend + Backend	El botón de cerrar sesión funciona adecuadamente.	Media	Dev	
16	QA-15	Frontend	El botón de ajustes abre de manera adecuada una página para editar los datos.	Baja	Dev	
17	QA-16	Frontend + Backend	Al editar y guardar los cambios en perfil, estos se mantienen y se actualizan en la BD.	Alta	Dev	
18	QA-17	Backend	Se crea el usuario de manera correcta.	Alta	Dev	
19	QA-18	Backend	Verifica que no estén los datos repetidos al crear el usuario.	Alta	Dev	
20						
21			IOTA: Los issues que están en estatus amarillo fueron corregidos en la branch QA-Responsive, pero cambia el diseño, esto por problemas de tiempo, pero no afecta el funcionamiento.			
22						
23						
24						
25						

En dado caso que algún compañero quisiera actualizar sus requerimientos debería de dar una explicación sobre el porque se realizo y porque tomo esa decisión en dado caso de que se le complicara todo el equipo debía estar enterado sobre que el requerimiento se actualizara

6to Requerimiento

Nos centramos en la preparación, revisión y entrega de los informes periódicos generados a lo largo del proyecto, así como la elaboración del reporte final, que resume los resultados y los avances obtenidos

Para gestionar los entregables, se estableció un calendario de entregas los reportes fueron diseñados de acuerdo con un formato que yo diseñe esto aseguraba un buen diseño para todos mis compañeros

Para el reporte final junte todos los reportes individuales de mis compañeros, asegurándome que todo lo escrito en dicho reporte coincidiera con lo establecido en los requerimientos, una vez terminado se les compartía a mis compañeros para que le dieran el visto bueno, en dado caso de que algo no les pareciera se actualizaría el reporte final, con dichas actualizaciones

7mo Requerimiento

La comunicación en el equipo fue fundamental para llevar a cabo un buen trabajo, esto se logró ya que todos estábamos dispuestos a contestar ciertas dudas que iban surgiendo, a la hora de las votaciones igualmente se llevó a cabo la comunicación tanto presencial como por mensajes

Para lograr este objetivo, se implementaron diversas estrategias de comunicación, comenzando con reuniones regulares de seguimiento, tanto presenciales como virtuales, estas reuniones se utilizaron para discutir el progreso de las tareas, resolver problemas o dudas y revisar el estado de los requerimientos, esta acción de comunicación nos facilitó a la hora de realizar el trabajo de una forma eficiente de igual manera en GitHub se encontraban los archivos correspondientes para solucionar cualquier tipo de duda que fuera surgiendo



Avance-De-Proyecto Public

Watch 1

main ▾

11 Branches 0 Tags

Go to file

t

Add file ▾

Code ▾



Eliudsito LEAD-Esteban-Se notifica los avances realizados por mis compañeros

ec8172 · 37 minutes ago 95 Commits

Documentos

LEAD-Esteban-Se notifica los avances realizados por mis co...

37 minutes ago

Fronted

Correcion de encabezado

yesterday

Reportes

CI/CD-Paola-Reportes-Se añadieron los reportes

1 hour ago

node_modules

QA-Pedro-Archivos necesarios-Se añaden los archivos y dat...

6 hours ago

views

Se incluye la barra lateral en las paginas necesarias y se adju...

yesterday

.env

QA-Pedro-Archivos necesarios-Se añaden los archivos y dat...

6 hours ago

README.md

CI/CD-Paola-Descarte-Se descarto un link de requerimientos...

4 hours ago

package-lock.json

Generación de archivo automatico de las dependencias NPM

2 days ago

package.json

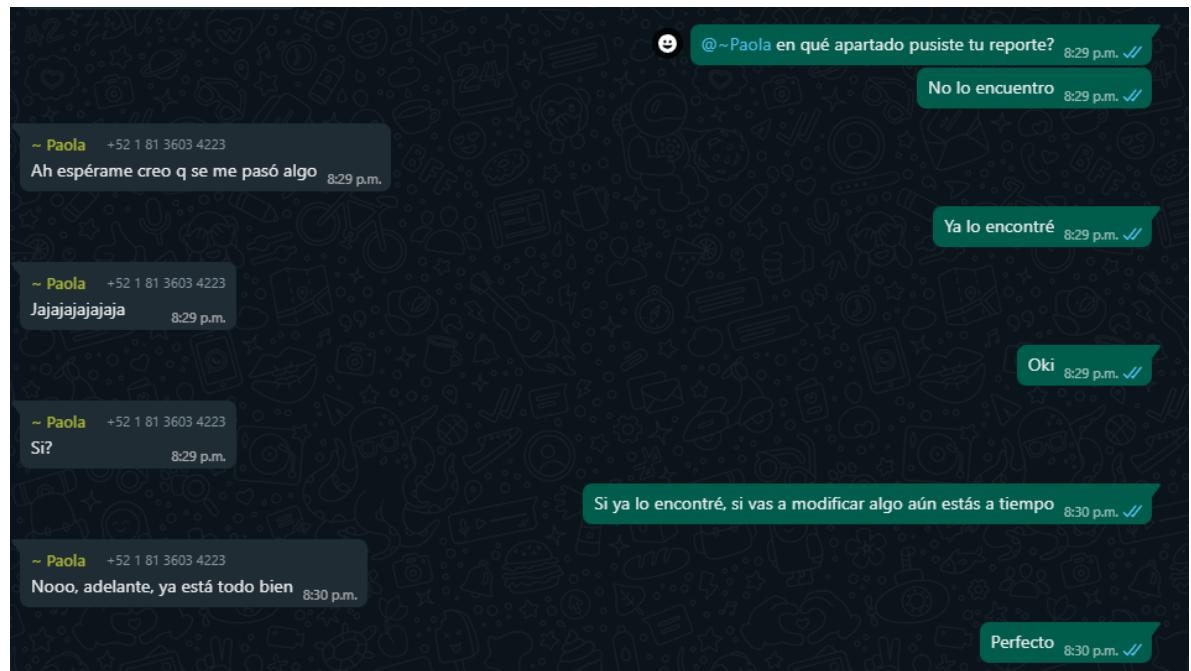
Se crea un servidor en Express, agregando las librerias neces...

yesterday

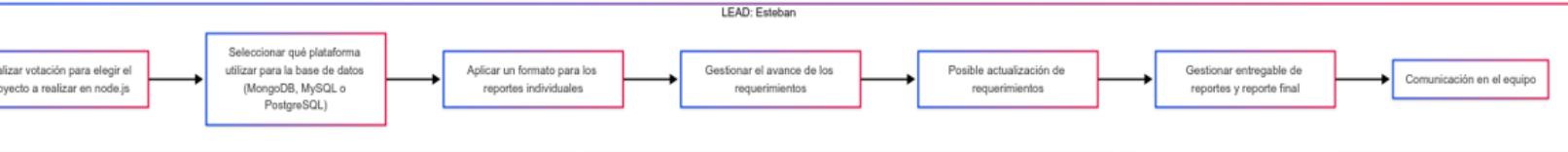
server.js

Se corrige error en eliminar usuario

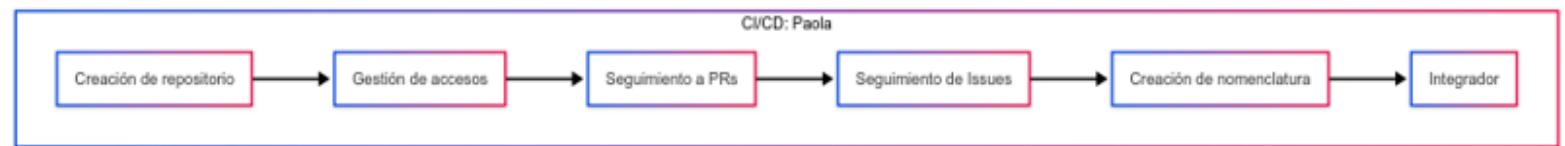
6 hours ago



Por ultimo realice un diagrama de actividades donde fui enlazando cada uno de los requerimientos impuestos para mis compañeros, si había algún error también se agregaba al diagrama
El diagrama comenzaba por el LEAD con los siguientes requerimientos



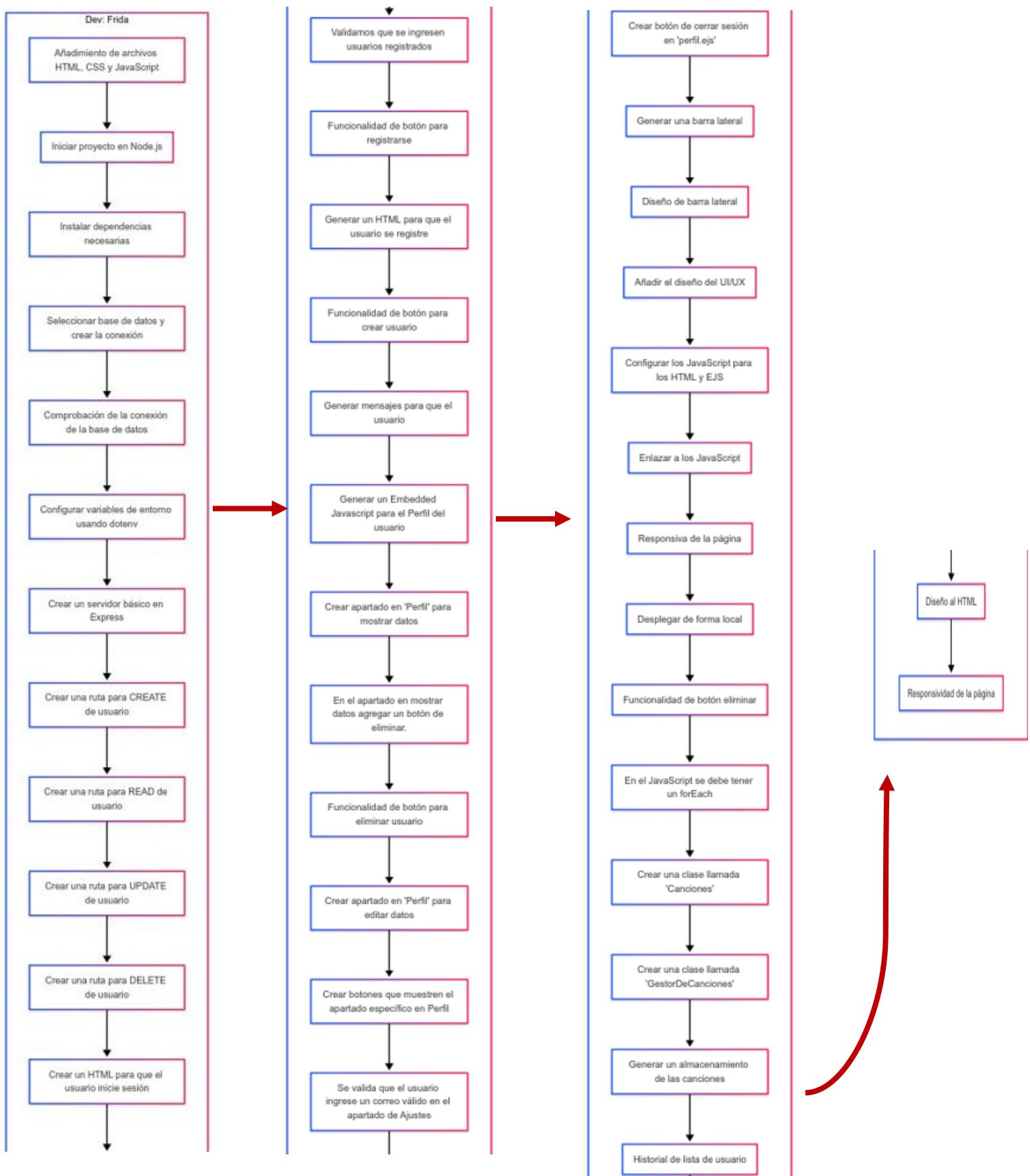
después se enlaza con el CI/CD



Para después continuar con el UI/UX



El siguiente sería el de DEV que este es el más extenso debido a los requerimientos asignados



Terminando por último el rol de QA



Inconvenientes/Problemas/Solución

A la hora de realizar mi reporte y mi rol asignado no tuve ningún inconveniente de por medio, me gusto mucho trabajar en este rol, en la creación de requerimientos, en el seguimiento de los avances y en la realización de el reporte final.

Link de GitHub: <https://github.com/PaolaUrdiales/Avance-De-Proyecto>

CI/CD-PAOLA

Para este avance del proyecto final, mi rol fue el de CI/CD, el cual es responsable de automatizar e integrar los cambios al repositorio y de estar al tanto de las modificaciones que hagan los demás colaboradores, es decir, en caso de que un integrante del equipo decida subir su cambio al repositorio (hacer un push o crear un pull request) el CI/CD debe verificar que los cambios pasen las pruebas automatizadas y sean confiables para que se fusionen con el main del repositorio.

Para tener una mejor organización de las funciones que tendría como CI/CD, me estuve basando en algunos de los requerimientos que el Lead definió y en algunos más que yo creí convenientes agregar.

Requerimiento: CI/CD-Paola-Creación de repositorio

- *Se crea el repositorio en el cual se deben depositar todos los archivos con los que se trabajarán*

Solución:

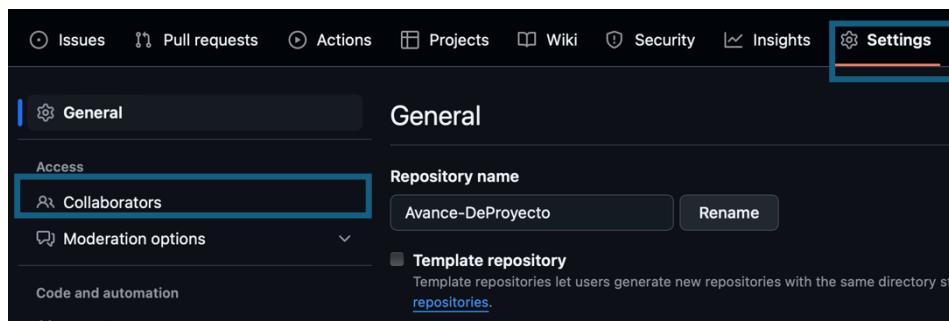
The screenshot shows the 'Create a new repository' form on GitHub. It includes fields for 'Owner' (PaolaUrdiales), 'Repository name' (Avance-De-Proyecto), and a 'Description (optional)' field. Under 'Initialize this repository with:', there is a checkbox for 'Add a README file'. At the bottom, there are sections for '.gitignore' template and 'Choose a license'.

Se comenzó con la interfaz de Github en donde se creó un nuevo repositorio para el proyecto, el repositorio ayuda a que todos los colaboradores que estamos trabajando en el proyecto, estemos al tanto de los cambios que se están haciendo en él, para la creación del repositorio se tuvo que definir un título, en este caso fue “Avance de Proyecto”, se dejó su configuración original para que fuera público, así mismo se decidió agregar un Readname para que ahí se almacenarán los datos más fundamentales que todos los colaboradores deben saber.

Requerimiento: CI/CD-Paola-Gestión de accesos

- Se agregan colaboradores al repositorio, los cuales podrán hacer cambios y branches en git.

Solución:



Una vez creado el repositorio, dentro de su configuración en el apartado de “colaboradores” podemos agregar usuarios para que puedan acceder a la edición del repositorio, eso es muy bueno ya que se puede trabajar paralelamente en el proyecto y así obtener mejores resultados de una manera más eficiente y rápida:



En el botón “add people” es donde ingresamos los usuarios de las personas que deseamos sean parte del repositorio. Es de esta forma como se agregaron los integrantes del grupo al repositorio, incluyendo al docente:

A screenshot of the 'Manage access' interface on GitHub. It lists several users with their status as 'Pending Invite'. A blue box highlights the first four users: 'Eliudsito', 'FridaGarzaG', 'Pedro', and 'eestrada'. A green box highlights the last user, 'Yugidar'. Arrows point from these boxes to labels: 'Integrantes de equipo' (Team members) for the first four and 'Docente' (Teacher) for the last one. At the bottom, there's a note about getting team access controls and discussions for contributors in an organization, and a 'Create an organization' button.

Requerimiento: CI/CD-Paola-Seguimiento a PRs

- Se asegura de dar continuidad a todo proceso/cambio que haga cada compañero.

The screenshot shows two GitHub pull request pages. The top page is for pull request #6, which has been merged. It lists several commits from 'FridaGarzaG' and 'PaolaUrdiales', with reviews from 'pedrodeleondev' and 'Yugilder'. The bottom page is for pull request #5, also merged, showing commits from 'pedrodeleondev' and 'PaolaUrdiales'. Both pages include sections for reviewers, assignees, labels, projects, milestones, development, and notifications.

Para dar un correcto seguimiento a los cambios que mis compañeros realizaban, se tenía que estar al tanto de los pull request que ellos realizaban, cuando me percataba de que existía uno nuevo, me dirigía a él y observaba su contenido, ya verificando que los cambios que ha realizado el colaborador son correctos, se le hace un review al pull, en donde se indican si los cambios están bien, prácticamente todos mis comentarios indicaban que los PR realizados eran correctos y que se aceptaban los cambios, esto porque lo que el contenido de ellos era conveniente. Posteriormente, para que se les pudiese hacer merge y que los cambios se mostraran en el main el PR debe contar con la aprobación de la mayoría de los integrantes del equipo, de esta manera se evitan cambios no deseados en el main. Ya que haya la mayoría de aprobación se le da merge para que los cambios sean fusionados definitivamente en el contenido del main.

Requerimiento: CI/CD-Paola-Seguimiento de Issues.

- Implementación de un proceso para el seguimiento y gestión de issues en GitHub.

Cuando el QA ingresaba un issue al repositorio, era necesario que dicho problema se le asignara a alguien para obtener una solución, es así como se dialogaba y el CI/CD los asignaba a algún colaborador.

Los issues fueron de gran ayuda para tener en cuenta los cambios urgentes que se le debían realizar a los archivos almacenados en el repositorio.

Issue creado por el QA, asignado a la DEV, finalmente solucionado y cerrado:

The screenshot shows a GitHub issue page for issue #9. It was opened by 'pedrodeleondev' and self-assigned by them. 'PaolaUrdiales' assigned 'FridaGarzaG' to the issue. A comment from 'pedrodeleondev' states: 'Se ha resuelto el problema, el issue con id QI-3 ha sido cerrado.' The issue is labeled as 'Closed'. The right sidebar shows assignees ('pedrodeleondev'), labels ('No labels'), projects ('No projects'), milestones ('No milestone'), relationships ('None yet'), development ('Create a branch for this issue or link a pull request.'), and notifications ('Customize').

Requerimiento: CI/CD-Paola-Creacion de nomenclatura.

- Definición de un estándar de nomenclatura para branches, commits e issues.

The screenshot shows a GitHub repository page for 'Avance-De-Proyecto'. The README.md file contains the following content:

```
Avance-De-Proyecto
Repository para el monitoreo de la creación del Avance de Proyecto de Full Stack
Integrantes de equipo: Frida, Pedro, Abraham, Esteban y Paola

Nomenclatura para Pushes La nomenclatura utilizada sera la siguiente: ROL - NOMBRE -QUE SE REALIZÓ - DESCRIPCIÓN

Nomenclatura para Branches La nomenclatura utilizada para crear nuevos branches sera la siguiente: ROL-QUE SE ESTA REALIZANDO

Link del Figma a la pagina de Perfil: https://www.figma.com/design/oajHWs7IAd6EcF8xwk2FOg/Untitled?node\_id=0-1&p=f&l=XhtrmPsZR7euyQ-0

Link de la matriz de requerimientos para testear la página: https://utmedu-my.sharepoint.com/:x/r/personal/al03103352\_tecmilenio\_mx/\_layouts/15/Doc.aspx?sourcedoc=%7B3B09122A-FF3F-44AE-98A5-BCD795381791%7D&file=QA-MatrizDeRequerimientos.xlsx&action=default&mobileredirect=true
```

La nomenclatura utilizada en avance de proyecto fue reutilizada de la actividad anterior y expuesta de forma más visible en el README, esto para que fuese mucho más sencillo para todos los colaboradores ubicarla y saber cómo se deben llamar tanto los pushes, commits, branches e issues.

Requerimiento: CI/CD-Paola-Integrador

- Desarrollo de un integrador de cambios para CI/CD, asegurando despliegues automatizados.

El objetivo que tiene el trabajo del **CI/CD**, es monitorear y automatizar los cambios que realizaron los colaboradores del equipo, todo esto para tener un flujo de trabajo mucho más uniforme y sin posibilidad a errores en la rama principal (**main**).

Algunas de las actividades que realicé como **CI/CD** fueron:

- Realizar **reviews** a los **PR** de mis compañeros
- Hacer **merge** cuando los cambios eran totalmente correctos
- Monitoreo y asignación de **issues**, es decir, aseguraba que los **issues** que se fueran presentando tuviesen un colaborador asignado y que el problema se resolviera y fuera cerrado.

Nota: Esto se realizaba siempre y cuando los PR cumplieran con la nomenclatura y su contenido fuera correcto.

QA-Pedro-Eliminar-se elimino un archivo innecesario del github pedrodeleondev authored 1 hour ago	Verified c5a9479 ⌂ ⌄
QA-Pedro-Nombre-se cambio el nombre del archivo pedrodeleondev authored 2 hours ago	Verified b7ddfb5 ⌂ ⌄
Rename Reporte-QA.docx to ReportePedro-QA.docx pedrodeleondev authored 2 hours ago	Verified ac0ab5d ⌂ ⌄
QA-Pedro-Archivos-Se añadieron los reportes de QA pedrodeleondev authored 2 hours ago	Verified 1ba2e60 ⌂ ⌄
CI/CD-Paola-Descarte-Se descarto un link de requerimientos ya que han sido actualizados en la carpeta de documentos PaolaUrdiales authored 2 hours ago	Verified 6275e3d ⌂ ⌄
CI/CD-Paola-Aprobacion-Se aprueban los cambios realizados en el README PaolaUrdiales authored 2 hours ago	Verified 0987549 ⌂ ⌄
QA-Pedro-Cambio README-Se agrego el link de la matriz de requerimientos pedrodeleondev authored 2 hours ago	Verified 7a6619c ⌂ ⌄
CI/CD-Paola-Aprobacion-Se aseguro que los cambios son correctos asi que se aprueban para que se fusionen con el demás contenido del main PaolaUrdiales authored 4 hours ago	Verified 4dc02cd4 ⌂ ⌄
QA-Pedro-Archivos necesarios-Se añaden los archivos y datos necesarios para el funcionamiento de la página. pedro321 committed 4 hours ago	2b92293 ⌂ ⌄
QA-Pedro-Archivos necesarios-Se añaden los archivos y datos necesarios para el funcionamiento de la página. pedro321 committed 4 hours ago	197062e ⌂ ⌄
CI/CD-Paola- Aprobación- Se fusionan los cambios realizados al main PaolaUrdiales authored 4 hours ago	Verified 581474f ⌂ ⌄
Se corrige error en eliminar usuario.	

Reporte individual de CI/CD

Abraham Isaí Garza Sánchez

CI/CD

Se trabajo a partir del Figma creado a partir de la Actividad 2. Se siguió utilizando la misma paleta de colores, pero se agregaron objetos o se cambiaron de lugar algunos elementos para dar lugar a nuevas funciones dentro de la página, así como la creación de nuevos botones para la misma.

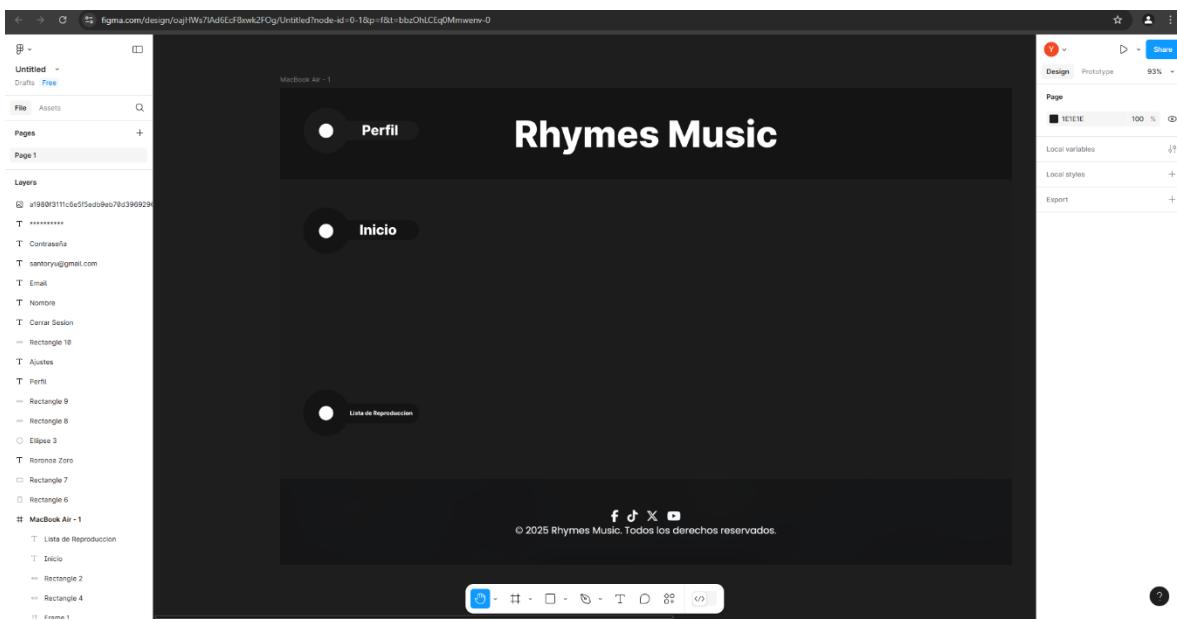
Requerimientos

- Realizar el diseño de las nuevas paginas creadas para el avance de proyecto
- Diseño de páginas totales
- Diseño para Móvil y Web
- Gestionar la experiencia de usuario

Realización

Idea inicial/Prototipo de las páginas a ser agregadas

Se empezó a trabajar desde una pagina vacía, pero con la estética inicial de la página, es decir sin contenido dentro pero con la misma paleta de colores, header y footer para darnos una idea de donde puede ir cada cosa. Al inicio se propuso que para llevar al usuario a los diferentes lugares dentro de la página, se usaran botones con forma circular, para simular que son discos haciendo alusión a que la página está enfocada en música, también se propuso que los botones fueran minimalistas en el sentido de que no incluirían ningún tipo de texto, si no que al tener el mouse por encima del botón, el disco sacaría una forma ovalada con el texto que dice a donde llevaría ese botón, agregándole que mientras el mouse este sobre el botón, el disco giraría, de nuevo haciendo alusión a como para poder leer un disco este mismo tiene que girar para ser leído, por lo que en general se tiene esta idea de que para que los discos “funcionen” o “reproduczan música” estos tienen que girar. A continuación, se muestra como fue el prototipo inicial para esto.



Problemas y presentación de iconos.

La idea anterior fue presentada al equipo para saber su opinión así como al Dev para saber si la realización de este sería posible, una vez que fue dada la luz verde se siguió el trabajo con estos, sin embargo al pensarlos un poco más surgió el problema de que quizás los botones no eran lo suficientemente intuitivos, lo que alearía la exploración de la página, por lo que se optó por los “discos” en lugar de tener un círculo en medio para simular la forma de estos, simplemente se tendría un ícono que daría una mejor idea de a dónde te lleva cada botón de la página. Donde los “assets” fueron puestos en la página para exponerlos y ver cuál podría quedar mejor para cada botón. Son los mostrados a continuación.

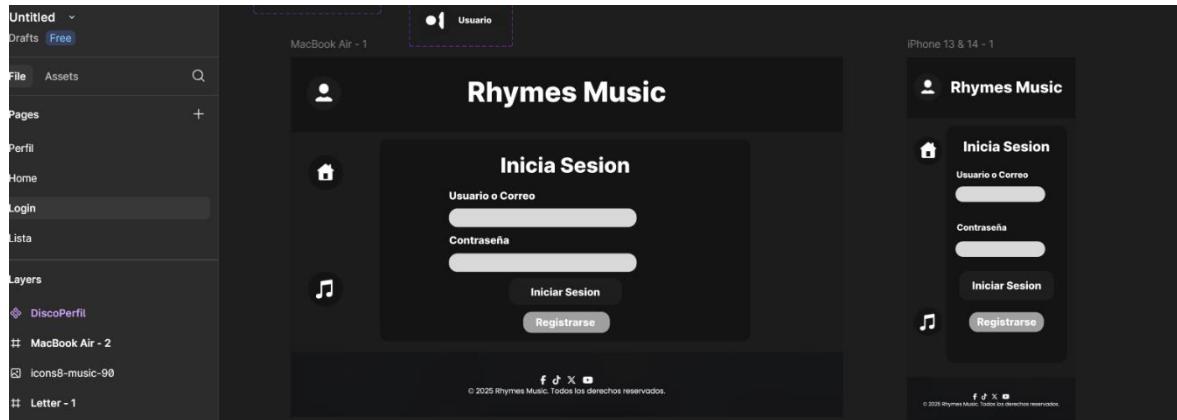


Tanto para los botones de perfil como el de “Home” o página de inicio, no fue difícil la elección, sin embargo, para elegir el de “Lista de reproducción” fue un poco más complicado, se tuvieron en cuenta 4 tipos de íconos para representar esta sección de la página. Durante la elaboración del Figma se utilizó el que parece una nota musical, o una corchea, pero para la elaboración el HTML se terminó utilizando el ícono que representa barras de sonido que salen de la parte inferior.

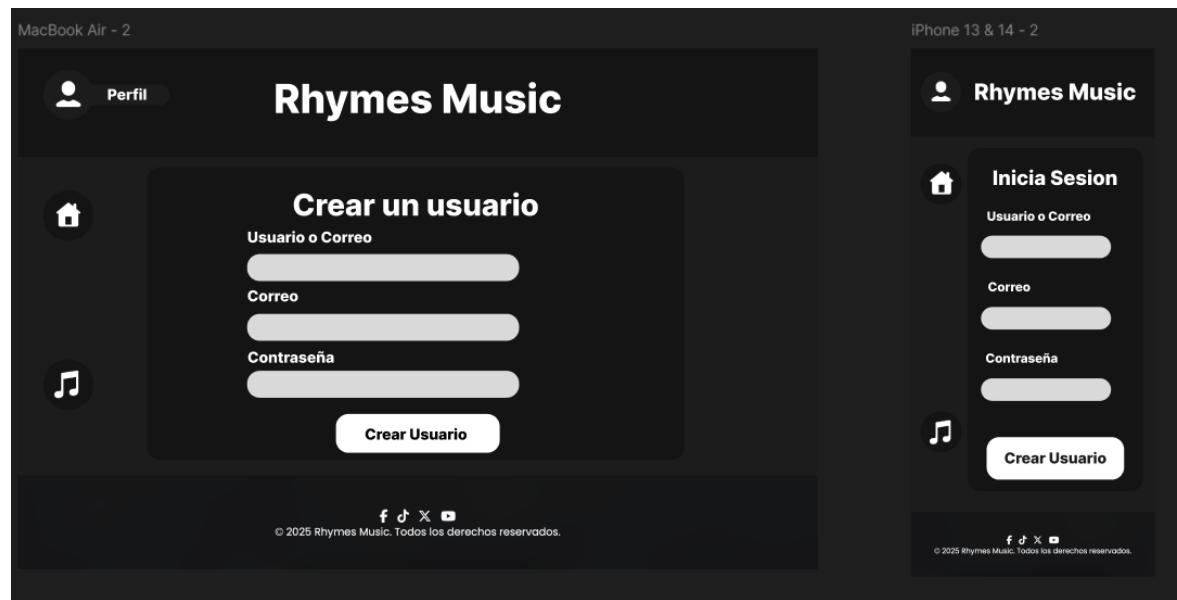
Elaboración de la pagina de Registro/Inicio de Sesión

Lo siguiente que se trabajo fue la página dedicada al Registro/Inicio de Sesión den nuevos usuarios.

Como dicho anteriormente, se siguió respetando la misma paleta de colores y la fuente utilizada de la anterior actividad. Al presionar el botón que se encuentra en la esquina superior izquierda con icono de persona, y de no estar registrado o de no haber iniciado sesión, te llevara a la página para Iniciar Sesión y Registrarse. Que son las paginas mostradas a continuación, también fue elaborada la versión para móviles, donde se le agrego la funcionalidad dentro el Figma para poder hacer “scroll” dentro de esta al momento de presentarla mediante el botón de reproducción en la esquina superior derecha.



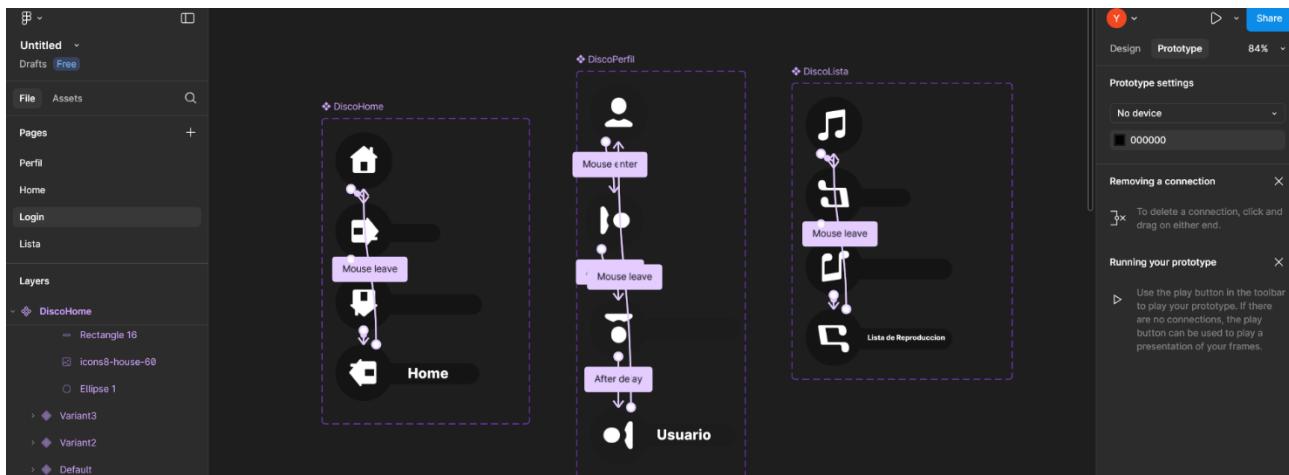
En caso de no contar con una cuenta y presionar el botón de registrarse la página será la siguiente.



Elaboración de los botones.

Después de tener la página de Inicio de sesión y registro terminadas se opto por empezar a trabajar en los botones de la pagina para dar una previsualización de cómo se podrían ver dentro del HTML.

Estos fueron elaborados a partida de una figura circular de 80x80 pixeles, donde en su centro se coloco el icono respectivo de a donde te llevara el botón. Luego que hacer el aspecto físico del botón se tuvo que realizar el aspecto lógico, es decir, luego de hacer el botón, luego se selecciono la pestaña de prototype dentro de Figma, y se le agrego una interacción. Esto hace que el botón se convierta en un “asset” dentro de Figma para poder agregarle funcionalidades, en este caso, se crearon cuatro variantes del mismo disco donde el icono del medio se gira 90 grados hasta cumplir una rotación completa, añadido a esto se agregó un pequeño ovalo que sale del disco para mostrar el texto de a donde te dirige el botón sobre el que estas haciendo “hover”. Después de esto, se hizo un enlace entre las 4 variantes creadas para que estas se accionen mientras el mouse esta sobre ellas para dar así el efecto giratorio que se tenia pensado para los botones, también se agregó la función para que al momento de que el mouse no se encuentre dentro del botón, este gire en reverse y esconda el cuadro de texto que salió de este para una mejor experiencia de usuario. Este mismo proceso se repitió para los 3 botones de la página y se muestra a continuación cómo funciona el proceso lógico.



Página de Perfil.

Para la página de perfil, se opto por un diseño donde en la parte izquierda te muestre la foto de perfil del usuario, acompañado de 3 botones en la parte inferior, uno para ver la información del perfil, otro para modificar la información del perfil, y otro para cerrar sesión. Los primeros dos siguieron respetando la paleta de colores y la estética de la página, mientras que para el ultimo se opto por un color rojo para resaltar que esta opción es diferente a las demás.

Por la parte derecha se mostrará contenido diferente dependiendo del botón seleccionado de la parte izquierda, por default o al presionar el botón de Perfil, en la parte derecha se mostrará el nombre de usuario que fue escrito al elaborar la cuenta y el email que fue creado en la elaboración de esta. Mientras que, al presionar el botón de Ajustes, el contenido de la parte derecha cambiara para mostrar cuadros de texto para modificar la información si así se desea y se agregó un botón color verde para resaltar que la información será actualizada. También se realizó la versión móvil para ambas.

Las páginas se muestran a continuación.

Rhymes Music

iPhone 13 & 14 - 1

Nombre: Roronoa Zero
Email: santoryu@gmail.com

iPhone 13 & 14 - 2

Nuevo Nombre: Roronoa Zero
Nuevo Email: santoryu@gmail.com
Nueva Contraseña: Actualizar informacion

Prototype settings

MacBook Air
Silver
000000

Rhymes Music

iPhone 13 & 14 - 2

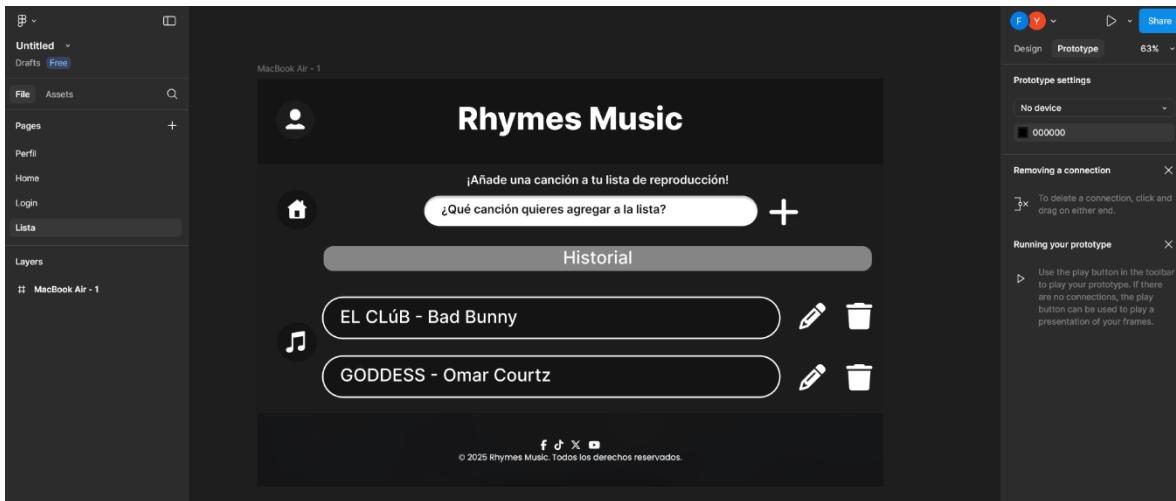
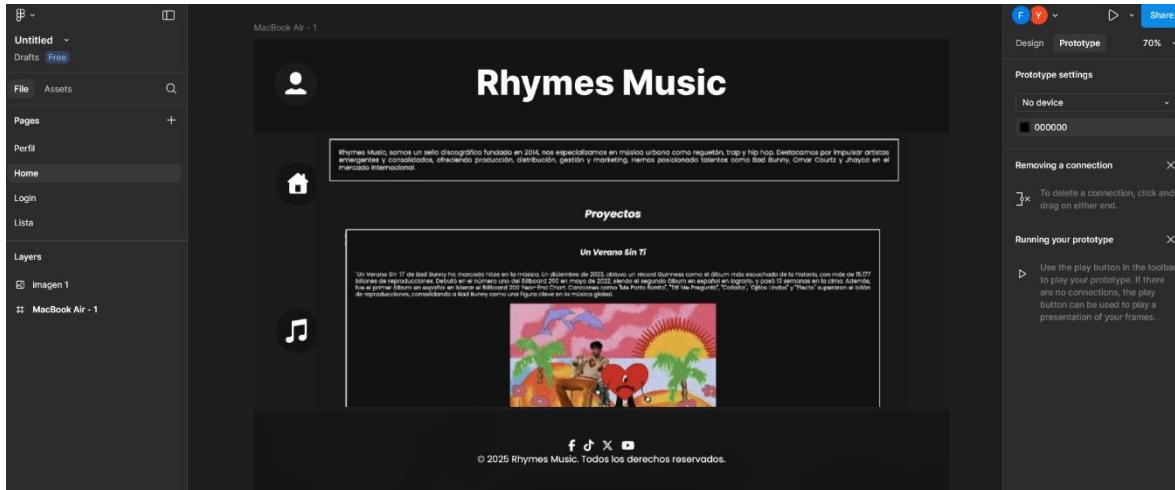
Nuevo Nombre: Roronoa Zero
Nuevo Email: santoryu@gmail.com
Nueva Contraseña: Actualizar informacion

Prototype settings

MacBook Air
Silver
000000

Página de Lista de Reproducción y Página principal.

La página de inicio y la de Lista de Reproducción fueron mínimamente alteradas, su única modificación fue el posicionamiento de las cosas, que fue recorrida a la derecha para dar un poco de espacio a los botones que fueron agregados, se muestran a continuación con la implementación de los nuevos botones,



Link del Figma

Finalmente se anexa el link para acceder al Figma del proyecto y poder visualizar de primera mano los cambios hechos y el trabajo realizado:

<https://www.figma.com/design/oajHWs7lAd6EcF8xwk2FOg/Untitled?node-id=65-188&t=m68SNJ2iSI5jACmP-1>

Reporte individual

Frida Sarahi Garza Gálvez

Dev

En esta actividad obtuve el cargo de Dev, que se encarga de desarrollar varios requisitos de la actividad, como el HTML, CSS y JavaScript que apoyan en el diseño de la página discográfica y las validaciones junto a las acciones de varios aspectos de la página web que buscamos ofrecer.

Se nos solicita la selección de una base de datos, la creación de un proyecto en Node.js, la creación de operaciones CRUD que se asocien a el tema de la página, por ello, decidimos que se hiciera un Login y Registro, ya que después de ello, se abarcaría la edición y eliminación de los datos del usuario.

Requerimientos

Estos son los requerimientos que genere debido a lo que se me solicita a lo largo de la creación del código y necesidades de la actividad:

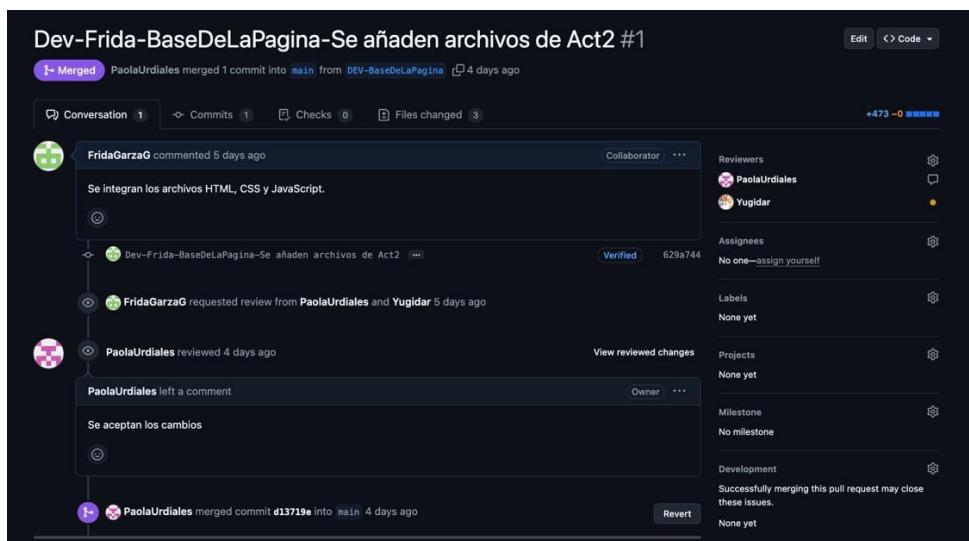
- Añadimiento de archivos HTML,CSS y JavaScript
- Iniciar proyecto en Node.js
- Instalar dependencias necesarias
- Seleccionar base de datos y crear la conexión
- Comprobación de la conexión de la base de datos
- Configurar variables de entorno usando Dotenv
- Crear un servidor básico en Express
- Crear una ruta para CREATE de usuario
- Crear una ruta para READ de usuario
- Crear una ruta para UPDATE de usuario
- Crear una ruta para DELETE de usuario
- Crear un HTML para que el usuario inicie sesión
- Validamos que se ingresen usuarios registrados
- Funcionalidad de botón para registrarse
- Generar un HTML para que el usuario se registre
- Funcionalidad de botón para crear usuario
- Generar mensajes para que el usuario
- Generar un Embedded JavaScript para el Perfil del usuario
- Crear apartado en 'Perfil' para mostrar datos
- En el apartado en mostrar datos agregar un botón de eliminar.
- Funcionalidad de botón para eliminar usuario
- Crear apartado en 'Perfil' para editar datos
- Crear botones que muestren el apartado específico en Perfil
- Se valida que el usuario ingrese un correo valido en el apartado de Ajustes
- Crear botón de cerrar sesión en 'perfil.ejs'
- Generar una barra lateral
- Diseño de barra lateral
- Añadir el diseño del UI/UX
- Configurar los JavaScript para los HTML y EJS
- Enlazar a los JavaScript
- Responsiva de la pagina
- Desplegar de forma local

Realización

Para mi realización de requerimientos fue necesario concordar con mi equipo lo que buscábamos crear y añadir, por ello, vimos correcto ampliar las páginas de la página y agregar nuevos HTML y diseño. Y a continuación los requerimientos y como fueron abarcados.

- Añadimientode archivos HTML, CSS y JavaScript

Para el requerimiento, fue necesario ingresar al repositorio en GitHub, creado por mi compañera Paola Urdiales, donde creamos una Branch en donde específico que es el añadimientode los archivos 'index.html' que le cambia el nombre a 'listaCanciones.html', el 'pagPrincipal' que es el archivo que se utilizó por nuestro compañero Pedro De León en la primera actividad, junto a esos documentos, se agregaron 'app.js'y 'style.css', siendo una base sólida de lo que nuestra página llegaría a ser.



Podemos observar en la captura de pantalla como se hizo el Pull Request y como fue aceptado por el CI/CD para así dar comienzo con la creación de la actividad.

- Crear el proyecto en Node.js

Para la creación en el Node.js se tuvieron que seguir ciertos pasos necesarios, como la comprobación de la instalación con el comando **node -v** que nos comenta si está instalado Node.js y **npm -v** que es el gestor de paquetes Node Package Manager.

```
test@Laptop ~ % node -v  
v23.6.1  
test@Laptop ~ % npm -v  
11.0.0
```

Podemos ver en la captura de pantalla como tras ingresar dichos comandos en la terminal nos da la versión instalada.

Y con ello confirmado, ahora ingresamos a la carpeta del proyecto, haciendo uso del comando **cd** junto al **nombre de la carpeta del proyecto**, dando el siguiente resultado:

```
test@Laptop ~ % cd AVANCE-DE-PROYECTO  
test@Laptop AVANCE-DE-PROYECTO %
```

Tras ello, haciendo uso del comando **npm init -y**, se nos genera un archivo llamado package.json en nuestra carpeta. En la consola se nos mostrara el contenido del archivo mismo y podremos ver que contiene varios puntos relevantes del proyecto como:

- Nombre del proyecto
- Versión del proyecto
- Archivo principal del proyecto
- Comandos que podemos ejecutar
- Palabras clave para describir el proyecto
- Autor del proyecto
- Tipo de licencia
- Definición de si se usan módulos o CommonJS

Esto nos ayudara en la funcionalidad del servidor de nuestro proyecto en Node.js

```
test@Laptop AVANCE-DE-PROYECTO % npm init -y
Wrote to /Users/test/AVANCE-DE-PROYECTO/package.json:

{
  "name": "avance-de-proyecto",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs",
  "description": ""
}
```

- Instalación de dependencias

Tras la creación del proyecto en Node.js, ahora debemos hacer las descargas de las dependencias que son necesarias en nuestra actividad, siendo muy relevantes para ciertas funciones que buscamos abarcar en nuestro proyecto, algunas de ellas son:

Bcryptjs

```
test@Laptop AVANCE-DE-PROYECTO % npm install bcryptjs
added 1 package, and audited 2 packages in 919ms
```

Librería empeñada en encriptar las contraseñas para antes de almacenarlas en la base de datos.

CORS (Cross-Origin Resource Sharing)

```
test@Laptop AVANCE-DE-PROYECTO % npm install cors
added 3 packages, and audited 5 packages in 721ms
found 0 vulnerabilities
```

Nos ayuda en el permiso de que el servidor de Backend acepte solicitudes desde dominios diferentes al Frontend, evitando que sean bloqueadas por el navegador debido a la seguridad.

Dotenv

```
test@Laptop AVANCE-DE-PROYECTO % npm install dotenv
added 1 package, and audited 6 packages in 624ms
```

Nos permite cargar variables de entorno desde el archivo .env de la página, manteniendo configuraciones sensibles fuera del código.

Ejs

```
test@Laptop AVANCE-DE-PROYECTO % npm install ejs
added 16 packages, and audited 22 packages in 1s
3 packages are looking for funding
  ↩ https://funding.js.org
```

Es un motor de plantillas para generar en el HTML dinámicas desde el Backend, siendo así que podamos insertar datos directamente en el HTML.

Express

```
test@Laptop AVANCE-DE-PROYECTO % npm install express
added 68 packages, and audited 90 packages in 2s
17 packages are looking for funding
  ↩ https://funding.js.org
```

Es un framework para la creación de un servidor en node.js, nos da una estructura sencilla para manejar rutas, peticiones, middleware y más funciones para un desarrollo óptimo en el servidor Backend.

Express-session

Nos permite manejar sesiones de usuario en la página, dando la oportunidad de almacenar información en el servidor sobre el estado de un usuario entre las peticiones, dandonos la gestión de autenticación de usuarios.

```
test@Laptop AVANCE-DE-PROYECTO % npm install express-session
added 6 packages, and audited 96 packages in 908ms
```

MySQ

```
test@Laptop AVANCE-DE-PROYECTO % npm install mysql mysql2
added 24 packages, and audited 120 packages in 2s
```

Ambas librerías son para interactuar con la base de datos MySQL, siendo útiles para operaciones CRUD en la base de datos.

Nodemo

n

```
test@Laptop AVANCE-DE-PROYECTO % npm install nodemon
added 25 packages, and audited 145 packages in 2s
```

Nos facilita el desarrollo en Node.js, siendo que al reiniciar el servidor se hace en automático cuando detecta cambios en los archivos del proyecto.

Sweetalert

2

```
test@Laptop AVANCE-DE-PROYECTO % npm install sweetalert2
added 2 packages, and audited 147 packages in 911ms
```

Esta librería nos permite mostrar alertas en el Frontend, dando una mejoría en la experiencia del usuario debido a la forma visualmente atractiva de los mensajes.

Cambios refejados en el package.json

```
{
  "name": "avance-de-proyecto",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs",
  "description": "",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.4.7",
    "ejs": "^3.1.10",
    "express": "^4.21.2",
    "express-session": "^1.18.1",
    "mysql": "^2.18.1",
    "mysql2": "^3.12.0",
    "nodemon": "^3.1.9",
    "sweetalert2": "^11.15.10"
  }
}
```

Tras instalar todas las dependencias necesarias para nuestro avance del proyecto, podemos ver en el archivo 'package.json' que se fueron integradas en la sección de dependencias las versiones de cada una de ellas, dándonos así información de este, como el nombre del paquete y la versión de este. Y si notamos, en la versión se tiene el signo '^' que aclara que es la versión más reciente para la compatibilidad con la versión mayor.

Cada una de estas dependencias es necesaria porque son fundamentales para el funcionamiento seguro, eficiente y dinámico de la página.

- Selección y creación de base de datos

Se es necesario el seleccionar una base de datos que sea optima, eficiente y fácil de manejar para las operaciones CRUD que buscamos manejar más adelante, por ello, vemos correcto tomar la base de datos MySQL, creemos que las siguientes características son fundamentales para nuestra aplicación:

- Facilidad de uso: Debido a la facilidad en la gestión de la base de datos.
- Confiabilidad: Debido al manejo que hemos hecho en anteriores materias y sabemos que es de las bases de datos más utilizadas.
- Escalabilidad: Porque permite ampliarse para abarcar las demandas que van surgiendo.
- Flexibilidad: Porque nos permite tener una flexibilidad para desarrollar aplicaciones de bases de datos SQL tradicionales y sin esquema SQL.

Creemos que MySQL es siendo una base de datos relacional se vincula perfectamente en lo que esperamos recibir en nuestra página, dando así almacenamiento de datos en tablas para generar un gran

almacén ofreciéndonos un modelo de datos lógico, como tablas de datos, vistas, filas y columnas, siendo una programación flexible.

Por ello, vemos correcto crear nuestra base de datos en MySQL, dando así comienzo en lo que buscamos abarcar en el avance del proyecto.

Creación de base de datos

Para la creación de la base de datos fue necesario instalar Homebrew debido a que el dispositivo donde se está realizando el trabajo es una MAC, fue necesario el implementarlo, para después utilizando el comando *brew install mysql*, el cual nos ofrece la descarga e instalación de la última versión que se encuentre en Homebrew de MySQL.

```
test@Laptop AvanceProyecto % brew install mysql  
==> Downloading https://formulae.brew.sh/api/formula.json
```

Después de la confirmación de la instalación, ingresamos el comando *brew services start mysql*, el cual nos ayuda a iniciar MySQL y le da la configuración de hacer que se ejecute en segundo plano como un servicio.

```
test@Laptop AvanceProyecto % brew services start mysql  
==> Successfully started `mysql` (label: homebrew.mxcl.mysql)
```

Tras ver el mensaje de que se muestra en consola, podemos conectarnos a MySQL con el comando *mysql -u root -p*, debido a que yo ya lo tenía instalado, cree una contraseña y sin ella no me sería posible conectarme a MySQL, y una vez ingresada la contraseña correcta, podemos ver cómo nos da una bienvenida.

```
test@Laptop AvanceProyecto % mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 14  
Server version: 9.0.1 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2025, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Con la confirmación de la entrada a MySQL, creamos la base de datos, y le daremos por nombre 'rhymes_proyecto' siguiendo las normas para la nomenclatura, donde nos pide que no sean con guiones medios.

```
mysql> CREATE DATABASE rhymes_proyecto;  
Query OK, 1 row affected (0,03 sec)
```

Tras la creación de la base de datos, se utiliza el comando USE, para poder seleccionar la base de datos donde deseamos trabajar, y podemos ver como después de dicho comando, se nos comenta que podemos modificar la base de datos.

```
mysql> USE rhymes_proyecto;  
Database changed
```

Después de estar en la posibilidad del cambio en nuestra base de datos, nos planteamos lo que esperamos que contenga y los datos que esperamos manejar en las operaciones CRUD, dando así la siguiente tabla:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRIMARY	NULL	Auto_increment
nombre	varchar(50)	YES		NULL	
correo	varchar(100)	YES	UNIQUE	NULL	
password	varchar(100)	YES		NULL	

Ocasionalmente el comando necesario para crear la base de datos sea el siguiente:

```
mysql> CREATE TABLE usuarios(id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(50), correo VARCHAR(100) UNIQUE, password VARCHAR(100));
Query OK, 0 rows affected (0,09 sec)
```

Podemos notar como especificamos el nombre, tipo, tamaño y orden de las columnas de la tabla usuarios, dándonos así una base fundamental para lo que haríamos en las operaciones CRUD, tomemos en cuenta que esto es para una base de datos de forma local, mi compañero Pedro De León se encargó de hacer que la base de datos se encuentre en la nube, dando una facilidad de acceso a los datos desde cualquier dispositivo.

-Configuración de variables en Dotenv

Para la configuración de variables en Dotenv fue necesario crear un archivo en la raíz de la carpeta de nuestro proyecto que tome por nombre '.env', siendo este el que contenga las variables de entorno, como podemos ver en la siguiente imagen, se portan datos relevantes como:

```
.env
1 #Definición de variables
2 DB_HOST=localhost
3 DB_USER=root
4 DB_PASSWORD=Clave1234
5 DB_NAME=rhymes_proyecto
```

- La dirección del servidor de la base de datos.
- El usuario con el que se conecta la base de datos.
- La contraseña del usuario para el ingreso a la base de datos.
- El nombre de la base de datos que se utilizará.

Teniendo las variables definidas, después, cuando creemos un servidor básico en Express, deberemos agregar a ese archivo lo siguiente:

Cargar las variables de entorno

Este comando nos ayuda a importar la librería dotenv, haciendo que cargue las variables de entorno desde el archivo '.env' que se encuentra en la raíz de la carpeta del proyecto.

```
require('dotenv').config();
```

Cargar el paquete de MySQL2

Luego agregamos que se importe el paquete de MySQL2, que nos permitirá interactuar con la base de datos desde Node.js, para así ofrecernos las consultas, conexión y manipulación de la base de datos.

```
const mysql = require('mysql2');
```

Crear la conexión a la base de datos

Agregamos el método que nos ayuda a conectar la base de datos de MySQL, siendo basada en los parámetros pasados con las variables de entorno para hacer la conexión correcta a la base de datos. Podemos notar como se define la dirección del servidor, el nombre del usuario, la contraseña del usuario y el nombre de la base de datos.

```
//Conexión a MySQL con variables de entorno
const connection = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME
});
```

Exportar la conexión de la base de datos

Esto nos ayuda a poder exportar desde el archivo actual, hacia otros, para hacerle uso desde otro archivo.

```
module.exports = connection;
```

Iniciar la conexión a la base de datos

Este método nos ayuda a comprobar que al iniciar la conexión con la base de datos nos dé un mensaje en consola basado en si hubo un fallo en la conexión o si fue exitosa, dando así, que, si ocurre un error, se mande un mensaje de que hubo una problemática en la conexión y la muestra detalles del error, y si fue exitosa, nos da un mensaje en consola donde nos afirma que todo fue exitoso.

```
//Prueba de la conexión a la base de datos
connection.connect((err) => {
  if (err) {
    console.error('Error al conectar a la base de datos:', err);
    return;
  }
  console.log('Conexión exitosa a la base de datos.');
});
```

-Creación de un servidor básico en Express

Iniciamos con la creación de un archivo llamado ‘server.js’, el cual contendrá las importaciones de las librerías y módulos que necesitamos, junto a las rutas de las páginas que esperamos manejar más adelante.

Notemos en las siguientes imágenes, como ya incluimos lo que hablamos en la configuración de las variables de entorno respecto a la conexión de la base de datos, pero podemos ver como se fueron agregados más líneas de código en el archivo.

```
ver.js > ...
//Variables de entorno
require('dotenv').config();

//Librerías
const express = require('express');
const cors = require('cors');
const mysql = require('mysql2');
const bcryptjs = require('bcryptjs');
const path = require('path');
const expressSession = require('express-session');
const methodOverride = require('method-override'); // Para manejar el método PATCH

const app = express(); //Iniciar app
```

En la sección de librerías que podemos notar por el comentario, podemos ver como se importan los módulos como Express, CORS, MySQL2, Bcryptjs, Path, Express-session y el Method Override.

Ya habíamos hablado de casi todos esos métodos, excepto Method Override que trata

acerca de poder importar un middleware que permita sobrescribir el método HTTP en solicitudes, junto al método Path, que se encarga de manejar y utilizar las rutas de los archivos de manera segura. Y junto a ello, se agrega la instancia para crear la aplicación Express, que permite definir rutas, middleware y la escucha de solicitudes.

Luego, hacemos uso de Middlewares para manejar cuerpos de solicitudes JSON, habilitación de CORS, habilitación de que los formularios HTML usen métodos como PATCH y DELETE, y la habilitación de análisis de solicitudes cuando se envían formularios HTML.

```
//Middleware
app.use(express.json());
app.use(cors());
app.use(methodOverride('_method')); //Method-override para sobrescribir el método HTTP
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'fronted')));
app.use('/js', express.static(path.join(__dirname, 'fronted', 'js')));
```

Después de ello, se asegura el definir que usaremos EJS (Embedded JavaScript) como motor de plantillas, dándonos así un HTML dinámico con código JavaScript implementado, con eso hecho, establecemos la carpeta donde se podrán encontrar las plantillas, dándonos la ruta del directorio.

```
//Motor de plantillas
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
```

Con la definición del motor de plantillas, se agrega lo que habíamos comentado en la configuración de las variables de entorno, siendo la conexión y la prueba de si fue correcta la conexión con la base de datos, y

```
//Conexión a MySQL con variables de entorno
const connection = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME
});

module.exports = connection;

//Prueba de la conexión a la base de datos
connection.connect((err) => {
  if (err) {
    console.error('Error al conectar a la base de datos:', err);
    return;
  }
  console.log('Conexión exitosa a la base de datos.');
});

//Configuración de la sesión
app.use(expressSession({
  secret: 'claveSecreta',
  resave: false,
  saveUninitialized: true,
  cookie: { secure: false }
}));

app.use(cors({
  origin: '*',
  methods: ['GET', 'POST', 'PATCH', 'DELETE'],
  allowedHeaders: ['Content-Type', 'Authorization']
}));
```

tras ello, continuamos con la configuración de la sesión con express-session, la cual es un middleware que nos ayuda a manejar las sesiones en la página, haciendo que el usuario esté autenticado a lo largo de varias páginas en las que se encuentre, para esto, necesitamos definir la cadena secreta que nos ayuda a firmar el ID de la sesión, haciendo que el usuario no la pueda manipular, el agregar que no se guardara la sesión nueva si no ha habido cambios provocando que evitemos operaciones innecesarias de guardado, para después agregar que se guardara la sesión nueva incluso si no ha sido modificada, y especificamos que la cookie este el false debido a que no necesariamente debemos meternos tanto en ese aspecto.

Después de eso, configuramos el CORS, que es un middleware que nos permite aceptar solicitudes desde dominios diferentes,

siendo así que podamos hacer solicitudes desde cualquier dominio, que pueda ser utilizado por el cliente ya sea el GET, POST, PATCH y DELETE, junto al definir las cabeceras que podrían ser enviadas en las solicitudes del usuario.

Con eso definido, se agregan las rutas para servir los archivos HTML, donde ponemos como inicio el HTML que tendrá por nombre 'sesión', después se tendrán las demás rutas como para la página después del Login, siendo 'pagPrincipal' basándonos en el hecho de que si el usuario no está autenticado mediante la propiedadloggedin de la sesión lo redirige a la página de inicio; y agregamos las demás rutas de los HTML, como 'listaCanciones' y 'registrarse'.

```
//HTML "sesión" como principal
app.get('/', (req, res) =>{
  res.sendFile(path.join(__dirname, 'fronted', 'sesion.html'))
})

//HTML "Pagina principal"
app.get('/pagPrincipal', (req, res) => {
  if (req.session.loggedin) {
    res.sendFile(path.join(__dirname, 'fronted', 'pagPrincipal.html'));
  } else {
    res.redirect('/');
  }
});

//HTML "Lista de canciones"
app.get('/listaCanciones', (req, res) => {
  res.sendFile(path.join(__dirname, 'fronted', 'listaCanciones.html'));
});

//HTML "Registrarse"
app.get('/registrarse', (req, res) => {
  res.sendFile(path.join(__dirname, 'fronted', 'registrarse.html'));
});
```

Por último, lo que se debería tener al momento, es la línea que nos ayuda a inicializar el servidor estando en el puerto 3000, donde usando Express nos apoya en que se inicie el servidor en nuestra maquina local.

```
app.listen(3000, ()=>{
  console.log('Server funciona en http://localhost:3000')
})
```

Para asegurar el funcionamiento, hacemos que se mande un mensaje a la consola donde nos dice que funciona y nos da la URL para poder tratar con la aplicación.

-Creación de ruta CREATED de usuario

Con el servidor creado, ahora nos empeñamos en la creación de rutas básicas para operaciones CRUD, comenzando con la operación de CREATED, donde debemos poder crear un usuario y sea agregado a la tabla de usuarios de la base de datos llamada 'rhymes_proyecto', tomando en cuenta que debemos ingresar datos como nombre, correo y contraseña.

Iniciamos con definir que la ruta sea POST en Express, haciendo que, si el usuario envíe un formulario a '/registrarse', esta función sea llamada para manejar dicha solicitud, continuando en las siguientes líneas, podemos ver que capturamos los valores que ingresa el usuario en el formulario, siendo el nombre que

```
app.post('/registrarse', validacionRegistro, async(req, res)=>{
  const usuario = req.body.usuario;
  const email = req.body.email;
  const pass = req.body.pass;

  console.log('Usuario:', usuario);
  console.log('Email:', email);
  console.log('Contraseña:', pass);
```

definimos como usuario, junto al correo y contraseña almacenándolos en las constantes que tienen nombres como 'usuario', 'email' y 'pass'.

Después de ello, hacemos que se impriman en la consola los valores que fueron ingresados para así

comprobar que todo está funcionando correctamente.

Tras eso hecho, hacemos uso de la librería Bycryptjs para poder cifrar la contraseña antes de almacenarla en la base de datos, siendo así que la función bcryptjs.hash tome la contraseña sin cifrar y el hash la cifra, tomemos en cuenta que especificamos que el número es 8, ocasionando esa cantidad de rondas de cifrado que se debía aplicar en la contraseña.

```
let passwordHash = await bcryptjs.hash(pass, 8);
connection.query('INSERT INTO usuarios (nombre, correo, password) VALUES (?, ?, ?)', [usuario, email, passwordHash], async(error, results)=>{
```

Continuando con el código, ahora hacemos un connection.query, siendo un método que nos ayuda a realizar consultas a la base de datos, y esta vez, estamos insertando el nuevo usuario a la tabla de usuario que habíamos creado con anterioridad; para poder hacer la inserción a la tabla debemos hacer la consulta SQL, como podemos ver en las letras verdes, especificando en que tabla y los valores de las columnas, agregando que los valores con signos de interrogación para marcar la posición de los valores que se insertaran, siendo que los valores [usuario, email, password] sean los insertados en la tabla en su campo correspondiente; y el async(error, results) es una función callback, donde después de realizar la consulta a la base de datos se ejecuta, y recibe dos parámetros ya sea por si ocurrió un error en la consulta o si los resultados fueron correctos y todo fue exitoso.

Después de eso, dentro del connection.query(), generamos un if(error) en donde si ocurre un error durante la consulta se ejecuta lo que está dentro del bloque, siendo así que se imprima en la consola el error y que de un mensaje de alerta donde tras decir que se debió a el email repetido, lo regresa a la página de registrarse para que intente de nuevo el registro de usuario.

```

} else{
  res.render('registrarse',{
    alert: true,
    alertTitle: "Registro",
    alertMessage: "¡Registro exitoso!",
    alertIcon: "success",
    showConfirmButton: false,
    timer: 1500,
    ruta: ''
  })
}

```

```

if(error){
  console.log(error);
  res.render('sesion', {
    alert: true,
    alertTitle: "Error",
    alertMessage: "Email repetido, favor de intentar de nuevo",
    alertIcon: "warning",
    showConfirmButton: true,
    timer: 1500,
    ruta: "registrar"
  });
}

```

Con ese if, ahora checamos si la consulta fue exitosa, provocando que el registro se haya podido realizar, se utiliza mismamente un mensaje, pero esta vez de que fue exitoso, en donde nos dice que el registro fue exitoso y a la par lo redirige a la página de sesion, cuando entras por primera vez a la página, para ahora sí, ingresar su usuario y su contraseña creada, vimos correcto hacer eso, para mayor seguridad y manejo de entrada,

Este código abarca una ruta CRUD, que se empeñó en el registro de un nuevo usuario en el HTML que se llamará 'registrar.html'.

- Creación de ruta READ de usuario

Para realizar la ruta READ, necesitamos primero hacer una ruta donde el usuario ingrese sus datos para validar su inicio de sesión y después de ello, poder mostrar los datos de lectura del usuario, siendo así que se cree la ruta GET para leer los datos como nombre y email del usuario que ingreso en sesión.

Empezamos con el generar la ruta POST, para manejar el inicio de sesión y hacemos uso de la función async porque usaremos await para comparar las contraseñas, con eso, ahora se extraen los valores que se enviaron en el formulario que se tendrá en el HTML 'sesion', y los datos que tomaremos serán el usuario y contraseña.

```

app.post('/sesion', async (req, res) => {
  const usuario = req.body.usuario;
  const pass = req.body.pass;
})

```

Con eso hecho, ahora continuamos con hacer uso de un if, donde se verifica que los dos campos tengan valores, si los tienen, ahora se hace una consulta a la base de datos con connection.query, donde consultamos la base de datos para encontrar el nombre de usuario ingresado, recordemos que el signo de interrogación es para marcar la posición para injectar el valor que tenemos, en async en results mandaremos los datos obtenidos del usuario si se obtuvieron.

```

if (usuario && pass) {
  connection.query('SELECT * FROM usuarios WHERE nombre = ?', [usuario], async (error, results) => {
}

```

Tras hacer la consulta, generamos un if para manejar el error en la consulta, donde notifica que hubo un error en la consulta en la consola y en la página donde muestra un mensaje llamativo notificándolo y redirigiéndolo a la página de inicio.

```

if (error) {
  console.log(error);
  return res.render('sesion', {
    alert: true,
    alertTitle: "Error",
    alertMessage: "Error en la base de datos. Intenta nuevamente.",
    alertIcon: "error",
    showConfirmButton: true,
    timer: null,
    ruta: ''
  });
}

```

Después se hace una verificación de usuario y contraseña, donde si no se encuentra el usuario en la base de datos, y si la contraseña ingresada no está almacenada en la base de datos le comenta que usuario y/o contraseña son incorrectos.

Dando así un mensaje dinámico donde le da a conocer al usuario su error.

```
...
} else {
  //Datos del usuario
  req.session.loggedin = true;
  req.session.userId = results[0].id;
  req.session.nombre = results[0].nombre;
  req.session.email = results[0].correo;

  //Mensaje de que se logro la sesion
  return res.render('sesion', {
    alert: true,
    alertTitle: "Inicio de sesion exitoso",
    alertMessage: "¡Bienvenido de nuevo!",
    alertIcon: "success",
    showConfirmButton: false,
    timer: 1500,
    ruta: 'perfil' //Manda a perfil
  });
}
```

else contendrá el mensaje de advertencia debido a que los campos están vacíos porque no ingreso nombre o contraseña.

```
if (results.length == 0 || !(await bcryptjs.compare(pass, results[0].password))) {
  return res.render('sesion', {
    alert: true,
    alertTitle: "Error de autenticación",
    alertMessage: "Usuario y/o contraseña incorrectos.",
    alertIcon: "error",
    showConfirmButton: true,
    timer: null,
    ruta: ''
  });
}
```

Después, de ese if, surge el else donde si los datos son correctos, se inicia la sesión, almacenando los datos del usuario para poder acceder a ellos en las próximas rutas a comentar; y tras almacenar los datos, se le da un mensaje de bienvenida al usuario y lo manda al archivo 'perfil.ejs', el cual detallaremos más adelante.

Y para finalizar esta parte de iniciar sesión, se genera un else del primer if donde validaba que se ingresaran datos y no estuvieran sin valores, y este

```
...
} else {
  return res.render('sesion', {
    alert: true,
    alertTitle: "Campos vacíos",
    alertMessage: "Por favor ingrese usuario y contraseña.",
    alertIcon: "warning",
    showConfirmButton: true,
    timer: null,
    ruta: ''
  });
}
```

Con la ruta para iniciar sesión hecha, ahora podemos hacer la ruta para mostrar el perfil, siendo una ruta

```
app.get('/perfil', (req, res) => {
  const userId = req.session.userId;
  //Verificar si el usuario en la sesion
  if (!userId) {
    return res.redirect('/');
  }
})
```

GET, en donde obtenemos el ID del usuario que fue almacenado en la sesión, y si el userID no llega a estar, nos redirigiría a la página de iniciar sesión.

Después se hace una consulta a la base de datos, basándonos en el ID que habíamos almacenado para obtener los datos como nombre y correo electrónico; y se hace el uso de un if en donde si el usuario fue

```
connection.query('SELECT nombre, correo FROM usuarios WHERE id = ?', [userId], (error, results) => {
  if (error || results.length === 0) {
    //En caso que ya haya sido eliminado.
    req.session.destroy((err) => {
      if (err) {
        console.log('Error al cerrar sesión después de eliminar la cuenta:', err);
      }
      return res.redirect('/');
    });
  }
});
```

eliminado de la base de datos se destruiría la sesión y luego redirige a la página para iniciar sesión.

Y si el usuario existe se mostraría en el perfil los datos, siendo solo el nombre y correo por seguridad, esto se abarca en el else, donde se extraen los datos del usuario y se envían a la vista que se generara en 'perfil.ejs'.

```
...
} else {
  //Si hay usuario se debe mostrar el perfil
  const user = results[0];
  res.render('perfil', {
    nombre: user.nombre,
    email: user.correo
  });
}
```

Creación de ruta UPDATE de usuario

Continuando con la creación de rutas para operación CRUD, ahora toca la ruta UPDATE, la cual nos ayudara a actualizar los datos del usuario, siendo desde el nombre, correo y la misma contraseña, y para darle comienzo, hacemos la ruta que tomara por nombre '/actualizarPerfil', la cual nos ayuda a definir la ruta como PATCH, y sirve para actualizar parcialmente los datos del usuario.

```
app.patch('/actualizarPerfil', async (req, res) => {
  const { newName, newPassword, newEmail } = req.body;
  const userId = req.session.userId; //ID del usuario desde la sesión
```

Con la ruta definida, ahora extraemos los valores que se enviaran desde el formulario que tendrá la página 'perfil.ejs', la cual puede recibir el nombre, correo o contraseña; y a la par se obtiene el ID desde la sesion, para asegurar que este autenticado.

```
if (!newName) {
  return res.render('perfil', {
    nombre: req.session.nombre || '',
    email: req.session.email || '',
    success: false,
    alert: true,
    alertTitle: "El nuevo nombre es obligatorio.",
    alertIcon: "error",
    showConfirmButton: false,
    timer: 1000,
    ruta: 'perfil'
  });
}

let updateValues = [newName];
```

Continuamos con la validación del correo electrónico, donde pedimos que si el usuario ingresa un nuevo correo se debe validar para que tenga el formato correcto, haciendo que porte @ y un punto; si el formato es invalido se le daría un mensaje de alerta sobre el error, pero si llegara a ser valido, se es agregado el newEmail el cual se incluye en updateValues que portara los valores que se están actualizando para la base de datos.

Tras la validación de email y nombre, ahora validamos la nueva contraseña, que será cifrada con bcryptjs para así, podrán almacenarla en la base de datos, y se hace uso de un await, ya que es un proceso asíncrono, el poder cifrar la contraseña, siendo así que después descifrarla, podamos agregarla al arreglo que tiene por nombre 'updateValues'.

```
//Verificar y encriptar la contraseña nueva si se ingresa una
let passwordHash = null;
if (newPassword) {
  passwordHash = await bcryptjs.hash(newPassword, 8);
  updateValues.push(passwordHash);
}

//Actualizar los datos en la base de datos
let updateQuery = 'UPDATE usuarios SET nombre = ?';
if (newEmail) {
  updateQuery += ', correo = ?';
}
if (passwordHash) {
  updateQuery += ', password = ?';
}
updateQuery += ' WHERE id = ?';
updateValues.push(userId);
```

Después validamos la entrada del nombre, donde si no proporciona un nuevo nombre, se le da una alerta indicando que es obligatorio, pero se mantienen los datos como nombre y email de la sesion para evitar que desaparezcan en la vista, y los redirige a perfil, para que lo vuelvan a intentar. Si se ingresa un nuevo nombre, se inicializa un arreglo que contenga el nuevo nombre como podemos ver en la linea let updateValues = [newName].

```
if (newEmail) {
  const emailRegex = /^[^@\s]+@[^\s]+\.\[^@\s]+$/;
  if (!emailRegex.test(newEmail)) {
    return res.render('perfil', {
      nombre: req.session.nombre || '',
      email: req.session.email || '',
      success: false,
      alert: true,
      alertTitle: "Correo electrónico inválido.",
      alertIcon: "error",
      showConfirmButton: false,
      timer: 1000,
      ruta: 'perfil'
    });
  }
  updateValues.push(newEmail);
```

```
//Verificar y encriptar la contraseña nueva si se ingresa una
let passwordHash = null;
if (newPassword) {
  passwordHash = await bcryptjs.hash(newPassword, 8);
  updateValues.push(passwordHash);
}
```

Con los valores actualizados en el arreglo ahora hacemos la creación de la consulta hacia la base de datos, la cual hace uso del comando que se muestra en la imagen el cual queda aclarar que los signos de interrogación son para dejar en claro la posición de donde deberá ser agregado en el nuevo valor, dicha actualización sólo podrá ser llevada a cabo si el usuario está autenticado y tenemos su ID correspondiente.

Para la actualización de la base de datos usamos 'connection.query', el cual ejecute la consulta a MySQL con los valores que fueron generados y agregados en el arreglo anteriormente comentado, en caso de qué llega a haber un error en la actualización, como un correo duplicado, se le daría un mensaje de error y especificando que fue lo que ocurrió.

```
connection.query(updateQuery, updateValues, (error, results) => {
  if (error) {
    console.log("Error en la actualización:", error);
    return res.render('perfil', {
      nombre: req.session.nombre || '',
      email: req.session.email || '',
      success: false,
      alert: true,
      alertTitle: "Error al actualizar los datos.",
      alertIcon: "error",
      showConfirmButton: false,
      timer: 1000,
      ruta: 'perfil'
    });
  }
})
```

Después hacemos la obtención de los datos actualizados, el cual misma mente hacemos uso de un 'connection.query' para validar que los datos hayan sido actualizados, haciendo que se haga una consulta

```
connection.query('SELECT nombre, correo FROM usuarios WHERE id = ?', [userId], (error, results) => {
  if (error) {
    console.log("Error al obtener los datos actualizados:", error);
    return res.render('perfil', {
      nombre: req.session.nombre || '',
      email: req.session.email || '',
      success: false,
      alert: true,
      alertTitle: "Error al obtener los datos actualizados.",
      alertIcon: "error",
      showConfirmButton: false,
      timer: 1000,
      ruta: 'perfil'
    });
  }
  const updatedUser = results[0];
  console.log('Datos actualizados:', updatedUser);
})
```

con el ID que teníamos del usuario, y en caso de qué no lleguen a ser obtenidos, se le mande un mensaje al usuario, respecto a que los datos no pueden ser dados. Asimismo, se genera un mensaje para que se muestren en consola respecto a qué datos fueron actualizados.

```
req.session.nombre = updatedUser.nombre;
req.session.email = updatedUser.correo;
//Mensaje donde se confirma la actualización
return res.render('perfil', {
  nombre: updatedUser.nombre,
  email: updatedUser.correo,
  alert: true,
  alertTitle: "Actualización exitosa",
  alertIcon: "success",
  showConfirmButton: false,
  timer: 1500,
  ruta: 'perfil'
});
```

En caso de que la actualización haya sido exitosa, se actualizan los datos en la sesión para que se puedan reflejar los cambios en la vista del usuario que se encontrará en 'perfil.ejs', asimismo se le dará al usuario un mensaje en el que se le da conocer que la actualización fue completamente exitosa.

- Creación de ruta DELETE de usuario

Traslación de la ruta UPDATE, ahora continuamos con la ruta que deberá encargarse de una de las operaciones CRUD que se nos es faltante, siendo la operación DELETE la cual debemos abarcar; esta operación se debe encargar de poder eliminar de la base de datos, la cuenta en el que se está el usuario, siendo así que una vez se ha eliminado dicha cuenta, el usuario tenga que regresar a la página de iniciar sesión.

```
app.delete('/eliminarUsuario', (req, res) => {
  console.log("Solicitud DELETE recibida para eliminar usuario con ID:", req.session.userId);
  const userId = req.session.userId;
```

Comenzamos con definir la ruta que tendrá por nombre '/eliminarUsuario', y comenzará con imprimir en la consola que se ha solicitado la eliminación de un usuario junto a la ID de solicitante, de a la par obtendremos el usuario que está autenticado guardando el valor en userID.

Y damos comienzo con la verificación de si userID tiene valor, en caso de qué no sea así, se nos imprimirán consola que no hay dicha usuario y a la parte se nos dará un mensaje donde se deja conocer dicha situación y lo redirige a la página de iniciar sesión.

Después de eso, en caso de qué si haya usuario, se ejecuta una consulta a la base de datos para poder eliminarlo, siendo así qué en el área donde se encuentra el signo de interrogación se ha puesto el valor que se tiene en

```
connection.query('DELETE FROM usuarios WHERE id = ?', [userId], (error, results) => {
  if (error) {
    console.log('Error al eliminar la cuenta:', error);
    return res.render('sesion', {
      success: false,
      alert: true,
      alertTitle: "Error al eliminar la cuenta.",
      alertIcon: "error",
      showConfirmButton: false,
      timer: 1000,
      ruta: 'perfil'
    });
  }
});
```

```
if (!userId) {
  console.log("No hay usuario.");
  return res.render('perfil', {
    success: false,
    alert: true,
    alertTitle: "No hay usuario.",
    alertIcon: "error",
    showConfirmButton: false,
    timer: 1000,
    ruta: ''
  });
}
```

userID, tras esa consulta se nos genera un if el cual validará si ha habido un error en la eliminación, y en caso que así sea, se registrará en la consola que ha habido un error al eliminar la cuenta y se dará una muestra de error en un mensaje dinámico en la página de iniciar sesión.

En caso de qué no haya sido así, ahora se destruiría la sesión del usuario para así poder asegurarnos de qué después de eliminar la cuenta ya no pueda seguir accediendo a la página, en donde también se generaría un if el cual abarcaría el error al cerrar la sesión, donde nos ofrecería un mensaje en la consola y a la par un mensaje de error, dando notificación en el Backend y en el Frontend.

```
return res.render('sesion', {
  success: true,
  alert: true,
  alertTitle: "Eliminación exitosa",
  alertIcon: "success",
  showConfirmButton: false,
  timer: 2000,
  ruta: '/'
});
```

```
req.session.destroy((err) => {
  if (err) {
    console.log('Error al cerrar sesión después de eliminar la cuenta:', err);
    return res.render('sesion', {
      success: false,
      alert: true,
      alertTitle: "Error al cerrar sesión.",
      alertIcon: "error",
      showConfirmButton: false,
      timer: 1000,
      ruta: 'perfil'
    });
  }
});
```

Con las operaciones CRUD abarcadas, ahora podemos dar comienzo a la generación de los HTML, para empezar a darle más forma a lo que buscamos abarcar en nuestro avance del proyecto.

-Creación de HTML para iniciar sesión

Tras unas conversaciones con el UI/UX, se concordó que se hiciera una página para iniciar sesión, para registrarse, y para demostrar y editar los datos que tendría el usuario; por ello, daremos comienzo con la creación del HTML para iniciar sesión, y dicho HTML será donde se dirigen apenas ingresen al servidor.

Dicho ya HTML, tendrá por nombre 'sesion.html', ahora, para la creación del HTML, se notó que se usó la misma estructura, tanto para encabezado como para el pie de página, es por ello por lo que se tomó la base de los anteriores HTML que han sido creados en la primera y segunda actividad.

```

Fronted > < sesson.html > ...
1  <!DOCTYPE html>
2  <html lang="es-MX">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="style.css">
8      <title>Rhymes Music - Login</title>
9  </head>
10
11 <body>
12     <div class="contenedor">
13         <header id="home">
14             <div class="fondonegro"></div>
15             <span class="nombre">Rhymes Music</span>
16         </header>
17
18         <footer>
19             <div class="redessociales">
20                 <a href="https://www.facebook.com"><i class="fa-brands fa-facebook-f"></i></a>
21                 <a href="https://www.tiktok.com"><i class="fa-brands fa-tiktok"></i></a>
22                 <a href="https://www.x.com"><i class="fa-brands fa-x-twitter"></i></a>
23                 <a href="https://www.youtube.com"><i class="fa-brands fa-youtube"></i></a>
24             </div>
25             <span class="copyright">© 2025 Rhymes Music. Todos los derechos reservados.</span>
26         </footer>
27     </div>
28     <script src="https://kit.fontawesome.com/9551891e69.js" crossorigin="anonymous"></script>
29     <script src="/app.js"></script>
30
31
32 </body>
33
34 </html>

```

Recordemos que estos archivos estarán en una carpeta llamada 'Frontend', la cual esperamos que contenga los HTML, los CSS, los JavaScript y las posibles imágenes que puedan llegar a ser utilizadas.

Podemos ver en el Head, que tiene el título de login, dando una especificación al usuario respecto a qué páginas se encuentra; a la par se le da un encabezado con el nombre del sitio y un Good que contendría las redes sociales como Facebook, TikTok, X y YouTube.

Y junto a eso podemos ver que tiene scripts, donde buscamos en enlazar un archivo JS para manejar la interactividad.

Con esa base ya obtenida, ahora agregamos el formulario para poder iniciar sesión, y dicho formulario es el que enviará los datos para la ruta '/sesión' que tiene el método POST, siendo así que empecemos a abarcar la ruta CRUD, específicamente la de READ; a la par le damos un título en el que se especifique que es para iniciar sesión, y después con ello, empezamos el general formulario en donde requerimos el campo de usuario y contraseña, en donde esos datos son obligatorios para ser ingresados, pero en el campo de contraseña hacemos que se oculten los caracteres debido al type que le colocamos, y después de ello generamos un botón que podrá enviar el formulario.

```

<div class = loginForm>
    <h1 id="titulo">Inicia Sesión</h1>
    <form action='/sesion' method="POST">
        <label for="usuario">Usuario</label>
        <input type="text" name="usuario" id="usuario" placeholder="Introduce tu usuario" required>
        <label for="pass">Contraseña</label>
        <input type="password" name="pass" id="pass" placeholder="Introduce tu contraseña" required>
        <input type="submit" class="loginBtn" value="Iniciar sesión">
    </form>
    <a href="/registrar.html" class="registrarBtn">Registrarse</a>
</div>

```

Después del formulario, agregamos un botón que sea específicamente para poder dirigir al usuario a la página para que se pueda registrar haciendo uso de un enlace para dicha página a la cual se debería de dirigir, y con ello empezaremos abarcar la operación CREATE de CRUD.

-Generación de HTML para registro de usuario

Para la generación del HTML para registrarse, damos primero la creación de dicho archivo dentro de la carpeta 'fronted', y dicho archivo tendrá por nombre 'registrar.html', dando así una dirección válida al botón de registrarse en cuál está en la página de iniciar sesión.

```

Fronted > registrarse.html ...
1  <!DOCTYPE html>
2  <html lang="es-MX">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="style.css">
8      <title>Rhymes Music - Registro</title>
9
10 </head>
11 <body>
12     <div class="contenedor">
13         <header id="home">
14             <div class="fondonegro"></div>
15             <span class="nombre">Rhymes Music</span>
16         </header>
17
18         <footer>
19             <div class="redessociales">
20                 <a href="https://www.facebook.com"><i class="fa-brands fa-facebook-f"></i></a>
21                 <a href="https://www.tiktok.com"><i class="fa-brands fa-tiktok"></i></a>
22                 <a href="https://www.x.com"><i class="fa-brands fa-x-twitter"></i></a>
23                 <a href="https://www.youtube.com"><i class="fa-brands fa-youtube"></i></a>
24             </div>
25             <span class="copyright">© 2025 Rhymes Music. Todos los derechos reservados.</span>
26         </footer>
27     </div>
28     <script src="https://kit.fontawesome.com/9551891e69.js" crossorigin="anonymous"></script>
29     <script src="/app.js"></script>
30
31 </body>
32
33 </html>

```

Como comentamos en el anterior requerimiento, hacemos una base que tienen la mayoría de los HTML, es que estábamos manejando, siendo la única diferencia, el título, el cual porta el nombre de registro.

Mismamente contiene un encabezado y pie de página en el cual abarcan tanto el nombre del sitio como las redes sociales que se manejan.

Obteniendo la base del HTML de registro, ahora podemos incluir el formulario que estará empeñado en el registro, el cual enviará los datos

a la ruta '/registrarse' con el método POST, siendo así que abarquemos la operación CREATE de CRUD, para la creación de dicho formulario fue necesario generar varios campos como para ingresar el usuario, el email y la contraseña, si es así que esto sean obligatorios en su ingreso, y por parte del email se tiene contemplado que tenga el formato con @; una vez hecho, eso se genera un botón, el cual tendrá por nombre crear usuario. Generando así, los datos necesarios que deberá enviar el formulario para la ruta, donde se crea el nuevo usuario en la base de datos.

```

<div class = "registroForm">
    <h1 id="registroTitulo">Crear un usuario</h1>
    <form action='/registrarse' method="POST">
        <label for="usuario">Usuario</label>
        <input type="text" name="usuario" id="usuario" placeholder="Crea tu usuario" required>
        <label for="email">Correo</label>
        <input type="email" name="email" id="email" placeholder="Introduce tu email" required>
        <label for="pass">Contraseña</label>
        <input type="password" name="pass" id="pass" placeholder="Introduce tu contraseña" required>
        <input type="submit" class="registrarBtn" value="Crear usuario">
    </form>
</div>

```

-Mensajes de éxito, error y advertencia

He comentado a lo largo de los requerimientos que se usan mensajes para dar a conocerle al usuario lo que ha ocurrido, y dichos mensajes son posibles debido a que se integró SweetAlert2 y EJS como motor de plantillas, obteniendo así, alertas dinámicas para las páginas que se han creado.

Recordemos que en 'server.js' si implementó el motor de plantillas y que serían obtenidas en una carpeta llamada 'views', dicha carpeta contendría los archivos '.ejs', cada uno de los archivos '.ejs', portan la misma estructura, siendo sólo el cambio en el título en el encabezado, donde define que página es en la que se debe encontrar.

```

//Motor de plantillas
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

```

```

<!DOCTYPE html>
<html lang="es-MX">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Rhymes Music - Login</title>
</head>

<body>
    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
    <% if(typeof alert != "undefined"){ %>
        <script>
            Swal.fire({
                title: '<%= alertTitle %>',
                text: '<%= alertMessage %>',
                icon: '<%= alertIcon %>',
                showConfirmButton: '<%= showConfirmButton %>',
                timer: '<%= timer %>'
            }).then(()=>{
                | window.location='/<%= ruta %>'
            })
        </script>
    <% } %>
</body>

</html>

```

Podemos ver como en el Body se contiene un script, el cual hace la integración del SweetAlert2, y después de ello se tiene otro script en el cual contiene el código para llamar una función llamada Swal.fire(), siendo esto, una función que nos ayudó a mostrar alertas personalizadas, las cuales pueden contener un título, texto, ícono, botones y el tiempo en el que se mostrará dicha alerta, siendo así que también con ello, pueda ser redirija a alguna ruta después de cerrar la alerta. Esto ocurre en la mayoría de las secciones, donde se generó un mensaje, ya sea de éxito, error y advertencia.

-Generación de EJS para mostrar el perfil del usuario

Con la generación de los HTML para iniciar sesión y registrarse, ahora nos empeñamos en generar el archivo donde se mostrarán los datos del usuario y a la par, el poder editarlos.

```

> ◊ perfil.ejs > ...
<!DOCTYPE html>
<html lang="es-MX">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Rhymes Music - Perfil</title>
</head>

<body>
    <div class="contenedor">
        <header id="home">
            <div class="fondonegro"></div>
            <span class="nombre">Rhymes Music</span>
        </header>

        <footer>
            <div class="redessociales">
                <a href="https://www.facebook.com"><i class="fa-brands fa-facebook-f"></i></a>
                <a href="https://www.tiktok.com"><i class="fa-brands fa-tiktok"></i></a>
                <a href="https://www.x.com"><i class="fa-brands fa-x-twitter"></i></a>
                <a href="https://www.youtube.com"><i class="fa-brands fa-youtube"></i></a>
            </div>
            <span class="copyright">© 2025 Rhymes Music. Todos los derechos reservados.</span>
        </footer>
    </div>

    <script src="https://kit.fontawesome.com/9551891e69.js" crossorigin="anonymous"></script>
    <script src="/app.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
    <% if(typeof alert != "undefined"){ %>
        <script>
            Swal.fire({
                title: '<%= alertTitle %>',
                icon: '<%= alertIcon %>',
                showConfirmButton: '<%= showConfirmButton %>',
                timer: '<%= timer %>'
            }).then(()=>{
                | window.location='/<%= ruta %>'
            })
        </script>
    <% } %>
</body>

</html>

```

Y para ello se decidió que el archivo sea '.ejs'; dicho archivo tiene por nombre, 'perfil', y se les ha agregado una base como en los anteriores HTML de iniciar sesión y registrarse, en los cuales se les da un encabezado y un pie de página, lo único que los diferencia de los otros HTML, es que aquí es posible incluir el script, el cual abarca los mensajes dinámicos que se buscan ofrecer al usuario, y dicho archivo estará en la carpeta 'views'.

Teniendo la base del archivo 'perfil', ahora podemos empezar a abarcar los apartados donde se deberán mostrar los datos del perfil, la edición de datos y la barra lateral, en donde se mostrarán las acciones que deberá aportar 'perfil.ejs'.

-Barra de secciones en 'perfil.ejs'

Para la barra de secciones, es necesario generar un contenedor para dicha sección, y empezamos a agregar un botón de perfil el cual es requerido por parte de UI/UX siendo un círculo gris, después de ello se generan botones ya sean para la sección de Perfil, Ajustes y Cerrar sesión.

Para la funcionalidad de dichos botones, es necesario ir al archivo 'app.js', el cual contendrá un evento que maneje la interacción con los botones en el perfil del usuario, ahora, podemos observar que es un 'document.addEventListener('DOMContentLoaded')', dicho evento solo ocurre cuando

```
<div class="perfilLateral">
  <div class="perfilBtn"></div>
  <button id="btnPerfil" class="botonPerfil">Perfil</button>
  <button id="btnAjustes" class="botonPerfil">Ajustes</button>
  <button id="btnCerrar" class="botonPerfil">Cerrar sesión</button>
</div>
```

el documento HTML ha sido cargado y está listo el DOM para ser manipulado; dentro del evento podemos ver como creamos una constante para almacenar la referencia de cada uno de los botones, y del área donde se mostrar ya sea la información del usuario o la actualización de los datos.

```
//Para cuando se haga click en alguno de los botones que se encuentran el perfil.ejs
document.addEventListener('DOMContentLoaded', () => {
  console.log('Documento cargado'); //Verifica que este el documento cargado

  // Verifica si los botones existen
  const btnPerfil = document.getElementById('btnPerfil');
  const btnAjustes = document.getElementById('btnAjustes');
  const btnCerrar = document.getElementById('btnCerrar');
  const infoPerfil = document.getElementById('infoPerfil');
  const actuInfo = document.getElementById('actuInfo');

  console.log(btnPerfil, btnAjustes, btnCerrar, infoPerfil, actuInfo); //Verifica que esten

  //Marca en consola que no se encontraron
  if (!btnPerfil || !btnAjustes || !btnCerrar || !infoPerfil || !actuInfo) {
    console.error('Uno o más elementos no se encontraron en el DOM.');
    return;
  }

  //Funcionalidad de botones, incluyendo un mensaje para que se muestre en la
  //consola que estén siendo usados
  btnPerfil.addEventListener('click', () => {
    console.log('Botón "Perfil" clickeado');
    infoPerfil.style.display = 'block';
    actuInfo.style.display = 'none';
  });

  btnAjustes.addEventListener('click', () => {
    console.log('Botón "Ajustes" clickeado');
    actuInfo.style.display = 'block';
    infoPerfil.style.display = 'none';
  });

  btnCerrar.addEventListener('click', () => {
    window.location.href = '/';
  });
});
```

estructura, donde especificamos que debe ocurrir si se presiona uno, por ejemplo, en el botón de perfil, si este es presionado, se mostrara en consola un mensaje notificándolo, y se hará un cambio en el CSS donde el elemento ya sea de infoPerfil y actuInfo cambien del valor, siendo un cambio para mostrar y ocultar un apartado específico. El botón distinto, es el de cerrar sesión, donde solamente redirige a la página de iniciar sesión.

-Apartado de Perfil en 'perfil.ejs'

Para el apartado de perfil, solo se hizo un contenedor que tendrá la información y a la par se le da el ID que es clave para que se muestre u oculte; con ello, lo que deberá contener son etiquetas que abarcan la información de nombre y email, y son los datos tomados de la información del usuario de la base de

datos, antes aseguramos que la variable tenga información, en caso de que no contenga, pues se da a conocer con un mensaje que diga 'No disponible'.

```
<!-- Apartado de Perfil -->
<div class="infoPerfil" id="infoPerfil">
  <p><strong>Nombre</strong><br><%= nombre || 'No disponible' %></p>
  <p><strong>Email</strong><br><%= email || 'No disponible' %></p>

  <button id="btnEliminar" class="botonPerfil">Eliminar Cuenta</button>
</div>
```

Junto a eso, está el botón de eliminar, donde hacemos uso de JavaScript para su funcionalidad.

-Funcionalidad de botón para eliminar usuario en perfil.ejs

Para la funcionalidad el botón fue necesario hacer uso de un evento en el cual, si se hace click, ocurra la siguiente función, la cual manda un mensaje de confirmación haciendo uso de SweetAlert, que le comenta al usuario si está seguro de querer hacer dicha acción, dándole dos botones, ya sean para cancelar o confirmar.

```
//Para cuando se haga click en el boton de eliminar en perfil.ejs
document.getElementById("btnEliminar").addEventListener("click", function() {
  Swal.fire({
    title: "Estás seguro?",
    text: "Esta acción no se puede deshacer.",
    icon: "warning",
    showCancelButton: true,
    confirmButtonColor: "#d33",
    cancelButtonColor: "#3085d6",
    confirmButtonText: "Sí, eliminar",
    cancelButtonText: "Cancelar"
  }).then(result) => {
    if (result.isConfirmed) {
      fetch("/eliminarUsuario", {
        method: "DELETE",
        headers: {
          "Content-Type": "application/json"
        }
      })
      .then(response => response.json())
      .then(data => {
        Swal.fire({
          title: data.success ? "Eliminado" : "Error",
          text: data.message,
          icon: data.success ? "success" : "error",
          timer: 2000,
          showConfirmButton: false
        }).then(() => {
          if (data.success) {
            window.location.href = "/"; //Mandara a sesion
          } else {
            window.location.href = "/";
          }
        });
      });
    }
  .catch(error => { //Aunque haya un error, se borra correctamente en la base de datos
    Swal.fire({
      title: "Exitoso",
      text: "Se eliminó la cuenta.",
      icon: "success",
      timer: 3000,
      showConfirmButton: false
    });
    window.location.href = "/";
    console.error("Eliminada la cuenta:", error);
    setTimeout(() => {window.location.href = "/"}, 3000);
  });
});
});
```

-Apartado de Ajustes en 'perfil.ejs'

Para el apartado de Ajustes se generó un formulario donde el atributo action abarca la ruta que debe ser usada para enviar los datos del formulario, siendo dirigidos a '/actualizarPerfil', pero se le agrega el parámetro '_method=PATCH' se utilizará para simular el método HTTP PATCH porque ciertos navegadores solo soportan GET y POST; aunque en el método se especifica que el formulario se enviará en método POST y el valor '_method=PATCH'; hace que se trate como una solicitud PATCH en el servidor.

Si el usuario llega a confirmar, se hace la solicitud al DELETE para eliminar el usuario, donde utiliza la ruta '/eliminarUsuario', siendo así que abarquemos la operación de DELETE de CRUD, tras hacer la solicitud se maneja la respuesta haciendo que se convierta en JSON devolviendo los datos por el servidor confirmando si fue exitosa o no la operación. Con eso hecho, se muestra una alerta con el resultado de la eliminación y después si data.success es true o false se muestra la alerta con el título de eliminado o error, y una vez se cierra la sesión la página lo redirige a la página de iniciar sesión.

Ahora, ocultamos el campo que nos ayuda a simular el método PATCH al incluirlo en el formulario nos ayuda a que en Backend nos permita que se interprete como una solicitud PATCH; y con ello damos comienzo a crear los campos para que se ingresen nuevo nombre, email y contraseña, especificando en el input sobre el tipo y cuales son obligatorios. Junto a eso, está el botón para enviar el formulario, que se encargara después de ser enviada la información al servidor con los datos ingresados por el usuario.

```
<div class="actuInfo" id="actuInfo">
  <form action="/actualizarPerfil?_method=PATCH" method="POST" id="updateForm">
    <input type="hidden" name="_method" value="PATCH"> <!-- Simula el método PATCH -->

    <label for="newName">Nuevo Nombre:</label>
    <input type="text" id="newName" name="newName" value="<% nombre %>" required>

    <label for="newEmail">Nueva Email:</label>
    <input type="email" id="newEmail" name="newEmail">

    <label for="newPassword">Nueva Contraseña:</label>
    <input type="password" id="newPassword" name="newPassword">

    <button type="submit" id="btnActu">Actualizar</button>
  </form>
</div>
```

-Funcionalidad del botón de actualizar datos en 'perfil.ejs'

Teniendo encuentra donde se encuentra el botón de actualizar, ahora nos centraremos en ver que nos ayuda en que sea funcional dicho botón.

Mismamente que la funcionalidad del botón para eliminar, hacemos uso de un even listener donde escuchamos el evento cuando se presiona el botón de actualizar, y con eso hecho evitamos que se envié

el formulario de manera tradicional para empezar a manejar la solicitud, ahora, con eso empezamos a obtener los datos de los campos que fueron ingresados haciendo que se guarden y después hacemos la solicitud al método PATCH, siendo así que abarquemos la última operación CRUD, la de UPDATE, cuando se hace la solicitud se deberán enviar los datos con formato JSON. Cuando se enviaron los datos, se empieza a procesar la respuesta del servidor, y se empieza a manejar la respuesta con un if, el cual verifica si todo salió de forma exitosa y si llegara ser así, se actualizarían los datos en la página con los nuevos valores para después redirigir a

```
//Botón para actualizar los datos del usuario
document.getElementById("updateForm").addEventListener("submit", async (e) => {
  e.preventDefault(); //Evita el envío del formulario

  const newName = document.getElementById("newName").value;
  const newPassword = document.getElementById("newPassword").value;
  const newEmail = document.getElementById("newEmail").value;

  const response = await fetch('/actualizarPerfil', {
    method: 'PATCH',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ newName, newPassword, newEmail }),
  });
  const data = await response.json();

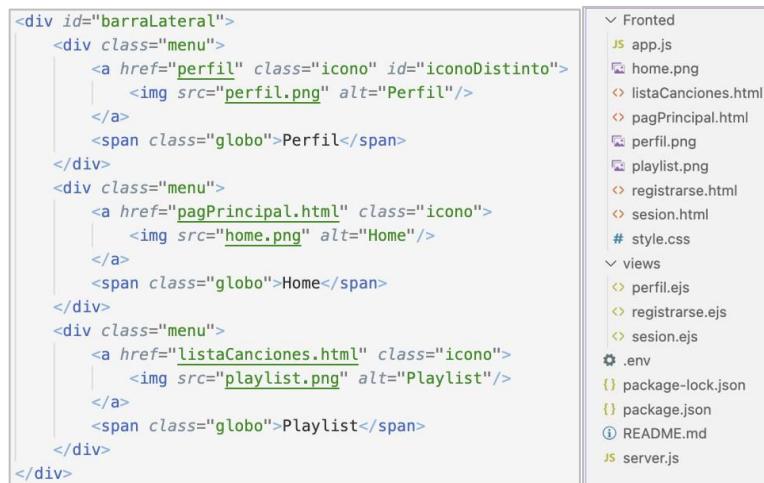
  if (data.success) {
    alert(data.message); //Mensaje de que todo salio correcto
    //Actualizó los datos de nombre y email
    document.getElementById("username").textContent = data.nombre;
    document.getElementById("userEmail").textContent = data.email;
    //Redirige a perfil con los datos actualizados
    window.location.href = 'perfil';
  } else {
    alert(data.message); //Mensaje de error
  }
});
```

'perfil.ejs', en caso de que no sea así, en el else se le daría conocer al usuario con un mensaje que hubo un error en la actualización.

-Creación de Barra lateral

Tras la explicación de lo que conllevo en el Backend y Frontend, ahora tratare de mostrar como abarque la creación de la barra lateral que se me fue pedida por UI/UX.

Podemos ver en el código que se hace un contenedor principal que abarca toda la barra lateral y después de ello se hace una sección de menú que es usado para cada elemento, podemos ver cómo hacemos uso de imágenes y de texto que tendrá mayor forma en el CSS, pero en total deben ser creados 3 ítems y deben redirigir a su respectiva página.



The screenshot shows a code editor with two panes. The left pane displays the following HTML code:

```
<div id="barraLateral">
  <div class="menu">
    <a href="#" class="icono" id="iconoDistinto">
      
    </a>
    <span class="globo">Perfil</span>
  </div>
  <div class="menu">
    <a href="#" class="icono">
      
    </a>
    <span class="globo">Home</span>
  </div>
  <div class="menu">
    <a href="#" class="icono">
      
    </a>
    <span class="globo">Playlist</span>
  </div>
</div>
```

The right pane shows the directory structure of the project:

- Fronted
 - JS app.js
 - home.png
 - listaCanciones.html
 - pagPrincipal.html
 - perfil.png
 - playlist.png
 - registrar.html
 - sesion.html
 - # style.css
 - views
 - perfil.ejs
 - registrar.ejs
 - sesion.ejs
 - .env
 - { package-lock.json
 - { package.json
 - ① README.md
 - JS server.js

Podemos notar como las imágenes son implementadas en el fronted, y las imágenes añadidas son las siguientes:



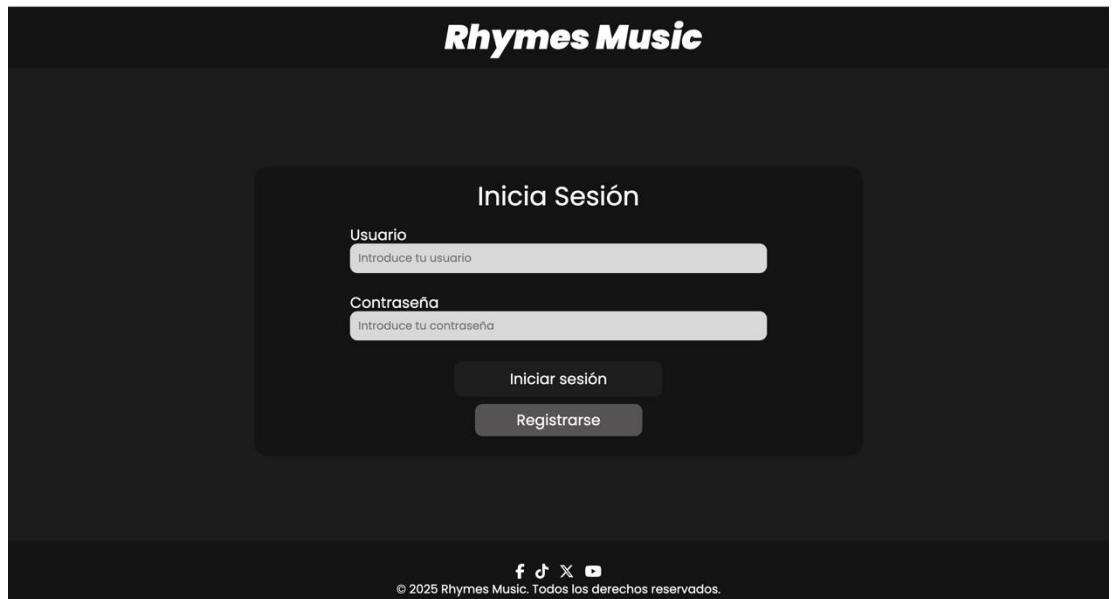
Dicha barra, solo podrá ser vista en los archivos 'pagPrincipal', 'Perfil' y 'listaCanciones'.

-Diseño de UI/UX y responsive en los HTML y EJS

Tras ver lo que fue creado, ahora veremos como quedó el diseño en las nuevas páginas, siguiendo el diseño dado por el UI/UX.

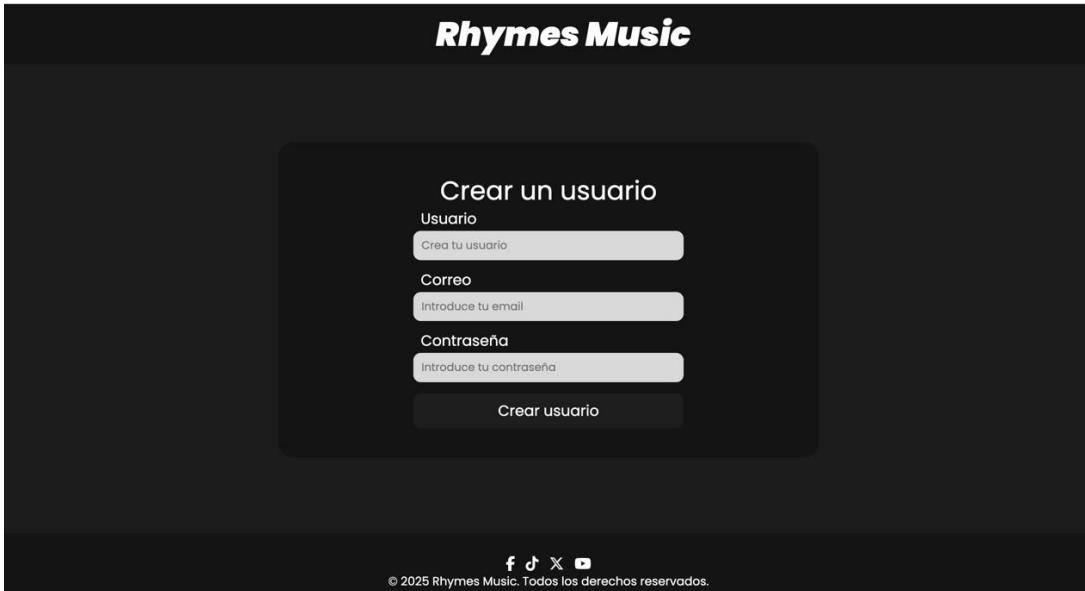
Iniciar sesión

Para la página de iniciar sesión, se da un cambio de color a los botones si el cursor esta encima y el botón de registrarse dirige a donde corresponde.



Registrarse

Para esta página, se podrá ver como el botón se cambia de color si el cursor está encima.



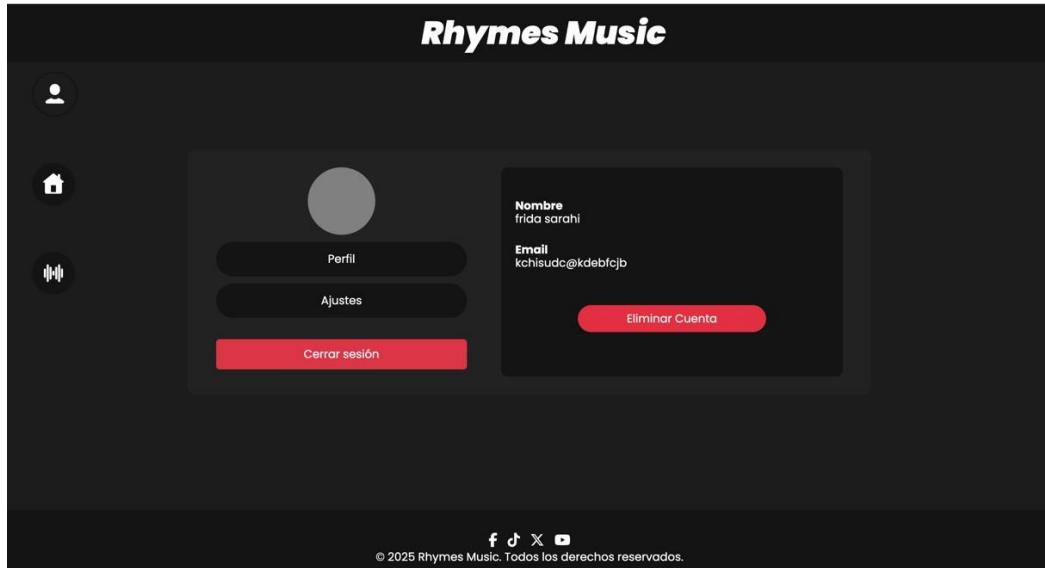
Barra lateral

Para la barra lateral, se llevó a cabo que puedan girar si es que el cursor está encima y a la par muestre un globo de texto, podemos ver en las siguientes imágenes como está en el lado izquierdo y cumple el que gire y muestre el globo de texto.

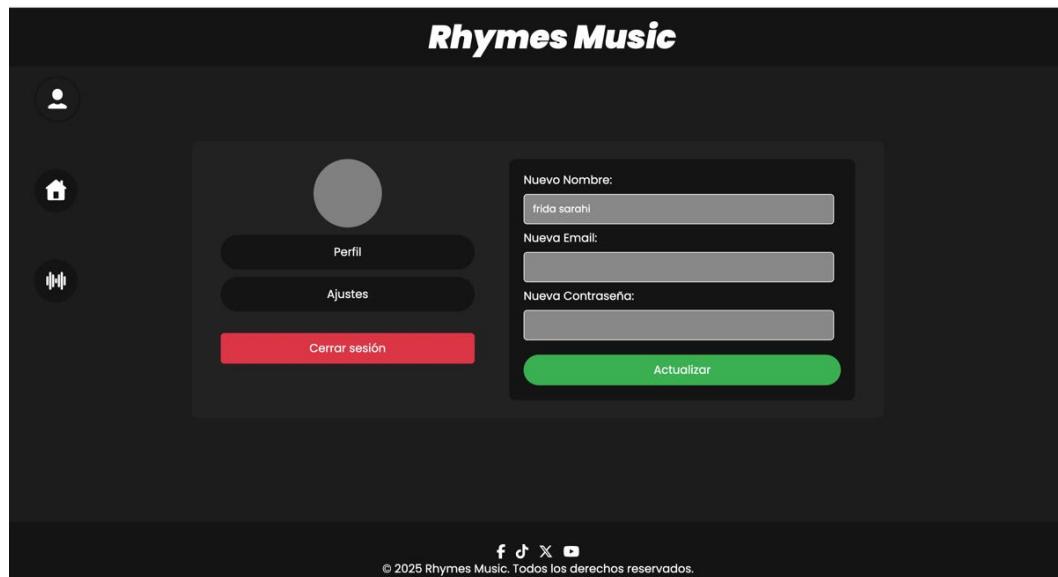
A screenshot of the Rhymes Music website. On the left side, there is a vertical sidebar with three icons: a profile picture, a house, and a sound wave. The main content area has a header "Rhymes Music". Below the header, there is a box containing text about the company's profile. To the right of the sidebar, there is a section titled "Proyectos" (Projects) with a sub-section titled "Un Verano Sin Ti". A small image of Bad Bunny is shown in the background of this section. At the bottom left of the page, there is a URL "localhost:3000/perfil".

Perfil del usuario

Para la página de perfil, podemos ver cómo nos muestra las secciones, siendo que si presionas el botón de perfil, te muestre el área de perfil, donde muestra los datos del usuario y el botón de eliminar.



Y después, esta que si presionas el botón de actualizar, se muestre ese apartado donde este la posibilidad de ingresar texto en los campos correspondientes.

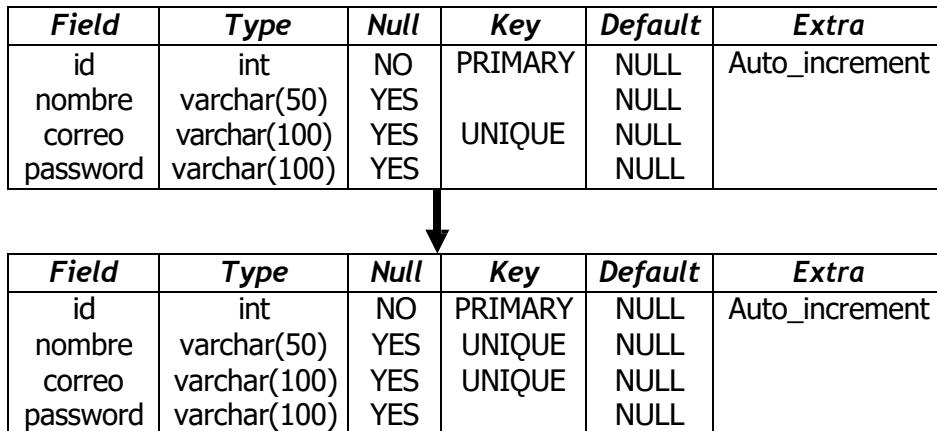


Todos los botones mostrados en la página son funcionales, respecto a lo responsive de la página, hubo inconvenientes respecto al tiempo, siendo así que se deba pedir ayuda a mi compañero Pedro De León para que abarque esa área a profundidad, cabe aclarar que el diseño estaba muy complejo para las habilidades que tenemos al momento.

Inconvenientes/Problemas y Solución

Se dio a conocer una problemática gracias al QA, donde notifico que hubo un problema a la validación de que solo se ingrese un usuario y sea único, debido a eso, se tuvo que corregir el código en ciertas áreas, empezando con la solución de corregir los datos de la tabla.

Notemos a continuación como cambian los datos de la tabla:



The diagram shows two tables representing database schemas. An arrow points from the top table to the bottom table, indicating a transformation or correction.

Field	Type	Null	Key	Default	Extra
id	int	NO	PRIMARY	NULL	Auto_increment
nombre	varchar(50)	YES		NULL	
correo	varchar(100)	YES	UNIQUE	NULL	
password	varchar(100)	YES		NULL	

Field	Type	Null	Key	Default	Extra
id	int	NO	PRIMARY	NULL	Auto_increment
nombre	varchar(50)	YES	UNIQUE	NULL	
correo	varchar(100)	YES	UNIQUE	NULL	
password	varchar(100)	YES		NULL	

Quedando así la creación de la base de datos de forma local corregida:

```
mysql> CREATE TABLE usuarios (    id INT AUTO_INCREMENT PRIMARY KEY,    nombre VARCHAR(50) UNIQUE,    correo VARCHAR(100) UNIQUE,    password VARCHAR(100));  
Query OK, 0 rows affected (0,02 sec)
```

En la corrección del código solo hubo cambio en el código donde abarcamos la operación de registrar un nuevo usuario. Recordemos que una obtenemos los datos mandados por el formulario, y de ahí empezamos con la modificación donde hacemos uso de un Try, que contiene la encriptación de la contraseña, la consulta a la base de datos donde hacemos uso de 'connection.query()' donde buscamos insertar el nuevo usuario con los datos dados.

```
app.post('/registrarse', async(req, res)=>{  
  try{  
    let passwordHash = await bcryptjs.hash(pass, 8);  
    connection.query('INSERT INTO usuarios (nombre, correo, password) VALUES (?,?,?)', [usuario, email, passwordHash],(error, results) => {  
      if (error) {  
        console.log(error);  
        let alertMessage = "Ocurrió un error, favor de intentar de nuevo";  
        //Verifica que el error es por entrada de datos duplicados  
        if (error.code === 'ER_DUP_ENTRY') {  
          //Revisamos en el mensaje del error que campo esté duplicado  
          if (error.sqlMessage.includes("nombre")) {  
            alertMessage = "El nombre de usuario ya existe, intenta con otro.";  
          } else if (error.sqlMessage.includes("correo")) {  
            alertMessage = "El correo ya está registrado, favor de intentar con otro.";  
          }  
        }  
      }  
    })  
  }  
})
```

Después de ello, se usa un if para manejar el error en la consulta donde genera una alerta para notificar al usuario y a la par dando a conocer el error en la consola, dentro de ese if, habrá otro que manejará los datos duplicados, donde abarca el error de datos duplicados por MySQL siendo dado este error debido a que la columna porta la restricción de único, y si el error llegara a ser porque el nombre o el email fueron repetidos, se le notificaría al usuario específicamente cual se repitió.

```

        return res.render('sesion', {
            alert: true,
            alertTitle: "Error",
            alertMessage: alertMessage,
            alertIcon: "warning",
            showConfirmButton: true,
            timer: 1500,
            ruta: "registrarse"
        });
    } else {
        return res.render('registrarse', {
            alert: true,
            alertTitle: "Registro",
            alertMessage: "¡Registro exitoso!",
            alertIcon: "success",
            showConfirmButton: false,
            timer: 1500,
            ruta: ''
        });
    }
}

```

Tras eso, se da una respuesta en caso de error, donde muestra una alerta en la página de sesión, donde le da a conocer al usuario el error detectado y siendo una alerta de warning.

Y también puede dar una respuesta de éxito, donde solo da a conocer al usuario con mensaje que el registro fue exitoso y sin inconvenientes.

Para finalizar la corrección del código, se maneja el bloque catch donde manejamos el error del servidor, dando a conocer que ocurrió un problema en el proceso de encriptación

```

        }
    }catch(err){
        console.error("Error al encriptar la contraseña:", err);
        return res.render('sesion', {
            alert: true,
            alertTitle: "Error",
            alertMessage: "Ocurrió un error en el servidor, por favor intente de nuevo.",
            alertIcon: "error",
            showConfirmButton: true,
            timer: 1500,
            ruta: "registrarse"
        });
    }
}

```

Referencias

<https://www.oracle.com/mx/mysql/what-is-mysql/#:~:text=MySQL%20es%20r%C3%A1pidamente%20confiable%20ampliable,de%20funciones>

Pedro De León QA

Matriz de requerimientos: https://utmedu-my.sharepoint.com/:x/g/personal/al03103352_tecmilenio_mx/ESoSCTs_65EmKW815U4F5EBxi8Ch_fn5b9q8JoKJfudug?e=LIVGBF

ID	Categoría	Descripción de la prueba	Prioridad	Responsable	Estatus			
QA-1	Frontend	El sitio web es responsive en móvil.	Alta	Dev	Verde			
QA-2	Frontend	El sitio web es responsive en computadora.	Alta	Dev	Verde			
QA-3	Frontend	El sitio web es responsive de manera general y tiene un correcto funcionamiento.	Alta	Dev	Verde			
QA-4	Frontend	Los botones funcionan correctamente.	Alta	Dev	Verde			
QA-5	Frontend	No permite enviar campos vacíos en ninguna de las páginas.	Alta	Dev	Verde			
QA-6	Frontend	Redirecciona adecuadamente.	Alta	Dev	Verde			
QA-7	Backend	Encripta la contraseña.	Media	Dev	Verde			
QA-8	Frontend	No permite campos negativos en ninguna de las páginas.	Alta	Dev	Verde			
QA-9	Frontend	Las validaciones (como la de email) están validadas en todas las páginas.	Media	Dev	Verde			
QA-10	Frontend + Backend	Los datos se guardan en la base de datos sin problemas.	Alta	Dev	Verde			
QA-11	Frontend	El texto no sobresale de los cuadros en ninguna de las páginas.	Media	Dev	Verde			
QA-12	Frontend + Backend	La página de perfil muestra la información correcta.	Alta	Dev	Verde			
QA-13	Frontend	Todo funciona a la perfección en perfil.	Media	Dev	Verde			
QA-14	Frontend + Backend	El botón de cerrar sesión funciona adecuadamente.	Media	Dev	Verde			
QA-15	Frontend	El botón de ajustes abre de manera adecuada una página para editar los datos.	Baja	Dev	Verde			
QA-16	Frontend + Backend	Al editar y guardar los cambios en perfil, estos se mantienen y se actualizan en la BD.	Alta	Dev	Verde			
QA-17	Backend	Se crea el usuario de manera correcta.	Alta	Dev	Verde			
QA-18	Backend	Verifica que no estén los datos repetidos al crear el usuario.	Alta	Dev	Verde			

NOTA: Los issues que están en estatus amarillo fueron corregidos en la branch QA-Responsive, pero cambia el diseño, esto por problemas de tiempo, pero no afecta el funcionamiento.

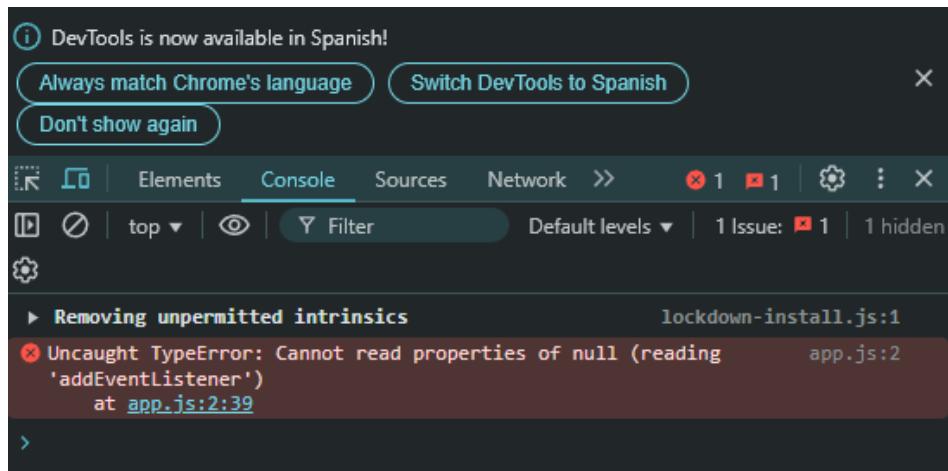
ID	Issue	Descripción a la que está relacionado	Responsable	Estatus
QI-1	Se testeó la página listaCanciones.html, el texto sobresale de los campos de texto.	El texto no sobresale de los cuadros en ninguna de las páginas.	QA	Verde
QI-2	Se puede mover la página de manera horizontal.	El sitio web es responsive de manera general y tiene un correcto funcionamiento.	QA	Verde
QI-3	El div de crear usuario te sigue.	El sitio web es responsive de manera general y tiene un correcto funcionamiento.	QA	Verde
QI-4	Se puede crear usuarios con nombres de usuarios repetidos.	Verifica que no estén los datos repetidos al crear el usuario.	Dev	Verde
QI-5	El botón de eliminar cuenta causa errores.	Los botones funcionan correctamente.	Dev	Verde
QI-6				
QI-7				
QI-8				
QI-9				
QI-10				
QI-11				
QI-12				
QI-13				
QI-14				
QI-15				
QI-16				

NOTA: Los issues que están en estatus amarillo fueron corregidos en la branch QA-Responsive, pero cambia el diseño, esto por problemas de tiempo, pero no afecta el funcionamiento.

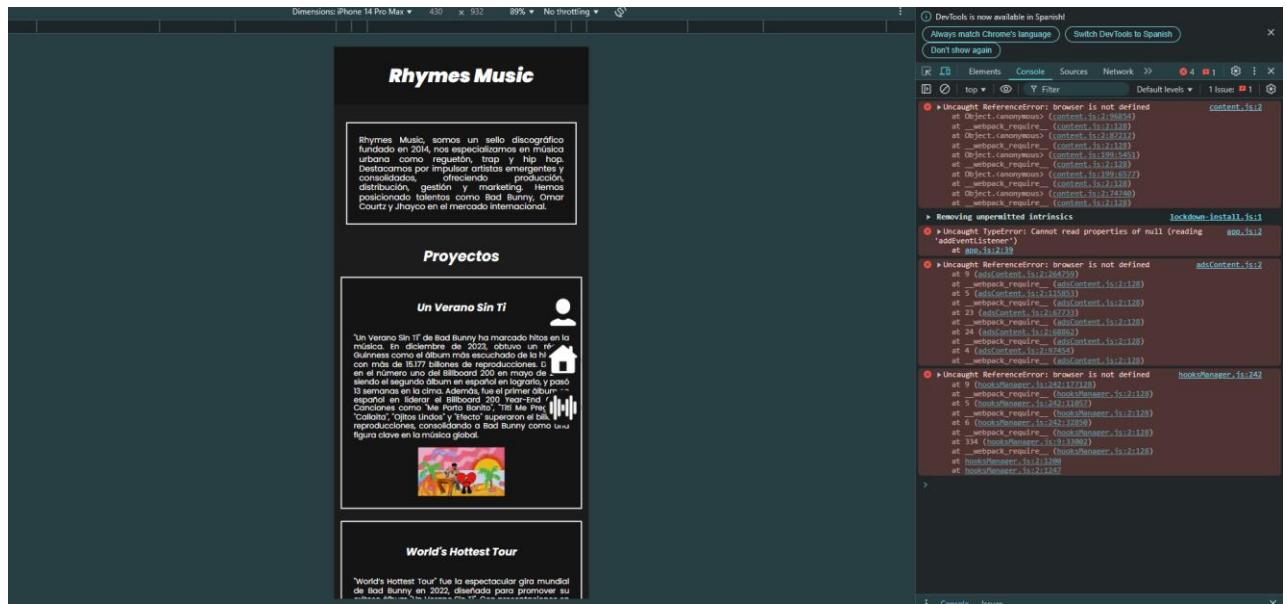
Errores generales

Durante las pruebas realizadas, se identificó un error general en la consola del navegador: "**Uncaught TypeError: Cannot read properties of null (reading 'addEventListener')**" en el archivo app.js línea 2. Este error suele ocurrir cuando se intenta agregar un event listener a un elemento que no existe o no ha sido correctamente cargado en el DOM.

A pesar de este error, se verificó que no afecta el funcionamiento general de la página, ya que todas las interacciones esperadas, incluyendo los botones y otras funcionalidades clave, operan de manera correcta. Sin embargo, se recomienda revisar el código para corregir este problema y evitar posibles inconvenientes en futuras implementaciones o actualizaciones.



En la consola aparecen varios errores, pero que no afectan cómo funciona la página. Estos errores salen en la página principal. El primero es Uncaught ReferenceError: browser is not defined, lo que significa que el código está tratando de usar browser, pero esta variable no está definida, probablemente porque falta alguna librería o se está usando en un entorno donde no aplica. También sale el error Uncaught TypeError: Cannot read properties of null (reading 'addEventListener'), que indica que se intenta agregar un evento a un elemento que no existe en el DOM en ese momento. Esto puede pasar si el script corre antes de que la página termine de cargar o si el selector que busca el elemento está mal escrito. Para solucionar esto, se puede revisar que todas las librerías estén bien instaladas, que los scripts se ejecuten después de que el DOM cargue por completo usando DOMContentLoaded, y que los selectores estén bien definidos para que apunten a los elementos correctos.



QA-1 El sitio web es responsivo en móvil. **QA-2** El sitio web es responsivo en computadora. **QA-3** El sitio web es responsiva de manera general y tiene un correcto funcionamiento.

Estos puntos no se cumplieron en la rama principal del proyecto debido al poco tiempo disponible para realizar los ajustes necesarios. Por esta razón, se creó una rama exclusiva llamada QA-Responsive, enfocada en adaptar el sitio a diferentes dispositivos. Sin embargo, en esta rama, el diseño UI/UX se ve comprometido, ya que algunos elementos pierden su alineación y la estética general del sitio se rompe.

A pesar de estos inconvenientes visuales, la rama QA-Responsive busca cumplir con los requisitos de responsividad, asegurando que el sitio se adapte tanto a dispositivos móviles como a computadoras. Aunque el diseño no es perfecto, la funcionalidad básica, como la operación de los botones, se mantiene intacta en esta versión.

ASÍ SE VE EN LA RAMA MAIN

Rhymes Music



Rhymes Music, somos un sello discográfico fundado en 2014, nos especializamos en música urbana como reguetón, trap y hip hop. Destacamos por impulsar artistas emergentes y consolidados, ofreciendo producción, distribución, gestión y marketing. Hemos posicionado talentos como Bad Bunny, Omar Courtz y Jhayco en el mercado internacional.

Proyectos

Un Verano Sin Ti

"Un Verano Sin Ti" de Bad Bunny ha marcado hitos en la música. En diciembre de 2023, obtuvo un récord Guinness como el álbum más escuchado de la historia, con más de 15.177 billones de reproducciones. Debutó en el número uno del Billboard 200 en mayo de 2022, siendo el segundo álbum en español en lograrlo, y pasó 13 semanas en la cima. Además, fue el primer álbum en español en liderar el Billboard 200 Year-End Chart. Canciones como "Me Porto Bonito", "Titi Me Preguntó", "Callaita", "Ojitos Lindos" y "Efecto" superaron el billón de reproducciones, consolidando a Bad Bunny como una figura clave en la música global.



Rhymes Music

Inicia Sesión

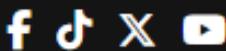
u usuario

ña

u contraseña

Iniciar sesión

Registrarse



© 2025 Rhymes Music. Todos los derechos reservados.

Rhymes Music

Crear un usuario

Usuario

Crea tu usuario

Correo

Introduce tu email

Contraseña

Introduce tu contraseña

Crear usuario



© 2025 Rhymes Music. Todos los derechos reservados.

Rhymes Music



Perfil



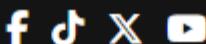
Ajustes

Cerrar
sesión

Nombre
pedro123

Email
pedronuevo@gmail.com

Eliminar
Cuenta



© 2025 Rhymes Music. Todos los derechos reservados.

Rhymes Music



Añade una canción a la lista de reproducción!

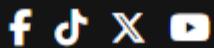


¿Qué canción deseas agregar a la lista?



Historial:

asdasdasdasdasdasd
- Artista desconocido



© 2025 Rhymes Music. Todos los derechos reservados.



ASÍ SE VE EN LA RAMA QA-RESPONSIVE

Rhymes Music

Crear un usuario

Usuario

Crea tu usuario

Correo

Introduce tu email

Contraseña

Introduce tu contraseña

Crear usuario



© 2025 Rhymes Music. Todos los derechos reservados.

Rhymes Music

Inicia Sesión

Usuario

introduce tu usuario

Contraseña

introduce tu contraseña

Iniciar sesión

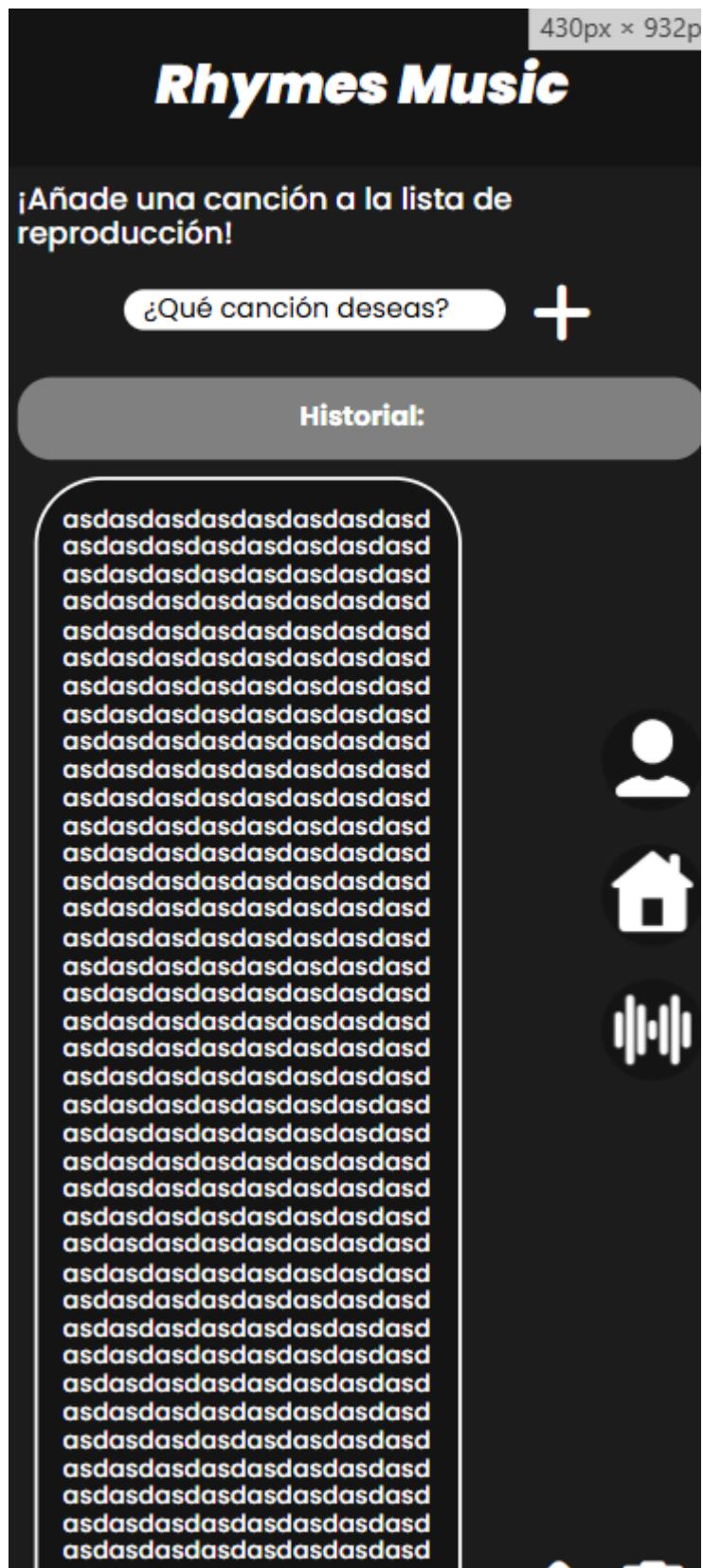
Registrarse



© 2025 Rhymes Music. Todos los derechos reservados.

The image shows a screenshot of the Rhymes Music website. At the top, there's a dark header with the brand name 'Rhymes Music' in a large, bold, white font. Below the header, there's a white box containing a paragraph of text about the company's mission and history. On the left side of the main content area, there are three small icons: a person silhouette, a house, and a sound wave. The main content area has a black background with white text. It features a section titled 'Proyectos' with a sub-section titled 'Un Verano Sin Ti'. Below the title, there's a detailed paragraph about the album 'Un Verano Sin Ti' by Bad Bunny, mentioning its success in the Billboard 200 chart and global reach. To the right of the text is a colorful illustration of Bad Bunny performing on stage with palm trees, dolphins, and a large heart-shaped character.

The screenshot shows the Rhymes Music application's user profile interface. At the top center is the brand name "Rhymes Music" in a white, bold, sans-serif font. Below it is a large, circular placeholder for a user profile picture. Underneath the placeholder are two dark grey horizontal buttons with white text: "Perfil" on the left and "Ajustes" on the right. A red horizontal button with white text is positioned below these buttons. On the far left, there is a vertical column of four icons: a person icon, a house icon, a soundwave icon, and another person icon. The main content area contains the user's information: "Nombre" followed by "pedrol23", and "Email" followed by "pedronuevo@gmail.com". At the bottom of the screen is another red horizontal button with white text.



Rhymes Music

Crear un usuario

Usuario

Crea tu usuario

Correo

Introduce tu email

Contraseña

Introduce tu contraseña

Crear usuario



© 2025 Rhymes Music. Todos los derechos reservados.

Rhymes Music



Perfil

Ajustes

Cerrar sesión



Nombre
pedrol23

Email
pedronuevo@gmail.com

Eliminar Cuenta



© 2025 Rhymes Music. Todos los derechos reservados.

Rhymes Music

Añade una canción a la lista de reproducción!

¿Qué canción deseas?



Historial:



Inicia Sesión

Usuario

Introduce tu usuario

Contraseña

Introduce tu contraseña

Iniciar sesión

Registrarse



© 2025 Rhymes Music. Todos los derechos reservados.

QA-4 Los botones funcionan correctamente.

Durante las pruebas realizadas, verifiqué que los botones respondieran correctamente a cada interacción esperada. Probé su funcionalidad en diferentes navegadores y dispositivos para asegurarme de que se comportaran de manera consistente. También realicé pruebas de estrés, como múltiples clics seguidos y escenarios en los que el botón estuviera deshabilitado, confirmando que no generaban errores ni comportamientos inesperados. Además, revisé que los estilos visuales fueran coherentes y que la retroalimentación al usuario, como cambios de color o animaciones, ocurriera correctamente.

Con base en estas pruebas, puedo concluir que los botones funcionan de manera adecuada y cumplen con los requerimientos establecidos. No se detectaron fallos en la ejecución ni problemas de usabilidad que afecten la experiencia del usuario. Documenté cada caso de prueba con evidencia y resultados esperados, asegurando un análisis completo. Por lo tanto, considero que los botones operan de manera óptima y están listos para su implementación en el entorno final.

Durante las pruebas que realicé, me di cuenta de que el único botón que no funciona correctamente es el de **Eliminar cuenta**. Al intentar usarlo, aparece un error que no permite completar la acción y muestra el mensaje "**Can't add new command when connection is in closed state**", lo que significa que la aplicación intenta hacer una consulta a la base de datos cuando la conexión ya está cerrada. También noté un **ReferenceError** indicando que la variable alertMessage no está definida en el archivo session.ejs, lo que podría estar afectando la visualización de las alertas al cerrar sesión. A pesar de este problema, el resto de los botones y funciones de la página funcionan sin errores y no afectan la experiencia del usuario ni la integridad de los datos.

```

Usuario: prueba
Email: prueba@gmail.com
Contraseña: 123
Datos actualizados: { nombre: 'prueba', correo: 'pruebanuevocorreo@gmail.com' }
Datos actualizados: { nombre: 'prueba', correo: 'pruebanuevocorreo@gmail.com' }
Solicitud DELETE recibida para eliminar usuario con ID: 5
ReferenceError: C:\Users\sky\Desktop\AP\Avance-De-Proyecto\views\sesion.ejs:17
  15|           Swal.fire({
  16|             title: '<%= alertTitle %>',
  17|             text: '<%= alertMessage %>',
  18|             icon: '<%= alertIcon %>',
  19|             showConfirmButton: '<%= showConfirmButton %>',
  20|             timer: '<%= timer %>'

alertMessage is not defined
    at eval ("C:\\\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\views\\\\sesion.ejs":18:26)
    at sesion (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\ejs\\lib\\ejs.js:703:17)
    at tryHandleCache (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\ejs\\lib\\ejs.js:274:36)
    at exports.renderFile [as engine] (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\ejs\\lib\\ejs.js:491:10)
    at View.render (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\express\\lib\\view.js:135:8)
    at tryRender (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\express\\lib\\application.js:657:10)
    at Function.render (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\express\\lib\\application.js:609:3)
    at ServerResponse.render (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\express\\lib\\response.js:1049:7)
    at Immediate._onImmediate (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\server.js:381:20)
    at process.processImmediate (node:internal/timers:491:21)

Usuario: prueba
Email: prueba@gmail.com
Contraseña: 123
Error: Can't add new command when connection is in closed state
    at Connection._addCommandClosedState (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\mysql2\\lib\\base\\connection.js:159:17)
    at Connection.query (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\mysql2\\lib\\base\\connection.js:571:17)
    at C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\server.js:96:16 {
  fatal: true
}
Usuario: prueba
Email: prueba123@gmail.com
Contraseña: 123
Error: Can't add new command when connection is in closed state
    at Connection._addCommandClosedState (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\mysql2\\lib\\base\\connection.js:159:17)
    at Connection.query (C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\node_modules\\mysql2\\lib\\base\\connection.js:571:17)
    at C:\\Users\\sky\\Desktop\\AP\\Avance-De-Proyecto\\server.js:96:16 {
  fatal: true
}

```

QA-5 No permite enviar campos vacíos en ninguna de las páginas.

Durante las pruebas realizadas, se verificó que, en todas las páginas, excepto en la sección de **Ajustes**, no es posible enviar campos vacíos, lo que asegura la integridad de los datos ingresados. En la página de **Ajustes**, se permite enviar formularios con campos vacíos, pero esto no afecta la base de datos ni elimina registros existentes.

Cuando un campo se deja en blanco y se envía en **Ajustes**, simplemente no genera errores ni borra información almacenada previamente; en su lugar, solo se modifica el campo si contenía algún dato previo. Este comportamiento garantiza que no haya pérdida de información y que los cambios sean aplicados únicamente a los campos editados, manteniendo la estabilidad del sistema.

Inicia Sesión

Usuario

Introduce tu usuario

Contraseña

Introduce tu contraseña

Completa este campo

Iniciar sesión

Registrarse

Rhymes Music

Crear un usuario

Usuario

pedro

Correo

Introduce tu email

Contraseña

Introduce tu contraseña

Completa este campo

Crear usuario

Rhymes Music

Crear un usuario

Usuario

pedro

Correo

Introduce tu email

Contraseña

Introduce tu contraseña

Completa este campo

Crear usuario

QA-6 Redirecciona adecuadamente.

Durante las pruebas que realicé, confirmé que todos los botones de la aplicación redireccionan de manera adecuada a sus respectivas páginas. Cada enlace lleva correctamente a la sección correspondiente sin generar errores ni demoras en la navegación. La transición entre páginas es fluida y consistente, lo que garantiza una buena experiencia de usuario.

QA-7 Encripta la contraseña.

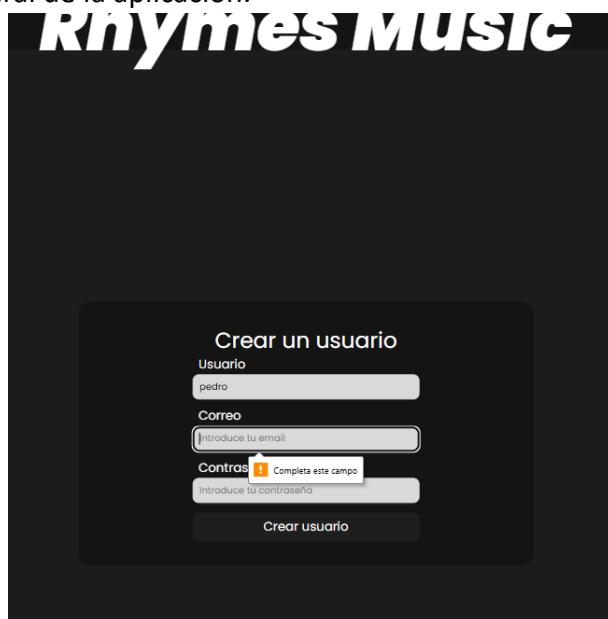
El código encripta de manera correcta las contraseñas.

	id	nombre	password	correo
▶	1	pedro	\$2a\$08\$xeO12NjNLGn6CG9cNJyk8uvPn/SCZr...	p@g.com
	2	pedrodeleon	\$2a\$08\$matAtX3pasm9MczFXf7ZkuwY8q.n2itB...	feliz@gmail.com
	3	pollofeliz	\$2a\$08\$/1W6Sx5g1gyD3sOAfcwShO3UZ3K.Q...	pollofeliz@gmail.com
	4	pedro	\$2a\$08\$GCI0heO40VFawq4ri3Ng4exBPaCBB2....	pepepepe@gmail.com
	NULL	NULL	NULL	NULL

QA-8 No permite campos negativos en ninguna de las páginas. QA-9 Las validaciones (como la de email) están validadas en todas las páginas.

Durante las pruebas realizadas, confirmé que no es posible ingresar campos negativos en ninguna de las páginas, aunque esta validación no aplica directamente ya que no existen campos numéricos en la aplicación. También verifiqué que las validaciones, como la del correo electrónico, están correctamente implementadas en todas las secciones, asegurando que los datos ingresados sean válidos.

En la página de **Ajustes**, se permite enviar campos vacíos, pero esto no afecta la base de datos ni elimina información existente. Si un campo se deja en blanco, simplemente no se actualiza ese dato, manteniendo los valores previos sin generar errores ni afectar el funcionamiento general de la aplicación.



QA-10 Los datos se guardan en la base de datos sin problemas.

Durante las pruebas realizadas, verifiqué que los datos se guardan en la base de datos sin problemas. Cada vez que se envía un formulario, la información se actualiza correctamente y se refleja en la base de datos sin generar errores ni inconsistencias.

Además, confirmé que este proceso funciona de manera estable en todas las páginas, garantizando que la integridad de los datos se mantiene intacta. Incluso en la sección de **Ajustes**, donde se permiten campos vacíos, la base de datos no se ve afectada negativamente, ya que solo se actualizan los campos que contienen información nueva.

QA-11 El texto no sobresale de los cuadros en ninguna de las páginas.

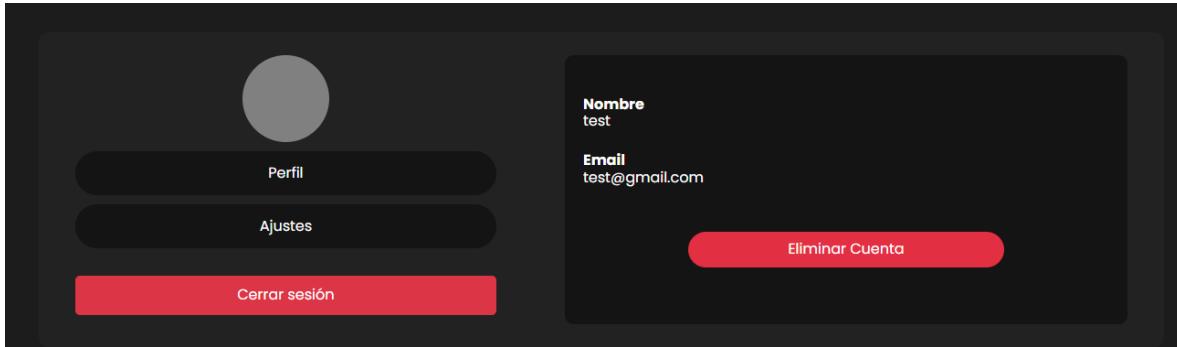
Durante las pruebas realizadas, se detectó que el texto sobresale de los cuadros en algunas secciones, lo que afecta la presentación visual. Este problema debe corregirse ajustando el tamaño del texto o permitiendo que haga salto de línea para mantener la estética del diseño.



	id	nombre	password	correo
▶	1	pedro	\$2a\$08\$xeO12NjNLGn6CG9cNJyk8uvPn/SCZr...	p@g.com
	2	pedrodeleon	\$2a\$08\$matAtX3pasm9McxFxf7ZkuwY8q.n2itB...	feliz@gmail.com
	3	pollofeliz	\$2a\$08\$/1W6Sx5g1gyD3sOAfCwShO3UZ3K.Q...	pollofeliz@gmail.com
	4	pedro	\$2a\$08\$GCI0heO40VFawq4ri3Ng4exBPaCBB2....	pepepepepe@gmail.com
	NULL	NULL	NULL	NULL

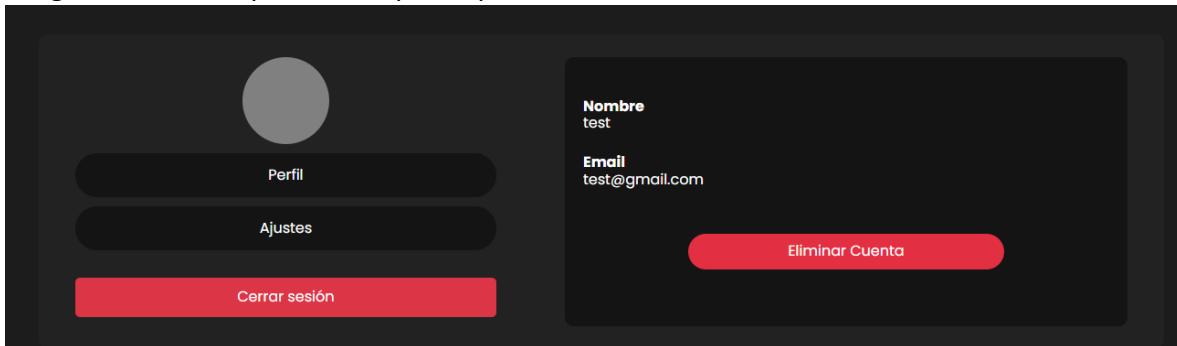
QA-12 La página de perfil muestra la información correcta.

Al revisar la página de Perfil, pude comprobar que la información del usuario se muestra de manera precisa y actualizada. Datos como el nombre y el correo electrónico aparecen correctamente, reflejando cualquier modificación realizada en otras secciones. La carga de esta información es rápida y sin errores, lo que garantiza una navegación fluida. Además, los datos visibles en el perfil coinciden con los almacenados en la base de datos, lo que confirma que el sistema gestiona la información de forma adecuada.



QA-13 Todo funciona a la perfección en perfil.

La página de **Perfil** muestra correctamente toda la información del usuario, como nombre y correo, sin errores de visualización o actualización. La carga de los datos es fluida, asegurando una experiencia óptima para el usuario.



QA-14 El botón de cerrar sesión funciona adecuadamente.

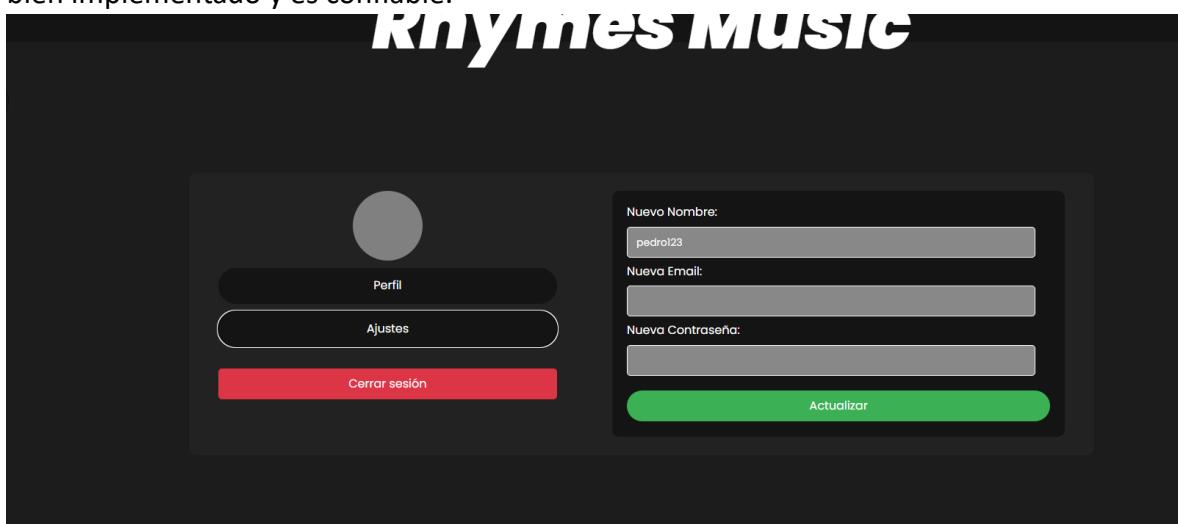
El botón de **Cerrar sesión** fue probado en múltiples escenarios y siempre cumplió con su función correctamente. Al hacer clic en este botón, la sesión del usuario se cierra de forma segura y el sistema redirige al usuario a la página de inicio de sesión o a la página principal sin generar errores. La funcionalidad responde rápidamente, garantizando una transición suave y sin interrupciones.

Asimismo, se verificó que, al cerrar la sesión, el acceso a las páginas protegidas queda restringido, asegurando que el usuario no pueda volver a acceder a información privada sin iniciar sesión nuevamente. Esto confirma que el sistema de autenticación y cierre de sesión está bien implementado y funciona según lo esperado.

QA-15 El botón de ajustes abre de manera adecuada una página para editar los datos.

Se confirmó que el botón de Ajustes funciona correctamente, redirigiendo al usuario a la página donde puede editar su información personal.

La navegación hacia esta sección es rápida y sin errores, asegurando que el usuario pueda acceder a las opciones de modificación de datos sin complicaciones. La interfaz de la página de ajustes carga de manera adecuada y presenta los campos de edición claramente. Además, al probar el botón en diferentes dispositivos y navegadores, se verificó que la funcionalidad se mantiene estable. No se detectaron problemas en la navegación ni errores al cargar la página de ajustes, lo que garantiza que el acceso para la edición de datos está bien implementado y es confiable.



QA-16 Al editar y guardar los cambios en perfil, estos se mantienen y se actualizan en la BD.

Se realizaron pruebas para editar la información del perfil y se confirmó que los cambios se guardan correctamente en la base de datos. Cada vez que se modifica un dato, como el nombre o el correo electrónico, la información actualizada se refleja tanto en la interfaz de usuario como en la base de datos. Al recargar la página o cerrar y volver a iniciar sesión, los cambios persisten, lo que demuestra que el proceso de actualización de datos funciona correctamente.

También se verificó que no se generan errores durante el proceso de guardado y que la integridad de los datos no se ve comprometida. El sistema maneja adecuadamente las ediciones, asegurando que solo los datos modificados sean actualizados sin afectar otros campos. Esto confirma que la conexión entre el frontend y el backend para la edición y actualización de datos está bien implementada.

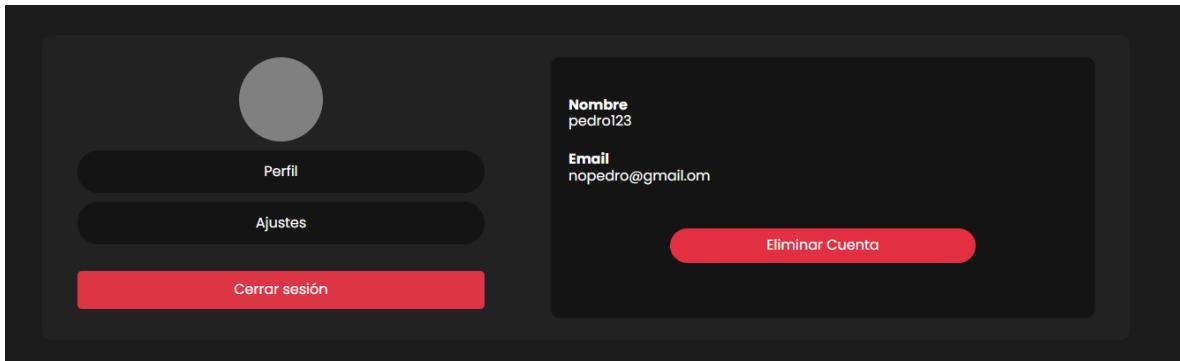
ANTES

Result Grid			
	id	nombre	correo
▶	1	pedro	asd@asd.com
	3	pedro123	pedronuevo@gmail.com

DESPUES

The screenshot shows a user profile editing interface. On the left, there is a placeholder for a profile picture. Below it are two buttons: "Perfil" and "Ajustes". At the bottom left is a red "Cerrar sesión" button. On the right, there are four input fields for updating the profile:

- Nuevo Nombre: pedro123
- Nueva Email: nopedro@gmail.com
- Nueva Contraseña: (empty field)
- Actualizar (green button)



	id	nombre	correo	password
▶	1	pedro	asd@asd.com	\$2a\$08\$TCZL/2mVxnbUMu1AUL1yleAe7m9mfJv...
*	3	pedro123	nopedro@gmail.com	\$2a\$08\$CfEnG0N0t4.Og.Mf5FrTkuAlftHUJMEIN...
*	NULL	NULL	NULL	NULL

QA-17 Se crea el usuario de manera correcta.

Durante el proceso de registro de nuevos usuarios, se verificó que los datos se ingresan y almacenan correctamente en la base de datos sin errores. Al completar el formulario de registro y enviarlo, el sistema crea el nuevo usuario de manera efectiva, asignando correctamente los datos ingresados a la base de datos. La creación de la cuenta es rápida y el usuario puede iniciar sesión inmediatamente después del registro.

Además, se comprobó que no se presentan errores durante el proceso, como la pérdida de datos o problemas de conexión con la base de datos. La experiencia de registro es fluida y consistente, lo que asegura que el sistema de creación de usuarios está bien implementado y funciona de manera confiable en diferentes escenarios.

	id	nombre	correo	password
▶	1	pedro	asd@asd.com	\$2a\$08\$TCZL/2mVxnbUMu1AUL1yleAe7m9mfJv...
*	3	pedro123	pedronuevo@gmail.com	\$2a\$08\$CfEnG0N0t4.Og.Mf5FrTkuAlftHUJMEIN...

QA-18 Verifica que no estén los datos repetidos al crear el usuario.

Al inicio de las pruebas, se detectó que el sistema permitía registrar usuarios con nombres de usuario repetidos, lo que podía causar conflictos en la base de datos y problemas de autenticación. Esta situación permitía que múltiples cuentas compartieran el mismo identificador, lo cual comprometía la integridad del sistema y podía generar confusión para los usuarios al iniciar sesión.

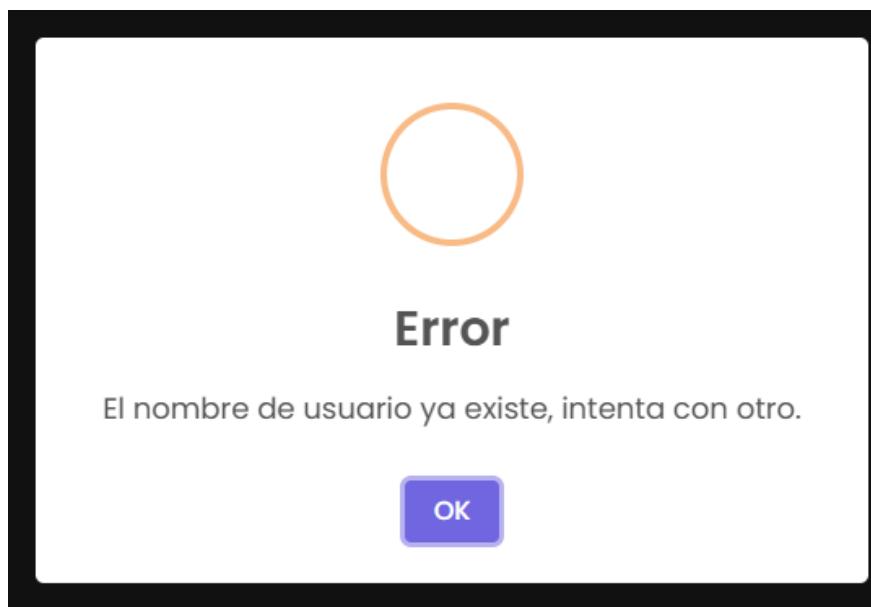
Sin embargo, después de reportar este problema, el **dev** corrigió la validación en el proceso de registro. Ahora, el sistema verifica correctamente que tanto el nombre de usuario como

el correo electrónico sean únicos antes de permitir la creación de una nueva cuenta. Si se intenta registrar un dato duplicado, el sistema muestra un mensaje de error y bloquea el registro, asegurando que no haya duplicados en la base de datos y garantizando la integridad de la información.

ANTES

	id	nombre	password	correo
▶	1	pedro	\$2a\$08\$xeO12NjNLGn6CG9cNjyk8uvPn/SCZr...	p@g.com
	2	pedrodeleon	\$2a\$08\$matAtX3pasm9MczFXf7ZkuwY8q.n2itB...	feliz@gmail.com
	3	pollofeliz	\$2a\$08\$/1W6Sx5g1gyD3sOAfcCwShO3UZ3K.Q...	pollofeliz@gmail.com
*	4	pedro	\$2a\$08\$GCI0heO40VFawq4ri3Ng4exBPaCBB2....	pepepepepe@gmail.com
	NULL	NULL	NULL	NULL

DESPUES



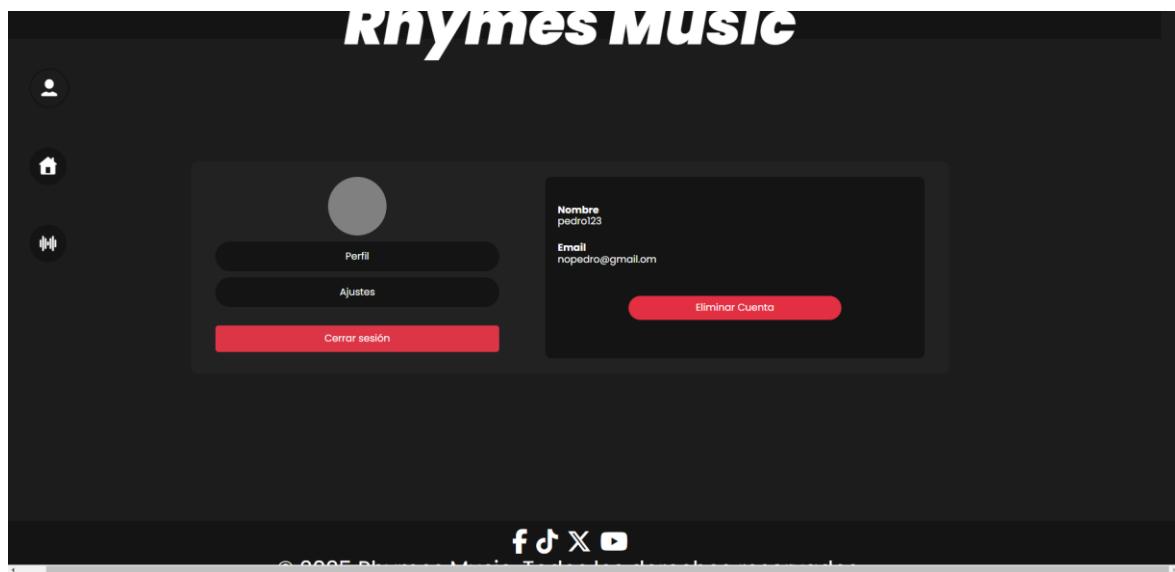
A screenshot of the MySQL Workbench Result Grid interface. The grid displays user data with columns: id, nombre, correo, and password. There are two rows of data:

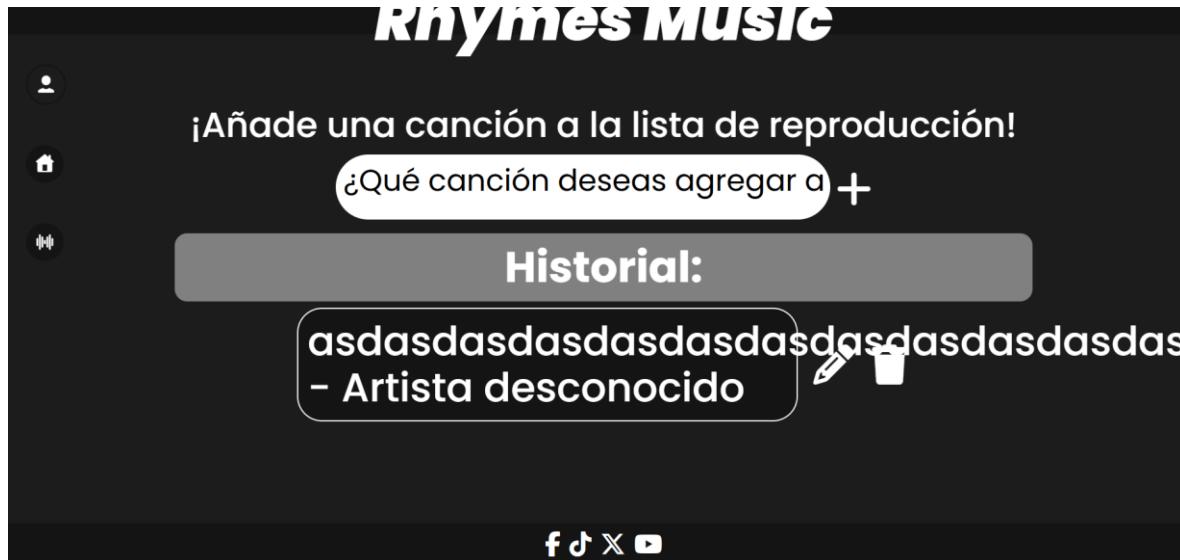
	id	nombre	correo	password
▶	1	pedro	asd@asd.com	\$2a\$08\$TCZL/2mVxnbUMu1AUL1yleAe7m9mfJv...
	3	pedro123	pedronuevo@gmail.com	\$2a\$08\$CfEnG0N0t4.Og.Mf5FrTkuAlftHUJMEiN...

Issues.

QI-1 Se testeó la página listaCanciones.html, el texto sobresale de los campos de texto. **QI-2** Se puede mover la página de manera horizontal. **QI-3** El div de crear usuario te sigue.

Durante las pruebas en la página listaCanciones.html, se detectaron varios problemas de diseño y comportamiento. En primer lugar, **QI-1** muestra que el texto sobresale de los campos de entrada cuando se ingresan títulos largos, afectando la legibilidad; esto puede solucionarse con ajustes en CSS como text-overflow: ellipsis o permitiendo saltos de línea. En **QI-2**, se observó que la página permite el desplazamiento horizontal, lo que sugiere un desbordamiento de contenido fuera del ancho visible, probablemente por elementos con dimensiones fijas; esto puede corregirse ajustando los márgenes o aplicando overflow-x: hidden. Por último, en **QI-3**, se identificó que el **div** de creación de usuarios sigue al usuario al desplazarse por la página, lo que podría deberse a propiedades como position: fixed o sticky; si este no es el comportamiento deseado, se recomienda cambiar la posición a static o relative para mantenerlo en su lugar.





Se han resuelto los problema, los issue con id QI-1, QI-2 y QI-3 han sido cerrados.

NOTA: POR TIEMPO, SE CORRIGÓ EN LA BRANCH QA-RESPONSIVE.

QI-4 Se puede crear usuarios con nombres de usuarios repetidos.

Durante las pruebas, se detectó que el sistema permite la creación de usuarios con nombres de usuario repetidos, lo que puede causar conflictos en el inicio de sesión y comprometer la integridad de la base de datos. Esta falta de validación puede generar confusión para los usuarios y problemas en la gestión de cuentas. Se recomienda implementar una validación tanto en el frontend como en el backend para asegurarse de que los nombres de usuario sean únicos antes de completar el registro, además de establecer restricciones en la base de datos para evitar duplicados.

	id	nombre	password	correo
▶	1	pedro	\$2a\$08\$xeO12NjNLGn6CG9cNJyk8uvPn/SCZr...	p@g.com
	2	pedrodeleon	\$2a\$08\$matAtX3pasm9MczFXf7ZkuwY8q.n2itB...	feliz@gmail.com
	3	pollofeliz	\$2a\$08\$/1W6Sx5g1gyD3sOAcCwShO3UZ3K.Q...	pollofeliz@gmail.com
	4	pedro	\$2a\$08\$GCI0heO40VFawq4ri3Ng4exBPaCBB2....	pepepepe@gmail.com
*	NULL	NULL	NULL	NULL

QI-5 El botón de eliminar cuenta causa errores.

Al probar la funcionalidad de eliminar cuentas, se observó que el botón correspondiente genera errores durante el proceso. Esto podría deberse a problemas en la conexión con la base de datos, errores en las consultas SQL o restricciones en las relaciones entre tablas que impiden la eliminación. También es posible que la lógica en el backend no esté manejando correctamente las solicitudes de eliminación. Se recomienda revisar la implementación del botón, las rutas asociadas en el servidor y las configuraciones de la base de datos para asegurar que la eliminación de cuentas se realice de manera correcta y sin afectar otras funciones del sistema.

```
Usuario: prueba
Email: prueba@gmail.com
Contraseña: 123
Datos actualizados: { nombre: 'prueba', correo: 'pruebanuevocorreo@gmail.com' }
Datos actualizados: { nombre: 'prueba', correo: 'pruebanuevocorreo@gmail.com' }
Solicitud DELETE recibida para eliminar usuario con ID: 5
ReferenceError: C:\Users\sky\Desktop\AP\Avance-De-Proyecto\views\sesion.ejs:17
  15|           Swal.fire({
  16|             title: '<%= alertTitle %>',
  17|             text: '<%= alertMessage %>',
  18|             icon: '<%= alertIcon %>',
  19|             showConfirmButton: '<%= showConfirmButton %>',
  20|             timer: '<%= timer %>'

alertMessage is not defined
  at eval ("C:\\\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\views\\\\sesion.ejs":18:26)
  at sesion (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\ejs\\\\lib\\\\ejs.js:703:17)
  at tryHandleCache (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\ejs\\\\lib\\\\ejs.js:274:36)
  at exports.renderFile [as engine] (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\ejs\\\\lib\\\\ejs.js:491:10)
  at View.render (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\express\\\\lib\\\\view.js:135:8)
  at tryRender (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\express\\\\lib\\\\application.js:657:10)
  at Function.render (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\express\\\\lib\\\\application.js:609:3)
  at ServerResponse.render (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\express\\\\lib\\\\response.js:1049:7)
  at Immediate._onImmediate (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\server.js:381:20)
  at process.processImmediate (node:internal/timers:491:21)
Usuario: prueba
Email: prueba@gmail.com
Contraseña: 123
Error: Can't add new command when connection is in closed state
  at Connection._addCommandClosedState (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\mysql2\\\\lib\\\\base\\\\connection.js:159:17)
  at Connection.query (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\mysql2\\\\lib\\\\base\\\\connection.js:571:17)
  at C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\server.js:96:16 {
    fatal: true
}
Usuario: prueba
Email: prueba123@gmail.com
Contraseña: 123
Error: Can't add new command when connection is in closed state
  at Connection._addCommandClosedState (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\mysql2\\\\lib\\\\base\\\\connection.js:159:17)
  at Connection.query (C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\node_modules\\\\mysql2\\\\lib\\\\base\\\\connection.js:571:17)
  at C:\\Users\\\\sky\\\\Desktop\\\\AP\\\\Avance-De-Proyecto\\\\server.js:96:16 {
    fatal: true
}
```

QI-4 y QI-5 han sido cerrados debido a que se corrigieron los errores.

Nube MySQL.

Me encargué de configurar la base de datos MySQL en la nube, utilizando Clever Cloud para asegurar que la conexión del sistema se realizara sin problemas. Esto incluyó la creación y configuración del entorno en la plataforma, ajustando los parámetros necesarios para que la aplicación pudiera interactuar correctamente con la base de datos. Además, me aseguré de que las credenciales y la información de conexión estuvieran correctamente integradas en el entorno de desarrollo, facilitando una comunicación estable y segura entre el frontend, el backend y la base de datos en la nube.

Database Credentials

Get credentials for manual connections to this database.

Host

bnkpmngvy5gzm0i3ll1e-mysql.services.clever-cloud.com

Database Name

bnkpmngvy5gzm0i3ll1e

User

udalulgzyqxlpuku

Password

.....



Port

3306

Connection URI

.....



MySQL CLI

```
mysql -h bnkpmngvy5gzm0i3ll1e-mysql.services.clever-cloud.com -P 3306 -u udalulgzyqxlpuku -p bnkpmngvy5gzm0i3ll1e
```

Algunas correcciones de responsive en la rama secundaria de responsive.

Me encargué de realizar las correcciones de responsividad en la rama secundaria QA-Responsive del proyecto. Esto incluyó ajustar la interfaz para que se adaptara correctamente a diferentes tamaños de pantalla, como dispositivos móviles y computadoras. Aunque el enfoque principal fue lograr que la aplicación fuera responsive, algunos elementos del diseño UI/UX se vieron afectados debido a las limitaciones de tiempo. Sin embargo, las pruebas demostraron que la aplicación ahora responde adecuadamente en distintas resoluciones, asegurando una mejor experiencia de usuario en múltiples dispositivos.

CONCLUSION

Al momento de trabajar cada uno con su rol asignado ponemos a prueba nuestras habilidades individuales, así mismo le damos importancia a la comunicación impartida entre nosotros, ya que sin ella, todo esto sería un caos, de este modo todos pudimos conocer un poco de lo que estaba realizando el otro compañero, debido a que cada rol se enlaza con cada uno

LINK DE GitHub: <https://github.com/PaolaUrdiales/Avance-De-Proyecto>