Documentación Proyecto Final

RegenMelody
https://github.com/eckzen/RegenMelody.git
Erick Montoya

Descripción del Proyecto:

Objetivos:

Web con distintos sonidos ambientales como la lluvia, río, olas, etc. que pueden ser activados individualmente o en conjunto.

Opción de registrarse como usuario para acceder a contenido protegido, como crear, guardar y recuperar mixes.

Público objetivo:

Es un sitio web orientado a cualquier tipo de público que desee relajarse u obtener una sencilla ayuda para conciliar el sueño, desestresarse, etc.

Planificación:

Cronograma del Proyecto				
Fase	Tarea	Días	Inicio	Fecha de fin
Planificación	Definir alcance y requisitos	1	04/06/24	04/06/24
Análisis	Analizar requisitos y diseñar arquitectura	1	05/06/24	05/06/24
Diseño	Crear diagramas UML, mockups, wireframes	1	06/06/24	06/06/24
Implementación	Configurar entorno de desarrollo	0,5	07/06/24(mañana)	07/06/24
Implementación	Desarrollar frontend (HTML,CSS, JS, Bootstrap)	1,5	07/06/24(tarde)	08/06/24
Implementación	Desarrollar backend Java (Spring Boot, Hibernate, Security) y PHP	2	09/06/24	11/06/24
Implementación	Diseñar y crear base de datos (MySQL)	0,5	12/06/2024 (mañana)	12/06/24
Implementación	Integrar frontend y backend	1,5	12/06/2024 (tarde)	12/06/24
Pruebas	Pruebas unitarias, integración y sistema	1	13/06/24	14/06/24
Despliegue	Implementar la aplicación en entorno local	0,5	15/06/2024 (mañana)	15/06/24
Mantenimiento	Redactar plan de mantenimiento futuro	0,5	15/06/2024 (tarde)	15/06/24
Buffer	Tiempo adicional para imprevistos	1	16/06/24	16/06/24
Revisión	Revisión final y ajustes	1	17/06/24	17/06/24

Requisitos funcionales y no funcionales:

Requisitos Funcionales:

1. Reproductor de Sonidos Ambientales:
   - Permitir activar sonidos como lluvia, bosque, olas, etc.
   - Permitir activar múltiples sonidos simultáneamente.
2. Guardar y Compartir Mixes:
   - Guardar configuraciones personalizadas de mezclas de sonidos.
   - Compartir mixes guardados en redes sociales.
3. Registro y Gestión de Usuarios:
   - Registro y autenticación de usuarios.
   - Guardar listas de mixes preferidos por usuario.
   - Enviar notificaciones de novedades por correo electrónico.
4. Vídeo Aleatorio:
   - Mostrar un vídeo aleatorio basado en la preferencia del usuario (mar, bosque, etc.).

Requisitos No Funcionales:

1. Interfaz de Usuario:
    - Debe ser responsiva y atractiva.
    - Usar Bootstrap para una experiencia de usuario consistente.
2. Seguridad:
    - Asegurar la comunicación entre frontend y backend con HTTPS.
    - Implementar autenticación y autorización de usuarios.
3. Rendimiento:
    - La aplicación debe ser rápida y eficiente.
4. Escalabilidad:
    - La aplicación debe ser escalable para soportar un número creciente de usuarios.
    - (Servidor externo)
5. Mantenibilidad:
    - Código bien documentado y estructurado.
    - Fácil de mantener y actualizar.

Estimación de tiempo y recursos:

1. Planificación (04/06/2024):
    - Definir alcance y requisitos: Se definen los objetivos del proyecto y los entregables.
2. Análisis (05/06/2024):
    - Analizar requisitos y diseñar arquitectura: Se analiza los requisitos y se diseña la arquitectura del sistema.
3. Diseño (06/06/2024):
    - Crear diagramas UML, mockups, wireframes: Se crean los diagramas necesarios para la arquitectura y la interfaz de usuario.
4. Implementación (07/06/2024 – 12/06/2024):
    - Configurar entorno de desarrollo.
    - Desarrollar frontend: Se desarrolla la interfaz de usuario usando HTML, CSS, JavaScript y Bootstrap mediante Visual Studio Code y recursos multimedia.
    - Desarrollar backend: Se desarrolla la lógica del servidor usando Java, Spring Boot y Hibernate mediante Netbeans.
    - Diseñar y crear base de datos: Se diseña la base de datos y crea las tablas necesarias mediante MySQL Workbench.
    - Integrar frontend y backend: Se trabaja en la integración del frontend y backend utilizando, entre otros, Javascript.
5. Pruebas (13/06/2024- 14-06-2024):
    - Pruebas unitarias, de integración y sistema: Se realizan las pruebas necesarias para asegurar

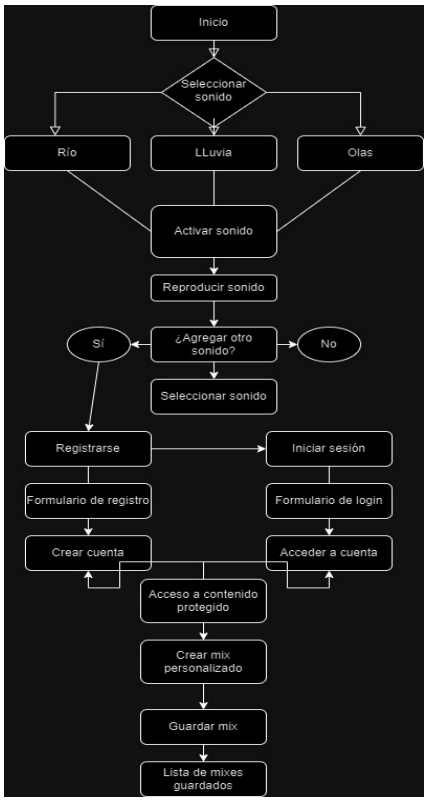
que todo funcione correctamente, mediante tests de las características ejecutables en el lado del cliente y las pruebas necesarias del servidor mediante Postman.

6. Despliegue (15/06/2024):
    - Implementar la aplicación en entorno local: Se implementa la aplicación en un entorno local mediante HTTPS.
    - Redactar plan de mantenimiento futuro: Se redacta un plan para el mantenimiento futuro de la aplicación.

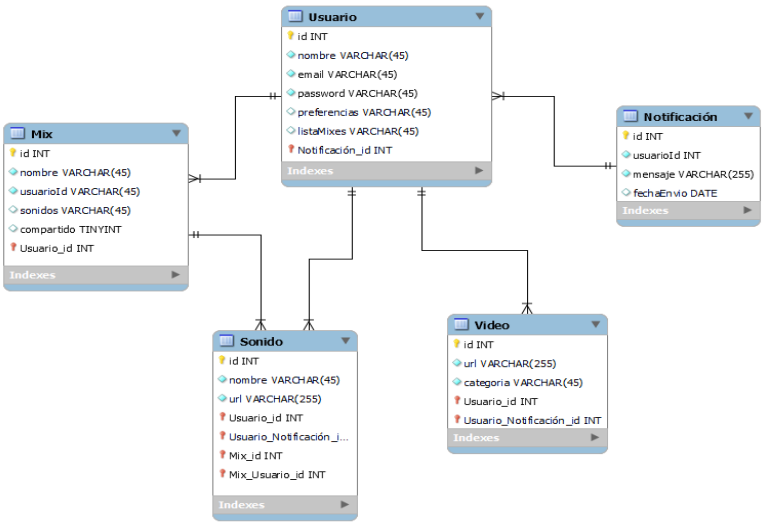
7. Buffer y Revisión (13/06/2024 - 15/06/2024):
    - Buffer para imprevistos: Se reserva tiempo adicional para manejar cualquier imprevisto.
    - Revisión final y ajustes: Se revisa el proyecto y realiza los ajustes necesarios antes de la entrega final.

Análisis y Diseño:

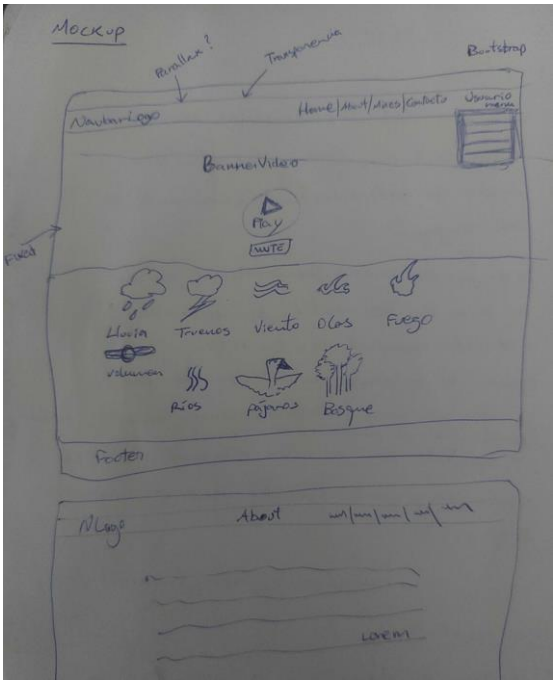
Diagrama de flujo:



UML:



Mockup:



Arquitectura del sistema:

Resumen:

Frontend

- **Tecnologías**: HTML, JavaScript, Bootstrap.
- **Funciones**:
  - Interfaz de usuario para seleccionar y reproducir sonidos.
  - Formulario de registro e inicio de sesión.
  - Interfaz para crear y guardar mixes personalizados.

Backend

Java Spring Boot (localhost:8080):

- **Funciones**:
  - API REST para manejo de autenticación (login y registro).
  - Gestión de usuarios utilizando Spring Security y Hibernate.

PHP (localhost:80):

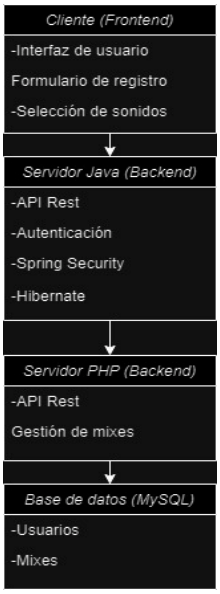
- **Funciones**:
  - API REST para manejar mixes de sonidos.
  - Almacenamiento y recuperación de mixes desde la base de datos compartida.

Base de Datos

MySQL:

- **Base de Datos**: `regenmelody`
- **Tablas**: `users` y `mixes`

Diseño de la arquitectura del sistema:



Diseño de la base de datos:

```sql
CREATE DATABASE regenmelody;
USE regenmelody;

CREATE TABLE users (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL
);

CREATE TABLE mixes (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    user_id BIGINT NOT NULL,
    name VARCHAR(100) NOT NULL,
    sounds JSON,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Implementación:

Descripción Detallada

1. **Frontend**:
    - **HTML**: Estructura de la página.
    - **JavaScript**: Lógica de interacción y llamadas AJAX (en PHP).
    - **Bootstrap**: Estilo y diseño responsivo.

2. **Backend Java Spring Boot (localhost:8080)**:
    - **Spring Boot**: Framework para construir el backend.
    - **Spring Security**: Manejo de autenticación y autorización.
    - **Hibernate**: ORM para interactuar con la base de datos.

3. **Backend PHP (localhost:80)**:
    - **PHP**: Manejo de la lógica de negocio para los mixes.
    - **APIs**: Endpoints para crear y leer los mixes.

4. **Base de Datos (MySQL)**:
    - **Usuarios**: Almacena la información de los usuarios registrados.
    - **Mixes**: Almacena los mixes creados por los usuarios, junto con los sonidos en formato JSON.

Ejemplo de Interacciones

1. **Registro de Usuario**:
    - El frontend envía una solicitud POST al servidor Java para registrar un nuevo usuario.
    - El servidor Java utiliza Spring Security y Hibernate para guardar el nuevo usuario en la tabla `users`.

2. **Inicio de Sesión**:
    - El frontend envía una solicitud POST al servidor Java para autenticar al usuario.
    - El servidor Java verifica las credenciales y genera un token de sesión.

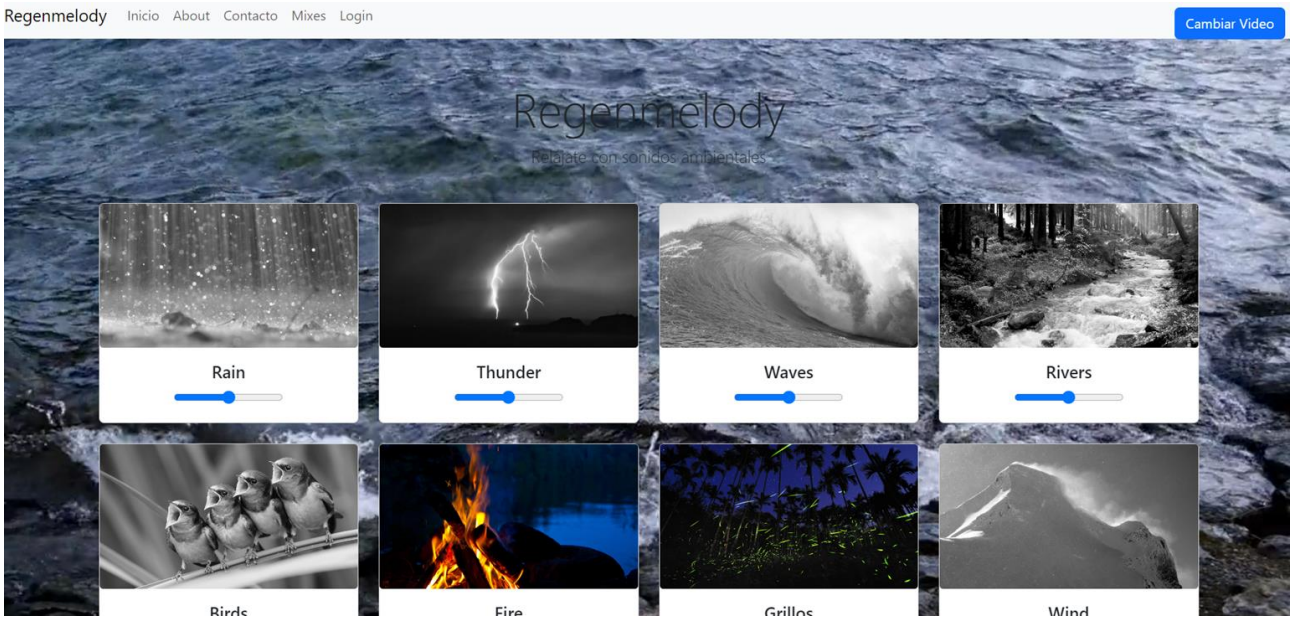
3. **Gestión de Mixes**:
    - El frontend envía una solicitud POST al servidor PHP para crear un nuevo mix.
    - El servidor PHP guarda el mix en la tabla `mixes` de la base de datos compartida.

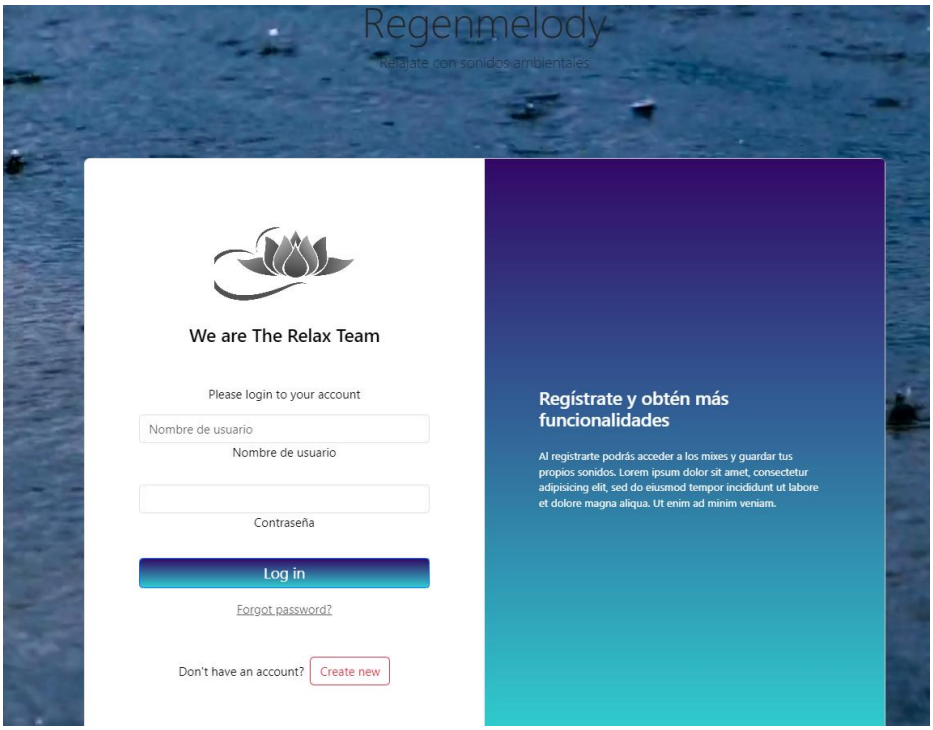
Este diseño proporciona una separación clara de responsabilidades entre el frontend y los dos backends, permitiendo que cada parte del sistema se maneje de manera eficiente y escalable.

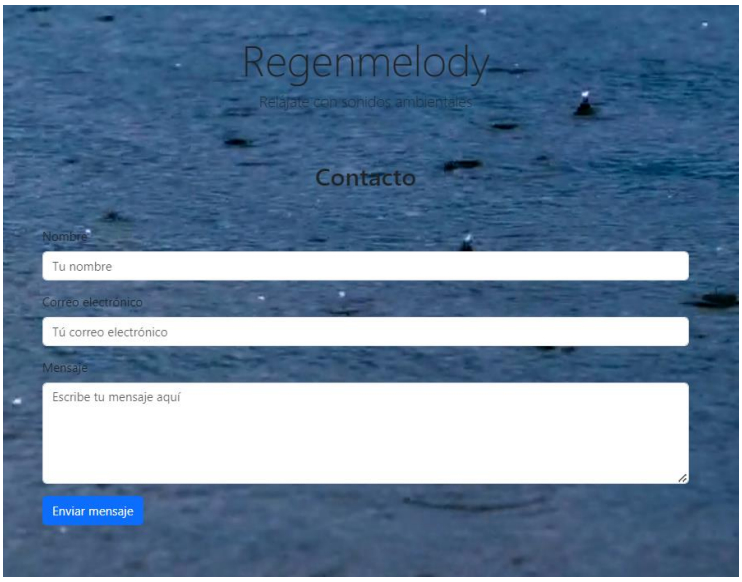
Explicación de su código y funcionamiento*:

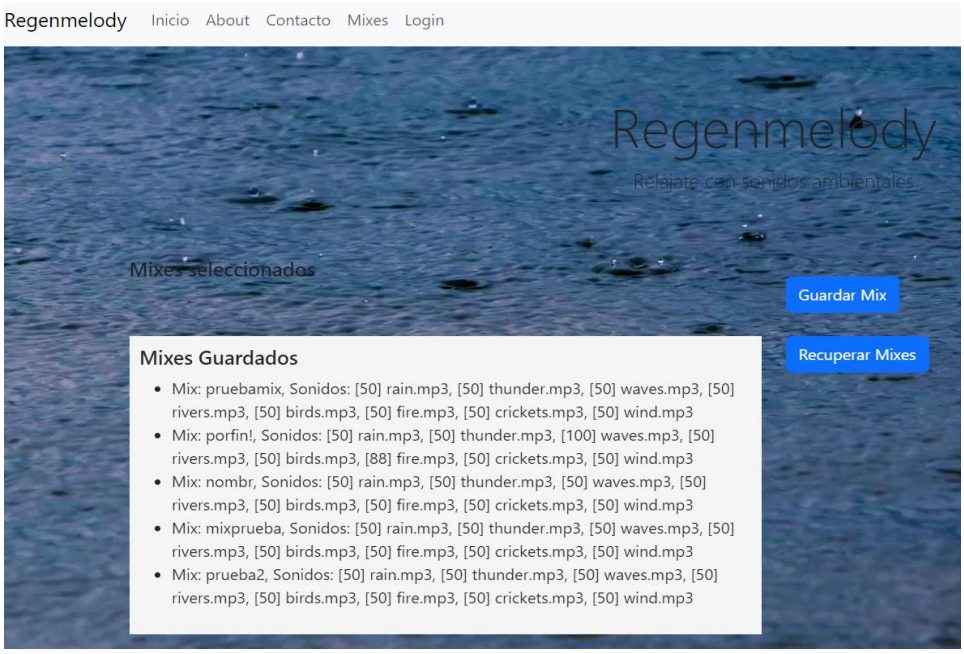
*Ver comentarios en el código.

Capturas de pantalla de la interfaz de usuario:






Regenmelody
Relájate con sonidos ambientales
Contacto
Nombre
Tu nombre
Correo electrónico
Tú correo electrónico
Mensaje
Escribe tu mensaje aquí
Enviar mensaje


Regenmelody
Inicio
About
Contacto
Mixes
Login
Regenmelody
Relájate con sonidos ambientales
Mixes seleccionados
Guardar Mix
Recuperar Mixes
Mixes Guardados
Mix: pruebamix, Sonidos: [50] rain.mp3, [50] thunder.mp3, [50] waves.mp3, [50] rivers.mp3, [50] birds.mp3, [50] fire.mp3, [50] crickets.mp3, [50] wind.mp3
Mix: porfin!, Sonidos: [50] rain.mp3, [50] thunder.mp3, [100] waves.mp3, [50] rivers.mp3, [50] birds.mp3, [88] fire.mp3, [50] crickets.mp3, [50] wind.mp3
Mix: nombr, Sonidos: [50] rain.mp3, [50] thunder.mp3, [50] waves.mp3, [50] rivers.mp3, [50] birds.mp3, [50] fire.mp3, [50] crickets.mp3, [50] wind.mp3
Mix: mixprueba, Sonidos: [50] rain.mp3, [50] thunder.mp3, [50] waves.mp3, [50] rivers.mp3, [50] birds.mp3, [50] fire.mp3, [50] crickets.mp3, [50] wind.mp3
Mix: prueba2, Sonidos: [50] rain.mp3, [50] thunder.mp3, [50] waves.mp3, [50] rivers.mp3, [50] birds.mp3, [50] fire.mp3, [50] crickets.mp3, [50] wind.mp3

Pruebas:

Plan de pruebas:

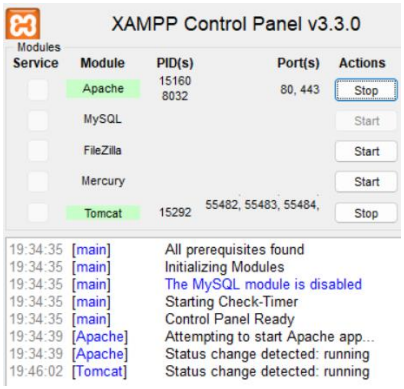
Pruebas realizadas mediante Postman con solicitudes GET y POST, y a través de interacciones (login, register, guardar mix, recuperar mix).

Resultados de las pruebas unitarias, de integración y de sistema:

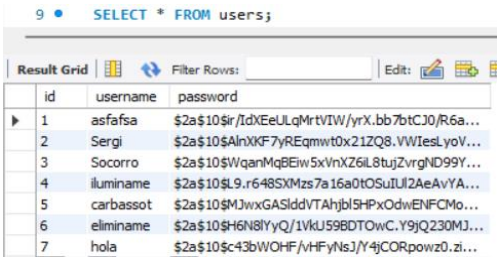
Inicialización de servidor local con Java Spring Boot Security Hibernate:

```
2024-06-16 19:46:01.537  INFO 15292 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with [org.springframework.security.web
2024-06-16 19:46:02.017  INFO 15292 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/index.html]
2024-06-16 19:46:02.542  INFO 15292 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2024-06-16 19:46:02.579  INFO 15292 --- [           main] com.mycompany.regenmelody.regenmelody    : Started regenmelody in 8.089 seconds (JVM running for 8.809)
```

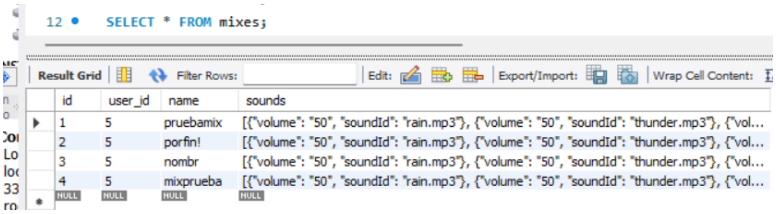
Inicialización de servidor local con PHP (Xampp):



Prueba de login y registro exitoso con backend Java y PHP (Java puerto 8080, PHP puerto 80):



Inserción de mixes en la BBDD mediante método POST en Backend PHP:



Despliegue:

Instrucciones para el Despliegue de la Aplicación:

Prerrequisitos

- **Java JDK** (versión 11 o superior)
- **NetBeans IDE** (preferiblemente la última versión)
- **Apache Tomcat** (puede ser incluido en NetBeans)
- **MySQL** (se puede utilizar MySQL Workbench)
- **XAMPP** (para el servidor PHP)

Configuración del Servidor MySQL

1. **Instalación y Configuración de MySQL**:
    - Instala MySQL y asegúrate de que está corriendo en el puerto 3306.
    - Crea la base de datos `regenmelody` y las tablas `users` y `mixes`

Configuración y Despliegue del Backend Java (Spring Boot)

1. **Configuración del Proyecto en NetBeans**:
    - Crea un nuevo proyecto Spring Boot en NetBeans.
    - Agrega las siguientes dependencias en tu `pom.xml`:
    - Configura la conexión a la base de datos y otros parámetros en `src/main/resources/application.properties`

Creación de Entidades y Repositorios:

- Crea las entidades `User` y `Mix`, los repositorios correspondientes y los servicios en tu proyecto.

2. **Implementación de Seguridad**:
    - Configura Spring Security para manejar la autenticación y autorización.

3. **Ejecuta el Proyecto**:
    - Ejecuta tu aplicación Spring Boot desde NetBeans.

Configuración y Despliegue del Backend PHP (XAMPP)

1. **Instalación y Configuración de XAMPP**:
    - Descarga e instala XAMPP. (no es necesario instalar MySQL en XAMPP puesto que ya se utiliza Workbench)
    - Asegúrate de que Apache y MySQL están corriendo.

2. **Configuración del Proyecto PHP**:
    - Crea un nuevo directorio en el directorio `htdocs` de XAMPP, por ejemplo, `regenmelody-php`.
    - Crea archivos PHP para manejar la lógica de los mixes (login, registro, CRUD de mixes).

3. **Conexión a la Base de Datos**:
    - Configura la conexión a la base de datos en tus scripts PHP, como en `db.php`

Implementación de Lógica de Negocios:

- Crea scripts PHP para manejar el login, registro y gestión de mixes, asegurándote de que pueden interactuar con la base de datos MySQL.

Configuración y Despliegue del Frontend

1. **Estructura del Proyecto**:
    - Crea los archivos HTML, CSS y JavaScript en el directorio de la página dentro de htdocs.

2. **Interacción con el Backend**:
    - Utiliza AJAX o Fetch API en JavaScript para interactuar con los endpoints del backend (tanto el de Spring Boot como el de PHP).

3. **Despliegue**:
    - Asegúrate de que tu frontend está correctamente configurado para hacer solicitudes al backend Java en `localhost:8080` y al backend PHP en `localhost:80`.

Pruebas

1. **Verifica la Conexión a la Base de Datos**:

- Asegúrate de que ambos backends pueden conectarse y operar con la base de datos MySQL.

2. **Prueba de Funcionalidad**:
    - Prueba el registro y login desde el frontend.
    - Prueba la creación, guardado y visualización de mixes.

3. **Depuración y Ajustes**:
    - Realiza cualquier depuración necesaria usando las herramientas de desarrollo de tu navegador y los logs de tu servidor backend.

Con estas instrucciones y configuraciones, deberías poder desplegar la aplicación en un entorno local utilizando Java Spring Boot y PHP con XAMPP.

Mantenimiento:

Plan de Mantenimiento

1. Mantenimiento Rutinario

- **Backup Regular**:
    - Programar copias de seguridad diarias/semanales de la base de datos y el contenido.
    - Almacenar los backups en una ubicación segura (ej. AWS S3, Google Cloud Storage).

- **Actualización de Dependencias**:
    - Revisar y actualizar periódicamente las dependencias del backend (Spring Boot, Hibernate) y del frontend (bibliotecas JavaScript, Bootstrap).
    - Asegurar que los servidores PHP y MySQL están actualizados a las últimas versiones estables.

- **Monitoreo y Logging**:
    - Implementar herramientas de monitoreo como Prometheus, Grafana o ELK Stack para supervisar el rendimiento del servidor y las aplicaciones.
    - Configurar alertas para detectar problemas de rendimiento o errores.

- **Seguridad**:
    - Realizar auditorías de seguridad periódicas para identificar y solucionar vulnerabilidades.
    - Mantener certificados SSL actualizados y configurar políticas de seguridad HTTP.
    - Actualizar la seguridad instalando HTTPS.

- **Optimización de la Base de Datos**:
    - Revisar y optimizar consultas SQL regularmente.
    - Realizar mantenimiento de índices y tablas para asegurar un rendimiento óptimo.

2. Mejoras y Actualizaciones

- **Expansión del Repositorio de Sonidos y Videos**:

    - Ampliar el catálogo de sonidos ambientales y añadir videos relacionados.
    - Crear catálogos de distintas temásticas, como motivación para hacer ejercicio.
    - Utilizar servicios de almacenamiento en la nube para gestionar grandes volúmenes de contenido multimedia.

- **Mejoras en la Experiencia de Usuario (UX)**:
    - Implementar nuevas características en la interfaz de usuario para mejorar la navegación y la interacción.
    - Realizar pruebas de usabilidad y recoger feedback de los usuarios para realizar mejoras continuas.

- **Nuevas Funcionalidades**:
    - **Creación de Playlists**:
        - Permitir a los usuarios crear y compartir playlists de sonidos y videos.
        - Ejemplo: Un usuario puede crear una playlist llamada "Relajación" con una combinación de sonidos de lluvia y olas, junto con videos de paisajes tranquilos y compartirla en redes sociales.
    - **Comentarios y Valoraciones**:
        - Añadir la funcionalidad de comentarios y valoraciones para los sonidos y videos.
        - Ejemplo: Los usuarios pueden comentar y calificar cada sonido o video, ayudando a otros a encontrar el contenido más popular y calidad.

- **Desarrollo de la Aplicación Móvil**:
    - **Tecnologías**: Utilizar frameworks como React Native o Flutter para desarrollar una aplicación móvil multiplataforma.
    - **Sincronización**: Asegurar que los usuarios puedan sincronizar su contenido y preferencias entre la web y la app móvil.
    - **Notificaciones Push**: Implementar notificaciones push para alertar a los usuarios sobre nuevos contenidos, actualizaciones, y recordatorios de uso.

Plan de Actualización de Funcionalidades

1. Integración de Videos

Base de Datos:

Añadir una nueva tabla `videos` en la base de datos.

Backend:

Ampliar las API del backend para manejar operaciones CRUD para videos.

Frontend:

Crear una interfaz para que los usuarios puedan explorar y reproducir videos.
Utilizar un reproductor de video embebido, como Video.js, para la reproducción de videos en el navegador.

Aplicación Móvil

- **Desarrollo**:
    - Configurar el entorno de desarrollo para React Native o Flutter.
    - Crear interfaces de usuario similares a la versión web para mantener la coherencia en la experiencia de usuario.
    - Implementar funciones offline para que los usuarios puedan descargar y reproducir contenido sin conexión a internet.
- **Lanzamiento**:
    - Publicar la aplicación en Google Play Store y Apple App Store.
    - Realizar campañas de marketing para promover la aplicación móvil entre los usuarios actuales y potenciales.

Este plan de mantenimiento y mejoras ayudará a garantizar la estabilidad, seguridad y crecimiento continuo de la plataforma, proporcionando una mejor experiencia para los usuarios y permitiendo la expansión de las funcionalidades y contenido a lo largo del tiempo.

Lecciones aprendidas:

Instalación General

1. **Importancia de la Configuración Inicial**:
    - **Configuración de Herramientas y Entornos**:
        - Aprender a configurar herramientas de desarrollo como NetBeans y XAMPP.
        - Comprender la importancia de configurar correctamente la base de datos MySQL y asegurarse de que los servidores Apache y MySQL estén corriendo correctamente en XAMPP.
    - **Organización de Archivos y Proyectos**:
        - La importancia de mantener una estructura de archivos organizada, especialmente cuando se trabaja con múltiples tecnologías y lenguajes.
        - Cómo estructurar el directorio de proyecto para facilitar el desarrollo y mantenimiento.
2. **Integración de Tecnologías Diferentes**:
    - **Conexión entre Frontend y Backend**:
        - Aprender a hacer solicitudes AJAX desde el frontend (HTML, JavaScript) hacia los endpoints del backend (PHP).
        - La importancia de las APIs RESTful para la comunicación entre el frontend y el backend.
    - **Uso de JSON para la Comunicación de Datos**:
        - Cómo estructurar y manejar datos en formato JSON para intercambiar información entre el frontend y el backend.
3. **Seguridad y Autenticación**:
    - **Implementación de Seguridad**:

- La necesidad de implementar mecanismos de seguridad robustos, como SSL/TLS para la comunicación segura.
- La importancia de sanitizar entradas de usuario y protegerse contra vulnerabilidades comunes (SQL injection, XSS).

Instalación y Configuración en Java

1. **Introducción a Java y Spring Boot**:
    - **Configuración del Entorno**:
        - Instalar y configurar el JDK (Java Development Kit).
        - Configurar NetBeans para el desarrollo de aplicaciones Spring Boot.
    - **Estructura de Proyectos en Spring Boot**:
        - Comprender la estructura de un proyecto Spring Boot y cómo organizar paquetes y clases.
        - Importancia de archivos como `application.properties` para la configuración de la aplicación.
2. **Uso de Dependencias y Maven**:
    - **Gestión de Dependencias**:
        - Uso de Maven para gestionar dependencias y plugins, y cómo agregar nuevas dependencias en el archivo `pom.xml`.
    - **Integración con Hibernate**:
        - Configurar Hibernate para ORM (Object-Relational Mapping) y entender cómo mapear entidades Java a tablas en la base de datos.
3. **Spring Security**:
    - **Autenticación y Autorización**:
        - Configuración básica de Spring Security para manejar la autenticación de usuarios.
        - Entender cómo proteger endpoints y rutas utilizando roles y permisos.
4. **Desarrollo de APIs RESTful**:
    - **Creación de Controladores y Servicios**:
        - Aprender a crear controladores REST y servicios en Spring Boot para manejar la lógica de negocio y las operaciones CRUD.
        - Ejemplo: Cómo crear un endpoint para registrar usuarios y otro para manejar el login.
    - **Pruebas y Depuración**:
        - Uso de herramientas de depuración en NetBeans para resolver errores y problemas en el código.
        - Importancia de las pruebas unitarias y de integración para asegurar la

funcionalidad correcta de la aplicación.

-

El proceso de instalación y configuración de la aplicación no solo involucra aspectos técnicos, sino también una comprensión profunda de cómo integrar diversas tecnologías para trabajar juntas de manera eficiente. Aprender Java y Spring Boot, desde la configuración del entorno hasta la implementación de seguridad y APIs RESTful, proporciona una base sólida para futuras expansiones y mejoras de la aplicación. La experiencia adquirida en este proceso ha sido invaluable para abordar proyectos más complejos en el futuro.