

Roberto André Rodríguez González

2023342

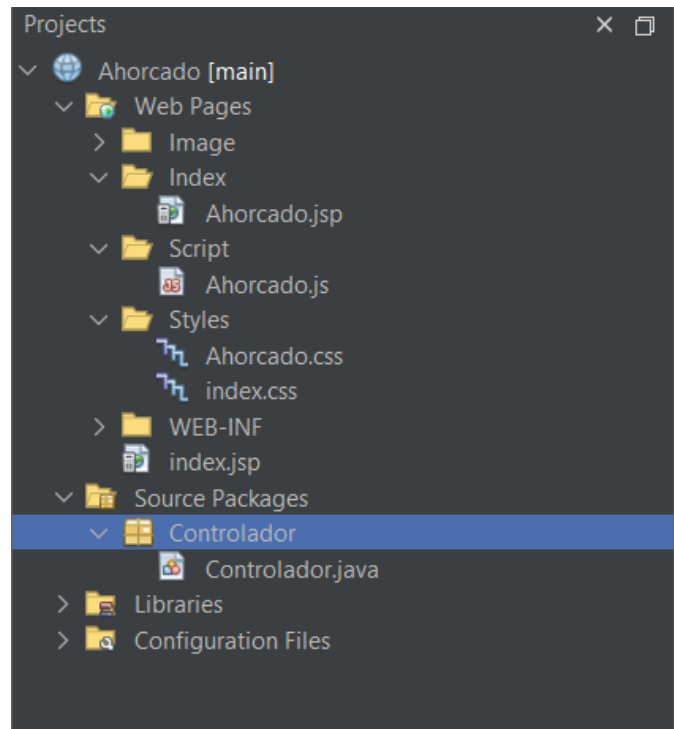
IN5BM

03/09/2025

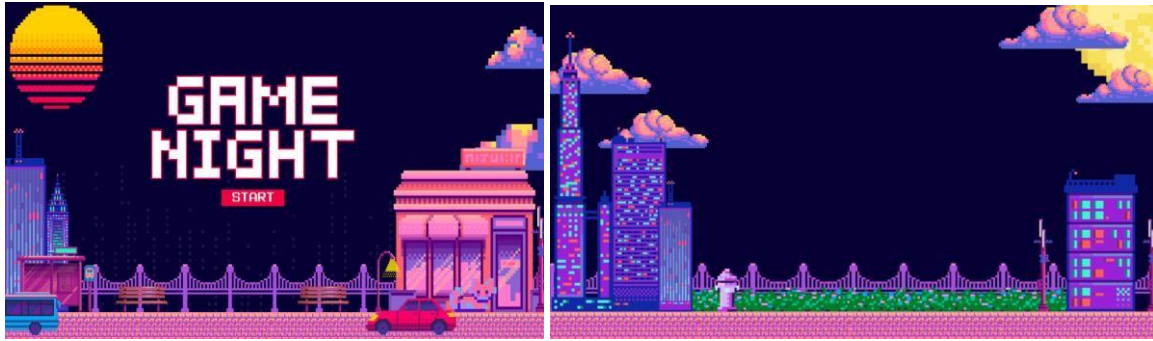
Reporte del Proyecto:

## Ahorcado

Primero se creo el repositorio en donde se iba a alojar el proyecto del Ahorcado, este fue el primer commit, luego cree un nuevo proyecto de Java Web y cree las clases y todos los archivos necesarios para el proyecto:



Una vez creados todos los archivos hice el segundo commit, en donde agregué dichos archivos. Una vez agregados al repositorio empecé por lo básico, hacer los bocetos para las vistas de las páginas web, utilice esas imágenes/fondos de referencia:



También hice el boceto de como quería estructurar el juego del ahorcado y quedo tal que así:

AHORCADO

05:00

— — — — —

ABCDEFGHIJKLM  
NNOPQRSTUVWXYZ  
X Y Z

INICIO

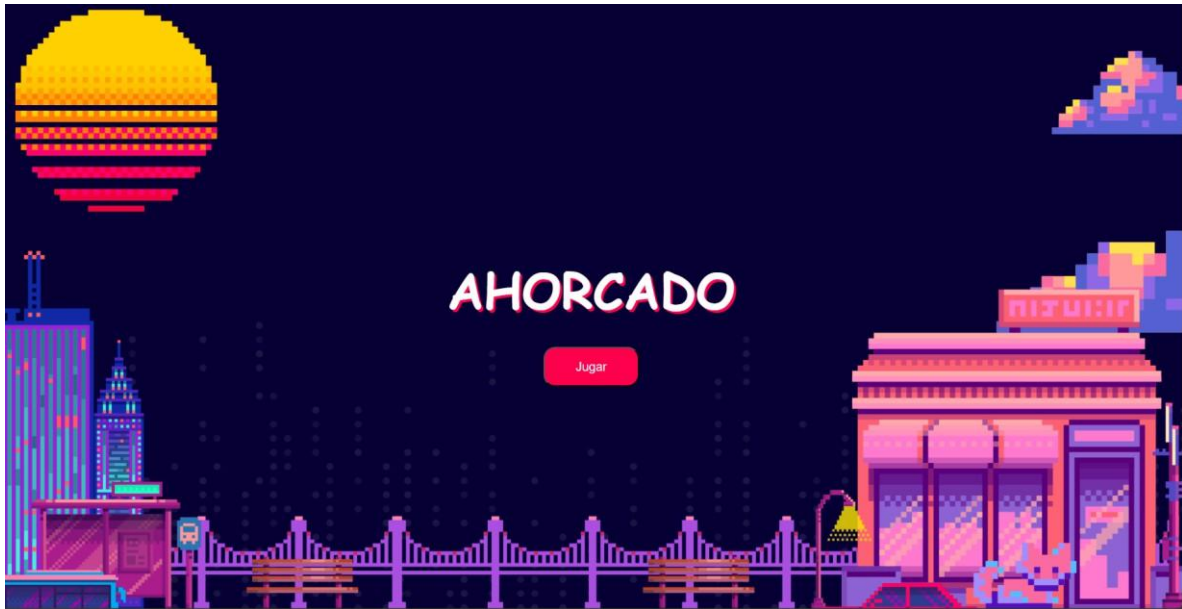
PAUSA

REINICIAR

SALIR

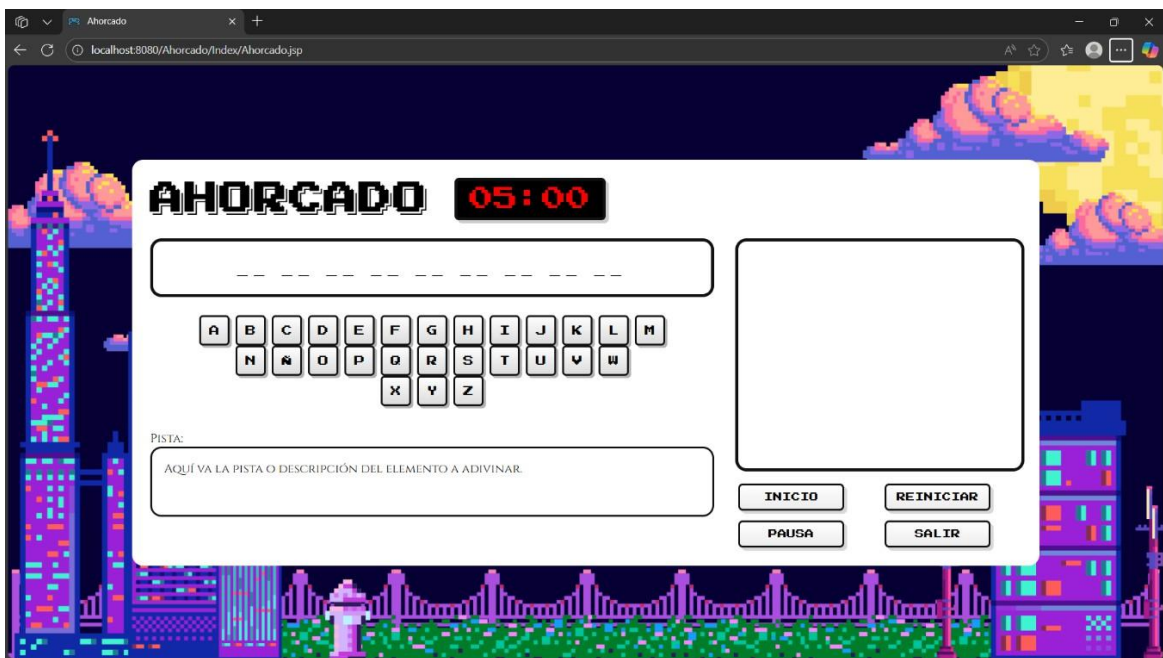
Pista:

Una vez hecho esto empecé a trabajar la vista del index tomando de guía lo anterior y termino tal que así:



Luego de terminar el estilo de del index conecté por medio de un link las dos páginas, una vez terminado hice el tercer commit.

Una vez terminado el index, empeze a trabajar en la estructura y diseño de la vista del ahorcado, para esta uno de los fondos, que los saque de canva, y poco a poco le fue metiendo diselo hasta que quedo así:



Luego de terminar de darle los pequeños detalles, procedi a hacer el cuarto commit, en donde subi el diseño y estilo de la vista del ahorcado.

Luego le pregunté al profesor si estaba bien conectar ambas vistas de la pagina solo por el link a lo que me dijo que implementara un servlet llamado Controlador en donde controlara las direcciones y así conectar ambas vistas y eso hice:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String menu = request.getParameter("menu");

    if (menu.equals("Ahorcado")) {
        request.getRequestDispatcher("Index/Ahorcado.jsp").forward(request, response);
    } else if (menu.equals("Index")) {
        request.getRequestDispatcher("index.jsp").forward(request, response);
    }
}
```

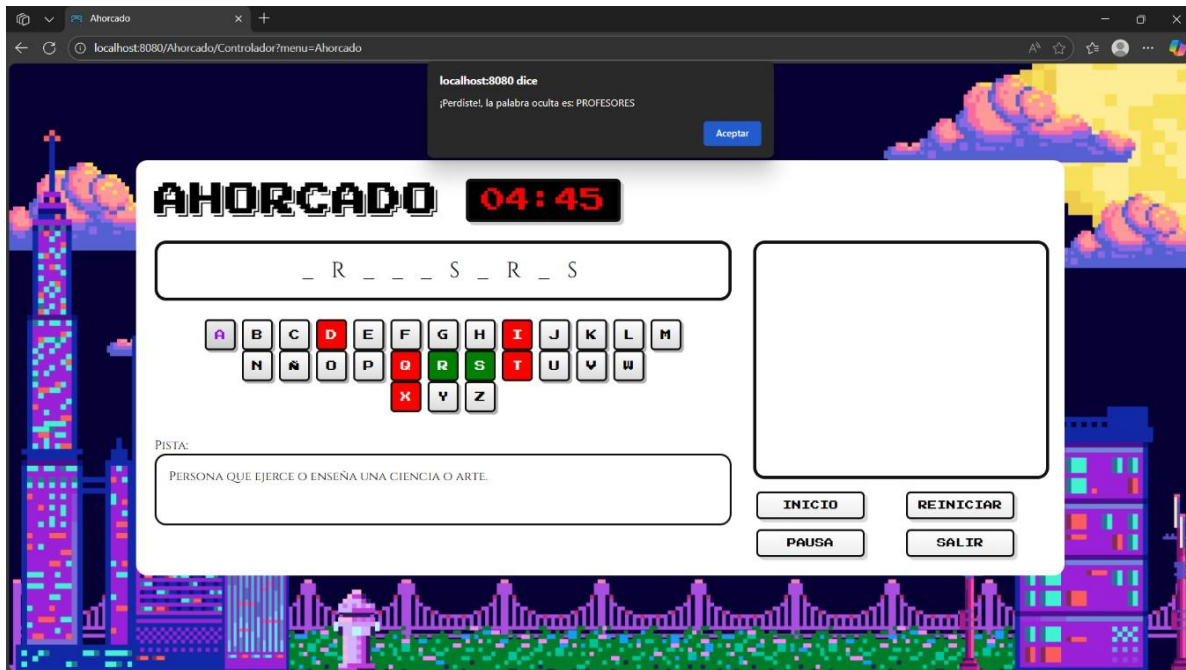
Una ves terminado y luego de haber comprobado que todo funcionar procedi a subir los cambios al repositorio, siendo así este el quinto commit.

Luego, ya con las vistas conectas y ambas con diseño, ya era hora de trabajar el backendy como primer paso hice las funciones de los botones inferiores, con los cuales se iba a manejar el juego. Esto lo único que hacia era:

- INICIAR: Iniciar la cuenta regresiva
- REINICIAR: Volvía a empezar desde cero todo.
- PAUSA: Paraba el tiempo.
- SALIR: Regresaba a la vista del index

Entre otra de las funcionalidades que se hizo en este tiempo también están, poder usar los botones de cada una de las letras, en el caso de estar la letra en la palabra setearla, verificar si esta la letra en dicha palabra, etc.

Adjunto imagen de referencia, para comprobar el funcionamiento:



Una vez hecho esto, se hizo el quinto commit, en donde se subió todos los cambios al repositorio.

Casi terminando hice un sistema de vida visual en la parte superior izquierda en donde por medio de imágenes de corazones/vidas, por cada vez que fallaba una letra se eliminaba una vida y al terminarse todas las vidas se acabara el juego:



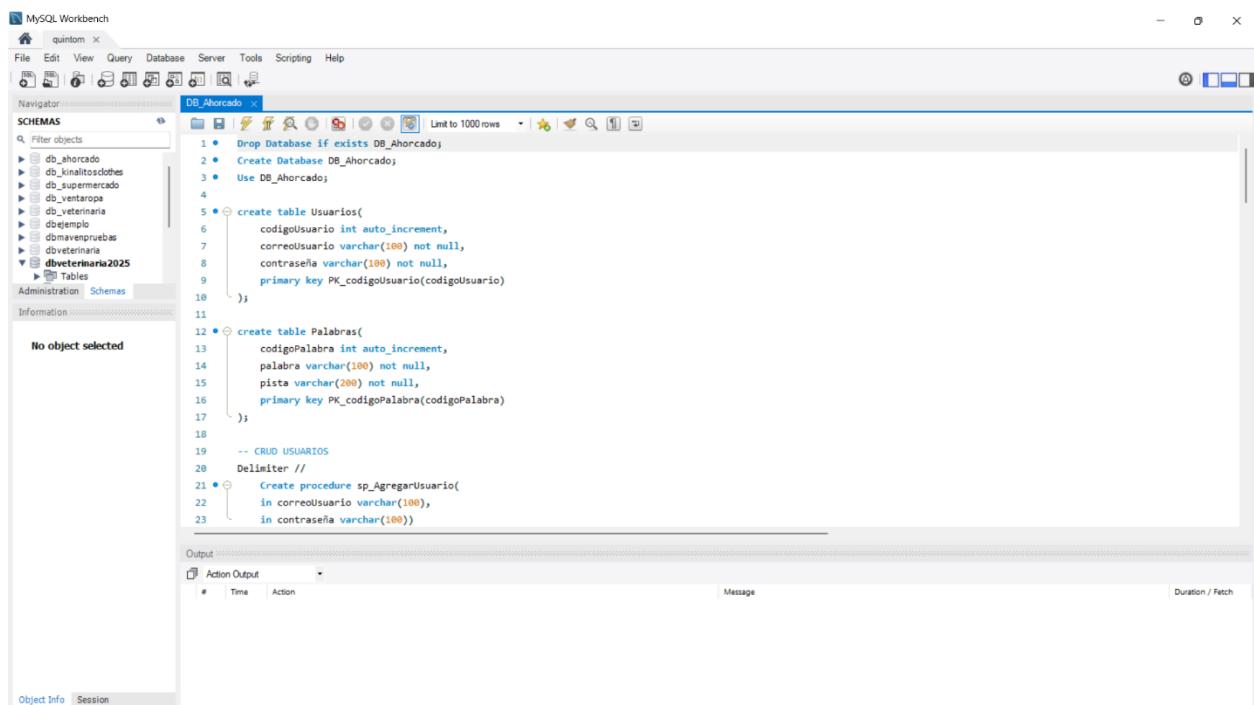
Una vez terminado hice sexto commit, en donde subí los cambios junto con el sistema de vidas.

Luego de estar probando el juego me percate que al terminarse al pausar el juego y volver a darle iniciar había un error, el cual cuando pausabas el juego y le volvías a dar iniciar en ves de continuar con la partida, te iniciaba una nueva, es decir cambiaba de palabra.

Una vez corregido este error, procedí hacer el séptimo commit, en donde subí los cambios aplicados para solucionar este error.

Por último, pensé en que en el cuadro vacío, cuando ganara se mostrara una imagen referente a la palabra, pero por falta de tiempo y por más que intente nunca logre que se mostrara la imagen por lo que aun se encuentra en proceso.

El profesor nos dijo que teníamos que implementar al proyecto una base de datos para implementar un login/registro y almacenar las palabras en la base de datos en vez de utilizar un array. Lo primero que hice definir las entidades que iba a utilizar, las cuales fueron solo dos, Usuarios y Palabras, una vez hecho eso crear el CRUD (Create, Read, Update y Delete) de cada una de estas entidades. Y con esto había terminado con la base de datos.



Luego había que conectarla con el proyecto, y para ello use el patron de diseño Singleton para asegurar una solo instancia. Y una vez echo esto ya podía empezar a trabajar con la conexión.

```
package Config;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexion {

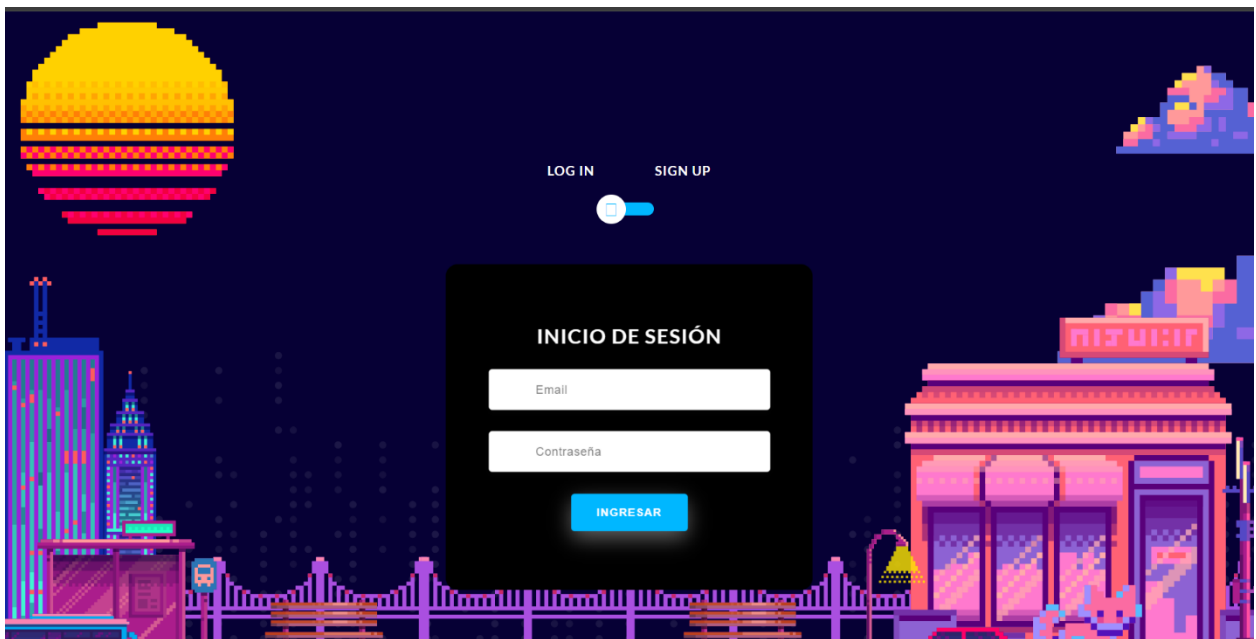
    Connection conexion;

    public Connection Conexion() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conexion = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/DB_Ahorcado?useSSL=false",
                "quintom",
                "admin"
            );
        } catch (ClassNotFoundException | SQLException e) {
            System.out.println("Error al conectar a la base de datos: " + e.getMessage());
            e.printStackTrace();
        }
        return conexion;
    }
}
```

Luego hice siguientes clases:

- **Usuarios:** Esta es el modelo de la entidad Usuarios en la base de datos.
- **Palabras:** Esta es el modelo de la entidad Palabras en la base de datos.
- **UsuariosDAO:** En esta están los métodos que use para esta entidad, como los métodos para el login y registrar.
- **PalabrasDAO:** En esta están los métodos que use para esta entidad, como el método que me sirve para mostrar las palabras.

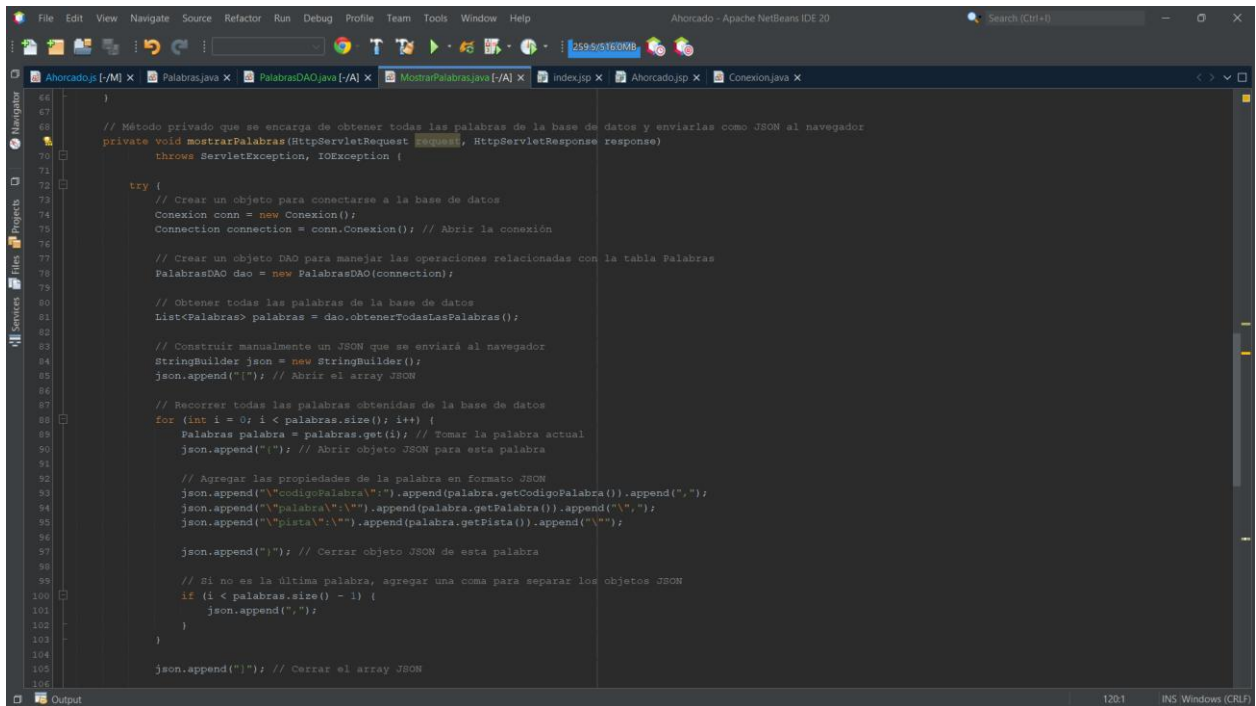
Luego me dedique a hacer la vista del Login y junto con un servlet llamado Validar implemente el inicio de sesión y el registro de nuevos usuarios. Después con todo esto echo subi los cambios al repositorio.



Lo último que tenía que hacer era que por medio de las palabras almacenadas en la base de datos pudiera mostrarlas en el juego. Para eso usando el método para obtener las palabras de la base de datos en PalabrasDAO y por medio de un servlet llamado MostraraPalabras, este consulta los datos obtenidos del PalabrasDAO y los devuelve en formato JSON y por último reemplaza el arreglo estático por una petición al servlet, cargando dinámicamente las palabras y sus pistas.

El resultado de esto es que ahora obtiene las palabras y sus pistas desde la base de datos, permite agregar, editar o eliminar palabras sin modificar el código del cliente. Gracias a estos cambios ahora también puede administrar usuarios junto con el inicio de sesión y el registrar del Login.





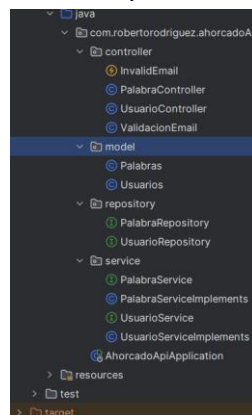
```
64 }
65
66 // Método privado que se encarga de obtener todas las palabras de la base de datos y enviarlas como JSON al navegador
67 private void mostrarPalabras(HttpServletRequest request, HttpServletResponse response)
68     throws ServletException, IOException {
69
70     try {
71         // Crear un objeto para conectarse a la base de datos
72         Conexion conn = new Conexion();
73         Connection connection = conn.Conexion(); // Abrir la conexión
74
75         // Crear un objeto DAO para manejar las operaciones relacionadas con la tabla Palabras
76         PalabrasDAO dao = new PalabrasDAO(connection);
77
78         // Obtener todas las palabras de la base de datos
79         List<Palabras> palabras = dao.obtenerTodasLasPalabras();
80
81         // Construir manualmente un JSON que se enviará al navegador
82         StringBuilder json = new StringBuilder();
83         json.append("["); // Abrir el array JSON
84
85         // Recorrer todas las palabras obtenidas de la base de datos
86         for (int i = 0; i < palabras.size(); i++) {
87             Palabras palabra = palabras.get(i); // Tomar la palabra actual
88             json.append("{"); // Abrir objeto JSON para esta palabra
89
90             // Agregar las propiedades de la palabra en formato JSON
91             json.append("\"codigoPalabra\":").append(palabra.getCodigoPalabra()).append(",");
92             json.append("\"palabra\":").append(palabra.getPalabra()).append(",");
93             json.append("\"pista\":").append(palabra.getPista()).append(",");
94
95             json.append("}"); // Cerrar objeto JSON de esta palabra
96
97             // Si no es la última palabra, agregar una coma para separar los objetos JSON
98             if (i < palabras.size() - 1) {
99                 json.append(",");
100             }
101         }
102
103         json.append("]"); // Cerrar el array JSON
104
105     }
```

Y ya con esto echo subi todos los cambios al repositorio.

Luego se solicito crear un API para manejar el CRUD de cada entidad del proyecto por medio de Spring Boot desde Postman.

Para ello desde Spring Initialz cree un proyecto con las dependencias necesarias para la API, una vez descargado lo abri en IntelliJ y empeze a trabajar.

Lo primero que hice fue crear la arquitectura que iba usar:



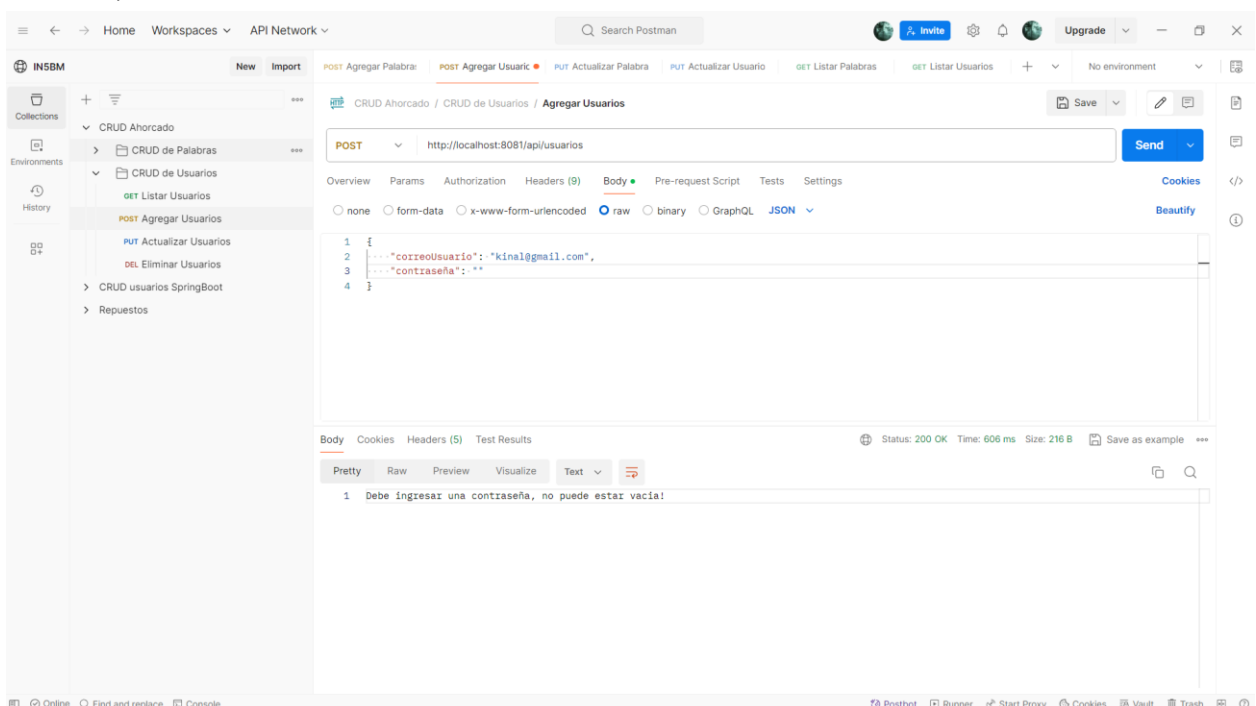
En cada una de las capetar se creo los archivos necesarios para el funcionamiento de la API.



Una vez terminada la creación de la API hice la colección para ejecutar las solicitudes de la API en Postman, cree una carpeta por cada entidad y agregue en cada una de ellas los request para probar mi API.



Luego se hizo el manejo de errores/excepciones, para evitar que reviente el código. En estos errores se encuentran: Evitar que los correos, contraseñas, palabras o pistas se repitan, también se maneja evitar que los parámetros se dejen vacíos al momento de enviar la solicitud, etc.



Para este proyecto utilice las siguientes tecnologías:

- NetBeans como el IDE de desarrollo
- GIT y GitHub como mi sistema de versiones.
- Trello el cual utilice para llevar orden de lo que debía hacer, además en el se encuentran cada uno de los commits realizados y los que aun están pendientes.
- MySQL para la creación y la administración de la base de datos.

Los lenguajes de programación que use para este proyecto son los siguientes:

- Java, utilizado tanto para el serverlet Controlador como para las vistas JSP.
- JavaScript para el backend y la lógica de programación.
- Conceptos de CSS y HTML para el diseño y estilo de las paginas del proyecto.
- SQL para crear y administrar la base de datos

Si desea verificar el código del proyecto se encuentra en el repositorio de GitHub:

<https://github.com/rrodriguez-2023342/Ahorcado.git>

Si desea ver la organización de como se fue trabajando el proyecto lo encuentra en Trello:

<https://trello.com/invite/b/68b6f9c612ad08b7c16273ea/ATTIdd217de70dcd2df215b607228d69af51261214A8/ahorcado>