PROYECTO DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

TIME PLANNER PLUS

SERGIO GARCÍA HERRERA

CURSO 2023/24

ÍNDICE

1. RESUMEN DEL PROYECTO

Mi proyecto va a tratar sobre una aplicación mobile para la gestión de tareas diarias, haciendo así que el usuario pueda aumentar su productividad y mejorar su organización diaria. La aplicación esta desarrollada con Ionic y Spring Boot, y es una aplicación híbrida y multiplataforma. Cuenta con su propia base de datos relacional (MySQL).

Esta estructurada con un menú en la pantalla principal de dos opciones estando sin loguear. Una vez logueado se muestra un menú inferior de tres opciones y un pequeño menú superior donde puede visualizar las tareas completadas, no completadas y todas las tareas. Además de incluir diversas ventanas dinámicas con las que el usuario estando logueado o no, les mostrará unas u otras.

Las funcionalidades que ofrece mi aplicación son las siguientes:

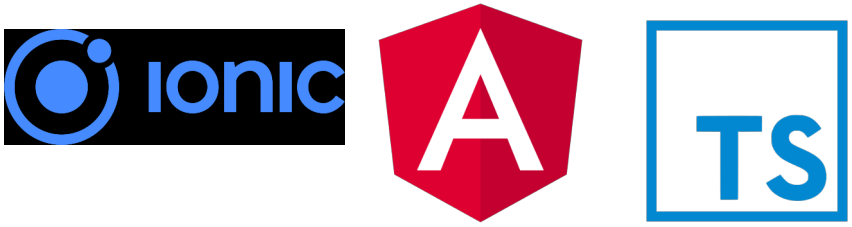
- ➔ **Consulta de todo el listado de tareas creadas en la base de datos en tiempo real.** Donde se pueden consultar las completadas, aún por completar y todas las tareas independientemente de su estado.
- ➔ **Creación de tareas**, donde por defecto la tarea se creará con estado "No completada".
- ➔ **Edición en el listado de tareas**, haciendo clic en cada elemento del listado de tareas, donde se podrá modificar el titulo, la descripción y el estado de la tarea. Actualizándose la lista a tiempo real para poder visualizar la edición de las tareas al instante.
- ➔ **Eliminación de tareas** en la misma ventana de la edición de tareas, actualizándose a tiempo real la lista y poder visualizarla sin recargar.
- ➔ **Posibilidad de marcar y desmarcar una tarea como completada o no completada** mediante un boton en cada elemento de la lista, y que este pase en tiempo real a la lista de completadas o no completadas.
- ➔ **Creación de cuentas mediante nombre de usuario y contraseña y inicio de sesión,** donde a cada usuario logueado se le generará un token único y diferente.
- ➔ **Edición de perfil**, donde se podrá cambiar el nombre y apellidos y la contraseña. (Esta funcionalidad me quedó sin implementar con éxito).

2. INTRODUCCIÓN

La gestión eficiente del tiempo y las tareas es un desafío constante tanto en el ámbito educativo como en el profesional. Con el objetivo de abordar esta necesidad, **he desarrollado un sistema de gestión de tareas** denominado "Time Planner Plus", diseñado para ayudar a los usuarios a organizar mejor sus actividades diarias, priorizar tareas y, en última instancia, mejorar su productividad y gestión del tiempo.

Este proyecto se centra en proporcionar una solución intuitiva y accesible que permita a los usuarios registrar, monitorear y gestionar sus tareas de manera efectiva. Con características como la creación de tareas, la edición, el marcado de tareas como completadas o pendientes, y la visualización de listas de tareas completadas y no completadas, "Time Planner Plus" **busca simplificar la organización personal y fomentar un enfoque más estructurado** para la gestión del tiempo.

2.1 ¿POR QUÉ IONIC CON ANGULAR?



Al final me decanté por usar en el front-end Ionic junto con Angular, porque me **parece más completo que usar React, ya que React al ser una librería y no un framework** como Angular hay que instalar librerías externas para hacer cosas básicas que ya Angular las incorpora de serie.
Otra cosa que me gustó, es que utiliza **TypeScript** que me parece más interesante de usar porque **los datos se pueden tipar, y esto a la hora de detectar errores es mejor, y el desarrollo con el es más comprensible** al estar acostumbrado a desarrollar aplicaciones con Java.

2.2 ¿POR QUÉ SPRING BOOT CON MYSQL?

En su momento estuve dudando mucho en usar Express y MongoDB o Spring Boot y MySQL, pero al final me decante por Spring Boot y MySQL por varias razones:

- ➔ Tengo más conocimiento desarrollando en Java con Spring Boot y MySQL que en Javascript con Node, Express y MongoDB.
- ➔ Spring Boot se beneficia de la optimización de la JVM (Máquina Virtual de Java).
- ➔ MySQL es una base de datos relacional y es más fácil asegurar la integridad referencial a través de claves foráneas.
- ➔ Es un framework más completo y robusto, lo que a la larga una aplicación se haría más fácil de que escale.

3. OBJETIVOS

El principal objetivo de esta aplicación es ofrecer una solución integral y eficaz para la gestión de tareas personales y profesionales. A través de este sistema, se busca alcanzar los siguientes objetivos:

Mejora de la Productividad: Facilitar la organización diaria de actividades y tareas para mejorar la productividad individual de los usuarios. Al proporcionar una visión clara y estructurada de las tareas pendientes, los usuarios pueden priorizar eficientemente su carga de trabajo.

Gestión Efectiva del Tiempo: Ayudar a los usuarios a gestionar su tiempo de manera más efectiva, permitiéndoles asignar tiempo adecuado para cada tarea y seguir su progreso. Esto incluye la capacidad de marcar tareas como completadas y visualizar tareas pendientes y finalizadas.

Simplificación de la Planificación: Ofrecer una plataforma intuitiva y fácil de usar que simplifique el proceso de planificación. Esto se logra a través de una interfaz de usuario clara, funcionalidades de edición y eliminación de tareas, y la capacidad de visualizar tareas en diferentes estados (completadas, pendientes).

Accesibilidad y Seguridad: Garantizar que los datos de los usuarios estén seguros y que la aplicación sea accesible desde cualquier dispositivo con conexión a internet.

4. MATERIAL

Para el desarrollo de la aplicación "Time Planner Plus", se ha utilizado una combinación de tecnologías modernas y herramientas de desarrollo probadas, con el objetivo de crear una solución robusta, escalable y fácil de mantener. A continuación, se detalla el material y las tecnologías utilizadas en el proyecto:

1. **Frontend:**

- **Angular**: Marco de trabajo para aplicaciones web desarrollado en TypeScript, utilizado para construir la interfaz de usuario de la aplicación, aprovechando su arquitectura basada en componentes y servicios para una mejor organización del código.
- **Ionic**: Framework para el desarrollo de aplicaciones móviles y web progresivas, utilizado en conjunto con Angular para mejorar la experiencia de usuario con componentes de interfaz optimizados y adaptativos.

2. **Backend:**

- **Spring Boot**: Marco de trabajo para el desarrollo de aplicaciones y microservicios en Java, elegido por su facilidad de configuración, su amplio ecosistema y su integración con Spring Security para manejar la autenticación y autorización.
- **MySQL**: Sistema de gestión de bases de datos relacional, seleccionado por su estabilidad, rendimiento y soporte para transacciones, utilizado para almacenar y gestionar los datos de usuarios y tareas de la aplicación.

3. **Seguridad:**

- **Spring Security**: Framework para la seguridad de aplicaciones Java, utilizado para implementar la autenticación y autorización en la aplicación, asegurando que solo los usuarios registrados puedan acceder a sus respectivas tareas.
- **JWT (JSON Web Tokens)**: Estándar abierto para la creación de tokens de acceso que permiten la transmisión segura de información entre partes, utilizado para gestionar las sesiones de usuario y proteger las rutas de la API.

4. **Herramientas de Desarrollo:**

- **Git**: Sistema de control de versiones para el manejo del código fuente, facilitando la colaboración entre desarrolladores y el seguimiento de cambios en el proyecto.
- **Visual Studio Code y Eclipse**: Editores de código fuente avanzados, utilizados para el desarrollo del frontend y backend respectivamente, elegidos por su soporte para múltiples lenguajes, herramientas de depuración y extensiones.

5. DIAGRAMAS UML

- **CASOS DE USO**
  - **Actores**
    - **Usuario**: Persona que interactúa con el sistema para gestionar sus tareas y datos personales.
  - **Casos de Uso**
    - **Registrar Usuario**: Permite a una persona crear una nueva cuenta de usuario en el sistema. Esto implica proporcionar un nombre de usuario único, contraseña, nombre y apellidos. Este caso de uso es fundamental para que los usuarios puedan empezar a interactuar con el sistema.

- **Iniciar Sesión**: Los usuarios deben autenticarse para acceder a sus datos y gestionar sus tareas. Este caso de uso es crucial para mantener la seguridad y privacidad de la información del usuario.
- **Editar Perfil**: Una vez autenticados, los usuarios pueden actualizar su información personal, como nombre y apellidos. Esto no incluye el cambio de contraseña, que se maneja en un caso de uso separado.
- **Editar Contraseña**: Permite a los usuarios cambiar su contraseña. Es una funcionalidad importante para la seguridad de la cuenta del usuario.
- **Obtener Datos de Usuario**: Los usuarios pueden consultar sus datos personales almacenados en el sistema. Este caso de uso es importante para la revisión y confirmación de la información del usuario.
- **Crear Tarea**: Los usuarios pueden añadir nuevas tareas a su lista, especificando detalles como el título y la descripción de la tarea.
- **Listar Tareas**: Permite a los usuarios ver todas sus tareas, facilitando la organización y planificación de sus actividades.
- **Editar Tarea**: Los usuarios pueden modificar los detalles de una tarea existente, como su título, descripción y estado (completada o no completada).
- **Completar Tarea**: Cambia el estado de una tarea a completada. Es útil para que los usuarios lleven un registro de sus progresos.
- **Desmarcar Tarea Como No Completada**: Permite a los usuarios revertir una tarea a su estado no completado, en caso de que necesiten indicar que aún no se ha finalizado.
- **Eliminar Tarea**: Los usuarios pueden eliminar tareas que ya no sean necesarias o relevantes.
- **Obtener Tareas Completadas/No Completadas**: Estos casos de uso permiten a los usuarios filtrar sus tareas basándose en su estado, ayudándoles a concentrarse en lo que aún necesitan completar o revisar lo que ya han finalizado.

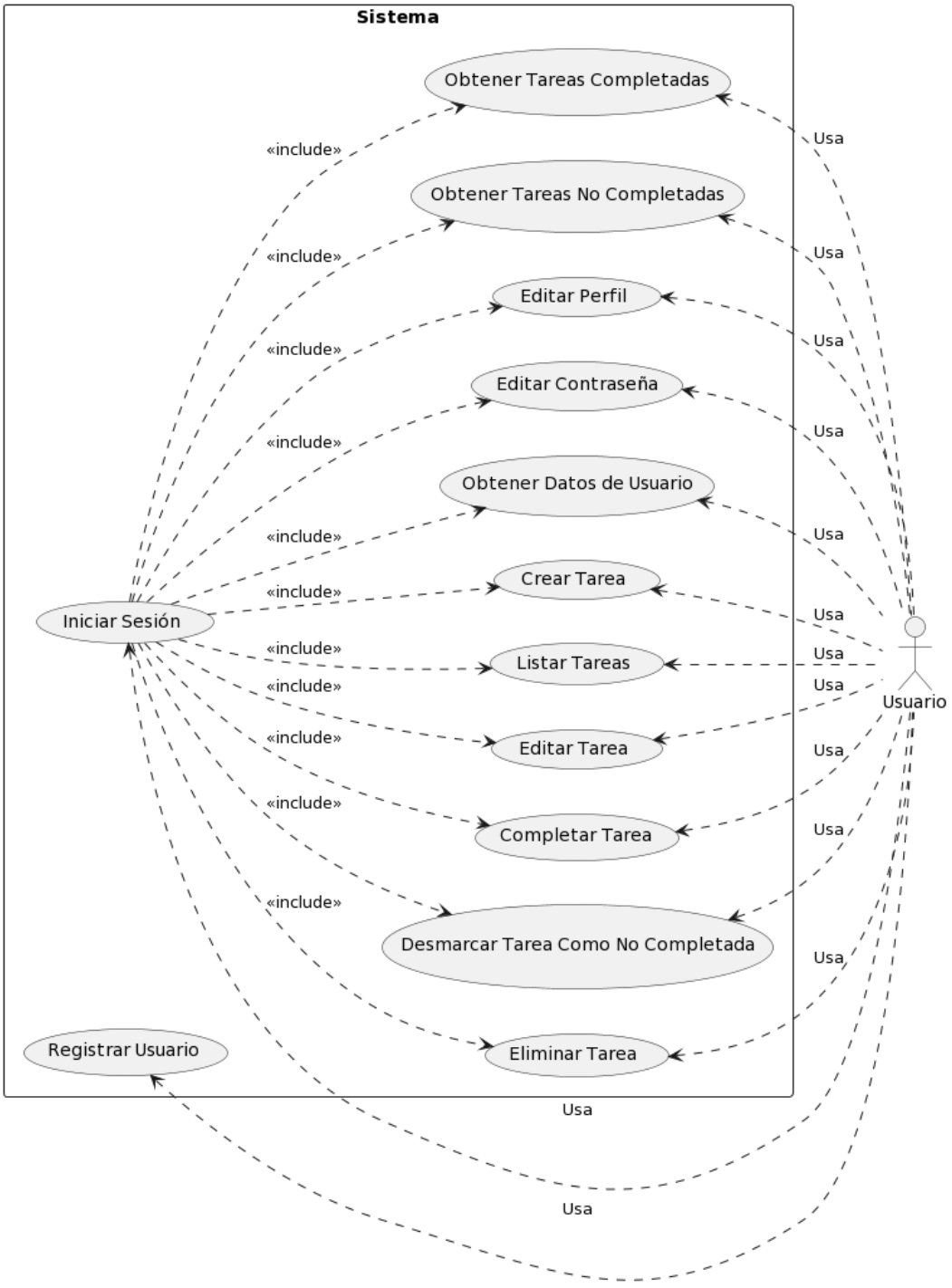
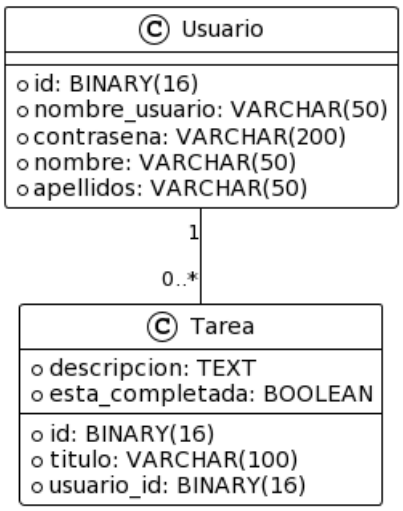


DIAGRAMA UML

He diseñado un modelo de base de datos con dos entidades principales: Usuario y Tarea. La entidad Usuario representa a cada usuario de la aplicación, incluyendo su información de identificación y contacto. La entidad Tarea describe cada tarea con su título, descripción y estado de completitud, además de estar vinculada a un usuario específico. Este diseño permite asignar tareas específicas a cada usuario y facilitar su seguimiento y

gestión dentro de la aplicación. El diagrama ilustra claramente la estructura de la base de datos y la relación entre estas dos entidades. La relación entre Usuario y Tarea se representa mediante una línea que conecta ambas clases, con la notación "1" -- "0..*", lo que indica que un usuario puede tener asociadas cero o más tareas. Esta relación es crucial para nuestro sistema, ya que permite a los usuarios crear, gestionar y visualizar sus propias tareas de manera organizada y eficiente.



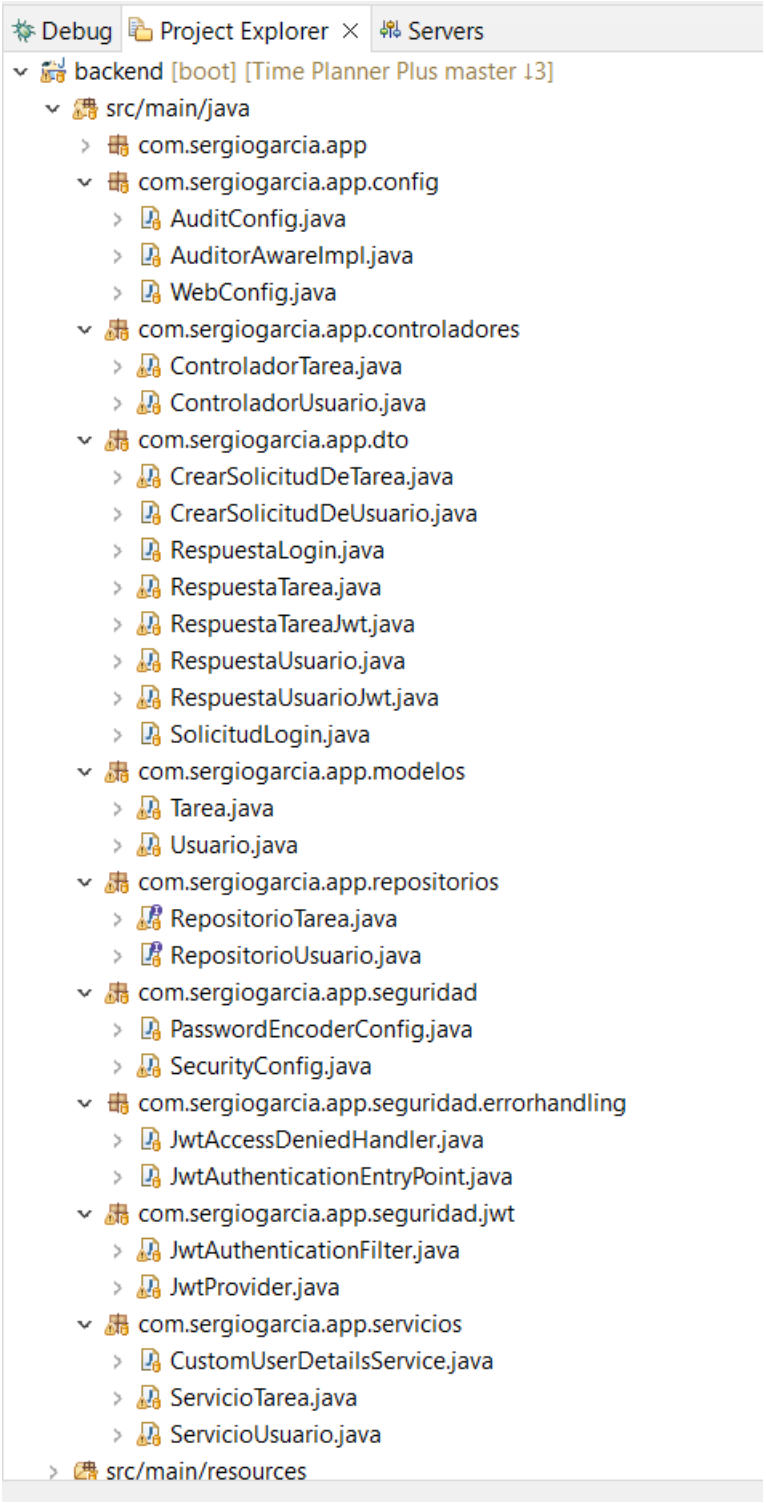
6. IMPLEMENTACIÓN

BACK-END

He creado una estructura organizativa de paquetes, para que todo esté más ordenado y con fácil acceso a cada archivo.

- **com.sergiogarcia.app.config**: Contiene clases de configuración como AuditConfig.java y WebConfig.java, que definen la configuración general de la aplicación y la auditoría.
- **com.sergiogarcia.app.controladores**: Incluye controladores como ControladorTarea.java y ControladorUsuario.java, que manejan las solicitudes del usuario y las comunicaciones entre la vista y el modelo.
- **com.sergiogarcia.app.dto**: Alberga los Objetos de Transferencia de Datos (DTOs), como CrearSolicitudDeTarea.java, que se utilizan para transferir datos específicos entre diferentes capas de la aplicación.
- **com.sergiogarcia.app.modelos**: Contiene los modelos de dominio como Tarea.java y Usuario.java, que representan las entidades y sus relaciones dentro de la aplicación.
- **com.sergiogarcia.app.repositorios**: Incluye interfaces que definen las operaciones de base de datos para los modelos.
- **com.sergiogarcia.app.seguridad**: Proporciona clases para la configuración de seguridad y manejo de autenticación y autorización.
- **com.sergiogarcia.app.servicios**: Aquí se definen los servicios que contienen la

lógica de negocio. Estos servicios son utilizados por los controladores para realizar operaciones más complejas, como la gestión de tareas y usuarios.



A continuación muestro las principales funcionalidades del back-end:

- **Controlador de Usuario**

  - Esta clase se encarga de manejar las solicitudes HTTP relacionadas con las operaciones de usuarios, como registro, inicio de sesión, edición de perfil y cambio de contraseña.

  - crearUsuario()

    - Este método permite a los nuevos usuarios registrarse en la aplicación proporcionando información básica como nombre de usuario, contraseña, nombre y apellidos. Este método se encarga de crear un nuevo registro de usuario en la base de datos.

```java
@RestController
@RequiredArgsConstructor
public class ControladorUsuario {

    private final ServicioUsuario servicioUsuario;
    private final AuthenticationManager authenticationManager;
    private final JwtProvider jwtProvider;

    //Método que crea el endpoint y la solicitud para registrar usuarios con rol de usuario
    @PostMapping("/autenticacion/registro")
    public ResponseEntity<RespuestaUsuario> crearUsuario(
            @RequestBody CrearSolicitudDeUsuario crearSolicitudDeUsuario
            ){
        Usuario usuario = servicioUsuario.crearUsuario(crearSolicitudDeUsuario);

        return ResponseEntity.status(HttpStatus.CREATED).body(RespuestaUsuario.deUsuario(usuario));
    }
```

- **Controlador de Tarea**

  - Esta clase gestiona las operaciones relacionadas con las tareas, como la creación, listado, edición, y eliminación de tareas, así como marcar tareas como completadas o no completadas.

  - listarTareas()

    - Este método devuelve una lista de todas las tareas existentes en la base de datos, permitiendo a los usuarios ver todas las tareas que se han creado, sin importar su estado de completitud.

```java
package com.sergiogarcia.app.controladores;

import java.util.List;

@RestController
@RequestMapping("/tareas")
@RequiredArgsConstructor
public class ControladorTarea {

    private final ServicioTarea servicioTarea;



    // Listar todas las tareas
    @GetMapping
    public ResponseEntity<List<Tarea>> listarTareas() {
        List<Tarea> tareas = servicioTarea.listarTareas();
        return ResponseEntity.ok(tareas);
    }
```

- **JWT Provider**
  - Esta clase se encarga de proveer funcionalidades relacionadas con la creación, validación, y manejo de tokens JWT, que son usados para la autenticación y autorización en la aplicación.
  - GenerarToken()
    - El método generarToken es responsable de crear un nuevo token JWT para un usuario que ha sido autenticado exitosamente. Este token incluye información sobre el usuario, como su ID, y tiene una fecha de expiración.

```java
    public String generarToken(Authentication authentication) {
        Usuario usuario = (Usuario) authentication.getPrincipal();

        Date expiracionTokenDateTime =
                Date.from(
                        LocalDateTime
                            .now()
                            .plusDays(jwtLifeInDays)
                            .atZone(ZoneId.systemDefault())
                            .toInstant()

                        );

        return Jwts.builder()
                .setHeaderParam("typ", TOKEN_TYPE)
                .setSubject(usuario.getId().toString())
                .setIssuedAt(new Date())
                .setExpiration(expiracionTokenDateTime)
                .signWith(secretKey)
                .compact();
    }
```

Front-End

- **Servicio de Usuario**
  - Se encarga de manejar todas las operaciones relacionadas con los usuarios, como autenticación, registro, actualización de perfiles, y manejo de sesiones.
  - Método crearUsuario()
    - Permite a nuevos usuarios registrarse en la aplicación enviando sus datos (como nombre de usuario, contraseña, nombre, apellidos, etc.) al servidor para crear una nueva cuenta de usuario.

```
@Injectable({
  providedIn: 'root'
})
export class UsuarioService {

  private apiUrl = 'http://localhost:8081/autenticacion';


  constructor(private router: Router, private http: HttpClient) {}



  crearUsuario(solicitud: CrearSolicitudDeUsuario): Observable<RespuestaUsuarioJwt> {
    return this.http.post<RespuestaUsuarioJwt>(`${this.apiUrl}/registro`, solicitud);
  }
```

- **Página de Login**
  - El método login() envía el nombre de usuario y la contraseña al servidor para autenticar al usuario. Si el inicio de sesión es exitoso, guarda el token proporcionado por el servidor para mantener la sesión del usuario y redirige a la página principal. Si hay un error, muestra un mensaje indicando que el inicio de sesión ha fallado.

```
export class LoginPage {
  nombreUsuario: string = '';
  contrasena: string = '';
  loginError: boolean = false; // Indica si hay un error de login

  constructor(private usuarioService: UsuarioService, private router: Router) { }

  login() {
    this.loginError = false;
    if (this.nombreUsuario && this.contrasena) {
      // Realiza la llamada al servicio de login
      this.usuarioService.login(this.nombreUsuario, this.contrasena).subscribe({
        next: (res) => {
          console.log('Login exitoso', res);
          localStorage.setItem('token', res.token);
          // Redirige al usuario a la página principal o donde corresponda
          this.router.navigate(['/main/tareas']);
        },
        error: (error) => {
          console.error('Error en el login', error);
          this.loginError = true; // Muestra mensaje de error en el template
        }
      });
    }
  }
```

7. CONCLUSIÓN

En conclusión, me llevo una gran experiencia realizando este proyecto, porque creía que no podría ser capaz de hacer funcionar una aplicación full stack. Me ha servido mucho este proyecto para ver que soy capaz de investigar por mi cuenta y filtrar bien la información que hay en internet para luego implementarla en mis aplicaciones, y seguir en continuo aprendizaje.

8. BIBLIOGRAFÍA

https://openwebinars.net/academia/aprende/api-rest-segura-spring-boot-jwt/

https://openwebinars.net/academia/aprende/angular/

https://ionic.io/docs

https://angular.io/docs