**Construcción de aplicación para el manejo de tareas.**La construcción de la siguiente aplicación para el manejo de tareas se realizó usando los siguientes lenguajes y frameworks:

Front-End: Angular versión 13.1.3

Back-End: Spring Boot 2.6.2

Java 11

Como gestores de dependencias para la aplicación tenemos:

NPM 8.1.2

Maven 3.6.3

Para correr la aplicación es necesario tener instalado Java, Maven, Node y el cliente Angular.

Comandos útiles para la instalación de dependencias, compilación y arranque de los componentes:  
  
mvn install

mvn clean install

mvn compile

mvn spring-boot:run (en la raíz del proyecto spring)  
  
ng serve (en la raíz del proyecto angular)  
  
  
Paquetes o dependencias resaltables en la implementación del backend:

-spring boot starter web: paquete complete para el Desarrollo de aplicaciones web y construcción de microservicios

-h2database: motor de base de datos relacional ligero el cual corre en memoria junto con el backend

-jsonwebtoken: paquete para la gestión de tokens de seguridad

-springfox swagger2: dependencia para la gestión de documentación de apis en swagger

Paquetes o dependencias resaltables en la implementación frontend:

-Angular material: biblioteca completa para la construcción de apps angular con estilos modernos, animaciones y otras características

-jwtdecode: dependencia para la gestión de tokens jwt, en este caso el decodificado de los mismos

-ngxtoastr: paquete para el despliegue de notificaciones  
  
  
Comandos usados en la construcción del frontend

ng new FrontendCanvas

Para instalar material

ng add @angular/material

ng add @angular/cdk

ng add @angular/animations

Para instalar toastr notificaciones

npm i ngx-toastr

Para instalar jwt decode

npm i jwt-decode

Agregar los estilos para las notificaciones en el angular.json en los styles

"./node\_modules/ngx-toastr/toastr.css",

"src/styles.css"

Toolbar, se construyó usando flex layout por tal motivo no es necesario crear estilos o código adicional

Creación de 5 componentes adicionales al principal creado por angular (app.component.ts)

ng generate component tareas-home

ng generate component tareas-registro

ng generate component usuarios

ng generate component usuarios-registro

ng generate component usuarios-login

Comentarios generales de la implementación  
Backend: Se construyeron los servicios de la aplicación usando una arquitectura en capas, en la que se separan diferentes responsabilidades:  
-Controladores: se encargan de recibir la petición y de retornar el response, no contiene ningún tipo de lógica de negocio

-Servicios: se encargan de comunicar con la capa mas baja del modelo (DAO) y de retornar la respuesta a la capa más alta (el controlador), contienen toda la lógica de negocio necesaria para la operación

-Dao: Son interfaces que extienden del repositorio general de JPA, contienen todas las operaciones a realizar contra la base de datos

-Config: Contiene configuración relevante para la integración del backend con el frontend especifico (localhost:4200), contiene el filtro jwt que intercepta las peticiones realizadas al backend para analizar el token enviado, contiene la configuración del swagger para la documentación de los servicios

De manera muy general, Dto contiene objetos para gestionar los request y response de los servicios, Entidades contiene clases que mapean contra las tablas de la bd y permiten ejecutar las operaciones sobre la misma, Útil contiene constantes para la aplicación y el manejo de errores por log en archivo plano, Validadores contiene clases para validar la estructura y datos de los request de los servicios y Adaptadores contiene métodos para generar diferentes respuestas de la aplicación de manera estandarizada

Se incluye en resources dos scripts sql, schema.sql con la configuración inicial de las tablas a nivel de base de datos y las secuencias y el script data.sql que contiene insert de datos iniciales. Ambos scripts se ejecutan en el arranque del backend.

Frontend: En primer lugar se usan environments para especificar el path base de todos los microservicios desarrollados; El componente principal contiene el toolbar de la aplicación únicamente. Se generaron 5 componentes para la aplicación, uno con el login, dos con la gestión de los usuarios y dos con la gestión de las tareas.  
Se creo un modulo para la implementación de las notificaciones toast, el cual contiene los 4 métodos para mostrar los diferentes tipos de mensajes. Por ultimo se crearon 3 servicios, uno para la decodificación del token almacenado en la sesión storage, otro para el llamado a los servicios de usuarios y otro para el llamado a servicios de tareas, estos servicios usan el cliente HttpClient de angular.

Funcionamiento de la aplicación

La raíz de la aplicación esta en la url: <http://localhost:4200/>

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente  
Se debe realizar el login con el botón ingresar, haga uso del usuario creado durante el arranque del backend para poder acceder a la app:  
usuario: usuario2022

Password: usuario2022  
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente  
  
Si hay algún error en los datos se generará un error:  
  
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En caso de autenticación exitosa se mostrará el home con las tareas sin asignar disponibles de la siguiente manera:  
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente  
Adicional se seteará en el sesión storage del usuario el Access\_token para poder consumir el resto de servicios:  
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente  
  
La consulta general de usuarios se hace por el botón dispuesto en el toolbar para tal fin  
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Puede consultar usuarios por el filtro de nombre el cual ejecuta una búsqueda con un like en la base de datos o filtrando por el tipo de usuario (administrador, miembro, invitado); Por defecto hay un usuario creado, que es el que se carga en el script data.sql del backend.  
Con los botones dispuestos en la grid en cada registro puede gestionar el comportamiento de los usuarios. A continuación una operación de guardado de nuevo usuario:  
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente  
  
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente  
Todos los campos del formulario son obligatorios, si todas las validaciones sobre los datos del formulario pasan se crea el nuevo usuario en el sistema. El password se guarda hasheado en la bd para mayor seguridad, este hash no es posible devolverlo a la cadena original, solo comparar una cadena ingresada a ver si coincide con el hash original, esta mecánica es la que aplica en el login.

En la opción de tareas aparece la grid consulta con el siguiente registro precargado del data.sql del backend:  
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente  
  
Se tiene un filtro por estado de la tarea, por defecto carga las tareas que no están asignadas a ningún usuario (where idUsuario is null)  
Se pueden crear tareas con el botón dispuesto en la parte inferior de la pantalla.  
Interfaz de usuario gráfica

Descripción generada automáticamente  
  
El campo estado se deshabilita ya que por defecto las tareas se crean sin asignación para que los diferentes usuarios las tomen paulatinamente.  
  
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente  
  
Enlaces de interés:  
Consola web h2: http://localhost:8060/sophos/servicios/h2/ (User: sa, pass: )  
Documentacion swagger: http://localhost:8060/sophos/servicios/v2/api-docs/

Backend resources: <http://localhost:8060/sophos/servicios/swagger-resources>  
Frontend App: http://localhost:4200/