Práctica de DBCS

Curso 2021/22

Laboratorio Práctica 2: Contenedores y servicio de autenticación

Introducción

En esta segunda práctica se va a construir un nuevo microservicio para la autenticación y se van a contenerizar tanto la API de gestión de usuarios, como el microservicio de autenticación, base de datos y Angular.

Descripción

Para esta práctica se necesita construir lo siguiente para la aplicación web:

- Implementar una nueva API para dar servicio de autenticación que permitirá iniciar sesión a los usuarios
- Levantar un contenedor para esta nueva API de autenticación
- Tener un contenedor para la API REST con Spring desarrollada en la práctica 1
- Levantar un contenedor para la aplicación web desarrollada con Angular que conecte con la API REST
- Tener un contenedor para la persistencia, base de datos MySQL/MariaDB del microservicio de usuarios

Esta aplicación permitirá realizar las acciones CRUD para la gestión básica y autenticación de usuarios. Teniendo esto en cuenta, la API de usuarios debe cumplir la siguiente interfaz:

- GET /users: Devolverá una lista de todos los usuarios existentes en la base de datos.
- GET /users/{id}: Devolverá el usuario con el id especificado.
- POST /users: Creará un nuevo usuario en la base de datos.
- PUT /users/{id}: Modificará el usuario con el id especificado.
- DELETE /users/{id}: Borrará el usuario con el id especificado.
- GET /users?enable=false: Devolverá la lista de los usuarios inhabilitados.
- PUT /users/enable?user_id=*Lista_Ids_Usuarios_A_Activar*. Activará (modificará el campo “enabled” de la tabla) de aquellos usuarios cuyo “id” esté incluido en la lista
- GET /users?email=email_a_consultar: Se obtendrá el usuario con este email o ninguno en caso de no existir

Para la API del microservicio de autenticación de usuarios, la interfaz será la siguiente.

- POST /login: Se le pasará el email y la contraseña para validar. Devolverá un token JWT (ver Desarrollo de la práctica)

El modelo del usuario no varía de la práctica anterior.

Desarrollo de la práctica

Se deberán realizar las acciones necesarias para crear la API de autenticación que permita a los usuarios que utilicen la aplicación, iniciar sesión y autenticarse con su contraseña. Las tecnologías para este nuevo microservicio son de libre elección, es decir, se puede desarrollar en el lenguaje que se prefiera, no es necesario utilizar Spring Boot. El proceso de autenticación será introduciendo el email y la contraseña. Si el usuario existe y la contraseña es correcta se devolverá un token JWT que será almacenado en Angular. En caso contrario devolverá un error HTTP 403 Forbidden. Del mismo modo, Angular deberá proteger las rutas que permiten la gestión de usuarios para detectar y bloquear intentos del CRUD de usuarios sin estar correctamente autenticado. Referencias en la siguiente página.

Por otro lado, el servicio de usuarios deberá ser actualizado para que se permita la comunicación con el microservicio de autenticación. Además, la contraseña que almacena de los usuarios se requiere que se almacene de manera encriptada (md5, sha256, etc).

Por la parte del cliente web (front), la aplicación Angular deberá ser actualizada con el formulario de login y la protección del resto de vistas si no se está autenticado. También habrá una opción para poder hacer logout. Se valorará la usabilidad y diseño. Se recomienda utilizar alguna librería como Bootstrap, Material, etc. para la maquetación. Se puede revisar la documentación de Boostrap sobre componentes y layout en la siguiente página. La aplicación deberá tener los modelos, servicios, componentes, módulos y configuración de rutas necesarios.

Para la puesta en marcha de todos estos servicios, se configurará un único docker-compose para crear las imágenes necesarias y orquestación de cuatro contenedores: microservicio API de gestión de usuarios, Base de datos de gestión de usuarios, Angular con el front y microservicio API de autenticación.

Se aconseja utilizar variables de entorno para aquellos valores que pueden cambiar de forma dinámica en los diferentes entornos de ejecución de un contenedor. Ejemplo: API key, key secrets, etc.

El código debe estar correctamente estructurado en proyectos, carpetas, etc., tal y como se ha indicado en los tutoriales.

Se debe realizar el tratamiento de excepciones.

Es necesario pensar bien la estructura, el control de versiones, etc., puesto que esta práctica es la base sobre la que se trabajará en la siguiente.

Todas las funcionalidades añadidas que aporten seguridad en la información y buenas prácticas serán valoradas positivamente.

Evaluación

Se evaluará:

- Código creado: valorando por parte del profesor el material entregado
- Funcionamiento: mediante defensa oral del proyecto durante las sesiones de laboratorio designadas.

Se valorará:

- Claridad en la organización de la aplicación.
- Comentarios explicativos en el código.
- Funcionalidades añadidas.

Entrega

Esta práctica será un fork de la anterior en el Gitlab de la Escuela. Se creará un grupo llamado practica2-dbcs-2021 y se hará un fork del proyecto de la práctica 1 en este grupo. Es necesario desvincular el nuevo proyecto del origen del fork. No se puede realizar ningún cambio sobre el repositorio de la práctica 1. Sobre el nuevo proyecto que reside en el nuevo grupo practica2-dbcs-2021 se dará acceso al profesor correspondiente con el rol de *reporter.* Es deseable otorgar acceso al profesor mientras se esté trabajando puesto que permite ir viendo el progreso y atender mejor las dudas cuando sea necesario. Se podrán hacer cambios en el repositorio mientras no haya finalizado la fecha límite de entrega.

Documentación que entregar a través del campus virtual:

- Actualización de la descripción de cómo poner en marcha la aplicación.
- Actualización del diagrama de despliegue con los componentes/artefactos de la aplicación donde se vea la estructura de componentes en cada entorno de ejecución, en cada nodo computacional y con sus protocolos de comunicación.
- Diagrama de componentes de Angular.

Límite de entrega del código y la documentación, así como de la defensa: ver documento de planificación.