Acceso a Datos

UT03 Spring Boot

Proyecto – API REST Segura

Prof. Diego Linares Ortiz

UT03 Proyecto
API REST Segura

Puesta en marcha

En este ejercicio práctico vas a realizar la implementación de una API REST segura donde aplicaremos todas las técnicas que hemos visto hasta ahora. **Es imprescindible que vayas paso a paso, para así asegurar que la aplicación se construye sobre unos cimientos sólidos y estables.**

Para empezar el proyecto deberás seguir una serie de pasos:

1- Plantea una idea de API REST, la cual deberás desarrollar. Esta idea de aplicación deberá tener unas tablas a las cuales se deberá acceder haciendo uso de la API.
2- Una vez tengas la idea, deberás plantear las tablas que vas a gestionar. Por ejemplo:
    a. Idea: Aplicación de gestión de reservas de viaje.
        i. Tabla 1: Usuarios
        ii. Tabla 2: Reservas
        iii. Tabla 3: Detalles_Reserva
    b. Idea: Aplicación de gestión de viajes en grupo
        i. Tabla 1: Usuarios
        ii. Tabla 2: Viajes
        iii. Tabla 3: Destinos
    c. Idea: Aplicación de reparto de tareas del hogar
        i. Tabla 1: Usuarios
        ii. Tabla 2: Tareas_Domesticas
        iii. Tabla 3: Asignaciones_Diarias
    d. Idea: Aplicación de reparación/tareas de mecánica a domicilio
        i. Tabla 1: Usuarios
        ii. Tabla 2: Talleres
        iii. Tabla 3: Citas_Taller
    e. Idea: Aplicación de planificador de presupuesto
        i. Tabla 1: Usuarios
        ii. Tabla 2: Tipos_Gastos
        iii. Tabla 3: Gastos_Diarios
3- Una vez tengas la idea y las tablas planteadas, deberás plantear todos los campos que las tablas deberán tener así como el tipo de dato y las restricciones que vas a aplicar.
4- Cuando hayas cumplido el punto 4, ya puedes pasar a realizar un diagrama de entidad-relación de las tablas.
5- Cuando hayas cumplido el punto 5, puedes pasar a crear el repositorio y demás especificaciones.

Objetivo

El objetivo del proyecto es que realices una API REST segura desde principio hasta el fin. Deberás terminar teniendo una API que cumplirá con los principios SOLID, segura, probada y bien documentada.

Enunciado

Construye e implementa una API REST segura siguiendo las siguientes restricciones:

1. Realiza los pasos explicados en la introducción del proyecto. Una vez realices esos pasos, puedes pasar al punto 2 del enunciado. NO SE CORREGIRÁ NINGÚN PROYECTO QUE NO HAYA ENTREGADO UN PLANTEAMIENTO PREVIO DEL MISMO
2. Crea un repositorio en GitHub, que sea público y que tenga al profesor como colaborador directo del repositorio. Crea un README.md donde plantees los siguientes puntos:
    a. Nombre del proyecto
    b. Idea del proyecto
    c. Justificación del proyecto
    d. Descripción detallada de las tablas que intervendrán en el proyecto
3. En el README anteriormente construido deberás incluir lo siguiente (aparte de lo ya descrito)
    a. Indicar los endpoints que se van a desarrollar para cada tabla
    b. Describir cada uno de los endpoints. Realiza una explicación sencilla de cada endpoint.
    c. Describe la lógica de negocio que va a contener tu aplicación.
    d. Describe las excepciones que vas a generar y los códigos de estado que vas a poner en todos los casos.
    e. Describe las restricciones de seguridad que vas a aplicar dentro de tu API REST

    SI FALTARA CUALQUIER PUNTO DE LOS ANTERIORMENTE DESCRITOS, EL PROYECTO SE DARÁ COMO INVÁLIDO
4. Realiza la implementación de la API siguiendo las directrices que has marcado en la parte de la documentación que ya has desarrollado. La aplicación deberá cumplir con los siguientes mínimos:
    a. Debe haber una entidad Usuario que:
        i. Tenga al menos los campos *username*, *password* y *roles*.
        ii. La password debe almacenarse "*hasheada*" en la BDD

   b. Debe haber al menos 2 entidades más además de la entidad Usuario, que:
      i. Tenga un CRUD mínimo implementado
      ii. Tenga diferentes restricciones de acceso a los endpoints asociados a dichas entidades.
   c. Debes implementar la seguridad de la aplicación haciendo uso de Spring Security y que cumpla:
      i. Que se usa un cifrado asimétrico por clave pública y privada para el control de acceso.
      ii. Que se usa JWT para el control de acceso.
      iii. Que hay variedad de restricciones de control de acceso a los endpoints (*lo mismo que el punto b.ii*)
5. Una vez realizada la implementación de la API, realiza pruebas del buen funcionamiento de la misma.
   a. Usa insomnia, swagger o postman para probar los diferentes endpoints de la API
   b. Plantea las pruebas que vas a realizar para comprobar el buen funcionamiento de la API implementada
   c. Documenta las pruebas realizadas para mostrar que la API ha cumplido con su funcionamiento.
6. Cuando finalices toda la implementación y las pruebas podrás concluir el proyecto incluyendo los siguientes puntos a la documentación:
   a. ¿Qué tecnologías has usado?
      i. Indica las dependencias que has incluido en tu proyecto.
      ii. Indica el software que has usado (IntelliJ, Insomnia, XAMPP, navegadores, etc)
      iii. Describe brevemente dichas tecnologías y su propósito dentro del proyecto
   b. ¿Qué es una API REST? ¿Cuáles son los principios de una API REST? ¿Dónde identificas dichos principios dentro de tu implementación?
   c. ¿Qué ventajas tiene realizar una separación de responsabilidades entre cliente y servidor?

Conclusión y entrega

Para concluir el proyecto, deberás realizar un documento en formato ***.pdf*** que contenga toda la documentación que se pide para el proyecto. Entrega también un enlace al repositorio de GitHub. Mucho cuidado con los commits, si hay muy pocos commits se investigará la validez del proyecto.

RAs y CEs evaluados

2. Desarrolla aplicaciones que gestionan información almacenada en bases de datos relacionales identificando y utilizando mecanismos de conexión.

a) Se han valorado las ventajas e inconvenientes de utilizar conectores.

b) Se han utilizado gestores de bases de datos embebidos e independientes.

c) Se utilizado el conector idóneo en la aplicación.

d) Se ha establecido la conexión.

e) Se ha definido la estructura de la base de datos.

f) Se han desarrollado aplicaciones que modifican el contenido de la base de datos.

g) Se han definido los objetos destinados a almacenar el resultado de las consultas.

h) Se han desarrollado aplicaciones que efectúan consultas.

i) Se han eliminado los objetos una vez finalizada su función.

j) Se han gestionado las transacciones.

6. Programa componentes de acceso a datos identificando las características que debe poseer un componente y utilizando herramientas de desarrollo.

a) Se han valorado las ventajas e inconvenientes de utilizar programación orientada a componentes.

b) Se han identificado herramientas de desarrollo de componentes.

c) Se han programado componentes que gestionan información almacenada en ficheros.

d) Se han programado componentes que gestionan mediante conectores información almacenada en bases de datos.

e) Se han programado componentes que gestionan información usando mapeo objeto relacional.

f) Se han programado componentes que gestionan información almacenada en bases de datos objeto relacionales y orientadas a objetos.

g) Se han programado componentes que gestionan información almacenada en una base de datos nativa XML.

h) Se han probado y documentado los componentes desarrollados.

i) Se han integrado los componentes desarrollados en aplicaciones.

Indicadores	Niveles de logro		
	Bien	Regular	Insuficiente
	100%	50%	0%

RA 6. Desarrolla aplicaciones de acceso a almacenes de datos, aplicando medidas para mantener la seguridad y la integridad de la información.

e) Se han utilizado conjuntos de datos para almacenar la información.

h) Se han probado y documentado las aplicaciones.

Indicadores	Niveles de logro		
	Bien	Regular	Insuficiente
	100%	50%	0%