Evaluación Practica Java

Evaluación Api RestFull, java SpringBoot

# Información general:

Pequeña aplicación con seguridad sencilla que tiene como fin crear a un usuario donde se valida su correo y el formato de este, y luego muestre al usuario creado cuando la información proporcionada es correcta.

# Como abordar el problema:

Lo primero era abordar la problemática desde lo mas “simple”, crear nuestro diagrama para ir paso a paso con nuestra solución:

Diagrama

Descripción generada automáticamente

Con esto ya tenia una idea clara de lo que tendría que hacer, ahora lo que venia era la forma de organizar el proyecto para llegar a la solución.

En primer lugar decidí crear mi proyecto con Start.spring.io :

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Como se ve en la imagen ocupe Maven para le gestión, ocupe la versión 2.7.8 de Spring Boot, el packging con JAR y java en su versión 1.8,

En las dependencias para el proyecto (para comenzar), fueron:

1. Spring Boot Dev Toools : nos permite reiniar de forma automática nuestra aplicación cada vez que se produce un cambio en el código.
2. Lombok: Permite eliminar el código repetitivo y nos facilita generar código, en este caso lo utilizaremos para crear nuestros Getters, Setters, contructores, un ejemplo de cómo lo ocupe en el Código:

Texto

Descripción generada automáticamente

1. Spring Web: Es la dependencia que nos permite trabajar con mvc y la creación de nuestra Api RestFull.
2. Spring Data JPA: Nos permite simplificar el desarrollo con una serie de interfaces para implementar la capa de persistencia a nuestra aplicación.
3. Spring Security: Nos dota de una serie de servicios para aplicar seguridad a nuestra aplicación, autentificación y autorización

Luego de tener las dependencias principales para comenzar a trabajar decidí organizar mi Código de la siguiente manera en diferentes package:

* Controller: es donde se procesan nuestras llamadas ya sean nuestras “request” o “response”, las clases que son “controller” deben tener la anotación “@RestController” y una respectiva “dirección” que será el nombre que le daremos a dicho controller la anotación se vería de esta forma “@RequestMapping(“/dirección”)”.
* Entity: Donde guardare mis entidades, que son la representación de información, en este caso cree, User, UserInfo, Phone.
* Repository: En este Package guardare los componentes encargados de resolver y gestionar la persistencia, en mi caso decidí extender mis interfaces con JpaRepository, que es nuestra forma de interactuar con la BD.
* Security: En este Package esta todo lo relacionado con la seguridad, JWTAtuthorizationFilter, SwaggerConfig, WebSecurityConfig
* Services: En este Package esta todo lo relacionado a la lógica que nos ayudara a relacionar,conectar nuestros distintos repositorios y que aglutine la funcionalidad de estos.
* Utils: Package donde se almacenan todas las clases o interfaces que podremos reutilizar

Texto

Descripción generada automáticamente

La lógica para la creación y orden fue la siguiente:

1. Configurar Properties, Url, drivers, user, pass, ddl-auto.
2. Creación y configuración de entidades, en esta parte lo mas importante es fijarse bien en relacionar las distintas entidades con los respectivos @OneToMany y @ManyToOne, @OneToOne, nuestra clase “Principal”, Acá el ejemplo principal :

Texto

Descripción generada automáticamente

Acá relaciono las diferentes entidades con la anotación “Uno es a muchos” y “uno es a uno”.

1. Creación de repository de las diferentes entidades para empezar con la lógica para gestionar las consultas.
2. Crear los servicios para relacionar los diferentes repositoris y agrupar mis consultas, la lógica y ordenar mi Código.
3. Configurar Swagger para la creación de la documentación.
4. Crear las clases para configurar la autentificación que consiste en configurar una serie de filtros para que la seguridad de spring security ya no se comporte por defecto y acepte nuestras configuraciones propias, en este caso ocupe para que la seguridad funcionara con el HEADER = “Authorization” y PREFIX = “Bearer “
5. Una vez configurada la seguridad , segui con los controllers para tener la “salida” de mis endpoints y posteriormente probarlos

Guía para probar con Postman:

1. Utilizar Endpoint : GET <http://localhost:8080/token> y copiar el token que arroja.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Utilizar el Endpoint : POST <http://localhost:8080/api/user> , con la siguiente estructura en su BODY

{ "name": "daniel segovia", "email": "danielsegoviavega@nisum.cl", "password": "nisum", "phones": [ { "number": "1234567", "citycode": "1", "contrycode": "1" } ] }

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

   Descripción generada automáticamenteMostrar usuario con el siguiente Endpoint : GET <http://localhost:8080/api/user>