***API REST Registro de Usuarios***

Esta aplicación proporciona una API RESTl para el registro de usuarios. El servicio acepta y retorna solamente data en formato JSON, incluyendo mensajes de error. La aplicación está construida como se solicito utilizando Java 17, Spring Boot, y utiliza JWT para la generación de tokens de acceso.

**Tecnologías utilizadas para la solución**

* Java 17
* Spring Boot
* Maven
* H2 Database (base de datos en memoria)
* JPA (Hibernate)
* JWT
* Pruebas Unitarias con Junit y Mockito
* Swagger para la documentación de la API

**Base de Datos**

Esta aplicación utiliza H2, una base de datos en memoria que se carga automáticamente al iniciar la aplicación. Para acceder a la consola de H2:

* URL: http://localhost:8080/h2-console
* JDBC URL: jdbc:h2:mem:testdb
* Usuario: sa
* Contraseña: (vacío)

**Script de creación de base de datos**

La base de datos H2 se genera automáticamente con JPA y no requiere scripts manuales. Sin embargo, se pueden consultar las entidades definidas en el código fuente para entender su estructura puesto que la secuencia, el trigger y la tabla son creadas desde el proyecto.

**JWT**

El token JWT se genera durante el registro de usuario y se devuelve en la respuesta, lo que indica que el usuario fue registrado y autenticado por medio de dicho token.

**Swagger**

La documentación de la API está disponible en Swagger, siempre y cuando la aplicación se encuentre instalada y ejecutando:

URL Swagger: <http://localhost:8080/swagger-ui/index.html>

**Pruebas**

Las pruebas unitarias están incluidas en el proyecto y se realizaron con Junit y Mockito

**Diseño**

El diseño de la aplicacion sigue los principios de separación de responsabilidades, utilizando patrones de diseño y buenas prácticas:

* DTOs para la transferencia de datos entre las capas.
* Servicios para la lógica de negocio.
* Controladores REST para gestionar las solicitudes HTTP.
* Repository Pattern para la interacción con la base de datos.
* Inyección de dependencias para gestionar las dependencias entre componentes.
* Manejo de errores estandarizado con respuestas en formato JSON.

Diagrama

**Instalación del proyecto**

1. **Clonar el repositorio**

Se debe clonar el repositorio de GitHub Ubicado en la ruta

1. **Abrir el proyecto desde un IDE(En este caso yo utilice Spring Tools Suite)**
2. **Compilar el proyecto con mvn**
3. **Validar Cobertura de código con Junit para garantizar las pruebas unitarias a las clases creadas en el desarrollo**
4. **Ejecutar la Aplicación**

La aplicación se ejecutará en <http://localhost:8080>

1. **Endpoint de la API**

**Registro de usuario**

**URL: /api/register**

**Método HTTP: POST**

1. **Validar Swagger del servicio**
2. **Formato de Request**

{

  "name": "Pedro Mesa",

  "email": "usuario@dominio.cl",

  "password": "Password/12",

  "phones": [

    {

      "number": "123456789",

      "citycode": "01",

      "contrycode": "+56"

    }

  ]

}

1. **Formato de Response**

{

    "id": "0cae411a-01c4-4419-93d7-180b661ca6c4",

    "created": "2024-09-05T23:31:25.218+00:00",

    "modified": "2024-09-05T23:31:25.218+00:00",

    "lastLogin": "2024-09-05T23:31:25.218+00:00",

    "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJQZWRybyBNZXNhIiwiZW1haWwiOiJ1c3VhcmlvQGRvbWluaW8uY2wiLCJleHAiOjE3MjU1ODI2ODV9.yccgM47Ls13rid-jY68CBy1B\_-jfumkTl9hVPzMdb6dBYzSMgXPaJXso-IvtSCF8wjgOySo8ZgYiqZ7c6Uopng",

    "isActive": 1

}

1. **Formato de Error**

{

    "statusCode": 400,

    "mensaje": "Error, Formato de email inválido. Debe seguir el formato aaaaaaa@dominio.cl"

}

1. **Abrir la consola de la BD H2**
2. **Se consulta la tabla para validar que esta vacia**
3. **Importar la colección de postman que se adjunto en el repositorio**
4. **Ejecutar pruebas desde postman**