2. Microservicio Usuario

Configurando el contexto de persistencia JPA/Hibernate

Volvi a crear el proyecto msvc-usuario con las siguientes dependencias



* Configurando el aplication.properties (Mi modo en paralelo con el curso)
* Configurando la carpeta domain (models.entity)
  + Clase Usuario
    - Atributos
    - Geter and setter

 Introducción a los repositorios (CrudRepository de Spring Data JPA)

Toda la documentación del repositorio la encuentro en <https://spring.io/projects/spring-data> y puedo verificar toda la implementación en los motores de DB

* Para el proyecto se implementa spring Data Repository

<https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories>

* SI quiero agregar un método extra lo puedo realizar por query methods y en la siguiente documentación encontrare palabras claves para realziar el método deseado <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods>
* Tambien puedo realizar una consulta por medio de una anotación sobre el metodo <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.at-query> las consultas son al METODO y no a la DB

Implementando el componente Repository de acceso a datos

* Creamos un nuevo paquete Repositories y la interface que contendrá los métodos para el repositorio CRUD -- ClaseDao

 Implementando el componente Service

El ingeniero implementa la clase service en el paquete repository, yo lo agregar por separado una paquete que contenga su implementación y su interface service

* Cree paquete Service
  + Cree Interface Service
  + Cree Clase ServiceImpl (ANotacion Service)
    - Configurando SeviceImpli, en autoamatico(@Override y un pull) el autowirred sobreescribe el método en la interface

DATO: como estoy implementado CRUD de repository todos los métodos son transaccionales porque el método es envuelto por un (BEGIN COMIT & ROLLBACK)

DATO: Comando para dar Formato es **Ctrl + Alt + L**.

 Implementando el controlador RestController y metodos handler GET

* Creo la carpeta Controllers
* Creo la clase usuario Controller ojo EXISTEN DOSTIPOS DE controladores en spring El general que es @RestController y otro controlador para manejar errores @Advice
* Para cada método INTERNO DEBE TENER UN TIPO DE PETICION HTPP
* Puedo conlocar la ruta a nivel Raiz del Controlador por medio de @RequestMapping o directamente la ruta al método, para este caso utilizamos ruta directamente al método

 RestController y metodos handler POST y PUT

Recordar que estamos utilizando Repository CRUD de este modo todas las operaciones deben se envueltas por su COMMIT, ROLBACK, BEGGIN para a fectar la DB ….

RestController y metodos handler DELETE

Se realiza igual se usa el método para la búsqueda por id ar aluego parasar al metdodo eliminar y realizar la operación si no pues enviar que no realizo ningua operación si le llega un null

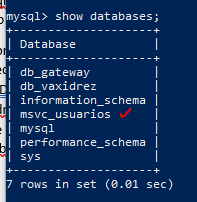
 Configurando en el application.properties conexión MySQL

Porpiedades del proyecto y datasource

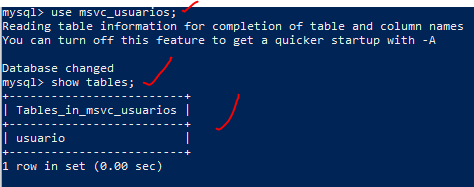
Instalando Base de Datos MySQL

El ingeniero instala el mysql y el worbench, para este caso la administración de mysql utilice un contenedor de un proyecto de vaxi-microservices

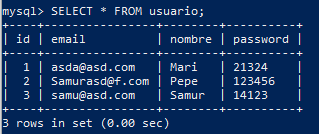
* Ingreso al sh del container
  + docker exec -it mysql-vaxi bash
* Ingreso al sh de la DB(nombre del USUARIOimg -u(user) -p(password))
  + mysql -uadmin -pAdminGeneral2026$
* Valido los Bases de Datos de mi Usuario admin
  + show databases;



* Uso la base datos de MSVC\_USUARIOS y valido sus tablas
  + use msvc\_usuarios;
  + show tables;



* Consulto la tabla Usuario (Pruebas Postman Ya realizadas)



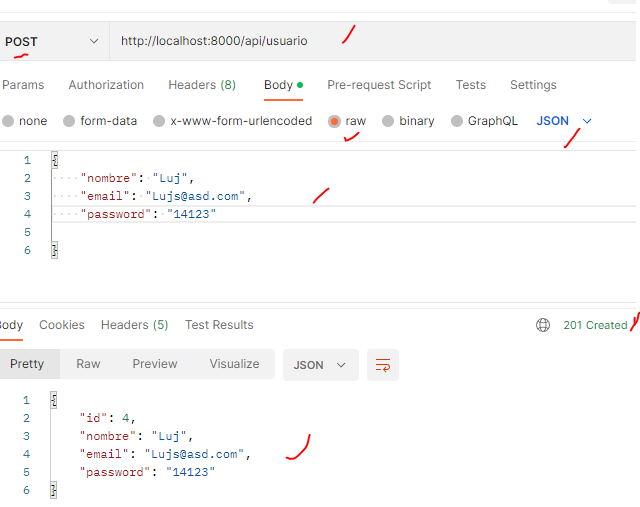
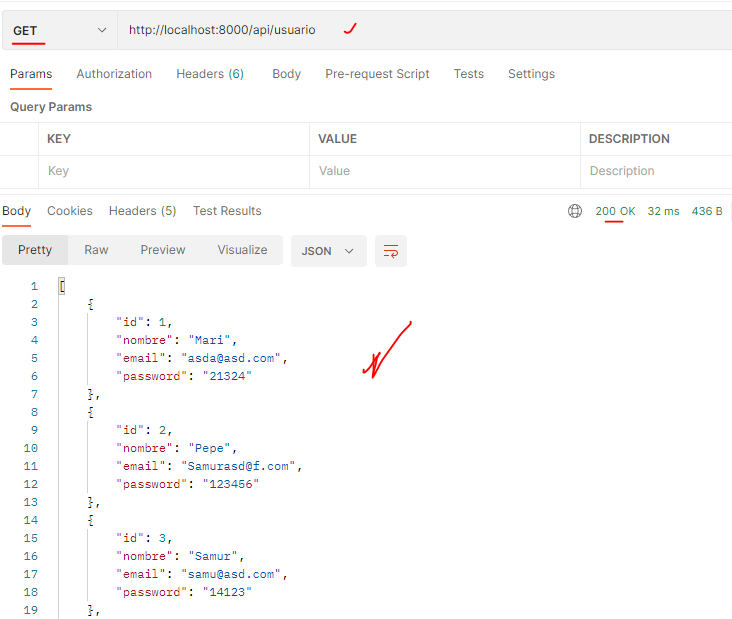
Probando la conexión a MySQL

* Configuro los datos de conexión del contenedor a al servicio usuario en app.properties

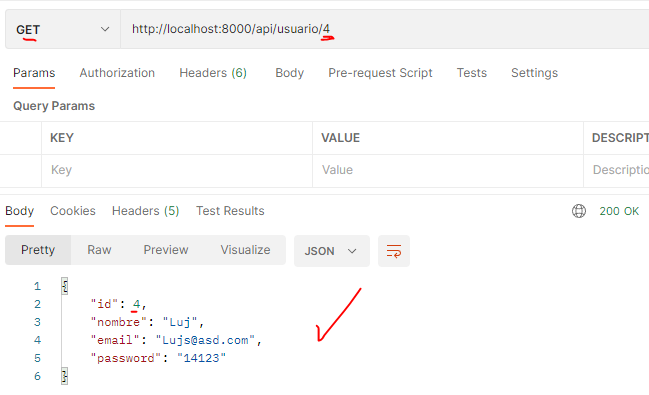
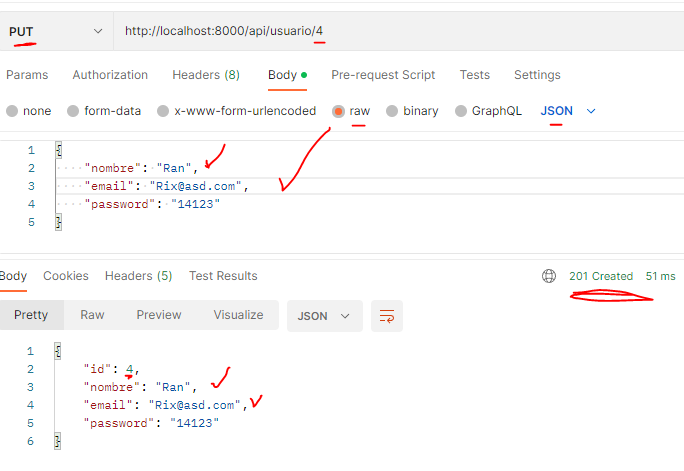
 Probando API Restful en la herramienta Postman

Por medio de postman realizo las peticios con los difrentes métodos al servicio

POST: Crear - GET: listar

GET: Listar por Id=4 – PUT: Actualizo el nombre y correo con id=4

DELETE: id=4

