**Prueba de ingreso ST&T**

**Presentado por: Jhon Alexis Romero Hernandez**

**Explicación del código creado**

**Creación de la base de datos usando MySQL**

Para este ejemplo se usó el motor de base de datos MySQL, creando la base de datos styt, con una tabla, cuyo nombre es cliente, para poder realizar las respectivas pruebas, código a continuación:

create database styt;

create table cliente (

id integer not null auto\_increment,

celular varchar(255),

ciudad varchar(255),

correo varchar(255),

nombre varchar(255),

telefono varchar(255),

primary key (id)

);

**Código creado**

Se editó el archivo de propiedades de la aplicación añadiendo parámetros como **dll-auto**: update, para que la tabla se cree a partir de la información encontrada en la entidad creada en la aplicación, **dialect**: org.hibernate.dialect.MySQL5Dialect, para que lea el lenguaje que usa MySQL, **driver-class-name**: com.mysql.cj.jdbc.Driver, para importar el Driver de conexión de MySQL, **url**: jdbc:mysql://localhost:3306/styt?zeroDateTimeBehavior=convertToNull&serverTimezone=UTC, **username**: root y **password**: 123456, datos usados para conectarse a la base de datos creada con anterioridad.

1. Métodos (REST): Recibe los parámetros necesarios para realizar un CRUD

* En el archivo pom.xml, se importaron las librerías spring-boot-starter-web, para servicios REST, mysql-connector-java, para conexión a base de datos, spring-boot-starter-data-jpa, para las funciones JPA que se usarán a lo largo del código para el CRUD realizado.
* Clase Cliente.java, se crean la entidad correspondiente a la tabla cliente, con sus parámetros, los definidos en la base de datos creada anteriormente y también sus respectivos getters and setters.
* Clase ClienteDAO.java, se crea la interfaz que contendrá los métodos por defecto de la librería JPA, y consultas sql realizadas a mano para comprobar el correcto funcionamiento.
* Clase ServiciosRest.java, se crea la funcionalidad de todos los métodos del CRUD, se devuelve un ResponseEntity en cada uno de los casos para que al momento de probarlo devuelva los valores correctamente, se crean los métodos de: getCliente, findById, findAdvancedCliente, findByCityAndGmailAdvanced, createCliente, deleteCliente y updateCliente, correspondientes al CRUD.

Los métodos devuelven, según la operación a realizar, una lista de objetos tipo cliente con la información de cliente o un objeto tipo cliente con la misma información y reciben como parámetros, información en el body al momento de probar con RequestBody o un PathVariable que recibe a través de la url.

1. Métodos (SOAP): Recibe los parámetros necesarios para realizar un CRUD

* En el archivo pom.xml, se importaron las librerías spring-boot-starter-web-services, para servicios REST, mysql-connector-java, para conexión a base de datos, spring-boot-starter-data-jpa, para las funciones JPA que se usarán a lo largo del código para el CRUD realizado, junto con wsdl4j.

Adicional se importó un plugin, jaxb2 plugin, para el correcto funcionamiento con el archivo xsd que se explicará posteriormente.

* Archivo ClienteSoap.xsd, se realiza el esquema de lo que se enviará a través de los métodos SOAP, la definición y estructura de dichos métodos para que puedan ser probados posteriores a su creación, así como crear la estructura de la tabla cliente.}
* Clase Cliente.java, se crean la entidad correspondiente a la tabla cliente, con sus parámetros, los definidos en la base de datos creada anteriormente y también sus respectivos getters and setters.
* Clase ClienteRepository.java, se crea la interfaz que tendrá la definición de los métodos CRUD usando CrudRepository, así como la definición de los métodos para obtener cada uno de los campos de la tabla cliente, nombre, ciudad, etc.
* Clase SoapServices.java, se definen todos los métodos para obtener cada uno de los campos de la tabla, así como definir las funciones básicas del CRUD que posteriormente será referenciada y creadas.
* Clase ClienteImplements.java, se implementan los métodos que se definieron en clases anteriores, cuál será su funcionalidad, usando las funciones básicas de la librería CrudRepository, realizando el respectivo CRUD, dichas funciones retornarán valores verdaderos, falsos, listas de objetos tipo cliente u objetos tipo cliente, todo esto con funciones básicas de la librería.
* Clase WebServiceConfig.java, se define de dónde se sacará la información para tomar en las respectivas pruebas, definir ubicación del archivo.xsd, es decir, la ubicación del archivo que tenga las funciones de web service definidas.
* Clase ClienteEndPoint.java, se implementan los métodos CRUD declarando que parte usarán, en este caso con localPart, que namespace usarán y que tipo de dato retornarán, lo que retornan dependerán del método a ejecutar, con la librería jaxb2 todo lo que se haya nombrado en el archivo ClienteSoap.xsd se importará de manera automática en archivos para poder ejecutar las funciones que estos contengan, basta con seguir la estructura y así poder ejecutar el CRUD.