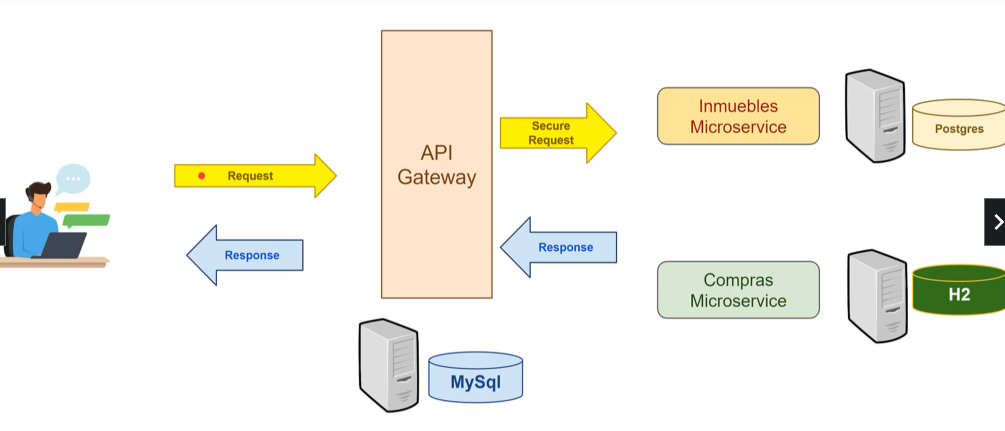
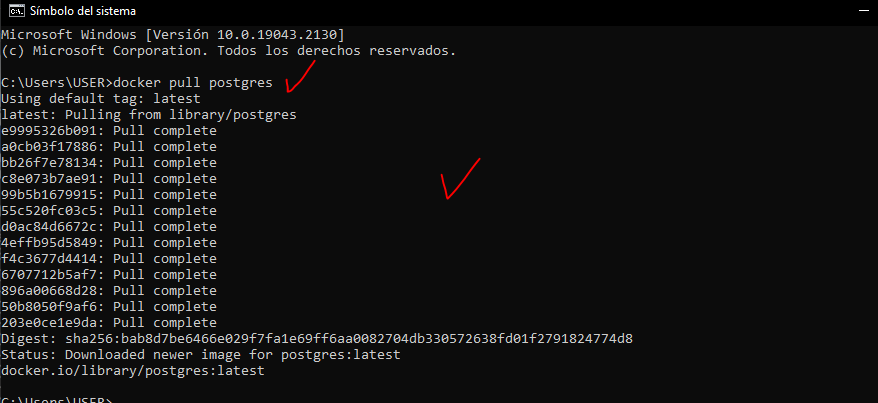
6. Microservicios con Spring Boot



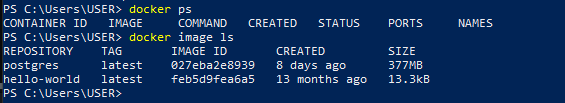
Creacion servidor Postgress

Vamos a crear el contenedor que contenga Postgres

<https://hub.docker.com/u/danielbel> busco postgres para copiar el comando para descargar la imagen



docker pull postgres, valido que se encuentre con image ls



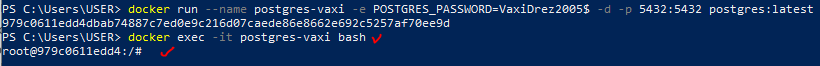
Procedo a crear el container, le doy el nombre y con una opción de posgrest asigno el password para acceder al container.

docker run --name postgres-vaxi -e POSTGRES\_PASSWORD=VaxiDrez2005$ -d -p 5432:5432 postgres:latest

Seguido me da el id del container

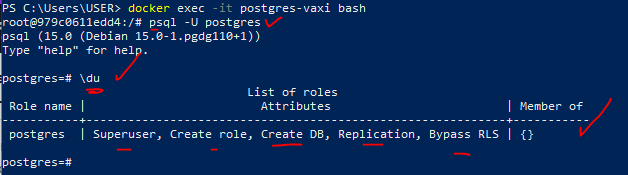
Procedo a ingresar al bash del container creado, si devuelve el numeral ya estamos adentro

docker exec -it postgres-vaxi bash

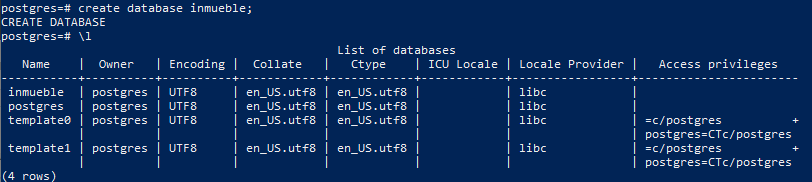


Y ya procedo a ingresar al ambiente de potgres (psql -U postgres). Para este caso -U habilita el usuario que ingresara al DB

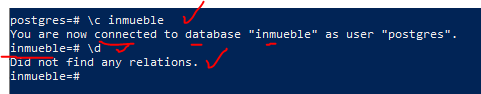
Y para ver el rol del usuario seria con el comando \du



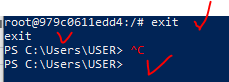
Y procedo a crear una database inmueble create database inmueble; y corroboro que las DB de mi container. \l



Conectándome a la DB de inmueble \c inmueble y procedo a ver si tengo tablas(relations) en la DB \d



Luego procedo a salir y apagar el container.



 Creacion de Microservice

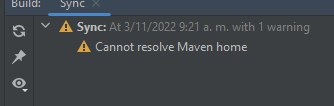
<https://start.spring.io/>



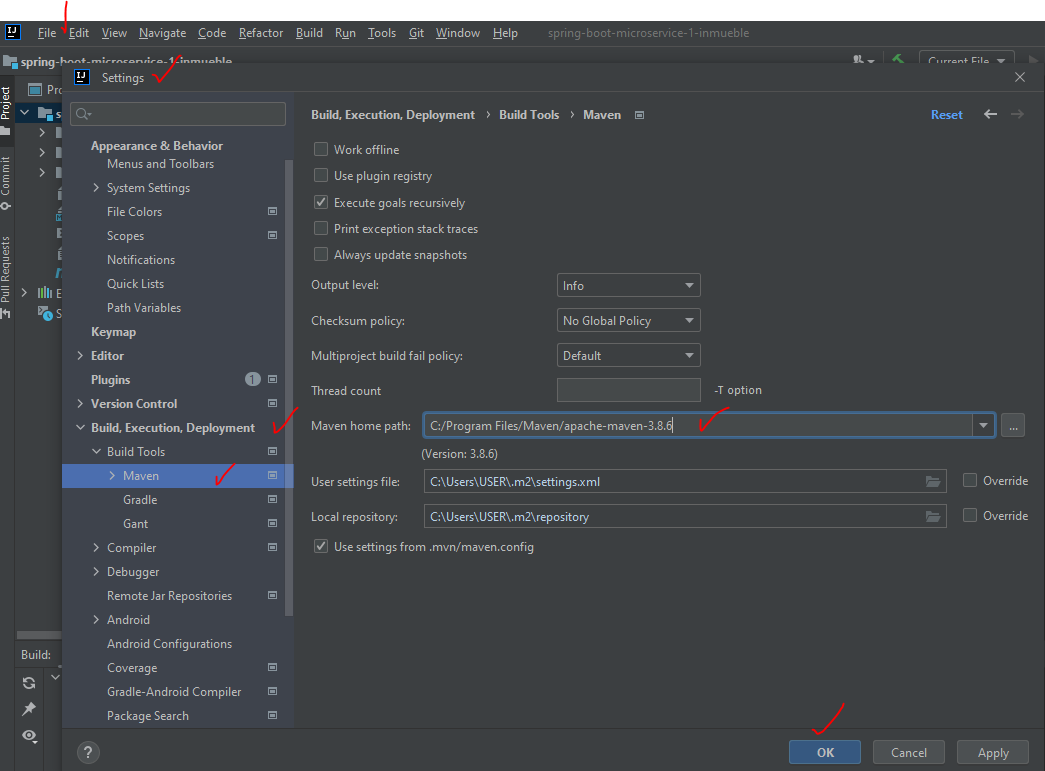
1. Tipo de proyecto
2. Seguridad al proyecto
3. Almacenamiento y DB
4. Métodos DB
5. Driver de conexión para postgre
6. Conexión entre los servicios
7. Código Limpio

El zip lo agrrego y extraigo en la carpeta del proyecto y procedp abrir intellij para ejecutar el proyecto.

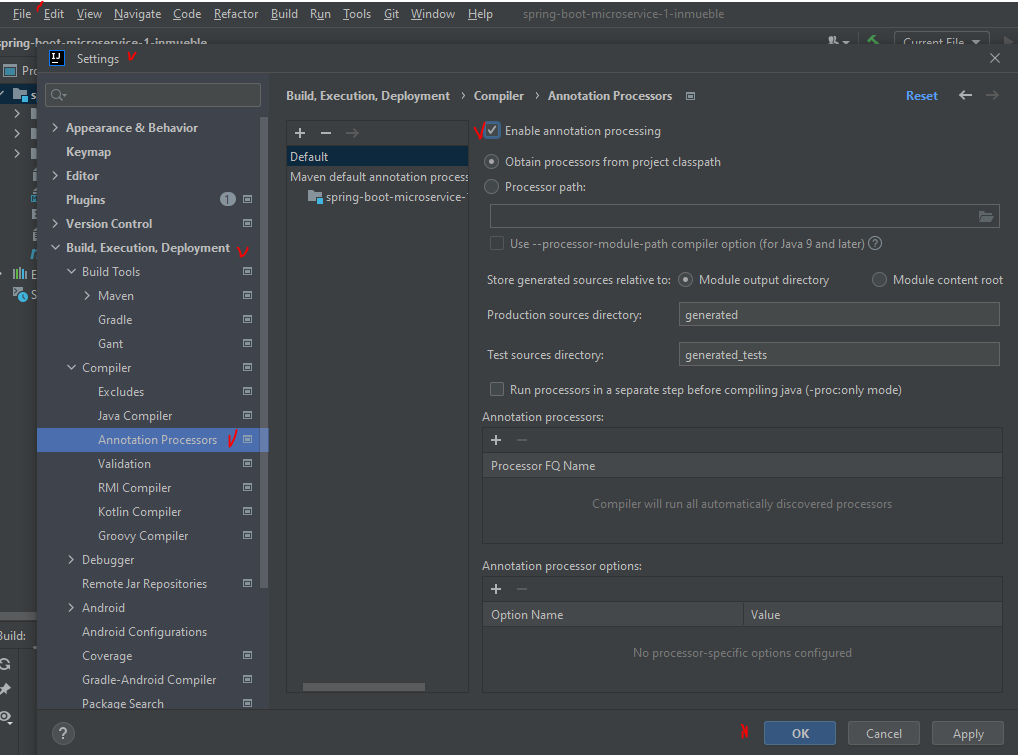
Procedo abrir el proyecto en intellij y espero a que descargue las dependencias, procedo a solucionar el error Maven



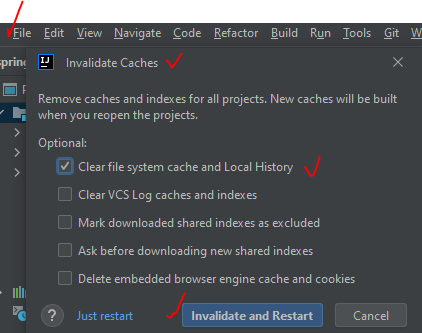
Selecciono la ruta donde instale el MAVEN OK



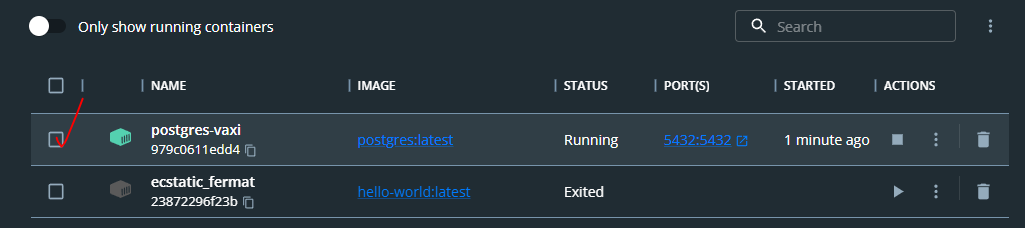
Procedo a instalar las anotaciones



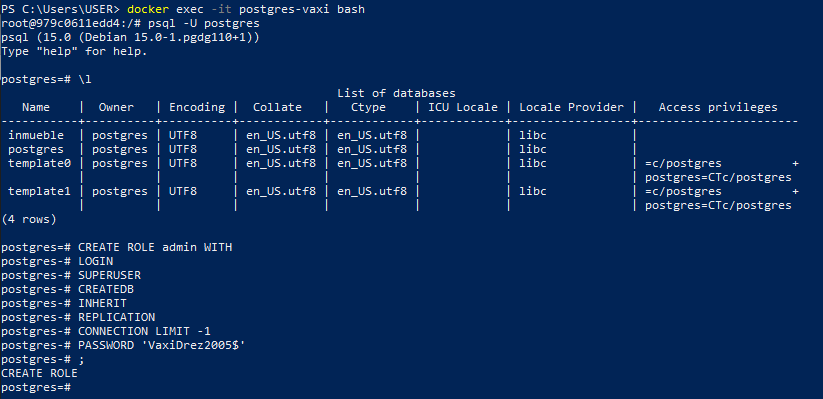
Habilito limpiar cache y vuelve a iniciar intellij con todos los ajustes



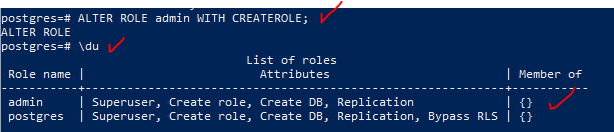
Creacion de Base de Datos y Schema

Creando una nueva base de datos con un rol diferente, ojo el container creado de postgres se debe estar ejecutando

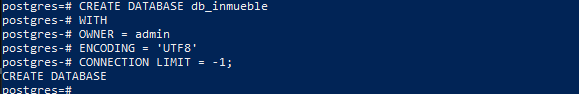
Ingreso al bash del container, y elijo el usuario de administrador postgres; Luego se creara un rol que permita hacer el login y conexión con el aplicativo .net java pyton (mas adelante), usuario tendrá los siguiente permisos, al finalizar todas lasw condiciones para el rol termino con un ; y verifico que el rol haya sido creado, el conection limit significa que tengo conexiones ilimitadas a la DB



Me falto agregar un atributo al rol, procedo a relizar un alter y validar que todos los atibutos se encuentren en el rol

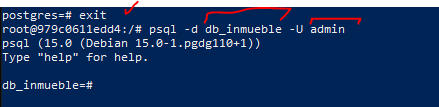


Procedo a crear la base de datos con el user admin

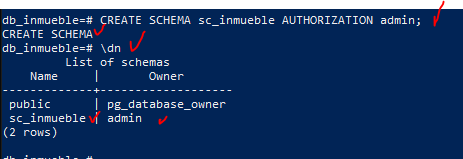


Recordar que las DB pueden trabajar con diferentes esquemas y que un usuario puede acceder a varios esquemas según los permisos.

Como vemos salimos del usuario POSTGRES y entramos a la DB de DB\_INMUEBLE que es del usuario ADMIN



Procedo a crear un esquema en la DB de inmueble y listo los esquemas para validar que si se creo OK



 Configuracion de Microservice

Configurando el archivo app.properties

Creacion entidad JPA

Creando clase entidad DOMAIN para describir un inmueble

Esta clase sera un entity persistente para jpa entity o hibernate

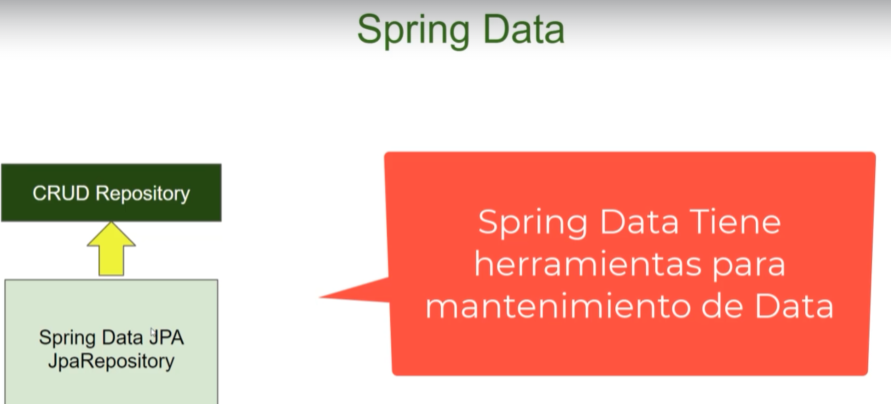
El @Id me genera un auto id pero como bien sabemos este aid también lo genera la DB en automatico y para ello debemos usar la anotación @GeneratedValue(strategy = GenerationType.IDENTITY)

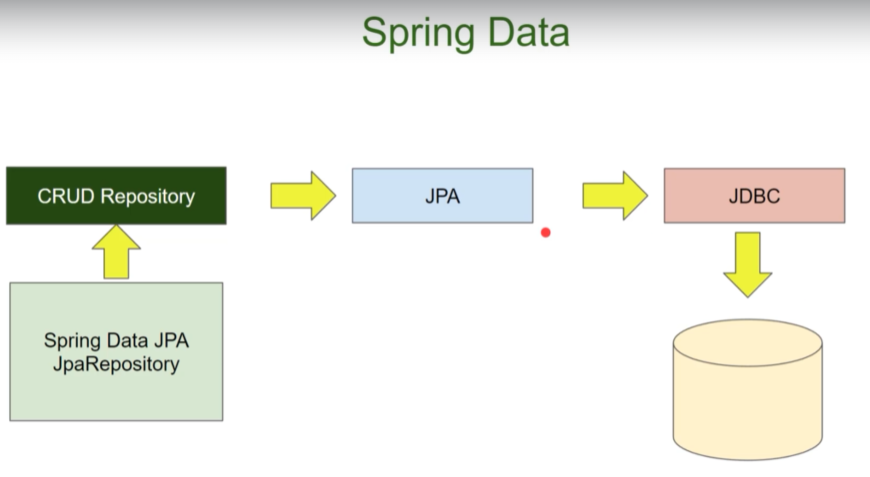
A las siguientes atributos de la tabla le agrego las condiciones siempre con la anomtacion @Column

De igual manera como son atributos provados procedo a generar su métodos seter y getter por medio de la anotación @Data que procede de la dependencia lombok



 Crear Repository con Spring Data

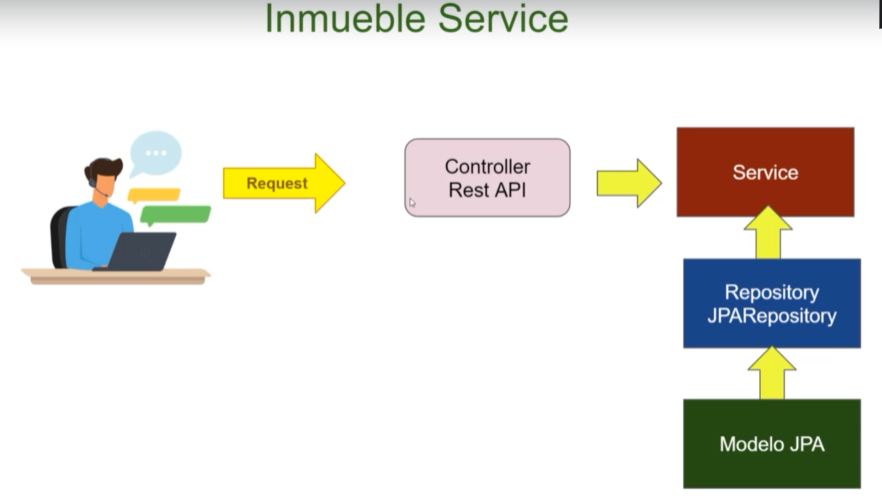




Creamos un nuvo parquete Repositori y la interface que contendrá los métodos para el repositorio CRUD -- ClaseDao

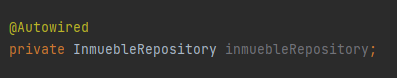


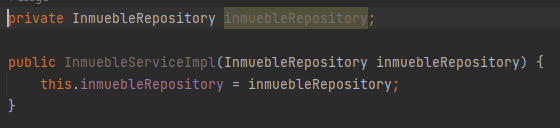
Crear Service



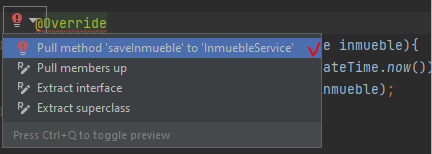
Crearemos la capa de servicio para ello el paquete servicio, la interface y la implementación del servicio inMueble ClaseWeb . Recordar que la implementación debe hacer ajustes a nivel de DB

Recordar que la anotación @AUTOWIRED es el que hace la independencia es decir inyecta a un constructor de la clase service con el parámetro de entrada InmuebleRepository

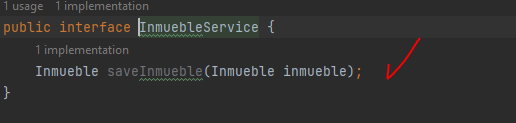
 es lo mismo que

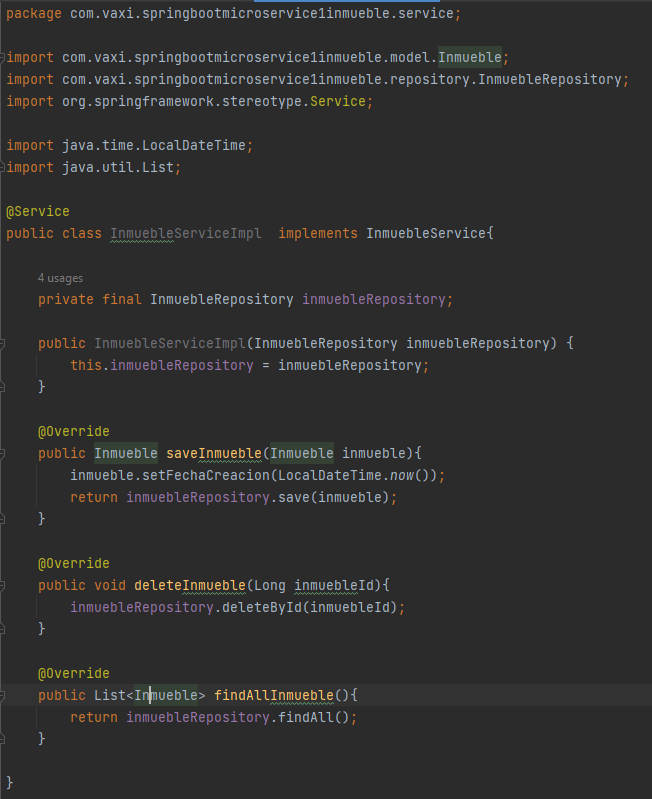


Siemrpe que declare un método en la logica de negocio lo debo hacer Con la anotación @Override de ese modo agrego el método también a la interface de negocio



En la interface de negocio se agrea en automatico al darle el pull





 Creacion de Controller

Crearemos la clase controller, por ender tendrá la anotación @RestController, esta clase tendrá acceso a los diferentes tipos de MAPPIN(Get post, put, delete….) y le agrego el path para acceder a este controlador.

Este rest recibe el request de un cliente con los datos según el formato Json, Xml, Wsdl…

Creo el método ResponseEntity maneja toda la respuesta HTTP incluyendo el cuerpo, cabecera y códigos de estado permitiéndonos total libertad de configurar la respuesta que queremos que se envié desde nuestros endpoints.

Cuando agrego el método response debo agregar el tipo de atributo HTTP para este caso como maneja data sera de tipo @PostMapping

