PostgreSQL y Spring

Referencias:

Todo el código fue sacado de:

<https://github.com/jokumar/postgress-boot>

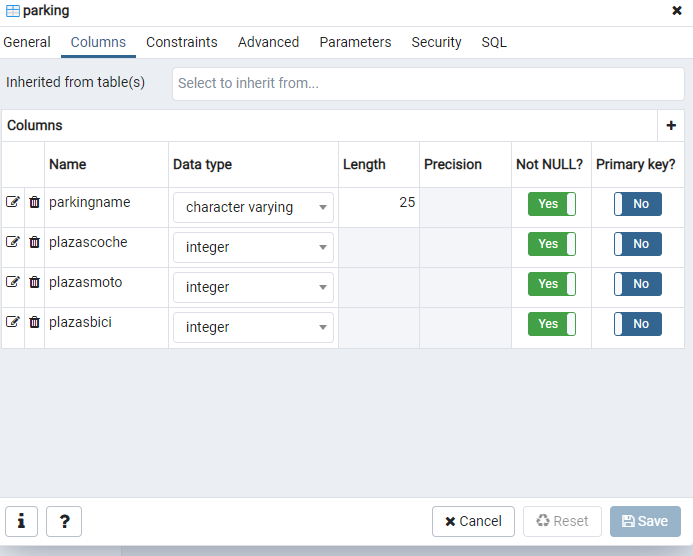
Y la explicación del código está en:

<https://dzone.com/articles/bounty-spring-boot-and-postgresql-database>

He cambiado algunas cosas, donde él ponía @Resource yo puse @Autowired, no sé hasta qué punto cambia algo, pero a mí me funciona así.

He cambiado todos los nombres de Employee a Parking y algunos métodos he tenido que cambiar el nombre de los parámetros porque en la base de datos las columnas tienen otros nombres.

**Opcional:** en la base de datos hay que crear una tabla, y saber bien los nombres de cada columna. Yo como solo quería probar que funcionara, creé una tabla llamada parking con 4 columnas: parkingName, plazasCoche, plazasMoto, plazasBici. Se puede hacer con comandos o interfaz gráfica (yo utilicé [pg admin4](https://www.pgadmin.org/download/pgadmin-4-windows/)).



Contenido

[La configuración y clases añadidas para que funcione: 3](#_Toc26870682)

[En el pom.xml 3](#_Toc26870683)

[En application.properties 6](#_Toc26870684)

[Código java: 8](#_Toc26870685)

# La configuración y clases añadidas para que funcione:

## En el pom.xml

En teoría ya están todas las dependencias necesarias, de todas formas, las dependencias a añadir son:

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

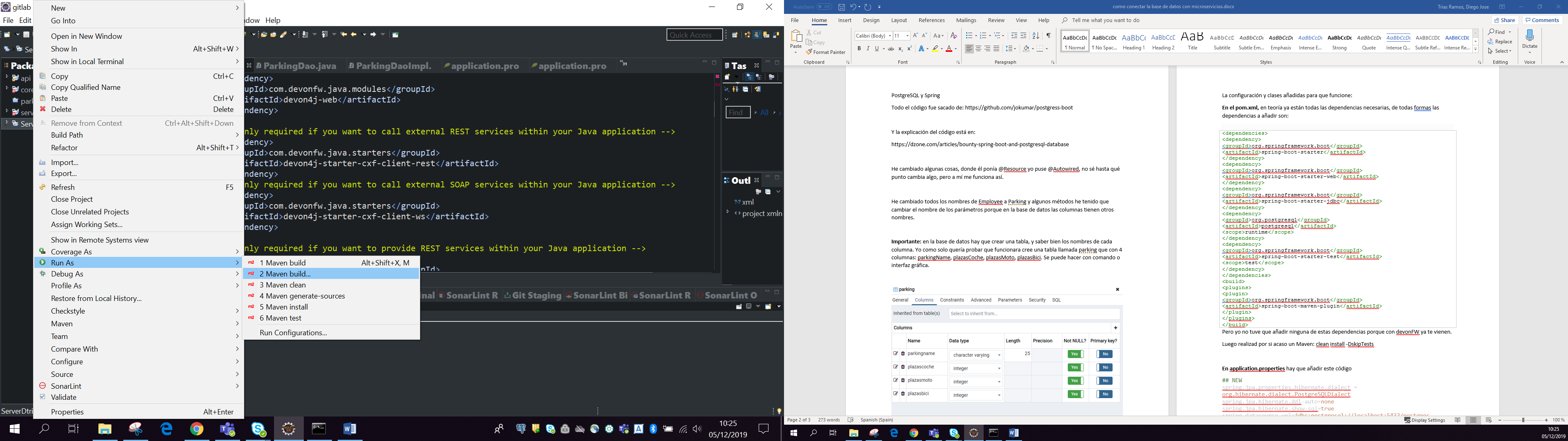
</plugin>

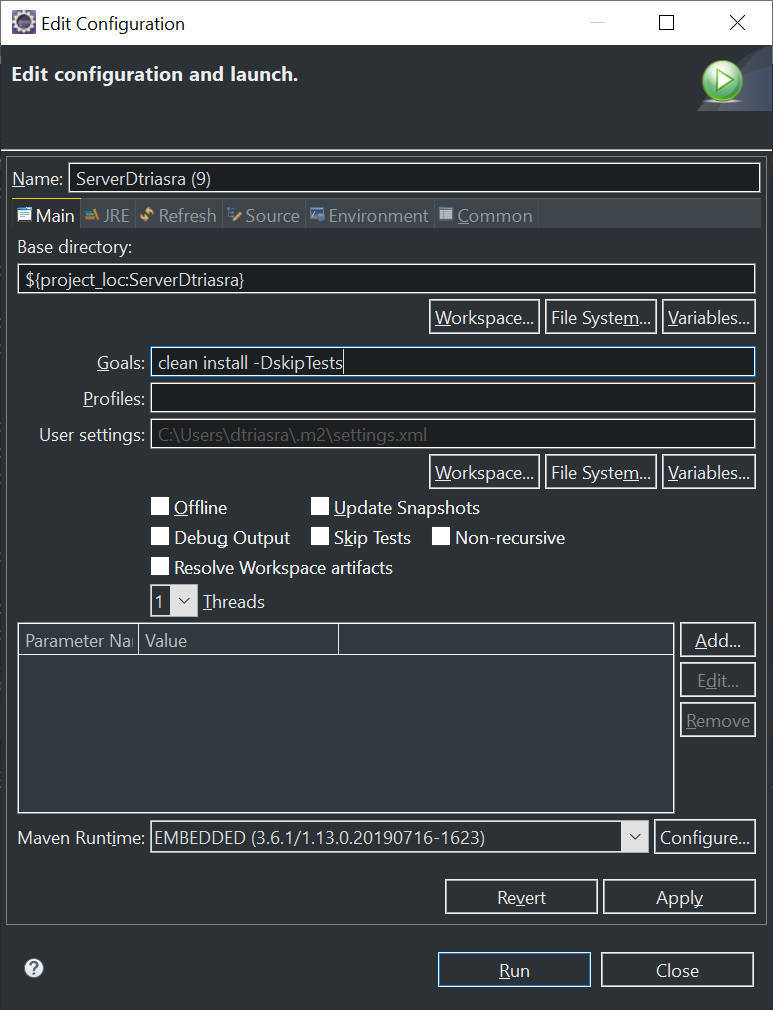
</plugins>

</build>

Pero yo no tuve que añadir ninguna de estas dependencias porque con devonFW ya te vienen.

**IMPORTANTE**: Luego realizad en la carpeta de vuestro proyecto (ServerDtriasra en mi caso) un, Maven: clean install -DskipTests





## En application.properties

Hay que añadir este código (los #new son comentarios, yo los pegaría también para saber qué es lo nuevo que has añadido):

|  |
| --- |
| ## NEW  spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect  spring.jpa.hibernate.ddl-auto=none  spring.jpa.hibernate.show-sql=true  spring.datasource.url=jdbc:postgresql://localhost:5432/postgres  spring.datasource.initialization-mode=always  spring.datasource.initialize=true  spring.datasource.schema=classpath:/schema.sql  spring.datasource.continue-on-error=true  ##NEW |

Es posible que salga algún error al añadirlo, si es así, simplemente hay que borrar las líneas de código que ya estaban para que deje de haber conflicto, en concreto, creo que spring.jpa.hibernate.ddl-auto=none da error porque ya hay una línea similar escrita antes, borrad las líneas viejas.

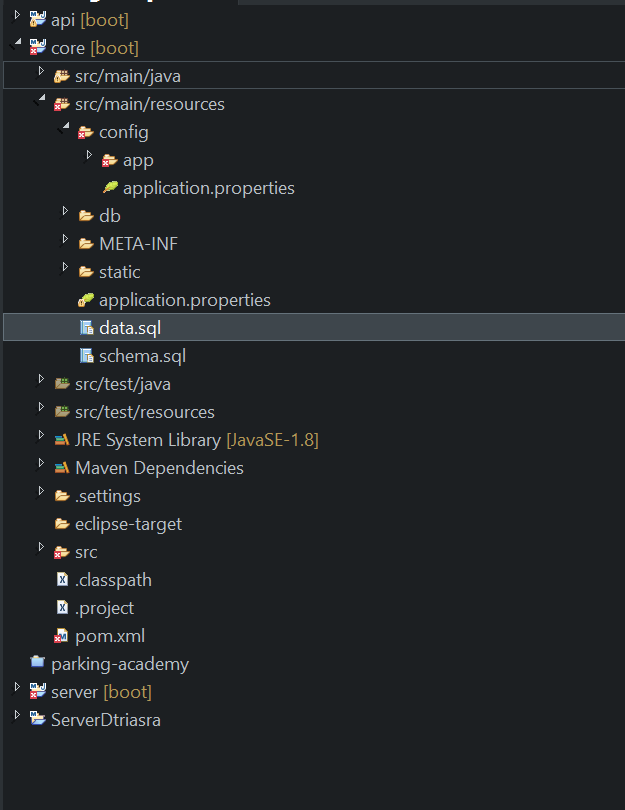
**IMPORTANTE:** Hay que añadir un archivo schema.sql a la carpeta resources del core, este schema.sql no puede estar vacío y nos servirá para crear la tabla en caso de que queramos crearla con comandos sql. Mi schema.sql tiene este código:

|  |
| --- |
| CREATE TABLE parking  (  parkingName varchar(25) NOT NULL,  plazasCoche int NOT NULL ,  plazasMoto int NOT NULL,  plazasBici int NOT NULL  ); |

También podéis poner un archivo llamado data.sql con el cual podeis añadir cosas a la tabla cada vez que se inicialice el programa. Mi data.sql se ve así: (Este archivo es opcional)

|  |
| --- |
| insert into parking(parkingName, plazasCoche , plazasMoto, plazasBici)  values('primero', 10, 12, 10); |

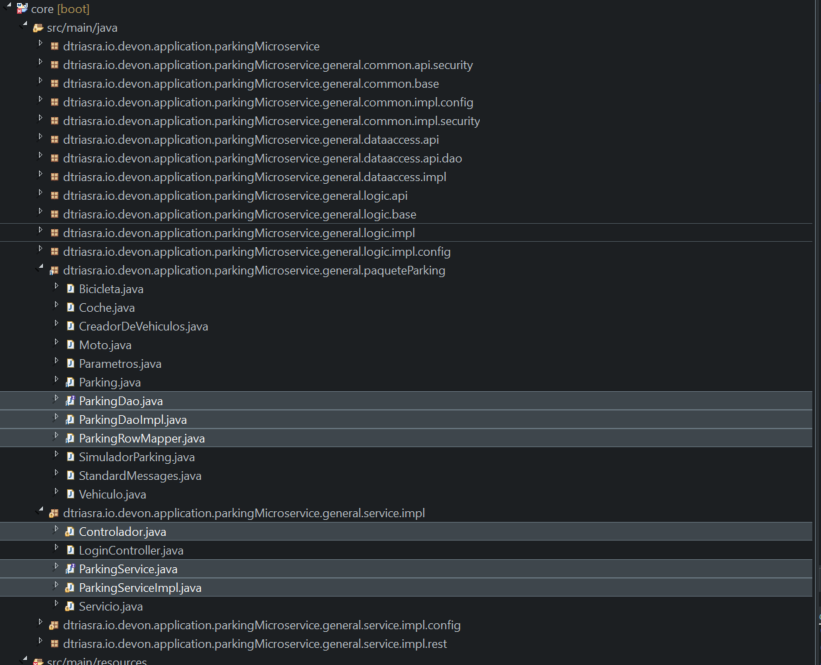
Schema.sql y data.sql van en los recursos de la carpeta core, concretamente en core/src/main/resourcesà



**Llegados este punto al correr la aplicación como spring boot app os debería funcionar, si no os funciona algo va mal xD. Probad haciendo el Maven clean install -DskipTests y asegurando que schema.sql está en el sitio correcto y que no está vacío**

## Código java:

Necesitas una clase DAO, una clase para mapear tu objeto Parking, un controlador y un servicio (con interfaces para que quede más bonito):



ParkingDao.java es una interfaz

ParkingDaoImpl.java es dónde realmente está escrito el código sql y los métodos a llamar para las distintas operaciones

ParkingRowMapper.java es la clase que se ocupa de crear objetos Parking a partir de las columnas en una fila en la base de datos

Controlador.java para los GET , POST, PUT…

ParkingService.java es una interfaz

ParkinServiceImpl.java es un servicio que llama a los métodos de ParkingDao.

Todavía no he implementado el método delete.