### FLOW DIAGRAM FOR CRUD OPERATIONS USING POSTGRES DATABASE

# **GOAL**

Perform Crud operations using Postgres database (External Database)

### **TECH STACK**

IntelliJ IDEA
Spring Boot 2.2.7
Maven 3.0+
PostgreSQL

Dependencies : Web, Jpa,Postgres Postman App

# Generated Zip file through start.spring.io. Version 2.2.7 for spring boot, Build tool as Maven, 3 dependencies: Web,Postgres and Jpa. Download the Postgres from its official website and install it on your system.. Imported the file in IntelliJ Created the main file named CurdApplication.java and an Error handling file. Created 4 packages: controller, object, repository, service

The Object package consists of a Product.java file that is used for creating entities and setting their values.

The Controller package consists of a ProductController.java file that is used for controlling all the requests using @GetMapping, @PostMapping, @DeleteMapping and @PutMapping annotations. With the @GetMapping we can get the details of all products or we can also access it through productID. Similarly with @PostMapping, @PutMapping and @DeleteMapping we can insert,update and delete the records respectively.

The repository package consists of a ProductRepository.java file that extends the interface CrudRepository.

The Service package consists of a ProductService.java file that is used for handling and providing response to the controller file.

Then I have edited the application.properties file. As it is needed to connect the program through Postgres, I have added certain properties. Postgres works on localhost:5432 by default.

Also, there is a data.sql file in resources that consists of insert queries that inserts the data into our database. And there is one schema.sql file that creates a new table and deletes the previous one if it exists.

Start the pgAdmin, it helps in maintaining the schema and database. We can also perform various queries with the help of pgAdmin. After running the program, when we visited the localhost:8080, we received the output as "Home". And localhost:8080/products gave us the list of all products. To test other operations, I have used the postman app. Here, I have used GET, POST, PUT, DELETE methods to read, create, update and delete the gueries respectively. All the functions worked properly and hence the goal is achieved. **END** 

## Some points that will help in removing roadblocks:

- Main java file should be above all the files.
- If the program throws some exception, check your pom.xml file, it should access the correct package.
- Control of the program must go in a sequence from one package to another.
- Error Handling program must be there.