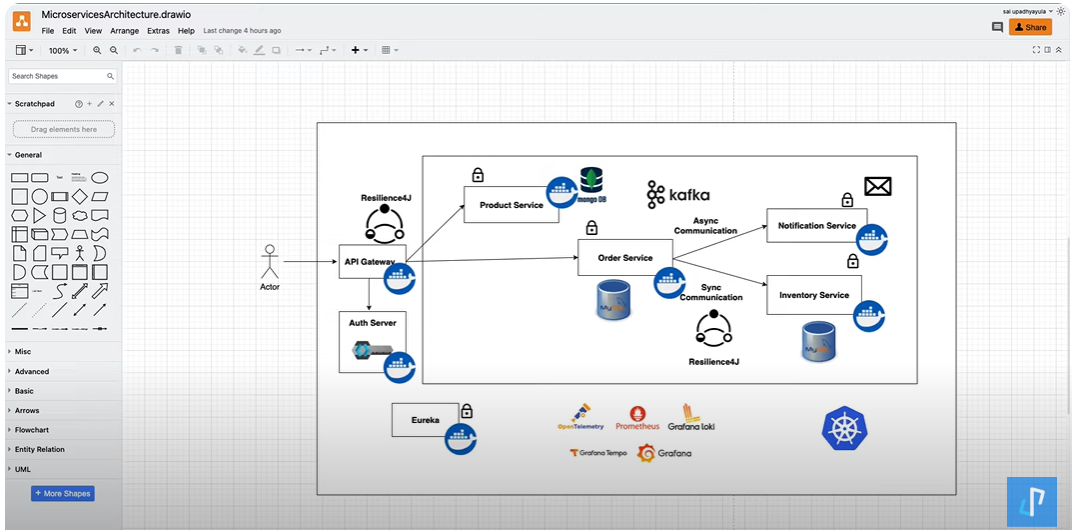
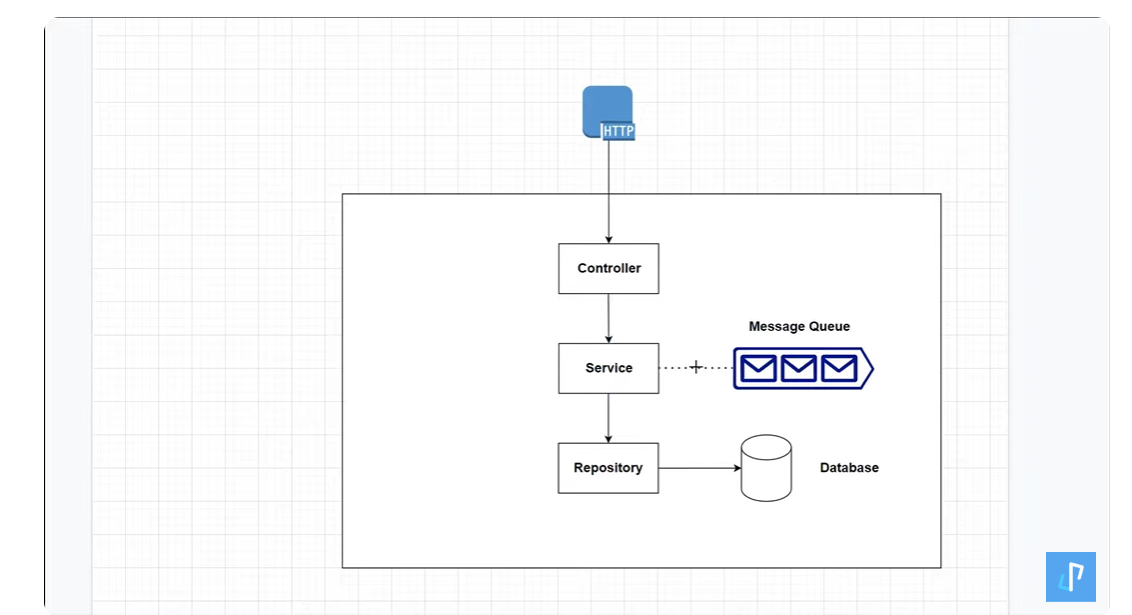
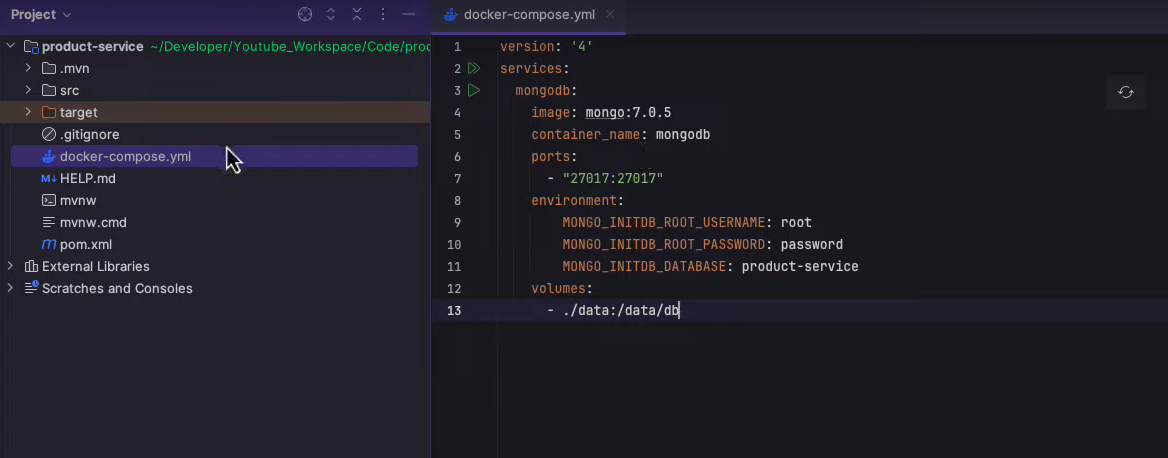
Microservices:



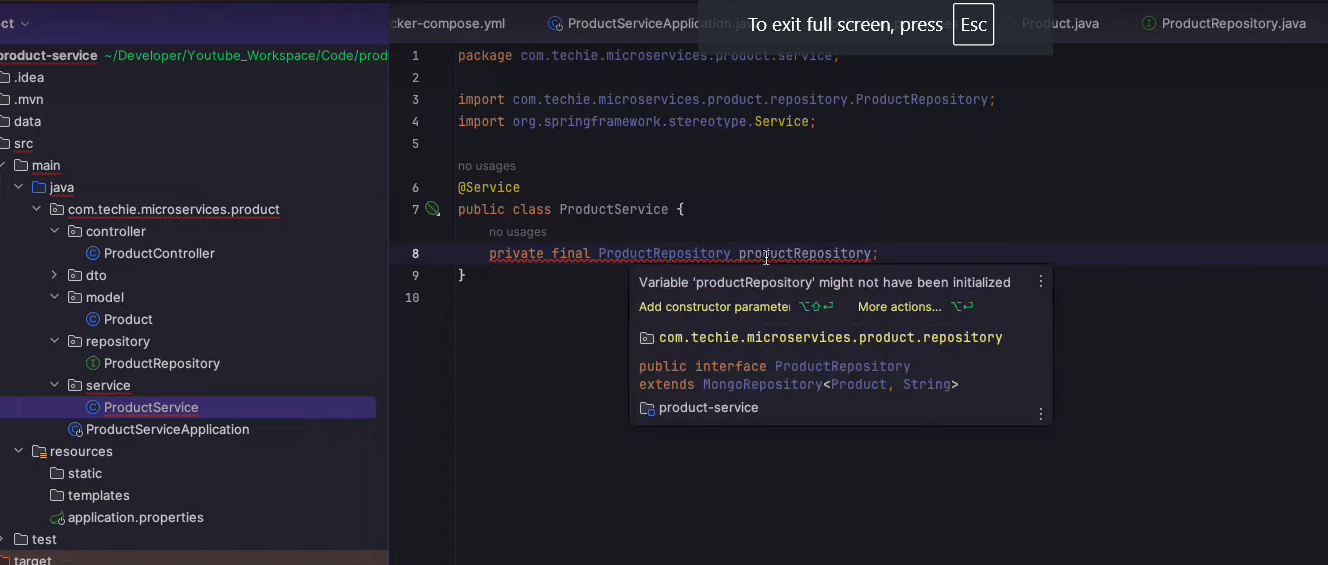
API gateway acts like an gatekeeper, which sends out the request from the user to the inner service, for which it wants to interact with.



As MongoDB is running inside a container, so we stop the container, all the data that will be inside the container will be lost, so we need to provide the volume folder where we need to store the data inside this part in our file system.



When compiler complains of adding the constructor for injecting the dependencies, we can inject that using the Lambok using the @RequiredArgsConstructor



In Java 21, we need to make a request class where we pass the full object:

example for class:

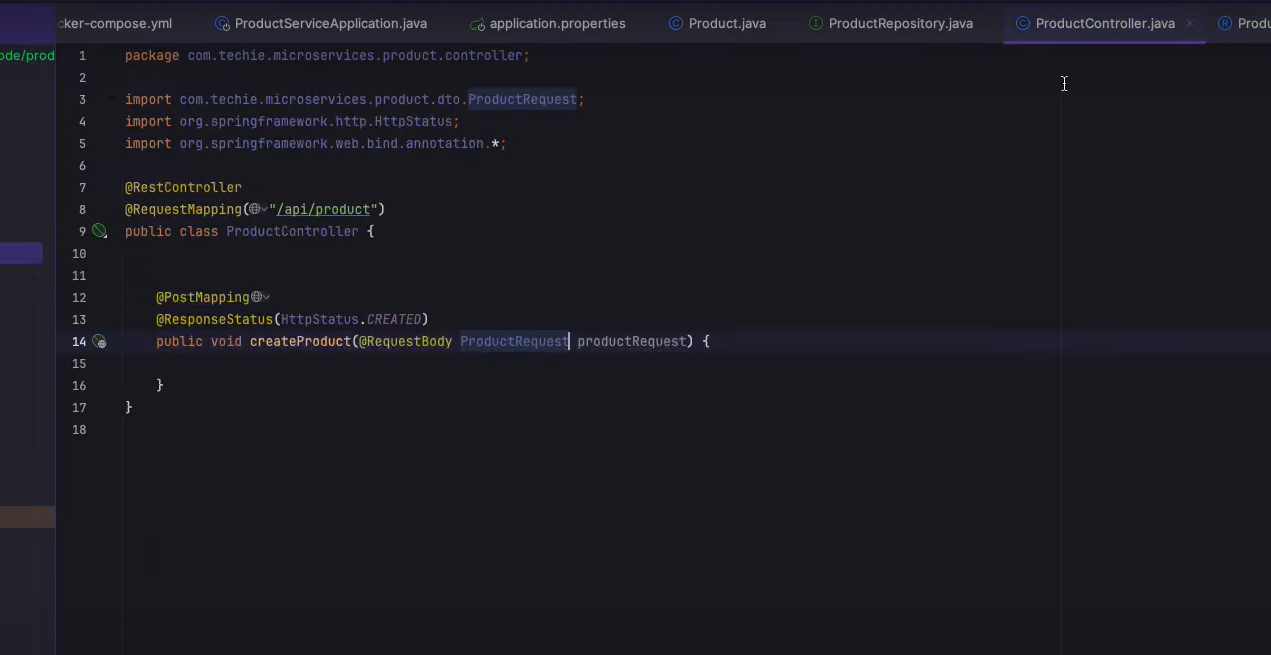
Product{id, name, description}

we need to use the ProductRequest class as follows:

public record ProductRequest(String id, String name, String description){}

And to access the same we need to write the

productRequest.name() instead of productRequest.getName();



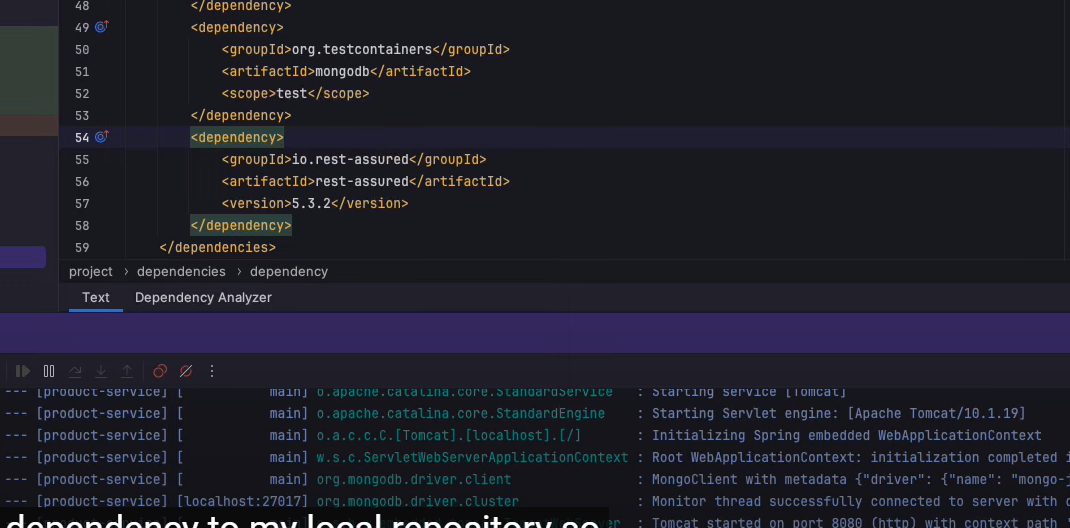
A screenshot of a computer

AI-generated content may be incorrect.

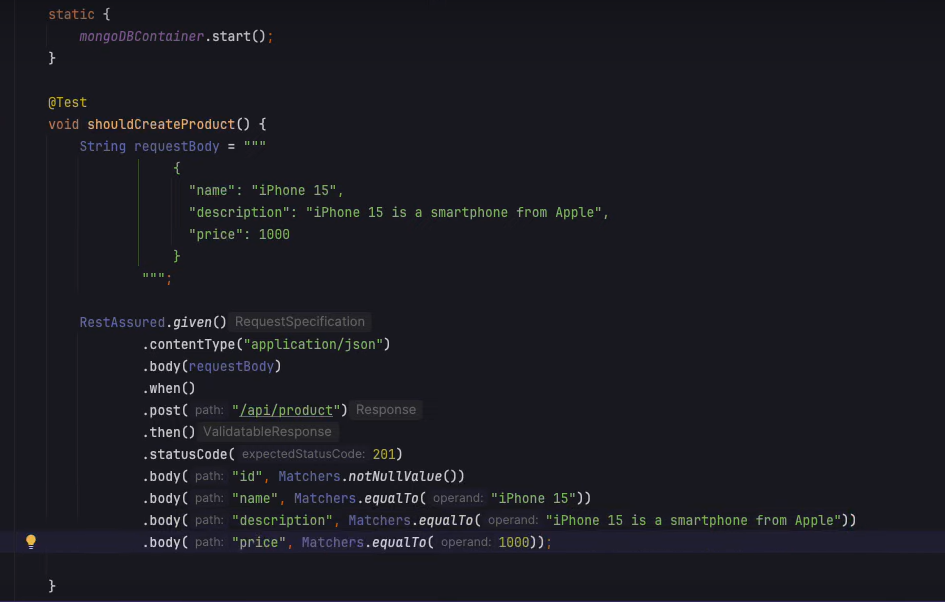
A screenshot of a computer screen

AI-generated content may be incorrect.

Io.rest-assured dependency to make use of the REST API calls in the TEST environment.



To check the MongoDB test:



A screenshot of a computer

AI-generated content may be incorrect.

SQL Docker:

A screenshot of a computer

AI-generated content may be incorrect.

SQL Configuration:

A screen shot of a computer

AI-generated content may be incorrect.

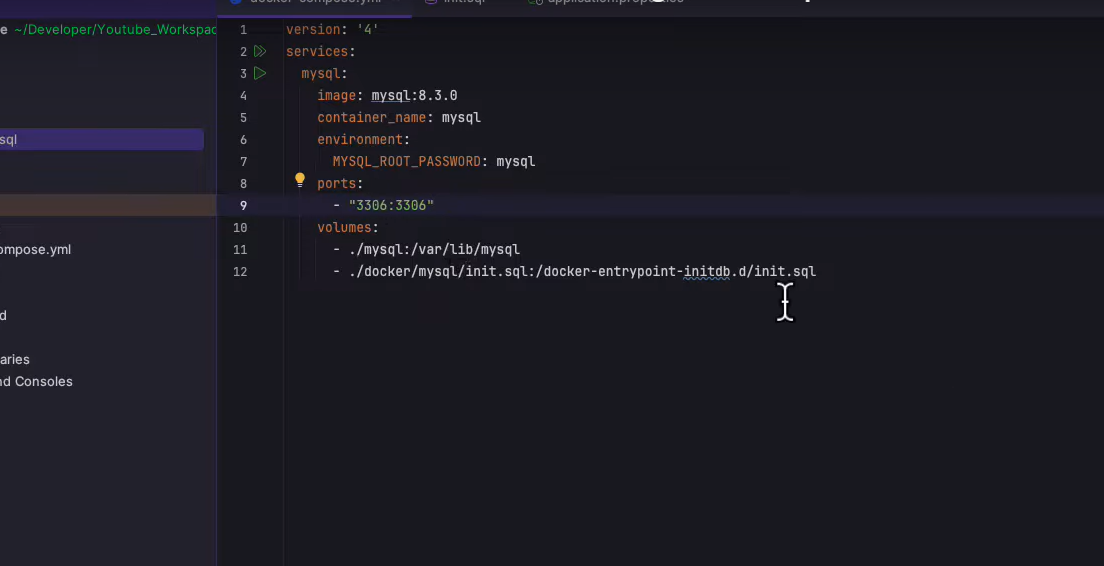
FOR THE ABOVE CONFIGURATION, if the table does not exist, we need to create using either mysql or using the docker file as following:

Creating a folder structure as: docker -> mysql -> init.sql

A screenshot of a computer

AI-generated content may be incorrect.

And after that, add the same in the docker-compose.yml file:



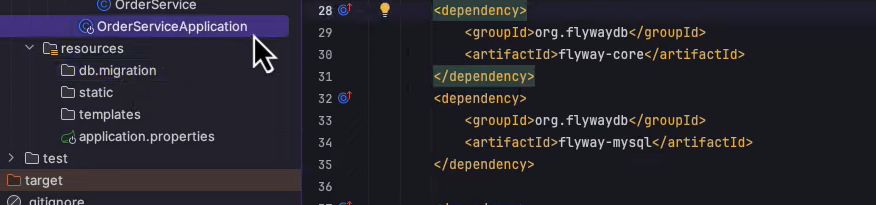
Spring.jpa.hiberate.ddl.auto=none, we can use this as none, but cannot use this as create in production.

So, we use the FLYWAY MIGRATION, which allows us to version control the database.

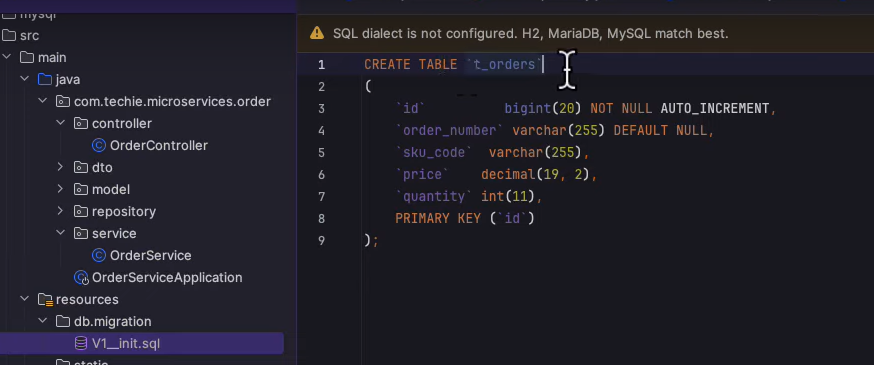
So, if we want to add a new field in database, so we add that in java and simply commit that code, but to roll back that is a tough option.

So, with Flyway, we have the option to rollback to the previous versions and also, write own SQL queries to update the database.

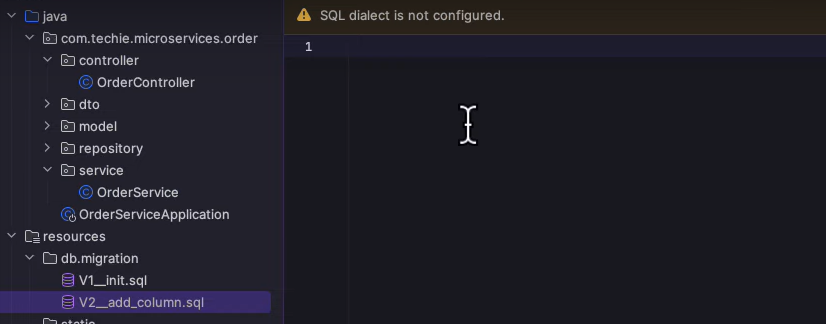
The scripts are under db.migration folder and dependency is mentioned in pom.xml.



So, we need to run the sql using the script placed in the V1\_\_init.sql and script is as follows, which allows us to create a new table in the mysql:



And now if we need to update some column in the database, we need to make the following file and then we need to write our logic in that file:



So when the tables are getting created, there is one more table that is getting created for the flyway\_schema\_history, which shows which script has been created for the flyway\_schema:

A screenshot of a computer

AI-generated content may be incorrect.

With the help of SPRING CLOUD OPENFEIGN, we need to use the inter process communication or the Synchronous communication, between the services:

A diagram of a process

AI-generated content may be incorrect.

HOW MONGODB interacts with spring boot using Docker:

We need to run Spring boot in one container and MongoDB in another container and then we need to check how they communicate between them using the **docker-compose file or we need to use Docker-Link**

The Docker Compose is a tool for running multi container docker application, we just need to make the configuration file, rest would be handled by the DOCKER itself.

A diagram of a blue whale with a blue container and green logo

AI-generated content may be incorrect.

We need to run both containers using the command prompt of docker using docker link command or we need to use docker-compose.yml to provide the configurations and can simply run the application.