Practice

Create a Single-entry Point to Route the Request Coming for Different Microservices Using Spring Cloud

API Gateway

Clients

/user

/register

/login

The clients send request for a service.

A P I G a t e w a y

The API Gateway routes the request to the service requested.

User Service

Product Service

Exercise

- **Practice 1 : Shopping Application**

Points to Remember

- Be sure to add the properties below in the `application.yml` file, or else the web dependencies will not work.

```yaml
main:
  web-application-type: reactive
```

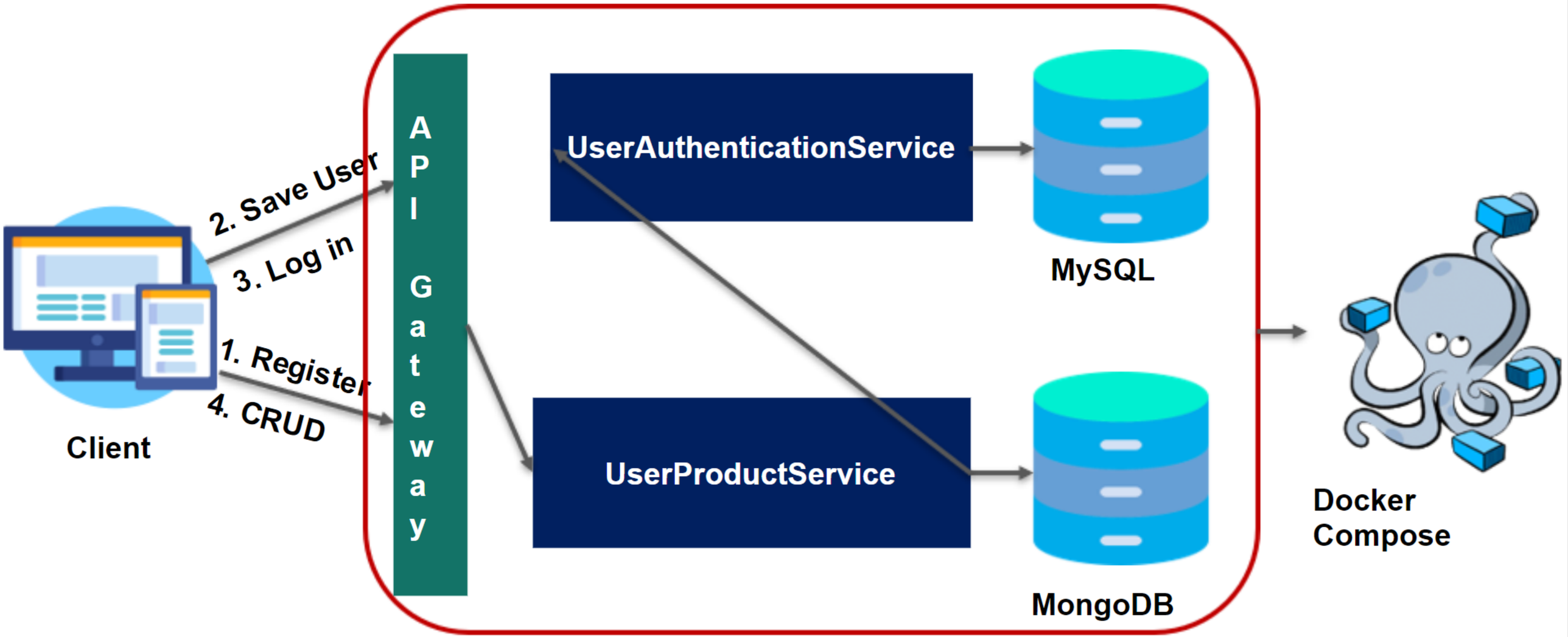
PRACTICE

Shopping Application

Consider a shopping application that enables users to shop for products on any smart device. It provides multiple features for all its registered users. The user needs to register with the application to access some of its features. Let us create multiple microservices for the application.

1. The user must first register with the application.
2. The user must log in with credentials such as id and password.
3. The user can access the features such as adding and deleting products, etc.
4. Implement Spring Cloud Gateway.

How Does the Application Work?



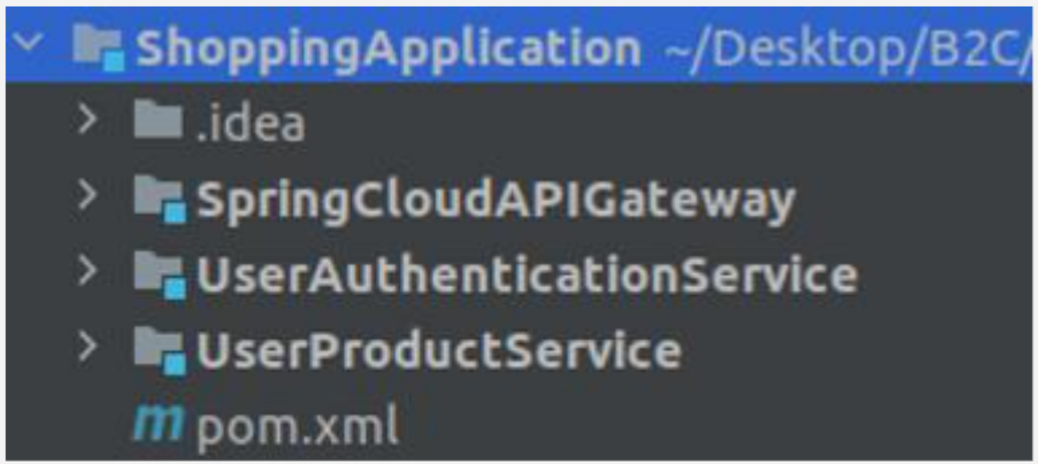
Instructions for the Practice

- Click on the boilerplate.
- Fork the boilerplate using the fork button
- Select your namespace to fork the project.
- Clone the project into your local system.
- Open the project in the IntelliJ IDE.

Task: Practice 1

- Create a new project using the Spring Initializer to build the Spring Cloud API Gateway.
- Add the dependencies for Spring Cloud Gateway in the pom.xml.
- Add the Spring Cloud API Gateway in the module of the parent pom.
- Configure the routes in the API Gateway.
- All services must be routed through the API Gateway.
- The API gateway must send the request to the corresponding service.
- Test the output in Postman.

Task: Practice 1



The structure of the Application

- `SpringCloudAPIGateway` is the service that will act as API Gateway.
- `UserAuthenticationService` is the service that will have the login and save the user functionalities.
- `UserProductService` is the service that will have all the CRUD functionalities related to the product.
- `pom.xml` is the parent pom.

Submission Instructions

- Before pushing the solution to the repository,
  - In the `application.properties` file there are two configurations to execute the application, one for local execution and other for Hobbes execution.
  - When executing the application on your local machine, comment the hobbes configuration and uncomment the local configuration and change username and password to connect to database as per your local config.
  - Before pushing the solution to the repository comment the local configuration and uncomment the hobbes configuration.
- Push the solution to git.

Submission Instructions (contd..)

- Submit the practice or challenge on hobbes.
- Login to hobbes using your credentials.
- Click on **Submission** in the left navigation bar.
- The **Submit for evaluation** page is opened.
- Select the solution repository `bej-spring-cloud-api-gateway-c4-s3-pc-1-shopping-application` against which your submission will be evaluated, under **Assignment Repository**
- Select your solution repository `s3-pc-1-shopping-application` under **Search Submission Repo**
- Click on **Submit**.
- The results can be viewed in the **Past Submissions** screen.