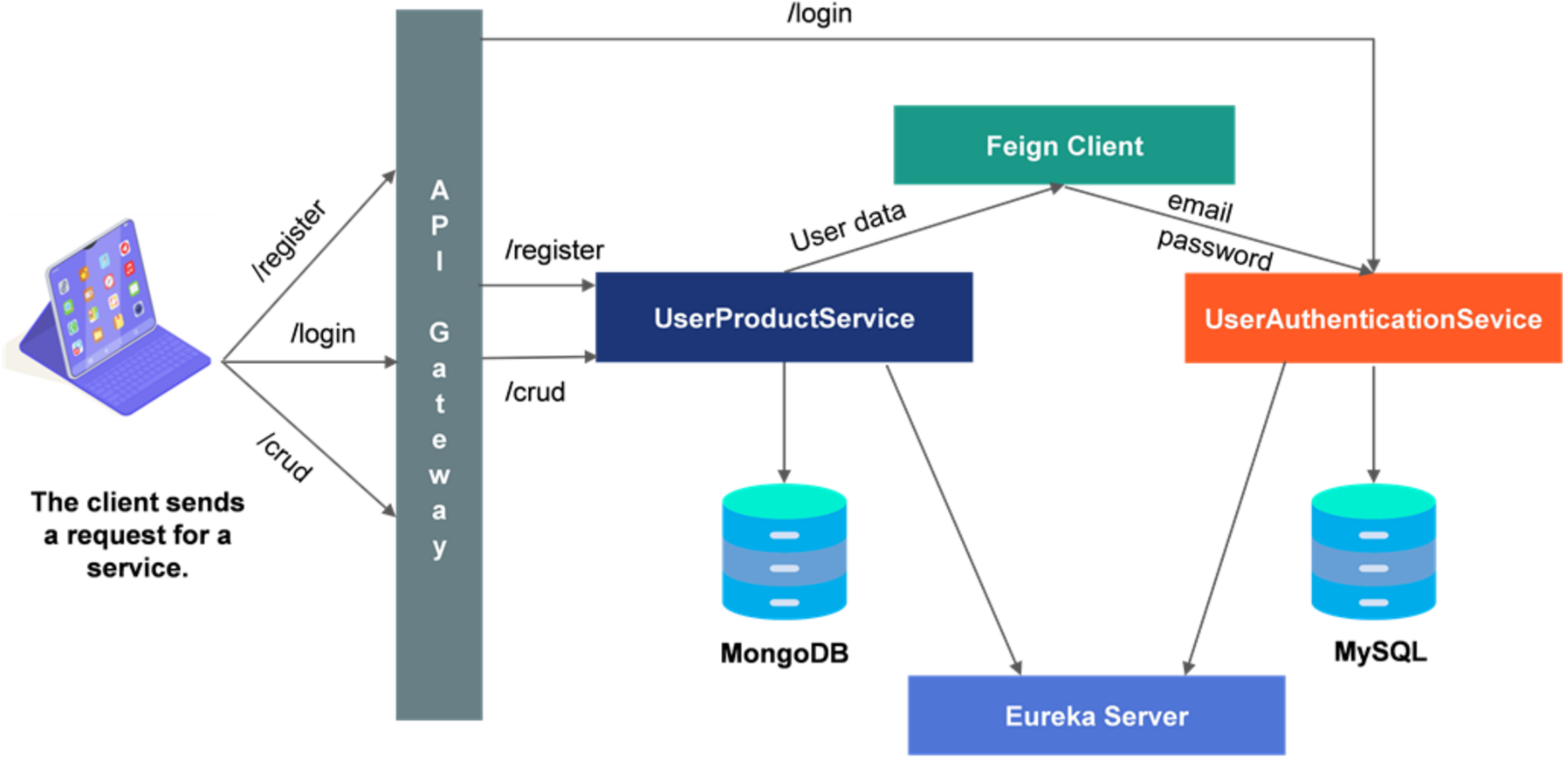
Practice

Establish Synchronous Communication Among Microservices by Using Feign Client

Communication Between Microservices



Points to Remember

- The proxy interface must be created in the microservice that wishes to establish communication with another microservice.
- The method in the proxy interface must have the same signature as the controller method of the microservice as it wishes to establish communication.

Exercise

- Practice 1: Shopping Application

PRACTICE

Shopping Application

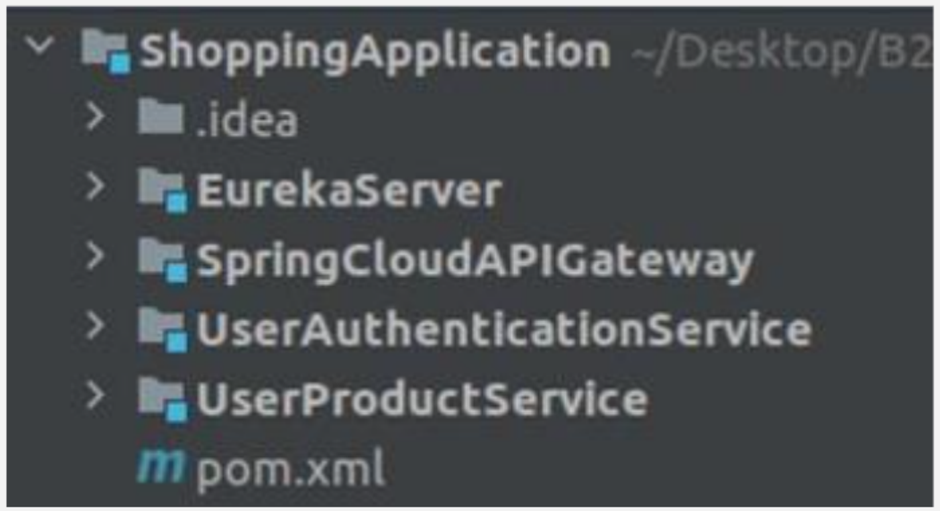
Consider a shopping application that enables users to shop for products on any smart device. The application provides multiple features for all its registered users. Still, the user must first register with the application to access these features.

Let’s create multiple microservices for the application.

1. The customer must first register with the application.
2. The customer must log in with credentials such as ID and password.
3. The customer can access features such as adding products, deleting products, etc.
4. Implement Feign Client to save the user's data once they’ve registered.

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.



The structure of the Application

Task: Practice 1

- `EurekaServer` is the service where all the other services will register themselves.
- `SpringCloudAPIGateway` is the service that will act as API Gateway.
- `UserAuthenticationService` is the service that will have 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.

Task: Practice 2

- 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. The route should be written using the application name in the `application.yml` file, instead of the URI of the application. Do not write the port number.
- 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 3

- Create a new project using the Spring Initializer to build the Eureka Server.
- Add the dependencies for Eureka Server in the pom.xml.
- Add this project as a module in the parent pom.xml.
- Add dependencies for Eureka client in UserAuthenticationService, UserProductService, SpringAPIGatewayService.
- Configure all the services as clients of the Eureka Server.
- Start the Eureka Server at http://localhost:8761/.
- Start the other services and ensure that they are registered on the Eureka Server.
- Run and test your application using Postman.

Task: Practice 4

- Add the dependencies for Spring .Cloud Feign Client in the pom.xml of `UserProductService`.
- Set up the UserProxy in the appropriate service.
- Autowire the UserProxy in the service layer.
- Use the proxy to send the data to the service.
- So now calling /register should call the /save method internally using microservices communication.
- Run and test your application using Postman.

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-feign-client-c4-s5-pc-1-shopping-application` against which your submission will be evaluated, under **Assignment Repository**
- Select your solution repository `pc-1-shopping-application` under **Search Submission Repo**
- Click on **Submit**.
- The results can be viewed in the **Past Submissions** screen.