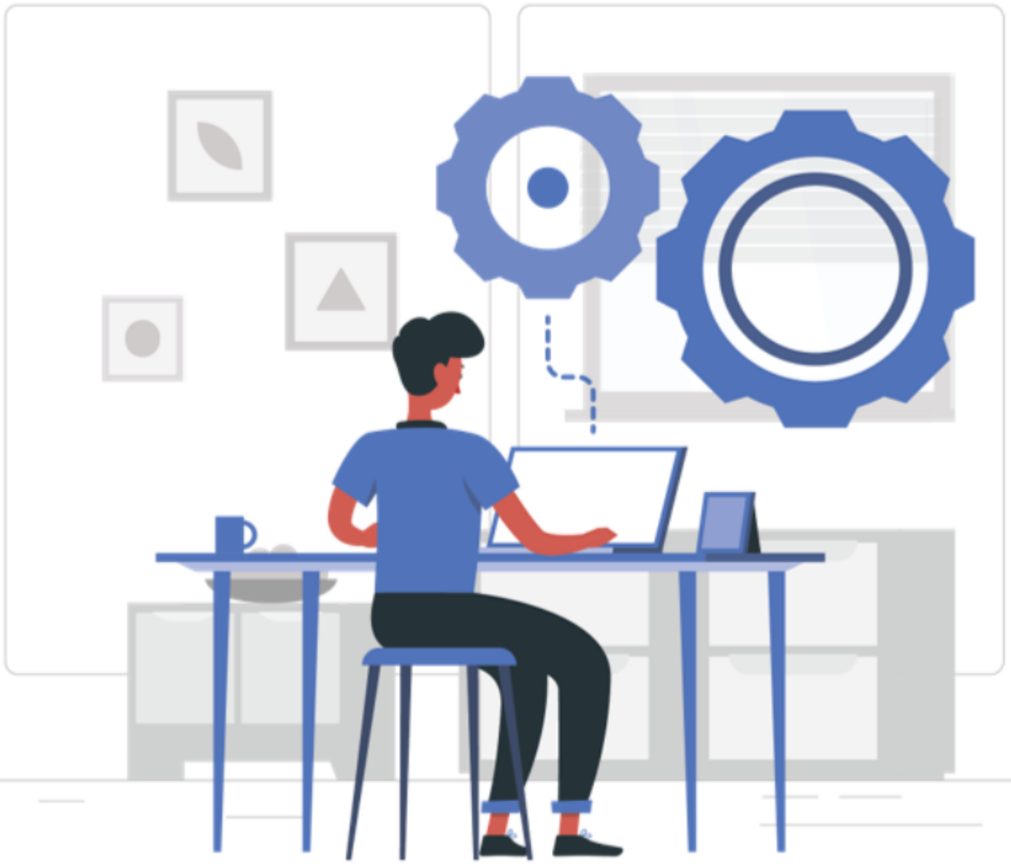


Challenge
Establish Synchronous Communication Among Microservices by Using Feign Client

Music Streaming Application

Implement a music streaming application that lets users stream music on any smart device. The application should provide multiple features for all registered users, like adding a track to a playlist, deleting a track from the playlist, updating a specific track, displaying all the tracks in the playlist, etc.

The user must register with the application to access some of its features.

CHALLENGE

Music Streaming Application (contd.)

Use the microservices approach to implement the application.
Also implement SpringCloud Gateway and Eureka Server.
All other services should register themselves to Eureka Server.
The Feign client, which will handle microservices communication, needs to be incorporated.

CHALLENGE

Communication Between Microservices

The client sends a request for a service.

/register

/login

/crud

API Gateway

/login

/register

/crud

Feign Client

User data

email

password

UserTrackService

UserAuthenticationSevice

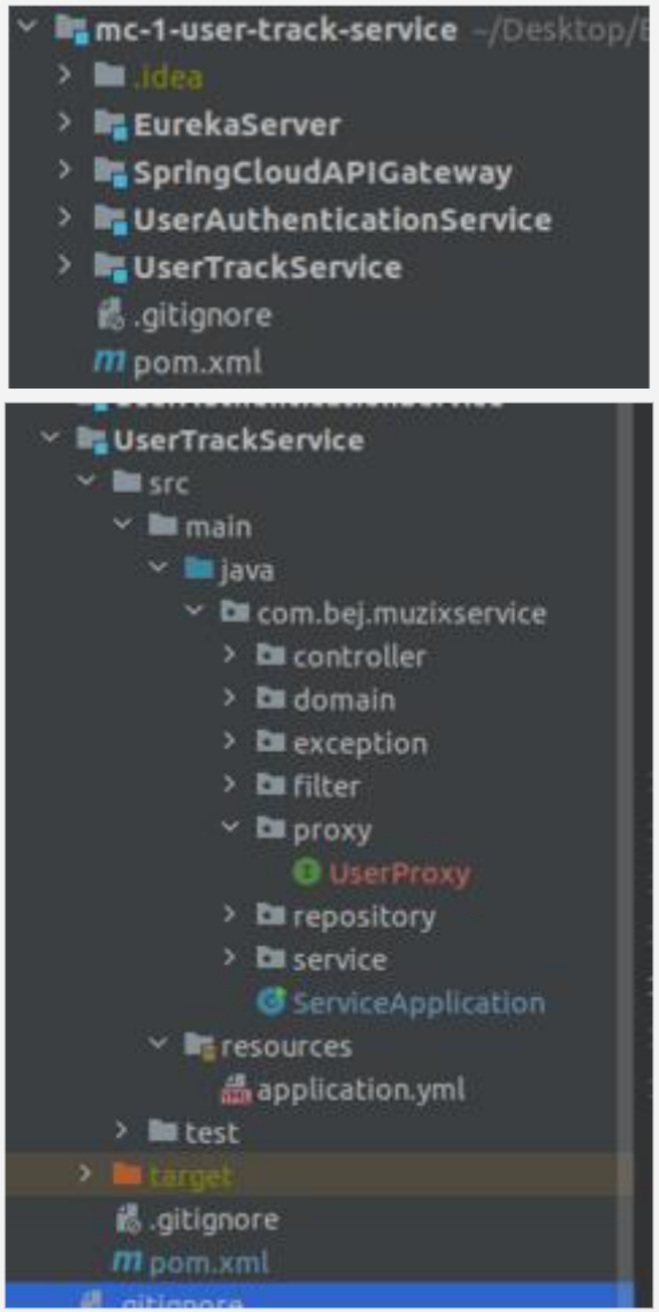
MongoDB

MySQL

Eureka Server

Instructions for the challenge

- 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: Challenge 1

- `EurekaServer` is the service that will register all other microservices.
- `SpringCloudAPIGateway` is the service that will act as API Gateway.
- `UserAuthenticationService` is the service that will have login and save the user functionalities.
- `UserTrackService` is the service that will have all the CRUD functionalities related to Track object.
- `pom.xml` is the parent pom.

Task: Challenge 1

- Create a new project using the Spring Initializr 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: Challenge 2

- Create a new project using the SpringInitializr to build the Eureka Server.
- Add the dependencies for the Eureka Server in the pom.xml.
- Add the Eureka Server in the module of the parent pom.
- Add the Eureka client dependency in the `UserAuthenticationServer`, `UserTrackService`, and `SpringCloudAPIGateway` services.
- All Services must get registered in Eureka Server.
- Configure the routes in the API Gateway with service name.
- 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: Challenge 3

- Add the dependencies for Spring Cloud Feign Client in the pom.xml of **UserTrackService.**
- Annotate the main class of UserTrackService with `@EnableFeignClients.`
- Create a package proxy inside this package to create the UserProxy class.
- This class will have same method saveUser that exists in the controller of UserAuthentication.
- Autowire the UserProxy in the service layer of UserTrackService.
- Use the proxy to send the data to the service.
- So now calling /register should call /save the 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-mc-1-music-streaming-application` against which your submission will be evaluated, under **Assignment Repository**
- Select your solution repository `mc-1-music-streaming-application` under **Search Submission Repo**
- Click on **Submit**.
- The results can be viewed in the **Past Submissions** screen.