Recruiting - Back-end - Homework (new)

<p style="text-align:center">WORK IN PROGRESS</p>

Goal

The goal of this homework is to assess the candidate's capability, to provide a well architected microservices based solution, given a monolith webapp.

Problem

Replace existing **Loan Request flow** of a monolith webapp with a fleet of microservices.

In the existing Monolith Loan Request flow we capture the below Objects in a single Json payload and save User, LoanApplication and Customer details in the same database

1. LoanApplication {amount, duration}
2. User {email, password}
3. Customer {firstName, lastName, phone, address, city, postalCode}
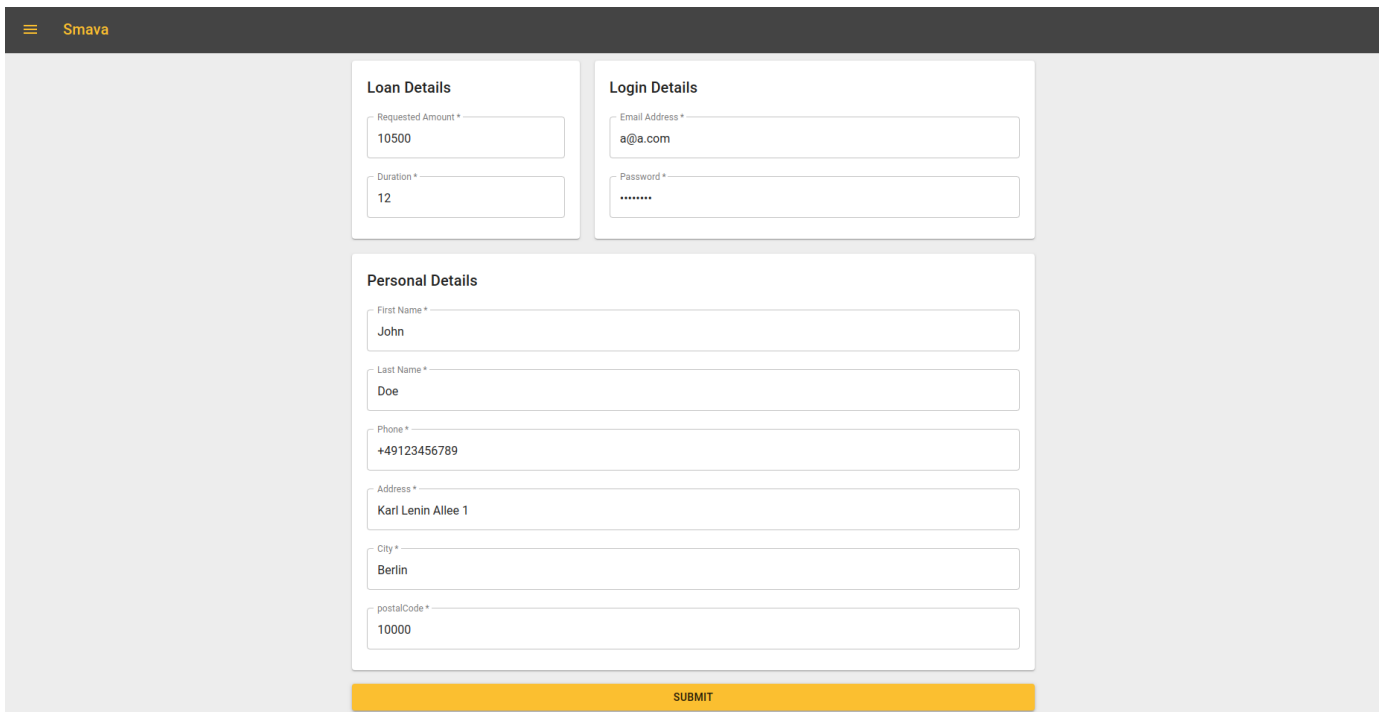
Repo of Monolith

https://github.com/smava/monolith

Refer **RegistrationController** in the above monolith to understand how the old loan request flow works.

New Loan Request Flow

To visualize the UI and how the new request flow would look like, refer the sample UI and the flow below

Assume backend processing starts immediately after the user submits the above form.

Imagine the frontend applications orchestrate the loan request flow in the way explained below

1. Call auth microservice to register new user. Get an oauth token in /register api response along with userId.
2. Call your new microservices through protected gateway microservice using the access_token obtained in Step1

3. Save Loan Application and Customer details in our system for processing - **This is the implementation we need from you**
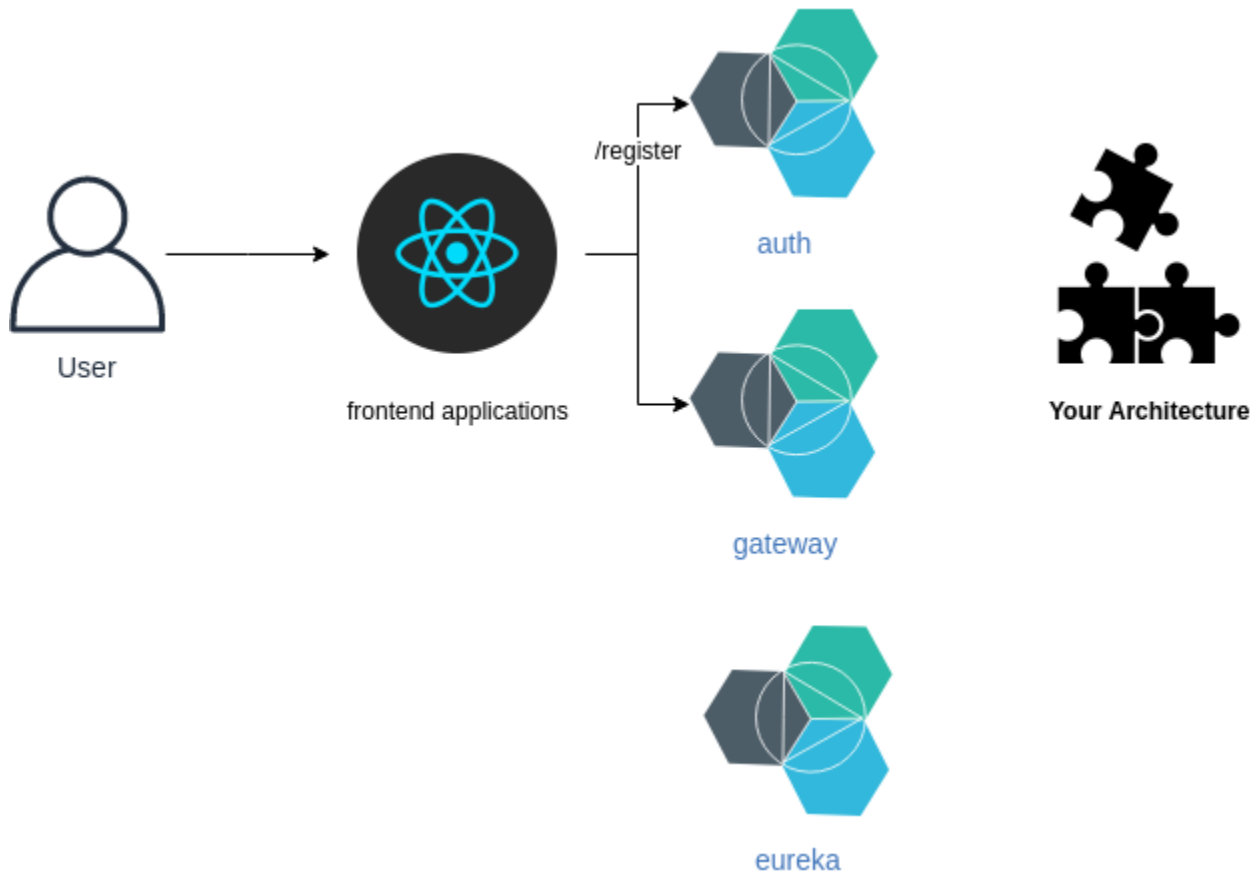
Challenge

The challenge is to **save** the Loan request data for processing in a fleet of microservices you will develop.

During Loan Request processing we carry out the below 2 backend operations for which you have to **setup 2 microservices**.

1. Fetching credit score from third party and updating the score of Customer
2. Fetching offers from different banks and updating the status of Loan Application accordingly (0 Offer = DENIED, 1 Offer = APPLIED)

Your job is to just create these microservices with best practices in mind and provide 3 REST Apis mentioned in TODO APIs section.

For the sake of simplicity let's just name them **customer-microservice** and **loan-application-microservice**

TODO APIs

The following are the APIs you need to implement

Save LoanApplication

> POST /api/loanapplications

Request

```
{
  "amount": 1000.00,
  "duration": 12
}
```

Response

```
{
  "id": 101,
  "amount": 1000.00,
  "duration": 12,
  "status": "CREATED"
}
```

Get Loans by CustomerId

> *GET /api/loanapplications/customers/{customerId}*

Response

```
[
  {
    "id": 101,
    "amount": 1000.00,
    "duration": 12,
    "status": "IN_PROGRESS"
  }
]
```

Save Customer

> *POST /api/customers*

Request

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "johndoe@example.com",
  "phone": "+49123456789",
  "address": "Karl Lenin Allee 1",
  "city": "Berlin",
  "postalCode": "10000"
}
```

We expect a clean Restful implementation.

Given

We have the below microservices ready, so that you can concentrate on the solution right away

1. **auth** microservice which acts as the authorization service, we store User data in auth microservice's database. i.e. email as username and password
2. **eureka** for service discovery
3. **gateway** microservice acting as the API gateway to those microservices you will develop

Additional Requirement

Assume that the microservices you provide will be used to create a search functionality of Customers based on Loan Application Amount and status.

Create a **Story.md** file in your repository directly under root folder with description on how to design such an API.

> *You are free to choose any microservice to implement this, with API path of your choice*

Request

```
{
  "minAmount": 1000.00,
  "maxAmount": 10000.00,
  "status": "APPLIED"
}
```

Response

```
[
  {
    "id": 101,
    "email": "a@a.com"
  },
  {
    "id": 102,
    "email": "b@b.com"
  }
]
```

DO NOT IMPLEMENT this search in code

Expectations

1. New Microservices should be created adhering to the best practices and Microservice design patterns
2. Your APIs should be accessible through secured Api gateway i.e. A registered user's oauth token should be able to authorize a client to access your APIs
3. Should use database and a data access layer in the program to exhibit experience in them
4. Microservices should be based on Spring and Spring boot
5. Microservices should be production ready in terms of logging and monitoring
6. Write atleast one Unit and one Integration Test
7. Feel free to reuse any ready made containers for mundane cloud setup or messaging system