Overview

Design Overview

The lending platform will be implemented as a set of microservices, following the Gateway architectural pattern. The microservices will be developed using Java and Spring Boot, and data will be stored in an H2 in-memory database.

Design Choices

To meet the requirements, the following design choices have been made:

- The application will be built using Spring Boot, a popular Java-based framework for building microservices. Spring Boot provides a range of tools and pre-configurations that make it easy to get started with building microservices. Ultimately Spring Boot is built on top of the Spring framework, which provides a wide range of features and tools for building enterprise applications. This means developers can easily integrate their applications with other Spring technologies such as Spring Data, Spring Security, and Spring Cloud.
- The system will be built as a set of microservices, each responsible for a specific aspect of the lending platform.
- The microservices will communicate with each other using REST APIs. The application would not be event-driven.
- An H2 database will be used to store loan and payment data. H2 is easier to set up for development and testing purposes compared to other relational databases like MySQL or Postgres.

Solution Overview

The lending platform microservice will consist of the following components:

Loan Service

This service will be responsible for receiving loan requests from customers, processing these requests, and presenting the customer with one or more loan offers based on their maximum loan qualification. This service will be called by the Payment Service to update the loan status.

Payment Service

This service will handle loan payment processing. It will allow customers to manually make loan payments before or after the due date, or wait for the system to automatically attempt a deduction from their mobile money wallet on the due date. This service will be called by the Loan Service when a loan disbursement is requested.

Notification Service

This service will handle sending notifications to customers. It will support both email and SMS notifications and will be called by both the Loan Service and Payment Service when necessary.

Mobile Wallet Service

This service will simulate a mobile wallet and will return "failed" if the amount is 5000 and "success" for any other amount. This service will be called by the Loan Service when a loan is requested and by the Payment Service when a loan payment is attempted.

Support Services

API Gateway Service

A Single entry point for all incoming API requests and acts as an intermediary between clients and microservices.

Eureka Discovery Service

Eureka Discovery Service helps to simplify communication between the microservices by allowing each microservice to register with a central service registry and discover other microservices that they depend on.

Common Service

Provide resources shared by all microservices.

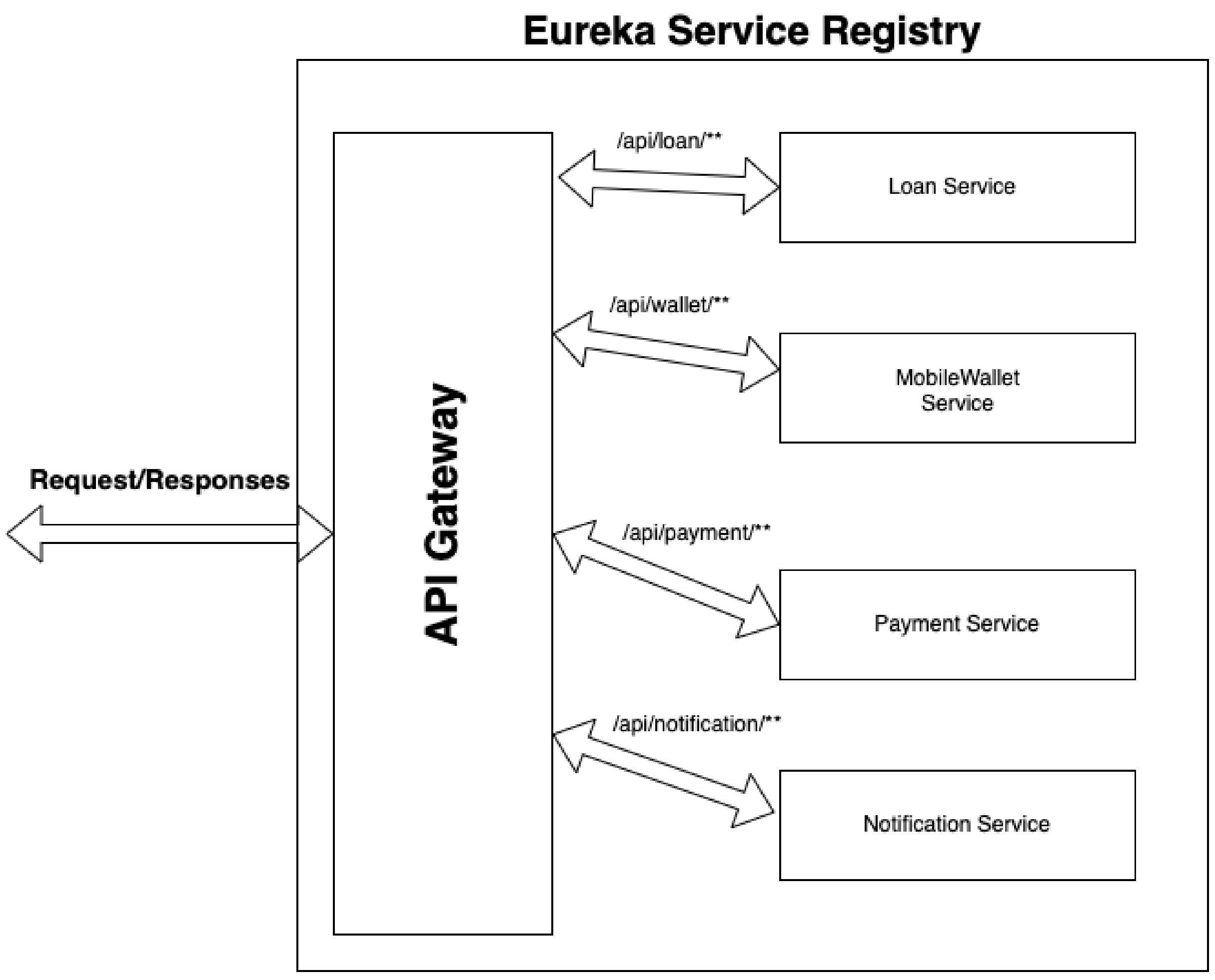
Assumptions

1. The loan offers presented to the customer will be based on their maximum loan qualification and will only include products that the customer is eligible for.
2. Customers can only accept one loan offer at a time.
3. The loan payment due date will be set to the end of the loan tenure.
4. The Payment Service will only attempt to deduct the loan payment once on the due date. If the payment fails, the customer will be notified and will have to manually make the payment.

Pitfalls and Improvements

1. The Loan Service currently does not take into account the customer's credit score, employment status, or other factors that may affect their loan eligibility. Adding these factors would make the loan offers more accurate and could help reduce the risk of default.
2. The Mobile Wallet Service currently only returns two values, "success" or "failed". Adding more values such as "insufficient funds" or "invalid account" would make it more realistic and help identify specific issues when loan disbursement or payment fails.
3. The Notification Service currently only writes to logs and returns a Boolean true. Adding actual email and SMS functionality would improve the user experience and ensure that notifications are actually delivered.
4. Have a Config Service. This microservice will manage the configuration data for the other microservices. It will provide a central repository for storing the configuration data, and the other microservices will use this data to configure their behavior.

Application Architecture



Loan Microservice

The loan microservice will receive loan requests from users, check the user's loan limit, and send back a list of loan offers. Once a user selects an offer, the service processes the loan credits to the user's wallet account, generates a payment record, sends it to the payment microservice, and sends a notification via the notification service to the user.

EndPoints

The Loan Microservice will expose the following endpoints:-

1. Loan Request

   This API receives a customerId and a walletAccountId and returns a list of loan offers that the customer is eligible for.

Part	Description
URL	/api/loan/
Method	POST

Request body	{
	    “customerId”: 1,
	    “walletAccountId”:: “XXX-2112312”
	
	}

Response body	[
	  {
	    "maxAllowableLimit": 1000,
	    "interest": 10.0,
	    "tenureInDays": 15,
	    "id": 1
	  }
	]

2. Process Loan

This API receives customerId, walletAccountId, and loanProductId and returns a LoanDto object after successfully processing a loan otherwise the customer receives an error message.

Part	Description
URL	/api/loan/process-loan-request
Method	POST

Request body	{
	   “customerId”: 1,
	   “walletAccountId”: “XXX-2112312”
	   “loanProductId”: 1
	}

Response body	{
	   "loanId": 1,
	   "loanProductId": 1,
	   "walletAccountId": null,
	   "userId": 1,
	   "interest": 100.0,
	   "loanAmount": 1000.0,
	   "totalAmount": 1100.0,
	   "createdOn":
	"2023-03-13T06:57:51.007+00:00",
	   "dueDate":
	"2023-03-28T06:57:51.007+00:00",
	   "status": "ACTIVE"
	}

3. Update Loan Payment Status

   This API receives loanId, and LoanStatus. Updates the loan status and returns 200.

Part	Description
URL	/api/loan/{loanId}/update-payment-status/{status}
Method	POST
Request body	
Response body	OK 200

MobileWallet Service

The mobile wallet service will return the user's loan limit. The returned loan limit will be used to generate a list of loan products the user is eligible for.

EndPoints

The MobileWallet Microservice will expose the following endpoints:-

1. Loan Limit

   This API receives an accountId URL param and returns a LoanLimit object for the account

Part	Description
URL	/api/wallet/{accountId}/loan-limit
Method	GET

Request body	
Response body	{ "walletAccountId": "1", "loanLimit": 1000 }

Payment Service

The payment service will process payments automatically when the loan matures and accept manual payment requests from users.

EndPoints

The Loan Microservice will expose the following endpoints:-

1. Payment Request

   This API receives a loanId URL param and returns Void. For successful payment, a notification will be sent to the customer after updating the loan to paid status.

Part	Description
URL	/api/payment/{loanId}/make-payment
Method	GET
Request body	
Response body	

Notification Service

The notification service will send notifications to users via email or SMS.

EndPoints

The Notification Microservice will expose the following endpoints:-

1. Notify Request

   This API receives a NotificationDto object and returns a list of loan offers that the customer is eligible for.

Part	Description
URL	/api/notification/notify
Method	POST

Request body	{
	“phoneNumber”: “+2547000000”,
	“mail”: “john@doe.com”,
	“message”:” Payment succeeded”
	
	}

Response body	{
	"Notification Sent"
	}