API DOCUMENTATION

POSTMAN
https://documenter.getpostman.com/view/9220824/2sA3dxEXTW

Technical Documentation Summary

Overview

This microservices-based application handles user registration, authentication, and authorization. It is built using Laravel 11 and Dockerized via Laravel Sail. The architecture includes multiple decoupled services for specific functionalities.

API Gateway

The API Gateway, a Laravel 11 application, acts as a proxy, routing requests to appropriate microservices:

- **Authentication**: Handles user registration and login.
- **Authorization**: Manages user permissions and roles.
- **Profile**: Manages user profile data.

Kafka Integration

Kafka is used for messaging between microservices. For example, when a user registers, the authentication service sends messages to the authorization and profile services to update user data.

Authentication Microservice

During the login action, the API Gateway checks with the authentication microservice to verify user existence and correctness. If the credentials are correct, it continues by sending a token back to the client. For profile checks, the gateway verifies authorization with the authorization microservice before proceeding with the request.

Authorization Microservice

This microservice manages user permissions and roles. It ensures that users have the necessary permissions to access specific resources. The API Gateway communicates with this service to validate user authorization before processing requests that require specific permissions.

Profile Microservice

This service manages user profile data. When a user registers, the authentication microservice sends a message via Kafka to the profile service to create and update user profiles. Kafka ensures that all microservices are synchronized and that user data is consistently updated across the system.

Security

- **JWT**: Token-based authentication.
- **Password Hashing**: Secure password storage.

Error Handling

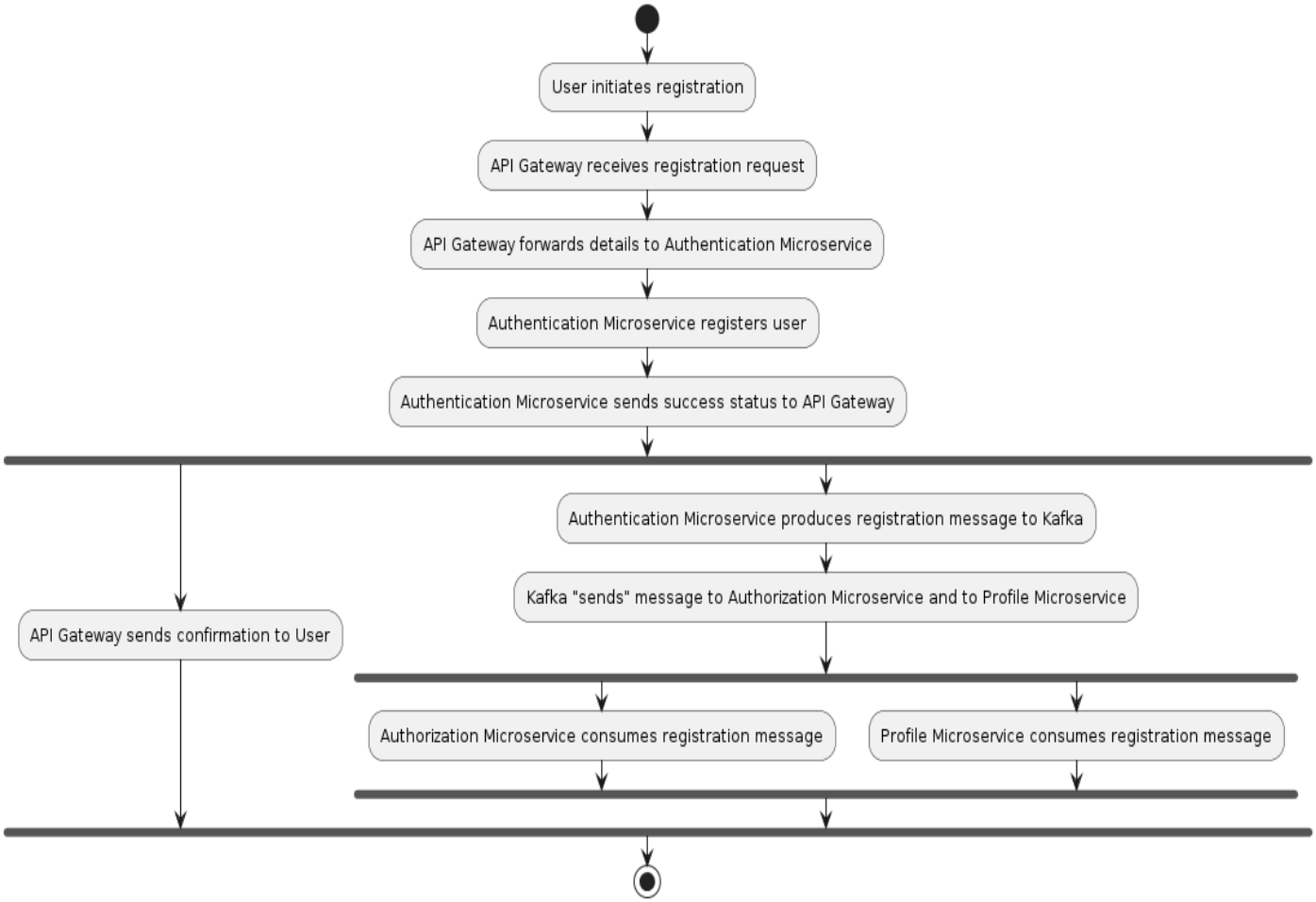
Standardized response codes and error messages ensure consistent API behavior.

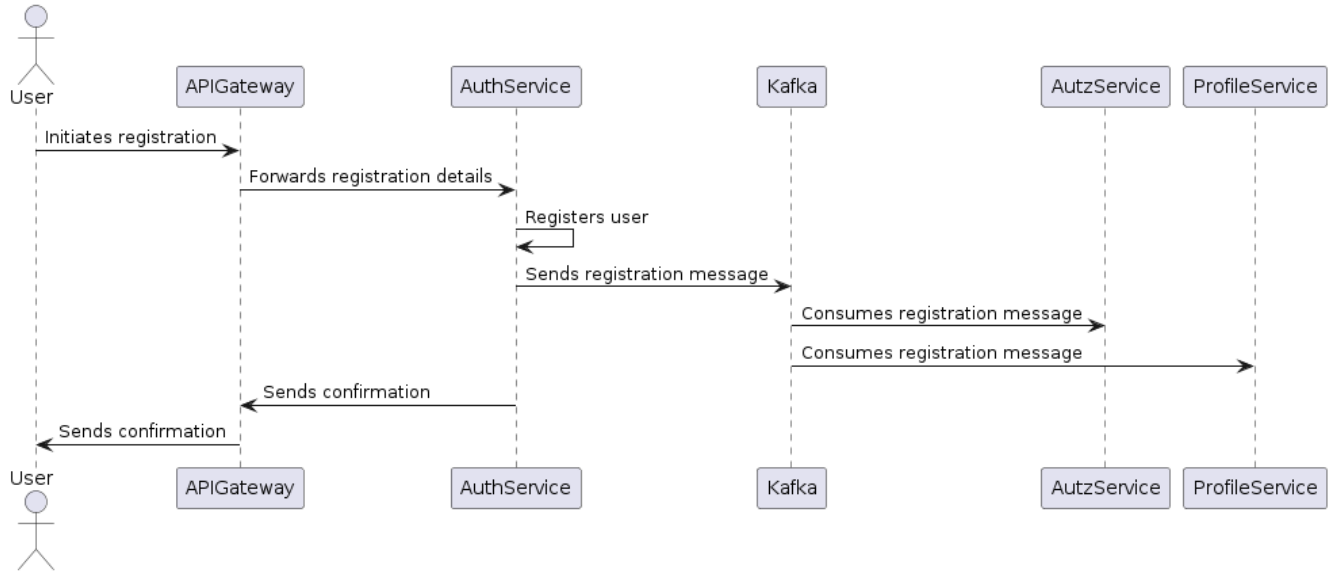
For detailed API specifications, you can access the full documentation here.

***

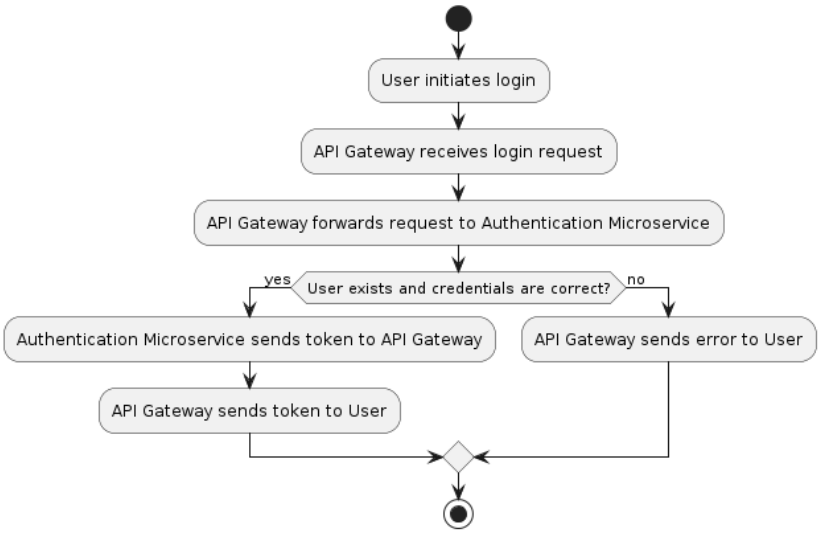
UML Diagrams

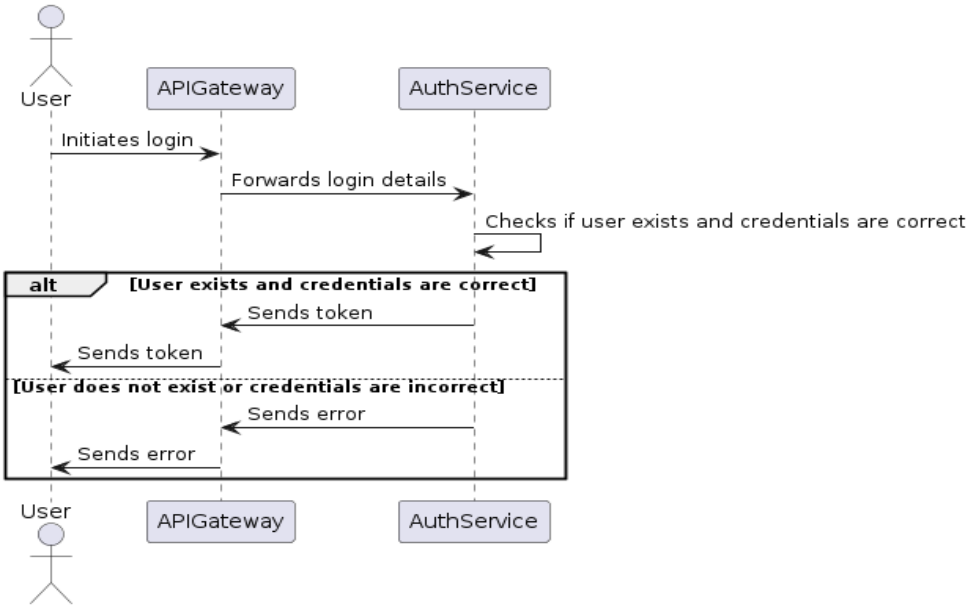
Registration



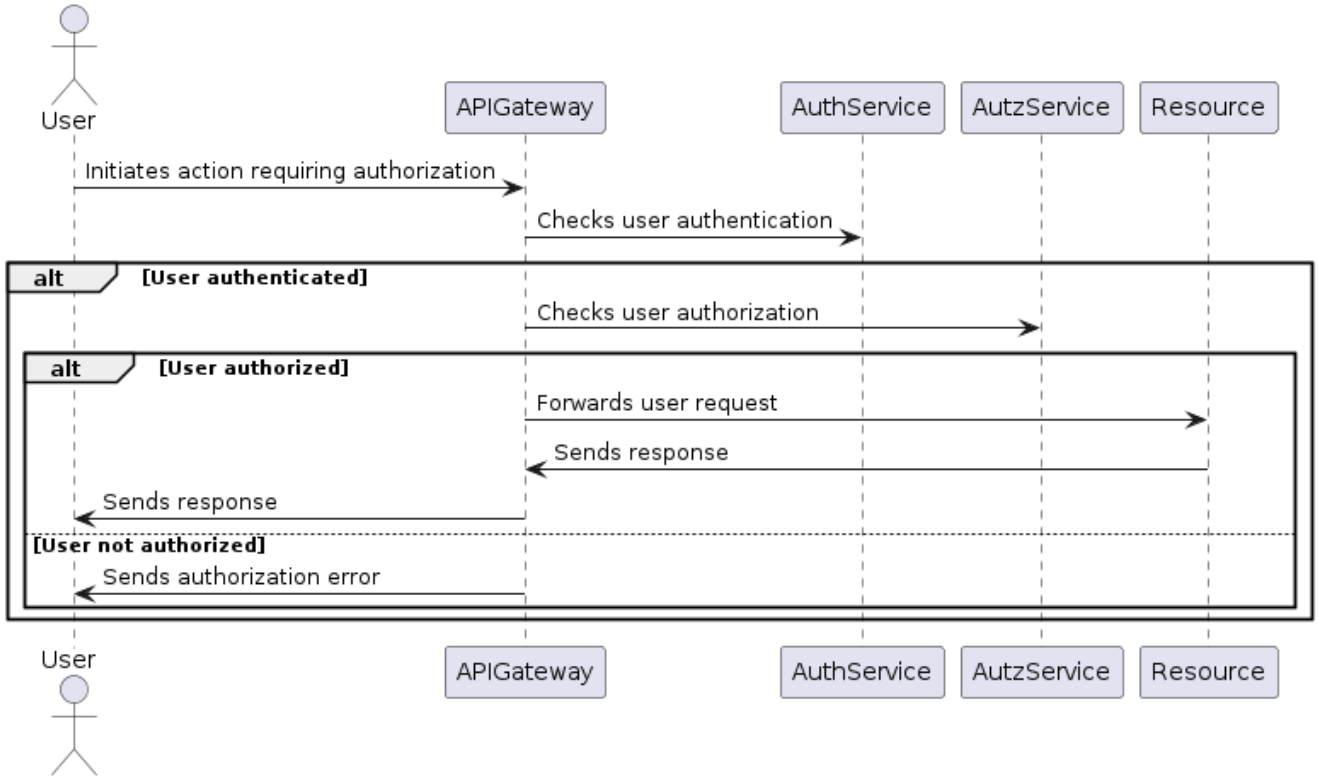

User
APIGateway
AuthService
Kafka
AutzService
ProfileService
Initiates registration
Forwards registration details
Registers user
Sends registration message
Consumes registration message
Consumes registration message
Sends confirmation
Sends confirmation
User
APIGateway
AuthService
Kafka
AutzService
ProfileService

Login


User initiates login
API Gateway receives login request
API Gateway forwards request to Authentication Microservice
yes
User exists and credentials are correct?
no
Authentication Microservice sends token to API Gateway
API Gateway sends error to User
API Gateway sends token to User


User
APIGateway
AuthService
Initiates login
Forwards login details
Checks if user exists and credentials are correct
alt
[User exists and credentials are correct]
Sends token
Sends token
[User does not exist or credentials are incorrect]
Sends error
Sends error
User
APIGateway
AuthService

Get Authorized Resource (Profile in our case)



***

Project Structure

The project consists of multiple microservices organized as follows:

```
- MicroServices (root)
  -- kafka
  -- datadog
  -- gateway
  -- authentication
  -- authorization
  -- profile
```

Setup Script

Inside the `gateway` project folder, there is a script named `setup_microservices.sh`. This script is designed to be executed after all the microservices have been cloned.

Usage

Please check here for details:
https://github.com/miloskec/gateway/blob/master/documentation/PlantUML/MicroServices.md

Instructions for basic **branch** - setup:

1. Clone all the microservices into the `MicroServices` directory as outlined above.
2. Prepare the development environment and install dependencies.
3. Navigate to the `gateway` folder.
4. Run the `setup_microservices.sh` script to initialize and configure all the microservices.

Instructions for **production-prepare** or **dockerhub-example** branch – setup:

1. Just download setup.sh or setup-from-dockerhub.sh script (https://github.com/miloskec/setupscript)
2. Execute it and follow instructions

Code Quality and Static Analysis

sail exec gateway ./vendor/bin/pint

or – for custom docker-compose

(***./vendor/bin/sail -f <docker-compose-file> exec <service-name> ./vendor/bin/phpstan analyze***)

sail exec gateway ./vendor/bin/phpstan analyze

or – for custom docker-compose

(***./vendor/bin/sail -f <docker-compose-file> exec <service-name> ./vendor/bin/phpstan analyze***)

More details:
https://github.com/miloskec/gateway/blob/master/documentation/PlantUML/MicroServices.md