FPT- APTECH COMPUTER EDUCATION

Project Report

STAFFLINK

Team 1

Team Members

Trương Đức Dương

Phùng Việt Hưng

Nguyễn Khánh Duy

Nguyễn Thành Đông

Mentor. Mr. Bùi Văn Dương

Batch: T20011E

Center: FPT-Aptech Hà Nội

Semester: 4

Contents

1. Acknowledgement

We, Team 1, are proud to announce the completion of our project, named StaffLink. We have been working hard for 6 weeks to accomplish this achievement. Within over a month, the team has overcome many problems, from lacks of knowledges about frontend technologies (android studio) and backend architecture models, to weak co-working skills. Through these difficulties, we have leant a lot of technologies and techniques in both frontend and backend, and more importantly, the essentiality of team-working in application development.

2. Introduction

Modern world has been witnessing a powerful transition in management and co-working orientations in the past decade. At many companies, showing up at work from 9.am to 5.pm is no longer a standard, especially companies in technology and creativity department. Lots of companies allow staff to work remotely, with flexible working hours and no commutes. Meanwhile, some other organizations implement hybrid work models where employees work on-site once or twice a week. The Covid-19 pandemic even accelerated the transition as employees were forced to work from home. Although the new workplace trend seemingly benefits employees' time tolerance, lower productivity should not come along. Companies demand employee and project management softwares that help managing employees' productivity and preventing negative impacts of the trend on projects' progress.

3. Problem Definition

3.1. Problem Abstraction

The application should allow users register and login, edit their profile details, and access to authorized functions. Admin can assign functions to roles and roles to users through a dynamic authorization system. Depending on assigned roles and functions, users will be given capabilities to manage projects, tasks and users, along with roles and functions.

3.2. Proposed System

The proposed system provides:

- User-friendly interface
- Built for and run smoothly on Android mobile phones

- Dynamic authorization feature
- Project and task management system with reactive data flow

3.3. Boundaries of System

- An user needs to login via an account created by Admin in order to access the application
- Authenticated users can access authorized functions
- Admin can manage users and authorizations
- Project managers can create and edit projects and tasks and assign them to employees
- Employees can access to assigned projects and tasks, manage tasks' statuses, tasks' checklists and discuss in tasks' chatrooms
- Observers can observe all projects and tasks without adding new or edit them

3.4. Development Environment

- Spring boot 3
  - Choreography architecture model (RabbitMQ 3 as message broker)
  - Spring Cloud Discovery: Eureka server, client and gateway
  - Spring MVC and Spring WebFlux
- MySql 8
- MongoDB
- Redis 6
- Android Studio
- Passive communications: WebSocket, RabbitMQ

4. Requirements and Business Flows

4.1. Customers' requirements

4.1.a. User authentication and authorization

A user first login to the application via an account provided by admin. Authenticated users can access to authorized functions and will be blocked from accessing unauthorized ones.

4.1.b. Authorization management

Admin can manage users, roles and functions, assign functions to roles and roles to users

4.1.c. Project management

Project manager can create, edit projects and add other authorized user to projects

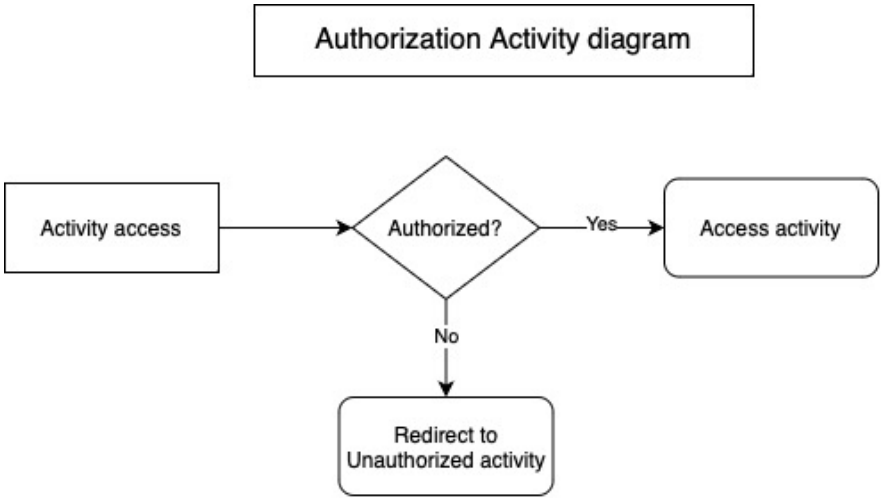
4.1.d. Task management

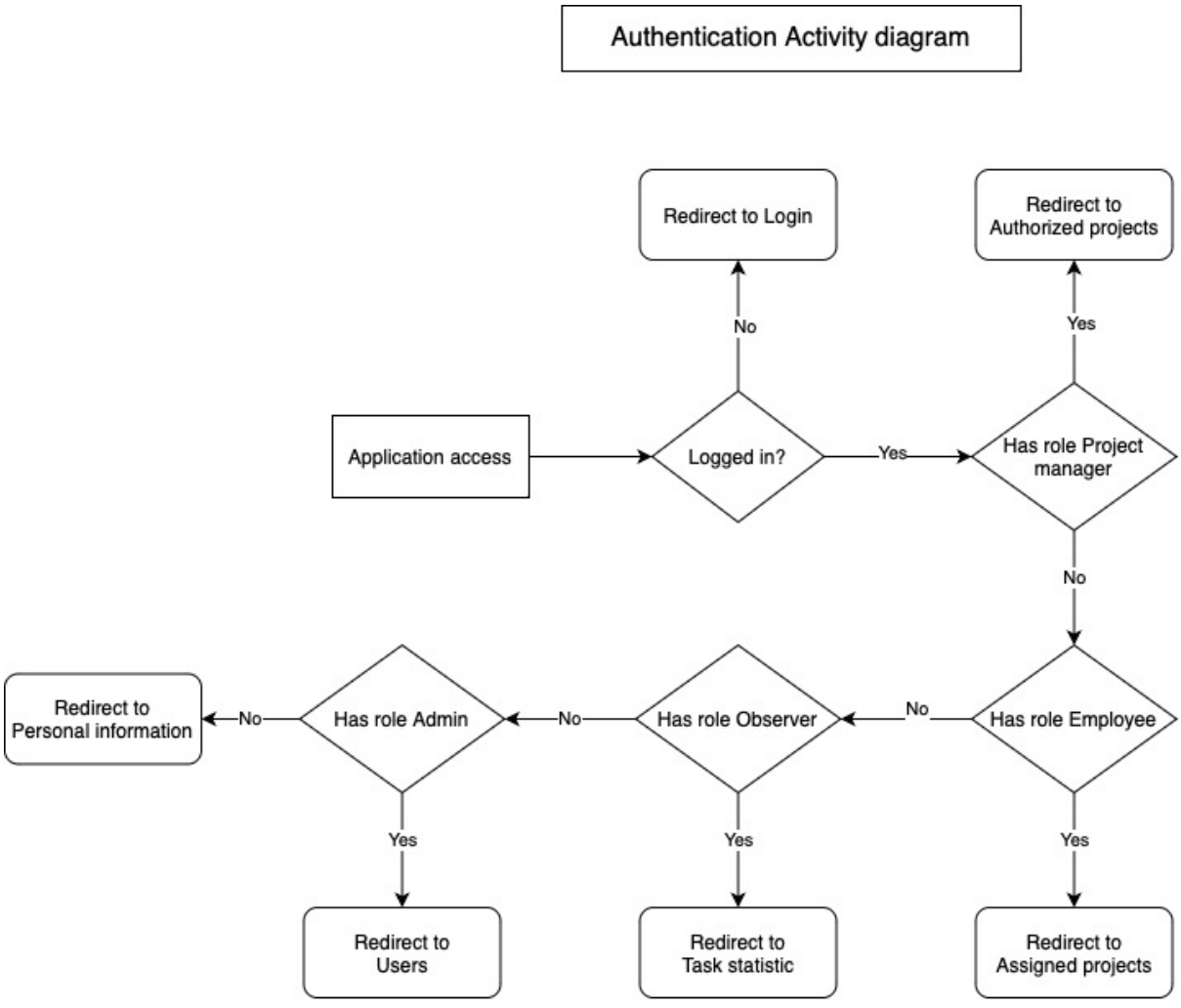
From an authorized project, a project manager can create new and edit tasks and assign them to employees

Employees are able to access assigned projects and tasks, update their statuses, download and upload attachments, manage checklists and discuss in chat section

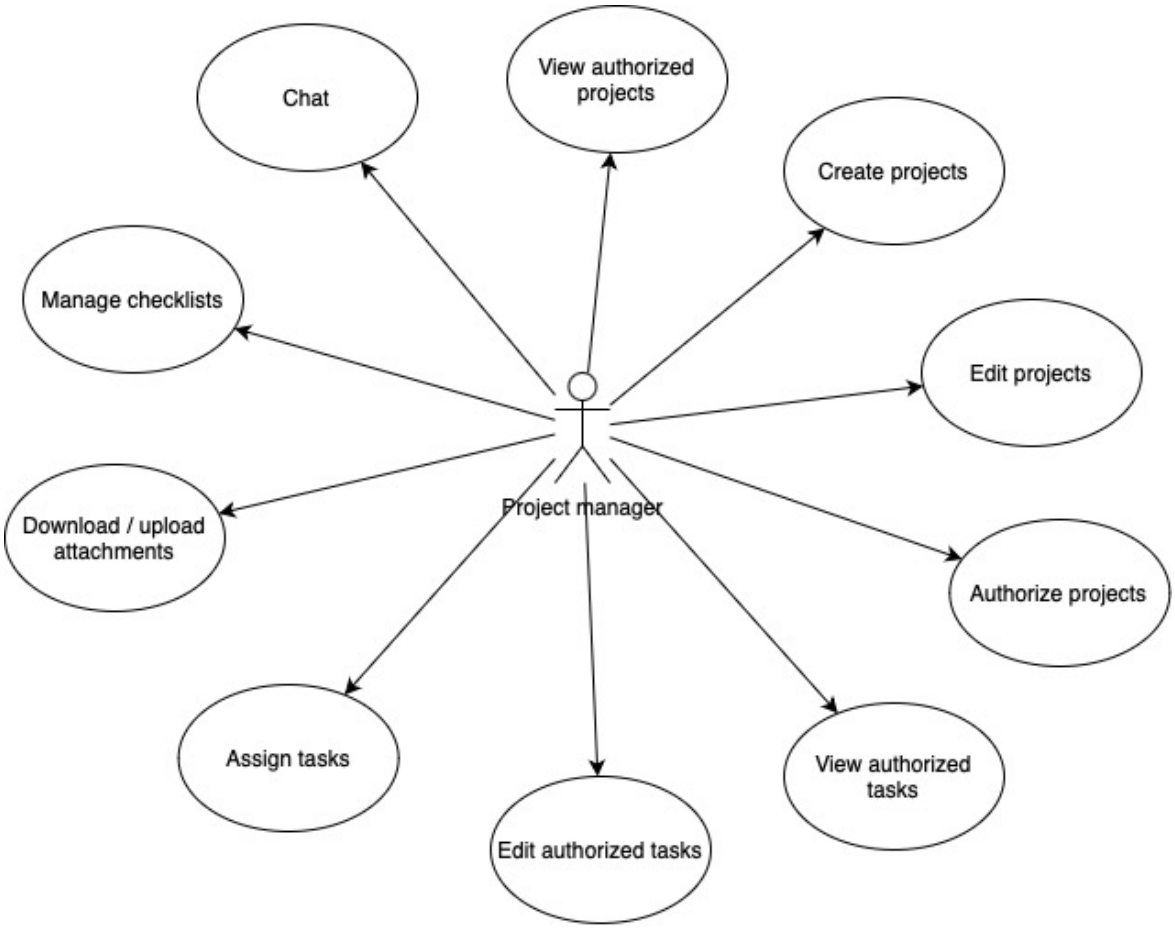
Observers are allowed to observe projects and tasks, download attachments and access task statistic function

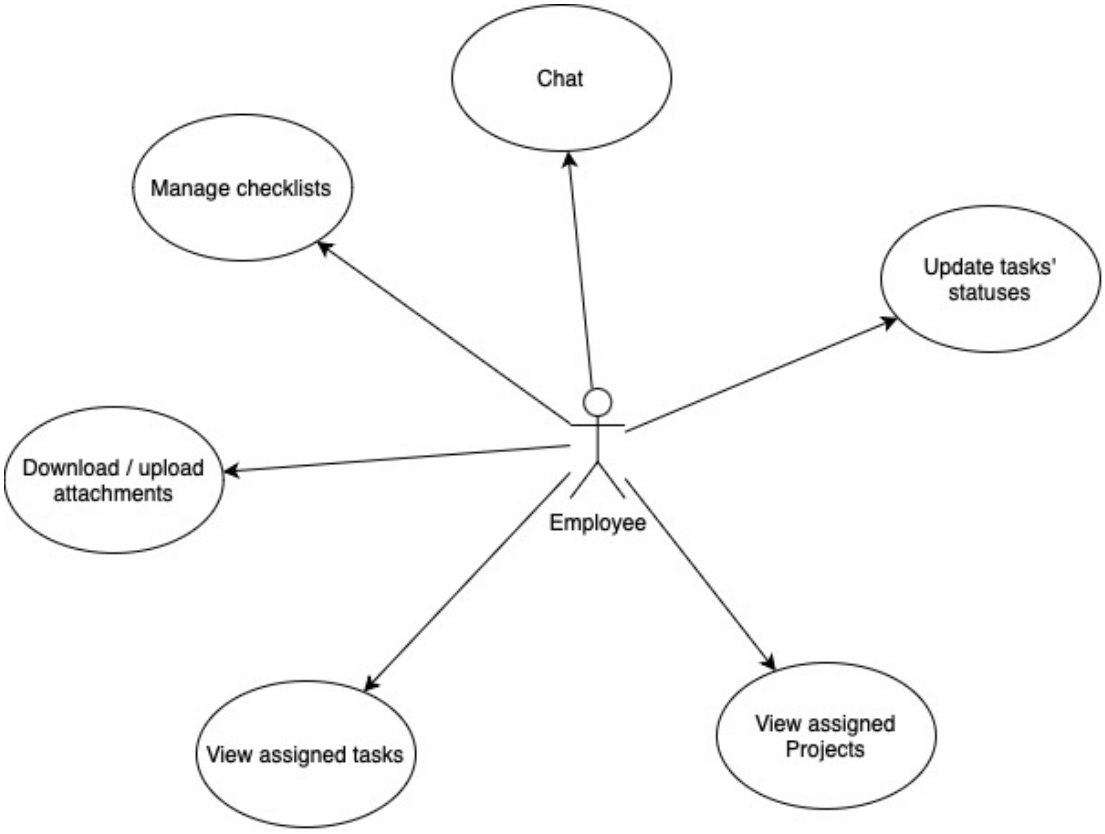
4.2. Activity Diagram





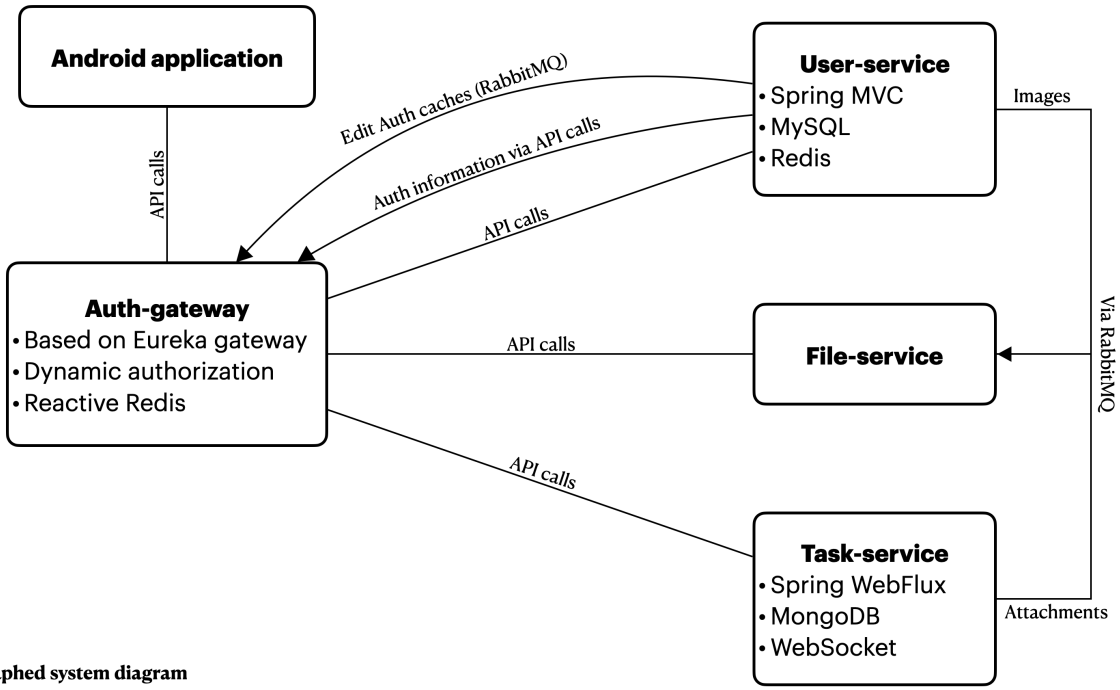
4.3. Use-case Diagram


Chat
View authorized projects
Create projects
Manage checklists
Edit projects
Download / upload attachments
Project manager
Authorize projects
Assign tasks
View authorized tasks
Edit authorized tasks


Chat
Manage checklists
Update tasks' statuses
Download / upload attachments
Employee
View assigned tasks
View assigned Projects

5. Design

5.1. System Architecture



Choreographed system diagram

5.1.a. Backend

Backend system is separated into micro-services. The services communicate to each other via choreography model, using RabbitMQ as massage broker. The services work independently, create a loosely coupled system, which prevents main threads from returning error if a failure occurs in a service.

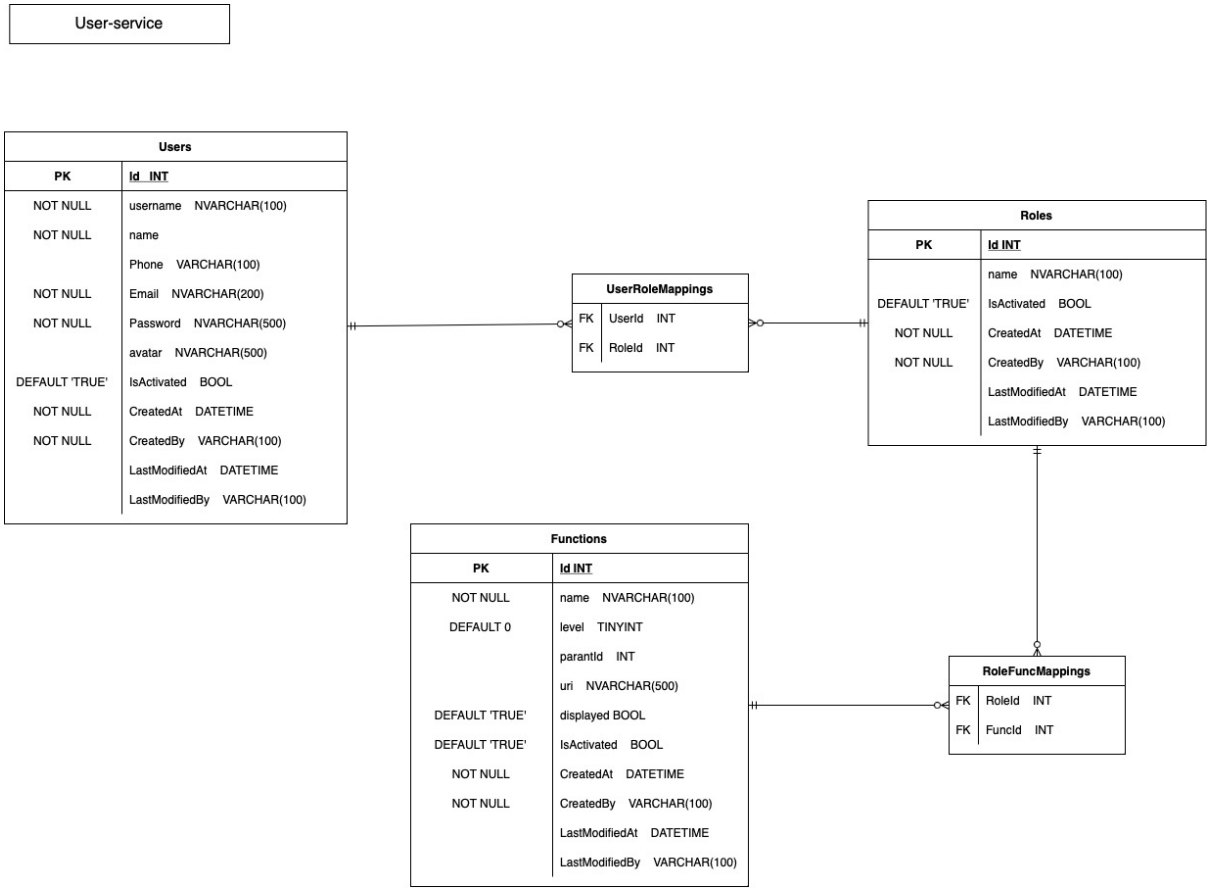
The gateway provides a fully functioning dynamic authorization system, which is able to block any unauthorized access.

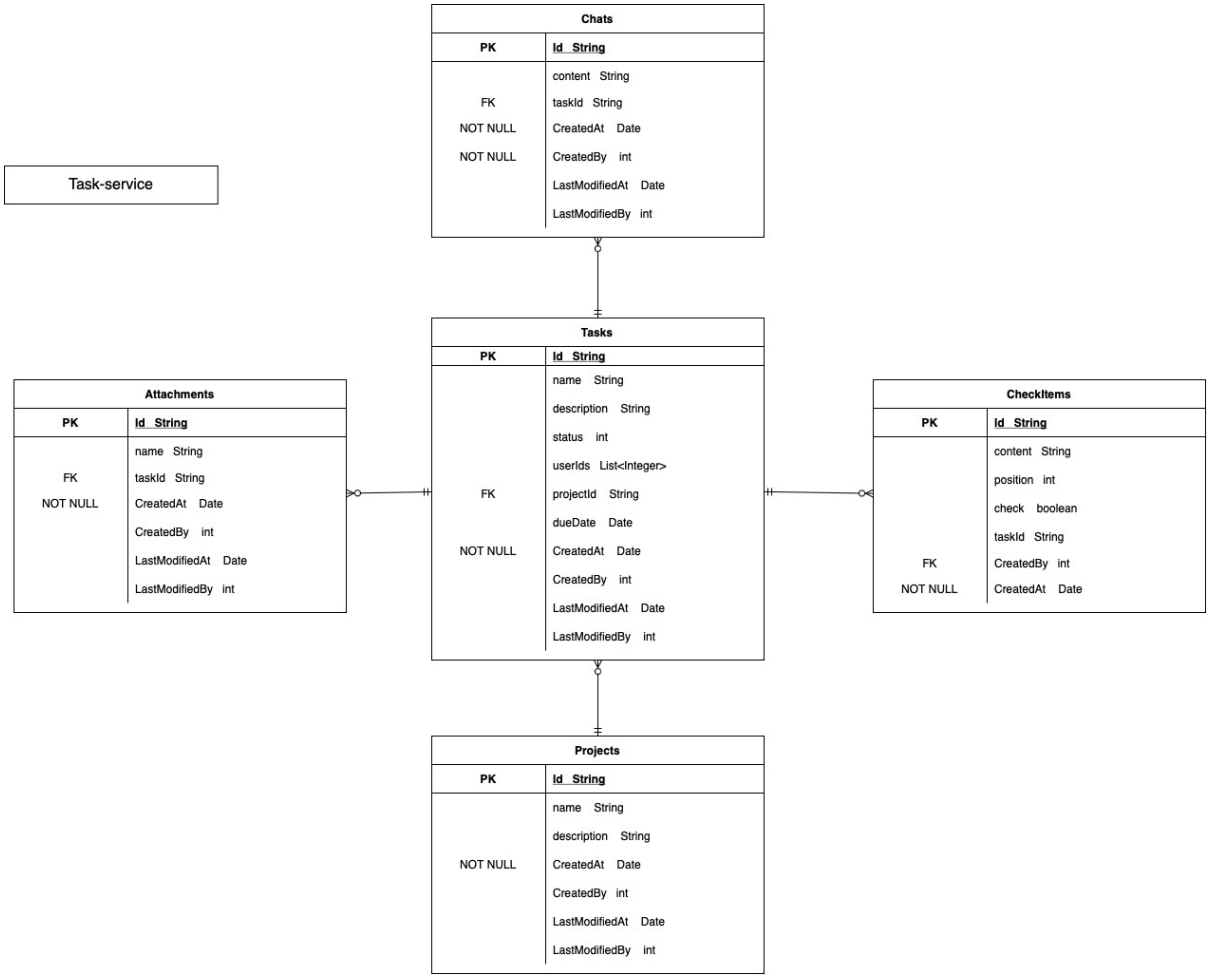
Task-service, gateway and part of file-service are based on Spring WebFlux (reactive Java) and deliver reactive responses that help improving user experience

5.1.b. Frontend

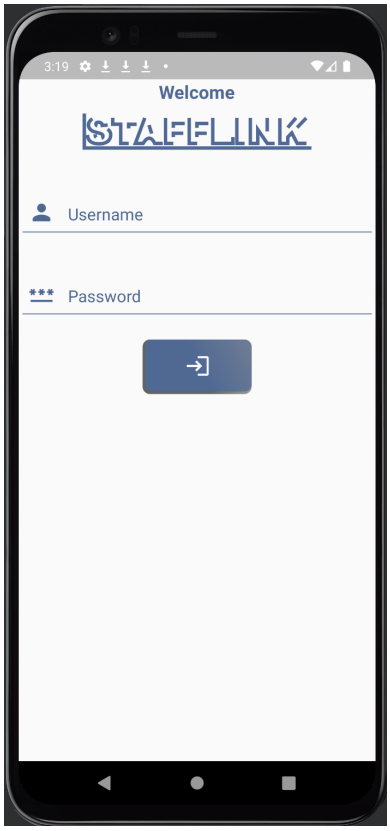
On the other hand, frontend section of the system is written for android natively. The frontend communicates with the backend system via the gateway

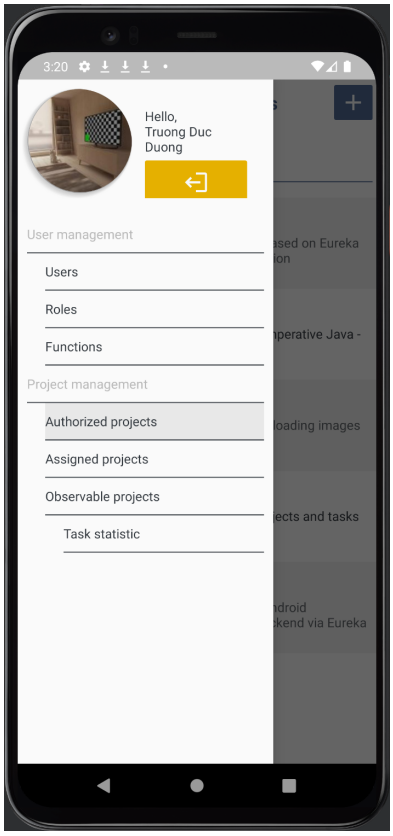
5.2. Database Design

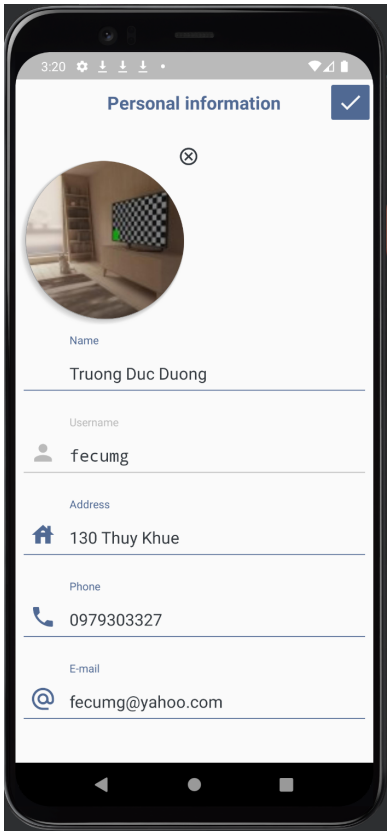


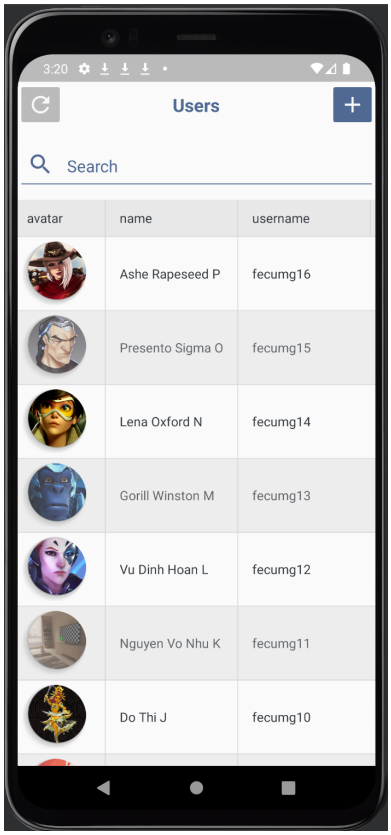

Task-service
Chats
PK
Id String
content String
FK
taskId String
NOT NULL
CreatedAt Date
NOT NULL
CreatedBy int
LastModifiedAt Date
LastModifiedBy int
Tasks
PK
Id String
name String
description String
status int
userIds List<Integer>
FK
projectId String
dueDate Date
NOT NULL
CreatedAt Date
CreatedBy int
LastModifiedAt Date
LastModifiedBy int
Attachments
PK
Id String
name String
FK
taskId String
NOT NULL
CreatedAt Date
CreatedBy int
LastModifiedAt Date
LastModifiedBy int
CheckItems
PK
Id String
content String
position int
check boolean
taskId String
FK
CreatedBy int
NOT NULL
CreatedAt Date
Projects
PK
Id String
name String
description String
NOT NULL
CreatedAt Date
CreatedBy int
LastModifiedAt Date
LastModifiedBy int

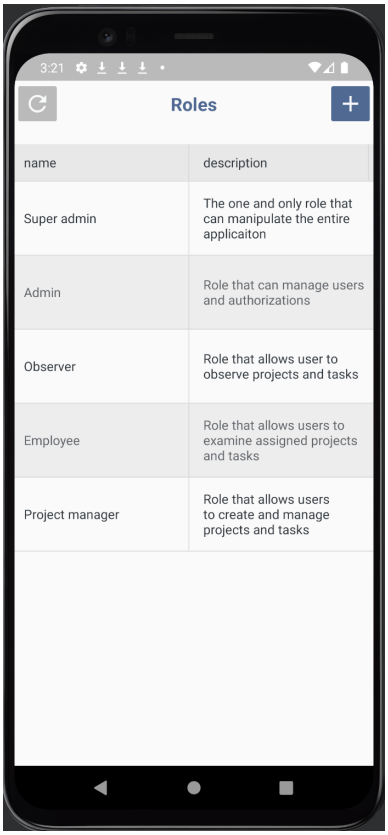
6. Screenshots

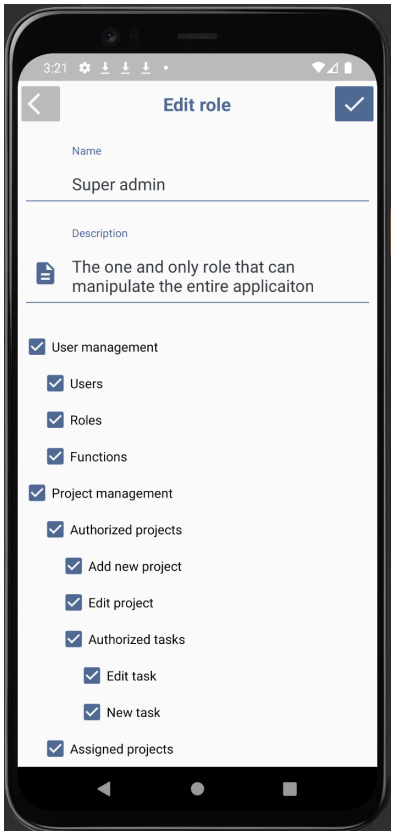


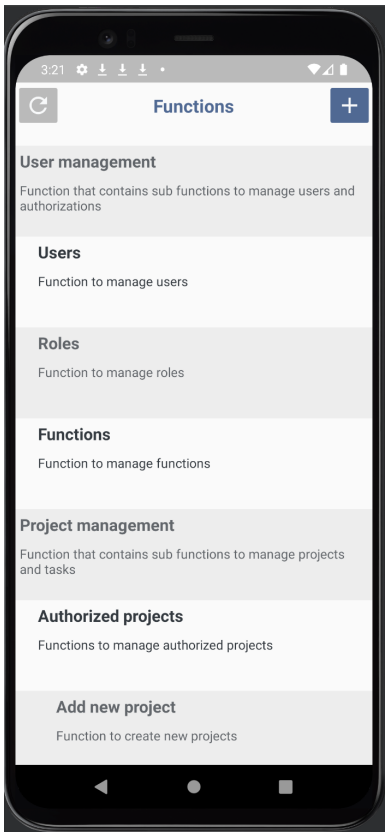


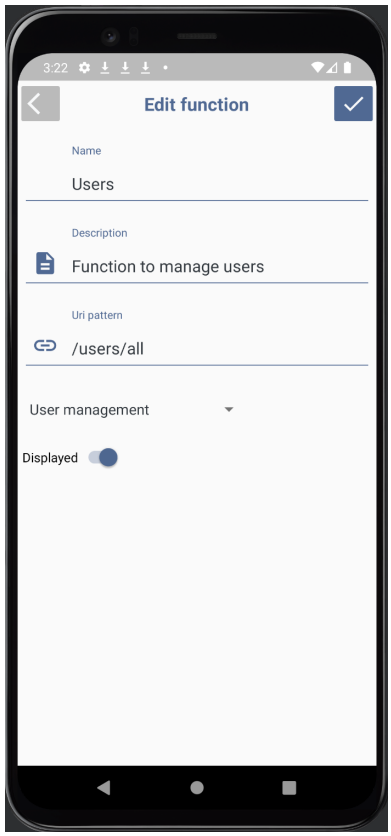


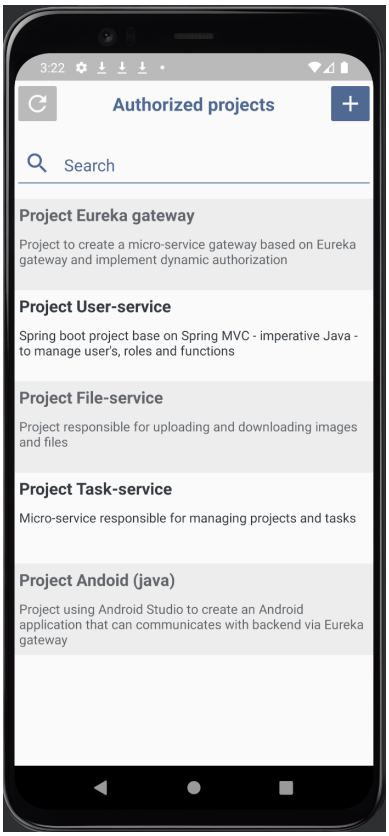


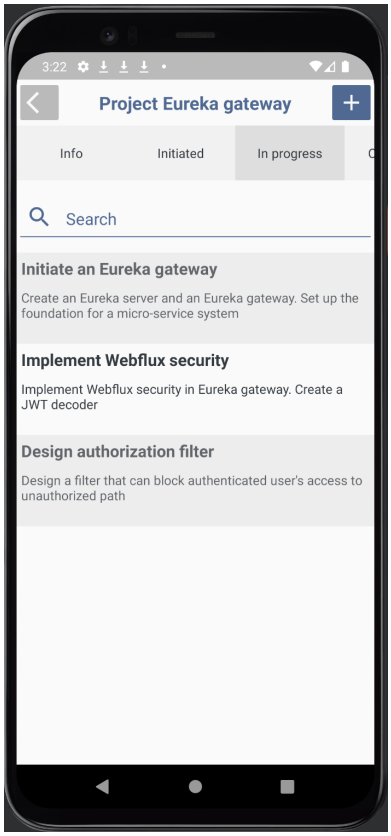


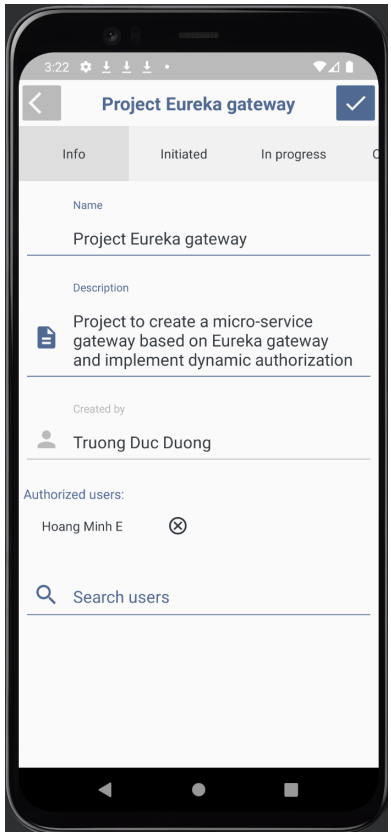


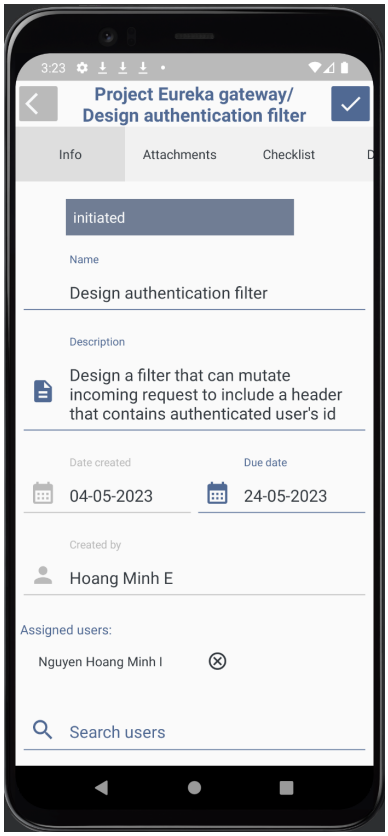

Functions
User management
Function that contains sub functions to manage users and authorizations
Users
Function to manage users
Roles
Function to manage roles
Functions
Function to manage functions
Project management
Function that contains sub functions to manage projects and tasks
Authorized projects
Functions to manage authorized projects
Add new project
Function to create new projects

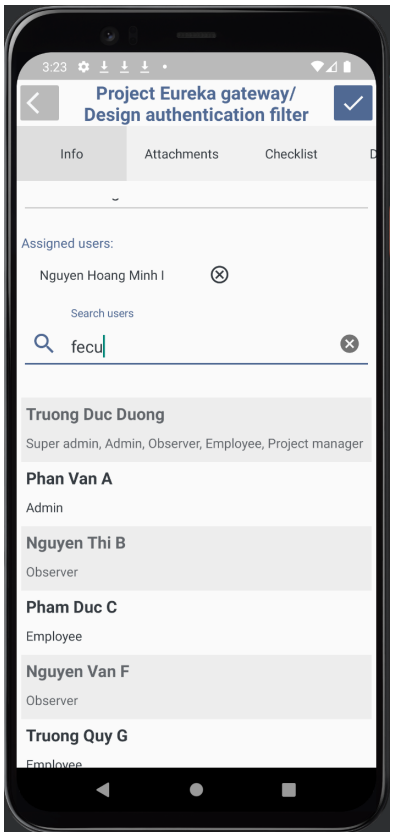

Edit function
Name
Users
Description
Function to manage users
Uri pattern
/users/all
User management
Displayed

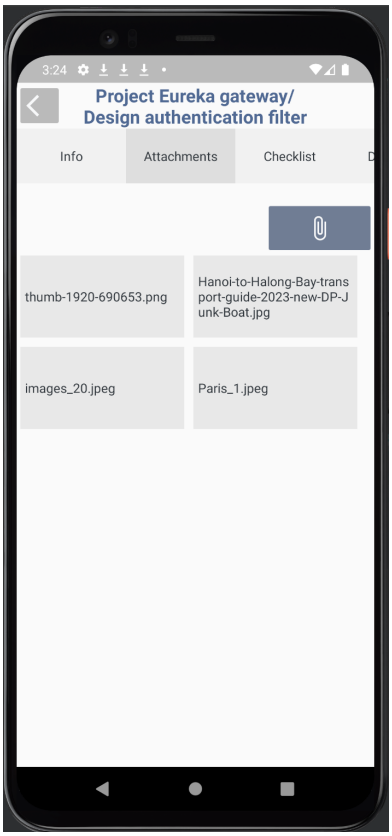

Authorized projects
Search
Project Eureka gateway
Project to create a micro-service gateway based on Eureka gateway and implement dynamic authorization
Project User-service
Spring boot project base on Spring MVC - imperative Java - to manage user's, roles and functions
Project File-service
Project responsible for uploading and downloading images and files
Project Task-service
Micro-service responsible for managing projects and tasks
Project Andoid (java)
Project using Android Studio to create an Android application that can communicates with backend via Eureka gateway

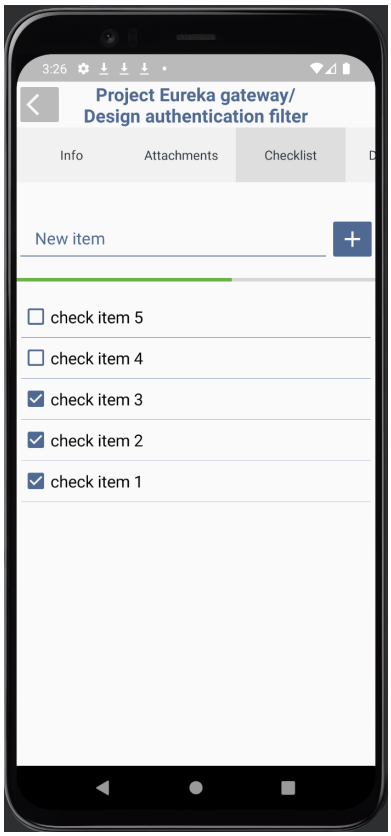

Project Eureka gateway
Info
Initiated
In progress
Search
Initiate an Eureka gateway
Create an Eureka server and an Eureka gateway. Set up the foundation for a micro-service system
Implement Webflux security
Implement Webflux security in Eureka gateway. Create a JWT decoder
Design authorization filter
Design a filter that can block authenticated user's access to unauthorized path

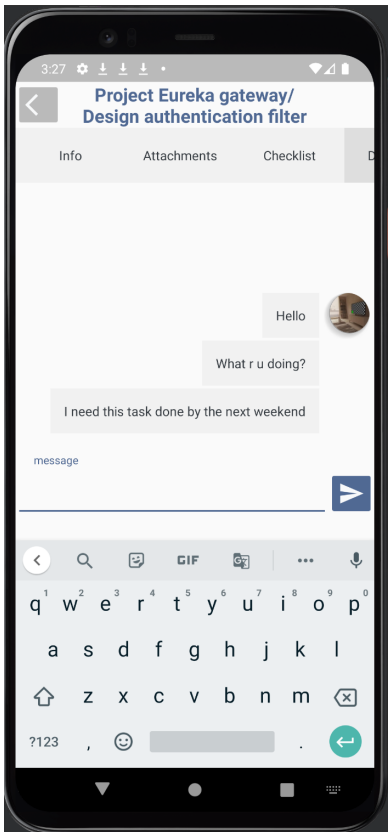

Project Eureka gateway
Info
Initiated
In progress
Name
Project Eureka gateway
Description
Project to create a micro-service gateway based on Eureka gateway and implement dynamic authorization
Created by
Truong Duc Duong
Authorized users:
Hoang Minh E
Search users

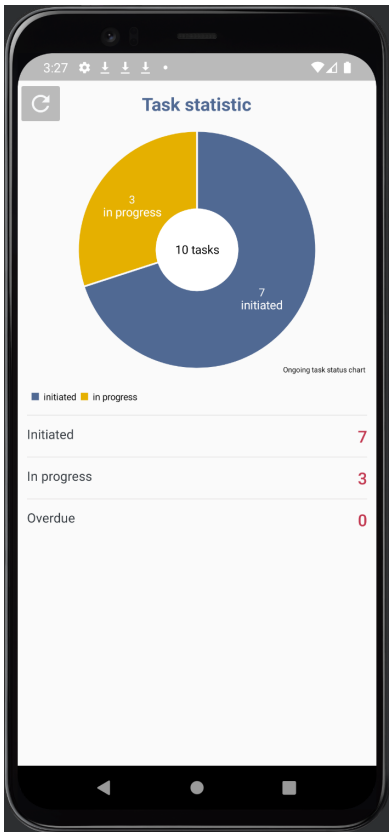

Project Eureka gateway/ Design authentication filter
Info
Attachments
Checklist
initiated
Name
Design authentication filter
Description
Design a filter that can mutate incoming request to include a header that contains authenticated user's id
Date created
04-05-2023
Due date
24-05-2023
Created by
Hoang Minh E
Assigned users:
Nguyen Hoang Minh I
Search users

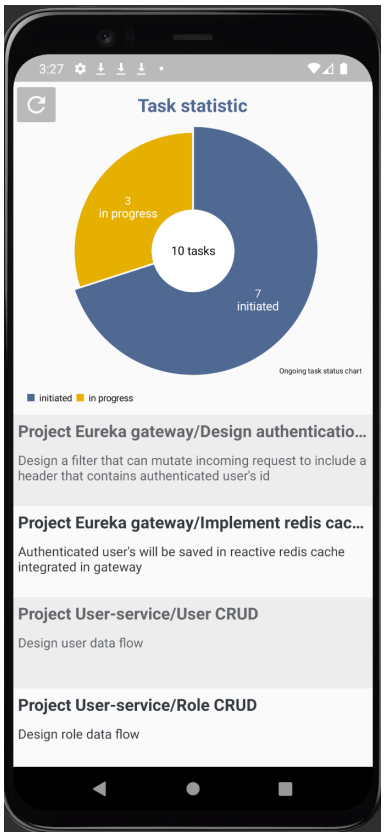

Project Eureka gateway/
Design authentication filter
Info
Attachments
Checklist
Assigned users:
Nguyen Hoang Minh I
Search users
fecu
Truong Duc Duong
Super admin, Admin, Observer, Employee, Project manager
Phan Van A
Admin
Nguyen Thi B
Observer
Pham Duc C
Employee
Nguyen Van F
Observer
Truong Quy G


Project Eureka gateway/
Design authentication filter
Info
Attachments
Checklist
thumb-1920-690653.png
Hanoi-to-Halong-Bay-transport-guide-2023-new-DP-Junk-Boat.jpg
images_20.jpeg
Paris_1.jpeg


Project Eureka gateway/
Design authentication filter
Info
Attachments
Checklist
New item
check item 5
check item 4
check item 3
check item 2
check item 1


Project Eureka gateway/
Design authentication filter
Info
Attachments
Checklist
Hello
What r u doing?
I need this task done by the next weekend
message


Task statistic
3
in progress
10 tasks
7
initiated
Ongoing task status chart
initiated
in progress
Initiated 7
In progress 3
Overdue 0


Task statistic
3
in progress
10 tasks
7
initiated
Ongoing task status chart
initiated
in progress
Project Eureka gateway/Design authenticatio...
Design a filter that can mutate incoming request to include a header that contains authenticated user's id
Project Eureka gateway/Implement redis cac...
Authenticated user's will be saved in reactive redis cache integrated in gateway
Project User-service/User CRUD
Design user data flow
Project User-service/Role CRUD
Design role data flow

7. Coding Convention

- JAVA coding convention

8. User Guide

- Download and install Docker:

https://www.docker.com/products/docker-desktop/

- Pull MySql to Docker:

```
sudo docker pull mysql/mysql-server:latest
```

- Pull RabbitMQ to Docker:

```
docker run -it --name rabbitmq -p 5672:5672 -p
15672:15672 rabbitmq:3.11-management
```

- Pull Redis to Docker:

```
docker run -d --name redis-stack-server -p 6379:6379
redis/redis-stack-server:latest
```

- Pull MongoDB to Docker:

```
docker pull mongo:latest
```

- Run those 4 images on docker in default settings
- Run Eureka-server first, then run Auth-gateway, User-service, Task-service and File-service at the same time (using IntelliJ IDEA)
- Run the frontend on an Android studio's emulator or install it on an actual Android smartphone