System documentation for queueing_app

Kristoffer Svedal
For administrators

https://github.com/ksvedal/queueing_app.git

Table of Contents

Quick setup and ports:

Most info about quick setup in README, but to quickly initialize project, install npm, enter the client folder and use the commands “npm build” -> “npm run serve”.
Ports used:

BACKEND: localhost:**42069**
CLIENT: localhost:**8080**

Users

This application uses basic auth to communicate with the API.

Users does not yet have a connection to database.
Therefore, admins must use mock users for testing and navigating the application:

Student:
Username: “bob”
Password: “bob”

Administrator:
Username: “linda”
Password: “linda”

NOTE: Application behaves differently when different users are logged in.
These users can be used to navigate the application and works without setting up database.
Some API-calls will not work as intended without the proper database setup.

Testing

There has not been set up many tests yet, but they will come.

Backend

Junit testing.

Frontend

Jest testing.

To run all tests:

```
npm run test:unit
```

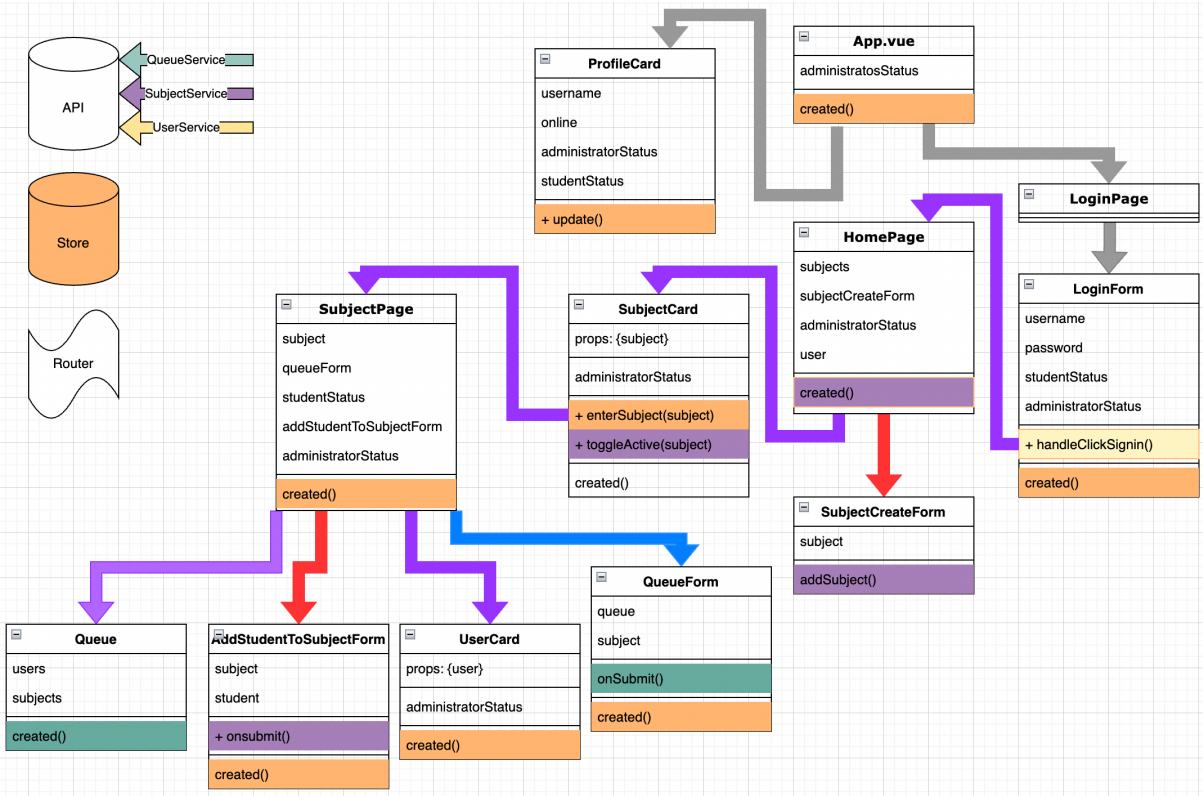
Client architecture

Shown under is a map of the client, its services, its components and where they get routed to.

API and store calls are colour coded by the faded colours green, purple, yellow and orange. The methods of the components are coloured by the services or store they access.

The fat brightly coloured arrows display what components/routes the user gets shown/routed to by what privileges they possess:

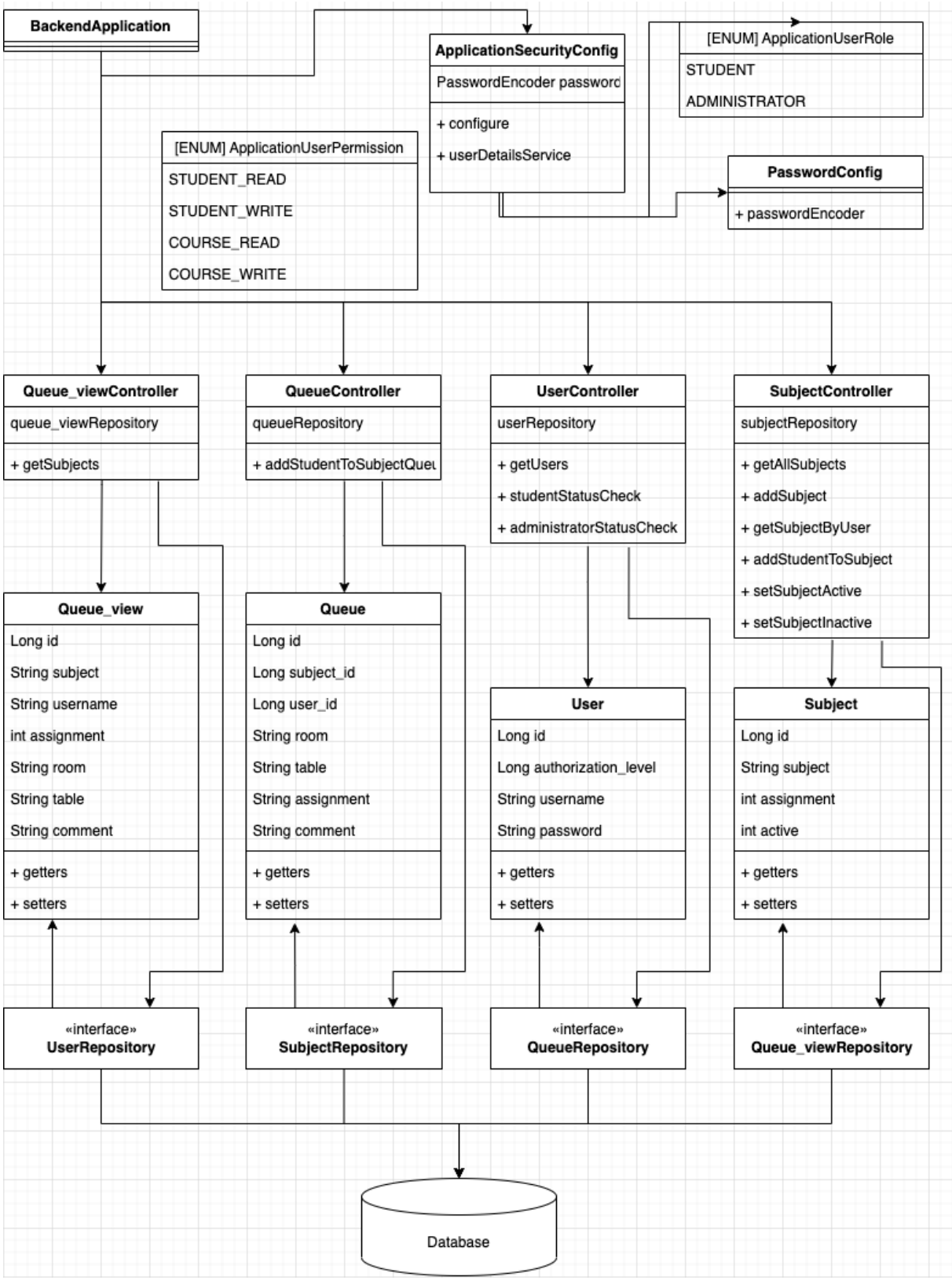
ADMINISTRATOR
STUDENT
BOTH
NOT LOGGED IN/ALWAYS VISIBLE



Backend architecture

Class diagram

Class diagram of spring backend:



Rest Mappings

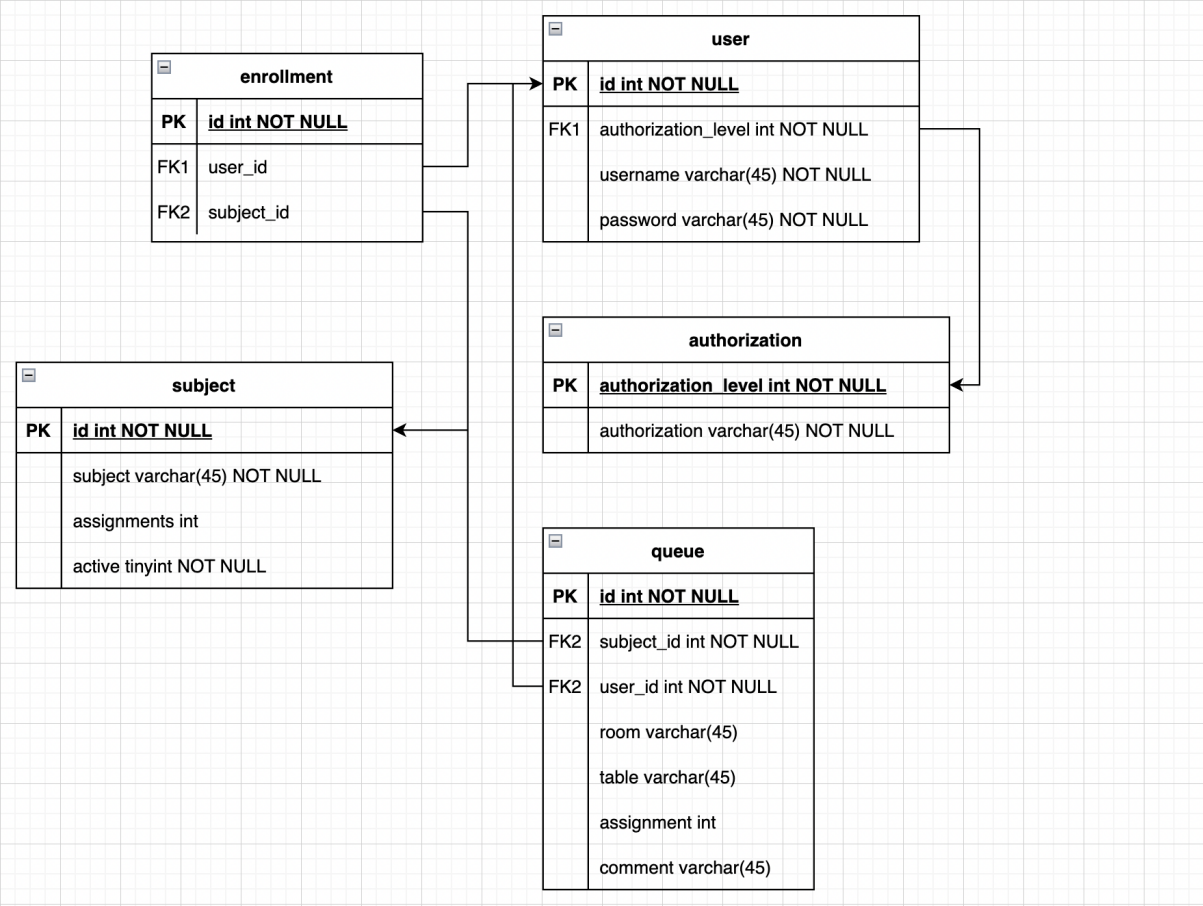
The rest mappings are documented using Postman:
https://documenter.getpostman.com/view/20349392/UVysywYu

Overview of endpoints:

- queueing_app
  - user
    - GET localhost:42069/administrator/status/check
    - GET localhost:42069/student/status/check
    - GET localhost:42069/administrator/user
  - queue
    - GET localhost:42069/queue/{subject}
    - POST localhost:42069/queue/addStudentToSubjectQueue
  - subject
    - GET localhost:42069/administrator/subject
    - GET localhost:42069/subject/bob
    - POST localhost:42069/administrator/subject/add/{subject}
    - POST localhost:42069/administrator/subject/addStudent
    - POST localhost:42069/administrator/subject/status/active
    - POST localhost:42069/administrator/subject/status/inact...

Database architecture

Architecture of database as visualized below:



There's also a view used by the queue portion of the app formatted like this

```
CREATE VIEW `queueing_app`.`queue_view` AS
  SELECT
    `queueing_app`.`queue`.`id` AS `id`,
    `queueing_app`.`subject`.`subject` AS `subject`,
    `queueing_app`.`user`.`username` AS `username`,
    `queueing_app`.`queue`.`assignment` AS `assignment`,
    `queueing_app`.`queue`.`room` AS `room`,
    `queueing_app`.`queue`.`table` AS `table`,
    `queueing_app`.`queue`.`comment` AS `comment`
  FROM
    ((`queueing_app`.`queue`
    LEFT JOIN `queueing_app`.`subject` ON ((`queueing_app`.`queue`.`subject_id` =
`queueing_app`.`subject`.`id`)))
    LEFT JOIN `queueing_app`.`user` ON ((`queueing_app`.`queue`.`user_id` =
`queueing_app`.`user`.`id`)))
```