Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №8

Вариант №8013

Выполнил

Макогон Ярослав Вадимович

Номер группы: P3118

Проверил

Карташев Владимир Сергеевич

г. Санкт-Петербург 2025г.

Содержание

Задание

* Изменить серверное приложение так, чтобы оно работало на Spring + Hibernate.
* Написать Frontend часть приложения на React с использованием typescript. Можно использовать любую библиотеку компонентов, чтобы не писать все с нуля. Я советую использовать shadcn/ui (красивее) или rsuite (легче).
* (опционально, можно не делать в принципе) Desktop приложение на JavaFX / Swing на выбор с отправкой http запросов.

Бекенд (серверное приложение)
эндпоинты:
* авторизация: /api/auth/login
* регистрация: /api/auth/registration
* назначением юзера администратором: /api/admin/appoint/{userId}
* снятие статуса администратора с юзера: /api/admin/appoint/{userId}
* блокировка аккаунта администратором: /api/admin/block/{userId}
* созданием объекта и CRUD операций с ним: /api/<object-name>
* экспортированием коллекций пользователя в формате CSV: /api/export
* импортированием коллекций пользователя в формате CSV: /api/import

Если пользователь администратор, то он может удалять и изменять чужие объекты, а также блокировать пользователей (устанавливается status: BLOCKED).

Если пользователь супер администратор, то он имеет доступ ко всем действиям обычных администраторов, а также может назначать пользователей админами и убирать их статус (администраторы ниже по рангу чем супер администраторы).

Аккаунт суперадминистратора должен создаваться при первом запуске приложения и выводить сгенерированный пароль.

Фронтенд
Компоненты:

- Header
  - Логотип, при нажатии на который переходим на главное меню
  - Кнопка, ведущая на панель администратора (если юзер администратор)
  - Кнопки “Sign in”, “Sign up” - если не авторизованы
  - Кнопка с открытием User info, если авторизованы
- Content
  - * Кнопки “Add”, “Import”, “Export”
  - * Поисковая строка для поиска объекта
  - * Таблица объектов с сортировкой по основным полям коллекции (фильтрация должна быть осуществлена на уровне Backend-приложения через передачу параметров: максимальное количество объектов на странице, страница, фильтры для сортировки (ASC, DESC)); должна быть возможность удаления объекта, если выступаем в роли админа
- Admin panel
  - Поисковая строка, где можно найти пользователя и получить User info пользователя
  - Таблица администраторов, ведущая на User info
- User info
  - Имя, фамилия пользователя, email, его аватар (использовать Minio для хранения файлов)
  - Если просматриваем User info человека с аккаунта администрации - должны быть кнопки “Block”, “Assign as administrator”

Безопасность
Использовать JWT Auth с информацией о юзере: email, firstName, secondName, role and others. Сохранять в localstore браузера. Если приложение кидает Unauthorize - выходим из аккаунта на фронтенде.

Результат

Код на GitLab: https://gitlab.se.ifmo.ru/Not-N0w/ticketmanager

(я еще планирую доделать пару вещей здесь в ближайшее время: поправить архитектуру, разобраться как грамотно поддерживать контекст в nextJS, поиграться с JWT еще нужно)

Использовал

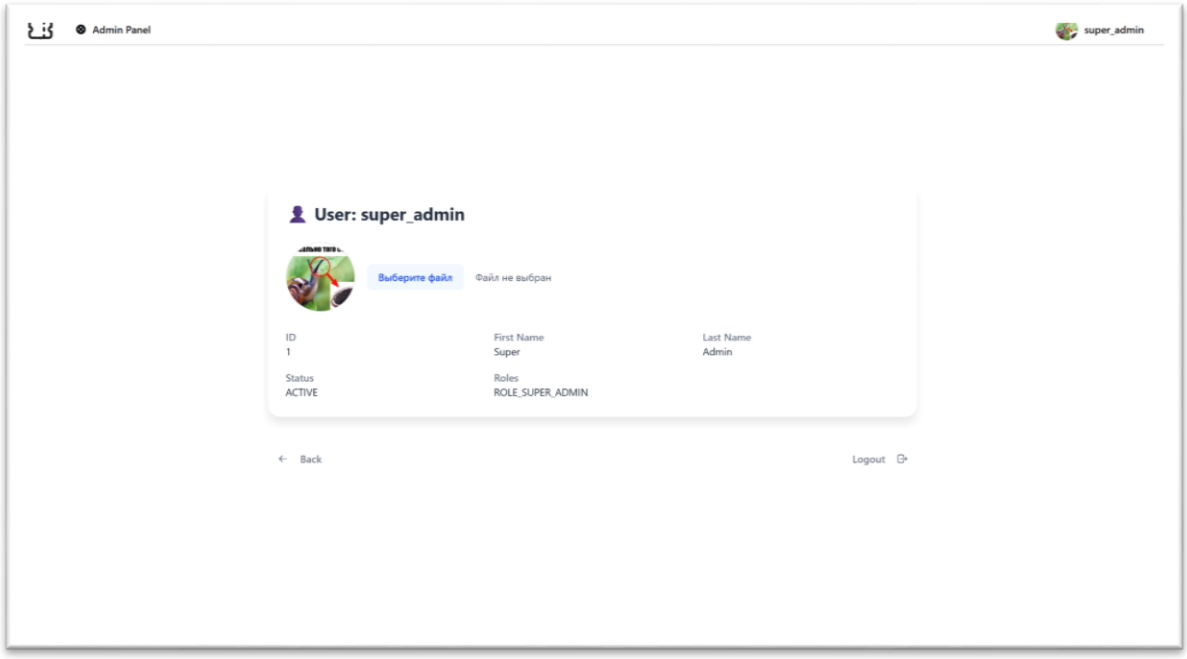
Backend:

- Spring Boot
- Hibernate
- Lombok
- MapStruct
- Minio
- Docker
- DB: PostgreSQL

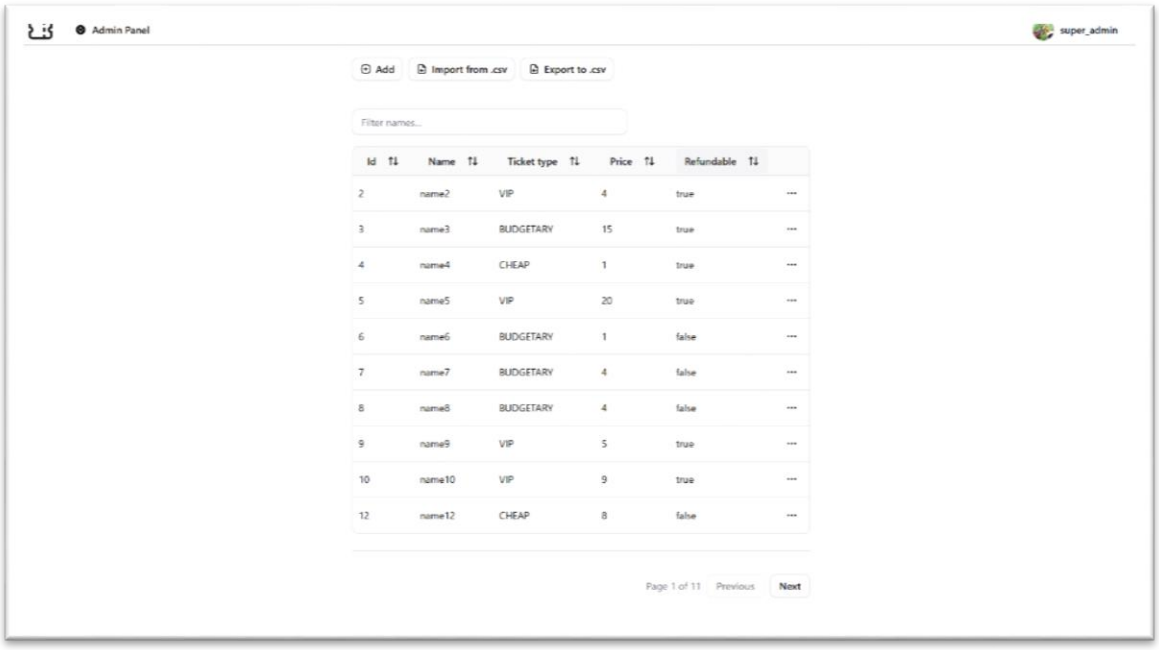
Frontend:

- NextJS
- Shadcn/ui: tilewind + tabler + radix

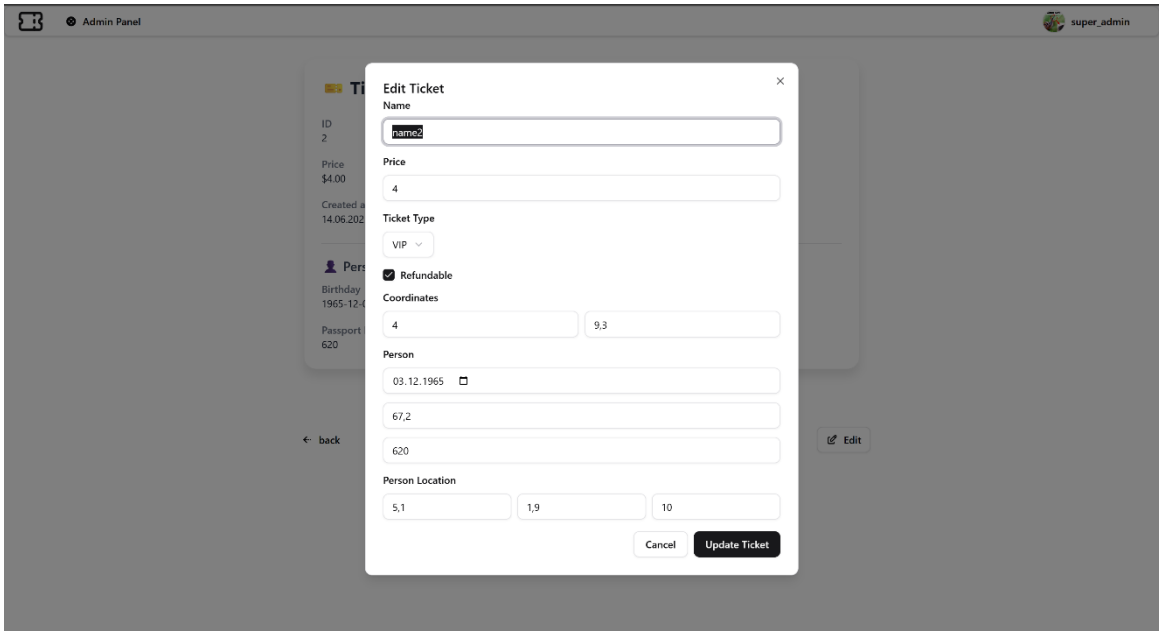
Несколько скриншотов



Страница текущего пользователя '/user/me'



Страница с билетами '/tickets'



Страница билета в режиме редактирования '/tickets/{id}'

Описание функционала:

- Вход в аккаунт/регистрация пользователя (JWT Auth)
- Блокировка/Разблокировка аккаунта старшим по роли юзером
- Назначение админом (может только супер-юзер)
- Удаление юзеров и просмотр информации о них на админской панели
- Изменение своей аватарки (картинка)
- Просмотр списка билетов
- Просмотр подробной информации о билете
- Редактирование своих билетов и билетов младших по роли юзеров (форма)
- Создание билетов (форма)
- Импорт и экспорт билетов в .csv

Docker:

Dockerfile.client:

```
FROM node:18
WORKDIR /app
COPY ../frontend/ticket-manager/package*.json ./
COPY ../frontend/ticket-manager/ .
RUN npm install
RUN npm run build
EXPOSE 3000
CMD ["npm", "start"]
```

Dockerfile.server:

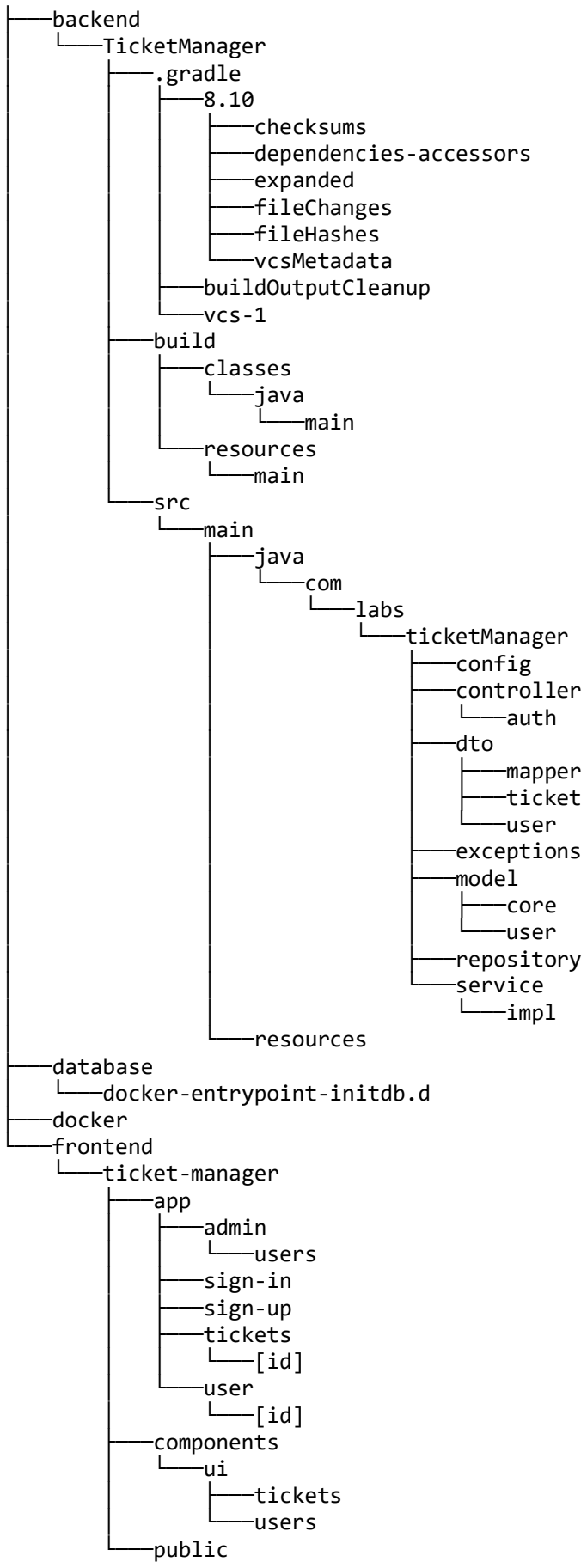
```
FROM eclipse-temurin:17-jdk
WORKDIR /app
COPY ../backend/TicketManager/build/libs/*.jar server.jar
EXPOSE 1804
ENTRYPOINT ["java", "-jar", "server.jar"]
```

Docker-compose.yml:

```
services:
  server:
    depends_on:
      - postgres
      - minio
    build:
      context: .
      dockerfile: docker/Dockerfile.server
    container_name: server-container
    ports:
      - "1805:1805"
    healthcheck:
      test: [ "CMD", "pgrep", "java" ]
      interval: 10s
      retries: 3
```

```
    tty: true

  minio:
    image: minio/minio:latest
    container_name: minio
    environment:
      - MINIO_ROOT_USER=minio_admin
      - MINIO_ROOT_PASSWORD=minio_admin
    command: server /data --console-address :9090
    ports:
      - '9090:9090'
      - '9000:9000'
    volumes:
      - minio-data:/data

  minio-init:
    image: minio/mc
    depends_on:
      - minio
    entrypoint: >
      /bin/sh -c "
      sleep 5;
      mc alias set local http://minio:9000 minio_admin minio_admin;
      mc mb -p local/images;
      mc anonymous set download local/images;
      "
    restart: on-failure

  client:
    build:
      context: .
      dockerfile: docker/Dockerfile.client
    container_name: client-container
    ports:
      - "3000:3000"
    tty: true

  postgres:
    image: postgres
    environment:
      POSTGRES_DB: "ticketManagerDB"
      POSTGRES_USER: "postgres_admin"
      POSTGRES_PASSWORD: "postgres"
    volumes:
      - ./database/docker-entrypoint-initdb.d:/docker-entrypoint-initdb.d
    ports:
      - "5432:5432"
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres_admin -d ticketManagerDB"]
      interval: 10s
      retries: 5
      start_period: 30s
      timeout: 10s
volumes:
  minio-data:
    driver: local
```

Структура проекта



Выводы

- Фронтенд… тяжело... Так я и не смог нормально подружиться с JS😒 Но продвижения явно имеются
- Попробовал NextJS
- Изучил Spring Boot
- Научился работать с Hibernate
- Lombok, MapStruct
- Попробовал Minio
- Изучил JWT Auth в связке со Spring