САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 1. “Написание Boilerplate для разработки сервиса с использованием TS, Sequelize и Express”

Выполнил:

Тимофеев Николай

Группа

K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

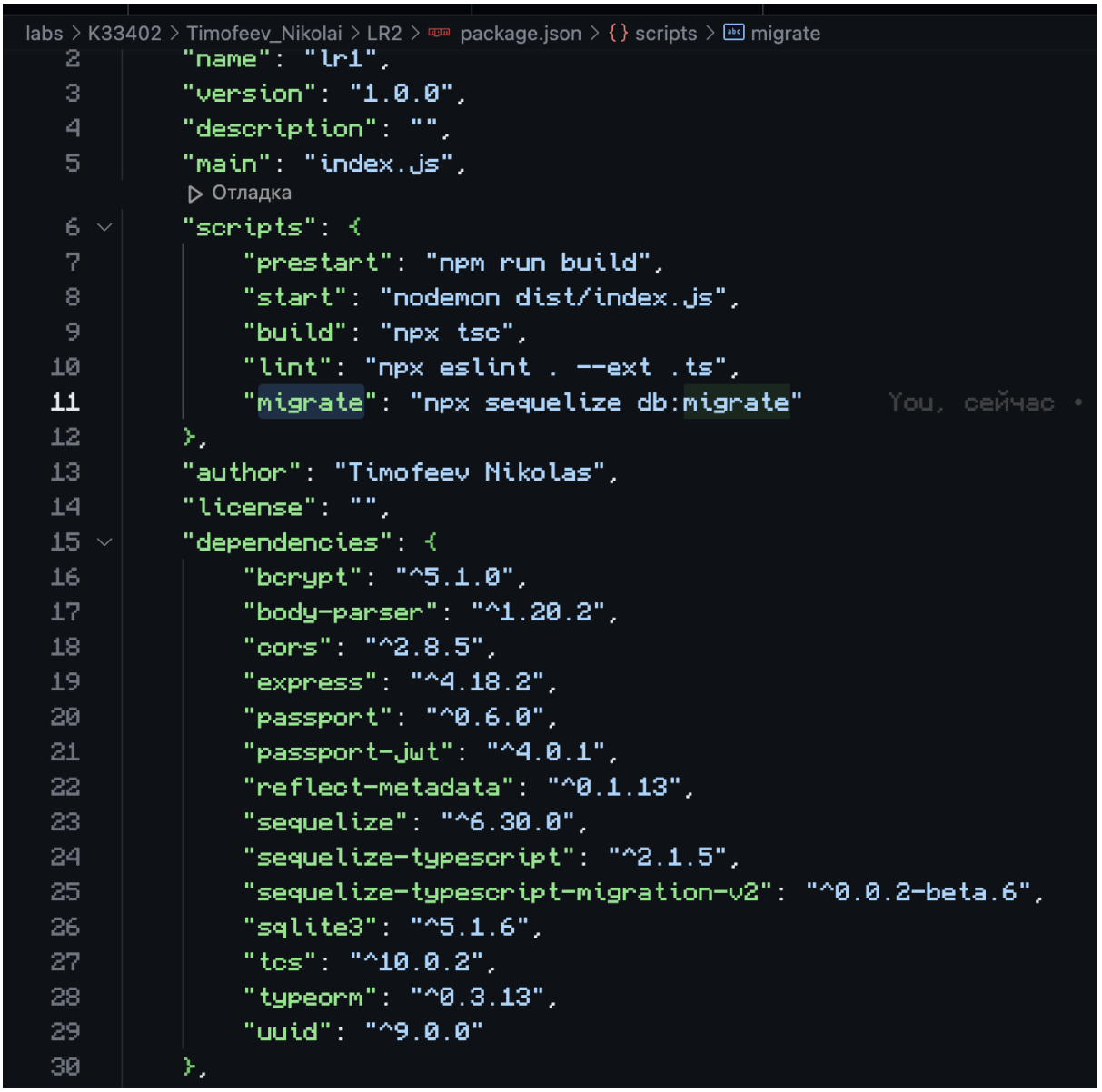
Задача
Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

Должно быть явное разделение на:

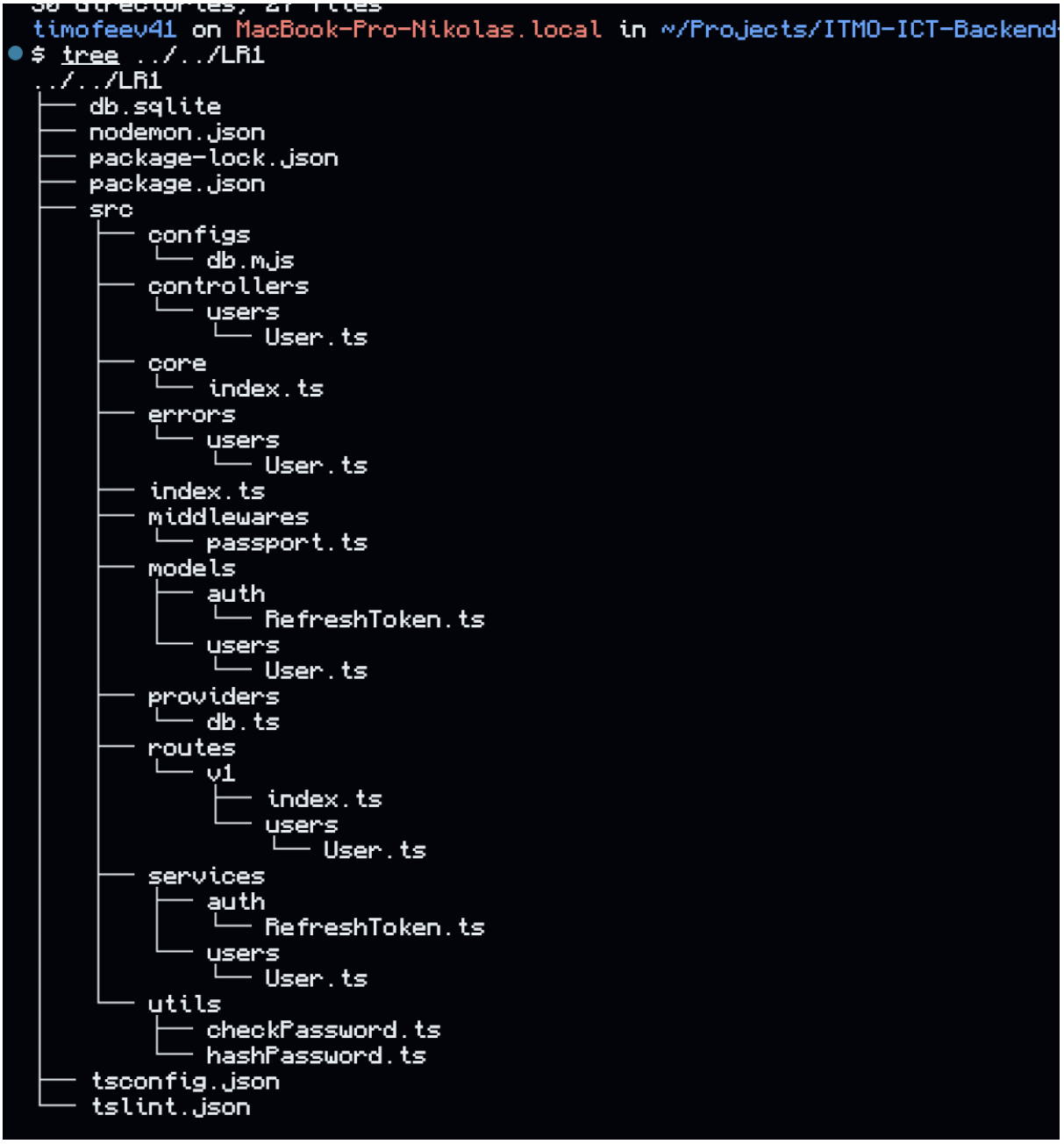
- модели
- контроллеры
- роуты
- сервисы для работы с моделями (реализуем паттерн “репозиторий”)

Ход работы

1. Создадим npm package



2. Структура проекта



Корень исходных файлов **src** включает в себя следующие директории:

- **core** – точка входа в приложение, объединяющая в себе все составляющие;
- **configs** – файлы конфигурации (файл для подключения к БД);
- **controllers** – контроллеры, отвечающие за логику обработки http-запросов;
- **models** – модели sequelize;
- **providers** – точки доступа к данным;
- **routes** – описание маршрутов express;
- **services** – службы, которые содержат запросы к базе данных и возвращают объекты или выдают ошибки;
- **utils** – вспомогательные файлы, которые используются во всем приложении
- **middlewares -** содержит аутентификацию с использованием passport.js.

3. Модели

Token:

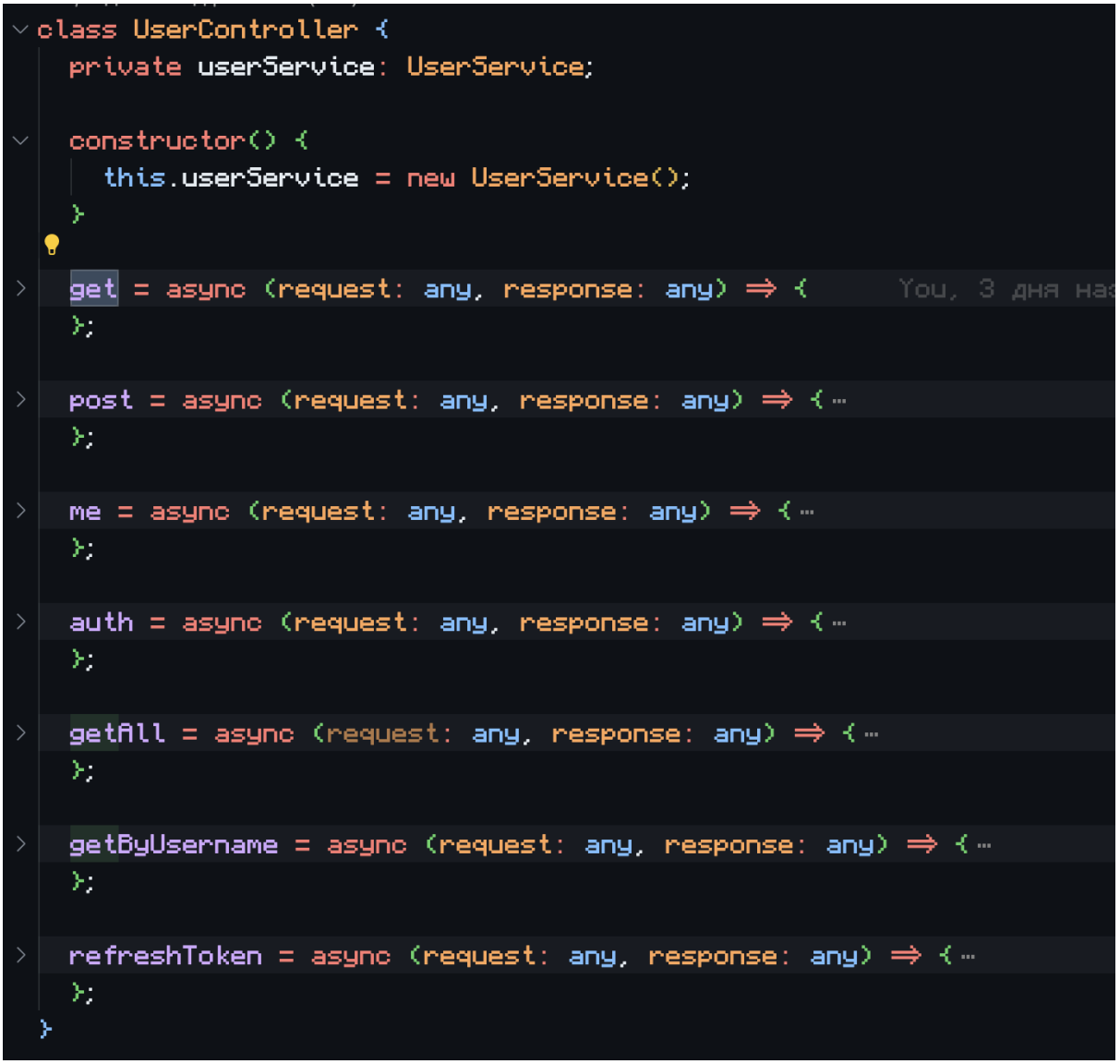
```
You, 3 дня назад | 1 author (You)
import { Table, Column, Model, Unique, AllowNull, ForeignKey } from 'sequelize-typescript'
import User from '../users/User'

You, 3 дня назад | 1 author (You)
@Table
class RefreshToken extends Model {
    @Unique
    @AllowNull(false)
    @Column
    token: string

    @ForeignKey(() => User)
    @Column
    userId: number
}

export default RefreshToken
```

User:


You, 3 дня назад | 1 author (You)
@Table
class User extends Model {
@Unique
@Column
username: string
@Column
firstName: string
@Column
lastName: string
@Unique
@Column
email: string
@AllowNull(false)
@Column
password: string
You, 3 дня назад • feat: lr1 added …

4. Контроллеры

UserController:



Методы класса UserController:

- get: находит пользователя по id;
- post: создание нового пользователя;
- me: возвращает данные о пользователе, полученные из объекта запроса;
- auth: генерирует новый токен доступа и токен обновления, если пользователь залогинился;
- refreshToken: генерирует новый JWT токен;
- getAll: получает информацию по всем пользователям
- getByUsername: находит пользователя

5. Сервисы

UserService - сервис для управления пользователями

6. Роуты

User.ts - авторизация и работа с юзерами

```
router.route("/reg").post(controller.post);      You, 3 дня назад • feat: lr1 added …
router.route("/account").get(passport.authenticate("jwt", { session: false }), controller.me)
router.route("/account/:id").get(controller.get);
router.route("/login").post(controller.auth);
router.route("/refresh").post(controller.refreshToken);
router.route("/accounts").get(controller.getAll);
router.route("/accounts/:username").get(controller.getByUsername);
```

Вывод

В ходе первой лабораторной работы был разработан boilerplate с продуманной структурой и необходимыми конфигурационными файлами с помощью следующих инструментов: express, sequelize/TypeORM, typescript.