САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа 2. “Знакомство с микрофреймворком Express. Знакомство с ORM Sequelize”

Выполнил:

Тимофеев Николай

Группа

K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

- Продумать свою собственную модель пользователя
- Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize
- Написать запрос для получения пользователя по id/email

Ход работы

1) Создадим модель пользователя с полями username, firstName, lastName и email:

```
timofeev41 on MacBook-Pro-Nikolas.local in ~/Projects/ITMO-ICT-Backend-2023/homeworks/K33402/Timofeev_Nikolai/HW2(5d6h7m|hw12*)
$ npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string,username:string
```

Timofeev_Nikolai > HW2 > models > JS user.js > <unknown> > exports

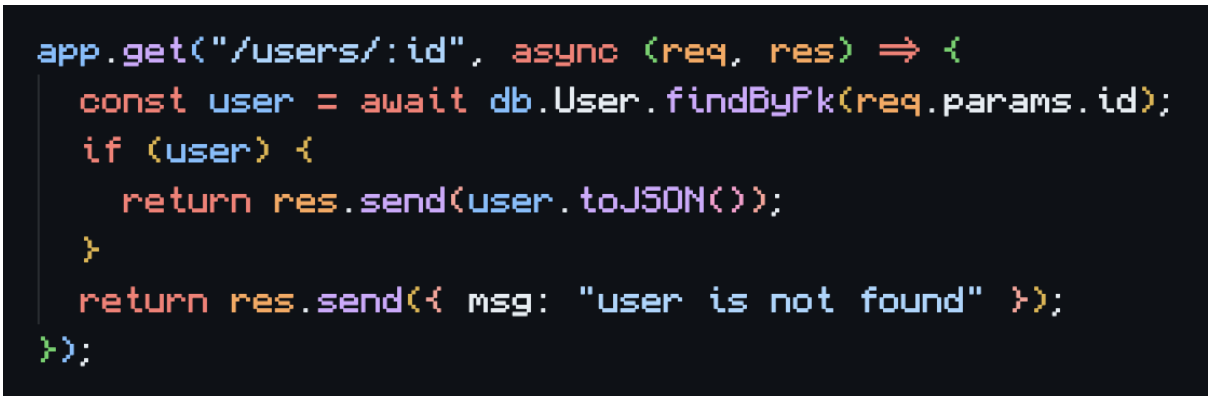
```
1   'use strict';
2 > const {…
4   } = require('sequelize');
5   module.exports = (sequelize, DataTypes) => {
6 >   class User extends Model {…
15    }
16    User.init({
17      firstName: DataTypes.STRING,
18      lastName: DataTypes.STRING,
19      email: DataTypes.STRING,
20      username: DataTypes.STRING
21    }, {
22      sequelize,
23      modelName: 'User',
24    });
25    return User;
26  };
```

2) Реализуем CRUD REST API Endpoints:

Просмотр списка всех пользователей:

```
app.get("/users", async (req, res) => {
    const users = await db.User.findAll({});
    return res.json({users});
})
```

Просмотр данных о конкретном пользователе по его id



```
app.get("/users/:id", async (req, res) => {
  const user = await db.User.findByPk(req.params.id);
  if (user) {
    return res.send(user.toJSON());
  }
  return res.send({ msg: "user is not found" });
});
```

Создание нового юзера. Для параметров передаваемых в теле запроса (body) была подключена валидация с использованием модуля Joi

```
const userCreateSchema = Joi.object({
  username: Joi.string().required(),
  firstName: Joi.string().required(),
  lastName: Joi.string().required(),
  email: Joi.string().email().required(),
});

app.post("/users", upload.array(), async (req, res, next) => {
  const { body } = req;
  const { value, error } = userCreateSchema.validate(body);
  if (error) {
    res.status(422).json({
      message: "Invalid request",
      data: body,
      error: error.details,
    });
  } else {
    await db.User.create(value);
    res.json({ message: "Resource created", data: value });
  }
});
```

Удаление пользователя по его id:

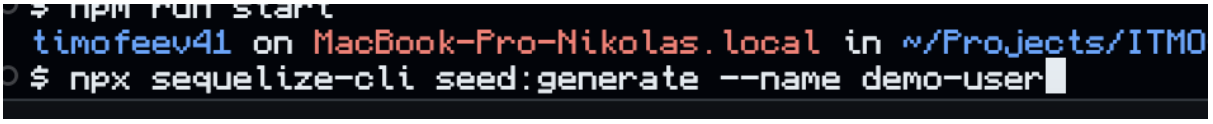
```
app.delete('/users/:id', async (req, res) => {
    const user = await db.User.findByPk(req.params.id);
    if (!user) {
        res.status(404).json({
            message: `User with id ${req.params.id} not found`
        })
        return;
    }
    await user.destroy();
    res.json({ message: "deleted ok", data: user });
})
```

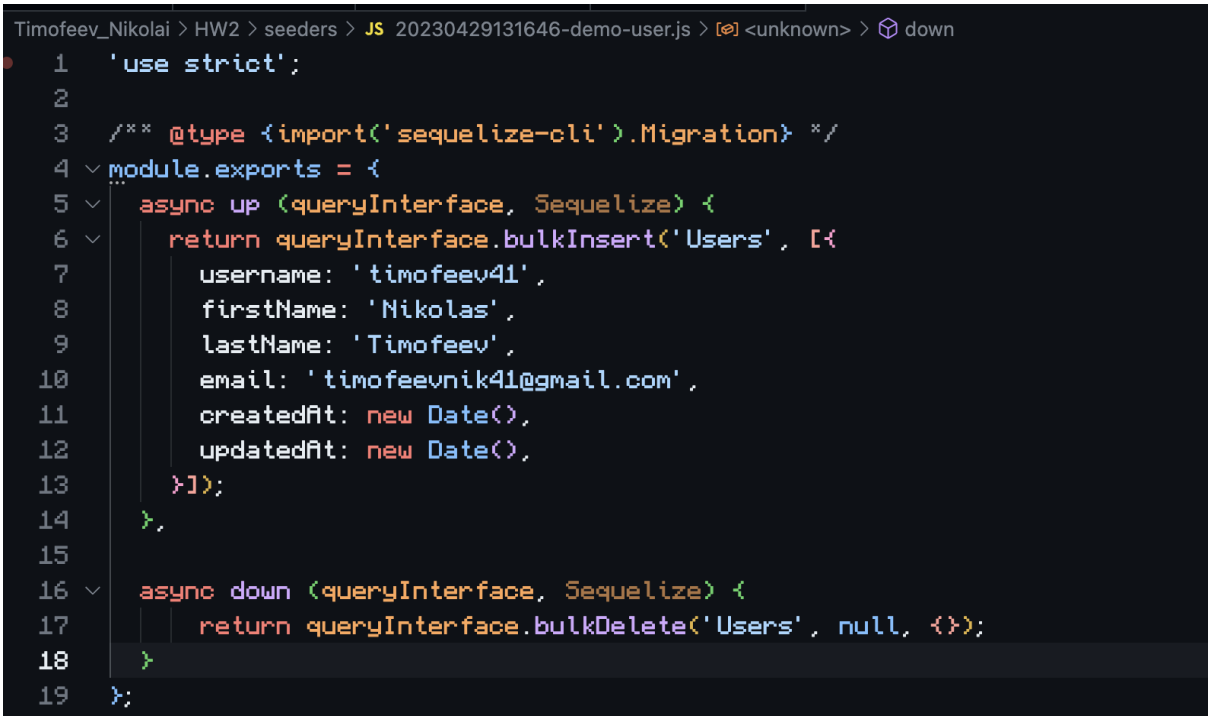
3) Написать запрос для получения пользователя по id/email

Для получения пользователя по id я воспользовался методом findByPk объекта User. Запрос для получения пользователя по email выглядит так:

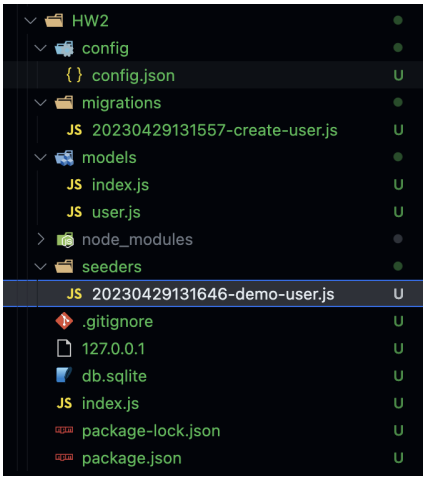
const user = await db.User.findOne({email: req.params.email});

Дополнительно я воспользовался функционалом seeders для заполнения таблицы данными перед запуском приложения:





Итоговая структура проекта:



Вывод

В ходе выполнения работы я изучил возможности ORM Sequelize и микрофреймворка ExpressJS для построения Backend-части веб-сервиса. Воспользовался sequelize-cli, npm и валидатором Joi.