

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:  
Кулёмин С. А.  
Группа К33401

Проверил:  
Добряков Д. И.

Санкт-Петербург  
2022 г.

## Задание

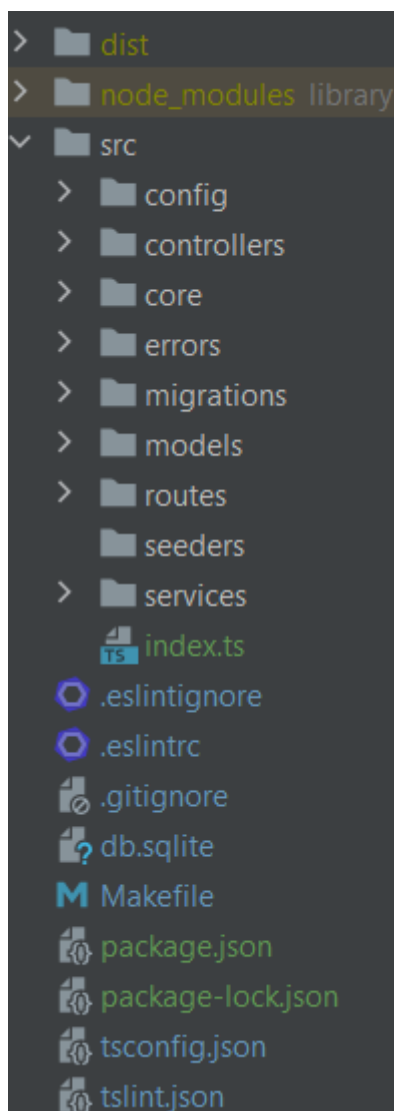
Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

Должно быть явное разделение на:

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

## Выполнение

Структура boilerplate:



Работа с БД осуществлялась при помощи ORM-библиотеки Sequelize  
src/models/index.ts

```
import { Sequelize } from 'sequelize';

const env = process.env.NODE_ENV || 'development';
const config = require(__dirname + '/../config/config.js')[env];

const sequelize = config.url
  ? new Sequelize(config.url, config)
  : new Sequelize(config.database, config.username, config.password, config);

export { Sequelize, sequelize };
```

src/config/config.ts

```
module.exports = {
  "development": {
    "username": null,
    "password": null,
    "database": "database_development",
    "host": null,
    "dialect": "sqlite",
    "storage": "db.sqlite"
  },
  "test": {
    "username": "root",
    "password": null,
    "database": "database_test",
    "host": "127.0.0.1",
    "dialect": "sqlite"
  },
  "production": {
    "username": "root",
    "password": null,
    "database": "database_production",
    "host": "127.0.0.1",
    "dialect": "sqlite"
  }
}
```

Создана модель User и таблица Users

src/models/user.ts

```
interface UserAttributes {
  username: string;
  password: string;
  email: string;
  hometown: string;
}

interface UserCreationAttributes
  extends Optional<UserAttributes, 'email'> { }

interface UserInstance extends Model<UserAttributes, UserCreationAttributes>, UserAttributes { ... }

const User = sequelize.define<UserInstance>(<
  modelName: 'User',
  attributes: {
    username: {
      type: DataTypes.STRING,
    },
    password: {
      type: DataTypes.STRING,
    },
    email: {
      type: DataTypes.STRING,
    },
    hometown: {
      type: DataTypes.STRING,
    }
  }
);
```

src/migrations/20220416121945-create-user.js

```
'use strict';
module.exports = {
  async up(queryInterface, Sequelize) {
    await queryInterface.createTable('Users', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      username: {type: Sequelize.STRING...},
      password: {type: Sequelize.STRING...},
      email: {type: Sequelize.STRING...},
      hometown: {type: Sequelize.STRING...},
      createdAt: {type: Sequelize.DATE...},
      updatedAt: {type: Sequelize.DATE...}
    });
  },
  async down(queryInterface, Sequelize) {
    await queryInterface.dropTable('Users');
  }
};
```

src/index.ts

```
import App from "../core/index"

const app = new App()

app.start()
```

src/core/index.ts

```
class App {
  public port: number
  public host: string
  private app: express.Application
  private server: Server
  constructor(port: number = 8000, host: string = "localhost") {
    this.port = port
    this.host = host

    this.app = this.createApp()
    this.server = this.createServer()
  }
  private createApp(): express.Application {
    const app = express()
    app.use(bodyParser.json())
    app.use('/v1', routes)

    return app
  }
  private createServer(): Server {
    const server = createServer(this.app)

    return server
  }

  public start(): void {
    this.server.listen(this.port, listeningListener: () => {
      console.log(`Running server on port ${this.port}`)
    })
  }
}
```

src/routes/v1/index.ts

```
import express from "express"
import userRoutes from "../users/index"

const router: express.Router = express.Router()

router.use('/users', userRoutes)

export default router
```

src/routes/v1/users/index.ts

```
import express from "express";
import UserController from "../../../controllers/users/index";

const router: express.Router = express.Router()

const controller: UserController = new UserController()

router.route( prefix: '/add/')
  .post(controller.post)

router.route( prefix: '/profiles')
  .get(controller.getAll)

router.route( prefix: '/profile/id/:id')
  .get(controller.getById)

|
router.route( prefix: '/profile/username/:username')
  .get(controller.getByUsername)

export default router
```

src/controllers/users/index.ts

```
import UserService from "../../services/users/index";

class UserController {
  private userService = new UserService

  constructor() {
    this.userService = new UserService()
  }

  post = async (request: any, response: any) => {...}

  getAll = async (request: any, response: any) => {...}

  getById = async (request: any, response: any) => {...}

  getByUsername = async (request: any, response: any) => {
    try {
      const user = await this.userService.getByUsername(
        request.params.username
      )

      response.send(user)
    } catch (error: any) {
      response.status(404).send({ "error": error.message })
    }
  }
}

export default UserController
```

src/services/users/index.ts

```
import UserError from "../../errors/users/index";
import User from "../../models/user"

class UserService {

  async create(userData: any) {...}

  async getAll() {...}

  async getById(id: number) {...}

  async getByUsername(username: string) {
    const user = await User.findOne({
      where: {
        username: username
      }
    })

    if (user) return user.toJSON()

    throw new UserError('User with this username not found')
  }
}

export default UserService
```



## Проверка работы

### Добавление пользователя



POST localhost:8000/v1/users/add/ [Send](#)

Params ☒ Auth Headers (8) Body ☒ Pre-req. Tests Settings [Cookies](#)

raw ☒ JSON [Beautify](#)

```
1 {
2   "username": "Simon",
3   "password": "password",
4   "email": "skulemin@icloud.com",
5   "hometown": "Saint-P"
6 }
```

Body ☒ 201 Created 39 ms 396 B [Save Response](#)

Pretty Raw Preview Visualize JSON ☒  

```
1 {
2   "id": 5,
3   "username": "Simon",
4   "password": "password",
5   "email": "skulemin@icloud.com",
6   "hometown": "Saint-P",
7   "updatedAt": "2022-04-16T14:13:58.879Z",
8   "createdAt": "2022-04-16T14:13:58.879Z"
9 }
```

### Получение списка всех пользователей



GET localhost:8000/v1/users/profiles [Send](#)

Params ☒ Auth Headers (8) Body ☒ Pre-req. Tests Settings [Cookies](#)

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body ☒ 200 OK 14 ms 1.05 KB [Save Response](#)

Pretty Raw Preview Visualize JSON ☒  

```
1 {
2   {
3     "id": 1,
4     "username": "test1",
5     "password": "test1",
6     "email": "test1@test.com",
7     "hometown": null,
8     "createdAt": "2022-04-16T13:13:27.559Z",
9     "updatedAt": "2022-04-16T13:13:27.559Z"
10  },
11  {
12    "id": 2,
13    "username": "test2",
14    "password": "test2",
15    "email": "test2@test.com",
16    "hometown": "Saint-P",
17    "createdAt": "2022-04-16T13:14:22.111Z",
18    "updatedAt": "2022-04-16T13:14:22.111Z"
19  },
20 }
```

## Получение пользователя по id

GET localhost:8000/v1/users/profile/id/3 Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
--	-----	-------	-------------	-----	-----------

Body 200 OK 9 ms 383 B Save Response

Pretty Raw Preview Visualize JSON Copy Search

```
1 {  
2   "id": 3,  
3   "username": "test3",  
4   "password": "test3",  
5   "email": "test3@test.com",  
6   "hometown": "Saint-P",  
7   "createdAt": "2022-04-16T13:27:03.562Z",  
8   "updatedAt": "2022-04-16T13:27:03.562Z"  
9 }
```

## Получение пользователя по username

GET localhost:8000/v1/users/profile/username/Simon Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
--	-----	-------	-------------	-----	-----------

Body 200 OK 7 ms 391 B Save Response

Pretty Raw Preview Visualize JSON Copy Search

```
1 {  
2   "id": 5,  
3   "username": "Simon",  
4   "password": "password",  
5   "email": "skulemin@icloud.com",  
6   "hometown": "Saint-P",  
7   "createdAt": "2022-04-16T14:13:58.879Z",  
8   "updatedAt": "2022-04-16T14:13:58.879Z"  
9 }
```

GET localhost:8000/v1/users/profile/username/namethatnotindatabase Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
--	-----	-------	-------------	-----	-----------

Body 404 Not Found 9 ms 264 B Save Response

Pretty Raw Preview Visualize JSON Copy Search

```
1 {  
2   "error": "User with this username not found"  
3 }
```

## **Вывод**

В ходе выполнения работы был написан boilerplate на Express, Sequelize, Typescript. Был настроен доступ к базе данных, реализован паттерн “репозиторий”. Созданный boilerplate может быть использован для разработки полноценного backend-приложения. Разделение кода на роуты, контроллеры, сервисы упростит разработку основного функционала.