САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет

Лабораторная работа 1

Выполнил: Ле Тхи Лан Ань

Группа K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

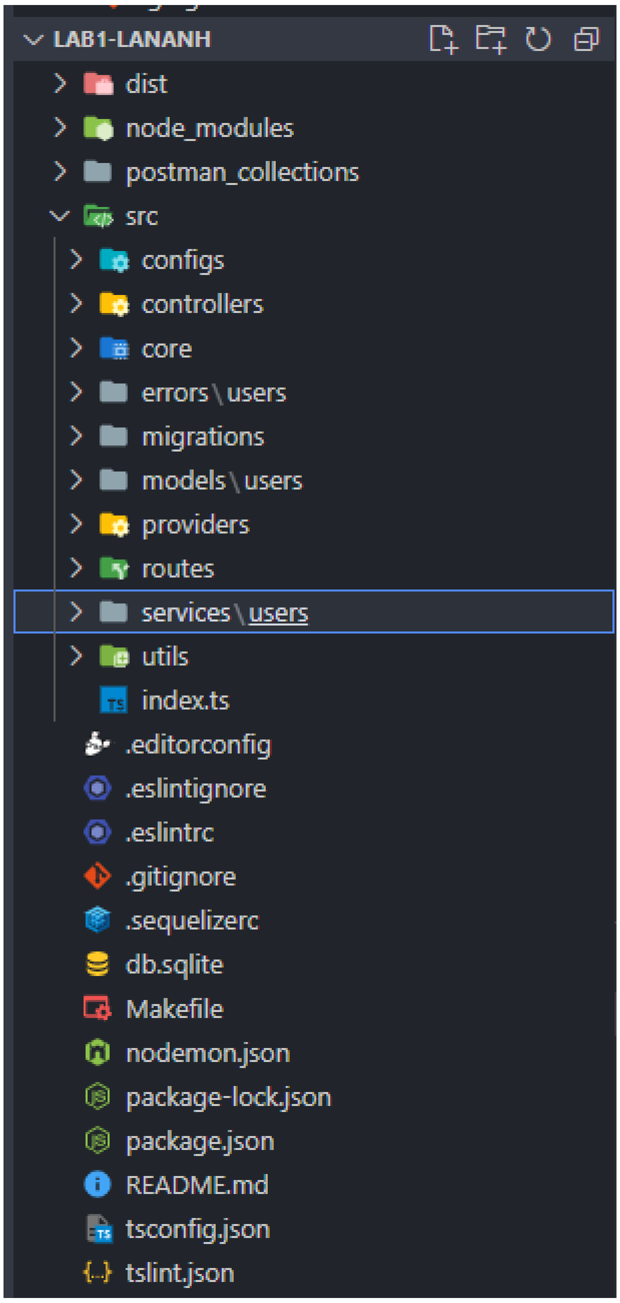
2022 г.

Задача

- Нужно написать свой boilerplate на express + sequelize + typescript.
- Должно быть явное разделение на:
  - модели
  - контроллеры
  - роуты
  - сервисы для работы с моделями (реализуем паттерн “репозиторий”)

Ход работы

1. Tree



```
LAB1-LANANH
  > dist
  > node_modules
  > postman_collections
  v src
    > configs
    > controllers
    > core
    > errors\users
    > migrations
    > models\users
    > providers
    > routes
    > services\users
    > utils
      index.ts
  .editorconfig
  .eslintignore
  .eslintrc
  .gitignore
  .sequelizerc
  db.sqlite
  Makefile
  nodemon.json
  package-lock.json
  package.json
  README.md
  tsconfig.json
  tslint.json
```

2.

3. controllers/index.ts

```
import User from "../../models/users/User";
import UserService from "../../services/users/User";
import UserError from "../../errors/users/User";

class UserController {
    private userService: UserService;

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

    get = async (request: any, response: any) => {
        try {
            const user: User | UserError = await
this.userService.getById(Number(request.params.id));

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

    post = async (request: any, response: any) => {
        const { body } = request;

        try {
            const user: User | UserError = await this.userService.create(body);

            response.status(201).send(user);
        } catch (error: any) {
            response.status(400).send({ error: error.message });
        }
    };

    put = async (request: any, response: any) => {
        const { body } = request;

        try {
            const user: User | UserError = await
this.userService.updateById(Number(request.params.id), body);

            response.status(200).send(user);
        } catch (error: any) {
            response.status(400).send({ error: error.message });
        }
    }

    delete = async (request: any, response: any) => {
        try {
            await this.userService.deleteById(Number(request.params.id));

            response.status(204).send();
        } catch (error: any) {
            response.status(400).send({ error: error.message });
        }
    }
}

export default UserController;
```

4. core/index.ts

```
            import express from "express";
import cors from "cors";
import { createServer, Server } from "http";
import routes from "../routes/index";
import sequelize from "../providers/db";
import { Sequelize } from "sequelize-typescript";
import bodyParser from "body-parser";
import configParser from "../utils/configParser";
import path from "path";

const configPath = path.resolve(__dirname, "../configs/settings.ini");
const config: any = configParser(configPath, "SERVER");

class App {
    public port: number;
    public host: string;

    private app: express.Application;
    private server: Server;
    private sequelize: Sequelize;

    constructor(port = 5994, host = "localhost") {
        this.port = config.port || port;
        this.host = config.host || host;

        this.app = this.createApp();
        this.server = this.createServer();
        this.sequelize = sequelize;
    }

    private createApp(): express.Application {
        const app = express();
        app.use(cors());
        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, () => {
            console.log(`Running server on port ${this.port}`);
        });
    }
}
```

```
export default App;
```

5. models/index.ts

```
import { Table, Column, Model, Unique } from "sequelize-typescript";

@Table
class User extends Model {
    @Column
    name: string;

    @Unique
    @Column
    email: string;

    @Column
    phone: string;

    @Column
    address: string;

    @Column
    age: number;

    @Column
    country: string;
}

export default User;
```

6.routes/index.ts

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

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

const controller: UserController = new UserController();

router.route("/:id").get(controller.get).put(controller.put).delete(controller.d
elete);

router.route("/").post(controller.post);

export default router;
```

```
import express from "express";
import userRoutes from "./users/User";

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

router.use("/users", userRoutes);

export default router;
```

7.services/user.ts

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

class UserService {
    async getById(id: number): Promise<User> {
        const user = await User.findByPk(id);

        if (user) return user.toJSON();

        throw new UserError("Not found!");
    }

    async create(userData: object): Promise<User | UserError> {
        try {
            const user = await User.create(userData);

            return user.toJSON();
        } catch (e: any) {
            const errors = e.errors.map((error: any) => error.message);

            throw new UserError(errors);
        }
    }

    async updateById(id: number, userData: object): Promise<User | UserError> {
        try {
            const user = await User.findByPk(id);

            if (!user) throw new UserError("User not found");

            user.set({ ...userData });

            await user.save();

            return user.toJSON();
        } catch (e: any) {
            const errors = e.errors.map((error: any) => error.message);

            throw new UserError(errors);
        }
    }

    async deleteById(id: number): Promise<void> {
        try {
            const user = await User.findByPk(id);

            if (user) {
                await user.destroy();
            }
        } catch (e: any) {
            const errors = e.errors.map((error: any) => error.message);
            throw new UserError(errors);
        }
    }
}

export default UserService;
```

Вывод

- Написал свой boilerplate на express + sequelize + typescript.
- Были разделены директория:
    - модели
    - контроллеры
    - роуты
    - сервисы для работы с моделями