

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэкенд-энд разработка

Отчет

Лабораторная работа №1

Выполнил:
Золотов Павел

Группа:
К33401

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

1) 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”)

2) Составьте Makefile, который будет автоматизировать ваши рутинные действия, такие как:

- проведение миграций через sequelize;
- запуск приложения;
- установка зависимостей и сборка приложения.

Ход работы

Makefile:

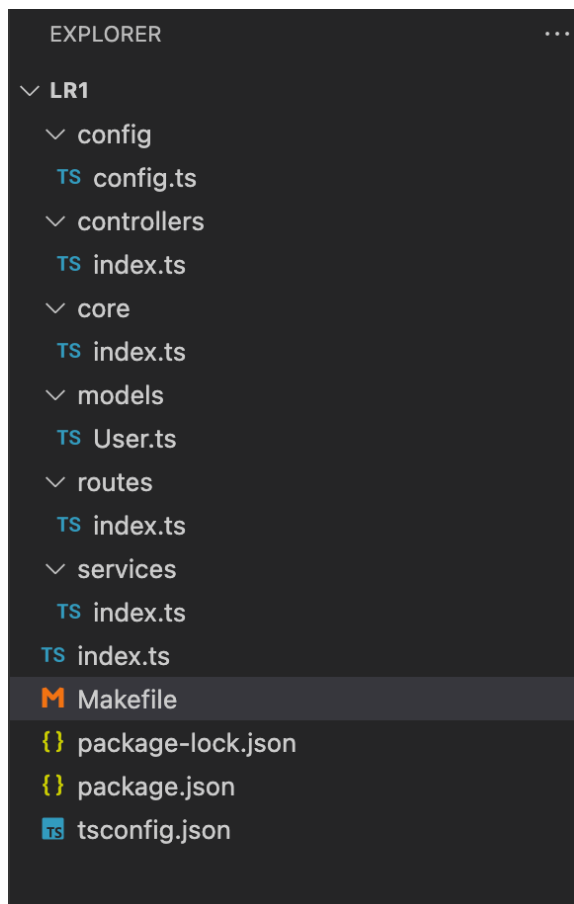
```
.PHONY: init
init:
    npm i
    npm run build

.PHONY: run
run:
    npm start

.PHONY: migrate
migrate:
    npx sequelize-cli db:migrate

.DEFAULT_GOAL := init
```

Структура проекта:



Сервис:

```
import User from '../models/User'
import { sequelize } from '../config/config'

export default class DefaultService {

  private repo = sequelize.getRepository(User)

  add(name: string, surname: string, email: string, address: string)
  {
    this.repo.create({ name: name, surname: surname, email: email,
address: address })
  }

  get() {
    return this.repo.findAll()
  }
}
```

Модель:

```
import { Table, Column, Model } from 'sequelize-typescript'

@Table
export default class User extends Model {
  @Column
  name: string

  @Column
  surname: string

  @Column
  email: string

  @Column
  address: string
}
```

Контроллер:

```
import DefaultService from '../services/index'

export default class ExampleController {

  private service = new DefaultService()

  post = async (request: any, response: any) => {
    try {
      const user = request.body
      await this.service.add(user.name, user.surname, user.email,
user.address)
      response.send('Successfully added')
    } catch (error: any) {
      response.status(400).send(error.message)
    }
  }

  get = async (request: any, response: any) => {
    try {
      const data = await this.service.get()
      response.send(data)
    } catch (error: any) {
      response.status(400).send(error.message)
    }
  }
}
```

Файл для запуска сервера:

```
import express from "express"
import { createServer, Server } from "http"
import routes from "../routes/index"
import { sequelize } from '../config/config'

export default class App {
  public port: number
  public host: string

  private app: express.Application
  private server: Server

  constructor(port = 8000, host = "localhost") {
    this.port = port
    this.host = host

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

  private createApp(): express.Application {
    const app = express()
    const bodyParser = require('body-parser')
    app.use(bodyParser.urlencoded({ extended: false }))
    app.use(bodyParser.json())
    app.use('/api', routes)
    return app
  }

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

  public start(): void {
    sequelize.sync().then(() => {
      console.log('DB connected')
    })
    this.server.listen(this.port, () => {
      console.log(`Running server on port ${this.port}`)
    })
  }
}
```

Роуты:

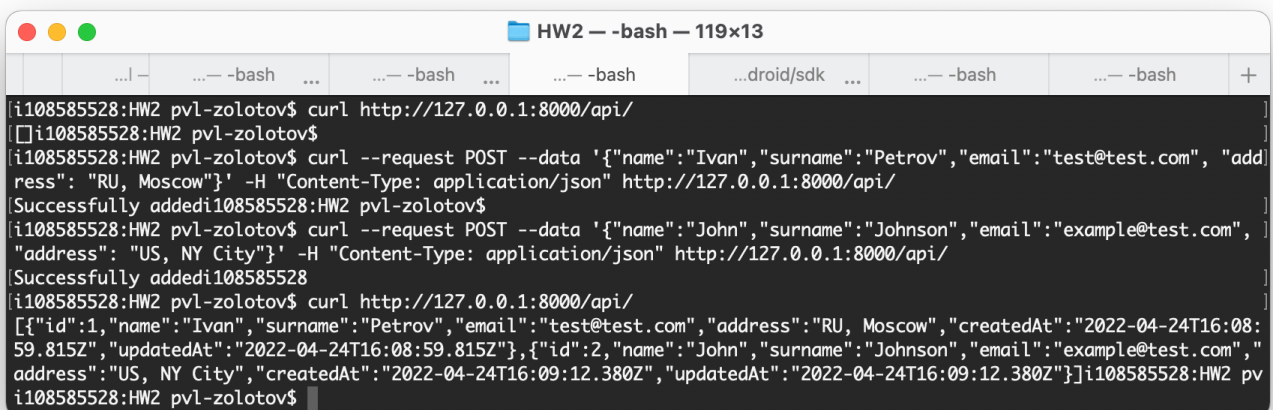
```
import express from "express"
import ExampleController from '../controllers/index'

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

const exampleController = new ExampleController()

router
  .route('/')
  .get(exampleController.get)
  .post(exampleController.post)
```

Результат выполнения:

A screenshot of a terminal window titled "HW2 — -bash — 119x13". The terminal shows a series of curl commands and their outputs. The first command is a GET request to http://127.0.0.1:8000/api/. The second and third commands are POST requests to the same endpoint with JSON data. The fourth command is a GET request that returns a JSON array of two objects. The terminal text is as follows:

```
i108585528:HW2 pvl-zolotov$ curl http://127.0.0.1:8000/api/
[]
i108585528:HW2 pvl-zolotov$ curl --request POST --data '{"name":"Ivan","surname":"Petrov","email":"test@test.com", "address": "RU, Moscow"}' -H "Content-Type: application/json" http://127.0.0.1:8000/api/
Successfully added
i108585528:HW2 pvl-zolotov$ curl --request POST --data '{"name":"John","surname":"Johnson","email":"example@test.com", "address": "US, NY City"}' -H "Content-Type: application/json" http://127.0.0.1:8000/api/
Successfully added
i108585528:HW2 pvl-zolotov$ curl http://127.0.0.1:8000/api/
[{"id":1,"name":"Ivan","surname":"Petrov","email":"test@test.com","address":"RU, Moscow","createdAt":"2022-04-24T16:08:59.815Z","updatedAt":"2022-04-24T16:08:59.815Z"}, {"id":2,"name":"John","surname":"Johnson","email":"example@test.com","address":"US, NY City","createdAt":"2022-04-24T16:09:12.380Z","updatedAt":"2022-04-24T16:09:12.380Z"}]
i108585528:HW2 pvl-zolotov$
```

Вывод

В результате выполнения работы был разработан boilerplate для проекта на Express, Sequelize и Typescript, а также создан Makefile для быстрого исполнения часто используемых команд.