

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэкенд-энд разработка

Отчет

Лабораторная работа №1

Выполнил:  
Траоре Мамуду

Группа:  
К33412

Проверил:  
Добряков Д. И.

Санкт-Петербург

2023 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:

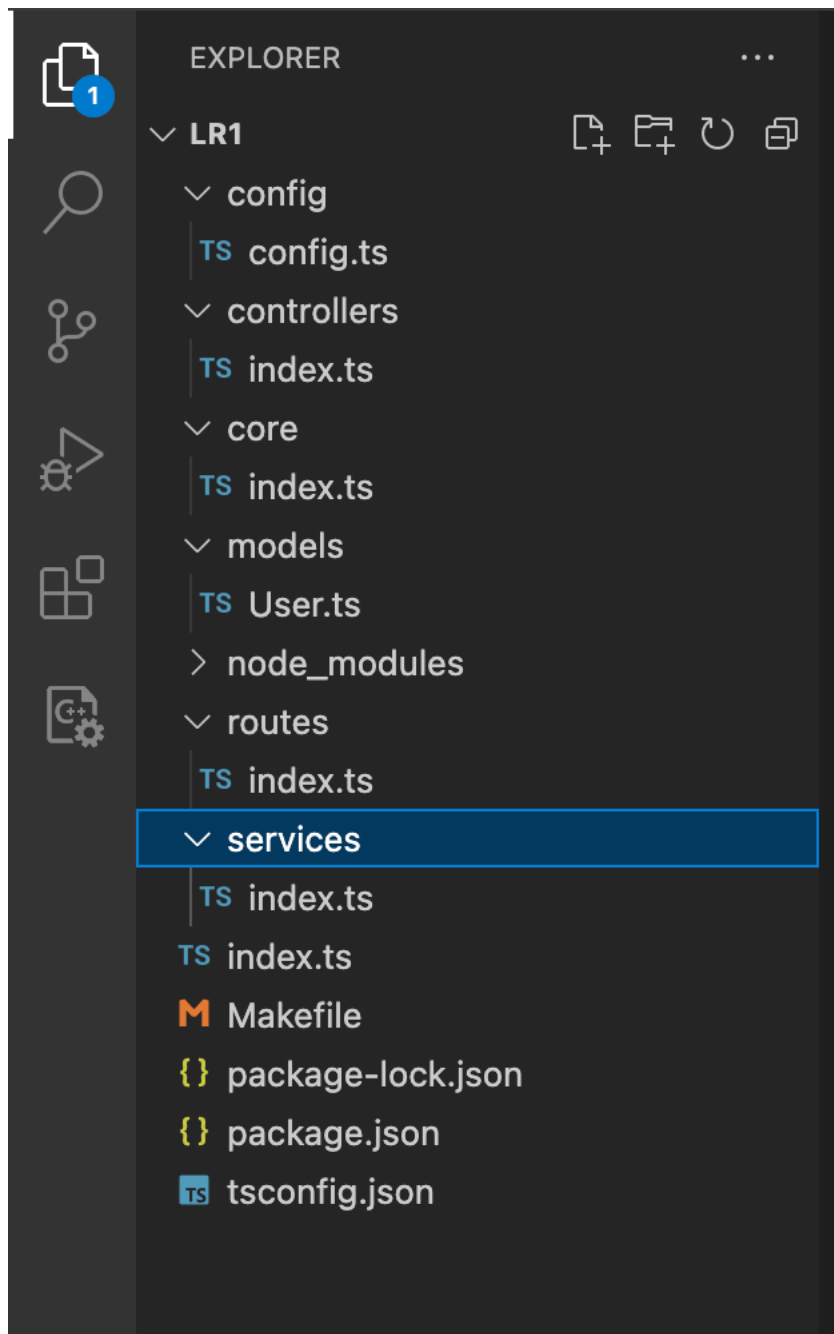
- проведение миграций через sequelize;
- запуск приложения;
- установка зависимостей и сборка приложения. **Ход работы**

### Makefile:

A screenshot of a code editor showing a Makefile. The editor has a dark theme. The Makefile content is as follows:

```
M Makefile •
M Makefile
1 .PHONY: init
2 init:
3     npm i
4     npm run build
5
6 .PHONY: run
7 run:
8     npm start
9
10 .PHONY: migrate
11 migrate:
12     npx sequelize-cli db:migrate
13
14 .DEFAULT_GOAL := init
15
```

### Структура проекта:



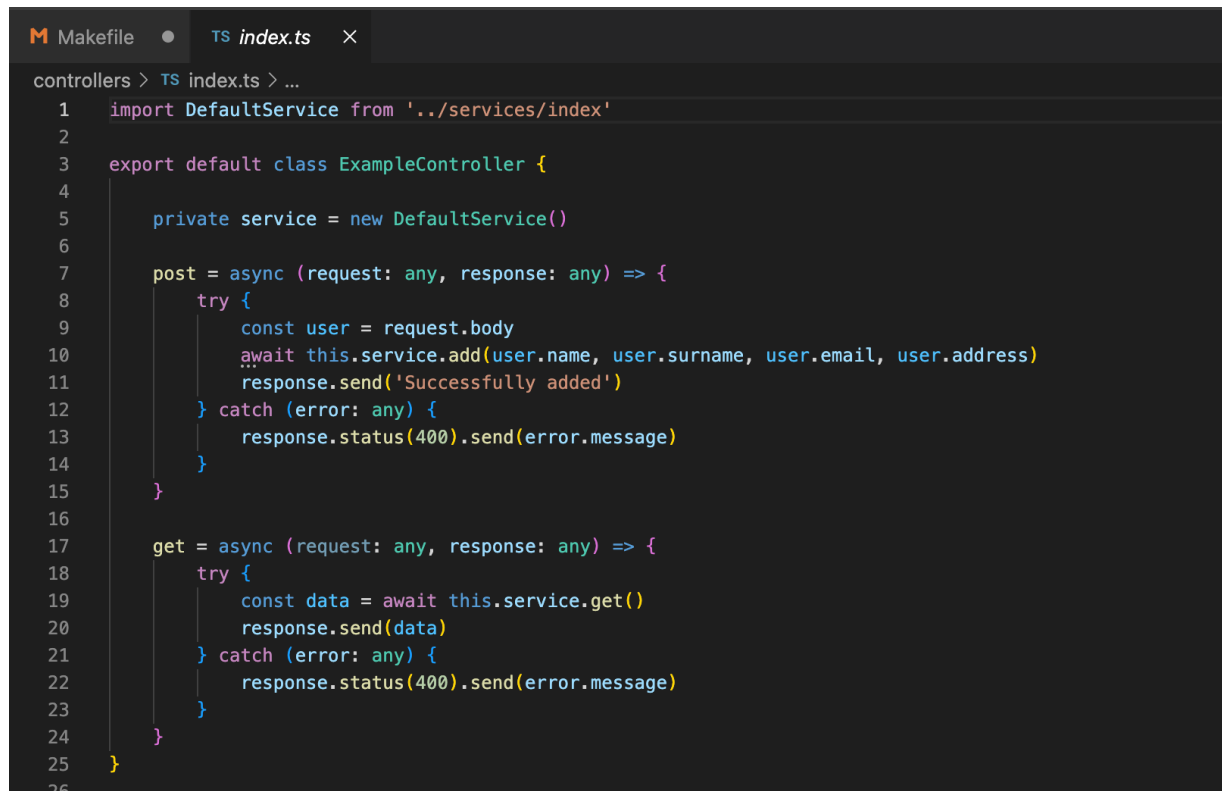
## Сервис:

```
Makefile • TS index.ts ×
services > TS index.ts > ...
1 import User from '../models/User'
2 import { sequelize } from '../config/config'
3
4 export default class DefaultService {
5
6     private repo = sequelize.getRepository(User)
7
8     add(name: string, surname: string, email: string, address: string) {
9         this.repo.create({ name: name, surname: surname, email: email, address: address })
10     }
11
12     get() {
13         return this.repo.findAll()
14     }
15 }
16
17
```

## Модель:

```
Makefile • TS User.ts ×
models > TS User.ts > ...
1 import { Table, Column, Model } from 'sequelize-typescript'
2
3 @Table
4 export default class User extends Model {
5     @Column
6     name: string
7
8     @Column
9     surname: string
10
11     @Column
12     email: string
13
14     @Column
15     address: string
16 }
```

## Контроллер:



```
1 import DefaultService from '../services/index'
2
3 export default class ExampleController {
4
5     private service = new DefaultService()
6
7     post = async (request: any, response: any) => {
8         try {
9             const user = request.body
10             await this.service.add(user.name, user.surname, user.email, user.address)
11             response.send('Successfully added')
12         } catch (error: any) {
13             response.status(400).send(error.message)
14         }
15     }
16
17     get = async (request: any, response: any) => {
18         try {
19             const data = await this.service.get()
20             response.send(data)
21         } catch (error: any) {
22             response.status(400).send(error.message)
23         }
24     }
25 }
26
```

## Файл для запуска сервера:

core > TS index.ts > ...

```
1 import express from "express"
2 import { createServer, Server } from "http"
3 import routes from "../routes/index"
4 import { sequelize } from '../config/config'
5
6 export default class App {
7   public port: number
8   public host: string
9
10  private app: express.Application
11  private server: Server
12
13  constructor(port = 8000, host = "localhost") {
14    this.port = port
15    this.host = host
16
17    this.app = this.createApp()
18    this.server = this.createServer()
19  }
20
21  private createApp(): express.Application {
22    const app = express()
23    const bodyParser = require('body-parser')
24    app.use(bodyParser.urlencoded({ extended: false }))
25    app.use(bodyParser.json())
26    app.use('/api', routes)
27    return app
28  }
29
30  private createServer(): Server {
31    return createServer(this.app)
32  }
33
34  public start(): void {
35    sequelize.sync().then(() => {
36      console.log('DB connected')
37    })
38    this.server.listen(this.port, () => {
39      console.log(`Running server on port ${this.port}`)
40    })
41  }
42 }
43
```

## Роуты:

```
routes > ts index.ts > ...
  1  import express from "express"
  2  import ExampleController from '../controllers/index'
  3
  4  const router: express.Router = express.Router()
  5
  6  const exampleController = new ExampleController()
  7
  8  router
  9    .route('/')
10    .get(exampleController.get)
11    .post(exampleController.post)
12
13  export default router
14
15
```

## Результат при запуске приложения:

```
o MacBook-Air-de-Mamoudou:LR1 mamoudoutraore$ npm start

> lr1@1.0.0 start
> ts-node-dev ./index.ts

[INFO] 22:55:21 ts-node-dev ver. 1.1.8 (using ts-node ver. 9.1.1, typescript ver. 4.6.3)
Running server on port 8000
Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='Users';
Executing (default): CREATE TABLE IF NOT EXISTS `Users` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `name` VARCHAR(255), `surname` VARCHAR(255), `email` VARCHAR(255), `address` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL);
Executing (default): PRAGMA INDEX_LIST('Users')
DB connected
Executing (default): SELECT `id`, `name`, `surname`, `email`, `address`, `createdAt`, `updatedAt` FROM `Users` AS `User`;
Executing (default): SELECT `id`, `name`, `surname`, `email`, `address`, `createdAt`, `updatedAt` FROM `Users` AS `User`;
█
```

## 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.