САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Русинов Василий

Группа К3340

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Нужно написать свой boilerplate на express + TypeORM + typescript.

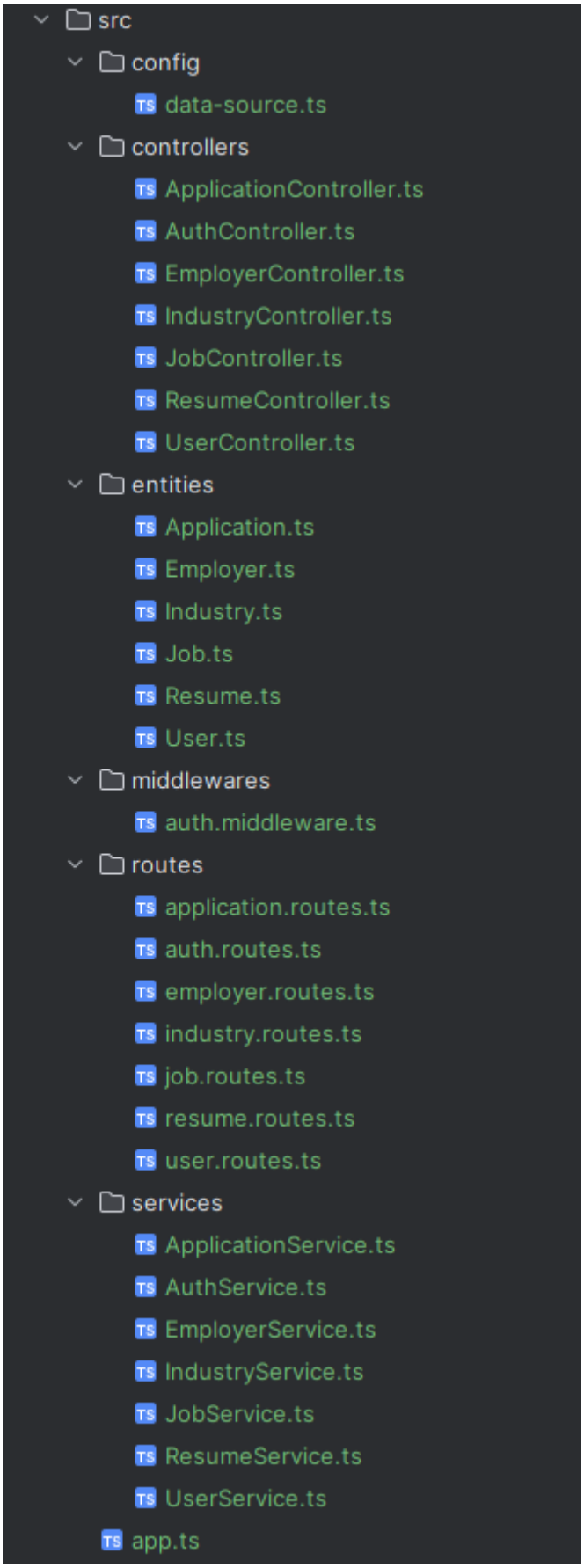
Должно быть явное разделение на:

- модели
- контроллеры
- роуты

Ход работы

Большая часть boilerplate была создана при выполнении ДЗ2. В ходе работы были добавлены: авторизация, регистрация, аутентификация по JWT, хэширование паролей.

Структура проекта:


src
config
data-source.ts
controllers
ApplicationController.ts
AuthController.ts
EmployerController.ts
IndustryController.ts
JobController.ts
ResumeController.ts
UserController.ts
entities
Application.ts
Employer.ts
Industry.ts
Job.ts
Resume.ts
User.ts
middlewares
auth.middleware.ts
routes
application.routes.ts
auth.routes.ts
employer.routes.ts
industry.routes.ts
job.routes.ts
resume.routes.ts
user.routes.ts
services
ApplicationService.ts
AuthService.ts
EmployerService.ts
IndustryService.ts
JobService.ts
ResumeService.ts
UserService.ts
app.ts

AuthService:

```
export class AuthService {
    private userRepo = AppDataSource.getRepository(User);

    async register(email: string, password: string, name: string, role =
"user") {
        const existing = await this.userRepo.findOneBy({ email });
        if (existing) throw new Error("User already exists");

        const password_hash = await bcrypt.hash(password, 10);

        const user = this.userRepo.create({
            email,
            name,
            password: password_hash,
        });

        await this.userRepo.save(user);
        return user;
    }

    async login(email: string, password: string) {
        const user = await this.userRepo.findOneBy({ email });
        if (!user) throw new Error("Invalid credentials");

        const isMatch = await bcrypt.compare(password, user.password);
        if (!isMatch) throw new Error("Invalid credentials");

        const token = jwt.sign(
            { id: user.id},
            JWT_SECRET,
            { expiresIn: "1h" }
        );

        return { token };
    }
}
```

AuthController:

```
export class AuthController {
    async register(req: Request, res: Response) {
        try {
            const { email, password, name, role } = req.body;
            const user = await authService.register(email, password, name,
role);
            res.status(201).json(user);
        } catch (err: unknown) {
            const error = err as Error;
            res.status(400).json({ error: error.message });
        }
    }


    async login(req: Request, res: Response) {
        try {
            const { email, password } = req.body;
            const result = await authService.login(email, password);
            res.json(result);
        } catch (err: unknown) {
            const error = err as Error;
```

```
            res.status(401).json({ error: error.message });
        }
    }
}
```

auth.middleware:

```
export function authenticateJWT(req: Request, res: Response, next:
NextFunction) {
    const authHeader = req.headers.authorization;
    if (!authHeader?.startsWith("Bearer ")) {
        return res.status(401).json({ message: "Unauthorized" });
    }

    const token = authHeader.split(" ")[1];

    try {
        const decoded = jwt.verify(token, JWT_SECRET);
        (req as any).user = decoded;
        next();
    } catch (err) {
        return res.status(403).json({ message: "Forbidden" });
    }
}
```

Защита роутеров пользователя:

```
router.post("/", authenticateJWT, controller.create.bind(controller));
router.get("/", authenticateJWT, controller.getAll.bind(controller));
router.get("/email/:email", authenticateJWT,
controller.getByEmail.bind(controller));
router.get("/:id", authenticateJWT, controller.getById.bind(controller));
router.put("/:id", authenticateJWT, controller.update.bind(controller));
router.delete("/:id", authenticateJWT, controller.delete.bind(controller));
```

Вывод

В ходе лабораторной работы был разработан boilerplate на express + TypeORM + typescript.