САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Реализация boilerplate

Выполнил:

Петухов Семён

Группа
К3339

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

Изначально была разработана иерархия файловой структуры бойлерплэйт`а.

```
└── boiler_plate
    │
    ├── src
    │   ├── config
    │   ├── controllers
    │   │   └── basicController.ts
    │   ├── models
    │   │   └── basicModel.ts
    │   ├── routes
    │   │   └── basicRoutes.ts
    │   ├── app.ts
    │   └── index.ts
    │
    ├── .env
    ├── nodemon.json
    ├── package.json
    ├── README.md
    └── tsconfig.json
```

Далее был разработана стандартная схема модели базы данных с несколькими полями.

```ts
import { Entity, PrimaryGeneratedColumn, Column, OneToMany } from "typeorm";

@Entity()
export class BasicModel {
    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    ColumnString: string;

}
```

Далее была описан контроллер, с описанием CRUD запросов к моделе выше. В каждой функции описан конкретный запрос с универсальным функционалом

```ts
import { Request, Response } from "express";
import { AppDataSource } from "../config/data-source";
import { BasicModel } from "../models/basicModel";

const basicRepo = AppDataSource.getRepository(BasicModel);

// Получить все записи
export const getAll = async (_: Request, res: Response) => {
    const items = await basicRepo.find();
    res.json(items);
};

// Получить запись по ID
export const getOne = async (req: Request, res: Response) => {
    const id = parseInt(req.params.id);
    const item = await basicRepo.findOneBy({ id });

    if (!item) {
        return res.status(404).json({ message: "Not found" });
    }

    res.json(item);
};

// Создать новую запись
export const create = async (req: Request, res: Response) => {
    const { ColumnString } = req.body;

    const newItem = basicRepo.create({ ColumnString });

    await basicRepo.save(newItem);
    res.status(201).json(newItem);
};

// Обновить запись
export const update = async (req: Request, res: Response) => {
    const id = parseInt(req.params.id);
    const item = await basicRepo.findOneBy({ id });

    if (!item) {
        return res.status(404).json({ message: "Not found" });
    }

    const { ColumnString } = req.body;
    item.ColumnString = ColumnString;

    await basicRepo.save(item);
    res.json(item);
};

// Удалить запись
export const remove = async (req: Request, res: Response) => {
    const id = parseInt(req.params.id);
    const item = await basicRepo.findOneBy({ id });

    if (!item) {
        return res.status(404).json({ message: "Not found" });
    }

    await basicRepo.remove(item);
    res.status(204).send();
};
```

Далее был описан файл с эндпоинтами, которые ссылаются на функции из контроллера

```
import { Router } from "express";
import {
    getAll,
    getOne,
    create,
    update,
    remove,
} from "../controllers/basicController";

const router = Router();

// 📌 Маршруты для BasicModel
router.get("/basic", getAll);            // Получить все записи
router.get("/basic/:id", getOne);        // Получить одну по ID
router.post("/basic", create);           // Создать новую запись
router.put("/basic/:id", update);        // Обновить существующую
router.delete("/basic/:id", remove);     // Удалить по ID

exprt default rout
```

Вывод

По результатам работы был разработан boilerplate для создания на его основе базы данных и CRUD-запросов к ней.