САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

Выполнил:
Ребров С. А.

Группа:
K3339

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Ход работы

За основу были взяты файлы из прошлой работы, и были дописаны возвращаемые статусы. Например, в каждом try – catch были добавлены статусы 500, а в запросах get, update, delete – статусы 200, 204.

Пример для application.controller.ts:

```
import { Request, Response, NextFunction } from "express";
import { Application } from "../entity/application";

export const createApplication = async (req: Request, res: Response, next: NextFunction): Promise<void> => {
  try {
    const { userId, jobId, coverLetter, status } = req.body;
    const app = new Application();
    app.user = { id: userId } as any;
    app.job = { id: jobId } as any;
    app.coverLetter = coverLetter;
    app.status = status;
    await app.save();
    res.status(201).json(app);
  } catch (error) {
    res.status(500).json({ message: "Internal Server Error" });
  }
};
```

```
export const getApplications = async (_req: Request, res: Response, next: NextFunction): Promise<void> => {
  try {
    const apps = await Application.find({ relations: ["user", "job"] });
    res.status(200).json(apps);
  } catch (error) {
    res.status(500).json({ message: "Internal Server Error" });
  }
};

export const getApplicationById = async (req: Request, res: Response, next: NextFunction): Promise<void> => {
  try {
    const id = parseInt(req.params.id, 10);
    if (isNaN(id)) {
    res.status(400).json({ message: "Invalid ID" });
    return;
  }
  const app = await Application.findOne({ where: { id }, relations: ["user", "job"] });
    if (!app) {
    res.status(404).json({ message: "Application not found" });
    return;
  }
    res.status(200).json(app);
  } catch (error) {
```

```
    res.status(500).json({ message: "Internal Server Error" });
  }
};

export const updateApplication = async (req: Request, res: Response, next: NextFunction): Promise<void> => {
  try {
    const id = parseInt(req.params.id, 10);
    const app = await Application.findOne({ where: { id } });
    if (!app) {
   res.status(404).json({ message: "Not found" });
   return;
  }
    app.coverLetter = req.body.coverLetter;
    app.status = req.body.status;
    await app.save();
    res.status(200).json(app);
  } catch (error) {
    res.status(500).json({ message: "Internal Server Error" });
  }
};

export const deleteApplication = async (req: Request, res: Response, next: NextFunction): Promise<void> => {
  try {
    const id = parseInt(req.params.id, 10);
    const app = await Application.findOne({ where: { id } });
```

```
    if (!app) {
  res.status(404).json({ message: "Not found" })
  return;
};
    await app.remove();
    res.status(204).json({ message: "Application deleted" });
  } catch (error) {
    res.status(500).json({ message: "Internal Server Error" });
  }
};
```

Вывод

В ходе работы был реализован полнофункциональный RESTful API на базе Express и TypeScript с использованием заранее подготовленного boilerplate-проекта. Были разработаны маршруты для всех CRUD-операций над сущностями, с соблюдением REST-принципов и корректной обработкой HTTP-статусов.