САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 2
Тестирование, разработка и документирование RestfulAPI

Выполнили:

Жижилева Арина
Строганова Елизавета

К3342

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

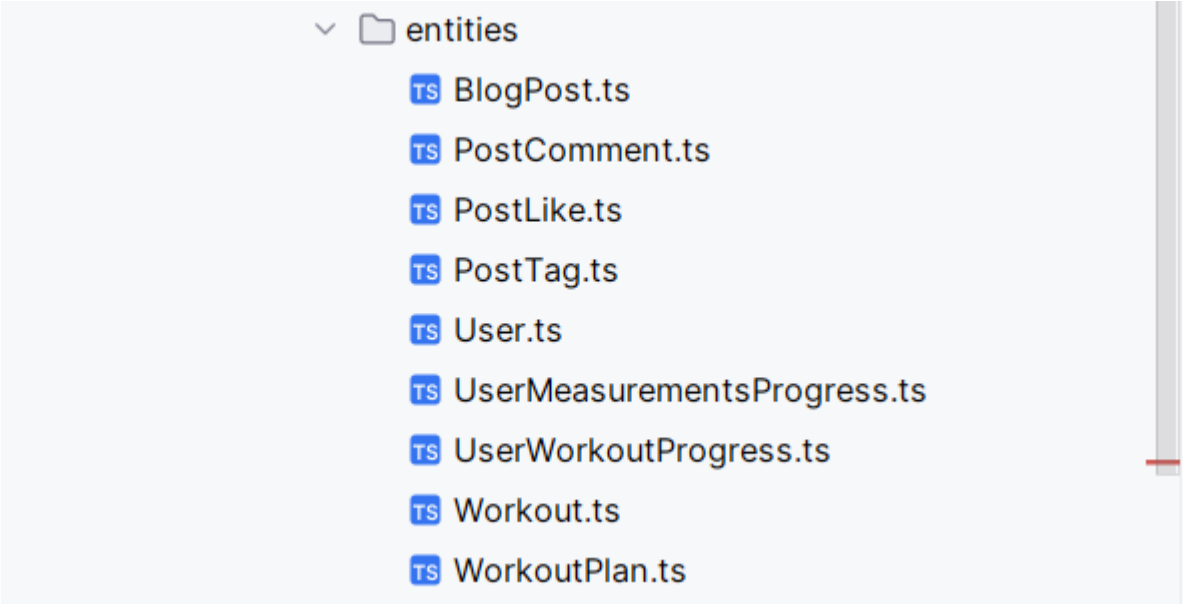
ЛР2: Реализация REST API на основе boilerplate

Задание:

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

Ход работы

Мы уже создали модели.

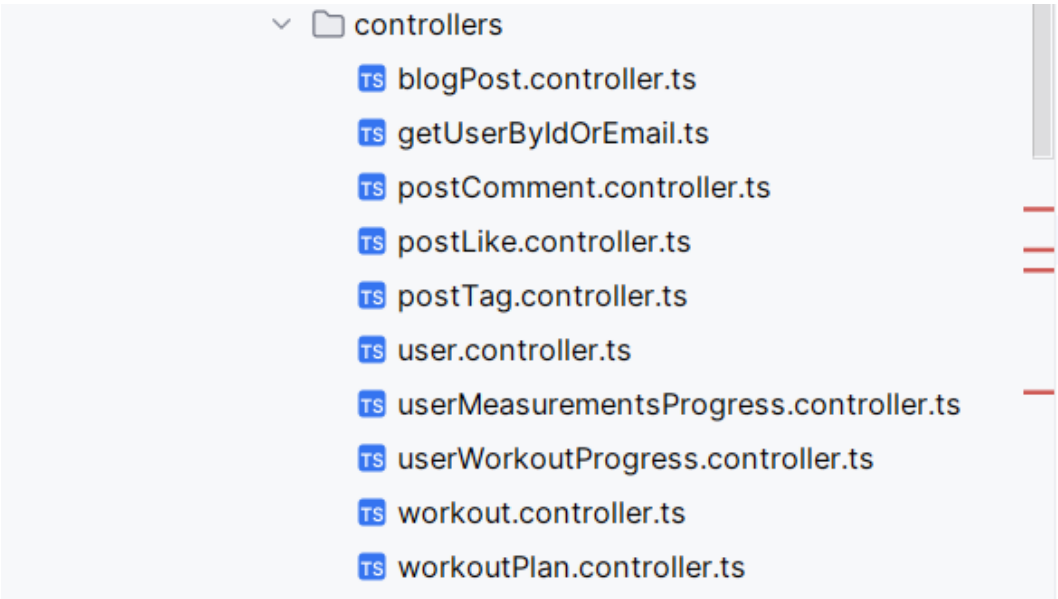


Код из модели Workout.ts

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

import { WorkoutPlan } from "./WorkoutPlan";

import { UserWorkoutProgress } from "./UserWorkoutProgress";


@Entity()

export class Workout {

 @PrimaryGeneratedColumn()
```

```
 workout_id!: number;


 @ManyToOne(() => WorkoutPlan, (plan) => plan.workouts)

 plan!: WorkoutPlan;


 @Column()

 title!: string;


 @Column()

 description!: string;


 @Column()

 video_url!: string;


 @Column()

 duration_minutes!: number;


 @Column()

 difficulty_level!: string;


 @Column()

 type!: string;


 @OneToMany(() => UserWorkoutProgress, (progress) => progress.workout)

 userWorkoutProgresses!: UserWorkoutProgress[];

}
```

Создали контроллеры



Пример код из контроллера для postLike

```
import { Request, Response } from "express";
import { postLikeRepository } from "../repositories/postLike.repository";


export const PostLikeController = {
 create: async (req: Request, res: Response) => {
   const like = await postLikeRepository.save(req.body);
   res.json(like);
 },


 getAll: async (_: Request, res: Response) => {
   const likes = await postLikeRepository.find();
   res.json(likes);
 },


 getById: async (req: Request, res: Response) => {
   const like = await postLikeRepository.findOneBy({ like_id: +req.params.id });
   if (!like) return res.status(404).json({ message: "Like not found" });
```

```
    res.json(like);
  },

  update: async (req: Request, res: Response) => {
    await postLikeRepository.update(req.params.id, req.body);
    res.json({ message: "Like updated" });
  },

  delete: async (req: Request, res: Response) => {
    await postLikeRepository.delete(req.params.id);
    res.json({ message: "Like deleted" });
  }
};
```

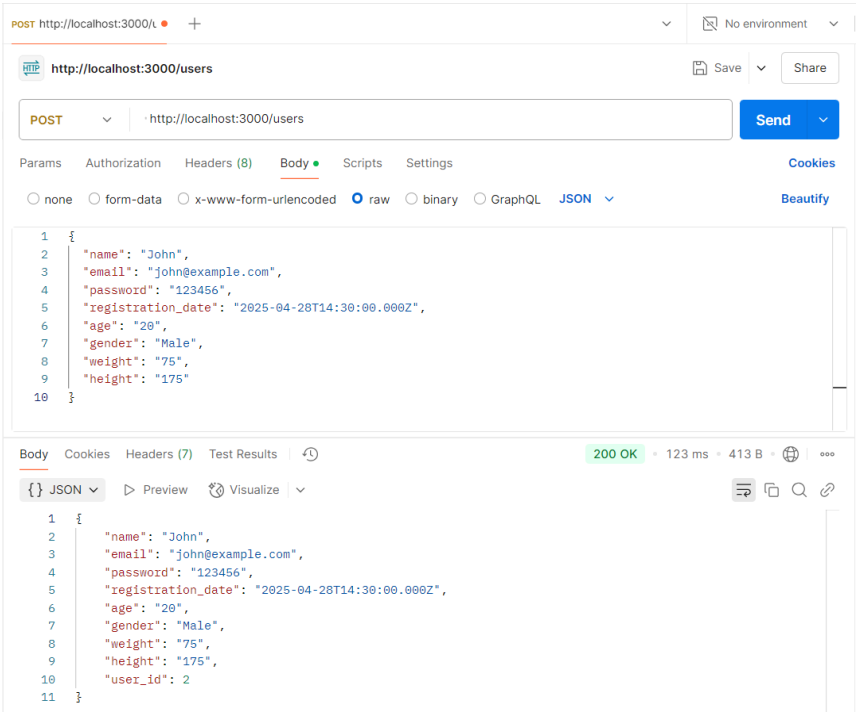
Написали роуты и подключили в server.ts

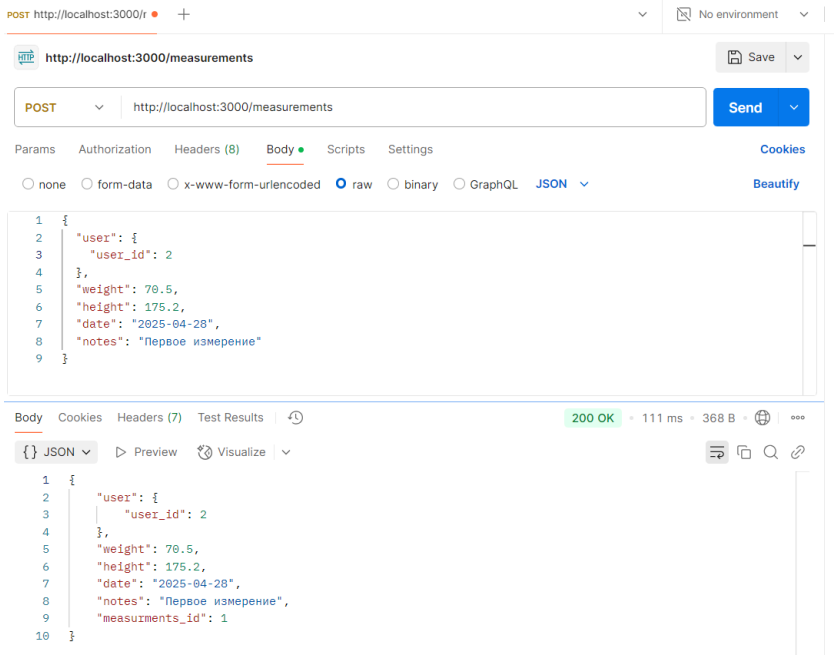
```
import { AppDataSource } from "./database/data-source";
import app from "./app";

const PORT = process.env.PORT || 3000;

AppDataSource.initialize()
  .then(() => {
    console.log("📦 Database connected!");
    app.listen(PORT, () => {
      console.log(`🚀 Server is running on port ${PORT}`);
    });
  })``
  .catch((error) => console.error("❌ Database connection error:", error));
```

App.ts

```
import express from "express";
import userRoutes from "./routes/user.routes";
```

```
import blogPostRoutes from "./routes/blogPost.routes";

import                        userMeasurementsProgressRoutes                        from
"./routes/userMeasurementsProgress.routes";

import workoutPlanRoutes from "./routes/workoutPlan.routes";

import workoutRoutes from "./routes/workout.routes";

import userWorkoutProgressRoutes from "./routes/userWorkoutProgress.routes";

import postLikeRoutes from "./routes/postLike.routes";

import postCommentRoutes from "./routes/postComment.routes";

import postTagRoutes from "./routes/postTag.routes";


const app = express();

app.use(express.json());


// Роуты

app.use("/users", userRoutes);

app.use("/posts", blogPostRoutes);

app.use("/measurements", userMeasurementsProgressRoutes);

app.use("/plans", workoutPlanRoutes);

app.use("/workouts", workoutRoutes);

app.use("/workout-progress", userWorkoutProgressRoutes);

app.use("/like", postLikeRoutes);

app.use("/comment", postCommentRoutes);

app.use("/tag", postTagRoutes);

export default app;
```


POST http://localhost:3000/
No environment
http://localhost:3000/users
Save
Share
POST
http://localhost:3000/users
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "John",
"email": "john@example.com",
"password": "123456",
"registration_date": "2025-04-28T14:30:00.000Z",
"age": "20",
"gender": "Male",
"weight": "75",
"height": "175"
}
Body
Cookies
Headers (7)
Test Results
200 OK
123 ms
413 B
JSON
Preview
Visualize
{
"name": "John",
"email": "john@example.com",
"password": "123456",
"registration_date": "2025-04-28T14:30:00.000Z",
"age": "20",
"gender": "Male",
"weight": "75",
"height": "175",
"user_id": 2
}


POST http://localhost:3000/r
No environment
http://localhost:3000/measurements
Save
POST
http://localhost:3000/measurements
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"user": {
"user_id": 2
},
"weight": 70.5,
"height": 175.2,
"date": "2025-04-28",
"notes": "Первое измерение"
}
Body
Cookies
Headers (7)
Test Results
200 OK
111 ms
368 B
JSON
Preview
Visualize
{
"user": {
"user_id": 2
},
"weight": 70.5,
"height": 175.2,
"date": "2025-04-28",
"notes": "Первое измерение",
"measurments_id": 1
}