

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 2, Лабораторная работа 1
Работа с TypeORM, Реализация boilerplate

Выполнила:

Жижилева Арина
Строганова Елизавета

К3342

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача Д32:

Реализовать все модели данных, спроектированные в рамках Д31

Реализовать набор из CRUD-методов для работы с моделями данных средствами Express + TypeScript

Реализовать API-эндпоинт для получения пользователя по id/email

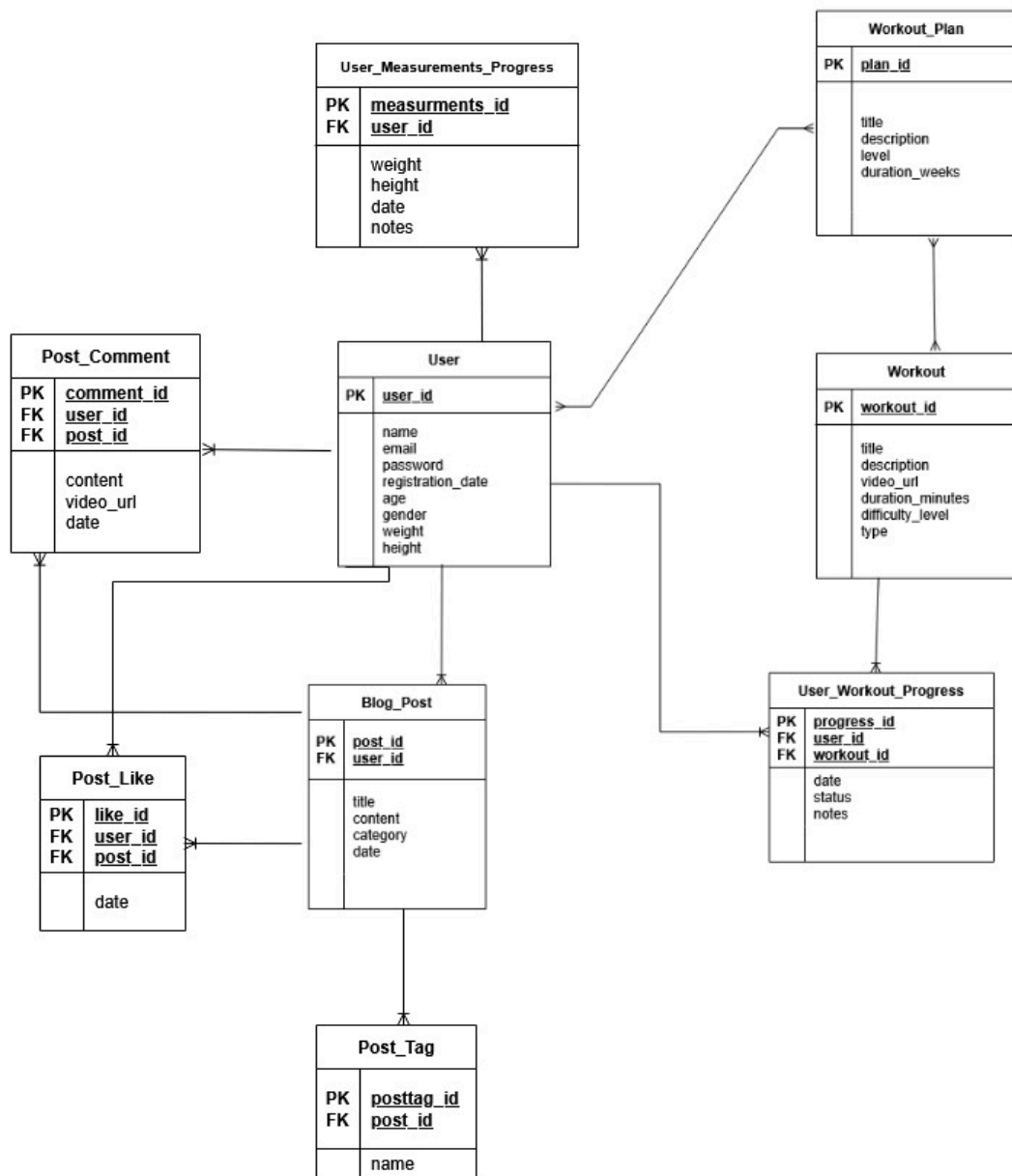
Задача ЛР1:

Нужно написать свой boilerplate на express + TypeORM + typescript.

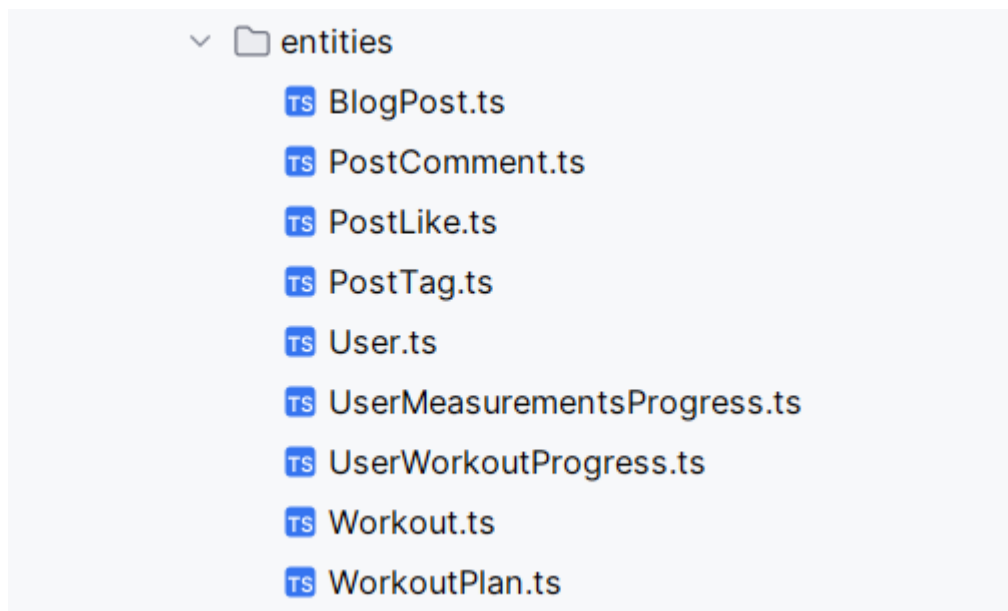
Должно быть явное разделение на:

- модели
- контроллеры
- роуты

Ход работы



Создание моделей



Пример кода из user.ts

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

```
import { BlogPost } from "../BlogPost";
```

```
import { UserMeasurementsProgress } from "../UserMeasurementsProgress";
```

```
import { UserWorkoutProgress } from "../UserWorkoutProgress";
```

```
@Entity()
```

```
export class User {
```

```
  @PrimaryGeneratedColumn()
```

```
  user_id!: number;
```

```
  @Column()
```

```
  name!: string;
```

```
  @Column({ unique: true })
```

```
email!: string;
```

```
@Column()
```

```
password!: string;
```

```
@Column()
```

```
registration_date!: Date;
```

```
@Column()
```

```
age!: number;
```

```
@Column()
```

```
gender!: string;
```

```
@Column("float")
```

```
weight!: number;
```

```
@Column("float")
```

```
height!: number;
```

```
@OneToMany(() => BlogPost, (post) => post.user)
```

```
blogPosts!: BlogPost[];
```

```
@OneToMany(() => UserMeasurementsProgress, (progress) => progress.user)
```

```
measurementsProgress!: UserMeasurementsProgress[];
```

```
@OneToMany(() => UserWorkoutProgress, (progress) => progress.user)
```

```
workoutProgress!: UserWorkoutProgress[];

}
```

Установка зависимостей

npm install

npm install -g ts-node-dev typescript

Создание бд в postgres

1. через win+r services.msc запускаю postgres.
2. в cmd psql -U postgres
3. CREATE DATABASE fitness_app;

Файл для подключения к бд:



```
1 import "reflect-metadata";
2 import { DataSource } from 'typeorm';
3 import { User } from "../entities/User";
4 import { BlogPost } from "../entities/BlogPost";
5 import { UserMeasurementsProgress } from "../entities/UserMeasurementsProgress";
6 import { WorkoutPlan } from "../entities/WorkoutPlan";
7 import { Workout } from "../entities/Workout";
8 import { UserWorkoutProgress } from "../entities/UserWorkoutProgress";
9
10 export const AppDataSource = new DataSource({
11   type: "postgres",
12   host: "localhost",
13   port: 5432,
14   username: "postgres",
15   password: "Arina2992",
16   database: "fitness_app",
17   synchronize: true,
18   logging: false,
19   entities: [User, BlogPost, UserMeasurementsProgress, WorkoutPlan, Workout, UserWorkoutProgress],
20   migrations: [],
21   subscribers: [],
22 });
23
```

Запуск проекта в рущарм терминале командой npm run dev

```
postgres=# \c fitness_app
Вы подключены к базе данных "fitness_app" как пользователь "postgres".
fitness_app=# \dt
```

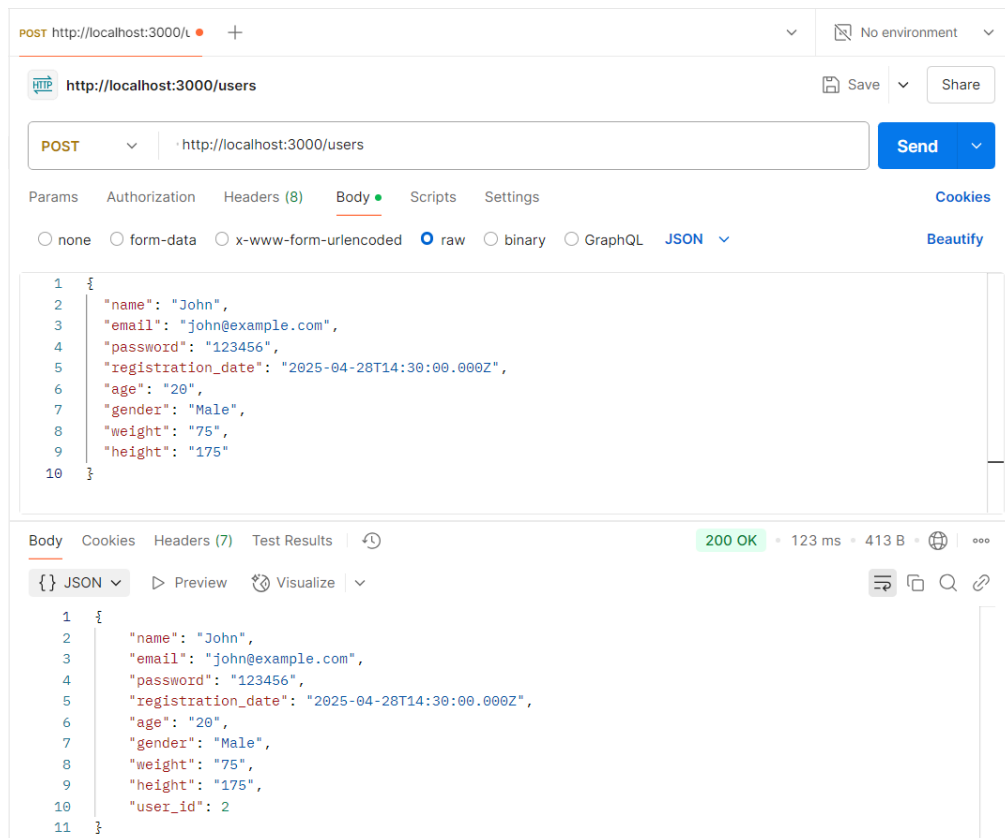
Схема	Таблица	Тип данных	Владелец
public	blog_post	text	postgres
public	user	text	postgres
public	user_measurements_progress	text	postgres
public	user_workout_progress	text	postgres
public	workout	text	postgres
public	workout_plan	text	postgres

```
(6 строк)
```

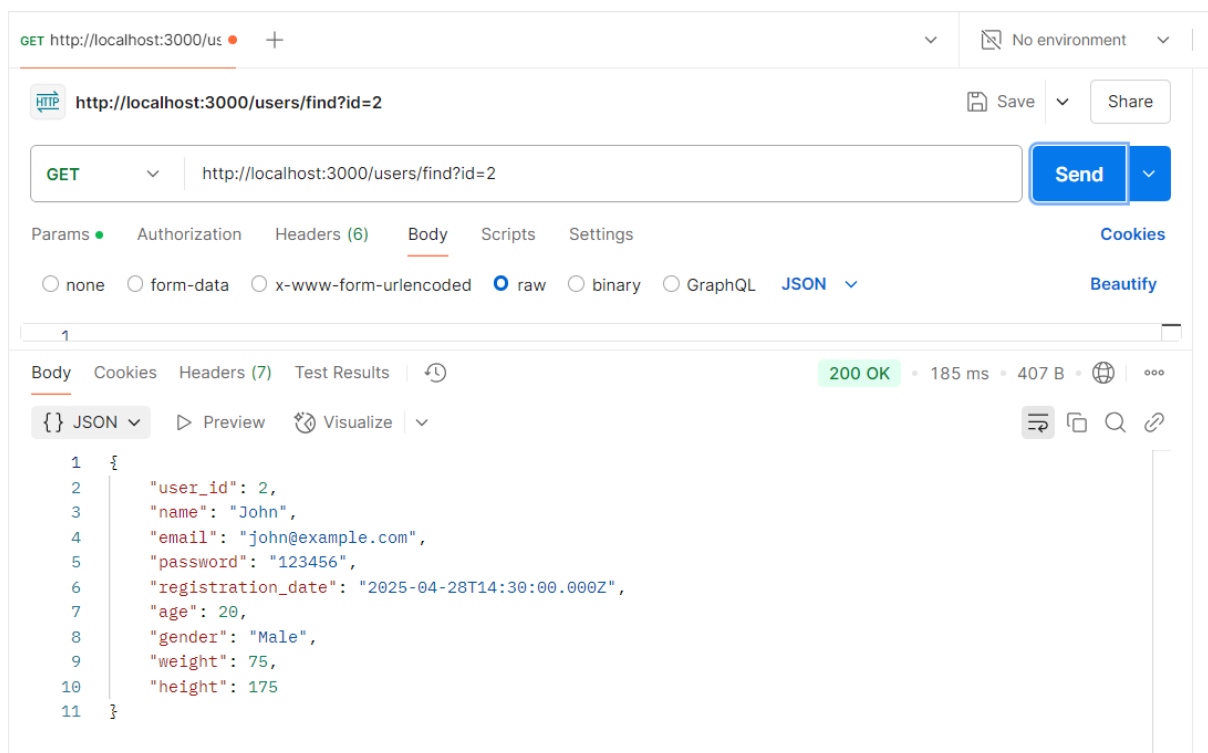
Файловая структура проекта:



Регистрация



Получение пользователя по id:



Получение пользователя по email:

GET http://localhost:3000/users/find?email=john@example.com

GET http://localhost:3000/users/find?email=john@example.com

Params Authorization Headers (6) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

200 OK • 68 ms • 407 B

JSON Preview Visualize

```
1 {
2   "user_id": 2,
3   "name": "John",
4   "email": "john@example.com",
5   "password": "123456",
6   "registration_date": "2025-04-28T14:30:00.000Z",
7   "age": 20,
8   "gender": "Male",
9   "weight": 75,
10  "height": 175
11 }
```

Measurements

POST http://localhost:3000/r

POST http://localhost:3000/measurements

Params Authorization Headers (8) Body Scripts Settings

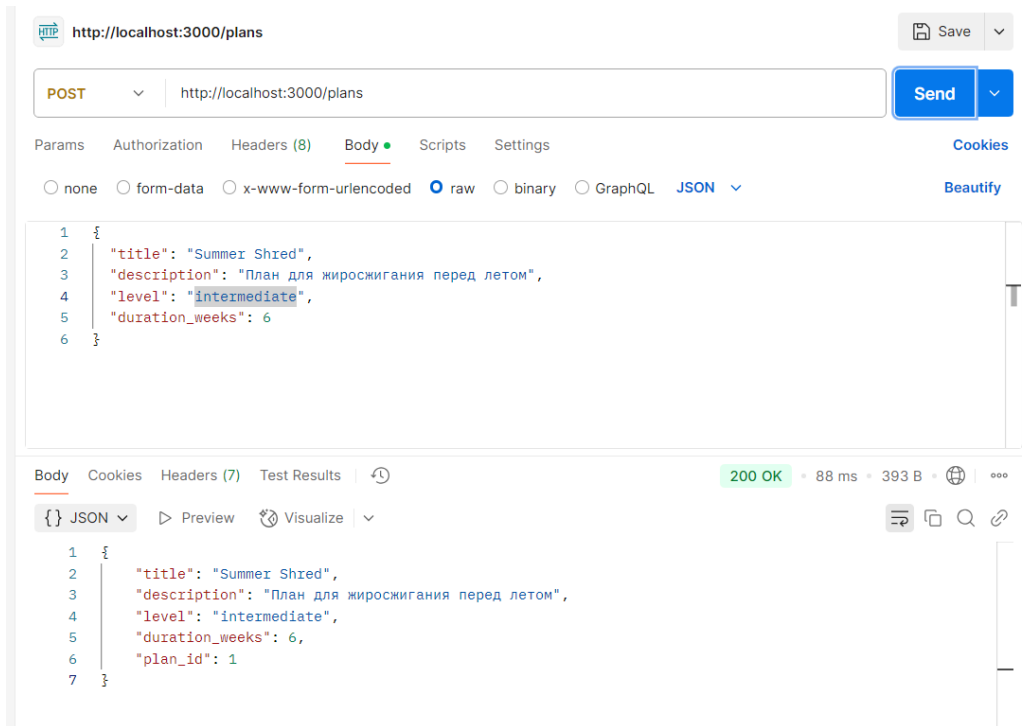
none form-data x-www-form-urlencoded raw binary GraphQL JSON

200 OK • 111 ms • 368 B

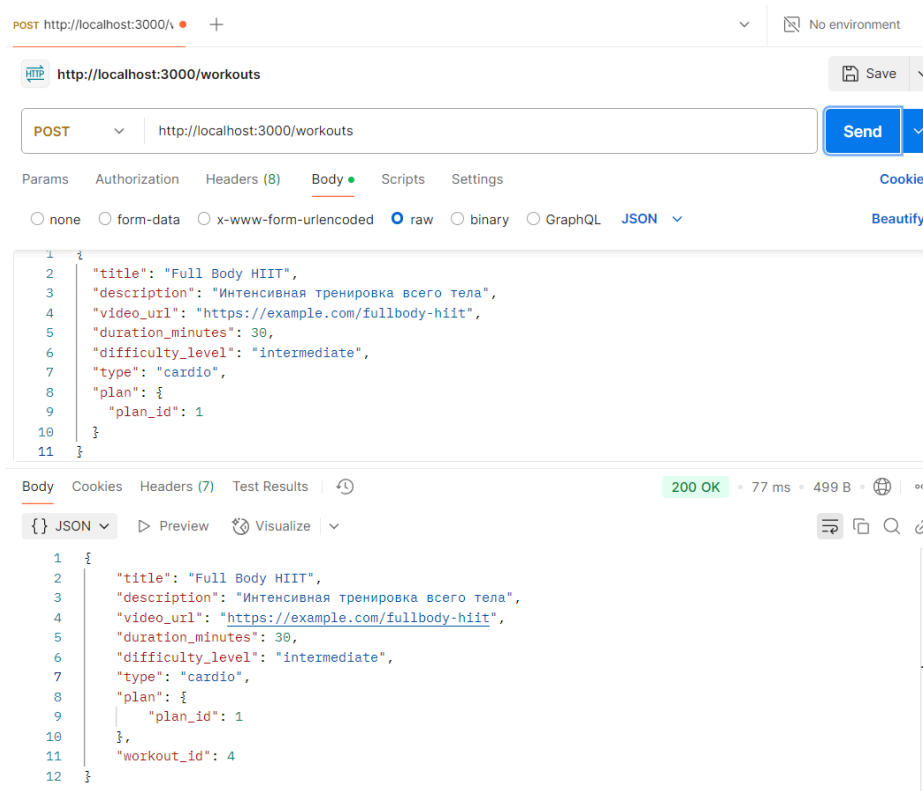
JSON Preview Visualize

```
1 {
2   "user": {
3     "user_id": 2
4   },
5   "weight": 70.5,
6   "height": 175.2,
7   "date": "2025-04-28",
8   "notes": "Первое измерение",
9 }
10 {
11   "user": {
12     "user_id": 2
13   },
14   "weight": 70.5,
15   "height": 175.2,
16   "date": "2025-04-28",
17   "notes": "Первое измерение",
18   "measurements_id": 1
19 }
```

Создание плана тренировок



Создание тренировки



Отслеживание прогресса

HTTP

http://localhost:3000/workout-progress

Save

▼

POST

▼

http://localhost:3000/workout-progress

Send

▼

Params

Authorization

Headers (8)

Body •

Scripts

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

☒ JSON

▼

Beautify

```
1 {
2   "user": {
3     "user_id": 2
4   },
5   "workout": {
6     "workout_id": 4
7   },
8   "date": "2025-04-30T14:30:00.000Z",
9   "status": "Done"
10 }
11
```

Body

Cookies

Headers (7)

Test Results

↺

200 OK

82 ms

364 B

🌐

⋮

{ } JSON

▼

▶ Preview

🔗 Visualize

▼

🔧

📄

🔍

🔗

```
1 {
2   "user": {
3     "user_id": 2
4   },
5   "workout": {
6     "workout_id": 4
7   },
8   "date": "2025-04-30T14:30:00.000Z",
9   "status": "Done",
10  "notes": null,
11  "progress_id": 1
12 }
```