

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 2

Выполнил:

Якунин Андрей

Группа
К3341

Проверил:

Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

- Реализовать все модели данных, спроектированные в рамках ДЗ1
- Реализовать набор из CRUD-методов для работы с моделями данных средствами Express + TypeScript
- Реализовать API-эндпоинт для получения пользователя по id/email

Ход работы

- 1) Реализовать все модели данных, спроектированные в рамках ДЗ1.

Все модели хранятся в папке models

Пример модели для категории:

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

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

  @Column()
  name: string;

  @OneToMany(() => Advertisement, advertisement =>
advertisement.category)
  advertisements: Advertisement[];
}
```

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

Crud методы хранятся в папке controllers, а маршруты к ним в routers

пример контроллера для users

```
import { Request, Response } from "express";
import { AppDataSource } from
"../config/AppDataSource";
import { User } from "../models/User";

const userRepo = AppDataSource.getRepository(User);
```

```

export const createUser = async (req: Request, res:
Response) => {
  try {
    const userData = req.body;
    const user = userRepo.create(userData);
    const savedUser = await userRepo.save(user);
    res.status(201).json(savedUser);
  } catch (error: any) {
    res.status(500).json({ message: error.message
});
  }
};

export const getAllUsers = async (req: Request, res:
Response) => {
  try {
    const users = await userRepo.find();
    res.json(users);
  } catch (error: any) {
    res.status(500).json({ message: error.message
});
  }
};

```

3) Реализовать API-эндпоинт для получения пользователя по id/email.

```

export const getUserByEmail = async (req: Request,
res: Response): Promise<any> => {
  try {
    const { email } = req.query;

    if (!email) {
      return res.status(400).json({ message:
"Email is required" });
    }
  }
};

```

```

    const user = await userRepo.findOne({
      where: { mail: String(email) }
    });

    if (!user) {
      return res.status(404).json({ message:
"User not found" });
    }

    res.json(user);
  } catch (error: any) {
    res.status(500).json({ message: error.message
});
  }
};

```

Пример выполнения запроса:

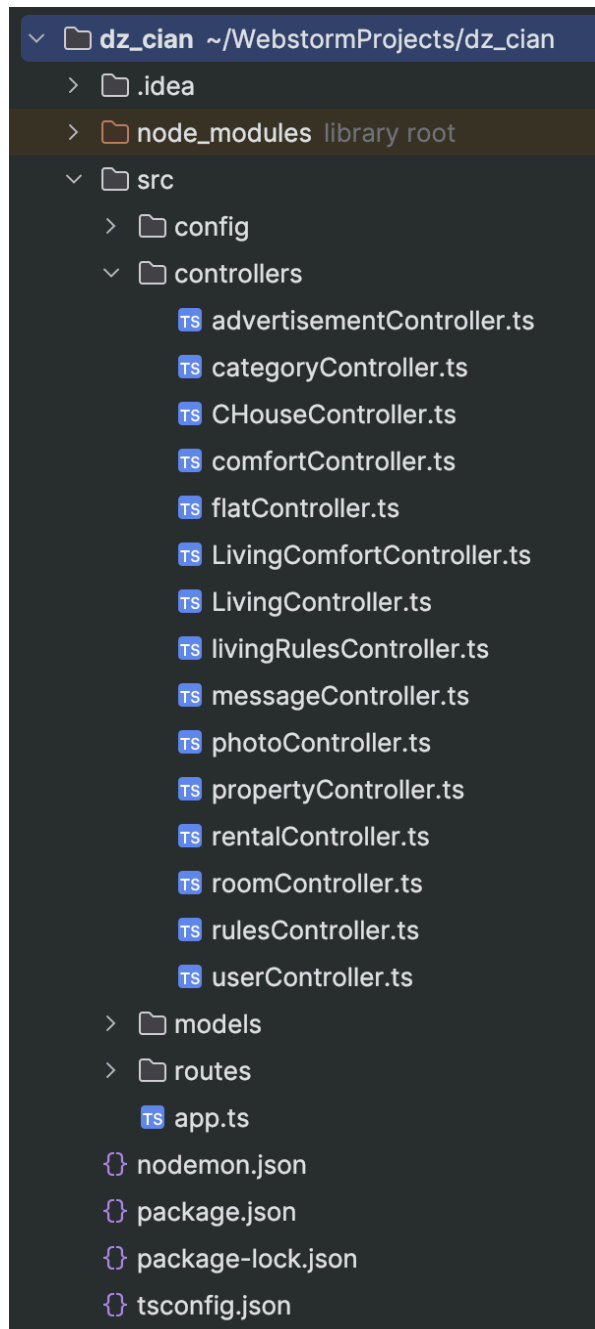
The screenshot shows a REST client interface. At the top, a GET request is defined with the URL `http://localhost:3000/users/by-email?email=andrucrut@mail.com`. Below the URL bar, there are tabs for Params, Authorization, Headers (8), Body, Scripts, and Settings. The 'Body' tab is selected, and the request body is set to 'raw'. Below the request definition, there is a section for the response. The 'Body' tab is selected, and the response is displayed in JSON format. The response status is 200. The JSON body contains the following data:

```

{
  "id": 2,
  "mail": "andrucrut@mail.com",
  "password": "1234567",
  "first_name": "Andrey",
  "second_name": "Ivanov",
  "phone": "+79991234567",
  "created_at": "2025-04-26T15:38:18.028Z",
  "updated_at": "2025-04-26T15:38:18.028Z"
}

```

4) Наш проект на данный момент имеет такую структуру:



Вывод

В ходе выполнения лабораторной работы были реализованы все необходимые модели данных, настроены контроллеры и маршруты. Проект структурирован логично и соответствует требованиям для эффективной работы и дальнейшего развития.