САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая/Лабораторная работа

Выполнил:

Цой Степан

Группа:

K3340

Проверил:
Добряков Д. И.

Санкт-Петербург

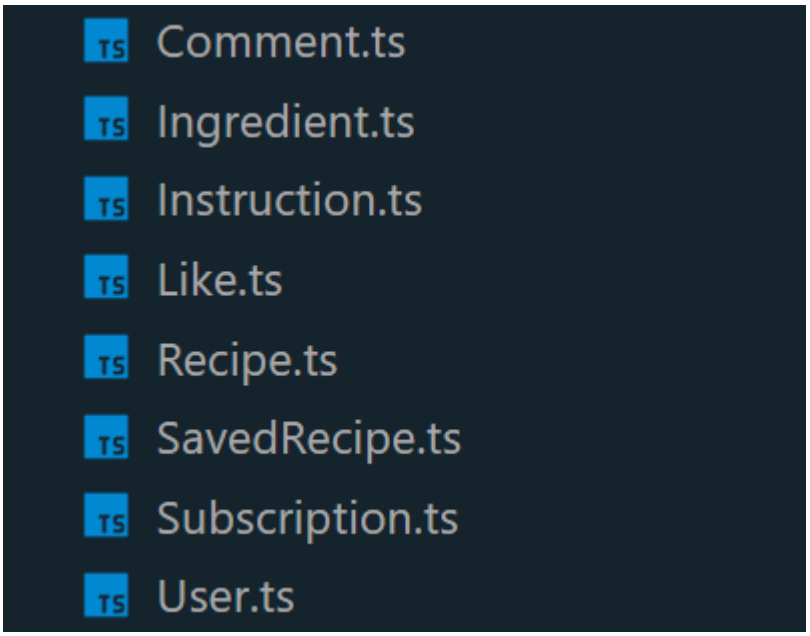
2025 г.

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. Были реализованы все модели данных, спроектированные в рамках ДЗ1.



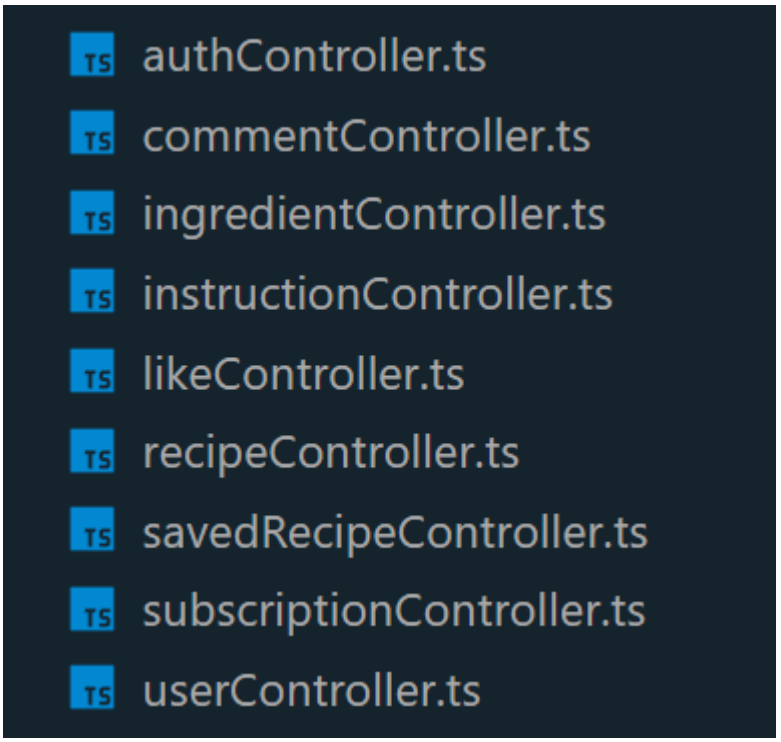
Пример модели User:

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

import Recipe from "./Recipe";

import Comment from "./Comment";

import Like from "./Like";

import SavedRecipe from "./SavedRecipe";

import Subscription from "./Subscription";

@Entity()

export default class User {
```

```
  @PrimaryGeneratedColumn()

  user_id: number;

  @Column({ unique: true })

  username: string;

  @Column({ unique: true })

  email: string;

  @Column()

  password: string;

  @Column({ nullable: true })

  profile_photo: string;

  @Column("text", { nullable: true })

  bio: string;

  @Column({ type: "timestamp", default: () => "CURRENT_TIMESTAMP"
})

  created_at: Date;

  @OneToMany(() => Recipe, (recipe) => recipe.user)

  recipes: Recipe[];

  @OneToMany(() => Comment, (comment) => comment.user)

  comments: Comment[];

  @OneToMany(() => Like, (like) => like.user)

  likes: Like[];

  @OneToMany(() => SavedRecipe, (savedRecipe) =>
savedRecipe.user)

  saved_recipes: SavedRecipe[];

  @OneToMany(() => Subscription, (subscription) =>
subscription.follower)
```

```
  following: Subscription[];

  @OneToMany(() => Subscription, (subscription) =>
subscription.followed)

  followers: Subscription[];
```

2. Были реализованы наборы из CRUD-методов для работы с моделями данных средствами Express + TypeScript .



3. Был реализовать API-эндпоинт для получения пользователя по id.

```
static getUserById = async (req: Request, res: Response) => {
  try {
    const userRepository = AppDataSource.getRepository(User);
    const user = await userRepository.findOne({
      where: { user_id: parseInt(req.params.id) },
      relations: ["recipes", "saved_recipes", "following", "followers"],
    });

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

    res.json({
      user_id: user.user_id,
      username: user.username,
      email: user.email,
      profile_photo: user.profile_photo,
      bio: user.bio,
      created_at: user.created_at,
      recipes: user.recipes,
      saved_recipes_count: user.saved_recipes?.length || 0,
      following_count: user.following?.length || 0,
      followers_count: user.followers?.length || 0,
    });
  } catch (error: any) {
    res.status(500).json({ message: error.message });
  }
};
```

Вывод

В ходе выполнения домашней работы мы выполнили все поставленные задачи.