САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет по

Домашней работе №2

Выполнил:

Суворин Иван

БР 2.2

Проверил:
Добряков Д. И.

Санкт-Петербург

2026 г.

1.1 Функциональные требования

Бэкенд-приложение представляет собой сервис для публикации, поиска и взаимодействия с рецептами. Система должна обеспечивать следующий функционал:

1. Регистрация и авторизация
- Регистрация пользователя по ФИ, email и паролю
- Подтверждение email
- У каждого пользователя свой никнейм и аватар
- Авторизация проходит через вход в систему
- Разграничение ролей: user и admin

2. Рецепты
- Просмотр списка рецептов. Функция доступна всем пользователям (авторизованным и неавторизованным + админ)
- Просмотр детальной информации о рецепте:
    - Название
    - Краткое описание
    - Полное описание
    - Изображение
    - Ингредиенты
    - Шаги приготовления
    - Время приготовления
    - Уровень сложности (easy, medium, hard)
    - Категория
    - Автор

3. Категории и теги
- У каждого рецепта есть одна категория
- У каждого рецепта есть свои теги

4. Работа с рецептом
- Создание рецепта (авторизованный пользователь)
- Редактирование рецепта (автор)
- Удаление рецепта (автор или администратор)
- Публикация рецепта
- Модерация рецепта администратором

5. Фильтрация рецептов
- По категориям
- По тегам

— По уровню сложности
— По времени приготовления
— По названию

6. Ингредиенты и шаги
— Добавление списка ингредиентов:
- Название ингредиента
- Количество
- Единица измерения

— Упорядоченные шаги
— Пошаговое описание приготовления

7. Социальные функции
— Лайк (авторизованный пользователь):
- Возможность поставить
- Возможность убрать
- Подсчет общего количества лайков

— Комментарии (авторизованный пользователь):
- Просмотр комментариев (все)
- Написание комментариев (авторизованный пользователь и администратор)
- Удаление комментария (авторизованный пользователь и администратор)

— Избранное (авторизованный пользователь):
- Добавление в избранное
- Удаление из избранного

8. Личный кабинет пользователя
— Просмотр информации о пользователе:
    - Никнейм (все)
    - ФИ (только себя)
    - Email (только себя)
    - Просмотр созданных рецептов
    - Просмотр избранных рецептов

9. Администрирование (администратор):
— Модерация рецептов (удаление / изменение)
— Удаление комментариев

1.2 Нефункциональные требования

1. Масштабируемость

— Архитектура должна быть рассчитана на последующее расширение функционала
— Поддержка разных типов устройств
— Увеличение нагрузки

2. Производительность

— Оптимизация запросов к БД
— Оптимизация запросов к серверу

3. Безопасность

— Хранение паролей в зашифрованном виде
— Валидация входных данных
— Защита от дублирования:

- 1 пользователь - 1 лайк на рецепт
- Уникальный никнейм

4. Надежность

— Обработка ошибок
— Возврат корректных HTTP-статусов
— Защита от неккоректных запросов
— Сохранение целостности данных

5. Удобство использования

— Единый формат ответов (JSON)
— Swagger

6. Поддерживаемость

— Разделение логики

7. Медиа

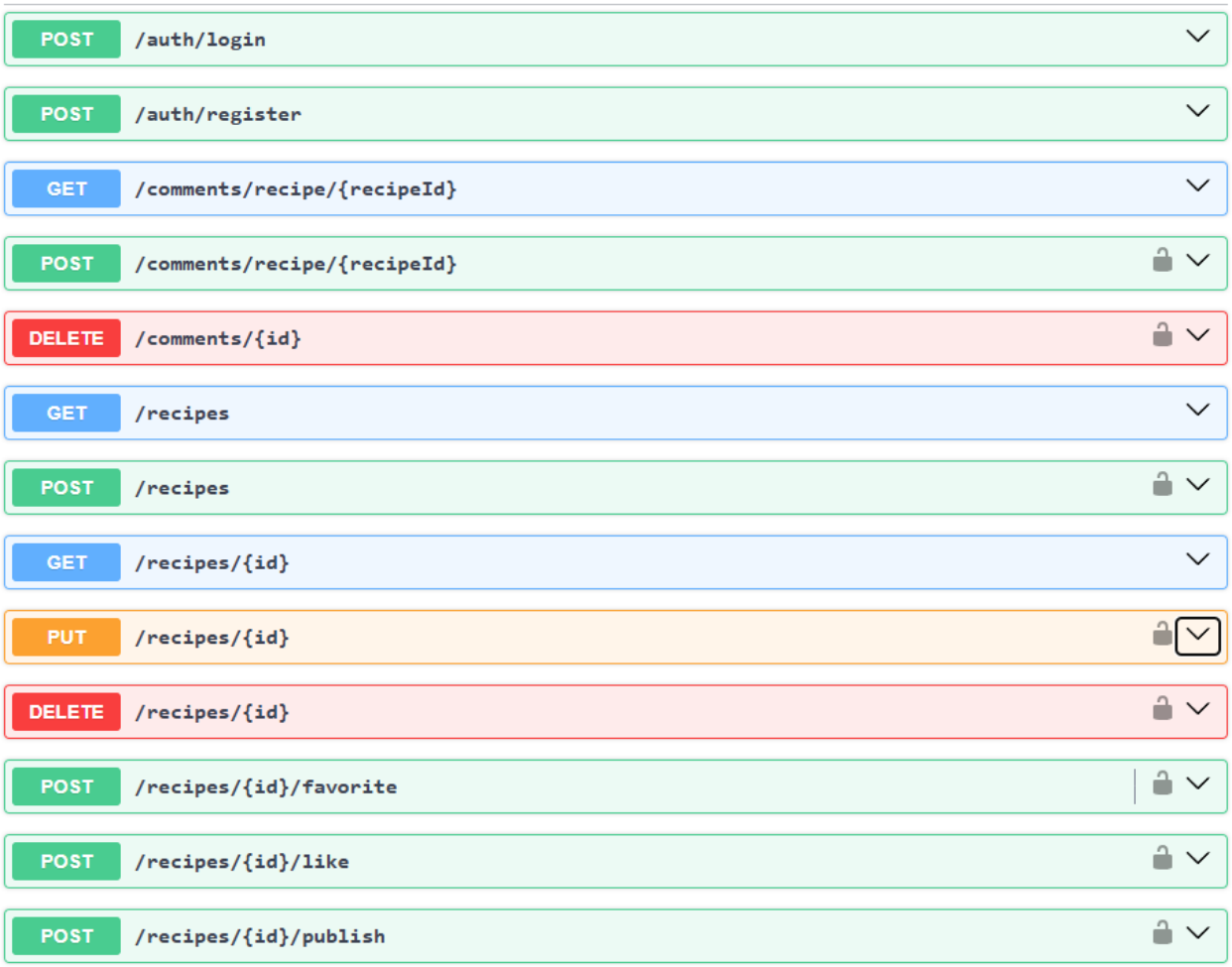
— Изображения не хранятся в БД

2. Формат реализации API, которого вы будете придерживаться: REST API или Backend For Frontend

В ходе работы выбор был сделан в пользу REST API, потому что:

- Простота реализации
- Понимание принципов
- Понятные статус-коды
- Не нужен подбор данных

3. Эндпоинты API в формате OpenAPI-схемы с учётом реализованной диаграммы БД.



4. Опишите тело каждого запроса и тело каждого ответа, продумайте возможные ошибки, возвращаемые из API

POST /auth/login

Request body

```
{
  "email": "string",
  "password": "string"
}
```

Responses (200)

```
{
```

```
  "token": "string"
}
```

POST /auth/registration

Request body

```
{
  "username": "string",
  "email": "string",
  "password": "string",
  "firstName": "string",
  "lastName": "string",
  "avatar": "string"
}
```

Responses

```
{
  "id": 0,
  "username": "string",
  "email": "string",
  "firstName": "string",
  "lastName": "string",
  "avatar": "string",
  "role": "user"
}
```

GET /comments/recipe/{recipeId}

Responses

```
[
  {
    "id": 0,
    "text": "string",
    "user": {
      "id": 0,
      "username": "string",
      "email": "string",
      "firstName": "string",
      "lastName": "string",
      "avatar": "string",
      "role": "user"
    }
  }
]
```

POST /comments/recipe/{recipeId}

Request body

```
{
  "text": "string"
}
```

Responses

```
{
  "id": 0,
  "text": "string",
  "user": {
    "id": 0,
    "username": "string",
    "email": "string",
    "firstName": "string",
    "lastName": "string",
```

```
    "avatar": "string",
    "role": "user"
  }
}
```


DELETE /comments/{id}

Удаляет комментарий по его идентификатору (id). После успешного удаления комментарий исчезает из базы данных и больше не отображается под рецептом. Требуется (Bearer token в заголовке Authorization).

Удалять комментарий могут:

— Автор самого комментария

— Администратор (роль admin)

Параметры:

id (в пути, path) — обязательный, integer (ID комментария).

Тело запроса отсутствует, поскольку это чистый DELETE-запрос без body.

Успешный ответ — 204 No Content. Тело ответа пустое.

Это стандартный код для успешного удаления ресурса, когда сервер подтверждает, что действие выполнено, но возвращать данные не нужно.

GET /recipes

Responses


```
{
  "id": 0,
  "title": "string",
  "shortDescription": "string",
  "fullDescription": "string",
  "difficulty": "easy",
  "cookingTime": 0,
  "author": {
    "id": 0,
    "username": "string",
    "email": "string",
    "firstName": "string",
    "lastName": "string",
    "avatar": "string",
    "role": "user"
  },
  "category": {
    "id": 0,
    "title": "string"
  },
```

```
    "tags": [
      {
        "id": 0,
        "title": "string"
      }
    ],
    "ingredients": [
      {
        "name": "string",
        "quantity": "string"
      }
    ],
    "steps": [
      {
        "order": 0,
        "description": "string"
      }
    ],
    "image": "string",
    "published": true
  }
]
```

POST /recipes

Request body

```
{
  "title": "string",
  "shortDescription": "string",
  "fullDescription": "string",
  "difficulty": "easy",
  "cookingTime": 0,
  "categoryId": 0,
  "tagIds": [
    0
  ],
  "ingredients": [
    {
      "name": "string",
      "quantity": "string"
    }
  ],
  "steps": [
    {
      "order": 0,
      "description": "string"
    }
  ],
  "image": "string"
}
```

Responses

```
{
  "id": 0,
  "title": "string",
  "shortDescription": "string",
  "fullDescription": "string",
  "difficulty": "easy",
  "cookingTime": 0,
  "author": {
    "id": 0,
    "username": "string",
    "email": "string",
    "firstName": "string",
    "lastName": "string",
```

```
    "avatar": "string",
    "role": "user"
  },
  "category": {
    "id": 0,
    "title": "string"
  },
  "tags": [
    {
      "id": 0,
      "title": "string"
    }
  ],
  "ingredients": [
    {
      "name": "string",
      "quantity": "string"
    }
  ],
  "steps": [
    {
      "order": 0,
      "description": "string"
    }
  ],
  "image": "string",
  "published": true
}
```


GET /recipes/{id}

Responses


```
{
  "id": 0,
  "title": "string",
  "shortDescription": "string",
  "fullDescription": "string",
  "difficulty": "easy",
  "cookingTime": 0,
  "author": {
    "id": 0,
    "username": "string",
    "email": "string",
    "firstName": "string",
    "lastName": "string",
    "avatar": "string",
    "role": "user"
  },
  "category": {
    "id": 0,
    "title": "string"
  },
  "tags": [
    {
      "id": 0,
      "title": "string"
    }
  ],
  "ingredients": [
    {
      "name": "string",
      "quantity": "string"
    }
  ],
  "steps": [
    {
```

```
      "order": 0,
      "description": "string"
    }
  ],
  "image": "string",
  "published": true
}
```

PUT /recipes/{id}

Request body

```
{
  "title": "string",
  "shortDescription": "string",
  "fullDescription": "string",
  "difficulty": "easy",
  "cookingTime": 0,
  "categoryId": 0,
  "tagIds": [
    0
  ],
  "ingredients": [
    {
      "name": "string",
      "quantity": "string"
    }
  ],
  "steps": [
    {
      "order": 0,
      "description": "string"
    }
  ],
  "image": "string"
}
```

Responses

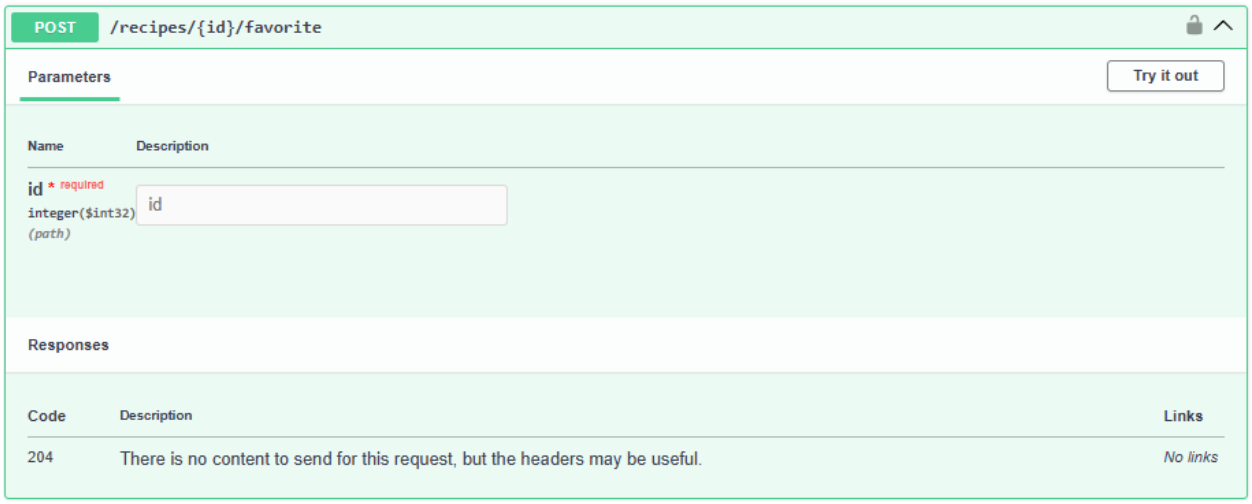
```
{
  "id": 0,
  "title": "string",
  "shortDescription": "string",
  "fullDescription": "string",
  "difficulty": "easy",
  "cookingTime": 0,
  "author": {
    "id": 0,
    "username": "string",
    "email": "string",
    "firstName": "string",
    "lastName": "string",
    "avatar": "string",
    "role": "user"
  },
  "category": {
    "id": 0,
    "title": "string"
  },
  "tags": [
    {
      "id": 0,
      "title": "string"
    }
  ],
  "ingredients": [
    {
      "name": "string",
      "quantity": "string"
    }
  ],
```

```
  "steps": [
    {
      "order": 0,
      "description": "string"
    }
  ],
  "image": "string",
  "published": true
}
```

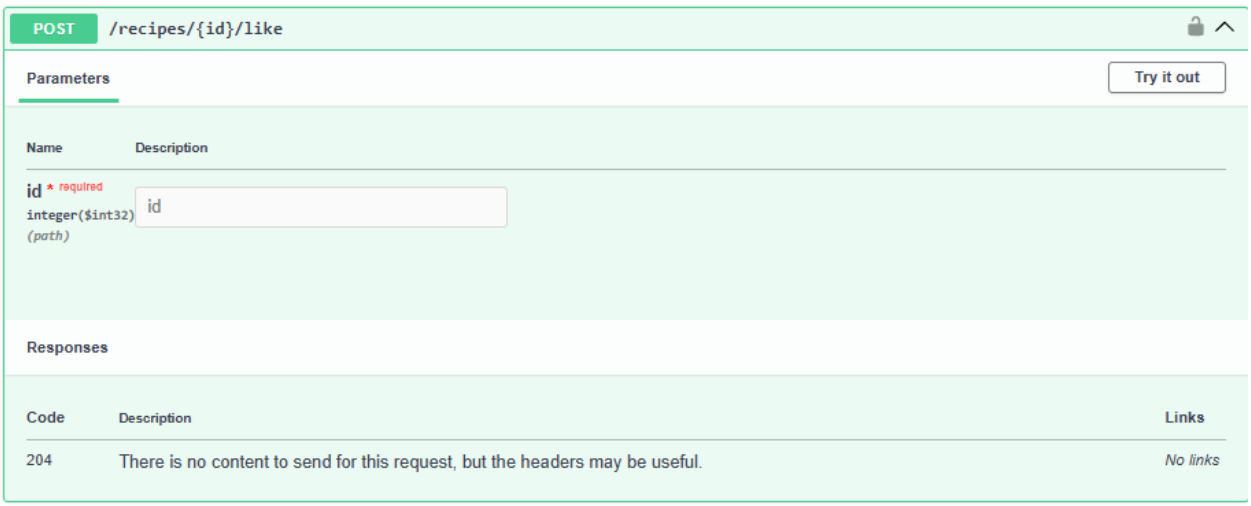

DELETE /recipes/{id}

Аналогично

POST /recipes/{id}/favorite



POST /recipes/{id}/like



Responses


```
{
  "id": 0,
  "title": "string",
```

```
  "shortDescription": "string",
  "fullDescription": "string",
  "difficulty": "easy",
  "cookingTime": 0,
  "author": {
    "id": 0,
    "username": "string",
    "email": "string",
    "firstName": "string",
    "lastName": "string",
    "avatar": "string",
    "role": "user"
  },
  "category": {
    "id": 0,
    "title": "string"
  },
  "tags": [
    {
      "id": 0,
      "title": "string"
    }
  ],
  "ingredients": [
    {
      "name": "string",
      "quantity": "string"
    }
  ],
  "steps": [
    {
      "order": 0,
      "description": "string"
    }
  ],
  "image": "string",
  "published": true
}
```