**Министерство науки и высшего образования**

**Российской Федерации**

**ФЕДЕРАЛЬНОЕ Государственное Автономное**

**образовательное учреждение высшего образования**

**«новосибирский национальный исследовательский государственный университет»**

**Факультет информационных технологий**

Кафедра общей информатики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) Программная инженерия и компьютерные науки

**ОТЧЕТ**

**о прохождении учебной практики, эксплуатационной практики**

(указывается наименование практики)

**Обучающегося** Садриева Владлена Дамировича **группы №** 20201 **курса 3**

(Ф.И.О. полностью)

**Тема задания**: Разработка программного REST интерфейса для клиентской части приложения Talky Chef

**Место прохождения практики:** ФГБУН Институт математики им. С. Л. Соболева СО РАН, Лаборатория теории вычислимости и прикладной логики, 630090, Новосибирская обл., Новосибирск, пр. Академика Коптюга, 4

(полное наименование организации и структурного подразделения, индекс, адрес)

**Сроки прохождения практики:** с 06.02.2023 г. по 27.05.2023 г.

**Руководитель практики   
от профильной организации** Пальчунов Дмитрий Евгеньевич, в.н.с. \_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О. полностью, должность, подпись)

**Руководитель практики от НГУ** Пальчунов Дмитрий Евгеньевич, зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О. полностью, должность, подпись)

**Руководитель ВКР** Пальчунов Дмитрий Евгеньевич, зав. кафедрой

(Ф.И.О. полностью, должность)

**Оценка по итогам защиты отчета:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(неудовлетворительно, удовлетворительно, хорошо, отлично)

**Отчет заслушан на заседании кафедры** общей информатики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование кафедры)

**протокол 16-23 от** «02» июня 2023 г.

Новосибирск 2023

Оглавление

[1 Введение 3](#_Toc136377421)

[2 Необходимое программное обеспечение 3](#_Toc136377422)

[3 Программный интерфейс и документация 3](#_Toc136377423)

[4 Реализация программного интерфейса 5](#_Toc136377424)

[5 Развертывание приложения на сервере 6](#_Toc136377425)

[6 Заключение 6](#_Toc136377426)

[7 Список литературы 7](#_Toc136377427)

# 1 Введение

На сегодняшний день существует много универсальных голосовых помощников, упрощающих жизнь человека, но в узких сферах они оказываются слабы. Поэтому разработка ассистентов под конкретные задачи является актуальной.

Во время приготовления пищи неудобно взаимодействовать с устройством, т.к. руки могут оказаться грязными. Приложение рецептов Talky Chef с внедрением интеллектуального голосового ассистента, разрабатываемое командой, в которой я нахожусь, позволит убрать это неудобство, т.к. можно будет взаимодействовать с приложением голосом.

Преимущества приложения Talky Chef в сравнении с другими голосовыми ассистентами в том, что кроме самого управления голосом будут представлены подборки блюд и большая база данных качественных рецептов.

Целью данной работы является разработка программного REST интерфейса для клиентской части приложения Talky Chef, а также его документирование.

Основные задачи практики:

1. Написание документации программного интерфейса
2. Реализация интерфейса и его тестирование
3. Развертывание приложения на выделенном сервере

Информационная база исследования – данные из открытых источников в сети Интернет.

# 2 Необходимое программное обеспечение

В качестве языка программирования серверной части был выбран язык Java. Данный язык хорошо себя зарекомендовал и широко используется при написании backend части приложений [2, 3]. В качестве системы контроля версий выбран Git, тестирования конечных точек — Postman.

Основным элементом проекта является Spring — универсальный фреймворк с открытым исходным кодом для Java-платформы, включающий в себя различные компоненты: Spring Boot, Spring Data, Spring MVC и т.д.. Данный фреймворк служит контейнером внедрения зависимостей и упрощающает архитектуру приложений и написание кода [10].

В качестве системы сборки использован Gradle, позволяющий с легкостью добавлять к проекту зависимости, собирать и запускать приложение. С помощью него были добавлены нужные компоненты Spring, а также упрощающие написание кода библиотеки: ModelMapper — помогает сопоставить данные между различными по структуре объектами [8] и Lombok — позволяет использовать аннотации, чтобы избавиться от написания шаблонного кода [9].

# 3 Программный интерфейс и документация

В приложении Talky Chef клиент общается с сервером, руководствуясь REST API. API (англ. Application Programming Interface — программный интерфейс приложения) — это описание способов взаимодействия одной программы с другими [1]. REST (Representational State Transfer) — это архитектурный стиль, который накладывает условия на то, как должен работать API [4].

REST API использует HTTP-запросы для доступа к ресурсам и взаимодействия с ними. Основными методами HTTP в REST являются [4]:

* GET — получение ресурса
* POST — добавление ресурса
* PUT — обновление ресурса
* DELETE — удаление ресурса

Данные операции соответствуют CRUD операциям: create, read, update, delete.

Была написана документация API сервера приложения Talky Chef согласно OpenAPI в Swagger editor. OpenAPI — это спецификация для создания и документирования REST API. Она обеспечивает стандартный, не зависящий от языка способ описания и документирования API, что упрощает их создание и использование. Данная спецификация обеспечивает подробный обзор конечных точек API, параметров и ответов [5, 6]. Swagger editor — удобный редактор для написания OpenAPI интерфейса [7].

На данный момент описано большинство нужных CRUD операций (рисунок 1):

* Добавление рецепта
* Обновление рецепта
* Удаление рецепта
* Поиск рецепта по подстроке
* Поиск медиа файла
* …

Пример того, как это выглядит в файле, показан на рисунке 2.

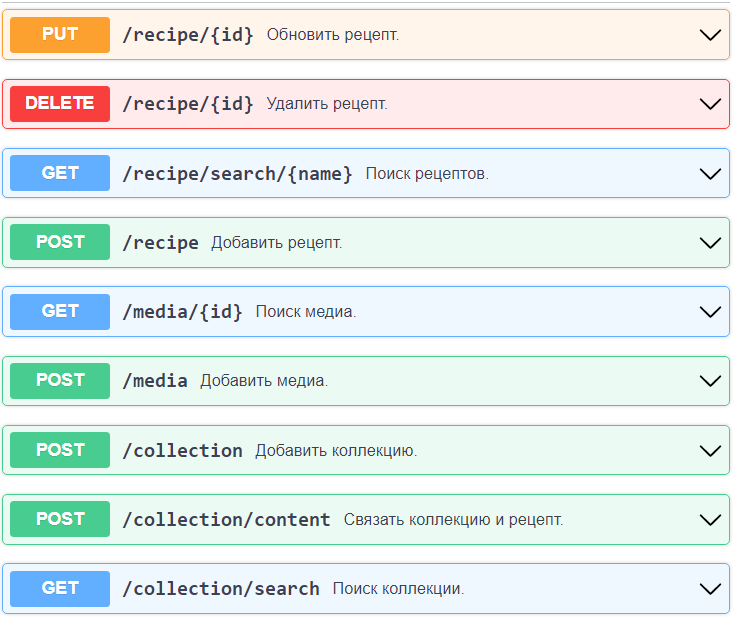
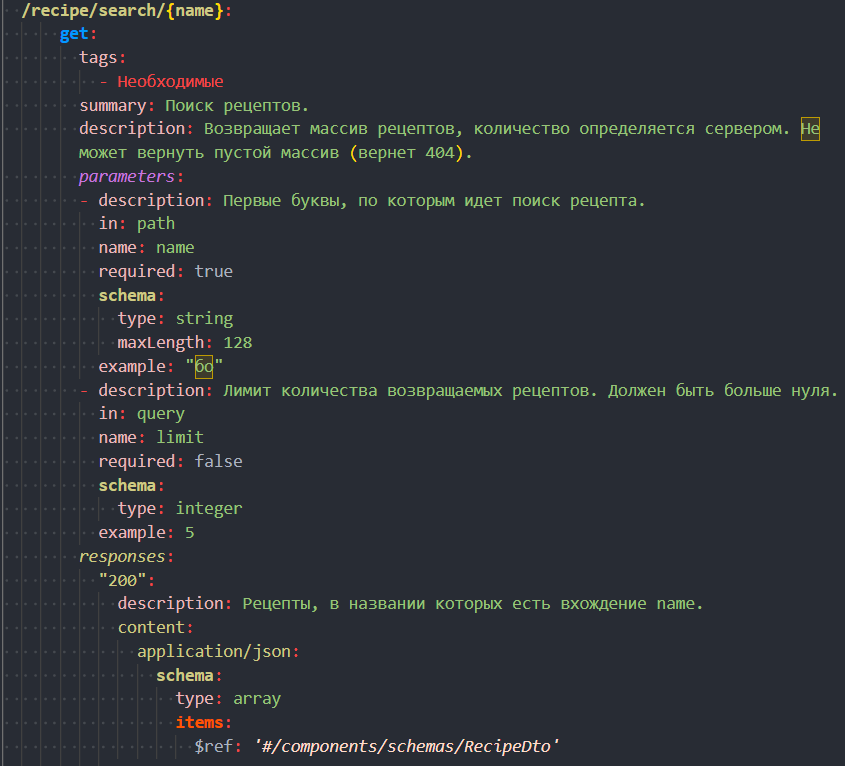


Рисунок 2 – Пример описания API в файле

Рисунок 1 – Описанный программный интерфейс

# 4 Реализация программного интерфейса

Для обработки входящих запросов и возврата ответа в Spring MVC существует понятие контроллера — класс, аннотированный @Controller. Для получения запросов от клиента были созданы интерфейсы, методы которых привязаны к определенным конечным точкам и типу HTTP запроса, а также реализующие их классы.

В контроллерах обычно используются следующие аннотации [11]:

* @RestController — помечает класс как контроллер, который возвращает JSON(JavaScript Object Notation) или XML(Extensible Markup Language)
* @RequestMapping — на классовом уровне позволяет задать единую точку входа в контроллер
* @GetMapping, @PostMapping, @PutMapping, @DeleteMapping —используются для обозначения HTTP запроса, обрабатываемого методом
* @Valid — позволяет проверить параметр на корректность
* @RequestParam — отображает query-параметр запроса в параметр метода
* @RequestBody — означает, что Spring должен десериализовать тело запроса в объект

В Spring MVC HTTP ответ представлен в виде объекта класса ResponseEntity, в котором можно указать необходимые заголовки, код состояния и тело ответа.

Пример метода интерфеса показан на рисунке 3, его реализация в контроллере на рисунке 4.

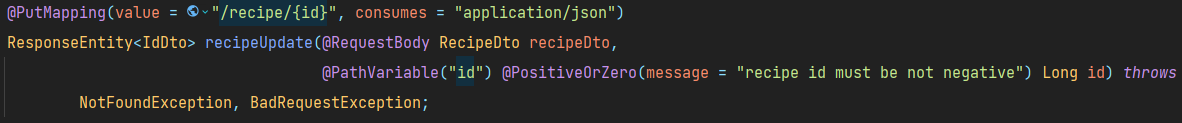


Рисунок 3 – метод обработки запроса обновления рецепта в интерфейсе

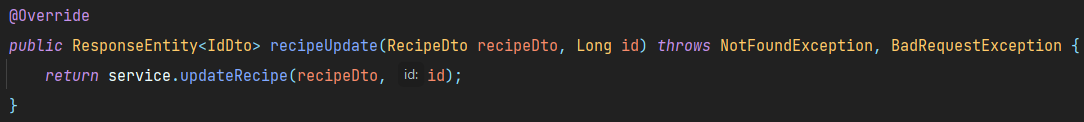


Рисунок 4 – реализация метода обработки запроса обновления рецепта

Объект service на рисунке 4 — это класс, в котором реализована основная бизнес логика, связанная с рецептами (см. рисунок 5). Для данного примера сначала в базе данных по идентификатору ищется рецепт, затем с помощью ModelMapper объект передачи данных из запроса преобразуется в объект-сущность, в который устанавливается id автора рецепта, сортируются по номерам шаги приготовления рецепта и устанавливается распределение ингридиентов. После этого удалаются неиспользуемые медиа файлы, сохраняется обновленный рецепт и отправляется ответ с телом в виде идентификатора рецепта.

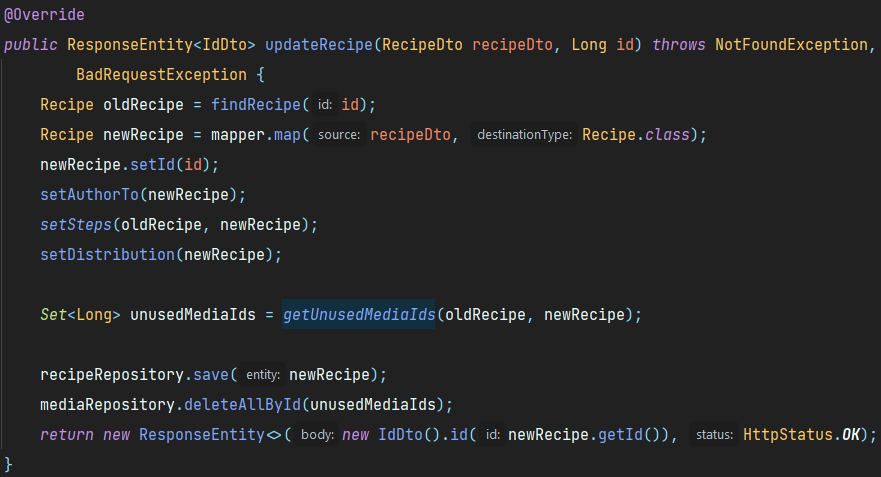


Рисунок 5 – логика обработки запроса на обновление рецепта в сервисе

# 5 Развертывание приложения на сервере

Был арендован выделенный сервер, настроено подключение на него по ssh ключу. После этого был склонирован GitHub репозиторий с проектом и создан .sh скрипт, устанавливающий нужные переменные окружения и запускающий приложение.

После запуска файла скрипта была проверена работоспособность приложения, правильность выведения логов в отдельный файл и проведено тестирование конечных точек с помощью Postman. Postman — инструмент тестирования API. Данный инструмент позволяет отправлять различные HTTP запросы по заданным эндпоинтам и просматривать HTTP ответ.

Была протестирована корректность возвращаемых сервером тел ответов и кодов состояния, обработки запросов с некорретными данными, а также то, что объекты добавляются в базу данных с правильными параметрами.

# 6 Заключение

В результате работы в семестре был проведен анализ спецификации OpenAPI и по ней написана собственная документация программного интерфейса. Данный программный REST интерфейс был реализован и протестирован на серерверной части приложения Talky Chef. Кроме этого, на выделенном сервере было развернуто реализованное приложение. Таким образом, цель работы достигнута.

За время прохождения практики я получил навыки написания документации API, установки ПО, разворачивания проекта на сервере и проверки его работоспособности, а также основы разработки серверной части web-приложения.

# 7 Список литературы

1. What is an API [электронный ресурс] // Aws. URL: <https://aws.amazon.com/what-is/api/> (дата обращения: 14.02.2023)
2. Why java is the best programming language [электронный ресурс] // The server side. URL: <https://www.theserverside.com/feature/5-reasons-why-Java-is-still-the-best-programming-language> (дата обращения: 18.03.2023)
3. Backend programming languages [электронный ресурс] // Back4app. URL: <https://blog.back4app.com/backend-programming-languages-list/> (дата обращения: 18.03.2023)
4. What is restful api [электронный ресурс] // Aws. URL: <https://aws.amazon.com/what-is/restful-api/> (дата обращения: 25.02.2023)
5. Introduction to OpenAPI [электронный ресурс] // Starkovden. URL: <https://starkovden.github.io/introduction-openapi-and-swagger.html> (дата обращения: 25.02.2023)
6. What is OpenAPI [электронный ресурс] // OpenAPIs. URL: <https://www.openapis.org/what-is-openapi> (дата обращения: 25.02.2023)
7. Swagger specification [электронный ресурс] // Swagger. URL: <https://swagger.io/docs/specification/about/> (дата обращения: 20.03.2023)
8. Modelmapper documentation [электронный ресурс] // Modelmapper. URL: <http://modelmapper.org/> (дата обращения: 11.04.2023)
9. Project Lombok documentation [электронный ресурс] // Lombok. URL: <https://projectlombok.org/> (дата обращения: 18.04.2023)
10. Spring framework [электронный ресурс] // Spring. URL: <https://spring.io/projects/spring-framework> (дата обращения: 3.05.2023)
11. Spring controller annotations[электронный ресурс] // Spring. URL: <https://docs.spring.io/spring-framework/docs/3.0.0.M4/reference/html/ch15s03.html> (дата обращения: 4.05.2023)
12. What is DTO [электронный ресурс] // Baeldung. URL: <https://www.baeldung.com/java-dto-pattern> (дата обращения: 10.05.2023)