**Сервис «Чат»**

Оглавление

[Задание 2](#_Toc131976715)

[Бизнес сценарий работы сервиса 3](#_Toc131976716)

[Модель данных 3](#_Toc131976717)

[Описание API системы 3](#_Toc131976718)

[Архитектура 5](#_Toc131976719)

[Тестирование 9](#_Toc131976720)

# Задание

Задание 2

Спроектировать и запрограммировать REST сервис по реализации чата. Сервис должен поддерживать одновременно чат с несколькими пользователями в одной «комнате». Имя пользователя чата можно передавать как параметр, реализовывать аутентификацию не обязательно.

Требования к коду:

·         Написан на Spring Framework

·         Должен собираться gradle

·         Иметь unit тесты

·         Иметь скрипты проверки решения (soapui, curl,  postman и т.д.)

·         Формат сообщении json

·         Приложить описание кода. UML будет плюсом.

# Бизнес сценарий работы сервиса

1. Клиент подключается к чату, получает userId.
2. Клиент отправляет сообщение с параметром userId и текстом сообщения, получает messageId.
3. Клиент может удалять на сервере сообщение messageId или все сразу.
4. Клиент может удалять на сервере клиента по userId или всех клиентов.

# Модель данных

Данные системы представлены 2 типами объектов: ChatUser – пользователи чата, ChatMessage – сообщение в чате.

Доменные сущности:

ChatUser

|  |  |  |
| --- | --- | --- |
| поле | тип | описание |
| id | String | Ид в формате UUID |
| name | String | Имя пользователя |
| active | boolean | Флаг активного пользователя (отправлявшего сообщение в течении конкретного времени) |

ChatMessage

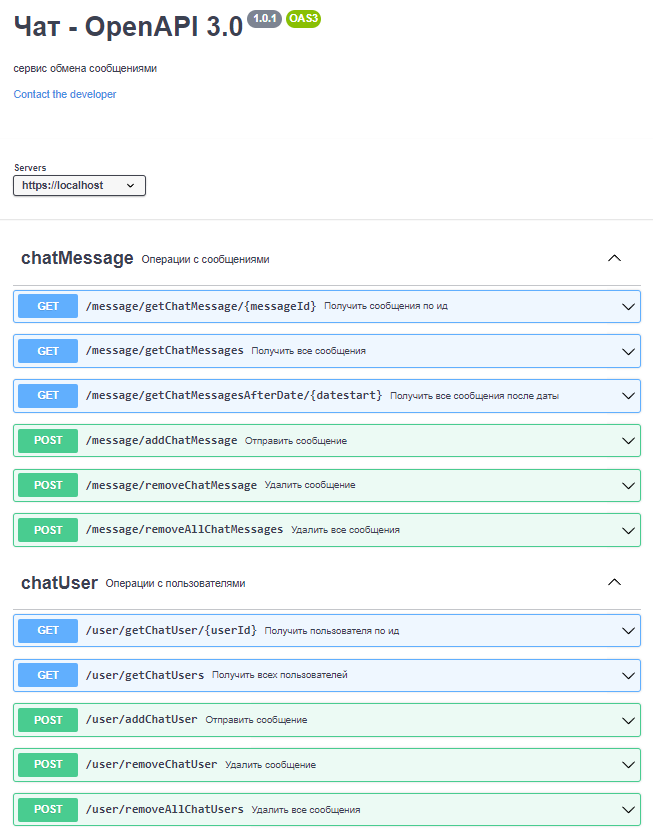
|  |  |  |
| --- | --- | --- |
| поле | тип | описание |
| id | String | Ид в формате UUID |
| dateTime | LocalDateTime | Дата формирования сообщения |
| userId | String | Ид пользователя в формате UUID |
| text | String |  |

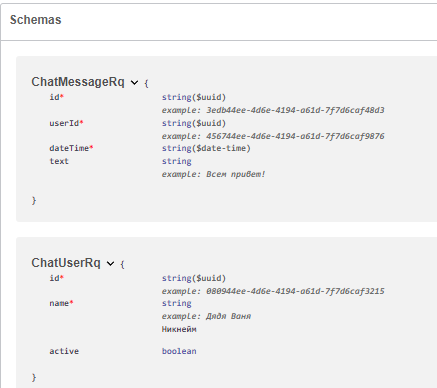
userId - поле для связи объектов, слабая связь.

Для передачи сообщений между клиентом и сервером случат модели dto (viewModel), автогенерируемые по схемам yaml (Swagger).

# Описание API системы

API системы разработано в swaggerUI (openapi: 3.0.3). Проект yaml приложен к проекту.





По разработанным схемам в проект выполнена генерация классов DTO () для выполнения rest-запросов (ChatUserRq, ChatMessageRq).

# Архитектура

Система выполнена в архитектуре Controller Service Repository (архитектурный паттерн, необходимый для разделения ответственности и помогающий соблюдать принципы SOLID).

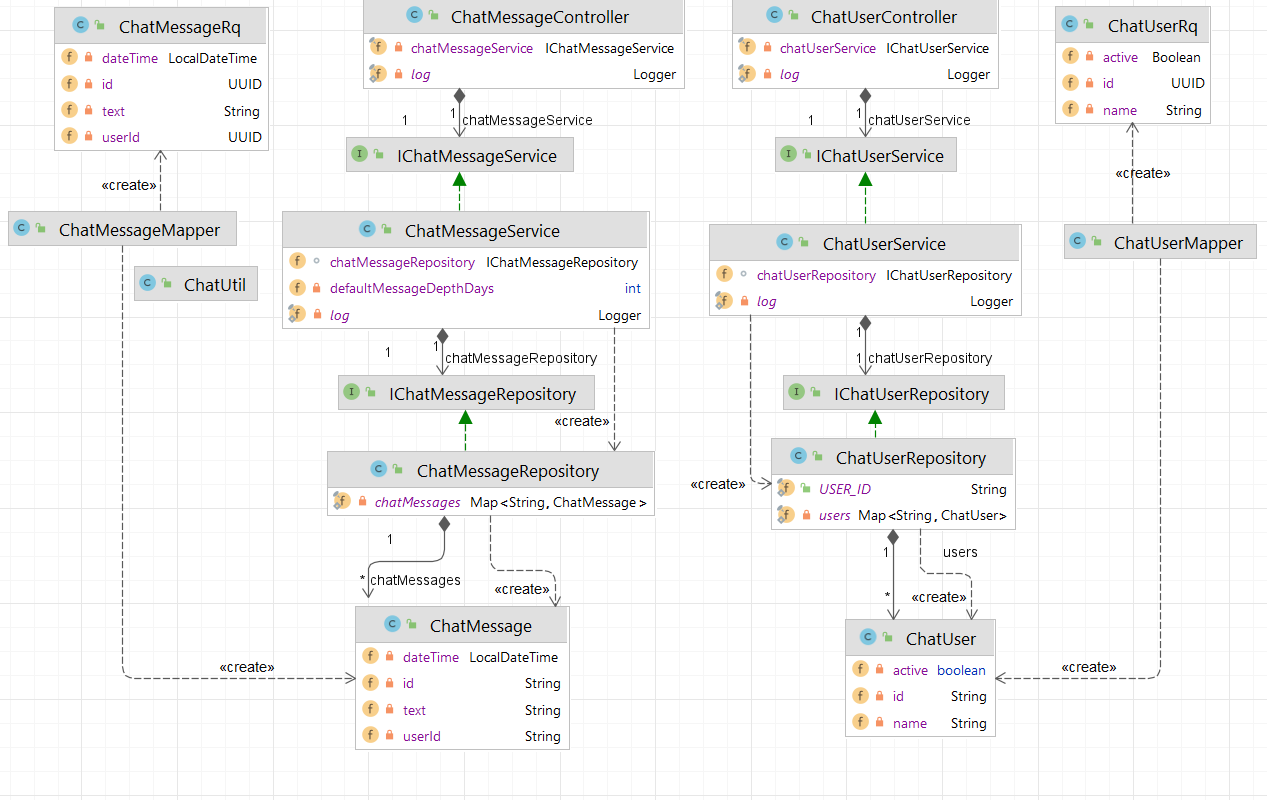


Рис. Диаграмма классов

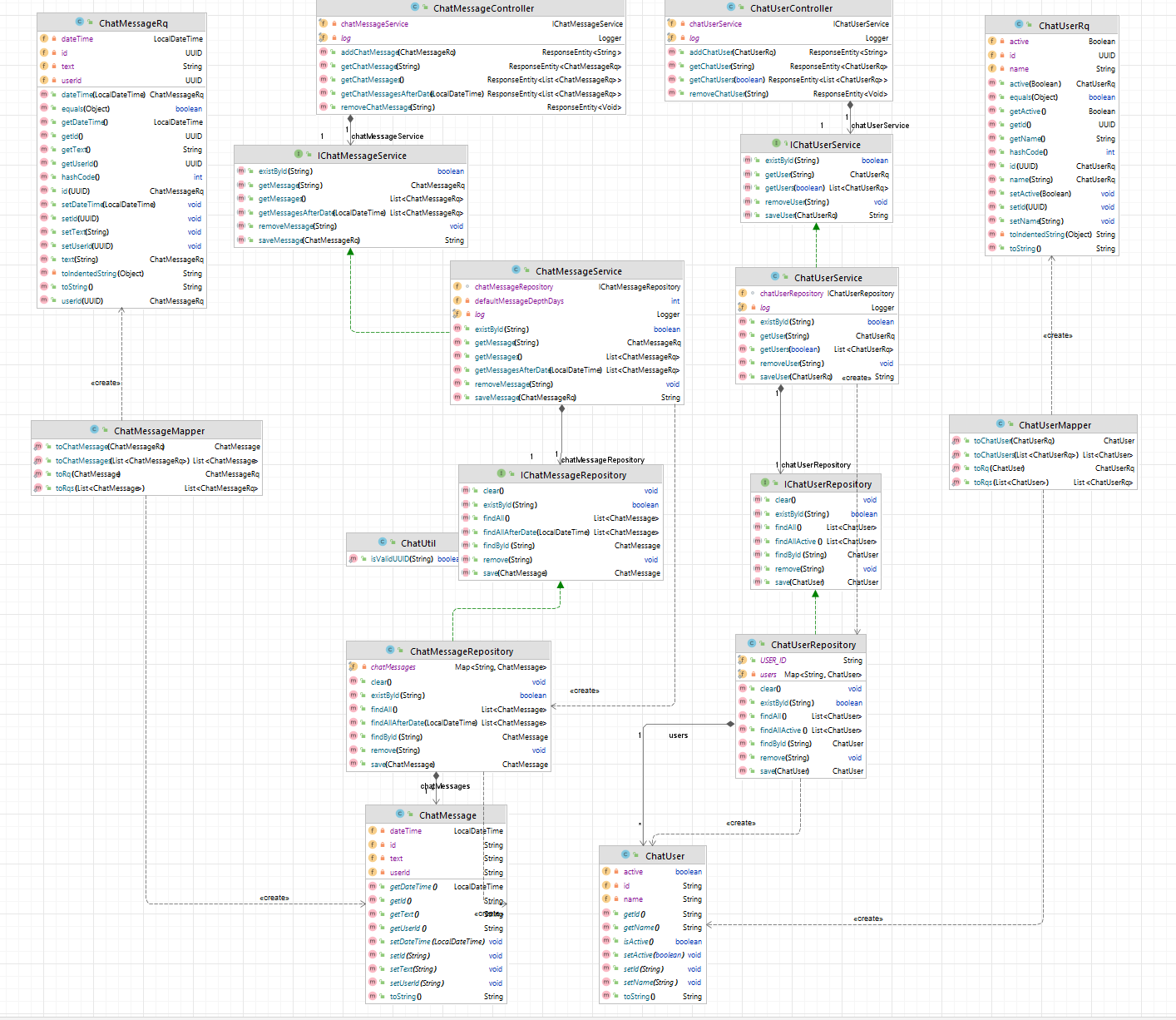


Рис. Диаграмма классов с методами

**Контроллеры**

Контроллеры – классы, отвечающие за обработку запросов. Таким образом ответственность контроллера — это формирование ответа на пользовательские запросы.

Метод контроллера должен:

1. Принять запрос (request)

2. Запустить метод сервиса

3. Обработать исключения или возвращенное сервисом значение

4. Ответить (response) в нужном формате

Система содержит 2 контроллера (RestController):

ChatUserController – обработка запросов по пользователям.

ChatMessageController – обработка запросов по сообщениям.

Перечень запросов и методов контроллера ChatMessageController

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **адрес на веб-сервере** | **Тип** | **Сигнатура метода** | **Описание** |
| 1 | /message/getChatMessages | get | getChatMessages() | Получить список всех сообщений |
| 2 | /message/getChatMessagesAfterDate | get | getChatMessagesAfterDate(@RequestParam LocalDateTime dateStart) | Получить список всех сообщений начиная с даты |
| 3 | /message/getChatMessage | get | getChatMessage(@RequestParam String messageId) | Получить сообщение по ид |
| 4 | /message/addChatMessage | post | addChatMessage(@RequestBody final ChatMessageRq rq) | Отправить сообщение |
| 5 | /message/removeChatMessage | post | removeChatMessage(@RequestBody String id) | Удалить сообщение |
| 6 | /message/removeAllChatMessages | post | removeAllChatMessages() | Удалить все сообщения |

Перечень запросов и методов контроллера ChatUserController

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **адрес на веб-сервере** | **Тип** | **Сигнатура метода** | **Описание** |
| 1 | /user/getChatUsers | get | getChatUsers(@RequestParam boolean onlyActive) | Получить список всех пользователей (с возможностью только активных) |
| 2 | /user/getChatUser | get | getChatUser(@RequestParam String userId) | Получить пользователя по ид |
| 3 | / user/addChatUser | post | addChatMessage(@RequestBody final ChatMessageRq rq) | Добавить пользователя |
| 4 | /user/removeChatUser | post | removeChatUser(@RequestBody String id) | Удалить пользователя |
| 5 | /user/removeAllChatUsers | post | removeAllChatUsers() | Удалить всех пользователей |

**Сервисы**

(отвечают за бизнес логику и ее переиспользование между компонентами, выполнение маппирования объектов доменных моделей и моделей DTO):

ChatMessageService

ChatUserService

**Репозитории** отвечает за получение данных из внешних источников, такие как база данных, локальное хранилище и пр:

ChatMessageRepository

ChatUserRepository

**Хранилище** реализовано в виде статичных членов класса репозиториев при помощи хеш-таблиц.

private static final Map<String, ChatMessage> *chatMessages* = new Hashtable<>();

Взаимодействие между компонентами показано на примере запроса на получение списка сообщений при помощи диаграммы последовательностей UML.

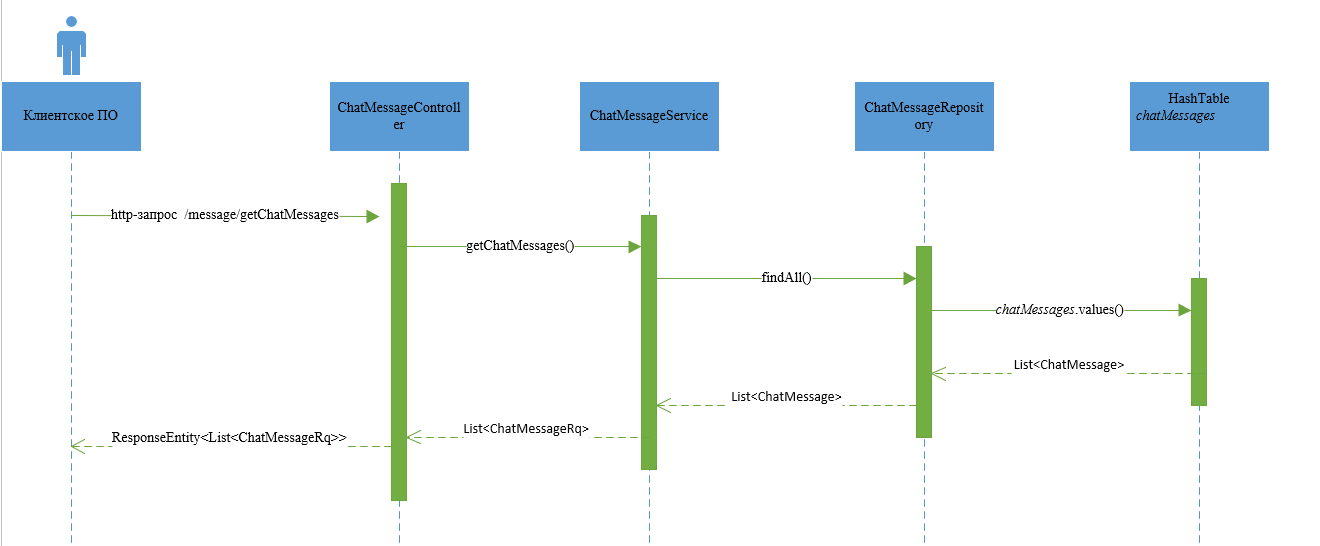


Рис. Получения списка сообщений

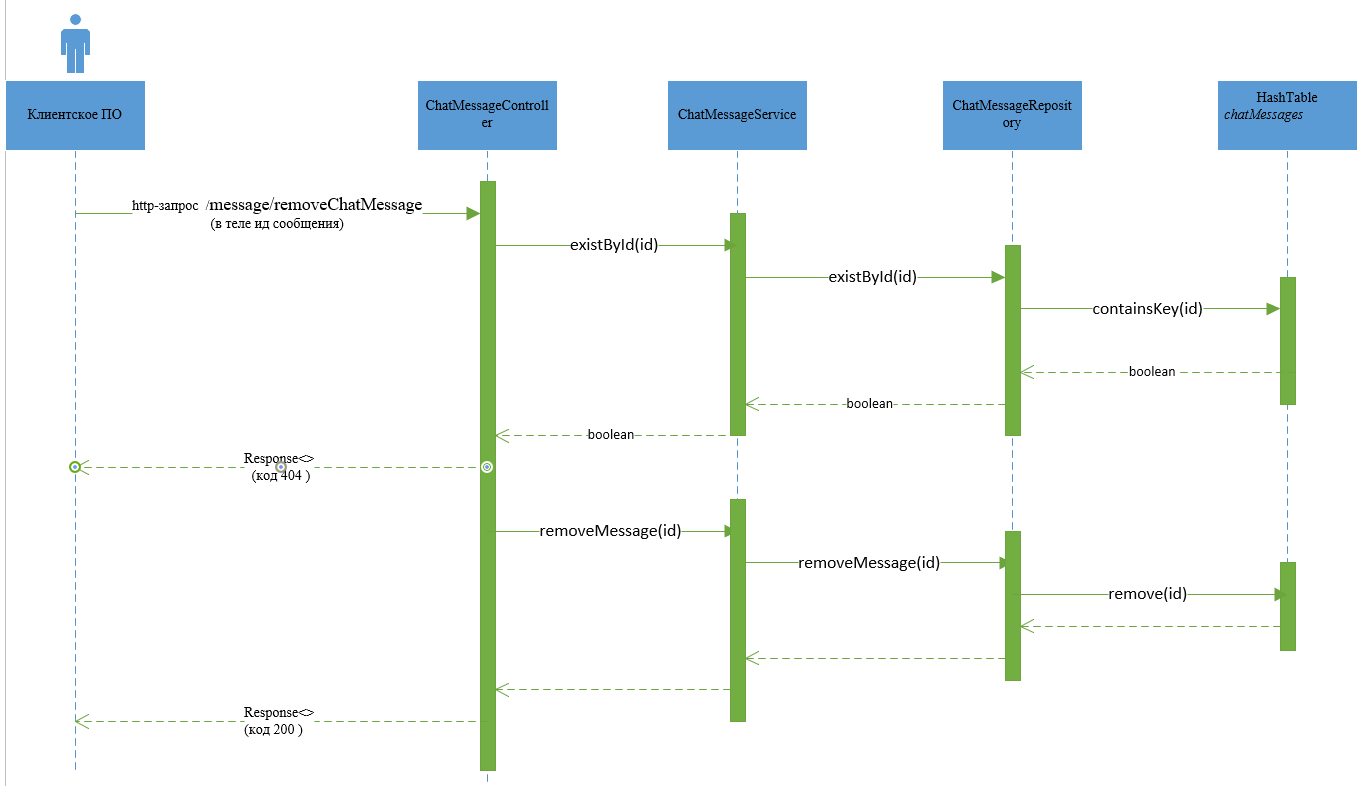
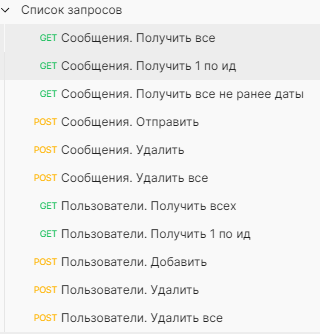


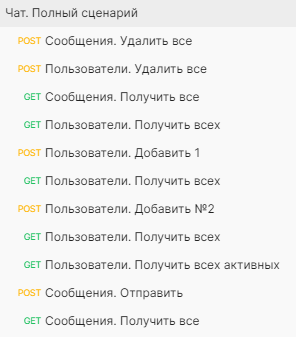
Рис. Удаление сообщения

# Тестирование

В рамках тестирования в проект добавлены модульные тесты на контроллеры, сервисы, репозитории, мапперы.

Также в рамках функционального тестирования выполнен проеткт postman, включающих запросы и последовательный сценарий с тестированием выполенения.





Материалы по тестированию (в формате Postman) представлены в открытом проекте Postman, а также размещены в репозитории Git проекта.

|  |  |
| --- | --- |
| Все запросы | <https://crimson-comet-400065.postman.co/workspace/%D0%93%D0%B0%D0%B7%D0%9F%D1%80%D0%BE%D0%BC%D0%91%D0%B0%D0%BD%D0%BA~e0337cf5-2190-436b-b825-90990968176e/collection/11236295-7b18728a-cd7d-4931-9aed-a175a11ddd50?action=share&creator=11236295> |
| Полный сценарий работы для выполнения со скриптами проверки | <https://crimson-comet-400065.postman.co/workspace/%D0%93%D0%B0%D0%B7%D0%9F%D1%80%D0%BE%D0%BC%D0%91%D0%B0%D0%BD%D0%BA~e0337cf5-2190-436b-b825-90990968176e/collection/11236295-2b42b52a-fb52-497f-a1b3-9e88732e6c11?action=share&creator=11236295> |

В проект добавлен пакет log4j2 для настройки логирования в консоль и файл.