Оглавление

Бизнес требования

1. User Story

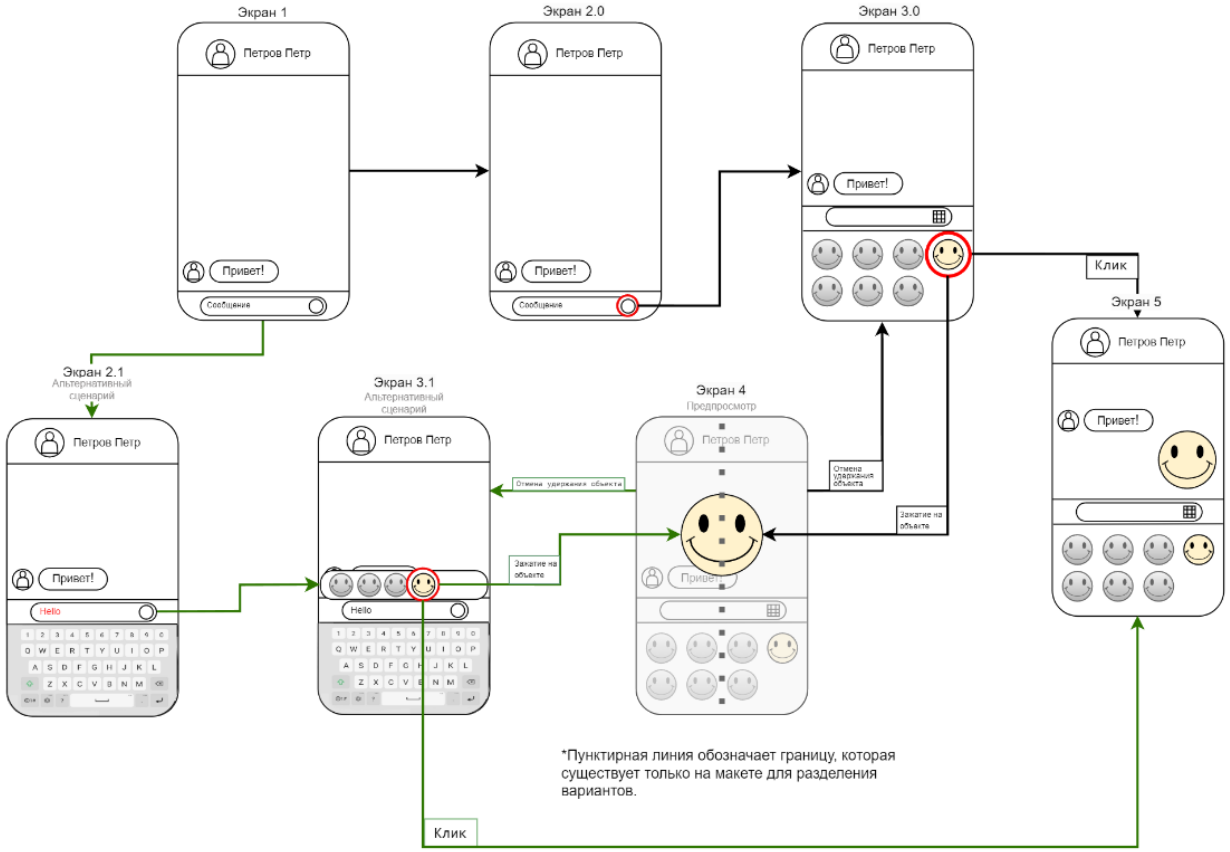
В виде пользовательской истории:

Я, как пользователь мобильного приложения ВКонтакте, **хочу иметь возможность** отправлять стикеры, **для того чтобы** быстрее отвечать на сообщения и передавать эмоции.

В виде таблицы:

Роль	Возможность	Цель
Пользователь мобильного приложения ВКонтакте	Отправлять стикеры	Быстрее отвечать на сообщения и передавать эмоции

2. Макеты

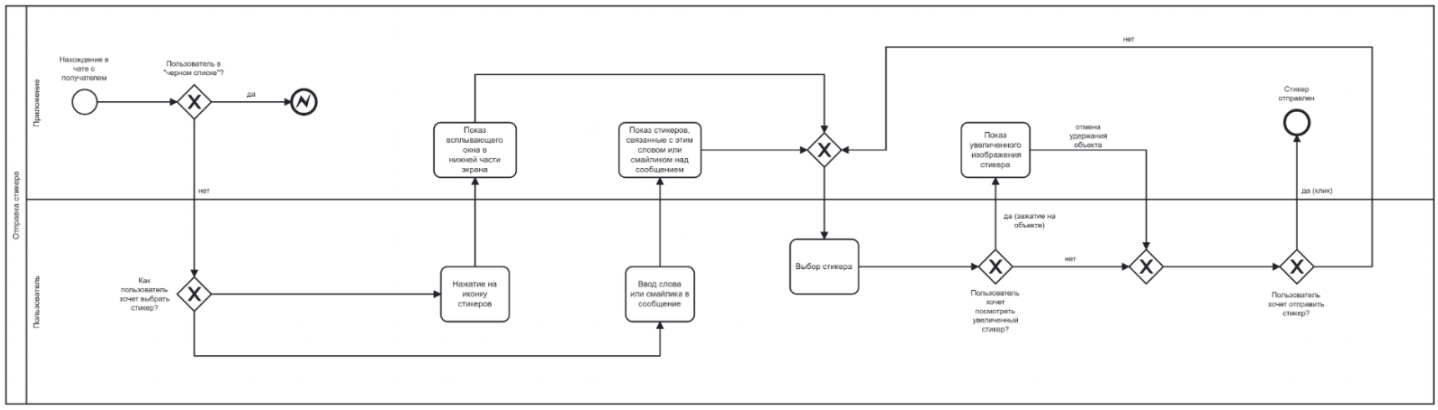


3. Use Case

Заголовок	Отправка стикера в чате приложения
Акторы	Пользователь
Предусловие	Пользователь зарегистрирован и авторизирован в приложении, а также находится во вкладке "Сообщения"
Ограничения	Пользователь не находится в "черном списке" у получателя
Триггер	Пользователь нажимает на иконку выбора стикеров, расположенную рядом с полем ввода сообщения
Исключительный сценарий	До основного и альтернативного сценария: Если пользователь в черном списке, система выводит сообщение об ошибке. Никакие действия совершить нельзя.

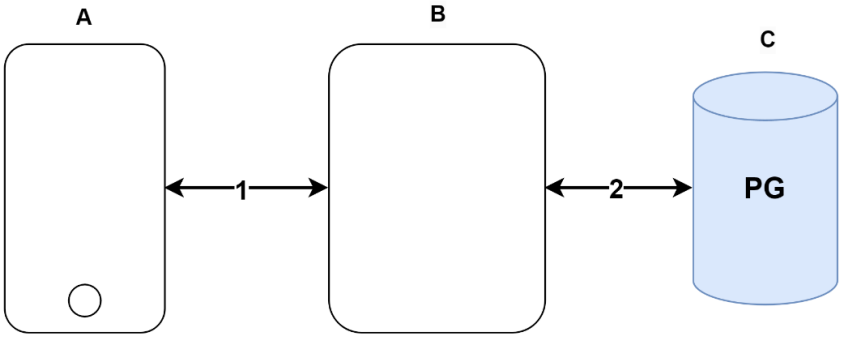
Основной сценарий	1. Пользователь нажимает на иконку выбора стикеров, расположенную рядом с полем ввода сообщения. (Экран 2.0) 2. Система показывает всплывающее окно в нижней части экрана, где можно выбрать стикер. (Экран 3.0) 3. Пользователь кликает по стикеру. 4. Система отправляет стикер. (Экран 5) **Критерий успеха:** Стикер успешно отображается в чате.
Альтернативный сценарий	1а. Пользователь вводит слово или смайлик в сообщение. (Экран 2.1) 2а . Система предлагает стикеры, связанные с этим словом или смайликом над сообщением. (Экран 3.1) -- Переход к шагу 3 основного сценария.

4. BPMN



Функциональные требования

1. Архитектура



A - фронтенд, мобильное приложение

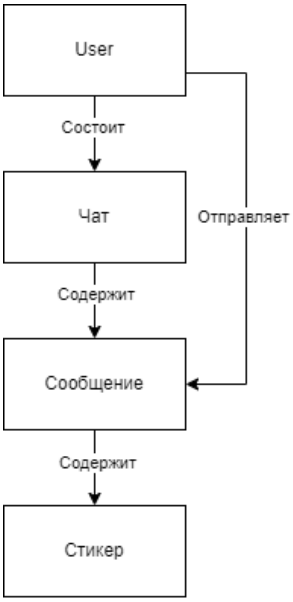
B - бэкенд/сервер

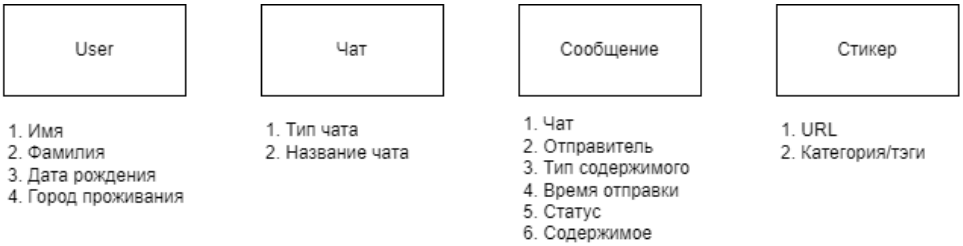
C - реляционная база данных, PG

1 - протокол взаимодействия HTTP/HTTPS

2 - протокол взаимодействия TCP/IP

2. Модель данных





Объект User

Атрибут	Описание
Имя	
Фамилия	
Дата рождения	В формате дд.мм.гггг
Город проживания	Название города (например, Москва)

Объект Чат

Атрибут	Описание
Тип чата	**Личный чат:** 1 получатель **Групповой чат:** от 2 до 500 получателей включительно
Название чата	**Личный чат:** название формируется автоматически в формате "Фамилия Имя" этого получателя **Групповой чат:** название задается пользователем (название чата должно быть от 1 до 50 символов)

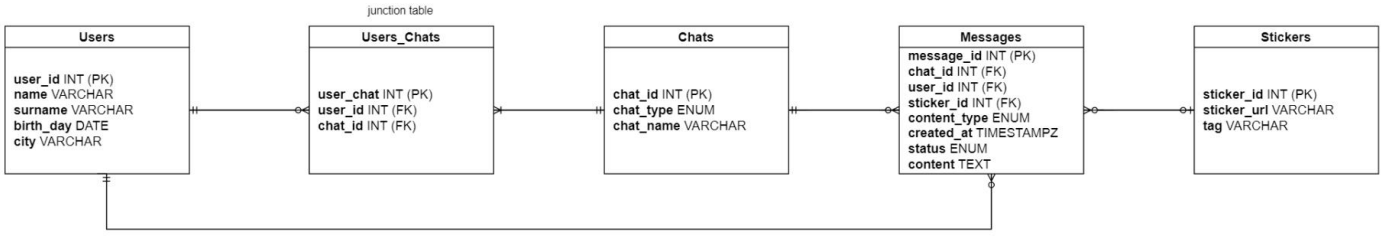
Объект Сообщение

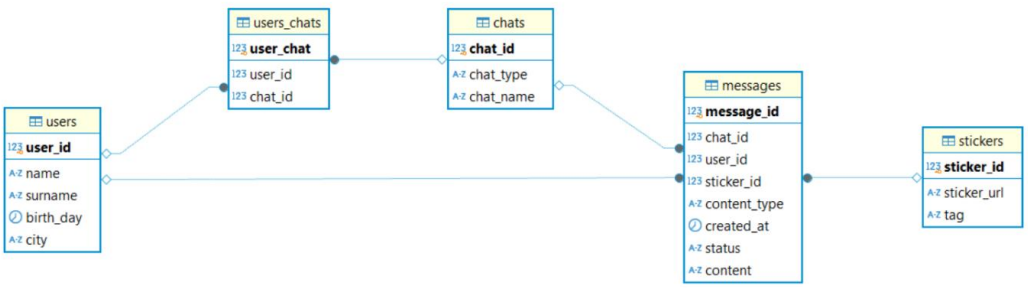
Атрибут	Описание
Чат	Получатель, тот, кто имеет доступ к чату
Отправитель	User
Тип содержимого	Текст, стикер
Время отправки	В виде чч:мм дд.мм.гггг
Статус	Статус сообщения (отправлено, прочитано) **Отправлено**: Этот статус присваивается сразу после отправки сообщения **Прочитано**: Личный чат: сообщение переходит в статус "прочитано", когда получатель открывает чат и видит сообщение Групповой чат: сообщение переходит в статус "прочитано", когда один из участников чата его просмотрел
Содержимое	Место, где хранятся данные, которые пользователь отправил в сообщении (текст/стикер). Это содержимое, будет отображаться в чате для получателей

Объект Стикер

Атрибут	Описание
URL	Путь к изображению стикера
Категория/тэги	Данные для удобства поиска и рекомендаций (например, "смешные", "животные"). У стикера есть только одна категория

3. ERD



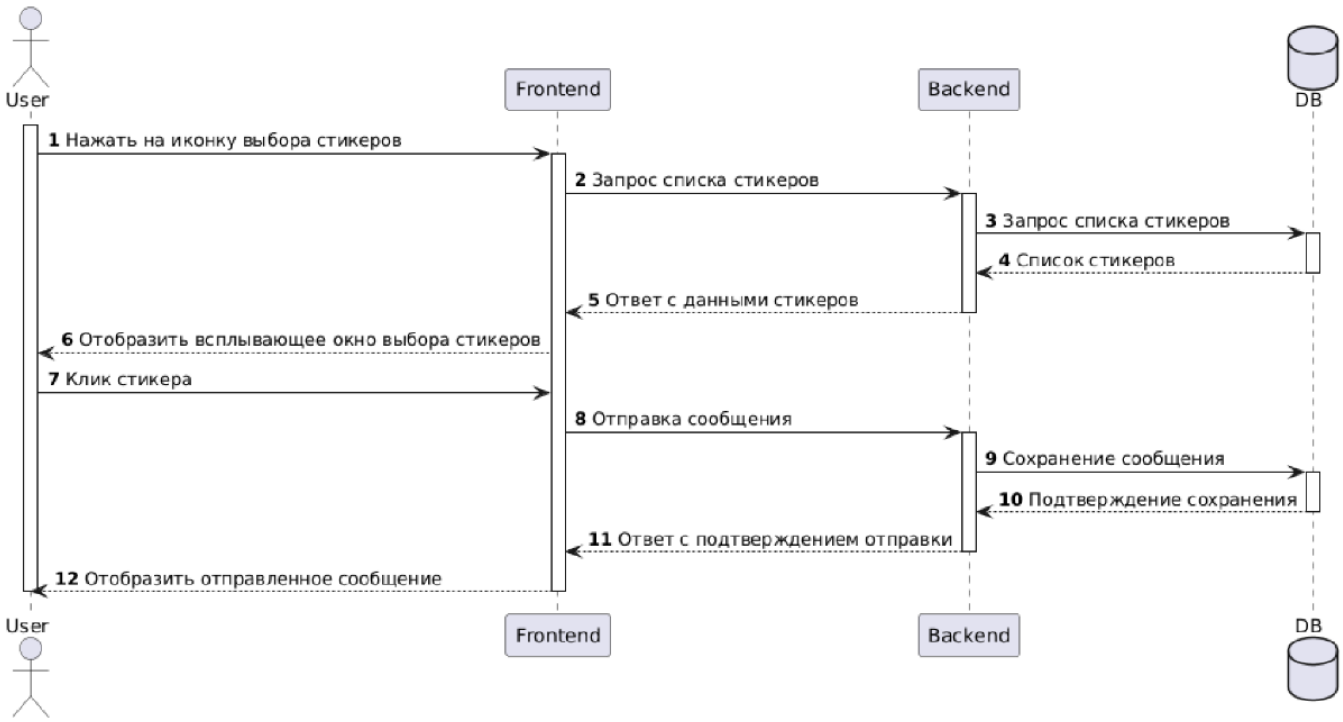


```sql
CREATE TYPE chat_type_enum AS ENUM ('private', 'group');
CREATE TYPE content_type_enum AS ENUM ('text', 'sticker');
CREATE TYPE status_enum AS ENUM ('sent', 'read');

-- таблица пользователей
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    birth_day DATE,
    city VARCHAR(100)
);

-- таблица чатов
CREATE TABLE chats (
    chat_id SERIAL PRIMARY KEY,
    chat_type chat_type_enum NOT NULL,
    chat_name VARCHAR(100)
);

-- таблица стикеров
CREATE TABLE stickers (
    sticker_id SERIAL PRIMARY KEY,
    sticker_url VARCHAR(255),
    tag VARCHAR(100)
);

-- таблица сообщений
CREATE TABLE messages (
    message_id SERIAL PRIMARY KEY,
    chat_id INT,
    user_id INT,
    sticker_id INT,
    content_type content_type_enum NOT NULL,
    created_at TIMESTAMPTZ,
    status status_enum NOT NULL,
    content TEXT,
    CONSTRAINT fk_messages_chat_id FOREIGN KEY (chat_id) REFERENCES chats(chat_id),
    CONSTRAINT fk_messages_user_id FOREIGN KEY (user_id) REFERENCES users(user_id),
    CONSTRAINT fk_messages_sticker_id FOREIGN KEY (sticker_id) REFERENCES stickers(sticker_id)
);

-- junction table (техническая таблица)
CREATE TABLE users_chats (
    user_chat SERIAL PRIMARY KEY,
    user_id INT,
    chat_id INT,
    CONSTRAINT fk_users_chats_user_id FOREIGN KEY (user_id) REFERENCES users(user_id),
    CONSTRAINT fk_users_chats_chat_id FOREIGN KEY (chat_id) REFERENCES chats(chat_id)
);
```

```sql
-- пользователи
INSERT INTO users (name, surname, birth_day, city)
VALUES
('Илья', 'Иванов', '1999-01-01', 'Москва'),
('Дарья', 'Сидорова', '1985-05-12', 'Минск'),
('Елизавета', 'Петрова', '2000-03-15', 'Махачкала'),
('Роман', 'Жуков', '2002-07-08', 'Уфа');

-- чаты
INSERT INTO chats (chat_id, chat_type, chat_name)
VALUES
(1, 'private', 'Сидорова Дарья'), -- личный чат
(2, 'group', 'Группа'), -- групповой чат с 3 участниками
(3, 'private', 'Петрова Елизавета'), -- личный чат
(4, 'group', 'Союз'); -- групповой чат с 4 участниками

-- связь между пользователями и чатами
INSERT INTO users_chats (user_id, chat_id)
VALUES
(1, 1), (2, 1), -- личный чат между Ильей и Дарьей
(3, 2), (4, 2), (1, 2), -- групповой чат с тремя участниками (Елизавета, Роман, Илья)
(1, 3), (3, 3), -- личный чат между Ильей и Елизаветой
(2, 4), (3, 4), (4, 4), (1, 4); -- групповой чат с четырьмя участниками (Дарья, Елизавета, Роман, Илья)

-- стикеры
INSERT INTO stickers (sticker_url, tag)
VALUES
('http://example.com/sticker1.png', 'смешной'),
('http://example.com/sticker2.png', 'крутой'),
('http://example.com/sticker3.png', 'злой'),
('http://example.com/sticker4.png', 'счастливый');

-- сообщения
INSERT INTO messages (chat_id, user_id, sticker_id, content_type, created_at, status, content)
VALUES
-- private, text, sent
(1, 1, NULL, 'text', NOW(), 'sent', 'Привет, Дашаа'),
-- private, text, read
(1, 2, NULL, 'text', NOW(), 'read', 'Ку, Илюха'),
-- group, sticker, sent
(2, 3, 1, 'sticker', NOW(), 'sent', NULL),
-- group, sticker, read
(2, 4, 2, 'sticker', NOW(), 'read', NULL),
-- private, sticker, sent
(3, 1, 3, 'sticker', NOW(), 'sent', NULL),
-- private, sticker, read
(3, 3, 4, 'sticker', NOW(), 'read', NULL),
-- group, text, sent
(4, 2, NULL, 'text', NOW(), 'sent', 'Это сообщение от Дашки'),
-- group, text, sent
(4, 3, NULL, 'text', NOW(), 'sent', 'Лиза передает приветы'),
-- group, text, sent
(4, 4, NULL, 'text', NOW(), 'sent', 'Рома написал это сообщение'),
-- group, text, sent
(4, 1, NULL, 'text', NOW(), 'sent', 'Я Илья'),
-- group, text, read
(4, 2, NULL, 'text', NOW(), 'read', 'Всем привееееет!'),
-- group, text, read
(4, 3, NULL, 'text', NOW(), 'read', 'Я пон'),
-- group, text, read
(4, 4, NULL, 'text', NOW(), 'read', 'Пока пока');
```

4. Sequence

```
@startuml

autonumber

actor User as User
participant "Frontend" as Frontend
participant "Backend" as Backend
database "DB" as DataBase

activate User
User -> Frontend: Нажать на иконку выбора стикеров

activate Frontend
Frontend -> Backend: Запрос списка стикеров

activate Backend
Backend -> DataBase: Запрос списка стикеров

activate DataBase
DataBase --> Backend: Список стикеров
deactivate DataBase

Backend --> Frontend: Ответ с данными стикеров
deactivate Backend

Frontend --> User: Отобразить всплывающее окно выбора стикеров
User -> Frontend: Клик стикера

Frontend -> Backend: Отправка сообщения

activate Backend
Backend -> DataBase: Сохранение сообщения
activate DataBase
DataBase --> Backend: Подтверждение сохранения
deactivate DataBase

Backend --> Frontend: Ответ с подтверждением отправки
deactivate Backend

Frontend --> User: Отобразить отправленное сообщение
deactivate Frontend
deactivate User

@enduml
```



Шаг	Взаимодействие	Описание
1	**User -> Frontend:** Нажать на иконку выбора стикеров	Пользователь инициирует процесс выбора стикеров (триггер)
2	**Frontend -> Backend:** Запрос списка стикеров	Фронтенд отправляет запрос на бэкенд, чтобы получить список доступных стикеров
3	**Backend -> DataBase:** Запрос списка стикеров	Бэкенд делает запрос к базе данных для получения списка доступных стикеров
4	**DataBase --> Backend:** Список стикеров	База данных возвращает список стикеров, который бэкенд использует для дальнейшей обработки
5	**Backend --> Frontend:** Ответ с данными стикеров	Бэкенд отправляет список стикеров обратно фронтенду
6	**Frontend --> User:** Отобразить всплывающее окно выбора стикеров	Фронтенд отображает окно выбора стикеров для пользователя, чтобы он мог выбрать нужный стикер
7	**User -> Frontend:** Клик по стикеру	Пользователь кликает на один из стикеров в всплывающем окне
8	**Frontend -> Backend:** Запрос на отправку сообщения	Фронтенд отправляет запрос на бэкенд с выбранным стикером
9	**Backend -> DataBase:** Сохранение сообщения	Бэкенд сохраняет сообщение (стикер) в базе данных
10	**DataBase --> Backend:** Подтверждение сохранения	База данных подтверждает успешное сохранение сообщения
11	**Backend --> Frontend:** Ответ с подтверждением отправки	Бэкенд возвращает подтверждение успешной отправки сообщения
12	**Frontend --> User:** Отобразить отправленное сообщение	Фронтенд отображает отправленное сообщение в интерфейсе пользователя

5. Документация API в табличном виде

Request

GET /stickers

Запрос списка стикеров

Название параметра	Принадлежность	Тип данных	Описание	Обязательность параметра
page	query	int	Текущая страница для пагинации	нет
limit	query	int	Количество элементов на странице для пагинации	нет
tag	query	string	Данные для удобства поиска	нет

Response 200 OK

Название параметра			Принадлежность	Тип данных	Описание	Обязательность параметра
stickers			body	array	Массив информацией о стикерах	да
	sticker		body	object	Объект, содержащий информацию о стикерах	нет
		sticker_id	body	int	Уникальный признак объекта «стикер»	да
		sticker_url	body	string	Путь к изображению стикера	да
		tag	body	string	Данные для удобства поиска	да
pagination			body	object	Объект, содержащий информацию о пагинации	да
	page		body	int	Текущая страница	да
	total_items		body	int	Общее количество стикеров	да
	limit		body	int	Установленный лимит	да

Request

POST /messages

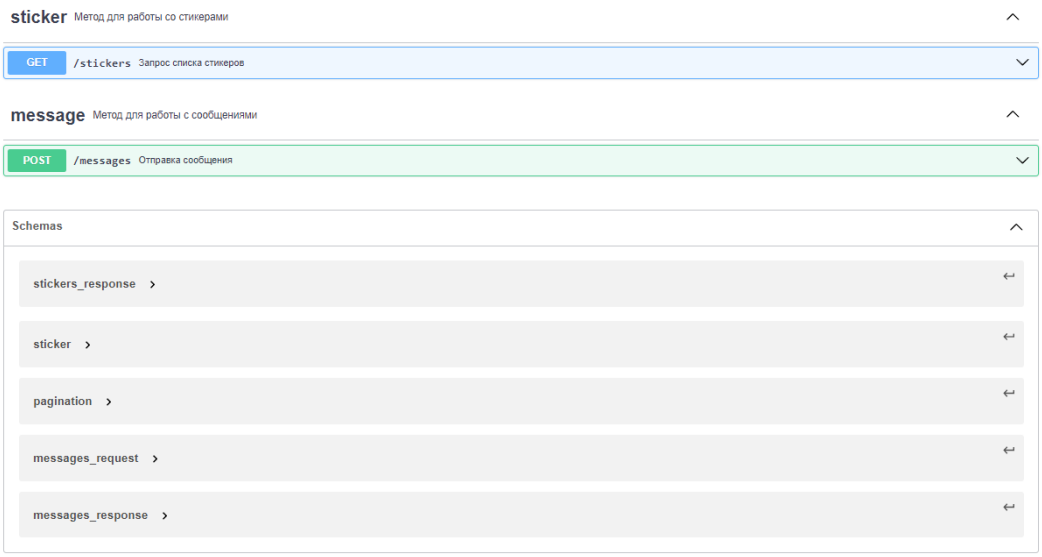
Отправка сообщения

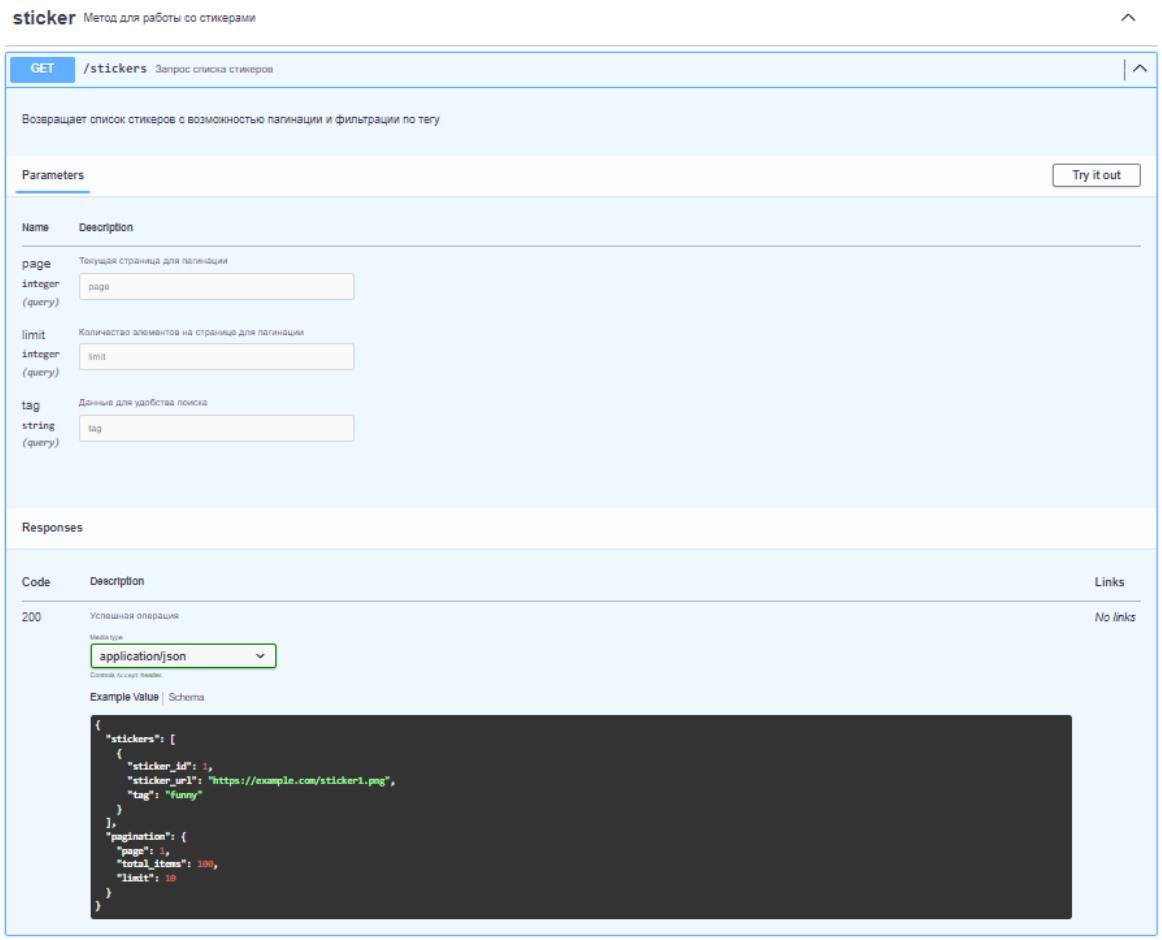
Название параметра	Принадлежность	Тип данных	Описание	Обязательность параметра
chat_id	body	int	Уникальный признак объекта «чат»	да
content_type	body	string	Тип содержимого ("text"/"sticker")	да
sticker_id	body	int	Уникальный признак объекта «стикер»	нет
content	body	string	Содержание текстового сообщения	нет

Response 201 Created

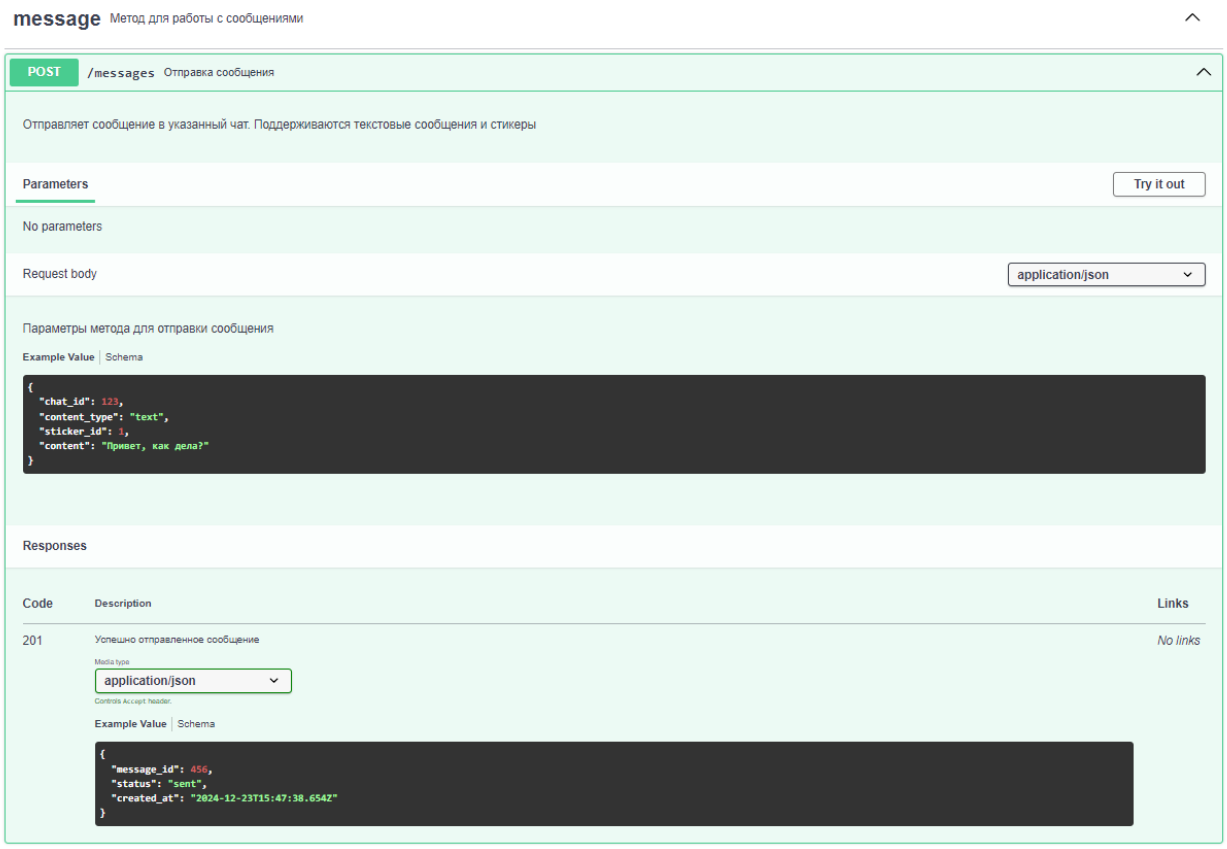
Название параметра	Принадлежность	Тип данных	Описание	Обязательность параметра
message_id	body	int	Уникальный признак объекта «сообщение»	да
status	body	string	Статус сообщения "sent"	да
created_at	body	timestampz	Время отправки	да

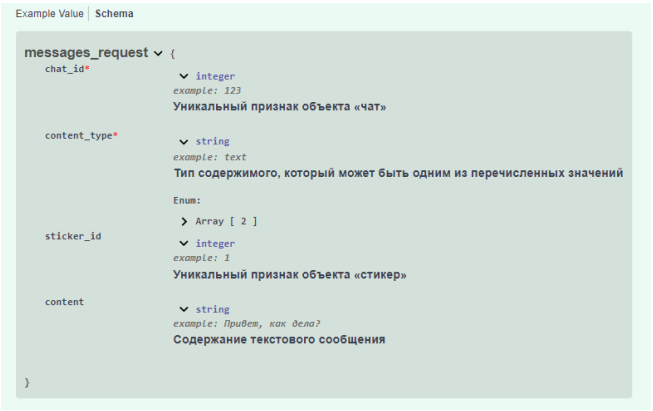
6. Swagger

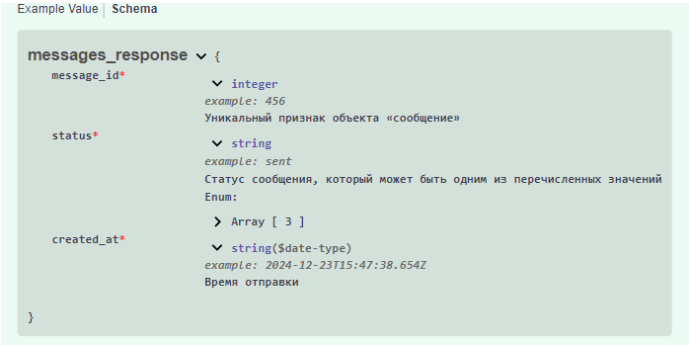

sticker Метод для работы со стикерами
GET /stickers Запрос списка стикеров
message Метод для работы с сообщениями
POST /messages Отправка сообщения
Schemas
stickers_response
sticker
pagination
messages_request
messages_response


sticker Метод для работы со стикерами
GET /stickers Запрос списка стикеров
Возвращает список стикеров с возможностью пагинации и фильтрации по тегу
Parameters
Try it out
Name
Description
page
integer
(query)
Текущая страница для пагинации
limit
integer
(query)
Количество элементов на странице для пагинации
tag
string
(query)
Данные для удобства поиска
Responses
Code
Description
Links
200
Успешная операция
No links
application/json
Example Value | Schema
{
"stickers": [
{
"sticker_id": 1,
"sticker_url": "https://example.com/sticker1.png",
"tag": "funny"
}
],
"pagination": {
"page": 1,
"total_items": 100,
"limit": 10
}
}


Example Value | Schema
stickers_response {
stickers*
[sticker {
sticker_id*
sticker_url*
tag*
}]
pagination*
pagination {
page*
total_items*
limit*
}
}







Нефункциональные требования

Требования надёжности:

1. Система должна корректно обрабатывать отправку стикеров 99% времени.

Требования производительности:

1. Меню стикеров должно открываться не более чем за 2 секунды.
2. Запрос на отправку стикера (например, POST /messages) должен выдерживать нагрузку в 1 rps.

Критерии приемки

Кейс 1

Функциональность: Пользователь может использовать платные стикеры после их покупки.

Дано: Пользователю видны стикеры над введённым словом.
Когда: Пользователь нажимает на тусклый стикер.
Тогда: Система выводит сообщение об ошибке: "Этот стикер доступен только после покупки. Хотите приобрести его?" — и предлагает перейти к оплате.

Кейс 2

Функциональность: Система предотвращает спам и злоупотребление отправкой стикеров.

Дано: Пользователь находится в меню стикеров.
Когда: Пользователь многократно нажимает на один и тот же стикер слишком быстро.
Тогда: Система выводит сообщение об ошибке: "Слишком много действий. Пожалуйста, подтвердите, что вы не робот" — и предлагает пройти капчу.