## 1. Описание архитектуры мессенджера

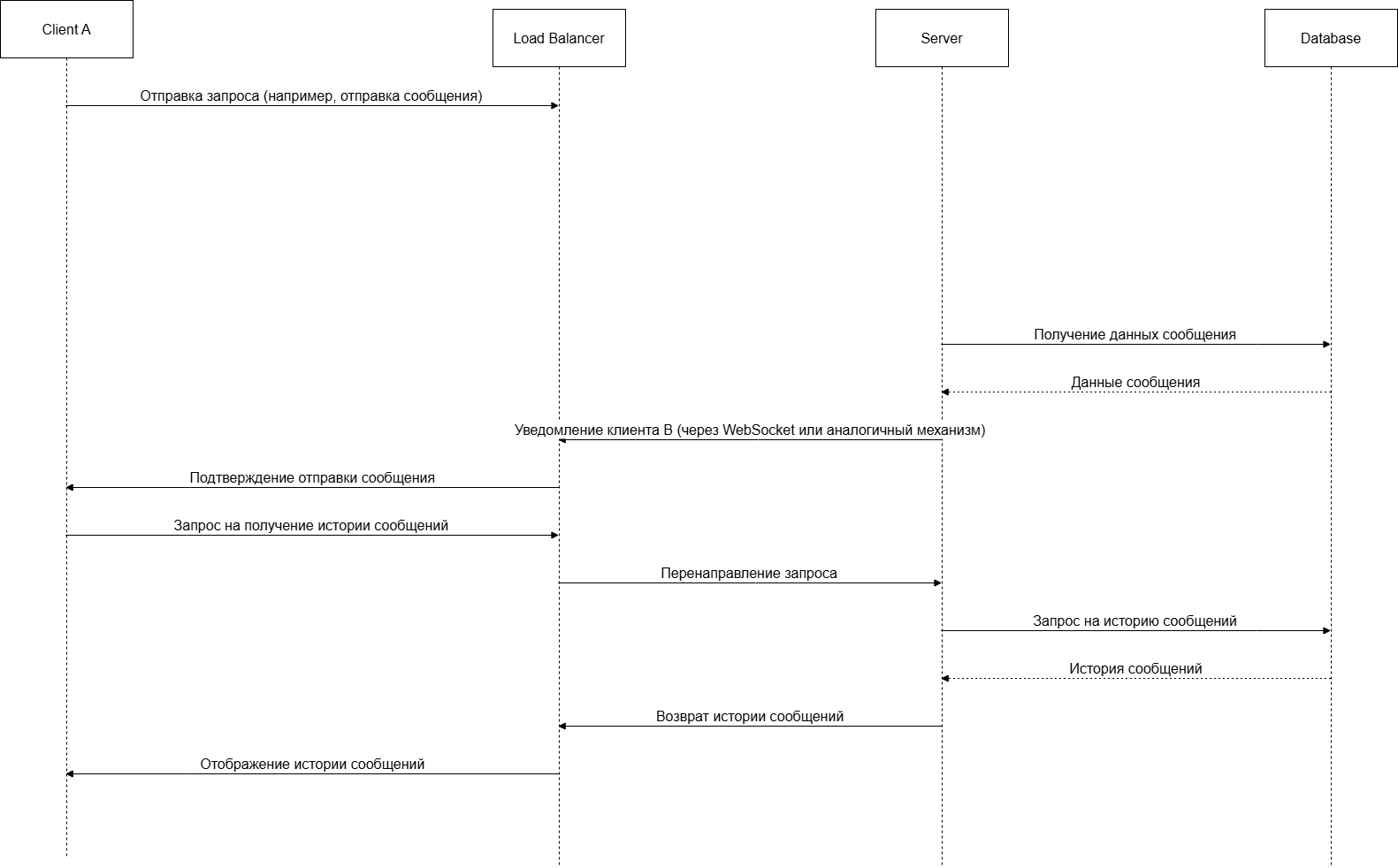
**Целью программы является** отправка и получение сообщений в реальном времени.

Компоненты:

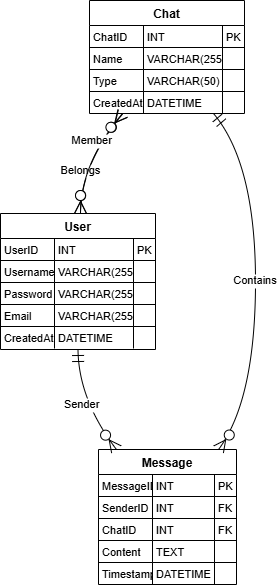
* Клиент: Приложение пользователя.
* Сервер приложений: Обрабатывает запросы, взаимодействует с БД и брокером.
* База данных: Хранит данные (пользователи, сообщения, чаты).
* Брокер сообщений: Отправляет уведомления в реальном времени.

Клиенты отправляют REST API запросы через балансировщик на серверы приложений. Серверы взаимодействуют с базой данных для хранения данных и публикуют уведомления в брокер сообщений. Брокер доставляет уведомления серверам, которые передают их клиентам через WebSocket или SSE.

**2. Схема взаимодействия компонентов**



**3. Логическая схема базы данных**



**4. Структура API (REST)**

**Базовый URL:** https://api.messenger.com/v1 (пример)

**Формат запросов:** JSON

**Формат ответов:** JSON

**Примеры API endpoints:**

1. **Аутентификация:**

**POST /auth/register** - Регистрация нового пользователя.

**Request Body:**

{

"username": "john\_doe",

"password": "secure\_password",

"email": "john.doe@example.com"

}

**Response (201 Created):**

{

"user\_id": 123,

"username": "john\_doe",

"email": "john.doe@example.com"

}

**Response (400 Bad Request):** (Например, username уже существует)

{

"error": "Username already exists"

}

**POST /auth/login** - Аутентификация пользователя.

**Request Body:**

{

"username": "john\_doe",

"password": "secure\_password"

}

**Response (200 OK):**

{

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c",

"user\_id": 123,

"username": "john\_doe"

}

**Response (401 Unauthorized):**

{

"error": "Invalid credentials"

}

1. **Пользователи:**

**GET /users/{user\_id}** - Получение информации о пользователе.

**Parameters:**

user\_id (path): ID пользователя.

**Headers:**

Authorization: Bearer <token> (Authentication required)

**Response (200 OK):**

{

"user\_id": 123,

"username": "john\_doe",

"email": "john.doe@example.com"

}

**Response (404 Not Found):**

{

"error": "User not found"

}

**GET /users?search={query}** - Поиск пользователей.

**Parameters:**

search (query): Строка поиска (например, по имени).

**Headers:**

Authorization: Bearer <token> (Authentication required)

**Response (200 OK):**

[

{

"user\_id": 123,

"username": "john\_doe",

"email": "john.doe@example.com"

},

{

"user\_id": 456,

"username": "jane\_doe",

"email": "jane.doe@example.com"

}

]

1. **Чаты:**

**GET /chats** - Получение списка чатов пользователя.

**Headers:**

Authorization: Bearer <token> (Authentication required)

**Response (200 OK):**

[

{

"chat\_id": 1,

"name": "John & Jane",

"type": "personal",

"last\_message": {

"content": "Hello!",

"timestamp": "2023-10-27T10:00:00Z"

}

},

{

"chat\_id": 2,

"name": "Developers Group",

"type": "group",

"last\_message": {

"content": "Let's have a meeting",

"timestamp": "2023-10-27T09:00:00Z"

}

}

]

**GET /chats/{chat\_id}** - Получение информации о чате.

**Parameters:**

chat\_id (path): ID чата.

**Headers:**

Authorization: Bearer <token> (Authentication required)

**Response (200 OK):**

{

"chat\_id": 1,

"name": "John & Jane",

"type": "personal",

"participants": [

{

"user\_id": 123,

"username": "john\_doe"

},

{

"user\_id": 456,

"username": "jane\_doe"

}

]

}

**POST /chats** - Создание нового чата.

**Request Body:**

{

"type": "personal",

"participant\_ids": [456] *// list of other user id(s) involved*

}

*// or*

{

"type": "group",

"name": "My New Group",

"participant\_ids": [456, 789]

}

**Headers:**

Authorization: Bearer <token> (Authentication required)

**Response (201 Created):**

{

"chat\_id": 3,

"name": "My New Group",

"type": "group"

}

1. **Сообщения:**

**GET /chats/{chat\_id}/messages?limit={limit}&offset={offset}** - Получение истории сообщений чата.

**Parameters:**

chat\_id (path): ID чата.

limit (query, optional): Максимальное количество сообщений для получения (default: 20).

offset (query, optional): Смещение для пагинации (default: 0).

**Headers:**

Authorization: Bearer <token> (Authentication required)

**Response (200 OK):**

[

{

"message\_id": 1,

"sender\_id": 123,

"content": "Hello!",

"timestamp": "2023-10-27T10:00:00Z"

},

{

"message\_id": 2,

"sender\_id": 456,

"content": "Hi there!",

"timestamp": "2023-10-27T10:01:00Z"

}

]

**POST /chats/{chat\_id}/messages** - Отправка нового сообщения в чат.

**Parameters:**

chat\_id (path): ID чата.

**Request Body:**

{

"content": "This is a new message"

}

**Headers:**

Authorization: Bearer <token> (Authentication required)

**Response (201 Created):**

{

"message\_id": 3,

"sender\_id": 123,

"content": "This is a new message",

"timestamp": "2023-10-27T10:02:00Z"

}

## 5. Стек технологий

Для разработки мы будем использовать:

* **Фронтенд:** HTML, CSS (интерфейс)
* **Backend:** Python(обработка логики).
* **БД:** PostgreSQL, Docker (хранение данных).
* **Связь:** Github (контроль версий)