# Обзор бэк-энд части приложения

## Стэк технологий

### JavaScript и Node.js

Бэк-энд часть приложения написана на языке JavaScript, на платформе Node.js. Это среда выполнения кода JavaScript вне браузера, которая позволяет писать серверный код для веб-страниц и веб-приложений, а также для программ командной строки.

Node.js — не отдельный язык программирования, а платформа для использования JavaScript на стороне сервера. С помощью платформы можно работать с файлами, сетью, базами данных и другими системными ресурсами на сервере.

Преимущества Node.js:

* Основным преимуществом этой технологии является использование асинхронной модели выполнения, что позволяет эффективно обрабатывать множество запросов одновременно (до 10000 активных соединений) без блокировки. Платформа позволяет писать асинхронный код из коробки, в то время как, например, в PHP для этого нужно использовать дополнительные инструменты.
* Благодаря Node разработчики могут использовать язык JavaScript как для клиентской, так и для серверной части, что значительно упрощает разработку.
* Платформа отлично справляется с потоковой передачей данных, особенно при работе с большими объемами данных или в режиме реального времени.
* Технология разрабатывалась специально для создания веб-продуктов, она отлично взаимодействует с различными базами данных.

### PostgreSQL

Любой клиент-серверный сервис нуждается в базе данных, и этот чат не исключение. В проекте используется реляционная база данных SQL, СУБД PostgreSQL.

PostgreSQL — это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире, Иногда СУБД называют бесплатным аналогом Oracle Database.

Основные преимущества PostgreSQL:

* **Бесплатное использование**;
* **Поддержка множества типов данных** (денежный, геометрический, перечисляемый, бинарный и другие типы);
* **Очень высокие пределы объемов хранимых данных** позволяют хранить практически любую информацию;
* **Поддержка сложных запросов;**
* **Одновременная модификация базы** позволяет расширить систему, добавив новые сервера**.**

### Протокол HTTP

В проекте использовался протокол HTTP – основная технология в веб программировании, которая позволяет осуществлять сообщение клиент-сервер. С помощью HTTP можно отправлять RESTful запросы из браузера на сервер, и получать ответы на них. Такой подход позволяет прописать на бэке эндпоинты, на которые отправляются запросы, в том числе таким образом реализованы регистрация и авторизация.

### Протокол WebSocket

Также использовался протокол WebSocket, он позволяет пересылать любые данные, на любой домен, безопасно и почти без лишнего сетевого трафика. В отличие от HTTP, веб-сокеты позволяют работать с двунаправленным потоком данных, т.е. сервер может не только отвечать на запрос клиента, но и самостоятельно передавать новую информацию по мере ее поступления.

Преимущества:

* Неограниченное время жизни соединения, его не нужно периодически обновлять, канал нужно открыть лишь один раз, потом можно спокойно передавать по нему данные;
* Множество одновременно открытых сессий;
* Быстрая передача динамически обновляемых данных без лишней нагрузки на сеть и сервер.

В общем виде механизм веб-сокета работает так:

1. **Клиент подключается к серверу** по TCP-соединению
2. **Сервер с поддержкой веб-сокетов отвечает**
3. **Клиент оставляет соединение открытым — канал готов**. Далее при любом обмене информацией между ним и сервером будет срабатывать коллбэк, установленного страницей или веб-приложением

Работа WebSocket строится на событийной системе, всего существует 4 события open, message, error, close.

## Реализация проекта

Рассмотрим, как применяются указанные технологии в работа бэк-энд части чата.

### Реализация WebSocket

В данном проекте WebSocket используется примерно таким образом:

1. Создается множество подключенных клиентов;
2. Для каждого принятого веб-сокета, клиент добавляется в множество;
3. Когда сообщение от пользователя получено, всем клиентам отправляется сообщение через открытое соединение;
4. Когда подключение закрыто, пользователь удаляется из множества, сообщения ему больше не шлются.

### База Данных

База данных в проекте примитивна, в ней представлена только 1 таблица для хранения зарегистрированных пользователей. Через запросы на базу данных реализуются системы авторизации и регистрации в системе.

Взаимодействие с базой данных осуществляется следующими функциями:

1. inputUser({nickname, password})

SQL запрос: insert into users (nickname, password) values (arg1, arg2)

Для добавления нового пользователя в базу данных.

1. getByNick(nickname)

SQL запрос: select \* from users where nickname = arg1

Для поиска пользователя в базе данных.

Поскольку сообщения в базе данных не хранятся, пользователь не получит сообщения, которые были отправлены до того, как он вошел в систему. В связи с этим, при закрытии вкладки с приложением все полученные ранее сообщения, нельзя будет увидеть снова.

# Источники

1. <https://ru.hexlet.io/blog/posts/zachem-izuchat-node-js-ili-o-perspektivah-bekenda-na-javascript>
2. <https://www.freecodecamp.org/news/git-clone-branch-how-to-clone-a-specific-branch/>
3. <https://learn.javascript.ru/websockets>
4. <https://devconnected.com/create-git-branch/>
5. <https://blog.skillfactory.ru/glossary/websocket/>
6. <https://habr.com/ru/companies/skyeng/articles/448968/>
7. <https://blog.skillfactory.ru/glossary/postgresql/>