Микрюков И.И. Реализация веб-чата с использованием WebSocket на языке Golang // Международный журнал информационных технологий и энергоэффективности. – 2023. – Т. 8 № 6(32) с. 115–119

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.773

РЕАЛИЗАЦИЯ ВЕБ-ЧАТА С ИСПОЛЬЗОВАНИЕМ WEBSOCKET НА ЯЗЫКЕ GOLANG

Микрюков И.И.
ФГАОУ ВО «Санкт-Петербургский государственный университет аэрокосмического приборостроения», Санкт-Петербург, Россия (190000, г. Санкт-Петербург, ул. Большая Морская, д. 67, лит. А), e-mail: ilya.mikryukov.2001@mail.ru

В данной статье рассматривается реализация веб-чата для обмена сообщениями между пользователями в реальном времени. Для создания чата использовался язык программирования Golang и библиотека Gorilla WebSocket. В статье также есть сравнение некоторых протоколов для передачи данным между клиентом и сервером, после был сделан выбор в пользу WebSocket.

Ключевые слова: Веб-чат, Веб-сервер, WebSocket, HTTP, Golang, протоколы передачи данных.

WEB-CHAT IMPLEMENTATION USING WEBSOCKET IN GOLANG LANGUAGE

Mikryukov I.I.
St. Petersburg State University of Aerospace Instrumentation, St. Petersburg, Russia (190000, St. Petersburg, Bolshaya Morskaya St., 67, letter A), e-mail: ilya.mikryukov.2001@ mail.ru

This article discusses the implementation of a web chat for real-time messaging between users. The chat was created using the Golang programming language and the Gorilla WebSocket library. The article also has a comparison of some protocols for transferring data between a client and a server, after which the choice was made in favor of WebSocket.

Keywords: Web chat, Web server, WebSocket, HTTP, Golang, data transfer protocols.

Данная работа посвящена вопросам разработки веб-чата, который можно использовать для прямой трансляции или в других системах, где необходимо обмениваться информацией в реальном времени с минимальной задержкой между большим количеством пользователей. На основе сравнительного анализа различных технологий передачи данных между сервером и клиентом выбран наиболее подходящий протокол для разработки такого веб-чата и описана работа как со стороны сервера, так и со стороны клиента.

Сравнение протоколов обмена данными в сети Интернет

Существуют различные способы обмена данными в сети Интернет. Рассмотрим и сравним самые популярные из них и на этой основе сделаем выбор. При сравнении протоколов использовалась информация с сайтов [1] и [2]. Самый используемый протокол передачи данных между клиентом и сервером, это HTTP (HyperText Transfer Protocol — «протокол передачи гипертекста»). HTTP – является однонаправленным протоколом передачи данных, хорошо подходит, если необходимо единожды сделать запрос информации с сервера, после

чего соединение будет автоматически разорвано. На рисунке 1 схематично представлена передача данных по HTTP протоколу. Для передачи большого объёма информации в реальном времени значительному количеству пользователей такая передача не подходит, так как имеет следующие недостатки:

- нет постоянного двунаправленного соединения;
- необходимо часто отправлять запросы на сервер, чтобы проверить, есть ли новая информация;
- каждый раз отправляется заголовок, где указывается, какой тип запроса хотим сделать;
- медленная скорость работы и лишняя нагрузка на сеть;
- перезагрузка страницы после выполнения запроса.

Есть некоторое усовершенствование HTTP запросов, это асинхронные запросы с использованием JavaScript и XML (eXtensible Markup Language — «расширяемый язык разметки»), то есть AJAX (Asynchronous Javascript and XML — «асинхронный JavaScript и XML») запросы. Они решают проблему того, что автоматически получают и отправляют данные без перезагрузки всей веб-страницы, а только некоторой части. Так как это асинхронные запросы, то пользователь во время выполнения запроса может совершать другие действия на веб-странице. Ответ от сервера можно получать не только в формате XML, но также в виде обычного текста и в формате JSON (JavaScript Object Notation — «текстовый формат обмена данными, основанный на JavaScript»). За счет асинхронных запросов последнюю проблему можно считать решенной, но предыдущие недостатки все-таки заставляют сделать вывод о нецелесообразности использования протокола HTTP при создании веб-чата.

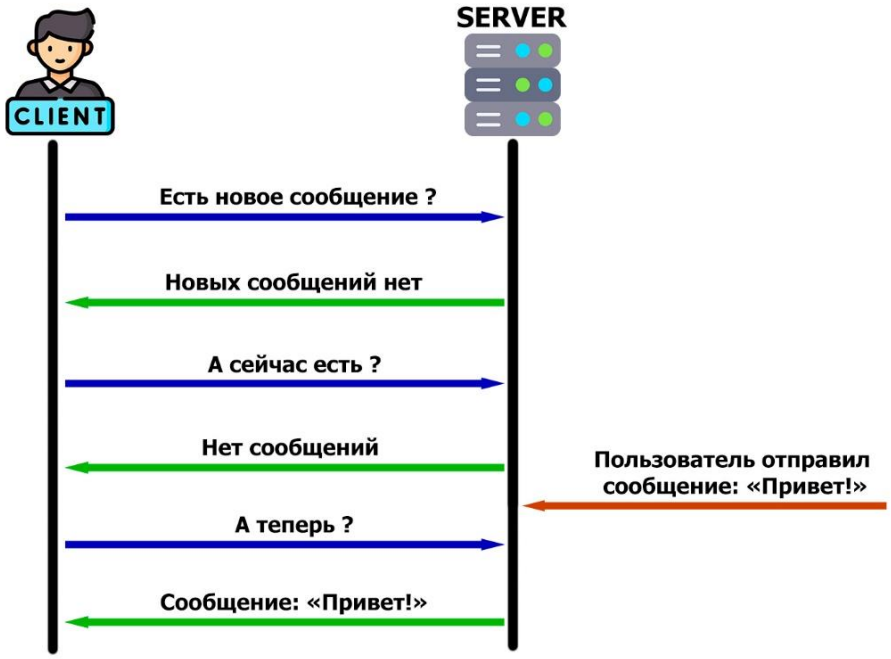

Рисунок 1 – Передача сообщения по протоколу HTTP

Еще один протокол передачи данных, заслуживающий внимания для решения поставленной задачи, это протокол WebSocket. Проанализировав информацию с сайта [3], отметим, что в данном протоколе проблемы, присущие протоколу HTTP, решены. Во-первых,

протокол WebSocket является двунаправленным, то есть можно одновременно отправлять и получать данные. На рисунке 2 схематично представлена передача данных по протоколу WebSocket. Кроме того, установленное соединение можно использовать повторно, что повышает скорость работы и уменьшает нагрузку на сеть. Пользователь отправляет на сервер запрос, при котором происходит как бы “рукопожатие” с сервером и “согласование” того, что будет использоваться WebSocket протокол. Также данный протокол поддерживает возможность сжатия данных. В протоколе реализован механизм, проверяющий подключён ли ещё пользователь или нет. Он работает следующим образом: сервер периодически посылает запрос пользователю, чтобы тот сделал “ответный” запрос на сервер, если ответ в течение некоторого времени не приходит, то происходит разрыв соединения. С его помощью можно реализовать не только веб-чат, но и другие системы, где передаётся большое количество информации между пользователями и сервером, например: биржи, push-уведомления, игровые приложения.

Рассмотрение особенностей протокола WebSocket показывает, что его использование в веб-чате более целесообразно, чем HTTP.

После выбора протокола перейдём к описанию технической части реализации веб-чата. Рассмотрим вопросы, связанные с проектированием серверной и клиентской сторон веб-чата.

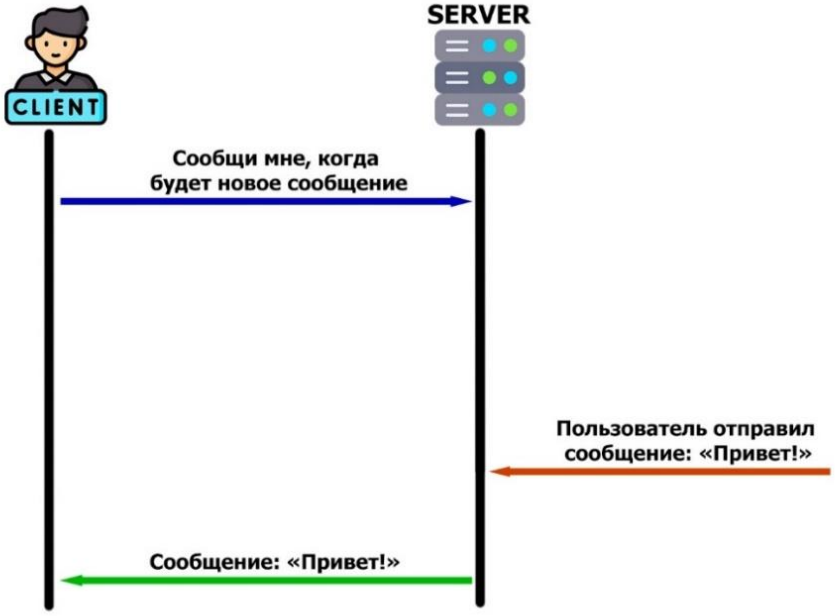

Рисунок 2 – Передача сообщения по протоколу WebSocket

Программный модуль сервера, обрабатывающего сообщения от пользователей, было решено написать на языке программирования Golang, так как его производительность в некоторых случаях выше, чем у языков PHP, JavaScript, Python и близка к C++. Golang – это компилируемый многопоточный язык, в котором есть такое понятие как goroutines (горутины). Горутины — это легковесные потоки, которые реализуют конкурентное программирование. С их помощью можно создать большое количество потоков и при этом потратить меньше памяти, чем если бы это реализовывалось на других языках программирования. Если рассматривать это в контексте веб-чата, то для масштабирования данной системы, горутины отлично подойдут.

При написании сервера использовалась информация, представленная на сайте [4]. Для реализации соединений с сервером по WebSocket была выбрана библиотека Gorilla WebSocket. Более подробная информация для работы с данной библиотекой имеется на сайте [5]. Библиотека имеет хорошую документацию и в настоящее время пользуется большой популярностью.

Рассмотрим детально, как выполняются этапы работы с созданным сервером.

- На сервере есть функция, которая принимает HTTP запросы и обновляет их до WebSocket, то есть совершает “рукопожатие”.
- Далее после успешного установления соединения, клиент записывается в массив подключенных в данный момент к серверу пользователей.
- Также есть функция, которая принимает сообщения от пользователя и передаёт их в канал данных.
- Следующая функция читает данные из канала и отправляет их всем пользователям, которые находятся в массиве активных пользователей.
- Имеется также функция, которая проверяет соединение с пользователями и разрывает его, если они неактивны.
- Сервер при получении нового сообщения от клиентов сдвигает значения ячеек массива с последними 10 сообщениями на одну позицию вверх пока не дойдёт последнего элемента, а в последнюю ячейку массива будет записываться только что полученное новое сообщение от клиента.
- При подключении к чату новый клиент будет получать последние 10 сообщений.

Теперь перейдём к рассмотрению клиентской части веб-чата. Она написана на JavaScript и работает в браузере. Изначально создаётся переменная типа WebSocket. Так как данные с сервера на клиент отправляются в JSON формате, то в функции получения сообщений, которая запускается при совершении события “получения сообщения” в вебсокете, необходимо распарсить полученные с сервера данные (записать данные в определенную структуру), после чего записать в элементы интерфейса.

Кроме того, имеется функция для отправки сообщения от клиента на сервер. В ней данные, полученные из пользовательского интерфейса, записываются в структуру, которая переводится в JSON формат и передаётся на сервер. На рисунке 3 показан веб-интерфейс разработанного чата.

Подводя итог проделанной работе, можно сказать, что обоснованный выбор протокола WebSocket, а также проектирование серверной и клиентской сторон веб-чата, подробное описание которых приведено в данной в статье, позволили создать полностью рабочий веб-чат, обеспечивающий обмен информацией между большим количеством пользователей в реальном времени с минимальной задержкой.

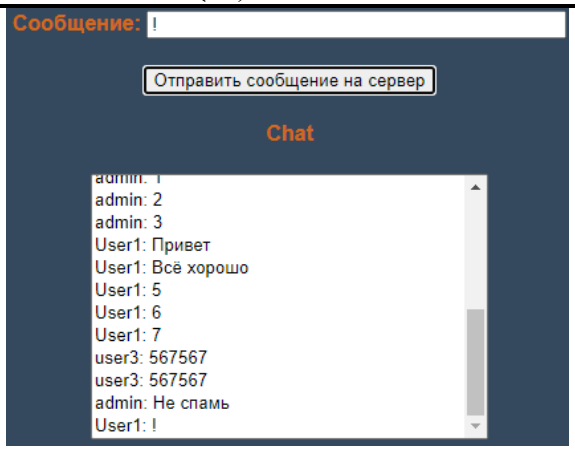

Рисунок 3 – Работа веб-чата в браузере

Список литературы

1. Почему протокол WebSocket - лучший выбор: сравнение с HTTP. [Электронный ресурс] – Режим доступа: https://webformyself.com/websockets-vs-http/
2. WebSocket (WSS): что это и как работают сокеты, асинхронный сервер. [Электронный ресурс] – Режим доступа: https://blog.skillfactory.ru/glossary/websocket/
3. WebSocket. [Электронный ресурс] – Режим доступа: https://learn.javascript.ru/websocket
4. Простейший сервер на Gorilla WebSocket / Хабр. [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/599737/
5. websocket package - github.com/gorilla/websocket - Go Packages. [Электронный ресурс] – Режим доступа: https://pkg.go.dev/github.com/gorilla/websocket

References

1. Why the WebSocket protocol is the best choice: comparison with HTTP. [Electronic resource] – Access mode: https://webformyself.com/websockets-vs-http/
2. WebSocket (WSS): what is it and how sockets work, asynchronous server. [Electronic resource] – Access mode: https://blog.skillfactory.ru/glossary/websocket/
3. Web socket. [Electronic resource] – Access mode: https://learn.javascript.ru/websocket
4. The simplest server on Gorilla WebSocket / Sudo Null IT News [Electronic resource] – Access mode: https://habr.com/ru/articles/599737/
5. websocket package - github.com/gorilla/websocket – Go Packages. [Electronic resource] - Access mode: https://pkg.go.dev/github.com/gorilla/websocket