O. B. Prydybailo, R. V. Prydybailo

TRAFFIC ROUTING USING THE ISTIO MANAGED PLATFORM

The article examines how traffic is managed through a managed platform. The managed Istio platform is advantageous in that it solves the complexities of applications based on a set of small services, each of which works in its own process and communicates with other mechanisms, usually HTTP, in other words, works with microservices. The technology discussed in this article has great advantages that allow you to connect, manage, and secure your microservices networks regardless of the runtime, source, and developer. The managed platform allows you to manage the input and output of the information so that it can track the response time of the request, retry requests and load balancing on computer systems; provides step-by-step program monitoring, continuous tracking, and user safety. It is very important that the Istio platform operates on the network and implements the submission of information about user systems and the process of collecting, aggregating and analyzing this data to improve the characteristics and behavior of system components. For the proper operation of the managed platform, there is a software part of the application architecture, such as a service grid, which provides safe, fast and reliable interaction between discrete software components — services. There are also several schematics in this article that clearly illustrate the managed platform architecture and the overall application architecture that can be realistically created, tested, and run on cloud platforms. Running a managed platform takes place on Google platforms, which is very easy to use, because it requires no extra cost and has a very simple code to connect. This code is also given in this article.

Keywords: managed platform; microservices; traffic management; service grid; data panel; control panel; traffic routing; network gateway; virtual services; microservice grids; cluster; configuration environment; user interface; namespace; monitoring; metrics.

УДК 004.04

DOI: 10.31673/2412-9070.2020.064043

А. О. БАВЕНКО, студент;
А. Б. КОБА, ст. викладач,
Державний університет телекомунікацій, Київ

ВИКОРИСТАННЯ ТЕХНОЛОГІЇ WebSocket ДЛЯ ПЕРЕДАВАННЯ ДАНИХ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Запропоновано застосування сучасної технології для передавання даних у режимі реального часу — WebSocket. Відображено статистику кількості веб-сайтів та використання інтернету населенням. Аргументується, чим використання веб-застосувань ефективне для проектів.

Визначено проблему швидкого передавання даних, а також актуальності інформації. Встановлено частини, на які поділяють веб-розроблення, та чим вони відрізняються. Наведено кілька прикладів.

Досліджено технологію для отримання даних на веб-сторінку AJAX. Як саме реалізується використання цієї технології, можливості налаштування та функціонал, який вона надає. Також розглянуто її переваги щодо отримання даних порівняно зі способом при перезавантаженні сторінки. Проаналізовано недоліки використання цієї технології у разі, коли потрібні найактуальніші дані.

Запропоновано технологію отримання даних на веб-сторінку Server-sent Event, її використання та функціонал, який вона надає. Визначено переваги, які вона дає в отриманні даних порівняно зі способом при використанні технології AJAX. Проаналізовано недоліки використання цієї технології.

Розглянуто технологію WebSocket як найкраще вирішення для проектів, де необхідно у найкоротший термін передавати та отримувати данні. Показано принцип роботи, переваги використання порівняно з технологією Server-sent Event.

Проілюстровано переваги та недоліки кожної технології, за допомогою яких ми можемо реалізувати передавання даних у веб-середовищі. Проаналізовано ефективність використання кожної технології порівняно між собою. Пояснюється чому технологія WebSocket заслуговує уваги як технологія для передавання даних у веб-середовищі і чому її можна назвати технологією для передавання даних у режимі реального часу.

Ключові слова: WEB; HTTP; WebSocket; веб-сервер; браузер; передавання даних; AJAX; Server-Sent Event; актуальність даних.

ВСТУП

У наш час мережа Інтернет набуває стрімкого розвитку. У повсякденні задля власних потреб ним користується більшість людей. На початок 2019 року на планеті налічувалось 4 100 667 287 інтернет-користувачів, що перевищує 53% усього населення Землі. Станом на січень 2019 року в інтернеті було зареєстровано 1,94 млрд веб-сайтів. Це пошукові системи, блоги, електронні бібліотеки, сайти-візитки, соціальні мережі, сайти компаній, поштові сервіси тощо. Із розвитком мобільних технологій та покриттям мобільного інтернету

дедалі більше людей використовують мобільні пристрої для доступу до мережі Інтернет, а не ПК. У 2018 році використання мобільного інтернету становило 48,2% [1].

На противагу програмному забезпеченню, яке необхідно встановлювати на пристрій для його використання, веб-застосування з легкістю доступні для користувача за допомогою браузера та доступу до інтернету, які зараз підтримуються на всіх ПК та мобільних пристроях.

Із розвитком технологій веб-розроблення сайти набули складнішої архітектури, більших можливостей, сформувалося чимало підходів до їх створення, надаючи свої переваги порівняно з іншими. Залежно від типу сайта встановлюються вимоги до його розроблення. Взагалі розроблення сайта поділяють на дві частини: front-end (все, що стосується зовнішнього вигляду та логіки інтерфейсу) та back-end (все, що належить до оброблення запитів на логіки серверної частини).

Наприклад, сайти-візитки мають чисто інформативний характер: це може бути презентація товару, послуги, компанії, події тощо. Зазвичай на такому сайті є контакти для зв'язку або можливість залишити свої. Приємний і відповідний зовнішній вигляд та інформативність змісту головне для сайта-візитки, тому найбільш акцентується увага на front-end (зовнішня складова) частині, а back-end (серверна складова) потребує мінімальної кількості логіки. Є сайти більш об'ємніші, зі складнішою архітектурою, де можуть бути задіяні сторонні сервіси, сервери бази даних тощо. На таких сайтах може бути авторизація та реєстрація користувачів, рівні доступу, внутрішній форум чи чат, система моніторингу динамічних значень, можливість скористатися банківськими послугами тощо. У таких сайтах більша частина розроблення припадає на back-end частину. Такі сайти можна назвати повноцінними веб-застосуваннями через складність архітектури. Для розроблення front-end та back-end частин веб-застосування використовуються різні технології та підходи, що, у свою чергу, поділяє розроблення на частини не тільки логічно, а й технічно.

ОСНОВНА ЧАСТИНА

Проблема актуальності даних

Сьогодні вся робота із сайтами в мережі Інтернет базується на протоколі HTTP. Основним призначенням протоколу HTTP є передавання веб-сторінок (текстових файлів із позначенням HTML), хоча за його допомогою успішно передаються й інші файли як пов'язані з веб-сторінками (зображення і застосування), так і не пов'язані з ними (у цьому HTTP конкурує зі складнішим FTP) [2]. HTTP — це протокол передавання даних прикладного рівня, що ґрунтується на «клієнт-серверній» архітектурі, яка передбачає клієнта, котрий встановлює з'єднання та відсилає запити, і веб-сервер, який очікує запит, обробляє його та надсилає готову відповідь на запит. Зазвичай у ролі клієнта виступає браузер. Саме з'єднання між клієнтом та веб-сервером не постійне, і після відповіді на запит веб-сервер закриває з'єднання.

Розглянемо ситуацію, де веб-сервер залишає позначку часу на веб-сторінці на етапі формування відповіді. Отримавши відповідь на запит, клієнт побачить час, який було встановлено веб-сервером на момент формування відповіді. Поки відповідь дійде до клієнта та обробиться браузером, час буде вже не актуальним. Щоб отримати актуальний час на сторінці, потрібно знову надіслати запит. Із кожним запитом веб-сервер буде готувати веб-сторінку, встановлювати в ній час та відсилати клієнтові. Така робота передбачає витрату зайвих ресурсів сервера, адже нам не потрібна кожного разу вся веб-сторінка. А якщо клієнтів дуже багато, то витрати ресурсів сервера на формування відповіді може бути занадто високими. З боку клієнта зросте час очікування відповіді, оскільки веб-сервер паралельно буде відповідати кільком клієнтам, і дані втратять ще більше актуальності. Ще один важливий недолік у тому, що веб-сервер не має можливості ініціювати відправлення змінених даних клієнту, оскільки з'єднання закривається. Отже, постає проблема передавання актуальних даних за мінімальний проміжок часу з максимальною ефективністю.

AJAX

За допомогою підтримання браузером мови програмування JavaScript та технології AJAX можна запропонувати вихід із цієї ситуації.

AJAX (*Asynchronous JavaScript And XML*) — підхід до побудови користувацьких інтерфейсів веб-застосувань, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачеві дані [3].

За допомогою JavaScript на веб-сторінці створюється об'єкт, який реалізує інтерфейс для користування AJAX. Для основного функціоналу зазначається необхідний URL для запиту на сервер, та що саме необхідно робити з результатом відповіді. Також є можливість задати дії на кожний етап запиту: установлення з'єднання із сервером, отримання запиту сервером, оброблення запиту та завершення запиту, коли результат отримано. Окрім цього є можливість задати дії на кожний код стану відповіді сервера. Код стану належить до HTTP протоколу і зазначає стан нашого запиту. Наприклад, 200 — успішно обробився, 404 — не існує файла або інформації.

AJAX реалізує стандартні HTTP запити. За допомогою їх ми можемо не тільки отримувати інформацію з сервера, а й надсилати її серверу.

AJAX частково розв'язує проблему, але цей підхід має чимало своїх недоліків. Наприклад, він не вирішує проблеми з запитами — їх потрібно надсилати і чекати відповіді, а якщо нам необхідно мати найактуальнішу інформацію, ці запити потрібно надсилати з найменшим інтервалом для максимальної відповідності даних. Час на отримання інформації при таких запитах менший, оскільки веб-сервер витрачає мінімум часу на формування відповіді, адже формує не всю сторінку, а тільки необхідні дані.

У разі, коли потрібна актуальна інформація багатьох значень на сторінці, цей підхід не дуже зручний. Є два способи вирішити цю проблему. Перший — веб-серверу доведеться обробляти запит, у відповіді на який будуть передаватися одразу всі значення, або другий, коли буде багато запитів по кожному зі значень. У першому випадку веб-серверу знадобиться більше часу, щоб підготувати дані, частина з яких може бути неактуальна, доки дійде до клієнта. У другому випадку веб-сервер швидше відповідатиме, але буде повільніше реагувати на запити, яких може бути дуже багато залежно від кількості необхідних даних. У разі, якщо клієнтів буде дуже багато, то веб-сервер не буде встигати вчасно всім відповідати, і дані можуть залишитись не актуальними на момент отримання їх клієнтом.

Server-Sent Events

Щоб організувати швидке передавання даних, можна використовувати технологію **SSE** (*Server-Sent Event*). SSE — це технологія відправлення повідомлень від сервера до веб-браузера у вигляді DOM-подій. SSE є стандартом, який описує способи початку передавання даних клієнтам із моменту організації клієнтом першого з'єднання. Стандарт широко використовується для надсилання повідомлень щодо оновлення або для надсилання безперервних потоків даних браузеру клієнта. Його спроектовано для поліпшення крос-браузерного мовлення за допомогою JavaScript API під назвою EventSource; з його допомогою клієнт задає URL для отримання потоку подій, який його цікавить [4].

Якщо ми підем шляхом використання SSE, на відміну від AJAX, то час надходження даних до клієнта буде набагато менший. Після отримання веб-сторінки за допомогою JavaScript відбувається запит для підписки на події веб-сервера. Запит виконується один раз, встановлюється з'єдання, яке залишається активним. Залежно від логіки роботи веб-сервера генеруються події, після яких веб-сервер відправляє дані клієнту. Наприклад, це може бути зміна значень, отримання додаткової інформації веб-сервером тощо. Це, у свою чергу, дає перевагу над підходом із використанням AJAX, де для отримання даних завжди надсилаються запити. Але і в цієї технології є свої недоліки. По-перше, технологія дає можливість передавати дані лише в одному напрямі — від веб-сервера до клієнта. Якщо потрібно надіслати дані від клієнта до веб-сервера, то нам не обійтись без AJAX. По-друге, але менш важливіше, за допомогою SSE ми можемо надсилати тільки текстові дані.

WebSocket

WebSocket — протокол зв'язку поверх TCP-з'єднання, призначений для обміну повідомленнями між браузером та веб-сервером у режимі реального часу. WebSocket розроблено для втілення у веб-браузерах та веб-серверах, але його також можна використовувати і для будь-якого клієнтського або серверного програмного забезпечення. Протокол WebSocket — це незалежний протокол, який базується на протоколі TCP. Він уможливлює більш тісну взаємодію між браузером і вебсервером, сприяє поширенню інтерактивного вмісту та створенню програмного забезпечення реального часу [5].

Технологія WebSocket, як і SSE, працює за подійною архітектурою.

Основні переваги використання WebSocket полягають в тому, що між веб-сервером та клієнтом встановлюється з'єднання, яке залишається активним, та на відміну від SSE створює двонапрямлений канал зв'язку. За допомогою JavaScript з боку клієнта надсилається запит на встановлення з'єднання з WebSocket-сервером (який зазвичай програмно створюється на боці веб-сервера, але не обов'язково на ньому), після чого відбувається операція «рукостискання» між ними. Після успішної операції «рукостискання» з'єднання встановлено. У сервера, як і у клієнта, є можливість за певних подій відсилати дані без додаткових запитів. Це в свою чергу дає значиму перевагу над технологією SSE, де застосовується однонапрямлений канал зв'язку. Ще однією перевагою WebSocket є те, що можна передавати не тільки текстові дані, а й бінарні.

ВИСНОВКИ

З огляду на сучасний стан інформаційних технологій створення веб-застосувань із використанням WebSocket дає багато переваг, якщо потрібно максимально швидко передавати дані у веб-середовищі. Подійна архітектура дає змогу зручно налагодити логіку передавання даних в обох напрямах. Такий спосіб дає можливість створити сторінки, які будуть обмінюватись даними в режимі реального часу. Використання технології

дуже ефективне для сервісів із постійним обміном даними, будь-то веб-сторінка з курсами валют, онлайн ігри, торгівельні площадки, важливі інженерні сервіси тощо.

Список використаної літератури

1. ***URL:*** *https://sdvv.ru/articles/elektronnaya-kommertsiya/statistika-interneta-2019-trafik-sayty-i-blogi-domeny-sotsialnye-media-onlayn-reklama-i-elektronnaya/*

2. ***URL:*** *https://uk.wikipedia.org/wiki/HTTP*

3. ***URL:*** *https://uk.wikipedia.org/wiki/AJAX*

4. ***URL:*** *https://ru.wikipedia.org/wiki/Server-sent_events*

5. ***URL:*** *https://uk.wikipedia.org/wiki/WebSocket*

А. О. Бавенко, А. Б. Коба

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ WebSocket ДЛЯ ПЕРЕДАЧИ ДАННЫХ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

Предложена статистика количества веб-сайтов и использования интернета населением. Аргументируется, чем использование веб-приложений эффективно для проектов.

Определена проблема быстрой передачи данных, а также актуальности информации. Установлены части, на которые разделяют веб-разработку, и чем они отличаются. Приведены примеры.

Исследована технология для получения данных на веб-страницу AJAX. Каким образом реализуется использования этой технологии, возможности настройки и функционал, который она предоставляет. Также рассмотрены ее преимущества относительно данных по сравнению со способом при перезагрузке страницы. Проанализированы недостатки использования этой технологии в случае, когда нужны самые актуальные данные.

Рассмотрена технология получения данных на веб-страницу Server-sent Event, ее использование и функционал, который она предоставляет. Определены преимущества, которые она дает в получении данных по сравнению со способом при использовании технологии AJAX. Проанализированы недостатки использования этой технологии.

Предложена технология WebSocket как лучшее решение для проектов, где необходимо за кратчайшее время передавать и получать данные. Показан принцип работы, преимущества использования по сравнению с технологией Server-sent Event.

Сформулированы преимущества и недостатки каждой технологии, с помощью которых мы можем реализовать передачу данных в веб-среде. Проанализирована эффективность использования каждой технологии в сравнении между собой. Объясняется почему технология WebSocket заслуживает внимания как технология для передачи данных в веб-среде и почему ее можно назвать технологией для передачи данных в режиме реального времени.

Ключевые слова: WEB; HTTP; WebSocket; веб-сервер; браузер; передача данных; AJAX; Server-Sent Event; актуальность данных.

A. O. Bavenko, A. B. Koba

USING WebSocket TECHNOLOGY FOR REAL-TIME DATA TRANSFER

This article is about the use of modern technology for real-time data transmission - WebSocket.

Web site statistics and population usage are displayed. It is argued that the use of web applications is effective for projects.

The problem of fast data transfer and the relevance of information are considered. It describes the parts that web development is divided into and how they differ. Here are some examples.

This article describes the technology used to get data to the AJAX website. How this technology is implemented, configurable, and the functionality it provides. It also looks at the benefits it has in retrieving data compared to how it reloads the page. The disadvantages of using this technology are analyzed when the most up-to-date data is needed.

It tells about the technology of getting data to the Server-sent Event website. How this technology is implemented and the functionality it provides. It also looks at the benefits it has in getting data compared to how it used to use AJAX technology. The disadvantages of using this technology are analyzed.

WebSocket technology is considered as the best solution for projects that need to transmit and receive data in the shortest possible time. The principle of operation, advantages of use in comparison with Server-sent Event technology are discussed.

The article helps us understand the advantages and disadvantages of each technology that we can use to deliver data in a web environment. The efficiency of the use of each technology in comparison with each other is analyzed. It explains why WebSocket technology is worth considering as a technology for data transmission in the web environment, and why it can be called technology for real-time data transmission.

Keywords: WEB; HTTP; WebSocket; web-server; browser; data transfer; AJAX; Server-Sent Event; data relevance.