DOI: https://doi.org/10.36910/6775-2524-0560-2023-53-27
УДК 004.042
Пікуляк Микола Васильович, к. т. н., доцент
https://orcid.org/0000-0003-2192-1899
Домбровський Станіслав Володимирович, магістр
Дутчак Марія Степанівна, викладач
https://orcid.org/0000-0002-3337-5613
Прикарпатський національний університет імені Василя Стефаника, м. Івано-Франківськ, Україна

ВДОСКОНАЛЕННЯ АСИНХРОННОГО МЕТОДУ СЕРВІСНОЇ ВЗАЄМОДІЇ У ВЕБ-ДОДАТКАХ

Пікуляк М.В., Домбровський С.В, Дутчак М.С. Удосконалення асинхронного методу взаємодії сервісів у веб-додатках. Проведено порівняльний аналіз відомих протоколів асинхронної взаємодії сервісів, що дозволило виділити їхні позитивні та негативні сторони. Визначено напрямок покращення сервісної взаємодії у веб-додатках, направлений на розробку методу з використанням протоколу gRPC за рахунок генерації Java коду на основі описаних сервісів за допомогою повідомлень Protocol Buffers (protobuf). Виконано тестування розробленого методу у веб-додатку для анонімних дискусій, який надає можливість спілкуватись користувачам за допомогою чату. Отримано результати дослідження, які підтвердили ефективність застосування запропонованого методу, оскільки його використання дозволило при навантаженні зменшити як час створення об'єктів так і час обробки запитів до сервера. Також покращення дозволило збільшити пропускну здатність при великій кількості паралельних підключень. Визначено перспективи подальших досліджень, направлених на розробку нових методів покращення для інших мов програмування, які підтримують генерацію gRPC сервісів.
Ключові слова: протоколи сервісної взаємодії, метод покращення асинхронної взаємодії, веб-додатки, повідомлення protobuf.

Pikuliak M., Dombrovskyi S., Dutchak M. Improving the asynchronous method of service interaction in web applications. A comparative analysis of known protocols for asynchronous interaction of services was carried out, which made it possible to highlight their positive and negative sides. The direction of improving service interaction in web applications is determined, aimed at the development of a method using the gRPC protocol due to the of Java code based on the described services using Protocol Buffers (protobuf) messages. Testing of the developed method was performed in a web application for anonymous discussions, which provides an opportunity to communicate with users using chat. The results of the study were obtained, which confirmed the effectiveness of the proposed method, since its use made it possible to reduce both the time of creating objects and the time of processing requests to the server under load. Also, the improvement made it possible to increase the bandwidth with a large number of parallel connections. Prospects for further research aimed at developing new improvement methods for other programming languages that support the generation of gRPC services are determined.
Keywords: service interaction protocols, asynchronous interaction improvement method, web applications, protobuf message.

Вступ. У сучасному світі різного роду веб-додатки стають все популярнішими та охоплюють все ширші предметні області застосування. Вони надають різноманітні сервіси, послуги та можливості, такі як дистанційне навчання, онлайн-торгівля, соціальні мережі, онлайн-банківські послуги та багато іншого.

Як правило, функціонування будь-якого веб-додатку базується на основі клієнт-серверної технології. Тому виникає необхідність аналізу процесу взаємодії між клієнтом та сервером, оскільки це є критично важливим для успішної роботи додатку.

У зв'язку з цим сьогодні проводиться багато досліджень, метою яких є застосування асинхронних методів сервісної взаємодії, які дозволяють виконувати більш ефективні запити до сервера, використовуючи менше системних ресурсів та зменшуючи тривалість обробки.

Асинхронність є концепцією програмування, яка означає, що дії можуть виконуватися паралельно та незалежно одна від одної. У контексті веб-додатків, асинхронність полягає в тому, що обробка запитів клієнта відбувається паралельно з іншими запитами, що дозволяє зменшити час очікування відповіді та покращити продуктивність веб-додатку [1]. Це забезпечує користувачам швидке та зручне використання додатків, а також дає можливість зменшити навантаження на мережу та підтримує їх безпеку.

Використання асинхронних методів також може допомогти забезпечити розширюваність та простоту масштабування веб-додатка, що дозволяє йому працювати з більшим обсягом даних та більшою кількістю користувачів.

Аналіз останніх досліджень і публікацій. Стосовно синхронної та асинхронної обробки запитів у веб-додатках на даний час є достатньо багато досліджень та розроблено ряд методів,

направлених на використання різного роду протоколів з метою покращення взаємодії між користувачами та сервісами [2].

Зокрема, в роботі [1] розглянуто можливості забезпечення надійності та безпеки застосування асинхронної технології, в [3] – проаналізовано можливість використання кешування результатів веб-сервісів для зменшення часу відповіді, в [4] – досліджено практичні аспекти взаємодії асинхронних методів у gRPC-додатках з використанням мови програмування Go.

Для забезпечення асинхронної взаємодії сервісів сьогодні використовуються такі відомі протоколи:

- gRPC: відкритий протокол, розроблений компанією Google. Він базується на протоколі HTTP/2 та підтримує різні мови програмування. gRPC використовує повідомлення Protocol Buffers для серіалізації даних та надає підтримку асинхронного програмування за допомогою промісів та потоків [5].
- WebSocket: протокол відкритого коду для двосторонньої комунікації між клієнтом та сервером в режимі реального часу [6]. Він може використовуватись для передачі різних типів даних, включаючи текст, бінарні дані, JSON, XML тощо. WebSocket також підтримує асинхронну взаємодію між клієнтом та сервером.
- MQTT: легкий протокол для IoT-пристроїв та мережі, який підтримує асинхронну взаємодію. MQTT використовує модель «підписки/публікації» для передачі повідомлень між клієнтами та серверами [7].
- AMQP: протокол для асинхронної взаємодії, який підтримує розподілену обробку та різні типи повідомлень. AMQP використовується для побудови великих систем з обробкою повідомлень та мікросервісів [8].
- Apache Kafka: розподілена платформа для обробки повідомлень, яка підтримує асинхронну взаємодію та забезпечує збереження повідомлень на сервері. Вона використовується для обробки великих потоків даних та побудови потужних систем з обробки повідомлень.

Постановка завдання. Вибір протоколу або фреймворку для асинхронної взаємодії сервісів залежить від конкретних потреб та напрямку проекту, оскільки описані протоколи мають свої переваги та недоліки. Наприклад, gRPC може бути корисним для веб-додатків, які характеризуються великою кількістю мікросервісів та потребують швидкої та ефективної взаємодії між ними. Він використовує бінарний формат обміну даними Protobuf, що зменшує розмір даних та збільшує продуктивність передачі.

WebSocket, з іншого боку, може бути корисним для веб-додатків, які потребують надійного та тривалого зв'язку між клієнтом та сервером, з можливістю обміну даними в режимі реального часу.

Перевагою AMQP є забезпечення захисту повідомлень за допомогою шифрування та автентифікації, що важливо для програм, де безпека даних є ключовою. Kafka забезпечує високий рівень надійності, завдяки реплікації та розподіленій архітектурі.

Тому подальші дослідження асинхронних методів сервісної взаємодії у веб-додатках, направлених на підвищення продуктивності та ефективності їх використання, покращення користувацького досвіду є актуальною задачею, яка потребує подальших як теоретичних розробок так і практичних вдосконалень.

Метою даної статті є розробка методу для покращення асинхронної взаємодії веб-сервісів з використанням протоколу gRPC на основі генерації Java класів, що дозволить підвищити швидкість обробки запитів та зменшити час очікування відповідей від сервера.

Викладення основного матеріалу дослідження. Для реалізації покращеного протоколу gRPC було розроблено веб-додаток, який дозволяє забезпечити асинхронну передачу даних від клієнта до сервера та між сервісами додатку при одночасному використанні. За основу для покращення було обрано додаток для анонімних дискусій, спілкування між користувачами в якому організовано за допомогою чату.

Запропонований клієнт-серверний додаток побудований на основі фреймворку SpringBoot та протоколів gRPC та Protobuf, додатково використано протокол WebSocket для отримання актуальної інформації для чатів. Така архітектура дозволяє з легкістю контактувати з веб-фронтендом за допомогою REST підходу та слідкувати за правильною роботою додатку. З іншої сторони веб-сервіс контактує з допоміжними чат-сервісом і логін-сервісом за допомогою протоколу gRPC та отримує відповідну інформацію (рис. 1).

Для зберігання інформації про чати, клієнтів та питання було використано базу даних PostgreSQL та створено відповідні таблиці chats, chat_clients, chat_questions. Перевагою використання PostgreSQL є те, що доступ з програм до бази даних здійснюється за допомогою спеціального процесу, під час якого клієнтські програми не можуть отримувати самостійний доступ до даних навіть у тому випадку, якщо вони функціонують на тому самому ПК, на якому здійснюється серверний процес.

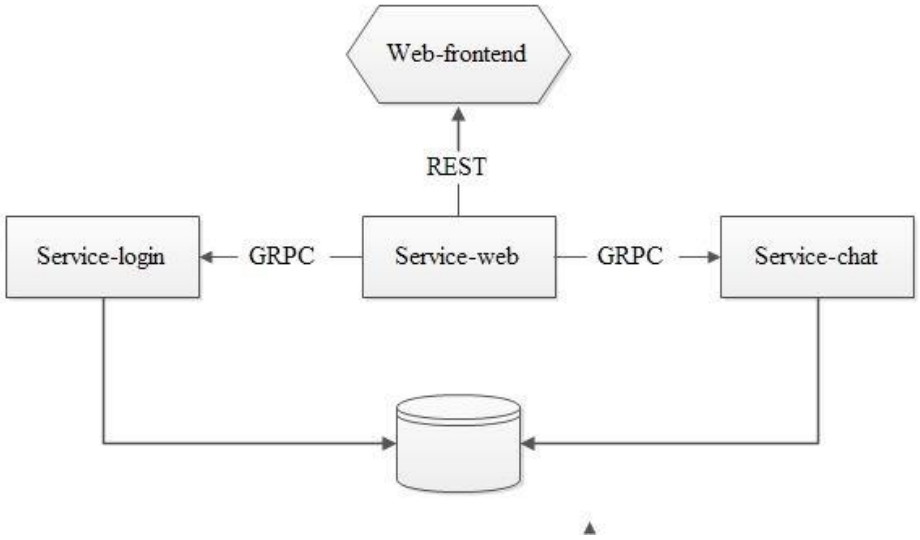

Рис. 1. Архітектура веб-додатку

Пересилка повідомлень Protocol Buffers (protobuf) за допомогою протоколу gRPC в програмному вигляді відбувається шляхом використання автоматично згенерованих клієнтських та серверних класів, створених на основі оголошень сервісу та повідомлень у файлах .proto.

Повідомлення protobuf представляються у бінарному компактному форматі. Кожен тип даних та повідомлення мають свій внутрішній байтовий формат для серіалізації та десеріалізації даних (табл.1):

Таблиця 1. Групи змінних у байтовому форматі

Тип	Значення	Використовується для
0	Varint	int32, int64, uint32, uint64, sint32, sint64, bool, enum
1	64-bit	fixed64, sfixed64, double
2	Length-delimited	string, bytes, embedded messages, packed repeated fields
3	Start group	groups (застарілий тип)
4	End group	groups (застарілий тип)
5	32-bit	fixed32, sfixed32, float

У protobuf кожен елемент даних має свій ключ (ціле число), яке ідентифікує елемент і тип даних, який визначає, які дані можуть бути включені в цей елемент. Типи даних включають цілі числа, змінні довжини, фіксовані довжини, числа з рухомою комою, булеві значення і рядки.

Однією з ключових особливостей gRPC є можливість генерувати клієнтський код різними мовами, такими як C++, Java, Python [9].

Цей код генерується компілятором protoc з файлів .proto. Згенерований код для класів Java на основі Protobuf може містити наступні методи:

1. Getters і Setters методи – для кожного поля в схемі Protobuf генерується відповідний метод доступу для отримання значення поля (getter) і метод для зміни значення поля (setter).
2. Методи серіалізації та десеріалізації – зазвичай генеруються методи для перетворення

об'єкта у формат Protobuf (серіалізація) і методи для аналізу даних, відформатованих Protobuf у внутрішню структуру об'єкта (десеріалізація).

3. Методи для порівняння об'єктів – генерується метод для порівняння двох об'єктів на рівність або нерівність.

4. Конструктори – генеруються конструктори для об'єктів, що дозволяє ініціалізувати об'єкт з відповідними значеннями.

5. Інші допоміжні методи – в залежності від фреймворка та специфікацій можуть генеруватися інші допоміжні методи для роботи з даними у форматі Protobuf.

На перший погляд, gRPC здавався ідеальним рішенням, але є певні проблеми, не прямо пов'язані з gRPC або Protobuf, але з виборами, зробленими конкретно в генерованому коді. Було вирішено покращити генерацію коду для структур з пакету com.google.protobuf, оскільки функціонал можна залишити незмінним, але спростити структуру класів. Структури, для яких вирішено покращити генерацію: DoubleValue, FloatValue, Int64Value, UInt64Value, Int32Value, UInt32Value, BoolValue, StringValue, BytesValue.

Для прикладу розберемо поле "question" типу "com.google.protobuf.StringValue". Генерований код надає можливість визначити чи поле містить певне значення, отримати значення у вигляді структури "com.google.protobuf.StringValue" та помістити значення того ж типу.

```
boolean hasQuestion();
com.google.protobuf.StringValue getQuestion();
public Builder setQuestion(com.google.protobuf.StringValue value);
```

Саме методи get та set є не завжди зручними і оптимальними, оскільки для полів типу string отримувати складну структуру, в якій міститься потрібне значення, є зайвим.

Теж саме стосовно методу set: для того щоб помістити значення, потрібно викликати білдер класу StringValue та виконати додаткові операції. Для даного прикладу створення об'єкта типу ChatQuestions з полем Question буде мати такий вигляд:

```
ChatQuestions.newBuilder()
 .setQuestion(StringValue.newBuilder()
 .setValue("value")
 .build())
 .build();
```

Під запропонованим покращення мається на увазі змінити генерацію коду Java класів опису proto повідомлень для комплексних структур типу …Value, з додаванням допоміжних методів get…Value та set…Value, які б приймали та віддавали прості типи без надбудов.

Таке покращення направлене на зменшення кількості операцій під час створення та отримання об'єктів, які містять …Value поля для збільшення перформенсу, а також для спрощення написання програм – оптимізації доступу до даних. Покращені класи нададуть більш швидкий та зручний доступ до даних, що сприяє підвищенню продуктивності додатку, а також покращення генерації коду призведе до зменшення споживання пам'яті та обчислювальних ресурсів.

Для перевірки ефективності запропонованого методу було проведено тестування роботи веб-додатку з використанням різної кількості користувачів та кількості даних.

Для вимірювання показників при навантаженні проекту до покращення та після було використано сервіс Guava від Google, оскільки він є простим інструментом для вимірювання часу та корисним для використання в будь-якому місці програми. Цей інструмент є потокобезпечним, тому можна використовувати кілька секундомірів паралельно без впливу на продуктивність. Кожне завдання, яке потрібно виміряти за допомогою секундоміра, зберігається в Hashtable.

Під час порівняння було проведено аналіз часу побудови protobuf об'єктів з та без покращення в розмірі 500000 об'єктів (рис. 2):

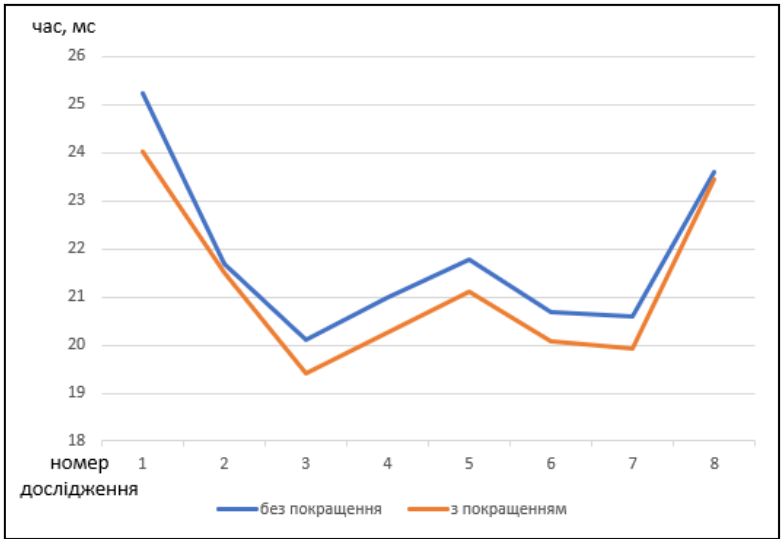

Рисунок 2. Порівняння продуктивності створення об'єктів

Створено діаграму для демонстрації результатів порівняння. При восьми перевірках результати мали певну розбіжність та в середньому результат показує, що покращення призвело до зменшення часу створення об'єктів в додатку при навантаженні на 2.8%.

Для наступного порівнянням було проведено аналіз часу отримання значення атрибута з уже створеного protobuf об'єкта з та без покращення в розмірі 500000 разів (рис. 3):

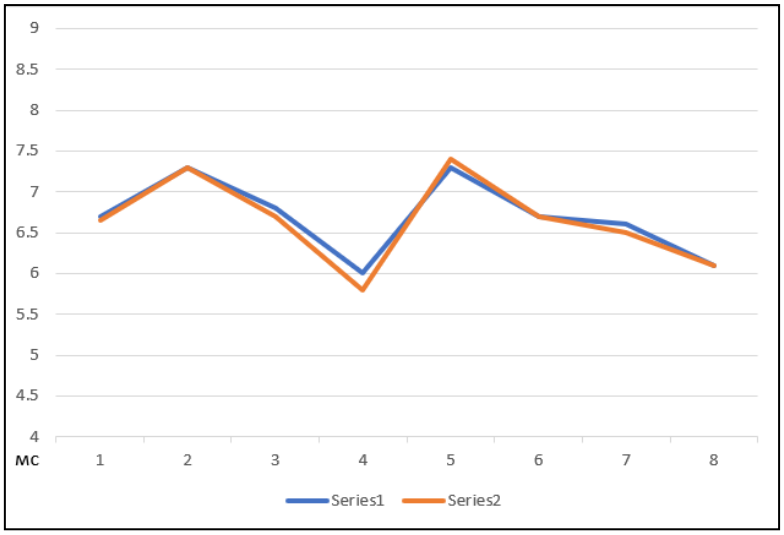

Рисунок 3. Порівняння продуктивності отримання значення атрибута

Створено діаграму для демонстрації результатів порівняння. Кількість перевірок така ж сама, результати також мали певну розбіжність. Проте в середньому результат показує, що покращення призвело до зменшення часу отримання значення атрибута в додатку при навантаженні на 0.4%.

Також для порівняння результатів передачі даних від клієнта до сервера протоколом gRPC було вирішено перевірити на прикладі двох тестових сценаріїв з різною кількістю користувачів. В першому сценарії брались до уваги 10 користувачів, в другому 100. Кожний користувач був представлений як окремий потік Java, але дані передавались через однаковий серверний стрім та розмір повідомлень був однаковий для кожного з них.

Щоб визначити залежність продуктивності протоколу від кількості повідомлень було вирішено розділити тестові дані на два різні набори даних. Таким чином, результатом першого тесту було отримання від сервера малої кількості повідомлень в розмірі 100 штук та 1000 штук в другому тесті.

Кожен набір даних перевірявся окремо в різному тестовому сценарії (табл. 2-3).

Таблиця 2. Результати порівняння для малої кількості паралельних підключень (10)

	100 повідомлень		1000 повідомлень	
	Без покращення	З покращенням	Без покращення	З покращенням
Середній час передачі, мс	0.029	0.028	1.27	1.24

Згідно таблиці 2 можна стверджувати, що покращення дає певне збільшення пропускної здатності. При малій кількості отриманих повідомлень покращення потребує на 3.5% менше часу та на 2.4% менше часу при більшій кількості надісланих даних.

Таблиця 3. Результати порівняння для великої кількості паралельних підключень (100)

	100 повідомлень		1000 повідомлень	
	Без покращення	З покращенням	Без покращення	З покращенням
Середній час передачі, мс	0.66	0.64	9.5	9.3

Відповідно до результатів, представлених в таблиці 3 видно, що покращення дозволяє збільшити також пропускну здатність при великій кількості паралельних підключень. При малій кількості отриманих повідомлень покращення потребує на 3% менше часу та на 2.1% менше часу ніж при більшій кількості надісланих даних

Висновки. В роботі виконано аналіз протоколів асинхронної взаємодії, які використовуються у веб-додатках з мето підвищення ефективності їх використання.

Запропоновано метод для покращення асинхронної взаємодії веб-сервісів з використанням протоколу gRPC за рахунок генерації Java класів на основі повідомлень protobuf, що дозволило шляхом спрощення структури класів надати більш швидкий та зручний доступ до даних та підвищити продуктивність додатку.

Тестування покращеного методу було виконано за допомогою веб-додатку для анонімних дискусій, створеного з використанням фреймворків gRPC, Protobuf, WebSocket, SpringBoot та бази даних PostgreSQL, застосування якого дало можливість практично реалізувати та продемонструвати ефективність його застосування. Отримані результати дослідження, проведеного у сервісі Guava із використанням звичайного та покращеного методів показали, що застосування запропонованої розробки дозволило зменшити час створення об'єктів в додатку при навантаженні на 2.8%, а час отримання значення атрибута в додатку при навантаженні зменшити на 0.4%. Також покращення дозволило збільшити пропускну здатність при великій кількості паралельних підключень.

Подальшим напрямком дослідження буде розробка методів, які дозволять оцінити та покращити часові параметри сервісної взаємодії для інших мов програмування, які підтримують генерацію gRPC з використанням опису сервісів Protobuf.

Список бібліографічного опису

1. М. Ємельянов, В. Шпак, "Асинхронна обробка запитів в RESTful веб-сервісах на основі Node.js", *збірник "Технічні науки та технології: теорія та практика"*, 2014, С. 36-44.
2. С. Крузе, О. Союз, "Synchronous and Asynchronous Web Service Interaction", *International Journal of Web Services Rese*arch. 2013.
3. М. Дутчак, М. Пікуляк, "Оцінка якості методів та програмних засобів інтелектуальних адаптивних освітніх вебсистем самонавчання," *2022 International Conference on Innovative Solutions in Software Engineering (ICISSE),* Прикарпатський національний університет імені В. Стефаника, Івано-Франківськ, Україна, 29-30 листопада 2022 р., С. 266-270.
4. S. Varghese. Building High Performance APIs in Go Using gRPC and Protocol Buffers [Електронний ресурс]. Режим доступу: https://shijuvar.medium.com/building-high-performance-apis-in-go-using-grpc-and-protocol-buffers-2eda5b80771b. Дата звернення: 18.10.2023.
5. gRPC – A high performance, open source universal RPC framework. [Електронний ресурс]. Режим доступу: https://grpc.io/. Дата звернення: 16.09.2023.

6. WebSocket. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/WebSocket. Дата звернення: 26.10.23.
7. Чому MQTT є важливою частиною IoT? [Електронний ресурс]. Режим доступу: https://fiberroad.com/uk/why-is-mqtt-an-important-part-of-iot/. Дата звернення: 21.10.2023.
8. AMQP is the Internet Protocol for Business Messaging. [Електронний ресурс]. Режим доступу: https://www.amqp.org/about/what. Дата звернення: 20.11.2023.
9. S. G. Du, J. W Lee, K. Kim "Proposal of GRPC as a new northbound API for application layer communication efficiency in SDN, " *IMCOM '18: the 12th international conference on ubiquitous information management and communication, Langkawi Malaysi*a. New York, NY, USA, 2018. DOI: https://doi.org/10.1145/3164541.3164563.

References

1. M. Yemelianov, V. Shpak, "Asynkhronna obrobka zapytiv v RESTful veb-servisakh na osnovi Node.js", *zbirnyk "Tekhnichni nauky ta tekhnolohii: teoriia ta praktyka*", 2014, pp. 36-44.
2. S. Kruze, O. Soiuz, "Synchronous and Asynchronous Web Service Interaction", *International Journal of Web Services Research*. 2013.
3. M. Dutchak, M. Pikuliak, "Otsinka yakosti metodiv ta prohramnykh zasobiv intelektualnykh adaptyvnykh osvitnikh vebsystem samonavchannia," *2022 International Conference on Innovative Solutions in Software Engineering (ICISSE)*, Vasyl Stefanyk Precarpathian National University, Ivano-Frankivsk, Ukraine, Nov. 29-30, 2022, pp. 266-270.
4. S. Varghese. Building High Performance APIs in Go Using gRPC and Protocol Buffers [Elektronnyi resurs]. Rezhym dostupu: https://shijuvar.medium.com/building-high-performance-apis-in-go-using-grpc-and-protocol-buffers-2eda5b80771b. Data zvernennia: 18.10.2023.
5. gRPC – A high performance, open source universal RPC framework. [Elektronnyi resurs]. Rezhym dostupu: https://grpc.io/. Data zvernennia: 16.09.2023.
6. WebSocket. [Elektronnyi resurs]. Rezhym dostupu: https://uk.wikipedia.org/wiki/WebSocket. Data zvernennia: 26.10.23.
7. Chomu MQTT ye vazhlyvoiu chastynoiu IoT? [Elektronnyi resurs]. Rezhym dostupu: https://fiberroad.com/uk/why-is-mqtt-an-important-part-of-iot/. Data zvernennia: 21.10.2023.
8. AMQP is the Internet Protocol for Business Messaging. [Elektronnyi resurs]. Rezhym dostupu: https://www.amqp.org/about/what. Data zvernennia: 20.11.2023.
9. S. G. Du, J. W Lee, K. Kim "Proposal of GRPC as a new northbound API for application layer communication efficiency in SDN, " *IMCOM '18: the 12th international conference on ubiquitous information management and communication, Langkawi Malaysi*a. New York, NY, USA, 2018. DOI: https://doi.org/10.1145/3164541.3164563.