УДК 004.42.519.8 DOI https://doi.org/10.35546/kntu2078-4481.2024.1.38

Б. В. ПАШКОВСЬКИЙ
кандидат технічних наук,
доцент кафедри комп'ютерних систем і мереж
Івано-Франківський національний технічний університет нафти і газу
ORCID: 0000-0003-1082-6837

М. І. СЛАБІНОГА
кандидат технічних наук, доцент,
доцент кафедри комп'ютерних систем і мереж
Івано-Франківський національний технічний університет нафти і газу
ORCID: 0000-0002-7296-0356

М. В. РОМАНІВ
студент кафедри комп'ютерних систем і мереж
Івано-Франківський національний технічний університет нафти і газу
ORCID: 0009-0005-4066-2898

ОПТИМІЗАЦІЯ РОБОТИ ВЕБ-ЗАСТОСУНКІВ ЗАСОБАМИ ГОРИЗОНТАЛЬНОГО МАСШТАБУВАННЯ З ВИКОРИСТАННЯМ АРХІТЕКТУРИ CQRS

У сучасному світі, де швидкість, продуктивність та масштабованість є критичними факторами успіху, оптимізація роботи веб-застосунків є невід'ємною частиною процесу розробки. За допомогою горизонтального масштабування та архітектури CQRS (Command Query Responsibility Segregation), розробники можуть досягти значних покращень у продуктивності та ефективності своїх веб-застосунків.

Оптимізація роботи за критерієм максимальної продуктивності веб-застосунків вимагає пошуку інноваційних підходів, які дозволять забезпечити швидкий та ефективний доступ до даних, обробку великих обсягів даних інформації навіть при зростаючому навантаженні.

Горизонтальне масштабування веб-застосунків дозволяє розподілити навантаження між багатьма серверами, що дає змогу обробляти більшу кількість запитів паралельно та підтримувати стабільну продуктивність системи. Архітектура CQRS надає можливості розділити операції читання та запису, використовуючи оптимізовані шляхи оброблення для кожного типу операцій. Це знижує складність системи, полегшує розширення та підтримку, а також дозволяє використовувати різні технології та інструменти для операцій читання та запису.

Застосування горизонтального масштабування та архітектури CQRS є актуальними з точки зору розвитку сучасних веб-застосунків, які стикаються з викликами обробки великих обсягів даних, швидкими змінами вимог користувачів та необхідністю забезпечення високої доступності та продуктивності. Ці підходи дозволяють створити масштабовані, ефективні та легкозрозумілі системи, що задовольняють потреби сучасних веб-застосунків.

У даній роботі розглянуто актуальність оптимізації роботи веб-застосунків з використанням горизонтального масштабування та архітектури CQRS. Проведені дослідження основних компонентів архітектури CQRS, переваги та недоліки такого підходу, а також способи поєднання цих підходів для досягнення оптимальної продуктивності та масштабованості. Також будуть розглянуті ключові фактори успішної оптимізації та приклади реалізації цих підходів у практичних веб-проектах.

Ключові слова: *масштабування, реплікація, веб-застосунок, CQRS, база даних.*

B. V. PASHKOVSKYI
Candidate of Technical Science,
Associate Professor at the Department of Computer Systems and Networks
Ivano-Frankivsk Technical University of Oil and Gas
ORCID: 0000-0003-1082-6837

M. I. SLABINOHA
Candidate of Technical Science, Associate Professor
Associate Professor at the Department of Computer Systems and Networks
Ivano-Frankivsk Technical University of Oil and Gas
ORCID: 0000-0002-7296-0356

M. V. ROMANIV
Student at the Department of Computer Systems and Networks
Ivano-Frankivsk Technical University of Oil and Gas
ORCID: 0009-0005-4066-2898

WEB APPLICATION PERFORMANCE OPTIMIZATION WITH CQRS AND HORIZONTAL SCALING

In today's world, where speed, performance, and scalability are critical success factors, web application performance optimization is an integral part of the development process. With horizontal scaling and the CQRS (Command Query Responsibility Segregation) architecture, developers can achieve significant improvements in the performance and efficiency of their web applications.

Optimizing work according to the criterion of maximum performance of web applications requires the search for innovative approaches that will allow to ensure fast and efficient access to data, processing of large volumes of data and information even with increasing load.

Horizontal scaling of web applications allows you to distribute the load among many servers, which allows you to process more requests in parallel and maintain stable system performance. The CQRS architecture provides the ability to separate read and write operations, using optimized processing paths for each type of operation. It reduces the complexity of the system, makes it easier to expand and maintain, and allows the use of different technologies and tools for read and write operations.

The application of horizontal scaling and CQRS architecture is relevant for the development point of view of modern web applications, which arise with the challenges of processing large volumes of data, rapid changes in user requirements, and ensuring high availability and performance. These approaches will make it possible to create scalable, efficient and easy-to-understand systems that meet the needs of modern web applications.

This work considers the relevance of web application optimization using horizontal scaling and CQRS architecture. The main components of the CQRS architecture, the advantages and disadvantages of such an approach, and how to combine these approaches to achieve optimal performance and scalability are explored. Key factors of successful optimization and examples of implementation of these approaches in practical web projects will also be considered.

Key words: *scaling, replication, web-application, CQRS, database.*

Постановка проблеми

Зі зростанням популярності web-застосунку його підтримка неминуче починає вимагати все більших і більших ресурсів. Спочатку з навантаженням можна і, безсумнівно, потрібно боротися шляхом оптимізації алгоритмів та/або архітектури самого додатка.

За допомогою горизонтального масштабування та архітектури CQRS розробники можуть досягти значних покращень у продуктивності та масштабованості своїх веб-застосунків.

Перш за все, зростає попит на веб-застосунки з високою продуктивністю, які здатні обробляти великі обсяги даних та забезпечувати швидкий доступ для користувачів. Застосування горизонтального масштабування дає змогу розподіляти навантаження між багатьма серверами, що забезпечує паралельну обробку запитів та збільшує загальну продуктивність системи.

Архітектура CQRS надає можливості розділити операції читання (queries) та запису (commands), що дозволяє використовувати оптимізовані шляхи обробки для кожного типу операцій. Це знижує складність системи, полегшує розширення та підтримку, а також дозволяє використовувати різні технології та інструменти для операцій читання та запису.

Веб-інтерфейси API слугують основою сучасних веб-додатків, забезпечуючи взаємодію між різними програмними компонентами. Тому їхня продуктивність безпосередньо впливає на загальну продуктивність веб-додатків.

Аналіз останніх досліджень і публікацій

У роботі [1] використана модель актора, event sourcing та CQRS для покращення масштабованості, продуктивності системи для маркетингових систем лояльності. Показано, що ця архітектура приносить приріст продуктивності.

Автори [2] розглядають застосування CQRS для побудови великого продукту. Виявлено та обговорено сім підшаблонів, пов'язаних з CQRS. Результати дослідження показують, що шаблон CQRS реалізований і як його різні підшаблони можуть призвести до високого рівня варіативності в програмному продукті та як різні підшаблони можуть взаємодіяти для досягнення цього.

У роботі [3] автори застосовують CQRS для роботи із денормалізованою базою даних у медичній інформаційній системі закладів первинної медичної допомоги.

Формулювання мети дослідження

Метою роботи є дослідження, виявлення, аналіз та впровадження оптимальних підходів до оптимізації роботи веб-застосунків з використанням горизонтального масштабування та архітектури CQRS. Дослідження спрямоване на пошук ефективних стратегій розподілу навантаження, покращення продуктивності, забезпечення масштабованості та зниження витрат на інфраструктуру веб-застосунків.

Для досягнення цієї мети необхідно розв’язати такі завдання:

1 Дослідити принципи та переваги горизонтального масштабування в парі з архітектурою CQRS у контексті веб-застосунків;

2 Проаналізувати існуючі методи та стратегії горизонтального масштабування для веб-застосунків;

3 Дослідити практики та інструменти, які допомагають оптимізувати роботу веб-застосунків з використанням горизонтального масштабування та архітектури CQRS;

4 Впровадити архітектури CQRS у веб-застосунку та оцінити його вплив на продуктивність та масштабованість;

5 Експериментально порівняти різні підходи до оптимізації роботи веб-застосунків та провести оцінку їх ефективності.

Викладення основного матеріалу дослідження

CQRS – підхід проектування програмного забезпечення, при якому код, що змінює стан, відокремлюється від коду, що просто читає цей стан. Подібний поділ може бути логічним і ґрунтуватися на різних рівнях. Крім того, воно може бути фізичним і включати різні ланки (tiers) або рівні. [4]

Основна ідея цього принципу полягає в тому що архітектура застосунку ділиться на:

– queries: методи повертають результат, не змінюючи стан об'єкта. Інакше кажучи, у Query немає побічних ефектів;

– commands: методи змінюють стан об'єкта, не повертаючи значення.

Розглянемо різницю класичної CQRS структури на рисунках 1 та 2.

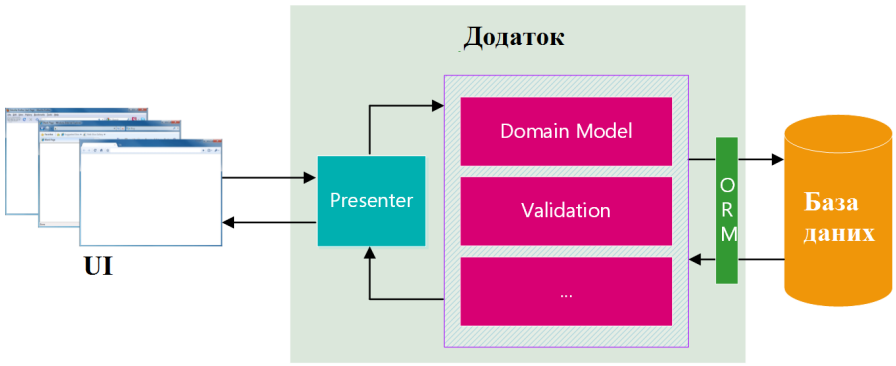

Рис. 1. Класична архітектура веб-застосунку

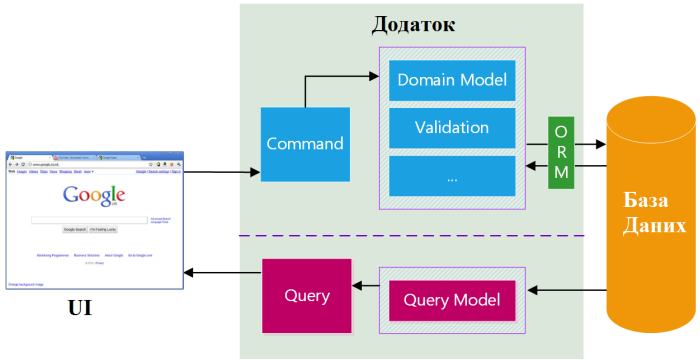

Рис. 2. Архітектура з використанням CQRS

Поділ, що переслідується CQRS, досягається групуванням операцій запиту на одному рівні, а команд – на іншому. Кожен рівень має свою модель даних, свій набір сервісів та створюється із застосуванням своєї комбінації шаблонів та технологій. Ще важливіше, що ці два рівні можуть бути навіть у двох різних ланках (tiers) і оптимізуватися окремо, ніяк не торкаючись один одного [5].

Просте розуміння того, що команди та запити є різними речами, глибоко впливає на архітектуру ПЗ. Наприклад, раптом стає легше передбачати та кодувати кожен рівень предметної області. Рівень предметної області (domain layer) стеку команд потребує лише даних, бізнес-правилах і правил безпеки виконання завдань. З іншого боку, рівень предметної області в стеку запитів може бути не складнішим за прямий SQL-запит [6].

Для прикладу розглянемо кінцеву схему з CQRS та двома різними сховищами даних представлено на рисунку 3.

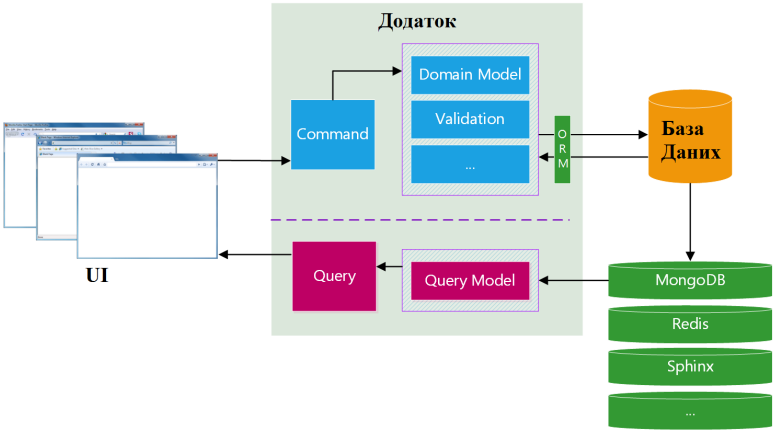

Рис. 3. Архітектура CQRS з двома різними сховищами даних

Спроектуємо тестовий інтернет магазин для порівняння ефективності застосування архітектури CQRS.

Для початку представимо базові сутності які будуть існувати в додатку. Діаграма класів для базових сутностей представлена на рисунку 4.

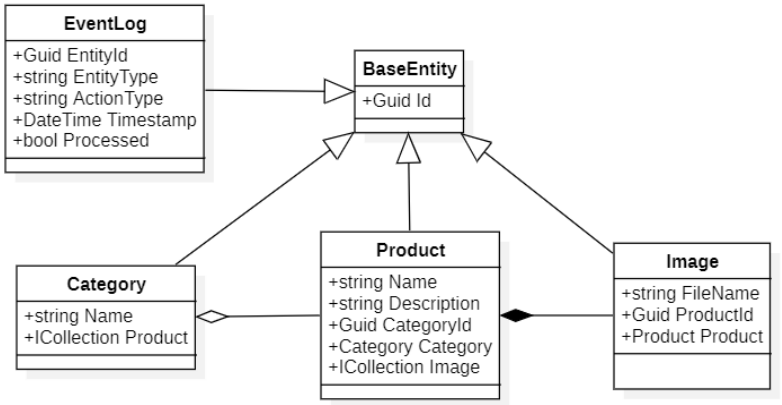

Рис. 4. Діаграма класів сутностей які існують в системі

Сутності та їх властивості представлені в таблицях 1–5.

Таблиця 1

Таблиці бази даних

Назва таблиці	Призначення
Categories	Збереження категорій продуктів
Products	Збереження продуктів
Image	Збереження фотографій продуктів
EventLogs	Збереження результатів операції з системою

Таблиця 2

Опис таблиці Categories

Назва стовпця	Тип даних	Призначення
Id	uniqueidentifier	Ід категорії
Name	nvarchar(MAX)	Назва категорії

Таблиця 3

Опис таблиці Image

Назва стовпця	Тип даних	Призначення
Id	uniqueidentifier	Ід фото
FileName	nvarchar(MAX)	Назва файлу
ProductId	uniqueidentifier	Зовнішній ключ, який вказує на ідентифікатор продукту

Таблиця 4

Опис таблиці Products

Назва стовпця	Тип даних	Призначення
Id	uniqueidentifier	Ід категорії
Name	nvarchar(MAX)	Назва продукту
Description	nvarchar(MAX)	Опис продукту
CategoryId	uniqueidentifier	Зовнішній ключ, який вказує на ідентифікатор категорії

Таблиця 5

Опис таблиці EventLogs

Назва стовпця	Тип даних	Призначення
Id	uniqueidentifier	Ід операції
EntityId	uniqueidentifier	Зовнішній ключ, який вказує на ідентифікатор сутності
EntityType	nvarchar(MAX)	Тип сутності з якою проводилась операція
ActionType	nvarchar(MAX)	Опис операції що відбувалась над сутністю
Timestamp	datetime2(7)	Час операції
Processed	bit	Прапорець, що вказує чи відбулася синхронізація між базами даних

Тип даних uniqueidentifier – це тип даних у базі даних для зберігання унікальних ідентифікаторів (GUID).

nvarchar(MAX) – тип даних для зберігання Unicode-рядків завдовжки до максимально можливого обсягу (2^31-1 або 2147483647 символів (2 ГБ)) [7].

Аналогічно створюється база і колекції MongoDB які будуть додаватися з бази даних SQL. База даних MongoDB представлена на рисунку 5.

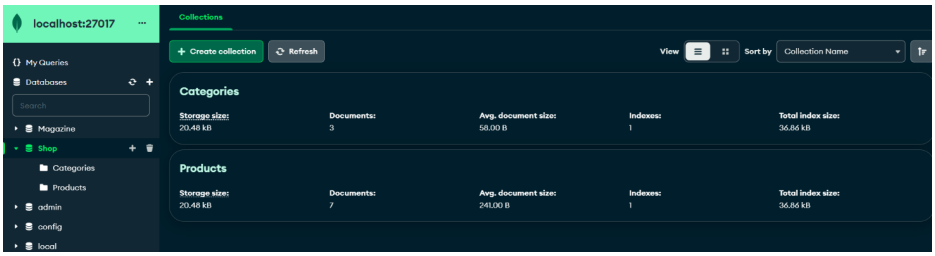

Рис. 5. Результати успішної синхронізації з базою даних SQL

Для реалізації поставленого завдання обрано мову програмування C# та платформа розробки веб-застосунків ASP.NET.

Структура реалізованого застосунку представлена на рисунку 6.

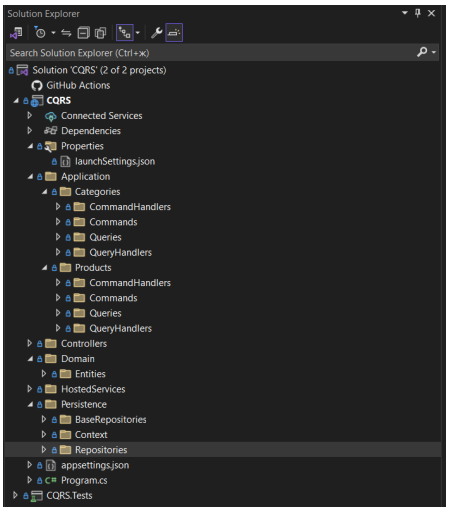

Рис. 6. Структура проєкта

Для визначення ефективності застосування архітектурного підходу CQRS проведемо серію порівняльних тестів, де ми будемо зчитувати по 1000 продуктів із баз даних розташованих у різних регіонах. Зчитування будемо проводити п’ятьма серіями по 100 запитів.

Отримані результати представлені в таблицях 6–9.

Таблиця 6

Результати зчитування з бази даних розташованої у Великій Британії

Кількість запитів	Кількість продуктів	Розташування БД	Час в секундах
100	1000	UK West	16.6373418
100	1000	UK West	17.5564886
100	1000	UK West	18.3571659
100	1000	UK West	19.9817602
100	1000	UK West	14.9301036

Таблиця 7

Результати читання з бази даних на Західному узбережжі США

Кількість запитів	Кількість продуктів	Розташування БД	Час в секундах
100	1000	West US	54.421575
100	1000	West US	52.7376366
100	1000	West US	59.4390936
100	1000	West US	62.0625805
100	1000	West US	60.6093189

Таблиця 8

Результати читання з бази даних розташованої в Польщі

Кількість запитів	Кількість продуктів	Розташування БД	Час в секундах
100	1000	PL	8.5560043
100	1000	PL	9.5461138
100	1000	PL	9.3760551
100	1000	PL	14.8564275
100	1000	PL	9.5309906

Таблиця 9

Зведена таблиця результатів тестування

Кількість запитів	Кількість продуктів	Розташування БД	Середній час, с.
100	1000	PL	10.37311826
100	1000	West US	57.85404092
100	1000	UK West	17.49257202

Таким чином можна зробити висновок, що розташування бази даних має суттєвий вплив на час відповіді застосунку. Тому застосування технології CQRS дозволяє використати репліку бази даних для читання і розташувати її максимально близько до кінцевого користувача.

Висновки

У роботі розроблено структуру архітектури веб-застосунку за допомогою та здійснено проектування БД.

Проведено порівняльний аналіз часу відповіді веб-застосунку в залежності від розташування бази даних.

Показано, що час відповіді застосунку збільшується при збільшенні відстані від користувача до бази даних.

Таким чином застосування технології CQRS дозволяє використати репліку бази даних для читання і розташувати її максимально близько до кінцевого користувача, що призводить до значного прискорення роботи системи порівняно з варіантом без його використання.

Список використаної літератури

1. Беттс, Домінік та ін. «Дослідження CQRS і джерела подій: подорож до високої масштабованості, доступності та зручності обслуговування з Windows Azure». 2013.

2. Каббедійк, Яап, Слінгер Янсен і Сяак Брінккемпер. «Прикладне дослідження наслідків мінливості шаблону CQRS у програмному забезпеченні для онлайн-бізнесу». Матеріали 17-ї Європейської конференції з мов шаблонів програм. 2012 рік. https://doi.org/10.1145/2602928.2603078

3. Райкович, Петар, Драган Янкович і Александар Міленкович. «Використання шаблону CQRS для покращення продуктивності медичних інформаційних систем». Proc. 6-ї Балканської конференції з інформатики. 2013 рік.

4. О. Л. Ярош Р. М. Бабаков Методи оптимізації продуктивності веб-застосунків. URL: https://jait.donnu.edu.ua/article/view/13969

5. Оптимізація веб застосунків: лайфхаки та інструменти. [Електронний ресурс]. Режим доступу: https://dou.ua/forums/topic/43011/

6. Вертикальне та горизонтальне масштабування Вступні відомості про масштабованість баз даних під час хмарних обчислень. [Електронний ресурс] Режим доступу: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/scaling-out-vs-scaling-up

7. Marcelo R, IBM Redbook: The RS/6000 SP Inside Out. Bernard Woo, New York 1999. P. 572.

References

1. Betts, D., Dominguez, J., Melnik, G., Simonazzi, F., & Subramanian, M. (2013). Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure.

2. Kabbedijk, J., Jansen, S., & Brinkkemper, S. (2012, July). A case study of the variability consequences of the CQRS pattern in online business software. In Proceedings of the 17th European Conference on Pattern Languages of Programs (pp. 1–10). https://doi.org/10.1145/2602928.2603078

3. Rajković, P., Janković, D., & Milenković, A. (2013, January). Using cqrs pattern for improving performances in medical information systems. In Proc. of the 6th Balkan Conference in Informatics (pp. 86-91).

4. O. L. Yarosh R. M. Babakov (2013, July) Methods of optimizing the performance of web applications URL: https://jait.donnu.edu.ua/article/view/13969

5. Vyazovoy, O. (2023, April 17). Оптимізація вебзастосунків: лайфхаки та інструменти. DOU. https://dou.ua/forums/topic/43011/

6. Scaling out vs scaling up: Microsoft Azure. Scaling Out vs Scaling Up | Microsoft Azure. (n.d.). https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/scaling-out-vs-scaling-up

7. Marcelo R, IBM Redbook: The RS/6000 SP Inside Out. Bernard Woo, New York 1999. P. 572.