УДК 004.8

М. М. Лесюк[1]

В. І. Перепелиця[1]

ПЕРСПЕКТИВИ ВИКОРИСТАННЯ ПАТЕРНІВ CQRS ТА MEDIATR ДЛЯ РОЗРОБКИ СУЧАСНИХ ENTERPRISE РІШЕНЬ

[1]Вінницький національний технічний університет

Анотація

У роботі розглянуто застосування архітектурних патернів CQRS та Mediatr у сучасних Enterprise системах. Проаналізовано їхні переваги у підвищенні масштабованості, гнучкості та надійності програмного забезпечення. Визначено ключові аспекти впровадження та описано перспективи їх подальшого розвитку в умовах зростання складності корпоративних рішень.

Ключові слова: CQRS, Mediatr, архітектурні патерни, Enterprise-системи, мікросервіси, розподілені системи.

Abstract

The paper examines the application of the CQRS and Mediatr architectural patterns in modern enterprise systems. Their advantages in improving scalability, flexibility, and reliability are analyzed. Key implementation aspects and the prospects for further development under increasing complexity of corporate solutions are discussed.

Keywords: CQRS, Mediatr, architectural patterns, enterprise systems, microservices, distributed systems.

Вступ

Сучасні Enterprise-рішення характеризуються високою складністю бізнес-логіки, значними обсягами даних та вимогами до масштабованості, гнучкості й надійності. Традиційні монолітні архітектури часто не можуть забезпечити необхідний рівень продуктивності та адаптивності у швидкозмінному середовищі. Саме тому розробники все частіше звертаються до архітектурних патернів, спрямованих на розподіл відповідальностей і спрощення обробки даних [1].

Одними з таких інструментів є патерни CQRS (Command Query Responsibility Segregation) та Mediator/Mediatr, які дозволяють значно підвищити ефективність систем за рахунок розділення операцій читання і запису, спрощення внутрішніх взаємодій між компонентами та забезпечення слабкого зв’язування [2].

У даній роботі розглянуто особливості використання CQRS та Mediatr, їх переваги над класичними підходами, а також можливості подальшого розвитку в контексті сучасних вимог до Enterprise-рішень.

Результати дослідження

Архітектурний патерн CQRS передбачає розділення операцій читання (Query) та запису (Command) на окремі моделі. На практиці це дозволяє оптимізувати продуктивність та масштабування системи завдяки незалежній еволюції кожної частини. Крім того, CQRS добре поєднується з такими патернами, як Event Sourcing, що сприяє повному аудиту змін у системі [2].

У традиційних CRUD-системах одна й та сама модель домену обробляє як читання, так і записи, що призводить до збільшення складності, дублювання логіки та труднощів масштабування. У великих корпоративних рішеннях це створює ризики зниження продуктивності й виникнення конфліктів при роботі з даними [3].

Використання CQRS дозволяє:

– знизити навантаження на основні моделі домену та бази даних;
– оптимізувати запити під потреби читання, без впливу на логіку запису;
– спростити тестування та відлагодження.
– забезпечити масштабування лише тих частин системи, які цього потребують.

Другим ключовим інструментом є Mediator, який реалізує шаблон «Посередник» та концентрує обробку запитів у єдиній точці. У .NET екосистемі особливо популярною є бібліотека MediatR, що дозволяє інкапсулювати логіку обробників, зменшити кількість залежностей та забезпечити слабке зв'язування між компонентами [4-5].

Переваги Mediatr полягають у:

– усуненні прямої залежності між контролерами, сервісами й репозиторіями;
– спрощенні структури застосунку;
– можливості легко додавати крос-функціональну логіку (логування, кешування, валідацію);
– підвищенні модульності та гнучкості надбудови бізнес-логіки.

У поєднанні CQRS та Mediatr формують потужний підхід до створення Enterprise-рішень. Mediatr дозволяє стандартизувати роботу з командами та запитами, тоді як CQRS — чітко відокремити операції читання від змін стану системи. Такий підхід добре масштабується як у межах монолітів, так і в мікросервісних архітектурах [6-7].

Сьогодні ці патерни широко застосовуються у розробці складних систем управління ресурсами, фінансових платформ, CRM/ERP-рішень, сервісів обробки великих обсягів даних. Вони дозволяють одночасно забезпечити високу продуктивність, надійність та структурованість бізнес-логіки.

Перспективи використання CQRS та Mediatr пов'язані із зростанням потреби у побудові подійно-орієнтованих систем, удосконаленням підходів до паралельної обробки даних та інтеграцією з хмарними сервісами. Зокрема, розвиток хмарних платформ, таких як Azure або AWS, відкриває нові можливості для масштабування систем, побудованих за патернами CQRS [8-10].

Висновки

Патерни CQRS та Mediatr є потужними інструментами, які дозволяють оптимізувати архітектуру сучасних Enterprise-рішень, підвищити їх продуктивність, масштабованість і гнучкість. Розділення операцій читання та запису, централізація логіки обробки запитів і зменшення зв'язності компонентів сприяють створенню надійних та розширюваних систем.

Перспективи подальшого розвитку пов’язані з інтеграцією CQRS із подійно-орієнтованими підходами, розвитком мікросервісних архітектур і переходом до хмарних середовищ. Це робить зазначені патерни актуальними для широкого спектру галузей та типів програмного забезпечення.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Young, G. CQRS Documents and Presentations [Електронний ресурс]. – Режим доступу: https://cqrs.files.com
2. Microsoft Architecture Center. CQRS and Event Sourcing patterns [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/azure/architecture/
3. Bogard J. MediatR Library Documentation [Електронний ресурс]. – GitHub. – Режим доступу: https://github.com/jbogard/MediatR
4. Fowler, M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2003. – 560 p.
5. Vernon, V. Implementing Domain-Driven Design. – Addison-Wesley, 2013. – 656 p.
6. Newman, S. Building Microservices: Designing Fine-Grained Systems. – O’Reilly Media, 2015. – 280 p.

7. Clemmons, A. Hands-On Domain-Driven Design with .NET Core. – Packt Publishing, 2020. – 440 p.
8. Richardson, C. Microservices Patterns: With Examples in Java. – Manning Publications, 2019. – 520 p.
9. Laganière P. Clean Architecture in .NET 6. – Packt Publishing, 2022. – 450 p.
10. Evans, E. Domain-Driven Design: Tackling Complexity in the Heart of Software. – Addison-Wesley, 2004. – 560 p.

Лесюк Михайло Михайлович – студент групи 1КН-24м кафедри «Комп'ютерних наук», факультет інтелектуальних інформаційних технологій та автоматицізації, Вінницький національний технічний університет, м. Вінниця.
Перепелиця В'ячеслав Ігорович – доктор філософії, асистент кафедри «Комп'ютерних наук», e-mail: pvi_92@ukr.net, Вінницький національний технічний університет, м. Вінниця.
Lesiuk Mykhailo Mykhailovych – student of group 4KN-21b of the Department of Computer Science, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia.
Perepelytsia Viacheslav Ihorovych - PhD, assistant of the Department of Computer Science, e-mail: pvi_92@ukr.net; Vinnytsia National Technical University, Vinnytsia