ларусь, Гомельский гос. ун-т им. Ф. Скорины, Главн. управл. образования Гомельского облисполкома ; редкол. : И. В. Семченко (гл. ред.) [и др.]. – Гомель : ГГУ им. Ф. Скорины, 2021. – С. 52–55.

В. В. Невзоров, Е. В. Рафалова
(*ГГУ имени Ф. Скорины, Гомель*)

ПАТТЕРН CQRS В РАМКАХ РАЗРАБОТКИ WEB API НА БАЗЕ ФРЕЙМВОРКА ASP.NET CORE

Паттерны проектирования представляют собой универсальные решения для распространенных архитектурных задач, возникающих в процессе создания программного обеспечения.

Паттерн CQRS (Command Query Responsibility Segregation) предполагает разделение операций на две категории: команды и запросы. В отличие от классической CRUD-архитектуры, где одна модель отвечает и за запись, и за чтение, CQRS предполагает независимые модели для каждой задачи. Команды акцентированы на бизнес-правилах, валидации и согласованности данных, гарантируя, что каждое изменение соответствует ключевым требованиям системы. Запросы, напротив, оптимизированы для быстрого доступа к информации – здесь допустимы денормализация, кэширование или использование специализированных хранилищ. Такое разделение не только повышает производительность, но и упрощает анализ системы, разработчики могут модифицировать логику записи и чтения независимо, избегая конфликтов между компонентами. CQRS особенно эффективен в сценариях, где нагрузка на запись и чтение неравномерна, а требования к масштабируемости критичны.

Библиотека MediatR, активно используемая в экосистеме .NET, предоставляет инструменты для реализации CQRS через паттерн «Посредник». Она устраняет прямые зависимости между компонентами, перенаправляя команды и запросы через центральный механизм обработки. Каждая операция определяется как отдельный объект-сообщение, а его логика инкапсулируется в специализированном обработчике. Это позволяет разделить сложную бизнес-логику на независимые модули, которые легко тестировать и модифицировать.

MediatR также поддерживает pipeline-обработку, что дает возможность внедрять сквозные функции через промежуточные компоненты. Например, валидация команды может быть вынесена в отдельный обработчик, который автоматически проверяет данные перед выполнением основной логики. Библиотека интегрируется с DI-контейнером ASP.NET Core, обеспечивая бесшовное взаимодействие с инфраструктурой приложения и ускоряя переход на CQRS-архитектуру.

Т. С. Новик
(*ГрГУ имени Янки Купалы, Гродно*)

ПРЕИМУЩЕСТВА И НЕДОСТАТКИ ИСПОЛЬЗОВАНИЯ ГОТОВЫХ ДИЗАЙН-БИБЛИОТЕК КОМПОНЕНТОВ

Использование готовых дизайн-библиотек компонентов, таких как Material UI, Consta и Ant Design, является популярным подходом в разработке интерфейсов пользователей. Эти инструменты позволяют ускорить процесс создания приложений и обеспечить единообразие в дизайне. Однако, как и любое технологическое решение, они имеют как сильные стороны, так и определенные недостатки.

Одним из главных преимуществ готовых дизайн-библиотек является значительное сокращение времени на разработку.

Еще одним ключевым преимуществом является поддержка темизации. Это позволяет легко адаптировать приложение под предпочтения пользователей или корпоративный стиль. Кроме того, библиотеки часто разрабатываются с учётом принципов адаптивного дизайна.

Однако использование готовых дизайн-библиотек не лишено недостатков. Одной из основных проблем является их размер. Крупные библиотеки содержат множество компонентов, которые могут так и не понадобиться в конкретном проекте, но всё равно увеличивают размер сборки приложения.

Другой распространённой сложностью является ограниченная кастомизация – готовые библиотеки не всегда позволяют реализовать уникальные дизайны.

Зависимость от сторонних решений также вызывает определённые риски. Библиотеки постоянно обновляются, и их разработчики мо-