В. А. Куриленко
Научный руководитель
А. Н. Семенюта
Белорусский торгово-экономический
университет потребительской кооперации
г. Гомель, Республика Беларусь

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ РЕАЛИЗАЦИИ АВТОМАТИЗИРОВАННЫХ СИСТЕМ НА ОСНОВЕ EXPRESS.JS + REACT И ASP.NET CORE MVC

Современные реалии предъявляют высокие требования к качеству, скорости работы и доступности веб-сервисов, особенно в таких прикладных областях, как автоматизированные системы онлайн-записи на услуги. Учитывая это, разработчику необходимо выбирать такие инструменты, которые обеспечивают не только надежную работу приложения, но и гибкость в его масштабировании, поддержку асинхронных операций и возможность интеграции с внешними API и сервисами. Особую значимость приобретает сравнение двух мощных стеков веб-разработки – JavaScript (Express.js + React) и C# (ASP.NET Core MVC) (таблица). Оба направления активно используются в веб-разработке, обладают обширной поддержкой сообщества и высокими показателями производительности. Однако каждый стек имеет свои преимущества и недостатки, определяющие их применимость в рамках определенных типов проектов [1–3].

Сравнительная характеристика двух стеков веб-разработки

Критерий	JavaScript (Express.js + React)	C# (ASP.NET Core MVC)
Языковая унификация	Одна платформа для клиента и сервера (JavaScript). Это упрощает разработку и поддержку	Разделение на серверную (C#) и клиентскую (Razor или JavaScript) части
Асинхронность и реальное время	Express.js нативно поддерживает асинхронную обработку запросов и WebSockets, что идеально для real-time приложений	В ASP.NET Core асинхронность реализуется через async (await), но требует дополнительных настроек и инструментов, таких как SignalR для real-time
Скорость разработки	Быстрая разработка благодаря множеству библиотек и фреймворков. С использованием инструментов Express.js и React проект можно разрабатывать значительно быстрее	Разработка может занять больше времени из-за сложности C# и ASP.NET Core, особенно для начинающих
Простота обучения	JavaScript – универсальный язык, широко используемый как для фронтенда, так и для бэкенда. Это делает обучение проще, особенно для новичков	C# требует более глубокой специализации и знаний платформы .NET, что может усложнить обучение
Масштабируемость	Легкость масштабирования с использованием микросервисной архитектуры и облачных технологий. Подходит для стартапов и приложений с высоким трафиком	Масштабируемость высока, особенно для крупных корпоративных решений, но может потребовать более сложной настройки и архитектуры
Гибкость и контроль	Полный контроль над архитектурой и выбором библиотек и инструментов. Легко адаптируется под различные бизнес-решения	Высокий контроль, но фреймворк ASP.NET Core имеет более строгие стандарты, что может затруднить настройку в редких случаях
Скорость работы с фронтендом	React предоставляет высокую скорость рендеринга и обновлений на фронтенде. В сочетании с Express.js проект можно быстро развивать и обновлять	ASP.NET Core MVC работает с Razor на фронтенде, но скорость работы может быть ниже при рендеринге большого числа динамических данных на сервере
Кросс-платформенность	Приложения на Node.js работают на всех платформах: Windows, Linux, macOS	Кросс-платформенность с .NET Core, но могут возникнуть проблемы с совместимостью на старых версиях или в специфических средах
SEO и рендеринг	React поддерживает серверный рендеринг (SSR) с помощью Next.js, что дает хорошую SEO-оптимизацию для SPA	ASP.NET Core MVC предоставляет встроенные возможности через Razor для серверной генерации контента, что тоже может быть полезным для SEO
Примечание – Источник: собственная разработка автора.		

По результатам анализа можно сделать вывод, что разработка на JavaScript (Express.js + React) имеет значительные преимущества перед C# (ASP.NET Core MVC) для реализации системы онлайн-записи. Использование единого языка на клиенте и сервере (JavaScript) упрощает архитектуру и ускоряет разработку. Express.js эффективно обрабатывает асинхронные запросы, что критично для приложений с высокой нагрузкой и обновлениями в реальном времени, а React обеспечивает динамичное обновление интерфейса и возможность создания PWA-приложений.

Список использованной литературы

1. **Разработка** приложений ASP.NET Core // Microsoft learn. – URL: https://learn.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-9.0 (дата обращения: 14.04.2023).

2. **Документация** Node.js // Справочник Node.js. – URL: https://nodejsdev.ru/guides/webdraftt/ (дата обращения: 14.04.2023).

3. **React** JavaScript-библиотека для создания пользовательских интерфейсов // React. – URL: https://ru.legacy.reactjs.org/ (дата обращения: 14.04.2023).