Научная статья
УДК 004.415.23
ББК 32.972
Р 17
DOI: 10.53598/2410-3225-2024-3-346-43-50

Разработка микрофронтенд модулей для цифровых компаний

(*Рецензирована*)

Альфира Менлигуловна Кумратова[1], Ярославна Олеговна Гайворонюк[2], Кирилл Эдуардович Чумаренко[3], Дмитрий Игоревич Шапошников[4]

[1–3] *Кубанский государственный аграрный университет имени И. Т. Трубилина, Краснодар, Россия*
[1] *kumratova.a@edu.kubsau.ru*
[2] *mrsipro2016@yandex.ru*
[3] *chumarenko_kir@mail.ru*
[4] *Института цифровых технологий Северо-Кавказской государственной академии, Черкесск, Россия, pi@ncsa.ru*

Аннотация. *Рассмотрены новые возможности для организации и оптимизации работы современных веб-приложений. Микрофронтенд – это архитектурный подход в разработке веб-приложений, который подразумевает разбиение фронтенда на несколько независимых и небольших модулей или компонентов. Каждый из этих модулей разрабатывается, разворачивается и управляется отдельно, но они могут взаимодействовать между собой в рамках общего приложения.*

Ключевые слова: *микрофронтенд, инструмент Webpack, технология Vite, веб-приложение, микросервисная архитектура, композиция микрофронтендов, бандлер, фреймворк*

Для цитирования: *Разработка микрофронтенд модулей для цифровых компаний / А. М. Кумратова, Я. О. Гайворонюк, К. Э. Чумаренко, Д. И. Шапошников // Вестник Адыгейского государственного университета. Сер. : Естественно-математические и технические науки. 2024. Вып. 3 (346). С. 43–50. DOI: 10.53598/2410-3225-2024-3-346-43-50*

Original Research Paper

Development of microfrontend modules for digital companies

Alfira M. Kumratova[1], Yaroslavna O. Gayvoronyuk[2], Kirill E. Chumarenko[3], Dmitry I. Shaposhnikov[4]

[1–3] *Kuban State Agrarian University named after I. T. Trubilin, Krasnodar, Russia*
[1] *kumratova.a@edu.kubsau.ru*
[2] *mrsipro2016@yandex.ru*
[3] *chumarenko_kir@mail.ru*
[4] *Institute of Digital Technologies of the North Caucasus State Academy, Cherkessk, Russia, Russiapi@ncsa.ru*

Abstract. *New possibilities for organization and optimization of modern web applications are considered. Microfrontend is an architectural approach to web application development that involves breaking the frontend into several independent and small modules or components. Each of these modules is developed, deployed and managed separately, but they can interact with each other within the overall application.*

Keywords: *microfrontend, Webpack tool, Vite technology, web application, microservice architecture, microfrontend composition, bandler, frameworks*

For citation: *Development of microfrontend modules for digital companies / A. M. Kumratova, Ya. O. Gayvoronyuk, K. E. Chumarenko, D. I. Shaposhnikov // The Bulletin of the Adyghe State University. Ser. : Natural-Mathematical and Technical Sciences. 2024. Iss. 3 (346). P. 43–50. DOI: 10.53598/2410-3225-2024-3-346-43-50*

Вопросам применения различных инструментов при проектировании клиентской части веб-приложения посвящены работы [1–6]. Микрофронтенд – это архитектурный подход в разработке веб-приложений, который подразумевает разбиение фронтенда на несколько независимых и небольших модулей или компонентов [1, 4, 7]. Каждый из этих модулей разрабатывается, разворачивается и управляется отдельно, но они могут взаимодействовать между собой в рамках общего приложения (рис. 1).

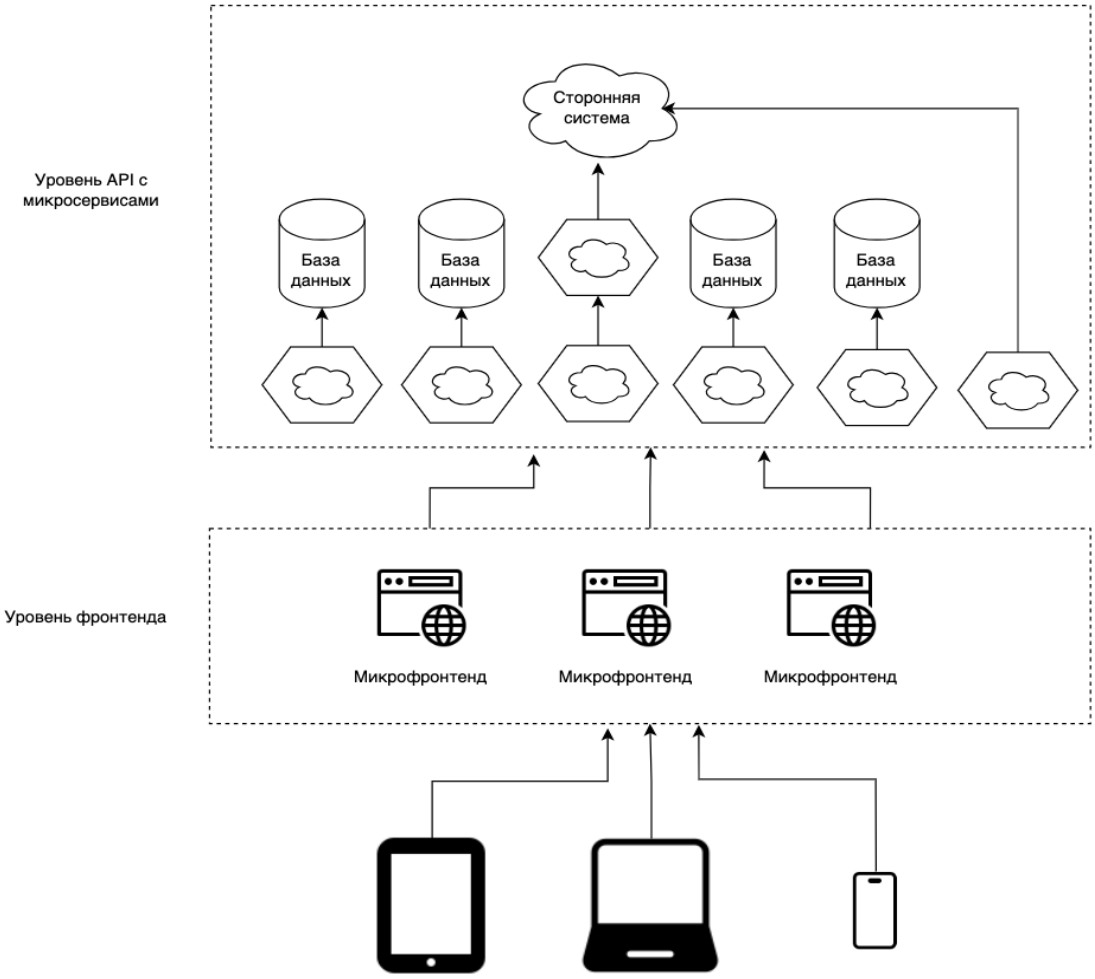

Рис. 1. Микросервисы с микрофронтендами [7]

Fig. 1. Microservices with microfrontends [7]

Перечислим основные возможности микрофронтендов:

– Масштабируемость: Распределение разработки между разными командами позволяет облегчить управление большими проектами.

– Гибкость: Каждая команда может выбирать технологии, подходящие для их модуля, без необходимости унификации всего приложения под одну технологию.

– Ускорение разработки: Разделение на микрофронтенды позволяет отдельным командам работать параллельно и независимо друг от друга, что приводит к сокращению времени на создание всего приложения.

– Упрощение обновлений: Обновления и исправления ошибок могут быть внедрены в один модуль, не затрагивая другие, что снижает риск сбоев.

– Повышение устойчивости: Ошибка в одном модуле не приводит к краху всего приложения, что повышает его надежность.

Вопрос масштабирования стал более актуальным не так давно, так как раньше использовали «толстый» сервер, в котором была заложена вся бизнес-логика, и «тонкий» клиент, который отображал результаты вычислений. Однако мир технологий

стремительно развивается, и ожидания пользователей также растут. Необходимо предоставлять больше интерактива и удобства при использовании веб-платформ. Для того чтобы можно было менять технологии, необходимо иметь независимые части.

В свою очередь монолитный подход на фронтенде означает, что вся фронтенд часть приложения, включая весь код и функционал, размещается в одном монолите. Это означает, что вся разработка, тестирование и деплоймент происходят в пределах этого одного приложения.

Существует ряд причин, по которым, появилась необходимость перейти с монолита на микрофронтенды:

1. Улучшение масштабируемости: микросервисная архитектура позволяет разделить фронтенд на отдельные модули, что упрощает масштабирование и управление кодом.

2. Большая гибкость и независимость: каждый микрофронтенд может быть разработан и поддерживаться отдельной командой, что позволяет быстро вносить изменения без влияния на другие части системы.

3. Улучшение производительности: микросервисы могут быть запущены и масштабированы независимо друг от друга, что обеспечивает более высокую производительность и отказоустойчивость.

4. Удобство при разработке и тестировании: разделение фронтенда на микрофронтенды упрощает разработку, тестирование и изменение кода, так как каждый модуль может быть независимо разрабатываться и тестироваться.

Веб-приложения на протяжении многих лет создавались как трехуровневая архитектурная система: интерфейс, серверная часть (бизнес-логика) и уровень данных (доступ к данным и их хранение) [2, 5, 7].

Проекты компании имеют монолитную архитектуру – это один из самых старых и простых подходов к построению программных приложений [3, 6, 7]. Она предполагает размещение всей логики приложения в одном цельном блоке, в отличие от архитектуры микрофронтенда, где каждая функциональность приложения представлена отдельным модулем.

Монолитные приложения были основным подходом к разработке программного обеспечения на протяжении многих лет. Они просты в понимании, развертывании и масштабировании. Однако с появлением микросервисной архитектуры [1, 4, 7], которая позволяет более гибко и эффективно управлять различными частями приложения, монолитная архитектура часто стала считаться устаревшей (рис. 2).

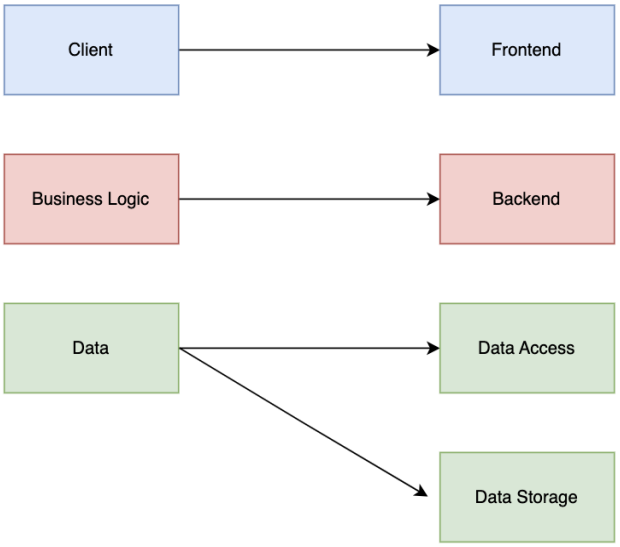

Рис. 2. Схема монолитной архитектуры фронтенда

Fig. 2. Scheme of monolithic frontend architecture

У такой архитектуры имеются определенные недостатки. В частности, с увеличением размера проекта монолитная архитектура может стать неэффективной. Ее сложно масштабировать и поддерживать, так как все компоненты находятся в одном месте и изменение одной части приложения может привести к неожиданным проблемам в других частях.

Большинство проектов используют Webpack в качестве сборщика [7]. Данная технология предоставляет возможность проектирования монолитной архитектуры или микрофронтендов, но все большее применение приобретает сборщик Vite [7]. Основной задачей стоит разработка микрофронтенд модулей на сборщике Vite, так как данная технология стремительно развивается и предоставляет ряд преимуществ.

Необходимо проанализировать, спроектировать, разработать и внедрить систему, которая позволит сократить время разработки, сохранить единообразие систем, а также улучшить сопровождение.

Система должна удовлетворять следующим требованиям:

1. Безопасность: Система должна быть защищена от несанкционированного доступа.
2. Доступность: Система должна быть доступна, когда это необходимо.
3. Техническое обслуживание: Система должна быть проста в обслуживании и обновлении.
4. Надежность: Система должна быть надежной и соответствовать требованиям пользователя.
5. Понятность: Система должна быть простой в использовании и понятной.
6. Совместимость: Система должна быть совместима с другими системами.

В последние годы популярность микросервисов резко возросла, и многие организации используют этот архитектурный стиль, чтобы избежать ограничений, связанных с большими монолитными серверными системами. Когда-то веб-приложения назывались многофункциональными Интернет-приложениями (Rich Internet Applications, RIA), чтобы их можно было отличать от более статичных традиционных корпоративных сайтов [7]. Сейчас же используются самые разные веб-приложения для выполнения повседневных задач. Выбирая архитектуру, разработчики руководствуются конкретными требованиями проекта.

Существует ряд принципов работы микросервисов. Они также применимы к микрофронтендам. Рисунок 3 демонстрирует эти принципы [7].

Рис. 3. Принципы микросервисов

Fig. 3. Principles of microservices

Первый принцип – это «Разделение на домены». Считается хорошей практикой разделять микрофронтенды по правилам предметно-ориентированного проектирования. Такой принцип позволяет достичь хороших результатов, если распределять по командам ответственность за отдельные домены.

Второй принцип – это «Культура автоматизации». Каждый проект по созданию микрофронтендов состоит из десятков и даже сотен компонентов, поэтому необходимы надежные конвейеры для непрерывной интеграции. Для успешной реализации микрофронтендов необходимо уделять должное внимание налаженной автоматизации и развертыванию с быстрой обратной связью.

Сокрытие деталей реализации и соблюдение контрактов являются важными аспектами при разработке микрофронтендов. Команды должны заранее прописать контракт взаимодействия и беспрекословно следовать этим контрактам на протяжении всего жизненного цикла разработки. Такой подход позволит командам менять детали реализации, не мешая другим командам, если эти изменения не касаются контракта API (Application Programming Interface). Это позволяет каждой из команд работать в своем темпе и не зависеть от других.

Благодаря децентрализации процессов, можно подбирать подходы и инструменты под конкретные задачи. Аналогично работе микросервисов – команда работает эффективнее, если она несет полные знания о своем домене. Такая культура обмена знаниями и информацией помогает внедрять успешные практики в разных командах.

Независимое развертывание позволяет командам при работе с микрофронтендами развертывать артефакты в своем темпе. Нет необходимости ждать разрешения внешних зависимостей, а можно самостоятельно принимать решения независимо от особенностей других доменов.

Если сбои происходят в одностраничном приложении, исправить их легче, чего не скажешь про микрофронтенды. В случае сбоев необходимо предоставить пользователю альтернативный контент или скрыть некоторые части приложения. Это практика разделения компонентов таким образом, чтобы сбой одного компонента не распространялся на остальные. Для этого используют различные подходы, такие как контейнеризация, изоляция с помощью фреймворков или библиотек, а также использование механизмов мониторинга и восстановления после сбоев. Изоляция сбоев в микрофронтенде повышает устойчивость системы в целом, а также упрощает отслеживание и устранение проблем в случае возникновения сбоев.

Одной из важных характеристик во фронтенде является «хорошая наблюдаемость». Существуют разные инструменты для ее эффективной реализации, такие как Sentry или LogRocket. Такие инструменты позволяют обнаружить место и причину сбоя.

Микрофронтенды предлагают разные возможности, а выбор подходящего варианта зависит от требований проекта, структуры организации и опыта разработчиков.

Таким образом, микрофронтенды позволяют создавать различные части, независимые друг от друга. Одна часть может быть реализована на React, другая на Vue, а третья на Angular [7].

Существует несколько подходов к композиции микрофронтендов:

1. На стороне клиента.
2. На границе сети.
3. На стороне сервера.

При композиции на стороне клиента оболочка приложения загружает несколько микрофронтендов напрямую из CDN или из источника, если микрофронтенд еще не кеширован в CDN (рис. 4 [7]).

При композиции на стороне клиента у микрофронтенда есть файл JavaScript

или HTML в качестве точки входа, чтобы оболочка приложения могла динамически добавлять узлы в модели DOM (если это HTML) или инициализировать приложение JavaScript.

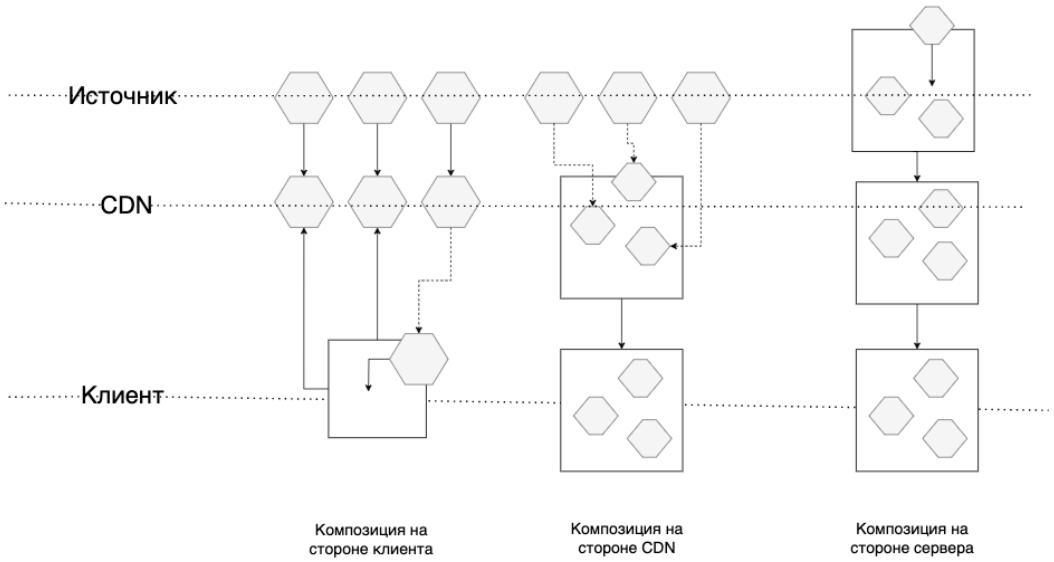

Рис. 4. Схема композиции микрофронтендов

Fig. 4. Scheme of microfrontend composition

Самые популярные сборщики для фронтенда включают в себя Webpack, Browserify, Parcel, Vite и другие. Они позволяют использовать современные возможности разработки веб-приложений, такие как модульность, код-сплиттинг, динамические импорты и прочее.

Когда речь идет о разработке модульных и масштабируемых приложений, как Vite Plugin Federation, так и Webpack Module Federation предлагают эффективные решения. Однако между ними существуют ключевые различия, которые делают Vite Plugin Federation более выгодным выбором в определенных сценариях.

Одним из главных преимуществ Vite Plugin Federation являются простота и скорость разработки. Vite, базовый инструмент для Vite Plugin Federation, известен своим быстрым процессом компоновки и эффективной заменой модулей «горячим» способом. Это означает, что разработчики могут сократить время сборки и упростить общий рабочий процесс разработки по сравнению с Webpack.

Еще одним преимуществом Vite Plugin Federation является его простота в использовании. Vite Plugin Federation легко интегрируется с Vite, что упрощает настройку взаимодействия между модулями. Напротив, объединение модулей Webpack требует более сложного процесса настройки и может оказаться более сложным для понимания разработчиками, особенно теми, кто не знаком с концепциями объединения модулей.

Кроме того, Vite Plugin Federation обеспечивает лучшую совместимость с современными функциями JavaScript, такими как модули ES и динамический импорт. Vite изначально поддерживает эти функции, гарантируя, что разработчики смогут легко воспользоваться ими без какой-либо дополнительной настройки. С другой стороны, объединение модулей Webpack может потребовать дополнительной работы для включения этих функций и обеспечения совместимости.

В целом Vite Plugin Federation обеспечивает более эффективный, удобный и современный подход к объединению модулей по сравнению с Webpack Module Federation. Выбирая Vite Plugin Federation, разработчики могут сократить время сборки, упростить настройку и улучшить совместимость с современными функциями JavaScript.

Далее представлен разрабатываемый прототип, который предназначен для уско-

рения разработки программного обеспечения (ПО), разделения сложных систем на более мелкие, повышения эффективности, надежности и удобства поддержки ПО за счет организации кода в модули.

Для запуска dev режима необходимо выполнить ряд команд:

- собрать билд микрофронтендов – npm run build;
- запустить сервер микрофронтендов – npm run server;
- запустить проект в dev режиме – npm run dev.

После запуска проекта пользователь увидит импортированный layout – контейнер, шапку и набор функций, предоставляемых данным модулем. На вкладке “Workspace кнопки” также продемонстрированы компоненты, импортированные из модуля (рис. 5).

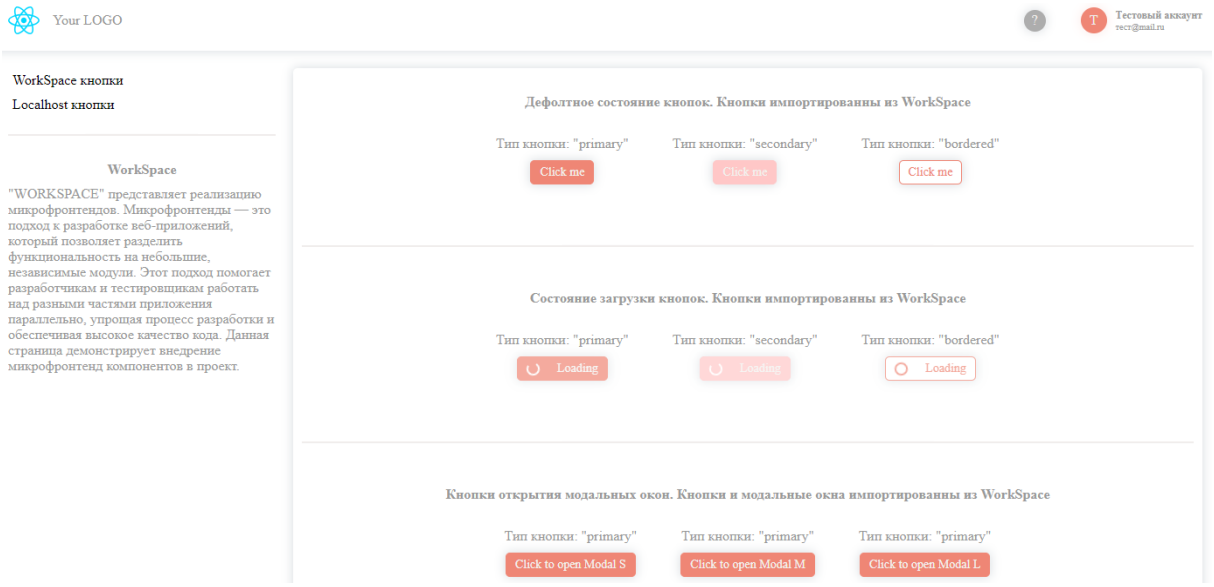

Рис. 5. Компонент “WorkspaceContainer”

Fig. 5. Component “WorkspaceContainer”

Использование микрофронтендов для организации веб-приложений является актуальным и перспективным подходом, который позволяет улучшить работу команды разработчиков, обеспечить независимость модулей, увеличить гибкость и масштабируемость приложения. Разделение фронтенда на небольшие модули позволяет эффективно управлять кодом, быстрее разрабатывать и внедрять новые функциональности, а также обеспечить быструю реакцию на изменения и требования рынка [8]. Использование микрофронтендов открывает новые возможности для организации и оптимизации работы современных веб-приложений и, вероятно, станет стандартным подходом в разработке в будущем.

Примечания

1. Вершинин Е. В., Исаев А. Б. угли, Поляков Р. А. Техники имплементации связей между частями информационной системы с микрофронтенд архитектурой // Электронный журнал: наука, техника и образование. 2022. № 3 (38). С. 15–19. – EDN PDNXGN.

2. Sultan A. S. Efficiency of micro frontend architecture: performance and convenience // Вопросы устойчивого развития общества. 2022. № 5. С. 902–908.

3. Ким П. Е., Кашапов К. Г., Голикова Е. А. Сравнение инструментов для реализации микрофронтендов (микросервисов) с интеграцией во время сборки SPA веб-приложений // Научно-технические инновации и веб-технологии. 2023. № 2. С. 33–38.

4. Шогенов А. М. Исследование архитектуры микрофронтенда: инструменты и рекомендации для использования // Актуальные исследования. 2023. № 39-1 (169). С. 24–28.

5. Прокофьев А П. Особенности архитектуры микрофронтендов // Инновации. Наука.

Образование. 2022. № 50. С. 2126–2132.

6. Обзор применения подхода микросервисной архитектуры при проектировании клиентской части веб-приложения / А. С. Черников, И. В. Никитин, Т. Ю. Гриценко, Д. В. Бородаенко // Дневник науки. 2020. № 4 (40). С. 31.

7. Меццалира Л. Создание микрофронтендов : пер. с англ. Астана : Фолиант, 2023. 320 с.

8. Оптимизация бизнес-процессов цифровых компаний путем внедрения инструментальных средств / А. М. Кумратова, Л. А. Чикатуева, И. И. Василенко, А. В. Абдулхаков // Современная экономика: проблемы и решения. 2022. № 11 (155). С. 20–29.

References

1. Vershinin E. V., Isaev A. B. ugli, Polyakov R. A. Techniques of implementation of communications between components of an information system with a micro fronted architecture // Electronic Journal: Science, Technology and Education. 2022. No. 3 (38). P. 15–19. – EDN PDNXGN.

2. Sultan A. S. Efficiency of micro frontend architecture: performance and convenience // Issues of Sustainable Development of Society. 2022. No. 5. P. 902–908.

3. Kim P. E., Kashapov K. G., Golikova E. A. Comparison of tools for the implementation of micro-frontends (microservices) with integration during the assembly of SPA web applications // Scientific and Technical Innovations and Web Technologies. 2023. No. 2. P. 33–38.

4. Shogenov A. M. The study of microfrontend architecture: tools and recommendations for use // Actual Research. 2023. No. 39-1 (169). P. 24–28.

5. Prokofyev A. P. Features of the architecture of microfrontends // Innovations. Science. Education. 2022. No. 50. P. 2126–2132.

6. Review of the application of the microservice architecture approach in the design of the client part of a web application / A. S. Chernikov, I. V. Nikitin, T. Y. Gritsenko, D. V. Borodaenko // The Diary of Science. 2020. No. 4 (40). P. 31.

7. Mezzalira L. Building Micro-frontends: Scaling Teams and Projects Empowering Developers. Sebastopol (USA) : O'Reilly, 2021. 317 p.

8. Optimization of business processes of digital companies through the introduction of tools / A. M. Kuratova, L. A. Chikatueva, I. I. Vasilenko, A. V. Abdulkhakov // Modern Economics: Problems and Solutions. 2022. No. 11 (155). P. 20–29.

Авторы заявляют об отсутствии конфликта интересов.

Статья поступила в редакцию 09.09.2024; *одобрена после рецензирования 14.09.2024*; *принята к публикации 15.09.2024*.

The authors declare no conflicts of interests.

The article was submitted 09.09.2024; *approved after reviewing 14.09.2024*; *accepted for publication 15.09.2024*.