ishonchliligini va tizim samaradorligini sezilarli darajada oshiradi. Shuningdek, ma'lumotlar sifati va xavfsizligi darajalarini arxitekturaga integratsiya qilish BI tizimlarining barqaror ishlashini, ko'p foydalanuvchili muhitda yuklamalarni muvozanatlashtirishni ta'minlaydi.

Kelgusidagi tadqiqotlarda BI tizimlari arxitekturasiga sun'iy intellekt (AI) asosli tahlil komponentalarini kiritish, real vaqt rejimidagi ma'lumotlar oqimini boshqarish hamda gibrid (on-premise + cloud) modellarni sintez qilish orqali arxitektura samaradorligini yanada oshirish imkoniyatlarini o'rganish maqsadga muvofiqdir.

Adabiyotlar ro'yxati

1. Chavva, Subbareddy & Sangam, Ravi & Rao, B.. (2019). A Survey on Business Intelligence Tools for Marketing, Financial, and Transportation Services: Proceedings of the Second International Conference on SCI 2018, Volume 2. 10.1007/978-981-13-1927-3_53

2.https://community.sap.com/t5/technology-blog-posts-by-members/bi-strategy-part-5-conceptual-architecture-and-system-landscape/ba-p/13088893

3. Monk, Ellen & Wagner, Bret. (2008). Concepts in Enterprise Resource Planning. https://www.researchgate.net/profile/Bret-Wagner/publication/235720403_Concepts_in_Enterprise_Resource_Planning/links/646fab1c69e9c9026db8edfa/Concepts-in-Enterprise-Resource-Planning.pdf, (page 159-183)

4. SAP SE. *SAP BusinessObjects Data Services – Administrator Guide*. (2024-03-25). https://help.sap.com/doc/ac12282da3e24906b753aa8dff052cf0/4.3.3/en-US/ds_43_admin_en.pdf

5. Samuel, Adebis. (2024). Mastering Enterprise Data Management with SAP NetWeaver MDM. https://www.researchgate.net/publication/386574161_Mastering_Enterprise_Data_Management_with_SAP_NetWeaver_MDM. (Page 2-5)

6. https://www.sap.com/products/data-cloud/hana/what-is-sap-hana.html

7. Nadine Côrte-Real, Pedro Ruivo, Tiago Oliveira and Aleš Popovič, Unlocking the drivers of big data analytics value in firms, Journal of Business Research, 10.1016/j.jbusres.2018.12.072, 97, (160-173)

8.Stacia Misner. *Microsoft SQL Server 2012 Reporting Services*. https://ptgmedia.pearsoncmg.com/images/9780735658202/samplepages/9780735658202.pdf, (page 83,91)

9. Färber, Franz & May, Norman & Lehner, Wolfgang & Große, Philipp & Müller, Ingo & Rauhe, Hannes & Dees, Jonathan. (2012). The SAP HANA database - An architecture overview. IEEE Data Eng. Bull. 35. 28-33

10. QlikView Developer Course Manual. – QlikTech International AB, o'quv qo'llanma. – n.d.– PDFhujjat.file:///C:/Users/Alex/Downloads/QlikView%20Developer%20Course%20Manual.pdf.

РАЗРАБОТКА ДИСКУССИОННОЙ ВЕБ - ПЛАТФОРМЫ С ИСПОЛЬЗОВАНИЕМ СТЕКА MERN, TYPESCRIPT И FIREBASE STORAGE

Гилемзянов Алмаз Фирдинантович
старший преподаватель, Казанский (Приволжский) федеральный университет
almazgilemzyanov@yandex.ru
Татьяна Юрьевна Горская
кандидат технических наук, доцент, Казанский государственный архитектурно-строительный университет
gorskaya0304@mail.com

Аннотация: В данной статье рассматривается процесс проектирования и реализации полнофункциональной дискуссионной веб-платформы. Платформа предназначена для публикации статей, создаваемых с помощью встроенного конструктора, и ведения тематических обсуждений в форме тредов. В качестве основного технологического стека был выбран MERN, который включает MongoDB, Express.js, React и Node.js. Для повышения надежности клиентской части и типизации кода был задействован язык TypeScript. Управление статическими медиафайлами, такими как изображения и видео, было организовано через облачное хранилище Firebase Storage. В работе последовательно описаны все этапы разработки: от проектирования пользовательского интерфейса в онлайн-сервисе Figma до реализации серверной логики на Node.js и клиентских компонентов на React. Детально освещены ключевые архитектурные решения, среди которых система аутентификации на основе JWT-токенов, модули создания и отображения контента, а

также применение концепции повторно используемых компонентов для оптимизации рендеринга. Отдельное внимание уделено serverless-подходу при работе с файлами, когда их загрузка и хранение происходят напрямую из клиентской части в облачное хранилище, минуя серверное API, что снижает общую нагрузку на систему.

Ключевые слова: MERN-стек, TypeScript, React, Node.js, Firebase Storage, одностраничное приложение (SPA), JWT, облачные хранилища, веб-разработка, дискуссионная платформа.

DEVELOPING A WEB-BASED DISCUSSION PLATFORM USING THE MERN STACK, TYPESCRIPT, AND FIREBASE STORAGE

Gilemzyanov Almaz Firdinantovich
Senior Lecturer, Kazan (Volga Region) Federal University
almazgilemzyanov@yandex.ru
Tatyana Yuryevna Gorskaya
Candidate of Technical Sciences, Associate Professor,
Kazan State University of Architecture and Engineering
gorskaya0304@mail.com

Annotation: This article examines the design and implementation of a full-featured web discussion platform. The platform is designed for publishing articles created using a built-in constructor and for conducting topic-based discussions in threads. MERN, which includes MongoDB, Express.js, React, and Node.js, was chosen as the primary technology stack. TypeScript was used to enhance client-side reliability and code typing. Static media files, such as images and videos, were managed using Firebase Storage. The paper describes all stages of development step by step: from designing the user interface in the Figma online service to implementing server-side logic in Node.js and client-side components in React. Key architectural decisions are covered in detail, including an authentication system based on JWT tokens, content creation and display modules, and the use of reusable components to optimize rendering. Particular attention is paid to the serverless approach to working with files, where their upload and storage occurs directly from the client to cloud storage, bypassing the server API, which reduces the overall load on the system.

Keywords: MERN stack, TypeScript, React, Node.js, Firebase Storage, single-page application (SPA), JWT, cloud storage, web development, discussion platform.

Введение. Актуальность разработки подобных веб-платформ обусловлена доминированием онлайн-сервисов в современной цифровой среде. Поскольку доступ к сети Интернет стал повсеместным, развертывание приложений непосредственно в вебе является наиболее удобным и логичным решением. Сфера веб-разработки постоянно эволюционирует, предлагая множество библиотек и фреймворков, которые кардинально упрощают и ускоряют процесс создания сложных приложений. Использование готовых технологических стеков, таких как MERN, позволяет разработчикам не тратить время на поиск совместимых инструментов, а сразу приступать к решению конкретных задач, обеспечивая высокую производительность, масштабируемость и поддерживаемость кода [1].

Объектом данного исследования выступает дискуссионная платформа, функционал которой включает создание, публикацию и обсуждение статей и тредов. Предметом исследования является практическое применение стека MERN в сочетании с TypeScript и Firebase Storage для построения полноценного веб-приложения. Целью работы была разработка веб-приложения, предоставляющего редакторам — верифицированным пользователям с расширенными правами — мощный инструмент для создания структурированных статей через встроенный конструктор. Для обычных пользователей платформа должна предоставлять возможность формировать треды и участвовать в их обсуждении путем комментирования.

Обзор используемых технологий и инструментов. Для достижения поставленной цели был сформирован следующий набор технологий и инструментов. В качестве основного инструмента для проектирования пользовательского интерфейса и работы с векторной графикой был использован онлайн-сервис Figma. Его применение наиболее целесообразно на начальных, концептуальных этапах проектирования веб-приложения, что позволяет сформировать целостное визуальное представление о будущем интерфейсе до перехода к непосредственной реализации. Данный подход является методически более предпочтительным по сравнению с

импровизированным проектированием в процессе верстки, поскольку способствует созданию продуманного и структурно согласованного дизайна [3].

Интерфейс самого сервиса отличается высокой степенью интуитивной понятности и эргономичности, что значительно снижает порог вхождения и позволяет эффективно использовать его специалистам с различным уровнем подготовки. Ключевым преимуществом Figma является ее облачная природа, обеспечивающая постоянную доступность проектов из любой точки, возможность коллективной работы в режиме реального времени, а также простоту внесения и контроля изменений. С методической точки зрения, наиболее продуктивными инструментами сервиса для разработки макетов веб-сайтов выступают система сеток и функционал фреймов. Сетка выполняет роль структурного каркаса, задающего точные пространственные соотношения и позволяющего выдерживать строгие пропорции и расстояния между элементами интерфейса. Фреймы, в свою очередь, предоставляют возможность логического группирования компонентов, что облегчает их манипуляцию, тиражирование и организацию сложных композиций. [9]

Важным с практической точки зрения представляется функционал экспорта макета, который позволяет просматривать и анализировать готовый дизайн в форматах, близких к веб-разработке (HTML, CSS). Эта особенность сервиса создает эффективный мост между этапами дизайна и верстки, существенно ускоряя и упрощая работу фронтенд-разработчиков, которые получают четкие визуальные и структурные ориентиры для последующей технической реализации.

Клиентская часть приложения была реализована с применением библиотеки React, которая представляет собой современный инструмент для построения пользовательских интерфейсов. Фундаментальным преимуществом данной библиотеки является концепция Virtual DOM, обеспечивающая оптимизацию процесса обновления интерфейса за счет минимизации прямых манипуляций с реальной объектной моделью документа. Это позволяет достичь высокой производительности и отзывчивости приложения даже при интенсивном динамическом изменении данных. Для разработки компонентов был выбран язык TypeScript, выступающий статически типизированным надмножеством JavaScript. Внедрение строгой типизации позволило существенно повысить надежность кодовой базы за счет выявления потенциальных ошибок, связанных с несоответствием типов данных, на этапе компиляции. Дополнительными преимуществами использования TypeScript стали улучшенные возможности автодополнения в интегрированных средах разработки, а также облегчение процесса рефакторинга кода благодаря явному описанию структур данных и контрактов между компонентами.

Управление глобальным состоянием приложения, включая такие критически важные данные как информация об аутентифицированном пользователе, было организовано с помощью библиотеки React-Redux. Данное решение обеспечивает централизованное и предсказуемое управление состоянием, что способствует соблюдению принципов однонаправленного потока данных и упрощает отладку сложных сценариев взаимодействия пользователя с интерфейсом [12].

В области стилизации компонентов был задействован препроцессор SASS, расширяющий возможности стандартного CSS. Его применение позволило реализовать более структурированный и модульный подход к написанию стилей, используя такие возможности как вложенность селекторов, переменные и миксины. Это не только повысило читаемость и поддерживаемость стилей, но и способствовало соблюдению принципа повторного использования кода и обеспечению визуальной согласованности интерфейса. [11] Серверная часть была построена на платформе Node.js, которая обеспечивает асинхронную и событийно-ориентированную обработку запросов. Фреймворк Express.js был использован для упрощения создания API-маршрутов и обработки HTTP-запросов. В качестве основной базы данных была выбрана документо-ориентированная MongoDB. Для взаимодействия с ней применялась библиотека Mongoose, которая предоставляет удобный способ описания схем данных и моделей, а также встроенные механизмы валидации. Сама база данных была развернута в облачном сервисе MongoDB Atlas. Для хранения медиафайлов (изображений, видео, анимаций) было интегрировано облачное хранилище Firebase Storage. Его ключевой особенностью в данном проекте стала реализация serverless-подхода: клиентское приложение может напрямую взаимодействовать с хранилищем, загружая и скачивая файлы без необходимости проксирования запросов через собственный сервер, что существенно разгружает backend.

Проектирование архитектуры и моделей данных. На основе анализа требований к функционалу были спроектированы и реализованы основные модели данных с использованием Mongoose. Модель пользователя (User) включает такие поля, как уникальный логин (username),

электронная почта (email), хэш пароля (password), имя и фамилия, а также служебные поля для восстановления пароля. Модель треда (Thread) содержит идентификатор автора, заголовок, текстовое описание, ссылку на прикрепленный медиафайл в Firebase, массив комментариев и систему рейтингов. Модель статьи (Article) схожа с моделью треда, но вместо системы рейтингов она содержит ключевое поле — массив блоков (blocks). Каждый блок (Block) представляет собой самостоятельную единицу контента и описывается своим типом (текст, заголовок, изображение, видео), содержимым (текст или ссылка на файл) и визуальными параметрами, например, размером. Модель комментария (Comment) является универсальной и содержит ссылку на целевой объект (тред или статью) и текст комментария.

Реализация пользовательского интерфейса. Клиентский интерфейс приложения был реализован в парадигме одностраничного приложения (Single Page Application) с использованием библиотеки React. Архитектурной основой разработки пользовательского интерфейса выступила концепция повторно используемых компонентов (Reusable Components), что позволило создать систему унифицированных интерфейсных модулей. В рамках данного подхода были разработаны типовые компоненты, такие как Thread.tsx для визуализации элементов дискуссионных тредов и ArticleShow.tsx для отображения контента статей, которые демонстрируют свойство полиморфизма - способность функционировать в различных контекстах приложения с разнородными наборами данных.

Внедрение компонентного подхода позволило достичь значительного снижения избыточности кодовой базы, минимизировав дублирование логики и разметки. Это, в свою очередь, упростило процессы поддержки и модификации системы, а также оптимизировало механизм рендеринга за счет стандартизации процедур обновления виртуального DOM. С точки зрения производительности, компонентная архитектура обеспечила эффективное переиспользование уже созданных экземпляров компонентов, сократив нагрузку на вычислительные ресурсы. В аспекте визуального проектирования была применена технология SASS как инструмент стилизации. Использование данного препроцессора обеспечило соблюдение принципов визуального единства на всех уровнях интерфейса, а также создало формализованную систему управления дизайн-параметрами. Особую значимость этот подход продемонстрировал в управлении цветовой схемой приложения, где доминирующий бирюзово-зеленый оттенок был инкапсулирован в переменные, что гарантировало его консистентное применение через все компоненты и состояний интерфейса. Данная методология позволила установить строгую корреляцию между дизайн-системой и ее технической реализацией, обеспечивая масштабируемость визуальных решений (рис.1).

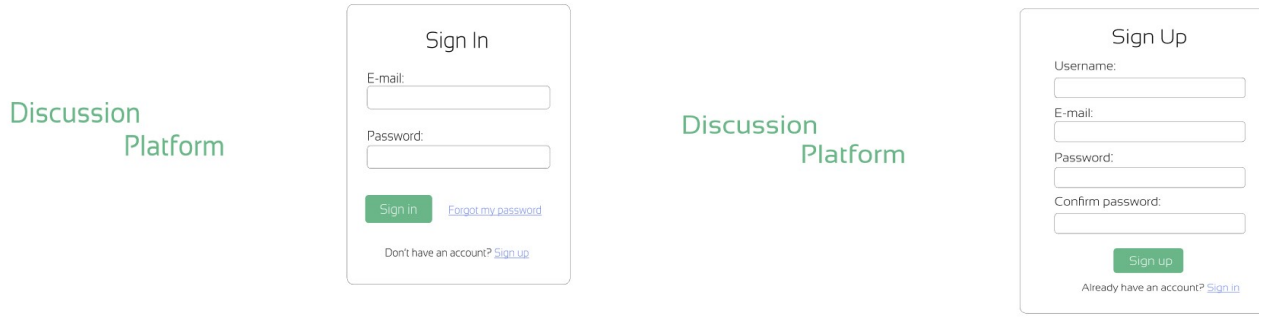

Рисунок 1. Форма авторизации/регистрации/восстановления пароля

Система аутентификации и авторизации. Безопасный доступ к ресурсам платформы был организован с помощью технологии JSON Web Tokens (JWT). Процесс начинается с того, что пользователь отправляет свои учетные данные (email и пароль) на сервер по маршруту /api/auth/. Сервер проверяет их, и в случае успеха генерирует JWT-токен. Этот токен содержит в зашифрованном виде идентификатор пользователя и его имя, а также имеет ограниченный срок жизни, заданный в настройках. Сгенерированный токен отправляется клиенту и сохраняется на его стороне. При каждом последующем запросе к защищенному ресурсу этот токен передается в заголовке Authorization. Сервер проверяет его валидность перед тем, как выполнить запрос. На клиенте для управления состоянием аутентификации (например, для отображения имени пользователя в шапке сайта) использовалась связка React-Redux. Асинхронные операции, такие как запрос на авторизацию, были реализованы с помощью middleware Redux Thunk (рис.2) [12].

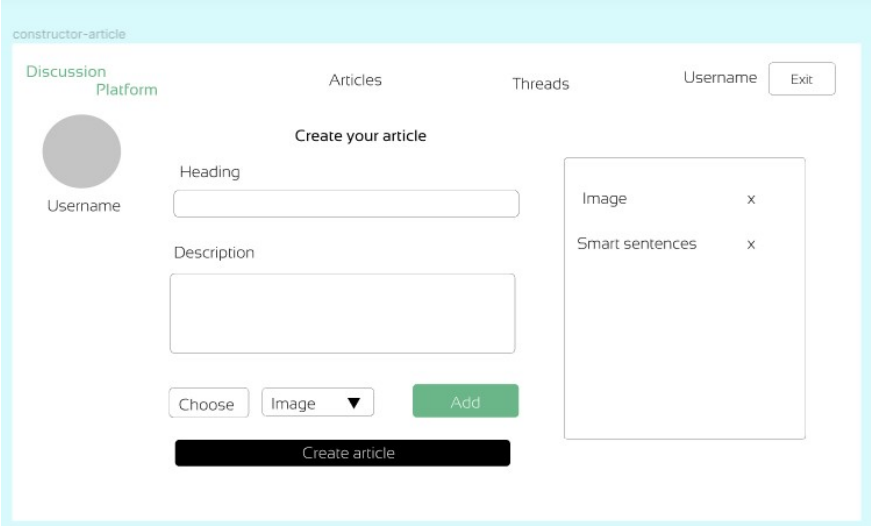

Рисунок 2. Конструктор статей

Функциональные модули платформы. Модуль тредов обеспечивает возможность создания и просмотра обсуждений. Процесс создания треда, реализованный в компоненте ThreadCreate.tsx, является двухэтапным. Сначала пользователь заполняет форму, вводя заголовок, описание и прикрепляя медиафайл. После предварительного просмотра, при подтверждении, данные треда отправляются на сервер для сохранения в MongoDB, а медиафайл напрямую загружается в Firebase Storage. Для отображения списка тредов используется компонент Threads.tsx, который получает данные с сервера и отображает их в виде карточек-превью. Компонент ThreadShow.tsx отвечает за отображение полной страницы треда со всеми комментариями (рис.3).

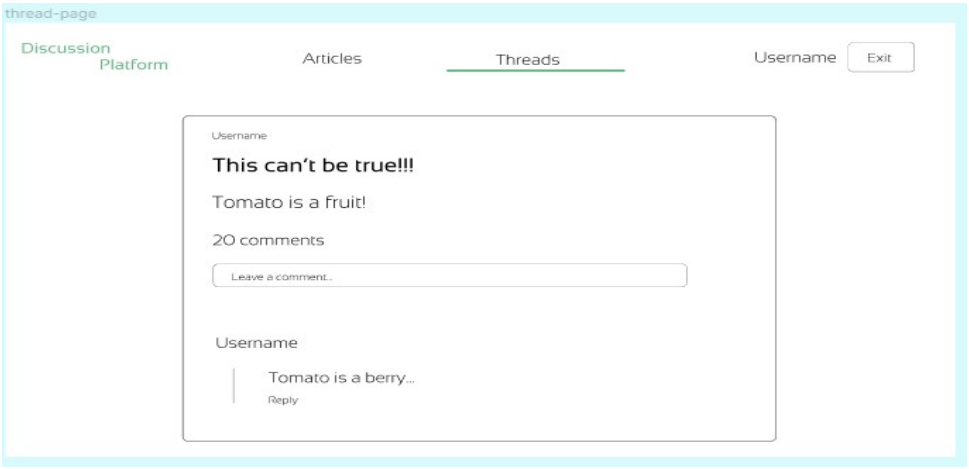

Рисунок 3. Главная страница треда

Модуль статей, или конструктор, представляет собой более сложный инструмент. Он позволяет редакторам формировать статьи из последовательности блоков. Интерфейс конструктора предоставляет возможность выбора типа следующего блока (текст, заголовок, медиа) и его добавления в статью. Текстовое содержимое накапливается в локальном состоянии компонента до момента окончательного сохранения всей статьи, что минимизирует количество запросов к серверу на этапе редактирования. Медиафайлы, как и в случае с тредами, сразу загружаются в Firebase Storage. Пользователь может просматривать и редактировать последовательность блоков, а также удалять их. После нажатия кнопки создания статьи, метаданные (заголовок, описание, массив блоков) сохраняются в базе данных через API (рис.4). Отображение готовых статей происходит в компоненте ArticleShow.tsx, который динамически рендерит последовательность блоков в соответствии с их типами(рис.5) [13].

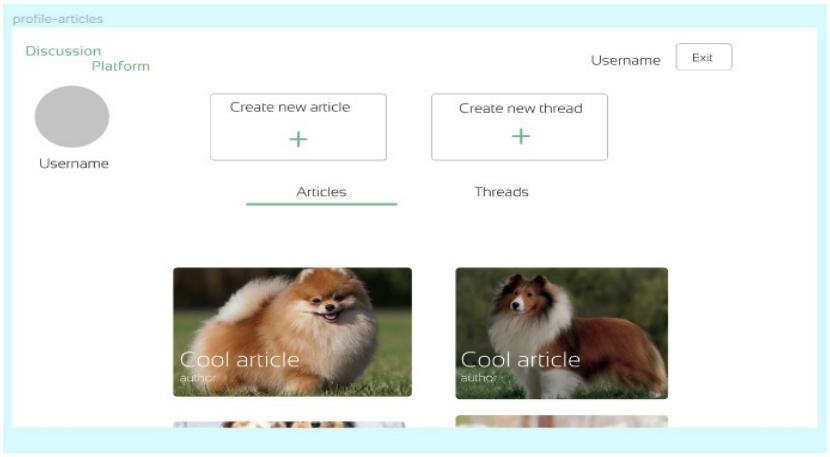

Рисунок 4. Страница профиля опубликованные статьи

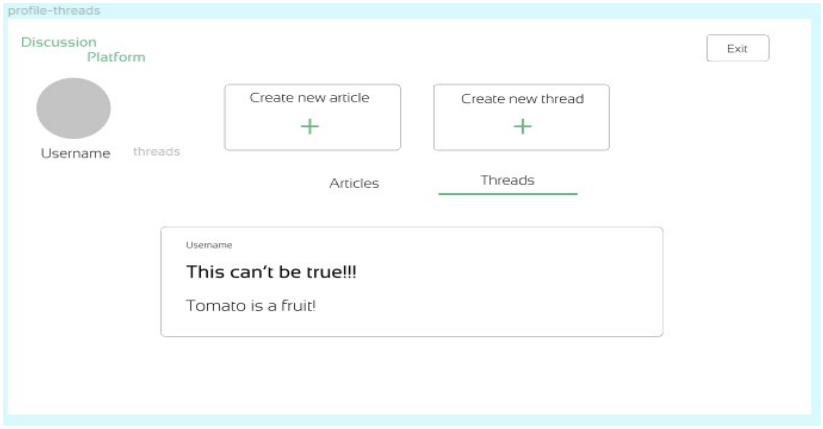

Рисунок 5. Страница профиля опубликованные треды

Заключение. В результате проведенной работы была успешно разработана и внедрена дискуссионная веб-платформа, полностью удовлетворяющая первоначально поставленным требованиям. В ходе проекта был освоен и применен на практике современный технологический стек, включающий MERN, TypeScript и Firebase Storage. Реализованы как серверная часть, обеспечивающая надежное API и взаимодействие с базой данных, так и клиентская часть, предлагающая пользователям интуитивно понятный и отзывчивый интерфейс.

Внедрение системы аутентификации, основанной на использовании JSON Web Tokens (JWT), позволило организовать безопасный механизм контроля доступа к защищенным ресурсам платформы. Данный подход обеспечивает надежную передачу аутентификационных данных между клиентом и сервером, минимизируя риски несанкционированного доступа к персональной информации пользователей и функционалу, предназначенному для редакторского состава.

Интеграция облачного хранилища Firebase Storage в соответствии с serverless-архитектурой продемонстрировала высокую эффективность в управлении статическими медиафайлами. Данное решение позволило делегировать задачи, связанные с обработкой и хранением файлов, специализированной облачной инфраструктуре, что существенно снизило нагрузку на серверную часть приложения и оптимизировало распределение вычислительных ресурсов.

Применение языка TypeScript для разработки клиентской части приложения внесло значительный вклад в повышение надежности и устойчивости кода к ошибкам. Система статической типизации обеспечила выявление противоречий в типах данных на этапе компиляции, что сократило количество потенциальных инцидентов во время выполнения программы. Параллельно с этим, реализация компонентного подхода в рамках библиотеки React способствовала созданию хорошо структурированной и модульной архитектуры, характеристики которой напрямую влияют на сопровождаемость и масштабируемость проекта в долгосрочной перспективе.

Проведенная работа служит практическим подтверждением производительности и адаптивности современной экосистемы JavaScript для создания сложных веб-приложений корпоративного уровня. Использованный технологический стек, включающий MERN, TypeScript

и облачные сервисы, продемонстрировал свою состоятельность для решения задач данного класса, обеспечив создание программного продукта, отвечающего ключевым требованиям современной веб-разработки: производительности, безопасности, надежности и удобству эксплуатации.

Список литературы

1. *Кириллов П. А*. Искусственный интеллект для образования. Адаптивная система обучения // Молодой ученый. 2020. №. 27. С. 39-43.

2. *Шитов С. Б.* Цифровые адаптивные системы обучения в условиях цифровизации экономики (социально-философский взгляд) //Alma mater (Вестник высшей школы). 2020. №. 8. С. 8-11.

3. *Сысоев П. В.* Технологии искусственного интеллекта в обучении иностранному языку //Иностранные языки в школе. 2023. №. 3. С. 6-16.

4. *Маткасимова Ш. Ш. и др*. Сравнительный анализ системы обучения // Экономика и социум. 2023. №. 5-1 (108). С. 602-608.

5. *Платов А. В., Гаврилина Ю. И*. Искусственный интеллект в образовании: эволюция и барьеры //Научный результат. Педагогика и психология образования. 2024. Т. 10. №. 1. С. 26-43.

6. *Золотокопова С. В. и др.* Высокотехнологичные приемы переработки рыбного сырья, выращенного в Астраханской области // Вестник Астраханского государственного технического университета. Серия: Рыбное хозяйство. 2021. №. 4. С. 134-144.

7. Малкова Т.В., Телякова И.Х., Хисматулина Н.В., Пугачева С.А. Изменение роли преподавателя и требований, предъявляемых к нему, в условиях модернизации системы образования // Вопросы педагогики. 2022. №5–2. С. 217–220.

8. *Горская Т.Ю., Замега Е.Н.* Перспектива дистанционного обучения: за и против // Человеческий капитал. 2022. 8(164). С. 92-99.

9. *Голованова И.И. и др*. Цифровая образовательная среда и онлайн-обучение глазами студентов: Плюсы и минусы // Образование и саморазвитие. 2022. Том 17. Из.3, С. 202-221.

10. *Кугуракова, В.В., Голованова, И.И., Шайдуллина, А.Р.* Цифровые решения в обучении педагогов: Концепция внедрения тренажера виртуальной реальности // Евразийский журнал математического, естественно-научного и технологического образования. 2021. 17(9). С. 1-10.

11. *Гафаров, Ф. М*. Нейронные сети в PyTorch: учебное пособие / Ф. М. Гафаров, А. Ф. Гилемзянов. - Казань: Казанский федеральный университет, 2024. 106 с.

12. Лукичев П. М., Чекмарев О. П. Риски применения искусственного интеллекта в системе высшего образования //Вопросы инновационной экономики. – 2024. – Т. 14. – №. 2. – С. 463.

13. Леон, У. Разработка веб-приложения GraphQL с React, Node.js и Neo4j / У. Леон ; перевод с английского А. Н. Киселева. Москва : ДМК Пресс, 2023. — 262 с. — ISBN 978-5-93700-185-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/314975 (дата обращения: 05.11.2025). — Режим доступа: для авториз. пользователей.

14. Сан, Ф. М. Разработка приложений с Quarkus и React : руководство / Ф. М. Сан ; перевод с английского А. Н. Киселева. Москва : ДМК Пресс, 2023. — 294 с. — ISBN 978-5-93700-207-5. — Текст : электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/455309 (дата обращения: 05.11.2025). — Режим доступа: для авториз. Пользователей.

АНАЛИЗ ИСПОЛЬЗОВАНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ОБРАЗОВАТЕЛЬНОЙ ДЕЯТЕЛЬНОСТИ ВУЗОВ

Татьяна Юрьевна Горская
кандидат технических наук, доцент,
Казанский государственный архитектурно-строительный университет
gorskaya0304@mail.com
Гилемзянов Алмаз Фирдинантович
старший преподаватель, Казанский (Приволжский) федеральный университет
almazgilemzyanov@yandex.ru
Гафаров Фаиль Мубаракович
кандидат физико-математических наук, доцент, КФУ
fgafarov@yandex.ru