

Futurum Digitalis (Цифровое Будущее)

Научная статья

УДК 004.415.2

Разработка мобильного приложения на React Native

Викторов Е. В.^{1, 2}

Научный руководитель: Куликова Е. В.¹, старший преподаватель

¹ Сибирский институт бизнеса и информационных технологий, г. Омск, Россия.

² eviktorovvv@gmail.com

Аннотация: В статье рассматривается процесс разработки мобильного приложения с использованием фреймворка кроссплатформенной разработки React Native, языка TypeScript и среды Expo. Актуальность исследования обусловлена ростом спроса на мобильные сервисы в фармацевтическом ритейле и необходимостью быстрого вывода приложений на платформы iOS и Android при сохранении единой бизнес-логики и пользовательского интерфейса. В работе выполнен анализ преимуществ архитектуры React Native в сравнении с нативными инструментами разработки, обоснован выбор архитектурного подхода Monorepo, обеспечивающего согласованное развитие серверной и клиентской частей приложения, а также применение контейнеризации Docker. Рассмотрены этапы проектирования пользовательского интерфейса, реализация навигации, системы авторизации и управления состоянием приложения. Описаны проектные решения по реализации прототипа мобильного приложения.

Ключевые слова: *приложение; ReactNative; кроссплатформенность; TypeScript; ИТ-бизнес*

Mobile app development with React Native

Viktorov E. V.^{1, 2}

Scientific supervisor: Kulikova E. V.¹, senior lecturer

¹ Siberian Institute of Business and Information Technology, Omsk, Russia.

² eviktorovvv@gmail.com

Abstract: This article examines the development of a mobile application using the React Native cross-platform development framework, TypeScript, and the Expo environment. The relevance of this study stems from the growing demand for mobile services in pharmaceutical retail and the need to quickly deploy applications to iOS

and Android platforms while maintaining a unified business logic and user interface. The paper analyzes the advantages of the React Native architecture compared to native development tools, substantiating the choice of the Monorepo architectural approach, which ensures the coordinated development of the server-side and client-side components of the application, as well as the use of Docker containerization. The article examines the stages of user interface design, navigation implementation, authorization system, and application state management. Design solutions for the implementation of a mobile application prototype are described.

Keywords: *application; ReactNative; cross-platform; TypeScript; IT business*

Введение

В современных условиях мобильный доступ к фармацевтическим услугам становится важным элементом повседневного обслуживания. Аптечные сети сталкиваются с задачей быстрого вывода приложений на платформы iOS и Android при сохранении единой логики работы и целостного пользовательского интерфейса, включая дизайн-концепцию. В данной работе рассматривается процесс создания мобильного приложения для аптечной сети на базе фреймворка React Native с использованием TypeScript и среды Expo.

Цель исследования – продемонстрировать эффективность кроссплатформенного подхода и автоматизации процессов разработки с применением современных инструментов, таких как Docker и технологии искусственного интеллекта.

1. Архитектура и проектирование системы

Проектирование информационной системы для аптечной сети требует особого внимания к отказоустойчивости и консистентности данных. В этой связи особенно важным является обоснование проектных решений по архитектуре.

В ходе работы был выбран подход Monorepo (единый репозиторий). Несмотря на то, что компоненты системы (Backend и Frontend) являются независимыми единицами, их хранение в рамках одного репозитория позволяет синхронизировать изменения в контрактах API и типах данных.

Структура проекта организована следующим образом (рис. 1):

1. Директория /backend: содержит исходный код серверной части на языке C#.
2. Директория /frontend: содержит мобильное приложение на React Native.
3. Директория /docker: включает конфигурационные файлы для сборки и развертывания системы.
4. Скрипты автоматизации (.sh файлы): расположены в корне проекта и предназначены для автоматической сборки Docker-образов и артефактов (APK, IPA, AAB) для публикации в сторсах.

Такое разделение позволяет разработчику запускать backend и frontend отдельно, но при этом иметь общую историю изменений и единый процесс CI/CD (Continuous Integration/Continuous Deployment).

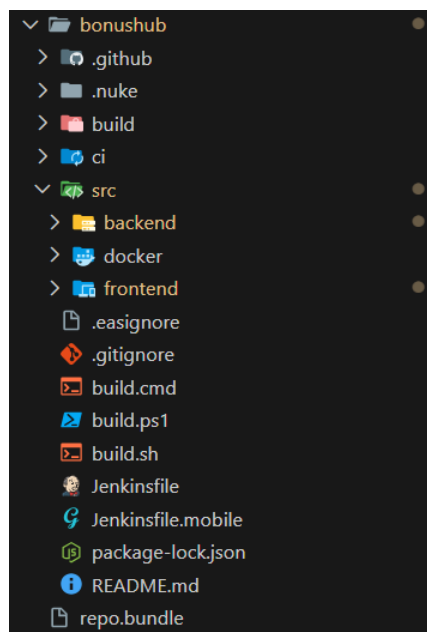


Рисунок 1. Структура проекта

Серверная часть приложения реализована на стеке .NET (C#). Выбор данного стека обусловлен необходимостью высокой скорости обработки запросов и наличием развитых инструментов для работы с базами данных (Entity Framework).

Для обеспечения мобильности и независимости от окружения, backend упакован в Docker-контейнеры. Благодаря использованию контейнеризации в процессе разработки сокращается количество ошибок, поскольку контейнеры содержат все необходимые компоненты для запуска приложения, а перенос программных решений между различными средами выполнения осуществляется без существенных затруднений [2]. Такое построение гарантирует, что приложение будет работать идентично как на локальной машине разработчика, так и на удаленном сервере.

Базовым инструментом взаимодействия между частями системы стала является Swagger. Она выполняет роль связующего звена:

- разработчик backend описывает методы (endpoints), параметры запросов и форматы ответов;
- разработчик frontend использует эту спецификацию для автоматической или ручной настройки сетевых запросов через библиотеку Axios.

Проектирование интерфейса аптечного приложения проходило в три этапа. Выделение этапов и процессов в жизненном цикле разработки позволяет минимизировать ошибки на каждой стадии за счет раннего выявления проблем, последовательной проверки проектных решений и поэтапного уточнения требований к пользовательскому интерфейсу.

Этап 1 «Концепт-макет (Wireframing)». На этой стадии создавались низкодетализированные схемы экранов (рис. 2). Основная задача – определить композицию элементов: расположение поисковой строки для лекарств, кнопок

корзины и карточек товаров. Здесь не учитывались цвета и шрифты, внимание уделялось только эргономике.

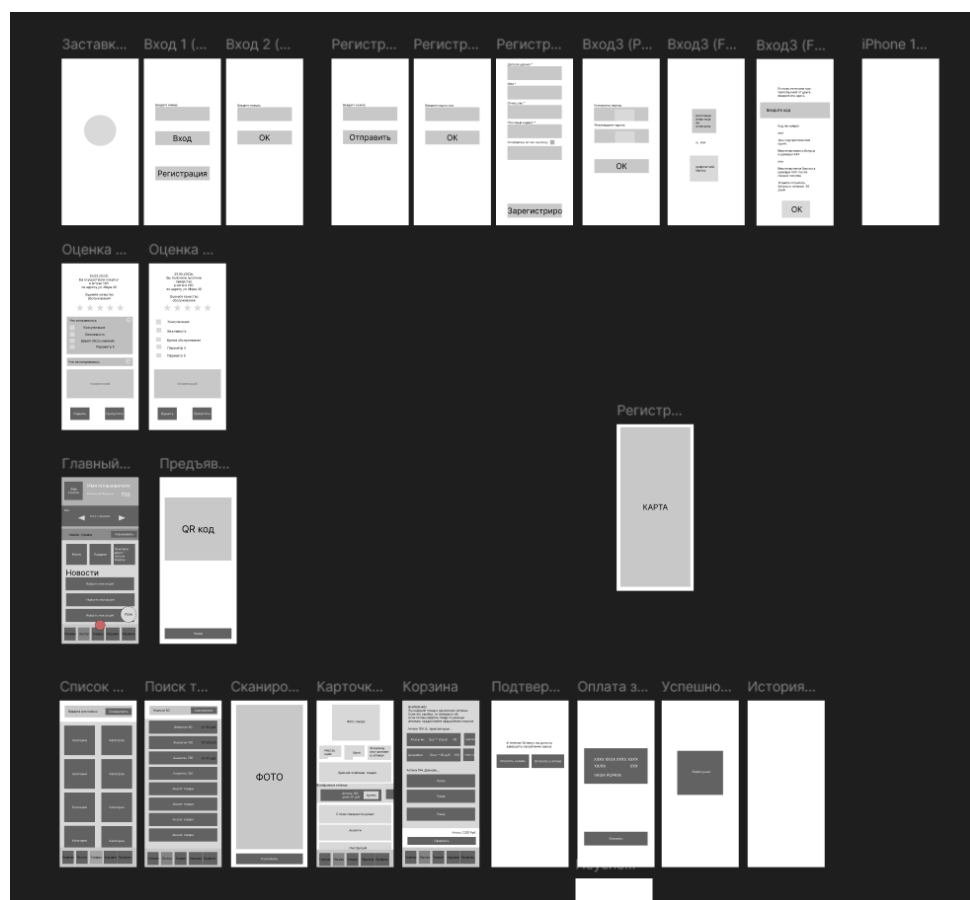


Рисунок 2. Низкодетализированные экраны

Этап 2 «Визуальный дизайн (UI Design)». После утверждения структуры была разработана визуальная оболочка. Для аптечной сети важно доверие, поэтому в дизайне использовались чистые цвета (белый, оттенки синего или зеленого) и четкая типографика.

Этап 3 «Интерактивный прототип (Figma Prototype)» (рис. 3). В среде Figma была настроена логика переходов. Процессы этого этапа необходимы, чтобы «прокликать» приложение до начала разработки, проверить удобство навигации между главной страницей и профилем, а также протестировать многоэтапный процесс регистрации.

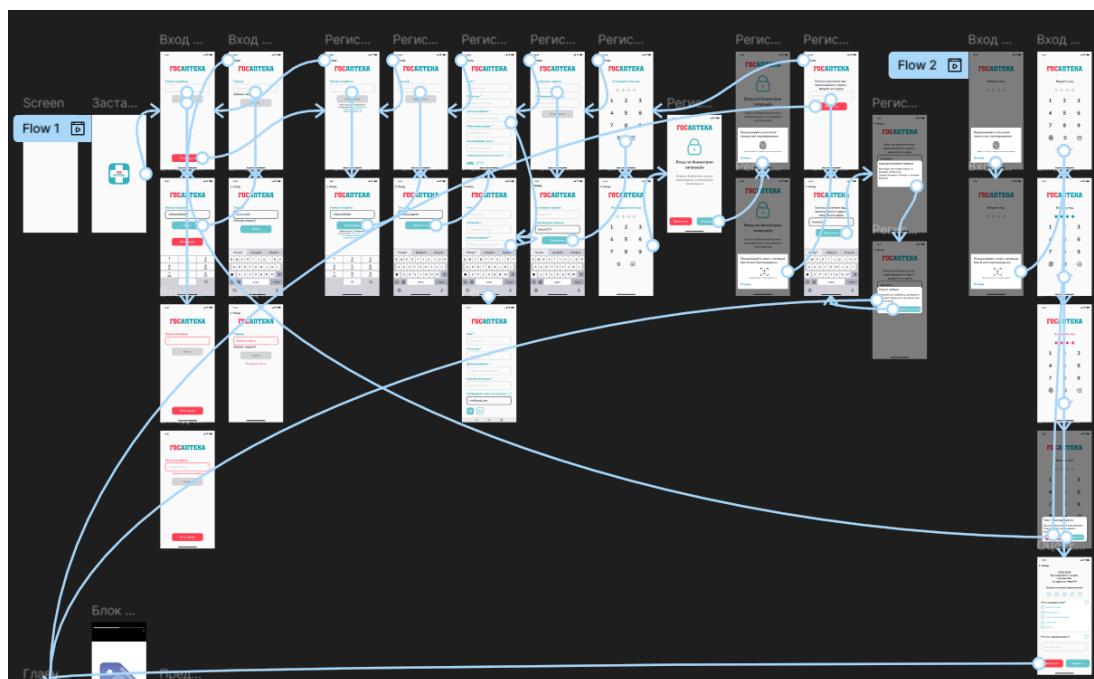


Рисунок 3. Визуальный макет с прототипированием

Для обоснования выбора технологического стека в работе выполнено сравнительное сопоставление фреймворка React Native с альтернативными подходами к разработке мобильных приложений: технологии (Swift/Kotlin) и фреймворк Flutter. Результаты сравнительного анализа представлены в таблице 1.

Таблица 1

Сравнение технологий разработки мобильных приложений

Критерий	React Native (выбранный)	Native (Swift/Kotlin)	Flutter
Скорость разработки	Высокая (единый код)	Низкая (два кода)	Высокая
Стоимость	Оптимальная	Высокая	Оптимальная
Производительность	Близкая к нативной	Максимальная	Высокая
Сложность поддержки	Низкая	Высокая	Средняя

2. Технологический стек и реализация Frontend-части

Разработка клиентской части приложения является наиболее трудоемким этапом, так как именно здесь формируется пользовательский опыт (UX) и реализуется визуальная составляющая (UI) [1]. Для аптечной сети критически важно, чтобы интерфейс был интуитивно понятным, а приложение работало стабильно на различных типах устройств.

Основой разработки стал фреймворк React Native. Его выбор обусловлен возможностью использования единой кодовой базы для компиляции приложений под iOS и Android. Одной из наиболее увлекательных особенностей React Native является его способность переносить современные веб-технологии на мобильные устройства, сохраняя при этом функциональность и производительность на высоком уровне [3]. В отличие от чистого JavaScript, в

проекте использован TypeScript. Применение строгой типизации в медицинском приложении позволяет:

- исключить ошибки при передаче данных о лекарственных препаратах (дозировки, цены, артикулы);
- облегчить поддержку кода при масштабировании сети;
- повысить читаемость кода для других разработчиков в рамках монорепоzitория.

В качестве среды разработки и сборки использовался фреймворк Expo. Он значительно упрощает доступ к нативным функциям устройства (камера для сканирования штрих-кодов, биометрия) и позволяет проводить тестирование через приложение Expo Go без необходимости постоянной пересборки тяжелых пакетов [4].

Приложение выстроено по принципу атомарного дизайна. Весь интерфейс разделен на переиспользуемые компоненты (кнопки, карточки товаров, поля ввода). Такое разделение не только ускоряет разработку, но и обеспечивает визуальное единство всего приложения.

Особое внимание уделено навигации. В проекте реализовано два типа навигационной логики:

1. Таб-навигация (Tab Navigation): Постоянное меню в нижней части экрана с тремя ключевыми разделами: «Главная», «Каталог товаров» и «Личный кабинет». Это обеспечивает быстрый доступ к покупкам из любой точки приложения (рис. 4).

2. Стековая навигация (Stack Navigation): используется в процессах, требующих последовательных действий, таких как регистрация или оформление заказа.

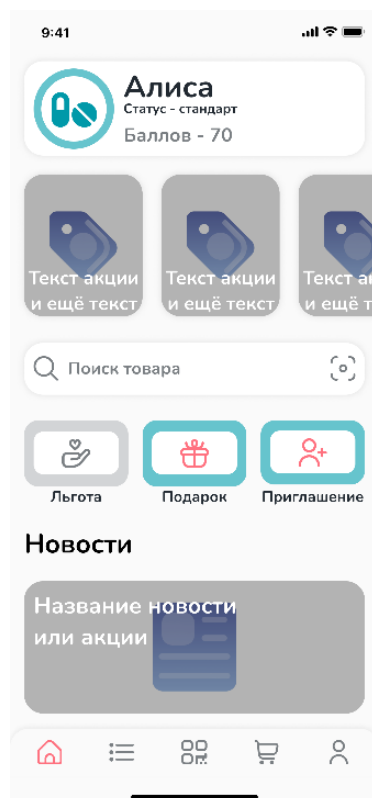


Рисунок 4. Страница главного экрана с системой «табов»

Система входа в приложение реализована как многошаговый сценарий.

1. Первый этап: ввод номера телефона и получение ОТП-кода (одноразового пароля) через СМС.

2. Второй этап (для новых пользователей): сбор профильных данных – ФИО, возраст, пол и электронная почта.

3. Программа лояльности: интегрирована система пригласительных кодов («Приведи друга»), которая проверяется на этапе регистрации через API.

Для защиты персональных данных и упрощения повторного входа интегрированы модули биометрии – FaceID и TouchID. Реализация выполнена с помощью стандартных библиотек Ехро, которые гарантируют корректную работу датчиков на обеих платформах. Структура экранов многоэтапной системы регистрации представлена на рис. 5.

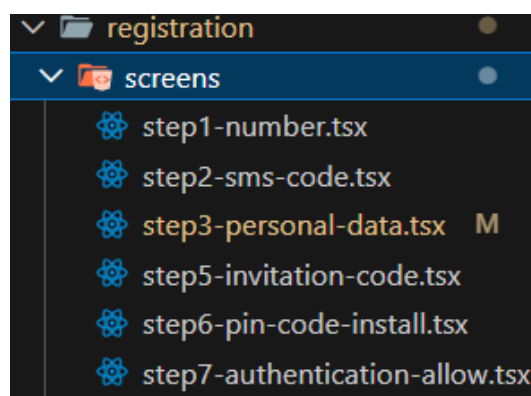


Рисунок 5. Структура экранов системы регистрации

Для обмена данными с backend-частью (C#) используется библиотека Axios. Она позволяет настроить глобальные перехватчики (interceptors) для обработки ошибок авторизации или проблем с интернет-соединением.

Для управления формами (особенно сложной формой регистрации с множеством полей) выбрана библиотека Formik. Она эффективно справляется с:

- отслеживанием значений в полях ввода;
- валидацией данных в реальном времени (например, проверка корректности ввода e-mail);
- синхронизацией состояния кнопок отправки данных.

Поскольку состояние приложения (авторизован ли пользователь, что лежит в корзине) достаточно обширно, была проработана архитектура управления состоянием, учитывающая кэширование данных для обеспечения работы приложения при нестабильном сигнале связи.

3. Применение технологий искусственного интеллекта в процессе разработки

Современная разработка программного обеспечения все чаще интегрирует инструменты искусственного интеллекта (ИИ) для оптимизации рутинных задач и решения сложных архитектурных проблем. В рамках данного проекта использование языковой модели ChatGPT было обосновано необходимостью ускорения цикла разработки и повышения качества кроссплатформенной адаптации приложения.

Одной из наиболее трудоемких задач в мобильном приложении аптечной сети является реализация многоступенчатых форм (регистрация, ввод данных рецепта, оформление заказа). Использование библиотеки Formik требует тщательной настройки схем валидации и управления сложными объектами состояния.

Искусственный интеллект применялся для:

- генерации шаблонов валидации данных на основе регулярных выражений (проверка номера телефона, формата электронной почты);
- проектирования логики обработки ошибок: ИИ помог составить условия, при которых пользователь получает подсказки о некорректно заполненных полях, что значительно улучшило пользовательский опыт (UX).

Несмотря на универсальность React Native, разработчики часто сталкиваются с проблемой «разного поведения» элементов на iOS и Android. В ходе реализации проекта возникли сложности с отображением модальных окон и поведением клавиатуры на устройствах под управлением iOS (перекрытие текстовых полей ввода клавиатурой).

С помощью ChatGPT были найдены и адаптированы решения, специфичные для нативных компонентов операционных систем (рис. 6):

1. Настройка компонента KeyboardAvoidingView с учетом специфических отступов (safe area) на iPhone.
2. Исправление багов визуализации теней и радиусов скругления, которые стандартно обрабатываются по-разному в движках рендеринга Android и iOS.
3. Оптимизация производительности списков товаров через настройку параметров FlatList.

Использование ИИ позволило значительно сократить время на написание визуальной составляющей. Вместо ручного подбора стилей (StyleSheet), нейросеть использовалась для генерации кода компонентов на основе описания из макетов Figma. Применение автоматической генерации позволило сосредоточить внимание автора на более сложных процессах, таких как интеграция с API через Axios и логика работы в контейнерах Docker.

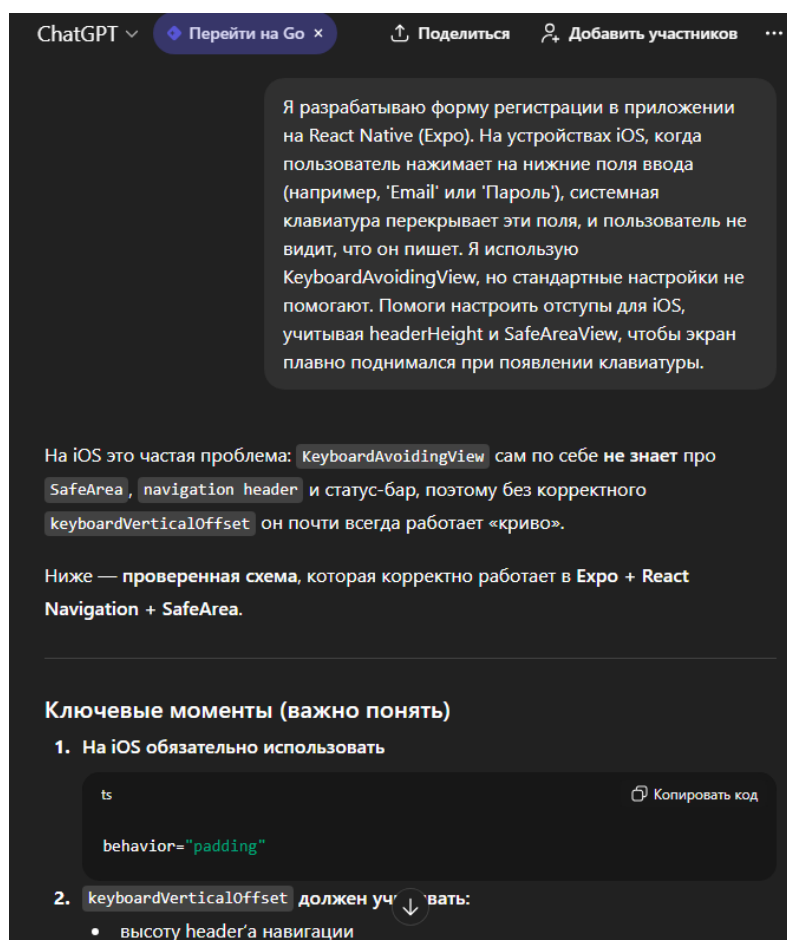


Рисунок 6. Пример решения бага с помощью ИИ (режим чат)

Полученные результаты разработки и практический опыт применения современных инструментов подтверждают целесообразность выбора используемого технологического стека. React Native в сочетании с TypeScript и средой Expo представляет собой эффективное решение для создания современных мобильных сервисов в ритейл-сегменте, в том числе для аптечных сетей, где важны скорость разработки, стабильность работы и масштабируемость приложения.

Опыт реализации проекта показал наличие ряда практических эффектов, влияющих на эффективность разработки и сопровождения приложения:

1. Экономическая эффективность. Использование кроссплатформенного подхода привело к снижению затрат на разработку и поддержку приложения на 40-50 % по сравнению с нативными решениями. Такой результат особенно важен при масштабировании продукта и одновременной публикации приложения на нескольких платформах (AppStore, Google Play и RuStore) при сохранении единой бизнес-логики.

2. Надёжность и безопасность. Применение TypeScript обеспечило высокий уровень контроля над данными, минимизируя вероятность критических ошибок при обработке заказов лекарственных средств. Интеграция биометрических механизмов аутентификации (FaceID и TouchID) подтвердила

возможность создания безопасной среды для работы с персональными данными пользователей в рамках гибридных приложений.

3. Сокращение трудоёмкости разработки. Использование инструментов искусственного интеллекта в процессе отладки и генерации компонентов позволило автоматизировать рутинные операции, такие как валидация форм и устранение кроссплатформенных особенностей верстки, высвобождая ресурсы для решения архитектурных и задач обеспечения безопасности.

4. Архитектурная гибкость. Применение модели Monorepo и контейнеризации с использованием Docker продемонстрировало эффективность синхронизации разработки серверной и клиентской частей приложения, упростило процессы развертывания и обеспечило стабильность работы серверной логики на языке C#.

В качестве перспективы дальнейшего развития проекта можно выделить внедрение систем машинного обучения для персональных рекомендаций товаров на основе истории покупок пользователя, а также глубокую интеграцию с государственными сервисами электронных рецептов. Таким образом, разработанное приложение представляет собой не просто программный продукт, а гибкую платформу, готовую к дальнейшей цифровой трансформации аптечного бизнеса.

Библиографический список

1. Брылёва А.А. Программные средства создания интернет-приложений: учебное пособие. Минск.: РИПО, 2022. 485 с.
2. Кнышов В.А. Преимущества использования Docker-контейнеров для разработки веб-приложений // Моя профессиональная карьера. 2022. Т. 2, № 43. С. 22-27.
3. Мухамедин А.А. Исследование React Native и Flutter, кроссплатформенных фреймворков для мобильных приложений // Вестник университета Шакарима. Серия технические науки. 2024. № 2(14). С. 26-33.
4. Пайгетов А.В. Подходы разработки на фреймворке Ехро // Молодой ученый. 2020. № 3(293). С. 22-25.