бытийная модель прослеживается на всем пути обработки запросов.

Список литературы

1. Амиров С. Н. Особенности разработки высоконагруженных систем // International Journal of Open Information Technologies. 2020. №8. URL: https://cyberleninka.ru/article/n/osobennosti-razrabotki-vysokonagruzhennyh-sistem (дата обращения: 11.12.2025).

2. Букреев Д. А., Луц С. М. Особенности разработки web-платформ автоматизированного взаимодействия с клиентом // Современные проблемы геометрического моделирования и информационные технологии: материалы II Межрегиональной научно-практической конференции преподавателей и студентов, посвященной 60-летию образования Мелитопольской школы прикладной геометрии (Мелитополь, 28 мая 2024 года). Мелитополь: Мелитопольский государственный университет, 2024. С. 76-85. EDN: BVUZYR.

3. Журавлев Д. В., Букреев Д. А. Современное состояние реактивных технологий frontend разработки // Современные проблемы геометрического моделирования и информационные технологии: материалы II Межрегиональной научно-практической конференции преподавателей и студентов, посвященной 60-летию образования Мелитопольской школы прикладной геометрии (Мелитополь, 28 мая 2024 года). Мелитополь: Мелитопольский государственный университет, 2024. С. 152-157. EDN: EGZAEX.

4. Латыпов Э. Ф. Сравнительный анализ работы веб-серверов apache и nginx // Экономика и социум. 2017. №6-2 (37). URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-raboty-veb-serverov-apache-i-nginx (дата обращения: 11.12.2025).

5. Хомякова А. А. Возможности программного обеспечения «веб-серверов» Apache и MS IIS по противодействию деструктивным информационным кибернетическим воздействиям // Новые информационные технологии в автоматизированных системах. 2019. № 22.

АНАЛИЗ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ РАЗРАБОТКИ И ПРОЕКТИРОВАНИЯ ВЕБ-РЕСУРСА

Задорожный А. Я., Букреев Д. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: dmitriy.bukreev@mel-su.ru

Научный руководитель: Букреев Д. А.

Введение

Стремительное развитие цифровых технологий привело к существенному усложнению задач, связанных с проектированием и разработкой веб-ресурсов. Если ранее веб-сайт выполнял преимущественно информационную функцию и строился на основе минимального набора технологий, то сегодня он представляет собой полноценную программную систему, обеспечивающую интерактивное взаимодействие, обработку данных, интеграцию со сторонними сервисами и адаптацию под различные пользовательские устройства. Рост пользовательских ожиданий в отношении удобства, скорости загрузки и безопасности существенно влияет на выбор инструментов и технологий, а сама веб-разработка становится междисциплинарным направлением, объединяющим программирование, проектирование интерфейсов, архитектуру приложений и инфраструктурные решения.

Современный веб-ресурс должен одновременно удовлетворять целому ряду требований: обеспечивать понятную навигацию и визуальную целостность интерфейса, корректно отображаться на различных устройствах, быстро реагировать на действия пользователя и гарантировать надёжную защиту данных. Достижение такого сочетания характеристик возможно только при грамотном выборе технологического стека и архитектурных подходов. Анализ современных технологий разработки позволяет выявить ключевые тенденции отрасли, связанные с переходом к одностраничным приложениям, широким распространением реактивных фреймворков, использованием гибких архитектурных подходов и развитием инструментов автоматизации. Значимую роль играет и этап проектирования веб-ресурса, включающий UX/UI-аналитику, создание прототипов, оценку пользовательских сценариев и выбор архитектурного решения. Именно сочетание технологической и проектной составляющих определяет качество конечного продукта.

Цель исследования – анализ современных технологий разработки и проектирования веб-ресурсов, выявление их особенностей, преимуществ и ограничений, а также формирование рекомендаций по выбору технологического стека в зависимости от задач веб-проекта.

Материалы и методы исследования

Материалом исследования послужил комплекс современных технологий, применяемых при проектировании и разработке веб-ресурсов, а также нормативные и методические документы, описывающие требования к веб-приложениям с точки зрения функциональности, удобства использования, производительности и безопасности. В обзор были включены как классические технологии, лежащие в основе веба (HTML5, CSS3, стандарты ECMAScript), так и современные фреймворки фронтенд- и бэкенд-разработки, архитектурные подходы к построению веб-приложений, инструменты автоматизации и средства обеспечения качества. Дополнительно учитывался практический опыт реализации веб-ресурсов образовательного и информационного профиля, что позволило сопоставить теоретические положения с реальными сценариями разработки.

Методологическая основа исследования опирается на сочетание теоретического анализа и сравнительного подхода. Дополнительно использовались элементы системного подхода и структурного анализа: технологии рассматривались не изолированно, а как части единого технологического стека, включающего этапы проектирования, реализации, тестирования и сопровождения веб-ресурса. Это позволило оценить совместимость инструментов между собой, а также определить типичные конфигу-

рации стеков, применяемых на практике. Обобщение полученных результатов осуществлялось посредством аналитического синтеза, в ходе которого были сформулированы выводы и рекомендации по выбору технологий для разработки и проектирования современных веб-ресурсов.

Результаты исследования и их обсуждение

Развитие веб-технологий отражает общую динамику цифровой среды, которая за сравнительно короткий период прошла путь от простых текстовых страниц к сложным интерактивным системам, обеспечивающим полноценную пользовательскую функциональность. Сегодня веб-ресурс уже не рассматривается как статический источник информации – он воспринимается как программный продукт со своей архитектурой, логикой взаимодействия и требованиями к качеству. Поэтому анализ современного состояния веб-технологий предполагает не только исторический обзор, но и понимание тех факторов, которые определяют подходы к проектированию и разработке современных веб-систем.

Первые веб-страницы были строго статичными и решали одну задачу – предоставление информации. HTML в ранних версиях обеспечивал лишь базовую структуру текста, а возможности оформления и взаимодействия были минимальны. На этом этапе именно доступность и простота являлись основными преимуществами веба.

Однако по мере роста аудитории и усложнения пользовательских запросов появилась необходимость в более динамичном содержимом. Появление JavaScript и развитие CSS фактически изменили характер веба: стало возможным не только управлять внешним видом страниц, но и реагировать на действия пользователя, обновлять данные без перезагрузки и создавать первые элементы интерактивности.

Эффективная разработка веб-ресурса начинается задолго до появления первой строки кода. Современная практика подчёркивает значимость проектирования как самостоятельного этапа, от которого напрямую зависит удобство взаимодействия пользователя с системой, её архитектурная устойчивость и дальнейшая масштабируемость. Веб-проектирование объединяет аналитическую, дизайнерскую и инженерную составляющие, что требует от разработчика не только владения инструментами, но и понимания логики построения пользовательских сценариев, особенностей работы интерфейса и архитектурных принципов будущего приложения.

Современный подход к проектированию веб-ресурсов строится вокруг идеи, что успешный продукт – это не просто корректно работающий сайт, а удобный, понятный и технически надёжный инструмент, который воспринимается пользователем естественно и предсказуемо. Поэтому на ранних этапах особое значение приобретает разработка структуры взаимодействия, а также выбор архитектурного решения, позволяющего обеспечить устойчивую работу приложения независимо от роста нагрузки или расширения функционала [1].

UX/UI-проектирование и прототипирование

Пользовательский опыт (UX) играет ключевую роль в формировании цифровых продуктов. Прежде чем приступить к разработке интерфейса, важно понять, каким способом пользователь будет взаимодействовать с системой, какие действия для него являются приоритетными и какие сценарии должны быть выполнены с минимальными усилиями. На основе таких требований формируются пользовательские пути, определяется логика переходов и выбираются основные паттерны интерфейса.

Инструменты прототипирования – такие как Figma, Adobe XD или Sketch – позволяют визуализировать структуру будущего веб-ресурса ещё до начала разработки. Преимущество такого подхода заключается в возможности быстро проверять идеи, корректировать структуру и обсуждать варианты взаимодействия с командой разработки и заказчиками, не прибегая к затратной переработке кода.

Параллельно создаются UI-макеты, в которых разрабатываются визуальные компоненты – сетка, типографика, цветовая схема, элементы управления. Важным аспектом является соответствие дизайн-системам, которые обеспечивают единообразие интерфейса и упрощают дальнейшую разработку.

Архитектурные подходы к разработке веб-ресурсов

После определения структуры интерфейса возникает вопрос выбора архитектуры, которая обеспечит устойчивость и гибкость будущей системы. Развитие веб-технологий привело к многообразию архитектурных решений, каждое из которых по-своему отвечает вызовам современного веба [2].

Традиционная монолитная архитектура всё ещё применяется в небольших проектах, где важна простота развертывания и целостность кода. Однако с увеличением функциональности она быстро становится громоздкой и менее удобной для сопровождения. На смену монолиту пришли микросервисные подходы, в которых крупная система разделяется на самостоятельные сервисы, взаимодействующие через API. Для веб-приложений это означает возможность независимого обновления модулей, масштабирования отдельных компонентов и гибкого распределения нагрузки.

Ещё один значимый тренд – микрофронтенды, позволяющие разделять клиентскую часть большого приложения между несколькими

ми командами и технологиями. Такой подход особенно актуален для крупных организаций, где разные модули интерфейса развиваются автономно. Отдельного внимания заслуживает архитектурная концепция JAMstack, ориентированная на ускорение загрузки и повышение безопасности за счёт разделения статического рендеринга интерфейса и динамических функций, вынесенных в облачные сервисы. Этот подход активно применяется при создании ресурсных сайтов, маркетинговых платформ и приложений, ориентированных на быструю доставку контента. Каждая архитектура предлагает своё сочетание преимуществ, и выбор зависит от характера проекта, его функциональных требований и перспектив развития.

Для осмысления современной фронтенд-среды важно рассматривать её не как набор отдельных технологий, а как экосистему, где каждый компонент играет свою роль: одни отвечают за структуру и стиль, другие – за реактивность и шаблоны, третьи – за сборку и оптимизацию. Такое взаимодействие формирует основу пользовательского опыта, на который опирается всё приложение. В основе фронтенда по-прежнему лежат три стандарта: HTML, CSS и JavaScript. Однако их современное состояние серьёзно отличается от того, каким оно было даже десять лет назад.

Ключевым шагом в развитии фронтенда стало появление реактивных фреймворков, которые стандартизировали способы создания сложных интерфейсов.

1. React, созданный компанией Meta, предложил компонентный подход, в котором каждый элемент интерфейса рассматривается как самостоятельный модуль со своим состоянием.

2. Vue.js сделал акцент на простоте внедрения и постепенном наращивании функциональности.

3. Angular, напротив, представляет собой комплексный фреймворк, включающий инструменты маршрутизации, работы с формами, модульной организацией кода.

Последние годы отмечены ростом интереса к более лёгким решениям, таким как Svelte, который выносит большую часть работы на этап сборки, что делает итоговый код компактнее и быстрее. Фреймворки перестали быть просто инструментами – они стали платформами для создания интерфейсов, задающими архитектурные и организационные решения.

Современная разработка невозможна без инструментов, которые обеспечивают удобство написания и оптимизацию итогового кода. Большинство проектов строится на экосистеме npm и использует сборщики – будь то Webpack, Parcel, Rollup или более современные решения вроде Vite.

Эти инструменты:

– упрощают структуру проекта;

– позволяют использовать модульность кода;

– оптимизируют загрузку ресурсов;

– обеспечивают живую перезагрузку интерфейса при разработке;

– поддерживают интеграцию TypeScript, препроцессоров и тестовых библиотек.

Параллельно развиваются средства анализа кода, тестирования и форматирования: Jest, Cypress, ESLint, Prettier. Они становятся частью культуры разработки, формируя привычку поддерживать качество и читаемость интерфейсной части.

Рост мобильного трафика сделал адаптивность не дополнительной функцией, а обязательным условием. Современные веб-ресурсы проектируются по принципу mobile-first, что означает первичную ориентацию на небольшие экраны. Использование гибких сеток, медиазапросов, относительных единиц измерения и адаптивных компонентов интерфейса стало нормой. Библиотеки наподобие Bootstrap или Tailwind CSS предлагают готовые решения, которые позволяют ускорить разработку и обеспечить единообразие дизайна.

Адаптивность сегодня – это не только подгонка интерфейса под разные устройства, но и оптимизация загрузки ресурсов, поддержка жестов, снижение энергопотребления и корректная работа в условиях ограниченного канала связи. Если фронтенд определяет то, как пользователь видит и воспринимает веб-ресурс, то бэкенд формирует основу его функциональности. Именно серверная часть отвечает за обработку данных, управление логикой приложения, безопасность, взаимодействие с внешними сервисами и хранение информации. Из-за роста требований к производительности и масштабируемости бэкенд-разработка стала направлением, где активно развиваются новые платформы, архитектурные подходы и инструменты автоматизации.

Современный бэкенд представляет собой сочетание языков программирования, фреймворков, сетевых протоколов, систем хранения данных и инфраструктурных решений. Вместе они образуют технологический фундамент, на котором строится веб-приложение, и именно от качества этих решений зависит стабильность и надежность системы. Разнообразие серверных платформ объясняется различными требованиями проектов: от быстрых прототипов до высоконагруженных корпоративных систем [7].

1. Node.js стал одним из наиболее распространённых решений благодаря своей событийной модели и использованию одного языка – JavaScript – по всему стеку разработки.

2. Python через фреймворки Django и Flask привлекает разработчиков лаконичностью синтаксиса и обширной экосистемой библиотеr.

3. PHP, несмотря на долгую историю критики, остаётся актуальным благодаря устойчивой

инфраструктуре и популярным фреймворкам, таким как Laravel.

4. Java и C# традиционно используются в крупных корпоративных системах, где важны строгая типизация, устойчивость и производительность.

5. Go привлекает вниманием благодаря своей лёгкости, низким накладным расходам и высокой производительности.

Фреймворки определяют структуру серверного приложения и существенно ускоряют разработку:

1. В экосистеме Node.js широко используется Express, который обеспечивает гибкость и минималистичную архитектуру. Более структурированным подходом обладает NestJS, объединяющий идеи модульности и строгой типизации.

2. В Python-среде Django остаётся наиболее комплексным решением: встроенная ORM, панель администратора, средства авторизации и валидации данных позволяют быстро создавать прототипы и полноценные веб-приложения. Flask выбирают в проектах, где важна лёгкость и минимализм.

3. В мире PHP доминирует Laravel, который предлагает удобный синтаксис, развитые инструменты для миграций БД, маршрутизации и шаблонизации. Это делает его одним из наиболее популярных фреймворков для разработки веб-сервисов.

4. Java-разработка традиционно опирается на Spring, представляющий собой универсальную платформу для корпоративных приложений, микросервисов и сложных распределённых систем.

Каждый из этих фреймворков формирует собственный подход к построению архитектуры проекта, и выбор конкретного решения определяется масштабом и требованиями веб-ресурса.

Современные веб-приложения редко ограничиваются статичными страницами, поэтому важной частью бэкенда является организация взаимодействия между клиентом и сервером [6].

Наиболее распространённым подходом остаётся архитектурный стиль REST, который обеспечивает понятный обмен данными с использованием стандартных HTTP-методов. Его преимущества – простота, универсальность и широкий набор инструментов. Альтернативой REST в последние годы стал GraphQL, позволяющий клиенту запрашивать только те данные, которые ему необходимы. Это снижает объём передаваемой информации и делает интерфейсы гибкими, особенно для сложных клиентских приложений. Для задач реального времени используются WebSocket-соединения, обеспечивающие двусторонний канал взаимодействия. Они незаменимы в приложениях, требующих мгновенной реакции – чаты, онлайн-игры, мониторинговые панели. Таким образом, выбор протокола определяется характером данных, нагрузкой и требованиями к скорости обмена [5].

Работа с данными – ключевой элемент серверной разработки. Выбор системы хранения зависит от структуры данных, масштаба проекта и требований к скорости обработки. Реляционные базы данных, такие как PostgreSQL и MySQL, сохраняют популярность благодаря надёжности, поддержке транзакций и гибкости запросов. Они хорошо подходят для систем, где важна согласованность данных. NoSQL-базы, включая MongoDB [4], Redis и Cassandra, ориентированы на гибкость структур данных, высокую скорость операций и горизонтальное масштабирование [3]. Они применяются в аналитических системах, веб-приложениях с активными пользовательскими данными и сервисах реального времени. Использование ORM-инструментов (например, SQLAlchemy, Prisma, Hibernate) облегчает работу с базами данных, снимая необходимость писать низкоуровневые запросы и позволяя сосредоточиться на логике приложения.

Современная разработка невозможна без инструментов, обеспечивающих поддержку процесса развертывания и сопровождения приложения.

1. Docker стал стандартом контейнеризации: он позволяет запускать приложение в одинаковой среде независимо от рабочего окружения разработчика или сервера. Это повышает устойчивость и предсказуемость развёртывания.

2. Системы CI/CD, такие как GitHub Actions, GitLab CI и Jenkins, автоматизируют тестирование, сборку и публикацию приложения. Это уменьшает человеческий фактор и ускоряет выпуск обновлений.

3. Использование облачных платформ (AWS, Azure, DigitalOcean) позволяет масштабировать веб-ресурс по мере роста нагрузки, обеспечивая распределение трафика, резервирование данных и гибкое управление инфраструктурой.

Практическая разработка веб-ресурса представляет собой последовательность шагов, которые, несмотря на различия в масштабах и специфике проектов, подчиняются общим закономерностям. На этом этапе теоретические решения – выбор архитектуры, стек технологий, проектирование интерфейсов – начинают воплощаться в конкретных технических действиях. Именно здесь становится особенно заметным, насколько удачно были сформулированы требования, насколько рационально выбран технологический стек и насколько продуманной оказалась архитектурная схема. Современная веб-разработка предполагает опору на гибкие процессы, позволяющие адаптировать продукт к изменяющимся условиям. Это требует не только технической компетентности, но и умения организовать разработку таким образом, чтобы каждый этап был логически связан с пре-

дыдущим и одновременно оставался открытым для изменений. Выбор технологий влияет не только на скорость разработки, но и на дальнейшую поддержку проекта. Разработчики ориентируются на несколько факторов: тип создаваемого веб-ресурса, предполагаемую нагрузку, требования к безопасности и взаимодействию с внешними сервисами.

Для информационных порталов или образовательных платформ важна устойчивость и гибкость – здесь нередко используется сочетание React или Vue для интерфейсной части и Django, Laravel или Node.js для серверной логики. В проектах, связанных с потоковыми данными или реалтайм-взаимодействием, предпочтение может отдаваться Node.js или Go. Критерии выбора включают зрелость технологии, активность сообщества, наличие документации, возможность масштабирования и соответствие долгосрочным планам развития продукта. В этом контексте стек перестаёт быть простым набором инструментов – он становится стратегическим решением.

Одним из ключевых элементов практической разработки является тестирование. Оно позволяет не только выявлять ошибки, но и предотвращать появление новых дефектов при дальнейшем развитии проекта. В современной практике используется несколько уровней тестирования:

– модульное, ориентированное на отдельные функции;

– интеграционное, проверяющее взаимодействие частей системы;

– сквозное (end-to-end), моделирующее реальные пользовательские сценарии;

– нагрузочное, оценивающее устойчивость под высокой нагрузкой.

Помимо тестирования, большое значение имеет автоматизация процессов – использование CI/CD, автоматической сборки, проверок качества кода. Это снижает зависимость результатов от человеческого фактора и делает процесс разработки более предсказуемым. Безопасность рассматривается как обязательный элемент современного веб-ресурса. Нарушение в этой сфере приводит не только к техническим сбоям, но и к потере доверия пользователей, что зачастую наносит больший ущерб. Поэтому при разработке учитываются рекомендации OWASP, применяются методы защиты от XSS- и CSRF-атак, реализуются механизмы шифрования данных, а процессы аутентификации и авторизации выстраиваются таким образом, чтобы минимизировать возможные уязвимости.

Заключение

Проведённый анализ современных технологий разработки и проектирования веб-ресурсов показывает, что веб-экосистема продолжает динамично развиваться, предлагая разработчикам всё более разнообразные инструменты и архитектурные решения. Эволюция веба – от статических страниц к многофункциональным интерактивным системам – сопровождается не просто увеличением количества технологий, но и качественным изменением подходов к созданию веб-приложений. Значение приобретают гибкость архитектурных решений, осознанный выбор технологического стека и внимание к пользовательскому опыту, который становится центральным элементом проектирования. Современные фронтенд-технологии создают условия для построения интерфейсов, сравнимых с настольными приложениями по уровню интерактивности и скорости работы, а серверные решения обеспечивают высокую производительность, возможность масштабирования и безопасное управление данными. Всё это делает разработку многоуровневым процессом, в котором важно учитывать не только технические, но и организационные аспекты: от грамотного прототипирования до корректной настройки процессов тестирования и развертывания. Отдельное значение имеет синтез проектных и технологических решений. Практика показывает, что успешный веб-ресурс формируется там, где архитектурная схема, UX-проектирование и выбор инструментов воспринимаются как единая система, а не как набор разрозненных решений. В этом контексте особую ценность приобретают методы, позволяющие оценивать технологии с точки зрения их совместимости, гибкости и потенциала развития. Таким образом, современная веб-разработка требует комплексного подхода, сочетающего техническую экспертизу, понимание пользовательских сценариев и умение работать с быстро меняющейся технологической средой. Перспективы дальнейшего развития связаны с усилением роли реактивных архитектур, интеграцией облачных технологий, распространением микрофронтенд-структур и повышением автоматизации процессов разработки. Всё это позволяет утверждать, что будущие веб-ресурсы будут ещё более гибкими, интеллектуальными и ориентированными на глубокое взаимодействие с пользователем.

Список литературы

1. Аблаева Л. Н., Гафаров А. Р. Современные технологии разработки web-сайта для коммерческих компаний // Информационно-компьютерные технологии в экономике, образовании и социальной сфере, 2020. С. 130-138. EDN: TISXDN.

2. Букреев Д. А., Барановская В. С. Персонализация интерактивных цифровых медиа в образовательной среде // Международный студенческий научный вестник. 2023. № 2. EDN: OSCHEC.

3. Bukreiev D. A., Chorna A. V., Serdiuk I. M., Soloviev V. N. Features of the use of software and hardware of the educational process in the conditions of blended learning. Proceedings of the symposium on advances in educational technology, AET, 2020.

4. Bukreiev D., Chornyi P., Kupchak E., Sender A. Features of the development of an automated educational and control complex for checking the quality of students. CEUR Workshop Proceedings, 2021.

5. Abowd G., Mynatt E. Charting past, present, and future research in ubiquitous computing // ACM Trans. Comput.-Hum. Interact. 2000. № 7(1). P. 29-58.

6. Koutrika G., Ioannidis Y. Personalizing queries based on networks of composite preferences // ACM Trans. Database Syst. 2010. № 35(2). P. 13:1-13:50.

7. Kientz J. Embedded capture and access: encouraging recording and reviewing of data in the caregiving domain // Personal Ubiquitous Comput. 2012. № 16(2). P. 209-221. EDN: OAQAUG.

ТЕХНОЛОГИИ И ИНСТРУМЕНТЫ РЕАЛИЗАЦИИ МОДУЛЬНЫХ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Иваненко О. А., Букреев Д. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: dmitriy.bukreev@mel-su.ru

Научный руководитель: Букреев Д. А.

Введение

Рост функциональной сложности цифровых продуктов и постоянное расширение пользовательских сценариев закономерно приводят к увеличению требований к архитектуре интерфейса. Сегодня пользовательский интерфейс перестает быть простой визуальной надстройкой над программной логикой и становится самостоятельным слоем, отвечающим за качество взаимодействия, удобство восприятия и эмоциональное восприятие продукта. В таких условиях особенно актуальными становятся подходы, позволяющие обеспечить гибкость, структурированность и масштабируемость интерфейсных решений. Одним из них является модульная архитектура пользовательских интерфейсов, которая предполагает построение UI как системы взаимосвязанных, но независимых компонентов.

Интерес к модульным интерфейсам объясняется несколькими факторами. Во-первых, современные цифровые продукты развиваются итеративно, а значит, архитектура должна быть устойчивой к изменению функциональности. Во-вторых, многоплатформенная среда – включая мобильные устройства, планшеты, веб-клиенты – требует единообразия и повторного использования интерфейсных элементов. В-третьих, рост числа технологий и инструментов разработки создаёт возможности для гибкого проектирования интерфейсов, но одновременно повышает требования к их согласованности. Модульный подход позволяет решить эти задачи, обеспечивая независимость компонентов, их композицию и возможность многократного использования в различных контекстах.

Особую значимость модульные интерфейсы приобретают в профессиональных и образовательных цифровых системах, где интерфейс должен быть не только удобным, но и предсказуемым, адаптивным и устойчивым к обновлениям. В таких проектах от интерфейса требуется соблюдение дизайн-системы, согласованность визуальных элементов и отсутствие противоречий в пользовательских сценариях. Именно это делает модульность не просто техническим приёмом, а архитектурной стратегией, влияющей на качество продукта в целом.

Цель исследования – анализ современных технологий и инструментов, применяемых для реализации модульных пользовательских интерфейсов, а также изучение специфики построения таких интерфейсов в условиях разработки образовательных мобильных приложений.

Материалы и методы исследования

Материалом исследования послужили современные технологии проектирования и реализации пользовательских интерфейсов, применяемые в мобильных и кроссплатформенных приложениях. В качестве эмпирической основы использованы практические решения, включая архитектурную модель приложения, структуру модульных компонентов, особенности их взаимодействия и принципы построения интерфейсной части проекта. Дополнительно анализ опирался на документацию инструментов Jetpack Compose, Flutter, React Native и SwiftUI, а также на рекомендации по структуре дизайн-систем и компонентных библиотек, которые определяют актуальные стандарты в сфере UI-разработки.

Методологически исследование основано на сочетании теоретического анализа и сравнительного подхода. На первом этапе были изучены концептуальные источники, посвящённые компонентному проектированию, архитектурным шаблонам пользовательских интерфейсов (MVC, MVP, MVVM) и принципам модульности в интерфейсных системах. Это позволило сформировать целостное представление о фундаментальных подходах, лежащих в основе модульных UI.

На втором этапе проведён сравнительный анализ технологий, применяемых для разработки модульных интерфейсов. Сравнение осуществлялось по ряду критериев: гибкость архитектуры, поддержка композиции компонентов, удобство прототипирования, совместимость с различными платформами, требования к инфраструктуре и возможности расширения функциональности. Такой подход позволил выделить особенности нативных и кроссплатформенных инструментов, а также определить их области рационального применения.

Результаты исследования и их обсуждение

Переход к модульным пользовательским интерфейсам стал естественным ответом на усложнение цифровых систем и рост требований к качеству взаимодействия с пользователем. Современные программы – особенно мобильные и кроссплатформенные – уже не состоят из од-