МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Направление подготовки
09.03.01 Информатика и вычислительная техника

Направленность (профиль)
«Технологии разработки программного обеспечения»

Выпускная квалификационная работа

Разработка обучающего приложения на Telegram-платформе

Обучающегося 4 курса
очной формы обучения
Журавского Матвея Антоновича

Руководитель выпускной квалификационной работы:
кандидат педагогических наук, доцент,
доцент кафедры информационных технологий и электронного обучения
Государев Илья Борисович

Санкт-Петербург
2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С развитием цифровых технологий и расширением возможностей мессенджеров, область образования и обучения переживает значительные изменения. В настоящее время обучающие приложения на Telegram-платформе представляют собой инновационный инструмент, позволяющий создавать гибкие и интерактивные интерфейсы, которые напрямую интегрированы в мессенджер. Одним из самых перспективных направлений в этой области является разработка Mini Apps, которые представляют собой технологию, обеспечивающую удобный и естественный способ запуска собственных веб-приложений прямо внутри Telegram. Кроме того, они предоставляют API для взаимодействия между вашим приложением и мессенджером, обеспечивая быстроту и отзывчивость системы.

Обучающее приложение на Telegram-платформе не обязательно является LMS (Learning Management System), хотя может иметь функциональные элементы обучения. Оно может быть ориентировано на обучение определенным навыкам, знаниям или концепциям через интерактивные методы, такие как викторины, упражнения по программированию и видеоуроки. Основное отличие от LMS заключается в том, что обучающее приложение на Telegram-платформе не обязательно предоставляет полный набор функций управления обучением, хотя может быть интегрировано с другими образовательными ресурсами для поддержки процесса обучения.

Актуальной является **задача** разработки обучающего приложения на Telegram-платформе, использующего Mini Apps. С ростом цифровых технологий и расширением возможностей мессенджеров, создание инновационных образовательных инструментов становится неотъемлемой частью современной образовательной парадигмы. В контексте быстро меняющихся требований к обучению, разработка приложений на Telegram-платформе становится не только актуальной, но и востребованной на рынке образовательных технологий.

Предметом исследования является разработка обучающего приложения на Telegram-платформе, использующего Mini Apps. Это включает в себя анализ существующих требований пользователей, определение основных функциональных возможностей приложения и разработку гибкого, интуитивно понятного интерфейса.

Теоретическая значимость работы заключается в выработке методологии разработки обучающих приложений на Telegram-платформе и изучении принципов дизайна, обеспечивающих удобство использования и эффективность обучения.

Практическая значимость проекта состоит в создании инновационного обучающего приложения, способного улучшить процесс обучения и развития пользователей на Telegram-платформе. Разработанное приложение может быть использовано как образовательными учреждениями, так и индивидуальными пользователями для повышения качества обучения и доступности образовательных ресурсов.

Целью данной выпускной квалификационной работы является создание обучающего приложения на Telegram-платформе с использованием Mini Apps, соответствующего современным требованиям и стандартам образовательных технологий.

Результатом работы станет разработанное обучающее приложение, предоставляющее пользователям доступ к качественному обучающему контенту и инструментам обучения на Telegram-платформе. Полученный опыт и результаты исследования могут быть использованы в дальнейшем развитии образовательных технологий и создании новых образовательных продуктов и сервисов.

INTRODUCTION

With the development of digital technologies and the expansion of messenger capabilities, the field of education and training is undergoing significant changes. Currently, educational applications on the Telegram platform represent an innovative tool that allows for the creation of flexible and interactive interfaces directly integrated into the messenger. One of the most promising areas in this field is the development of Mini Apps, a technology that provides a convenient and natural way to launch custom web applications directly within Telegram. Additionally, they offer an API for interaction between your application and the messenger, ensuring system speed and responsiveness.

An educational application on the Telegram platform is not necessarily an LMS (Learning Management System), although it can have functional learning elements. It can be focused on teaching specific skills, knowledge, or concepts through interactive methods such as quizzes, programming exercises, and video lessons. The main difference from an LMS is that an educational application on the Telegram platform does not necessarily provide a full set of learning management functions, although it can be integrated with other educational resources to support the learning process.

The task of developing an educational application on the Telegram platform using Mini Apps is relevant. With the growth of digital technologies and the expansion of messenger capabilities, the creation of innovative educational tools becomes an integral part of the modern educational paradigm. In the context of rapidly changing learning requirements, the development of applications on the Telegram platform becomes not only relevant but also in demand in the educational technology market.

The subject of the research is the development of an educational application on the Telegram platform using Mini Apps. This includes analyzing existing user requirements, determining the main functional capabilities of the application, and developing a flexible, intuitive interface.

The theoretical significance of the work lies in the development of a methodology for creating educational applications on the Telegram platform and studying design principles that ensure usability and learning effectiveness.

The practical significance of the project consists in the creation of an innovative educational application capable of improving the learning and development process of users on the Telegram platform. The developed application can be used by educational institutions as well as individual users to enhance the quality of learning and the accessibility of educational resources.

The goal of this graduation thesis is to create an educational application on the Telegram platform using Mini Apps, in accordance with modern requirements and standards of educational technologies.

The result of the work will be a developed educational application providing users with access to quality educational content and learning tools on the Telegram platform. The experience and results obtained from the research can be used in the further development of educational technologies and the creation of new educational products and services.

ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Основные термины и понятия

Для полного понимания процесса разработки обучающего приложения на Telegram-платформе необходимо ознакомиться с основными терминами и понятиями. Это поможет установить ключевые концепции и принципы, лежащие в основе создания образовательных приложений на данной платформе.

Telegram — это мощный и многофункциональный мессенджер, который сочетает в себе высокую степень безопасности, скорость и удобство использования. Он предоставляет пользователям и разработчикам инструменты для эффективного взаимодействия и создания инновационных решений прямо внутри платформы.

Telegram-платформа — это универсальное и мощное средство для обмена сообщениями, которое предлагает широкий спектр инструментов для взаимодействия и интеграции, включая боты, каналы, группы и Mini Apps. Telegram известен своей безопасностью, скоростью и многофункциональностью, предоставляя пользователям возможность создавать и управлять контентом, общаться в режиме реального времени, а также интегрировать сторонние приложения и сервисы.

Telegram-бот — это автоматизированное приложение, которое работает внутри мессенджера Telegram и взаимодействует с пользователями через текстовые сообщения, команды и специальные кнопки. Telegram-боты могут выполнять множество функций, таких как предоставление информации, проведение опросов, организация игр и многое другое.

Для создания Mini Apps, которые представляют собой веб-приложения, интегрированные в Telegram, необходим Telegram-бот. Mini Apps используют API

Telegram-ботов для авторизации, интеграции платежей и отправки уведомлений пользователям. Без Telegram-бота невозможно создать и запустить Mini Apps внутри платформы Telegram.

Веб-приложение - это программное решение, доступное через веб-браузер на компьютерах и мобильных устройствах. Оно обладает разнообразными функциональными возможностями, от предоставления информации до выполнения сложных операций.

Mini Apps - это технология, обеспечивающая запуск собственных веб-приложений внутри Telegram, а также предоставляющая API для взаимодействия с мессенджером.

Отличие между веб-приложением и Mini Apps заключается в способе их запуска и интеграции с Telegram. Веб-приложение - это независимое приложение, доступное через браузер, в то время как Mini Apps - это специализированная технология, обеспечивающая работу веб-приложений внутри Telegram. Mini Apps обычно имеют более простой и легкий интерфейс, специально адаптированный для мессенджера, и предоставляют дополнительные функции API для взаимодействия с Telegram.

Mini Apps делают Telegram-платформу не только средством общения, но и мощным инструментом для выполнения различных задач, от простых ботов до полноценных веб-приложений, интегрированных в мессенджер.

Обучающее приложение - это программное средство, спроектированное для обучения пользователей определенным навыкам, знаниям или концепциям с использованием интерактивных методов. Такие приложения могут быть доступны для использования в различных средах, включая платформу Telegram.

LMS (Learning Management System) - это система управления обучением, которая обеспечивает организацию и контроль обучающих материалов, курсов и процессов обучения. Она обычно предоставляет широкий набор функций, таких как управление курсами, оценивание студентов, отслеживание прогресса и другие возможности.

Отличие между обучающим приложением и LMS заключается в их функциональности и назначении. Обучающее приложение сконцентрировано на обучении конкретным навыкам или темам через интерактивные методы, часто имея более ограниченный функционал, чем LMS. В то время как LMS предоставляет полный набор инструментов для управления образовательным процессом, включая организацию курсов, оценку студентов и отчетность.

Эти знания являются фундаментом для более глубокого вхождения в процесс создания обучающего приложения на платформе Telegram, учитывая особенности каждого типа приложений и их применимость к конкретным ситуациям.

1.2 Этапы разработки

Этапы разработки обучающего приложения на Telegram-платформе представляют собой важный процесс, включающий последовательность шагов, необходимых для создания функционального и эффективного приложения. Каждый этап имеет свою значимость и направлен на достижение определенных целей, начиная от изучения требований и анализа концепции приложения до его реализации, тестирования и выпуска. В данной главе мы подробно рассмотрим каждый этап разработки и узнаем, как они взаимосвязаны и способствуют успешному завершению проекта.

Этапы разработки включают:

1. Предпроектное исследование, анализ целей и требований.
2. Создание структуры веб-приложения и прототипов страниц.
3. Проектирование (UX и UI дизайн), разработка дизайн-макетов.
4. Программирование и разработка.
5. Реализация на платформе Telegram.
6. Тестирование функциональности.
7. Релиз приложения на Telegram-платформе и поддержка после выпуска

В процессе предпроектного исследования и анализа целей и требований проводится В процессе предпроектного исследования проводится детальный анализ потребностей пользователей и основных целей проекта. Этот этап включает

изучение рынка, анализ конкурентной среды и потребностей целевой аудитории. На основе собранной информации определяются основные требования к функциональности и дизайну будущего приложения, что является ключевым шагом для успешного развития проекта. Создание структуры веб-приложения и прототипов страниц — один из ключевых этапов разработки. На этом этапе выстраиваются основные элементы и функциональные блоки, определяя общую архитектуру и логику взаимодействия. Первым шагом является создание общей структуры приложения, включая определение основных страниц и разделов, их иерархию и взаимосвязи. Обычно это начинается с создания wireframe-макетов, которые являются простыми схемами интерфейса без деталей дизайна, но отражают основные элементы и расположение контента. Далее разрабатываются прототипы страниц — более детальные макеты с учетом дизайна и интерактивности. Прототипы помогают визуализировать элементы интерфейса и определить пользовательский опыт (UX). После создания структуры и прототипов происходит этап внесения корректировок, что позволяет доработать интерфейс до оптимального состояния. В результате получается четкое представление о том, как будет выглядеть и работать веб-приложение, что является основой для последующей разработки и программирования.

Проектирование UX и UI дизайна создаёт удобный и привлекательный интерфейс, соответствующий потребностям пользователей. Программирование и разработка включают фронтенд и бэкенд. Фронтенд отвечает за внешний вид и удобство использования, используя HTML, CSS, JavaScript и фреймворки, такие как React, Angular или Vue. Бэкенд обрабатывает данные, хранение информации и взаимодействие с внешними сервисами, используя Node.js, Python, Java или PHP и фреймворки, например, Express для Node.js или Django для Python. Оба компонента тесно взаимодействуют, обеспечивая функциональность приложения. Реализация на платформе Telegram включает настройку среды разработки, получение API ключей, создание бота или Mini Apps, и ознакомление с документацией Telegram API. Разработка функциональности может включать создание чат-ботов и

интеграцию с функциями мессенджера. Реализация требует комплексного подхода для оптимальной работы и пользовательского опыта.

Тестирование функциональности является неотъемлемой частью разработки веб-приложения на платформе Telegram. Это включает разработку плана тестирования, ручное и автоматизированное тестирование. Ручное тестирование выявляет разнообразные сценарии использования, проверяя функциональность и взаимодействие с Telegram API. Основные типы тестирования включают модульное, интеграционное, системное и приемочное тестирование. После тестирования проводится анализ результатов, исправление ошибок и доработка функциональности. Тестирование обеспечивает высокое качество и надежность приложения перед его релизом.

Релиз приложения на платформе Telegram включает несколько ключевых шагов. Перед выпуском проводится финальное тестирование для проверки работоспособности, безопасности и соответствия требованиям. После выпуска важна поддержка и обновления, включая исправление ошибок, добавление новых функций и поддержку пользователей. Обновления улучшают функциональность, обеспечивают безопасность и соответствие новым версиям операционных систем. Релиз и поддержка после выпуска важны для успешного использования и развития приложения, удовлетворения потребностей пользователей и достижения целей.

1.3 Подходы к веб-разработке

Подходы к веб-разработке охватывают широкий спектр методов, инструментов и стратегий, которые разработчики применяют для создания веб-приложений и сайтов. Веб-разработка - это динамичная область, постоянно совершенствующаяся и адаптирующаяся под новые технологии, требования пользователей и рыночные условия. Рассмотрим основные подходы к веб-разработке, их особенности и применение в современной индустрии информационных технологий.

Методология Waterfall (водопад) является одним из классических подходов к управлению проектами разработки программного обеспечения. Ее основной

принцип заключается в последовательном выполнении этапов проекта, где каждый следующий этап начинается только после завершения предыдущего. В этом контексте, вступая в изучение методологии Waterfall, мы взглянем на ее преимущества, недостатки и области применения в современной сфере разработки программного обеспечения. На рисунке изображены последовательные этапы проекта, начиная с определения требований и заканчивая релизом и поддержкой после выпуска. Этот подход подразумевает линейное выполнение этапов, где каждый следующий этап начинается только после завершения предыдущего. Подход Waterfall (рис. 1):

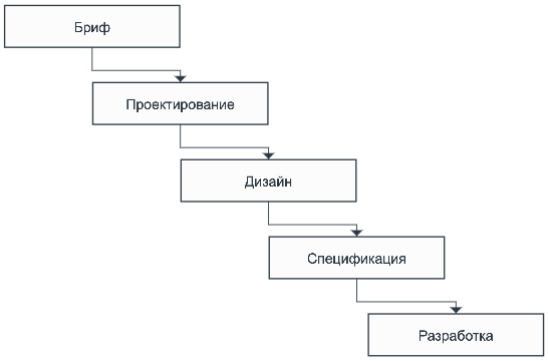

Рисунок 1 – Подход Waterfall

Подход Waterfall получил своё название за то, что этапы работ на календарном плане (диаграмме Ганта) идут сверху вниз, последовательно, подобно водопаду. Этот подход считается классическим, поскольку требования, определяющие работу на протяжении всего проекта, разрабатываются и остаются неизменными с самого начала.

Основные признаки проекта, подходящего для водопадного метода, включают чётко определённый и известный объём работ. Это может быть как небольшая задача, например, создание лендинга или корпоративного сайта, так и крупный проект с жёсткими нормативами, например, разработка программного обеспечения для нефтяной компании. Здесь ключевое значение имеет точное

выполнение плана и минимизация рисков. Waterfall также подходит для государственных и других крупных проектов.

Под термином Agile объединяются гибкие методологии разработки, при которых нет чёткого представления о конечном результате. Например, это может быть разработка продукта, для которого ещё не определён спрос.

Работа ведётся короткими спринтами, главная цель которых - проверка продуктовых гипотез. В гибких методологиях нет необходимости в большом техническом задании (ТЗ). Небольшие кросс-функциональные команды несут ответственность за принятие решений для достижения поставленной цели. Например, программист вправе самостоятельно решать, как реализовать ту или иную функцию для достижения бизнес-результата.

Гибкие методологии подходят для стартапов и проверки гипотез на спрос, когда нет ясного представления о конечном продукте.

1.4 Анализ предметной области

При разработке обучающего приложения на платформе Telegram требуется провести детальный анализ предметной области, связанной с образовательными технологиями и онлайн-обучением. В настоящее время образование переживает значительные изменения, и активное внедрение передовых технологий является основным направлением развития.

Примерами таких передовых технологий являются «Mini Apps» в Telegram. Эти инновационные решения улучшают процесс обучения и расширяют его возможности, делая его более интерактивным, доступным и эффективным.

Актуальное изучение предметной области включает анализ существующих образовательных платформ, предлагающих обучающие материалы и инструменты не только на данной платформе, но и в общем в цифровой среде. Этот обзор поможет выявить сильные и слабые стороны существующих решений, а также определить аспекты, которые можно улучшить или дополнить в новом обучающем приложении.

Понимание требований и ожиданий целевой аудитории при разработке обучающего приложения имеет критическое значение. Это включает студентов различных возрастных групп и уровней образования, специализаций и предпочтений. Адаптация приложения к потребностям и интересам пользователей играет ключевую роль в его приемлемости и эффективности.

Основные шаги анализа предметной области включают:

1. **Изучение требований и целей проекта:** на данном этапе требуется получить всю необходимую информацию о целях и требованиях, которые необходимо реализовать в веб-приложении.
2. **Выявление потребностей пользователей:** следует изучить потребности конечных пользователей. Это включает в себя определение основных задач, которые пользователи будут выполнять с помощью приложения, а также выявление их ожиданий и предпочтений.
3. **Анализ специфики деятельности и аудитории:** важно понимать специфику бизнеса, который будет поддерживаться веб-приложением, а также особенности аудитории, к которой оно будет обращено. Это позволит адаптировать функционал и дизайн приложения под конкретные потребности и ожидания пользователей.
4. **Оценка конкурентной среды:** не менее важным является проведение анализа конкурентов и имеющихся решений на рынке. Это поможет выявить сильные и слабые стороны конкурентов, а также определить уникальное предложение приложения.

Анализ предметной области играет ключевую роль в определении дальнейших шагов разработки и формировании технического задания на основе выявленных требований и потребностей.

1.5 Обзор существующих решений

Для эффективного проектирования и разработки веб-приложения необходимо провести обзор существующих решений на рынке. Этот этап играет важную роль в формировании стратегии разработки и позволяет выявить сильные

и слабые стороны конкурентов, а также определить тенденции и лучшие практики в данной области.

Основные цели обзора существующих решений:

1. **Выявление сильных и слабых сторон:** анализ конкурентов помогает понять, что уже существует на рынке, и выявить как положительные, так и отрицательные аспекты их продуктов. Это позволяет избегать повторения ошибок конкурентов и использовать их успешные решения в собственном проекте.
2. **Определение тенденций и лучших практик:** изучение существующих решений позволяет выявить актуальные тенденции в разработке веб-приложений, а также лучшие практики, которые можно использовать для улучшения своего продукта. Это включает в себя анализ дизайна, функционала, пользовательского опыта и других аспектов.
3. **Определение потребностей рынка:** изучение существующих решений помогает понять, какие потребности рынка уже удовлетворены, а какие еще представляют собой потенциальные возможности для развития нового продукта. Это помогает сфокусировать усилия на ключевых аспектах разработки.

Обзор существующих решений является необходимым этапом, который помогает сформировать стратегию разработки, определить особенности и преимущества продукта, а также выявить потенциальные риски и вызовы, связанные с конкуренцией на рынке.

Обзор существующих образовательных платформ с учетом указанных аспектов:

1. Яндекс Практикум:

- **Сильные стороны:** практическая направленность обучения, что способствует получению реальных навыков. Менторская поддержка и обратная связь, что помогает студентам преодолевать трудности и развиваться быстрее. Актуальность курсов и технологий, благодаря сотрудничеству с ведущими компаниями.
- **Слабые стороны:** высокая стоимость обучения, что может быть недоступно для некоторых пользователей. Ограниченный выбор направлений

обучения по сравнению с другими платформами. Отсутствие гибкости в формате обучения, так как большинство курсов требует полного присутствия и участия в определенное время.

- **Тенденции и лучшие практики:** интенсивное использование практических заданий и проектов для обучения. Внедрение современных технологий, таких как машинное обучение и анализ данных, в образовательный процесс. Уделяют внимание развитию soft skills, таких как коммуникация и работа в команде.
- **Потребности рынка:** Востребованность специалистов с практическими навыками на рынке труда. Необходимость обучения новым технологиям и методам работы, которые востребованы работодателями. Запрос на обучение с учетом актуальных трендов и технологий в различных отраслях.

2. Stepik:

- **Сильные стороны:** гибкий формат обучения, позволяющий проходить курсы в удобное время и темпе. Большое количество бесплатных курсов, доступных для широкой аудитории. Фокус на интерактивности и практических заданиях для лучшего усвоения материала.
- **Слабые стороны:** ограниченное количество продвинутых курсов по сравнению с другими платформами. Не всегда высокий уровень менторской поддержки и обратной связи.
- **Тенденции и лучшие практики:** активное применение методов онлайн-обучения, таких как использование видеоуроков и интерактивных заданий. Постоянное обновление курсов и тематических направлений в соответствии с современными трендами.
- **Потребности рынка:** растущий интерес к онлайн-образованию и самообразованию. Запрос на курсы, охватывающие широкий спектр областей знаний от программирования до гуманитарных наук.

3. Skillbox:

- **Сильные стороны:** акцент на креативности и дизайне в обучении. Профессиональные курсы, разработанные индустрией и специалистами с опытом. Возможность прохождения практических проектов с реальными заказчиками.
- **Слабые стороны:** высокая стоимость некоторых курсов, что может быть ограничивающим фактором для определенной аудитории. Ограниченная география обучения и доступности курсов.
- **Тенденции и лучшие практики:** использование современных методик обучения, таких как онлайн-мастер-классы и вебинары. Постоянное развитие контента в соответствии с требованиями рынка и потребностями студентов.
- **Потребности рынка:** спрос на курсы, ориентированные на практическое применение знаний в реальной работе. Необходимость обучения новым технологиям и трендам в сфере дизайна и digital-индустрии.

4. **GeekBrains:**

- **Сильные стороны:** широкий спектр курсов по информационным технологиям и программированию. Профессиональные преподаватели с практическим опытом в сфере IT. Активное участие студентов в проектах и хакатонах.
- **Слабые стороны:** высокий уровень технической сложности некоторых курсов, что может быть недоступно для начинающих. Не всегда достаточная актуализация контента и учет современных технологий.
- **Тенденции и лучшие практики:** использование онлайн-платформы для обучения с доступом к видеоурокам, учебным материалам и практическим заданиям. Активное развитие сообщества студентов и обмен опытом.
- **Потребности рынка:** спрос на квалифицированных специалистов в области информационных технологий и программирования. Необходимость обучения с учетом последних технологических тенденций и требований рынка труда.

5. **Coursera:**

- **Сильные стороны**: широкий выбор курсов от ведущих университетов и экспертов по различным областям знаний. Гибкий формат обучения с

возможностью самостоятельного темпа и прохождения курсов на разных языках. Сертификаты завершения курсов, признанные мировым сообществом.

- **Слабые стороны:** высокая стоимость сертифицированных курсов и программ специализации, что может быть недоступно для широкой аудитории. Не всегда достаточная глубина материала по некоторым темам.
- **Тенденции и лучшие практики:** активное применение онлайн-обучения с использованием видеолекций, тестирований и практических заданий. Постоянное обновление курсов и адаптация под изменяющиеся требования рынка.
- **Потребности рынка:** растущий интерес к профессиональному образованию и развитию навыков для карьерного роста. Запрос на обучение ведущими экспертами и университетами мира с возможностью получения международно признанных сертификатов.

На текущий момент ни одна из рассмотренных образовательных платформ не имеет полноценного обучающего приложения, созданного специально для платформы Telegram. Вместо этого они используют свои веб-сайты и мобильные приложения для предоставления образовательных услуг. Это может быть связано с тем, что разработка полноценного обучающего приложения для Telegram требует значительных ресурсов и времени, а также с ограничениями самой платформы, которая изначально не была предназначена для комплексных образовательных приложений.

Общие наблюдения по образовательным платформам:

Сильные стороны:

- **Гибкий формат обучения:** многие платформы предлагают гибкий график обучения, что позволяет студентам изучать материалы в удобное время.
- **Широкий выбор курсов:** образовательные платформы обычно предлагают огромное разнообразие курсов по различным областям знаний, что удовлетворяет разнообразные интересы и потребности студентов.
- **Интерактивность и практические задания:** многие платформы активно используют интерактивные методы обучения, включая практические

задания, проекты и кейс-стади, что способствует более глубокому усвоению материала.

- **Мировые эксперты и университеты:** некоторые платформы сотрудничают с мировыми экспертами и университетами, что обеспечивает высокое качество обучения и признание сертификатов на международном уровне.

Слабые стороны:

- **Высокая стоимость сертифицированных курсов:** некоторые платформы предлагают платные сертифицированные курсы, что может быть недоступно для некоторых пользователей.
- **Ограниченная менторская поддержка:** некоторые платформы имеют ограниченный уровень менторской поддержки и обратной связи, что может затруднить процесс обучения для некоторых студентов.
- **Ограниченная гибкость в формате обучения:** некоторые курсы требуют полного присутствия и участия в определенное время, что может быть неудобно для студентов со сложным графиком.

Тенденции и лучшие практики:

- **Интеграция современных технологий:** платформы активно внедряют современные технологии в обучение, такие как машинное обучение, анализ данных, виртуальная реальность и др., чтобы обеспечить более эффективное и интересное обучение.
- **Развитие soft skills:** некоторые платформы уделяют особое внимание развитию soft skills, таких как коммуникация, решение проблем, работа в команде и т. д., что имеет большое значение для успешной карьеры.
- **Обновление и адаптация контента:** образовательные платформы постоянно обновляют свой контент и адаптируют его под изменяющиеся требования рынка и потребности студентов.

Потребности рынка:

- **Востребованные навыки:** рынок труда высоко ценит специалистов с актуальными и практическими навыками в различных сферах, таких как информационные технологии, маркетинг, дизайн и т. д.

- **Глубокое знание области:** специалисты с глубоким знанием и опытом работы в своей области востребованы на рынке труда.
- **Гибкость и адаптивность:** рынок труда требует от специалистов гибкости, способности к быстрому обучению и адаптации к новым технологиям и требованиям.

Эти общие наблюдения помогают понять, как образовательные платформы соответствуют современным требованиям рынка и какие аспекты можно улучшить для более эффективного обучения и подготовки специалистов.

1.6 Определение функциональных и нефункциональных требований

После тщательного анализа предметной области и обзора существующих решений необходимо определить функциональные и нефункциональные требования к разрабатываемому веб-приложению. Этот этап является ключевым для успешной реализации проекта, поскольку отчетливо сформулированные требования определяют основу его функционирования и качества обслуживания.

Функциональные требования к приложению представляют собой описание основных функций, возможностей и интерактивных элементов, которые должны быть реализованы в приложении. Эти требования определяют, как приложение должно взаимодействовать с пользователем и выполнять свою основную задачу. Функциональные требования могут включать в себя:

- **Гибкий график обучения:** приложение должно предоставлять возможность студентам изучать материалы в удобное для них время.
- **Широкий выбор курсов:** для удовлетворения разнообразных интересов и потребностей студентов, в приложении должен быть разнообразный выбор курсов по различным областям знаний.
- **Интерактивность и практические задания:** важным аспектом является активное использование интерактивных методов обучения, таких как практические задания, проекты и кейс-стади, для более глубокого усвоения материала.

- **Мировые эксперты и университеты:** сотрудничество с мировыми экспертами и университетами обеспечит высокое качество обучения и признание сертификатов на международном уровне.

Нефункциональные требования к приложению определяют атрибуты и характеристики, которые не связаны непосредственно с его функционалом, но влияют на его общую производительность, безопасность и удобство использования. Эти требования обычно включают в себя следующие аспекты:

- **Стоимость курсов:** Платформа должна предоставлять разные варианты ценообразования, включая бесплатные курсы и доступные платные сертифицированные курсы, чтобы быть доступной для различных категорий пользователей.
- **Менторская поддержка:** Платформа должна обеспечивать достаточный уровень менторской поддержки и обратной связи для всех студентов, чтобы облегчить процесс обучения и развития.
- **Гибкость в формате обучения:** Платформа должна предлагать гибкие форматы обучения, позволяющие студентам изучать материалы в удобное для них время и темпе.
- **Технологическая поддержка:** Платформа должна быть надежной, быстрой и удобной в использовании на различных устройствах и операционных системах.
- **Актуальность контента:** Платформа должна постоянно обновлять и адаптировать свой контент под изменяющиеся требования рынка и потребности студентов.

Определение функциональных и нефункциональных требований является основой для последующих этапов разработки, таких как проектирование архитектуры, разработка интерфейса, тестирование и внедрение. Корректно сформулированные требования позволяют учесть все необходимые аспекты и создать продукт, который будет отвечать потребностям пользователей и бизнес-задачам заказчика.

При разработке обучающего приложения на Telegram-платформе критически важно четко определить функциональные и нефункциональные требования. Это позволяет точно описать, какие функции приложение должно выполнять, а также какие атрибуты и возможности должны быть обеспечены для эффективной и удобной работы пользователей.

Методы сбора требований могут включать в себя брифы на разработку, личные интервью, изучение документации и участие представителей компании-заказчика. Это позволяет составить подробный документ с требованиями, который послужит основой для разработки технического задания и дальнейшей работы над проектом.

1.7 Прототипирование и дизайн веб-приложения

Процесс прототипирования и дизайна веб-приложения на Telegram-платформе играет важную роль в создании удобного и привлекательного пользовательского интерфейса. Этот этап позволяет не только уяснить видение клиента, но и оперативно вносить изменения до начала фактической разработки интерфейса. Для достижения этих целей мы будем использовать программу Figma, которая является лучшей программой для создания дизайна интерфейсов благодаря своей мощности, гибкости и возможности совместной работы в режиме реального времени.

Сначала определяем расположение кнопок, форм и других элементов интерфейса, создавая варфреймы, которые являются планом расположения элементов на странице. Это помогает представить структуру приложения и пути, которые будет проходить пользователь.

Затем переходим к выбору цветовой палитры, шрифтов, изображений и других дизайнерских элементов, которые визуализируют концепцию приложения и делают его более привлекательным и удобным для использования. Проектирование дизайна включает создание черно-белых схем с комментариями, описывающих экраны и взаимодействия пользователя с ними.

На этапе проектирования также проверяем логику работы приложения и вносим необходимые корректировки, чтобы обеспечить его эффективное функционирование. При создании карты экранов учитываем поведение обычного пользователя при использовании приложения и определяем состояния интерфейса при каждом его взаимодействии.

Для демонстрации работы приложения инвесторам и пользователям можно разработать интерактивный прототип, который позволяет наглядно показать функциональность и удобство использования приложения. Однако, стоит отметить, что мокап в сочетании с user flow diagram также является эффективным прототипом, который описывает будущий продукт с точки зрения пользовательского опыта.

При выборе стиля интерфейса учитываем текущие тенденции в дизайне, адаптивность приложения, время на разработку и внедрение дизайна, чтобы создать привлекательный и функциональный пользовательский интерфейс. На данном рисунке представлен интерфейс, разработанный в программе Figma, для обучающего приложения. Этот интерфейс был создан с учетом современных требований к пользовательскому опыту и функциональности приложений для обучения. Интерфейс обучающего приложения (рис. 1):

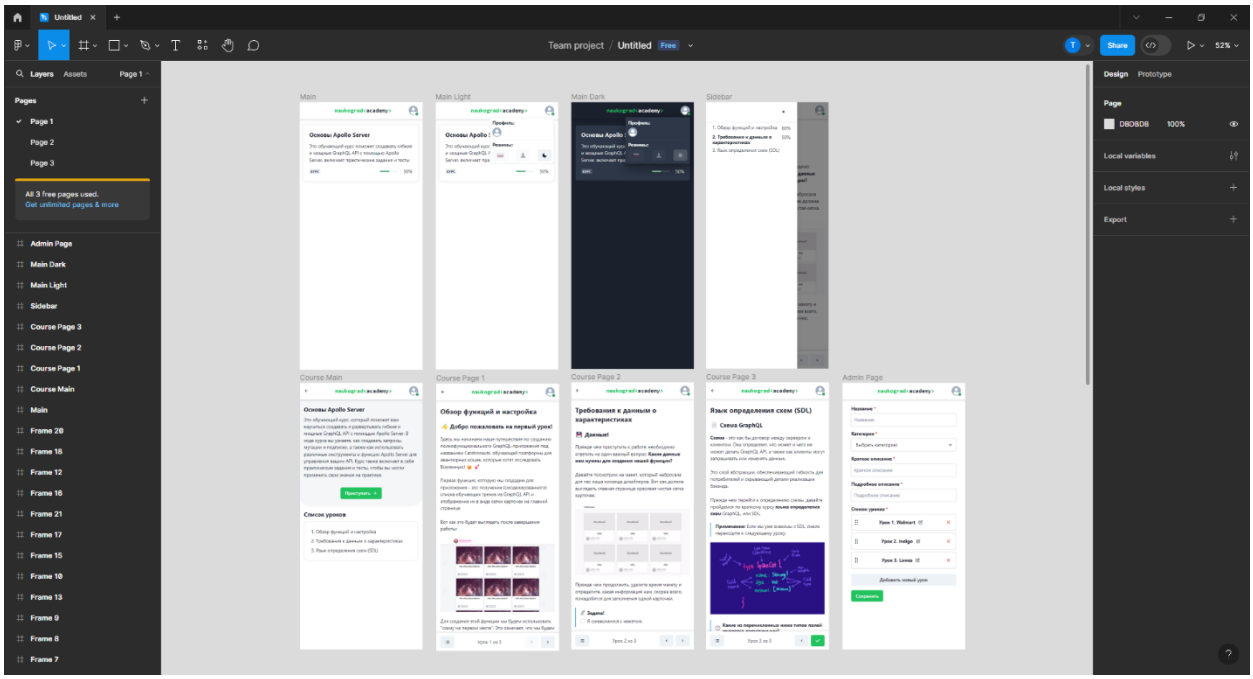

Рисунок 1 – Интерфейс обучающего приложения

Интерфейс в Figma был разработан с учетом лучших практик дизайна пользовательского опыта и современных тенденций в области образовательных приложений. Он представляет собой гармоничное сочетание функциональности, удобства использования и эстетической привлекательности, что делает обучающее приложение привлекательным и эффективным инструментом для обучения и развития пользователей.

Прототипирование и дизайн веб-приложения обеспечивают создание интуитивно понятного и эстетически привлекательного интерфейса, что улучшает пользовательский опыт и повышает качество конечного продукта. Использование современных инструментов и методологий помогает эффективно организовать процесс проектирования, экономить ресурсы и обеспечивать успешную реализацию проекта.

1.8 Проектирование архитектуры веб-приложения

Архитектура веб-приложений определяется как метод организации и структурирования программного кода, который необходим для функционирования приложения. Её можно сравнить с фундаментом, поддерживающим всю структуру приложения. Ключевые компоненты веб-архитектуры включают клиентскую и серверную части, базу данных, а также интерфейсы для взаимодействия с пользователями. В построении надёжной архитектуры приложения следует руководствоваться следующими критериями:

1. **Эффективность:** обеспечение высокой производительности приложения за счёт оптимизации загрузки и обработки данных. Это включает в себя сокращение времени отклика приложения и улучшение общей скорости его работы.
2. **Гибкость:** способность архитектуры адаптироваться к изменяющимся бизнес-требованиям или технологическим условиям без значительных издержек

или перестройки системы. Это важно для поддержания актуальности и конкурентоспособности приложения.

3. **Последовательность и успешность в решении задач:** способность архитектуры обеспечивать надёжное и точное выполнение задач по заранее определённым правилам. Это включает в себя последовательное выполнение процессов и достижение ожидаемых результатов без ошибок или отклонений.

4. **Читаемость и структурированность кода:** организация кода таким образом, чтобы он был легко доступен для понимания и анализа другими разработчиками. Чистый и хорошо структурированный код упрощает поддержку, обновление и масштабирование приложения.

5. **Масштабируемость в процессе разработки:** возможность приложения адаптироваться к растущему числу пользователей или объёму данных, сохраняя при этом стабильность и производительность. Это включает в себя способность системы легко интегрировать новые модули и функции.

6. **Простота использования:** разработка пользовательских интерфейсов, которые являются интуитивно понятными и легкими в использовании для конечных пользователей, что способствует более эффективному взаимодействию с приложением и улучшению общего пользовательского опыта.

1.9 Выбор технологического стека

Выбор технологического стека для разработки обучающего приложения на платформе Telegram с использованием Mini Apps зависит от множества факторов, среди которых требования проекта, опыт разработчика, бюджет и сроки выполнения. Следует отдавать предпочтение технологиям, которые обеспечат эффективную и надежную работу приложения, а также удовлетворят потребности пользователей.

1.9.1 Графический редактор

Графические редакторы являются ключевыми инструментами для дизайнеров, художников и веб-разработчиков. Рассмотрим несколько популярных графических редакторов и выделим преимущества Figma.

Редактор	**Описание**	**Преимущества**	**Недостатки**
Adobe Photoshop	Растровый графический редактор для редактирования изображений и создания цифрового искусства	Обширный набор инструментов, поддержка слоев и масок	Высокая стоимость подписки, сложный для новичков
Adobe Illustrator	Векторный графический редактор для создания логотипов, иконок и иллюстраций	Инструменты для работы с векторной графикой, интеграция с продуктами Adobe	Высокая стоимость подписки, крутая кривая обучения
Sketch	Графический редактор для UI/UX-дизайна, популярен среди веб-дизайнеров	Удобен для создания интерфейсов, простота использования	Доступен только для macOS, ограниченные возможности для растровых изображений
CorelDRAW	Векторный графический редактор для графического дизайна и иллюстраций	Мощные функции для работы с векторной графикой, доступен на Windows и macOS	Высокая стоимость, меньше поддерживаемых форматов, чем у Illustrator
Figma	Облачный графический редактор для дизайнеров интерфейсов и	Совместная работа в реальном времени, облачное	Требуется постоянный доступ в Интернет, некоторые

	UX/UI-дизайнеров	хранилище, универсальные функции для макетов и прототипов, бесплатная версия	функции доступны только в платной версии

Таблица 1 – Список графических редакторов

Среди рассмотренных графических редакторов Figma выделяется благодаря своим уникальным преимуществам, особенно для командной работы и проектирования пользовательских интерфейсов. Вот почему Figma является предпочтительным выбором для многих дизайнеров:

1. **Совместная работа в реальном времени:** Figma позволяет нескольким пользователям одновременно работать над проектом, что значительно упрощает командную работу и ускоряет процесс разработки.

2. **Облачное хранилище:** все проекты хранятся в облаке, что обеспечивает доступ к ним с любого устройства и позволяет легко делиться ими с коллегами и клиентами.

3. **Универсальность:** Figma сочетает в себе функции как для создания макетов, так и для прототипирования, что делает ее универсальным инструментом для UX/UI-дизайнеров.

4. **Бесплатная версия:** Figma предлагает бесплатный тарифный план, который включает в себя большинство необходимых функций для индивидуальных пользователей и небольших команд.

Таким образом, Figma является оптимальным выбором для дизайнеров, которые ищут мощный, гибкий и доступный инструмент для создания и совместной работы над графическими проектами.

1.9.2 Фронтенд

Для реализации фронтенда обучающего приложения на платформе Telegram с использованием Mini Apps был выбран JavaScript. Этот выбор обусловлен

несколькими важными аспектами: распространённостью и поддержкой, так как JavaScript широко используется в веб-разработке и поддерживается всеми современными браузерами; богатой экосистемой, включающей множество библиотек и фреймворков, таких как React, Vue и Angular, которые позволяют создавать интерактивные интерфейсы; опытом разработчика, что ускоряет процесс разработки и снижает кривую обучения; интеграцией с другими технологиями, так как JavaScript легко интегрируется с API и базами данных, что необходимо для Mini Apps; а также адаптивностью и интерактивностью, что позволяет создавать высокоадаптивные и интерактивные веб-приложения, что особенно важно для обучения.

Выбор JavaScript для фронтенда обусловлен его широкими возможностями, совместимостью с современными веб-технологиями и практическими соображениями. Анализ популярных JavaScript-фреймворков показывает, что Angular обеспечивает высокую производительность благодаря Angular CLI и строгим стандартам, имеет обширное сообщество с множеством ресурсов, подходит для крупных проектов и обладает богатой экосистемой. Vue отличается оптимизированной производительностью, лёгкостью изучения благодаря понятной документации, активным сообществом, гибкостью и масштабируемостью, а также хорошей, хотя и не такой обширной, как у конкурентов, экосистемой. React обеспечивает высокую производительность благодаря виртуальному DOM и эффективным методам рендеринга, имеет большое и активное сообщество, подходит для малых и крупных проектов благодаря компонентной архитектуре и обладает богатой экосистемой, включая такие инструменты, как Redux и Next.js.

Среди сборщиков проектов выделяется Vite, который обеспечивает мгновенный запуск и обновление модулей благодаря использованию ES-модулей, требует минимальной конфигурации, легко интегрируется с популярными фреймворками, поддерживает современные функции, такие как HMR, TypeScript, JSX и CSS-препроцессоры, и использует Rollup для сборки продакшен-версии, что обеспечивает высокую производительность и минимальный размер бандла.

Комбинация React, Vite и TypeScript представляет собой мощный и современный стек для фронтенд-разработки, обеспечивая высокую производительность, надёжность и предсказуемость кода, современные возможности, простоту и гибкость в настройке и управлении проектом, а также удобство разработки благодаря улучшенным инструментам. Таким образом, выбор стека React, Vite и TypeScript является оптимальным решением для создания современных веб-приложений.

1.9.3 Бекенд

При разработке современного веб-приложения выбор технологий для бэкенда играет ключевую роль, определяя эффективность обработки запросов, взаимодействия с базой данных и обеспечения безопасности. Node.js — это среда выполнения JavaScript на движке V8 от Google, используемая для серверной разработки. Преимущества Node.js включают однопоточную архитектуру и неблокирующий I/O, что позволяет обрабатывать большое количество одновременных соединений, асинхронность, которая повышает производительность операций ввода-вывода, большую экосистему npm, упрощающую разработку, унификацию языка, которая упрощает процесс разработки, и знание Node.js, ускоряющее процесс разработки.

ORM (Object-Relational Mapping) позволяет разработчикам взаимодействовать с базой данных через объектно-ориентированное программирование. Популярные ORM для Node.js включают Sequelize, поддерживающий различные SQL базы данных, TypeORM, поддерживающий TypeScript и декораторы для определения моделей, и Prisma, предлагающий типизированные запросы и высокую производительность. Преимущества Prisma включают типизированные запросы, обеспечивающие безопасность кода, простоту использования и интеграцию с Node.js и TypeScript, поддержку SQL и NoSQL баз данных, а также автоматическую миграцию баз данных.

GraphQL — это язык запросов для API, разработанный Facebook, который позволяет клиентам запрашивать только необходимые данные, повышая

эффективность. Преимущества GraphQL включают возможность запрашивать только нужные данные, что уменьшает объем передаваемых данных, единую точку доступа, объединяющую данные из различных источников, и сильную типизацию, улучшающую документацию и разработку. Apollo — платформа для создания GraphQL API, включающая сервер, клиент и другие инструменты. Преимущества Apollo включают простоту интеграции с Node.js и другими фреймворками, инструменты Apollo Client для управления состоянием и кэшированием данных, а также развитую экосистему для мониторинга и оптимизации запросов.

Использование стека Node.js, Prisma и Apollo/GraphQL для разработки бэкенда предоставляет множество преимуществ, таких как высокая производительность и масштабируемость благодаря асинхронной архитектуре Node.js, сильная типизация и безопасность данных с помощью Prisma и TypeScript, эффективность запросов благодаря GraphQL, простота интеграции и удобство разработки с Apollo и Prisma, а также опыт разработчика, ускоряющий реализацию проектов. Таким образом, выбор стека Node.js, TypeScript, Prisma и Apollo/GraphQL является оптимальным для создания высокопроизводительных, надежных и удобных API, удовлетворяющих современным требованиям веб-разработки.

1.9.4 База данных

Выбор базы данных играет ключевую роль в эффективности работы современного веб-приложения, поскольку различные базы данных предлагают разные возможности и производительность. Рассмотрим несколько популярных баз данных и объясним, почему мы выбрали PostgreSQL для нашего проекта. MySQL — известная и широко используемая реляционная база данных, славится своей производительностью, надежностью и простотой в использовании. Она поддерживает ACID-транзакции и применяется в таких веб-приложениях, как WordPress. PostgreSQL — высокопроизводительная реляционная база данных с открытым исходным кодом, известная своей расширяемостью и поддержкой сложных запросов. Она поддерживает ACID-транзакции, репликацию, JSON и географические данные. SQLite — встраиваемая база данных, хранящая все данные

в одном файле, часто используется в мобильных приложениях и небольших проектах. Она не требует настройки и легко интегрируется. MongoDB — NoSQL база данных, хранящая данные в формате JSON-подобных документов, хорошо подходит для приложений с неструктурированными данными и высокой масштабируемостью. Она поддерживает репликацию и шардинг. Microsoft SQL Server — коммерческая реляционная база данных от Microsoft, предлагающая мощные инструменты для управления данными, высокую производительность и интеграцию с экосистемой Microsoft.

PostgreSQL выделяется по нескольким причинам. Во-первых, она поддерживает множество расширений, таких как PostGIS для геоданных и полнотекстовый поиск. Во-вторых, PostgreSQL полностью поддерживает ACID-транзакции, обеспечивая надежность и консистентность данных. В-третьих, она предлагает мощные возможности для работы с запросами, включая сложные соединения, подзапросы и оконные функции. В-четвёртых, PostgreSQL поддерживает хранение и запрос данных в формате JSON, что полезно для приложений с полу-структурированными данными. Также она известна своей высокой надежностью, встроенными механизмами шифрования и поддержкой различных механизмов аутентификации и авторизации. Наконец, обширное сообщество и отличная документация облегчают процесс обучения и решения проблем.

Выбор PostgreSQL для проекта обусловлен несколькими причинами: расширяемость и поддержка сложных запросов позволяют работать с большими объемами данных и выполнять сложные запросы; надежность и совместимость с ACID обеспечивают высокую надежность и консистентность данных, что критично для приложения; поддержка JSON идеально подходит для работы с полу-структурированными данными; обширное сообщество и документация помогают быстро решать возникающие вопросы и эффективно использовать базу данных. Таким образом, PostgreSQL как база данных предоставляет все необходимые инструменты для создания эффективного и масштабируемого приложения.

1.10 Архитектура информационной системы

Архитектура информационной системы (ИС) играет ключевую роль в организации и структурировании программного кода веб-приложений, обеспечивая их эффективность и функциональность.

Основой архитектуры веб-приложений является модель «клиент-сервер», которая распределяет нагрузку между серверами (поставщиками услуг) и клиентами (потребителями услуг). Веб-браузеры выступают в роли клиентов, используя DOM (Document Object Model) и JavaScript для интерактивного взаимодействия с данными и функционалом приложения. JavaScript, хотя и не является обязательным, активно используется для создания динамичного и интерактивного пользовательского интерфейса.

Взаимодействие между клиентом и сервером осуществляется через Интернет, что обеспечивает доступ к приложению из любого места и устройства с подключением к сети. Это повышает мобильность и доступность приложения для широкой аудитории.

Одной из ключевых особенностей архитектуры веб-приложений является способность к масштабированию. Гибкость и возможность масштабирования серверной части приложения позволяют обрабатывать большие объемы данных и удовлетворять потребности растущего числа пользователей без ущерба для производительности.

Архитектура информационной системы веб-приложений обеспечивает устойчивую основу для разработки современных, масштабируемых и удобных приложений, делая их привлекательными как для пользователей, так и для разработчиков.

1.11 Взаимодействие между фронтенд и бэкенд

Для проектирования базы данных необходимо выполнить несколько этапов. Первый этап - анализ требований, включающий определение основных требований к базе данных, таких как хранение прогресса пользователей, курсов, уроков и связей между ними, а также уточнение необходимости хранения информации о

пользовательском прогрессе (Progress), курсах (Course) и уроках (Lesson). Второй этап - идентификация сущностей и отношений, предусматривающий выделение основных сущностей и определение их связей и отношений друг с другом, а также определение ключевых полей и уникальных идентификаторов для каждой сущности. Третий этап - нормализация, включающий процесс нормализации для устранения избыточности данных и оптимизации структуры базы данных, а также разделение сущностей на отдельные таблицы для соблюдения нормальных форм. Четвертый этап - создание SQL-схемы базы данных на основе выявленных сущностей и отношений, включая определение типов данных, ключевых ограничений, связей между таблицами и других аспектов структуры базы данных.

На основе предоставленной модели данных выделяются следующие ключевые элементы: таблица Progress, отслеживающая прогресс пользователей в уроках курса; таблица Course, хранящая информацию о курсах, их категориях, описаниях и других атрибутах; таблица Lesson, хранящая данные об уроках, включая контент, порядок отображения и связи с курсами. Каждая таблица имеет свои уникальные идентификаторы (id), а также определены связи между таблицами для обеспечения целостности данных.

После создания схемы базы данных необходимо провести тестирование, проверку схемы на наличие ошибок и некорректных данных; оптимизацию, включающую оптимизацию запросов и структуры базы данных для улучшения производительности; обеспечение безопасности, внедрение мер для защиты данных от несанкционированного доступа и обеспечения их целостности.

Таким образом, проектирование базы данных включает тщательный анализ требований, идентификацию сущностей и их связей, нормализацию, создание SQL-схемы, последующее тестирование и оптимизацию, что обеспечивает надежную, производительную и безопасную работу базы данных в приложении.

1.12 Общий вывод

В данной главе подробно рассмотрены теоретические аспекты, необходимые для успешной разработки веб-приложения. Обсуждены основные термины и

понятия, этапы разработки, различные подходы к веб-разработке, а также проведен анализ предметной области и существующих решений. Определены функциональные и нефункциональные требования, что является фундаментом для дальнейшей работы над проектом.

Прототипирование и дизайн веб-приложения, проектирование архитектуры, выбор технологического стека, включая графический редактор, инструменты для фронтенда и бекенда, база данных и взаимодействие между фронтендом и бекендом, — все эти элементы обеспечивают комплексный подход к созданию качественного и функционального продукта. Проектирование базы данных является важным шагом для эффективного управления данными и обеспечения надежности приложения.

Эти теоретические знания и принципы будут крайне важны для успешной реализации проекта. Они помогут создать стабильное, масштабируемое и удобное в использовании веб-приложение, соответствующее современным стандартам и требованиям рынка.

ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Разработка макета

Разработка макета — ключевой этап создания интерфейса, включающий визуальную компоновку элементов страницы. Начинается с определения целей и требований проекта, использования инструментов, таких как Figma. В Figma создают проект, добавляют основные элементы (заголовки, текст, кнопки, изображения), настраивают стили (цвета, шрифты) и работают с компонентами для согласованного дизайна. Затем добавляют интерактивные элементы и переходы, создавая прототип для тестирования и демонстрации. Макет обеспечивает ясное представление конечного продукта и служит основой для дальнейшей разработки.

В верхней части страницы располагаются логотип сайта «naukograd<academy>» и значок пользователя с аватаркой. Слева вверху надпись «Main» указывает на главную страницу приложения. Центральная часть макета содержит карточку курса с заголовком «Основы Apollo Server». В описании курса говорится, что он помогает создавать гибкие и мощные GraphQL API с использованием Apollo Server, включая практические задания и тесты. На карточке также имеется индикатор прогресса с меткой «КУРС» и прогресс-бар, показывающий, что курс завершен на 50%.

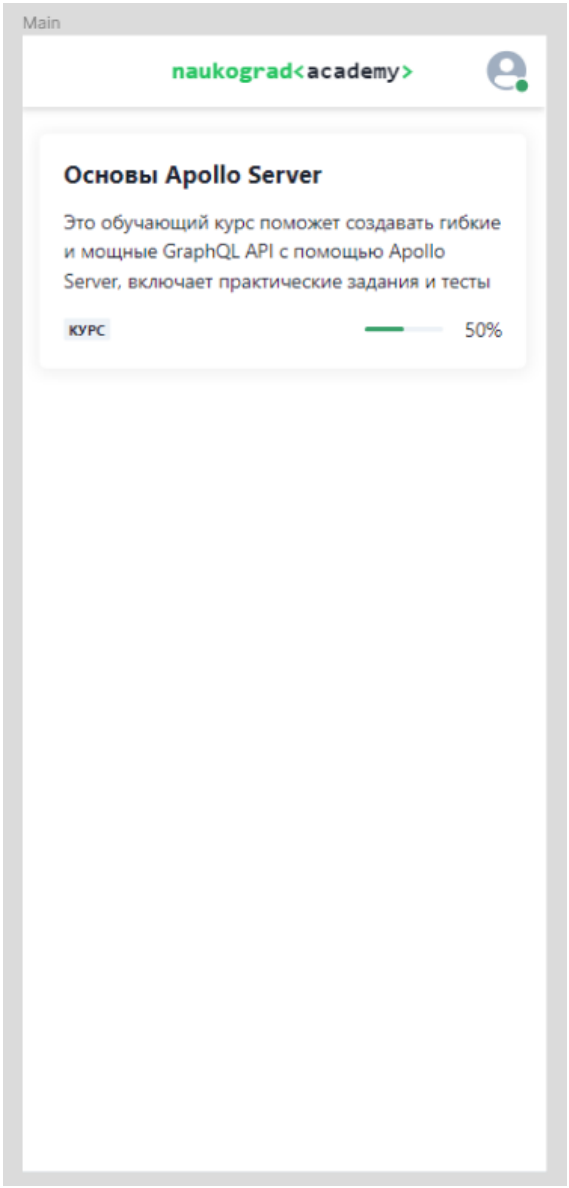

Рисунок 1 – Главная страница обучающего приложения

На изображении представлены три макета главной страницы образовательного приложения «naukograd<academy>» в разных режимах: основной, светлый и темный.

Левый макет:

- Логотип «naukograd<academy>» и значок пользователя с аватаркой.
- Карточка курса «Основы Apollo Server» с описанием и индикатором прогресса (50%).

Средний макет:

- Те же элементы, что и в основном режиме.
- В верхнем правом углу карточки добавлено выпадающее меню профиля с кнопками для изменения режима отображения.

Правый макет:

- Те же элементы, что и в светлом режиме.
- Темный фон для удобства использования в условиях низкой освещенности.

Каждый режим показывает одну и ту же информацию с различными вариантами отображения для улучшения восприятия пользователями в разных условиях.

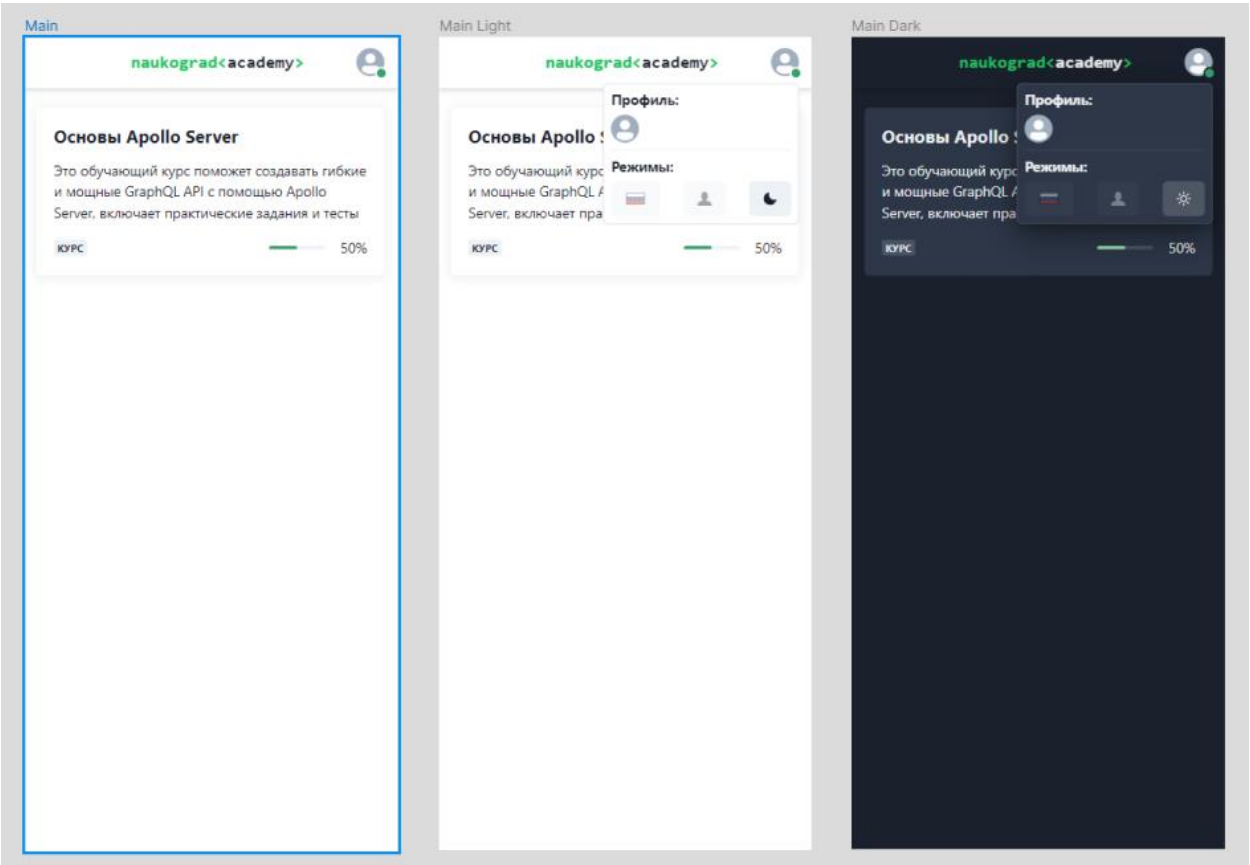

Рисунок 2 – Главная страница в различных цветовых режимах

На изображении представлена страница курса «Основы Apollo Server», включающая детальное описание курса и структуру уроков. Список уроков предоставляет пользователю возможность перейти к конкретному уроку посредством нажатия. Кнопка «Приступить» инициирует переход к первому уроку курса, тогда как кнопка с иконкой стрелки назад возвращает пользователя на главную страницу.

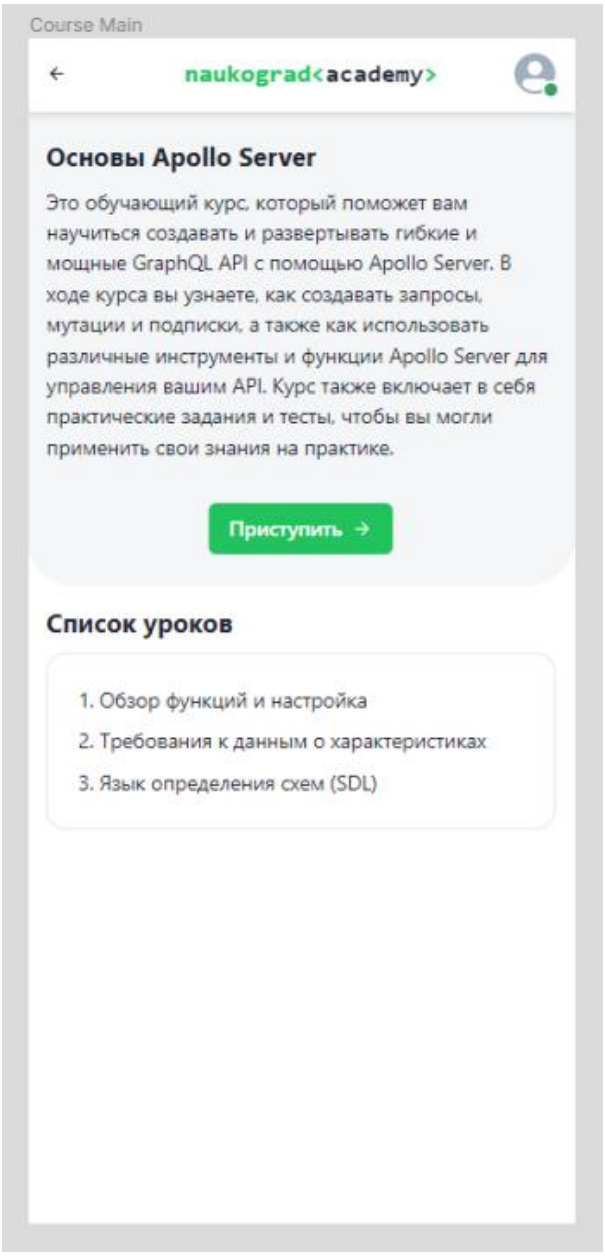

Рисунок 3 – Страница с описанием курса и списком урока

На изображении представлено три страницы учебного курса.

На первой странице курса (Course Main) слева показано меню курса, в котором перечислены три урока:

1. Обзор функций и настройка (66% завершено).
2. Требования к данным о характеристиках (50% завершено).
3. Язык определения схем (SDЛ).

На первой странице курса (Course Page 1) размещен раздел «Обзор функций и настройка». Здесь участники курса начинают изучение создания полнофункциональных приложений на GraphQL. Указано, что обучение будет охватывать моделирование, отображение данных и использование сетки карточек как главного экрана. Также представлены изображения с примерами макетов приложения.

На второй странице курса (Course Page 2) рассматриваются «Требования к данным о характеристиках». Этот раздел объясняет, какие данные необходимы для функционирования приложений. Представлены схемы и макеты, иллюстрирующие организацию данных и взаимосвязи между элементами.

На третьей странице курса (Course Page 3) описывается «Язык определения схем (SDL)». В этом разделе рассматриваются основы использования языка схем (SDL) для описания серверов и клиентов на GraphQL. Приводится пример структуры схемы с типами данных и взаимосвязями между ними.

Таким образом, на изображении демонстрируется последовательное обучение технологиям и инструментам, необходимым для создания и настройки приложений на базе GraphQL.

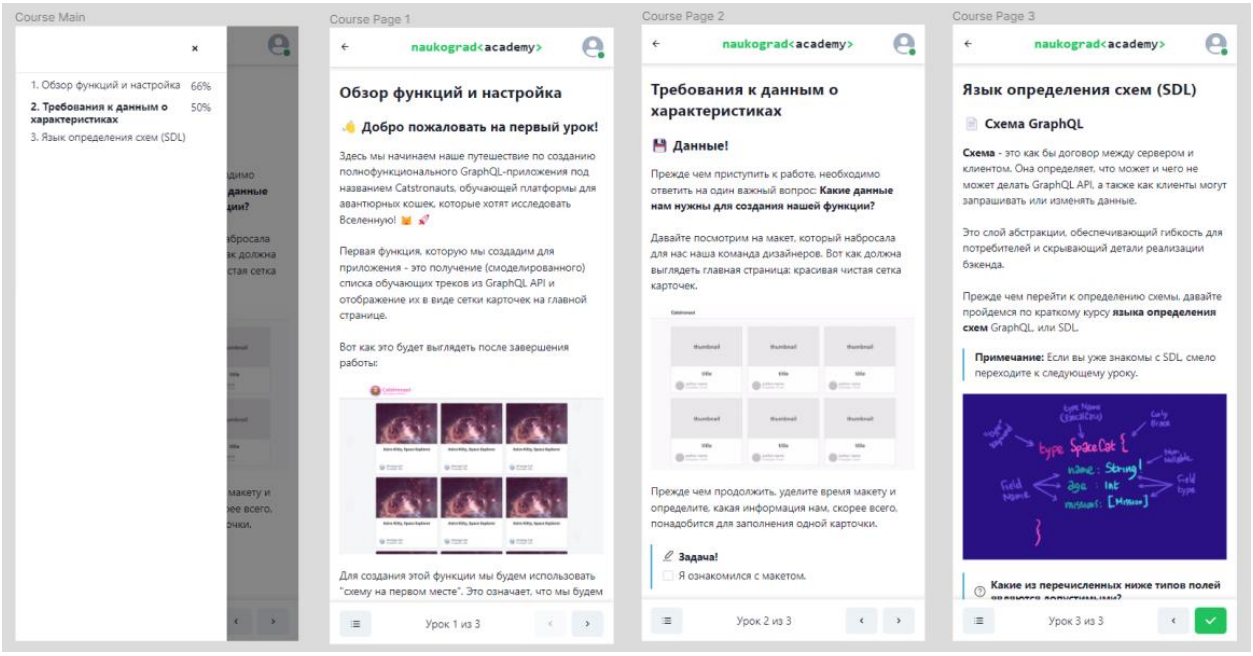

Рисунок 4 – Страницы уроков

На изображении представлена админ-панель для создания курса. В верхней части панели расположены логотип «naukograd<academy>» и значок пользователя с аватаркой. Под ними находятся следующие поля и элементы управления:

1. «Название» (обязательное поле) – текстовое поле для ввода названия курса.
2. «Категория» (обязательное поле) – выпадающее меню для выбора категории курса.
3. «Краткое описание» (обязательное поле) – текстовое поле для ввода краткого описания курса.
4. «Подробное описание» – текстовое поле для ввода подробного описания курса.
5. «Список уроков» (обязательное поле) – список уроков, содержащий следующие элементы:

• Урок 1. «Walmart» – с кнопками для редактирования и удаления.

• Урок 2. «Indigo» – с кнопками для редактирования и удаления.

• Урок 3. «Lowes» – с кнопками для редактирования и удаления.

6. Кнопка «Добавить новый урок» – для добавления нового урока в список.

7. Кнопка «Сохранить» – для сохранения созданного курса.

Элементы формы помечены обязательными полями, что помогает пользователю правильно заполнить всю необходимую информацию для создания нового курса.

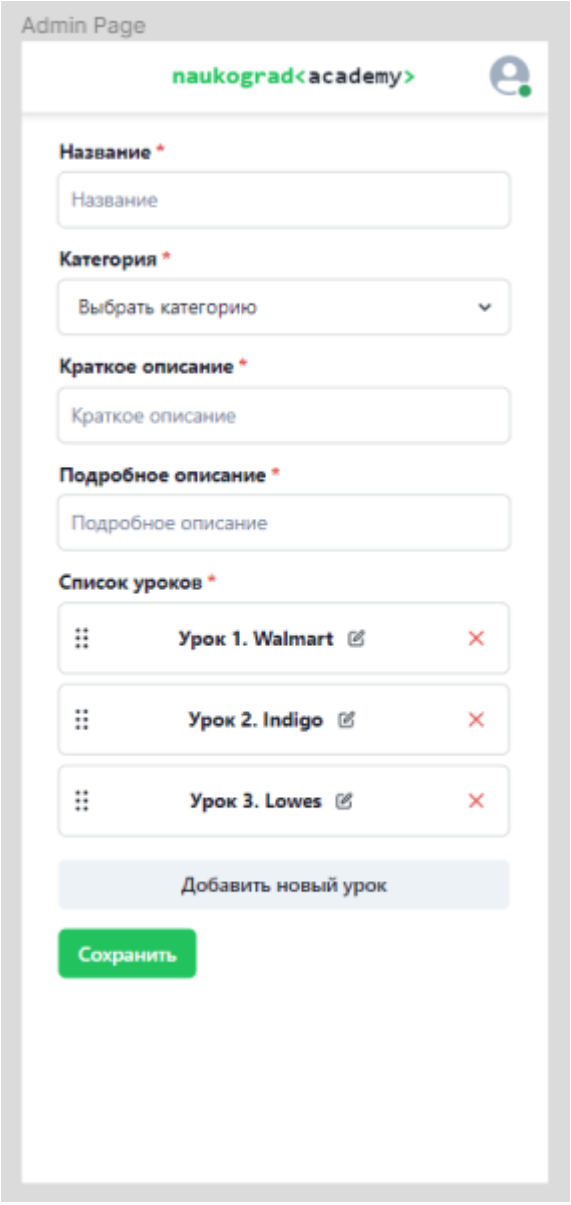

Рисунок 5 – Страницы админ панель

Разработка макета является критически важным этапом создания интерфейса, определяющим структуру и визуальное представление конечного

продукта. Она включает определение целей и требований проекта, использование инструментов проектирования, таких как Figma, настройку стилей, добавление интерактивных элементов и тестирование. Макет обеспечивает ясное представление о внешнем виде и функциональности интерфейса, служит основой для дальнейшей разработки и помогает команде синхронизировать видение конечного продукта. Он упрощает процесс внесения изменений на ранних этапах, повышая эффективность и качество разработки.

2.2 Разработка фронтенда

Архитектура фронтенда этого проекта тщательно организована, чтобы обеспечить максимальную модульность, переиспользуемость и легкость в поддержке кода. Основные директории и файлы проекта аккуратно распределены по нескольким ключевым разделам, каждый из которых играет важную роль в общей структуре приложения.

Директория apollo является важной частью проекта, поскольку содержит конфигурацию и реализации, связанные с Apollo Client. Apollo Client используется для взаимодействия с GraphQL-сервером, что позволяет эффективно управлять запросами и мутациями данных. В этой директории расположены файлы, которые содержат запросы и мутации GraphQL, а также типы данных, используемые в приложении. Эти файлы обеспечивают взаимодействие с сервером и поддерживают консистентность данных во всем приложении.

Директория components представляет собой собрание переиспользуемых React-компонентов, которые разбиты на поддиректории по функциональным областям. Например, поддиректория @Common включает в себя компоненты общего назначения, которые могут быть использованы в различных частях приложения. Это позволяет разработчикам быстро и легко включать готовые компоненты в разные части кода, уменьшая дублирование и облегчая поддержку. Другие поддиректории, такие как Admin, Course, Home и Lesson, содержат компоненты, специфические для административной части, курсов, главной

страницы и уроков соответственно. Это деление на функциональные области помогает структурировать проект и делает его более понятным и легким для навигации.

Директория pages содержит основные страницы приложения, каждая из которых представляет собой отдельную страницу или набор страниц. Эти страницы служат основными точками входа для пользователей и определяют основной функционал приложения. В этой директории находятся файлы, такие как Admin, Course, Home и Lesson, которые связаны с соответствующими функциональными областями административной части, курсов, главной страницы и уроков. Также здесь расположены важные файлы, такие как index.ts, Lesson.tsx и App.tsx, которые являются основными точками входа для различных страниц и разделов приложения.

Таким образом, каждая директория и файл в архитектуре фронтенда имеют свое четкое назначение, что позволяет разработчикам эффективно организовать и поддерживать код приложения. Такой подход обеспечивает высокую степень модульности и переиспользуемости, что в свою очередь способствует более быстрой разработке и легкости вносить изменения и обновления в проект.

2.3 Разработка бэкенда

Разработка бэкенд-части приложения является ключевым этапом в создании современных веб-приложений. Бэкенд отвечает за обработку запросов от клиентов, управление базой данных, бизнес-логику приложения, а также обеспечивает безопасность и производительность системы. Одним из основных инструментов для реализации этих задач является Apollo Server, который позволяет разработчикам легко создавать схемы GraphQL, определять резолверы и управлять взаимодействием с различными источниками данных. Более того, Apollo Server поддерживает интеграцию с различными библиотеками и фреймворками, такими как Express, Koa и другие, что делает его универсальным решением для различных проектов. Помимо этого, он предоставляет множество возможностей для

оптимизации и мониторинга запросов, обеспечивая высокую производительность и гибкость при разработке серверной части приложений.

Вместе с Apollo Server активно используется Prisma для работы с базой данных и связанными задачами. Prisma обеспечивает удобное создание, управление и манипуляции данными в БД, что значительно упрощает разработку. Этот инструмент позволяет определять схему данных, создавать модели, выполнять миграции и осуществлять запросы к БД для получения необходимой информации. Кроме того, Prisma гарантирует высокую производительность и надежность при работе с данными, что делает его незаменимым помощником в разработке бэкенд-части современных приложений.

Resolvers и schema в GraphQL играют ключевую роль в процессе работы с данными и запросами в GraphQL API. Эта таблица описывает две основные директории в рамках разработки GraphQL API: Resolvers и Schema.

Директория	**Назначение**	**Функционал**
resolvers	Resolvers определяют логику обработки запросов GraphQL. Они связывают типы данных с соответствующими данными из источников, такими как базы данных или внешние сервисы.	Resolvers содержат методы для каждого типа и поля в схеме GraphQL. Они получают запросы от клиента и возвращают соответствующие данные. Resolvers могут также выполнять дополнительную обработку данных, валидацию запросов и другие операции.
schema	Schema в GraphQL определяет типы данных, операции и отношения между ними. Она является контрактом между клиентом и сервером, определяя структуру доступных данных и возможных запросов.	Schema GraphQL описывает все доступные типы данных, какие запросы можно выполнять и какие данные можно получить. Она также определяет взаимосвязи между типами, что позволяет клиентам

		запрашивать связанные данные с помощью одного запроса.

Таблица 1 – Описание resolvers и schema

Объединение resolvers и schema обеспечивает гибкость и мощность для создания и использования GraphQL API. Resolvers позволяют обрабатывать запросы, а schema определяет структуру данных, с которыми работает API. Это позволяет эффективно создавать API с учетом конкретных потребностей и обеспечивать высокую производительность и надежность обработки запросов.

Resolvers и schema в GraphQL играют ключевую роль в процессе работы с данными и запросами в GraphQL API.

При разработке бэкенда для веб-приложения используется GraphQL схема, которая определяет типы данных, запросы и мутации. GraphQL позволяет точно указать, какие данные нужны клиентской части приложения, что облегчает передачу данных и улучшает производительность за счет сокращения избыточной информации.

Типы данных в GraphQL схеме определяют структуру данных, которые можно запросить или изменить. Запросы (Query) используются для получения данных, мутации (Mutation) — для создания, обновления и удаления данных. Эта таблица описывает различные запросы (Query) в GraphQL схеме и их логику обращения к базе данных через Prisma.

Название	**Описание**	**Логика обращения к базе данных через Prisma**
getProgress	Этот метод используется для получения информации о прогрессе пользователя в системе.	Использует метод findFirst для получения прогресса пользователя по заданным параметрам.
getCourses	Этот метод возвращает список курсов доступных в системе.	Включает подсчет прогресса для каждого курса и вычисление

		общего прогресса курса.
getCourse	Этот метод возвращает информацию о конкретном курсе, включая уроки.	Возвращает информацию о курсе с уроками.
getLessons	Этот метод возвращает список всех уроков в системе.	Возвращает список всех уроков.
getLesson	Этот метод возвращает информацию о конкретном уроке, включая информацию о прогрессе пользователя и результаты выполнения.	Возвращает информацию об уроке с информацией о прогрессе пользователя и результатами выполнения.

Таблица 2 – Query запросы

Мутации также реализованы с использованием методов Prisma для создания, обновления и удаления данных. Эта таблица описывает различные методы мутаций (Mutation).

Название	**Описание**	**Логика обращения к базе данных через Prisma**
createProgress	Этот метод используется для создания новой записи о прогрессе пользователя в системе или обновления существующей записи.	Использует метод upsert для создания нового прогресса или обновления существующего.
createCourse	Этот метод используется для создания нового курса в системе.	Создает новый курс с уроками и устанавливает связи между уроками.
createLesson	Этот метод используется для создания нового урока в рамках определенного курса.	Создает новый урок.

updateLesson	Этот метод используется для обновления информации об уроке в системе.	Обновляет информацию об уроке.
deleteLesson	Этот метод используется для удаления урока из системы.	Удаляет урок по заданному идентификатору.

Таблица 3 – Mutation запросы

Apollo Playground - это инструмент, который помогает разработчикам тестировать и отлаживать запросы к GraphQL API. На изображении с Apollo Playground видно, как отправляется запрос на создание курса.

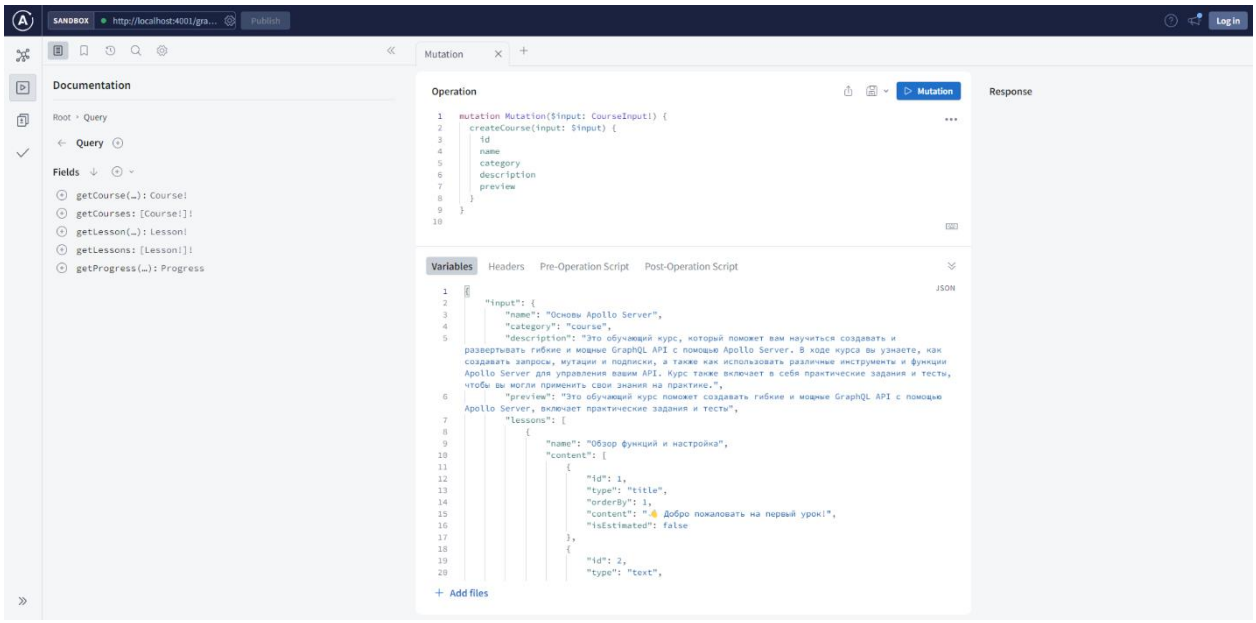

Рисунок 1 - Apollo Playground

В верхней части окна видны основные детали запроса, такие как URL, метод запроса (POST), а также есть возможность выбора типа запроса (например, Query или Mutation) для отправки.

В центре окна находится редактор запроса, где разработчик написал GraphQL запрос на создание курса (mutation createCourse). Запрос содержит необходимые поля для создания курса, такие как название, описание и другие детали.

После отправки запроса, Apollo Playground отображает ответ от сервера в панели результатов. Результат запроса включает информацию о созданном курсе, например, его идентификатор, название, описание и другие атрибуты.

GraphQL позволяет точно управлять данными, избегая избыточной информации. Типы данных в схеме определяют структуру данных для запросов (Query) и мутаций (Mutation), облегчая работу с базой данных через Prisma. Apollo Playground помогает тестировать и отлаживать запросы к GraphQL API, что является важным для разработчиков веб-приложений.

2.4 Рефакторинг веб-приложения

Рефакторинг веб-приложения является неотъемлемой частью процесса разработки, направленной на улучшение его качества, поддерживаемости и расширяемости. Этот процесс включает в себя анализ текущей структуры кода, выявление узких мест и проблемных зон, а также внесение изменений для улучшения общей архитектуры приложения.

Одной из ключевых целей рефакторинга является улучшение структуры кода. Это включает в себя разделение больших и сложных компонентов на более мелкие и управляемые, использование композиции компонентов для уменьшения зависимостей и повторного использования кода, а также применение лучших практик по именованию переменных, функций и компонентов для повышения читаемости кода.

Еще одной важной задачей рефакторинга является улучшение читаемости кода. Это достигается путем удаления избыточного кода, использования ясных и понятных комментариев, а также соблюдения единообразного стиля кодирования во всем проекте. Читаемый и понятный код значительно облегчает поддержку и развитие приложения в будущем.

Для обеспечения единообразного стиля кодирования в проекте часто используют инструменты автоматического форматирования, такие как Prettier. Prettier позволяет автоматически форматировать код в соответствии с заданными правилами, что упрощает поддержку стиля кодирования и снижает вероятность

ошибок из-за несоблюдения форматирования. Использование Prettier в проекте способствует созданию чистого, читаемого и структурированного кода, что в свою очередь улучшает его поддерживаемость и понимание другими разработчиками. Эта таблица описывает различные настройки для форматирования кода и их соответствующие значения и описания в Prettier:

Настройка	**Значение**	**Описание**
singleQuote	true	Использование одинарных кавычек для строковых литералов
semi	true	Добавление точек с запятой в конце выражений
tabWidth	2	Ширина табуляции (в пробелах)
printWidth	120	Максимальная ширина строки кода
trailingComma	none	Отсутствие запятой в конце массивов и объектов
bracketSpacing	true	Добавление пробелов вокруг скобок
jsxBracketSameLine	false	Расположение закрывающей скобки JSX на новой строке
arrowParens	avoid	Избегание оборачивания одиночного аргумента стрелочной функции в круглые скобки
endOfLine	auto	Автоматическое определение символа конца строки (LF для Unix, CRLF для Windows и т. д.)

Таблица 1 – Prettier правила

Эти настройки представляют собой рекомендации или правила форматирования, которые могут быть применены для улучшения стиля

кодирования и обеспечения единообразия в проекте. Например, использование одинарных кавычек для строковых литералов или добавление пробелов вокруг скобок помогает сделать код более читаемым и понятным для других разработчиков. Эта таблица представляет собой описание команд для форматирования и проверки стиля кода с использованием инструмента Prettier:

Команда	**Действие**	**Описание**
format	npx prettier . --write	Команда для автоматического форматирования файлов кода с использованием инструмента Prettier.
check-format	npx prettier . --check	Команда для проверки соответствия стиля файлов кода стандартам Prettier без фактических изменений в файлах.

Таблица 2 – Prettier команды

Данная таблица предоставляет информацию о командах "format" и "check-format" для работы с Prettier, их действиях и общих описаниях, что помогает пользователям понять, как эти команды используются и чем они полезны.

Prettier используется для форматирования кода как на фронтенде, так и на бекенде своих проектов.

Кроме использования инструмента Prettier для автоматического форматирования кода, в проекте также применяется ESLint для обеспечения согласованности стиля кодирования, выявления потенциальных проблем и обеспечения соблюдения стандартов качества кода. Ниже приведены основные настройки и команды для работы с ESLint:

На фронтенде используются настройки и рекомендованные правила ESLint, такие как 'eslint:recommended', 'plugin:@typescript-eslint/recommended', ‘plugin:react/recommended’, 'plugin:react-hooks/recommended'. Эти правила помогают обеспечить согласованность стиля кодирования, выявлять потенциальные проблемы и поддерживать стандарты качества кода для фронтенд-разработки на TypeScript и React. На бекенде также применяются основные рекомендации по правилам ESLint, включая "eslint:recommended" и "plugin:@typescript-eslint/recommended". Эта таблица представляет собой описание различных настроек и расширений (extends) для использования с инструментом ESLint в проекте.

Настройка	**Значение**	**Описание**
extends	eslint:recommended	Настройка для использования рекомендованных правил ESLint, включая базовые правила для поддержания стандартного стиля кодирования.
	plugin:@typescript-eslint/recommended	Плагин @typescript-eslint/recommended рекомендует правила для TypeScript, что помогает обеспечить согласованный и качественный код.
	plugin:react/recommended	Этот плагин предоставляет набор рекомендуемых правил для статического анализа кода в проектах, использующих React. Он охватывает различные аспекты разработки на React, помогая поддерживать

		согласованный стиль написания JSX, правильное использование компонентов и обработку событий.
	plugin:react-hooks/recommended	Рекомендации по использованию правил плагина react-hooks/recommended для React Hooks, что способствует соблюдению стандартов качества кода в React-проектах.

Таблица 3 – ESLint плагины

Эти настройки обеспечивают соблюдение стандартов качества кода и стиля кодирования на всем проекте, как на клиентской, так и на серверной стороне. Эта таблица описывает команды для работы с инструментом ESLint в проекте.

Команда	**Действие**	**Описание**
lint	eslint . --ext .ts,.tsx --report-unused-disable-directives --max-warnings 0	Команда для запуска ESLint с определенными настройками (расширения файлов .ts и .tsx, отчет об неиспользуемых отключениях директив, максимальное количество предупреждений 0).
lint-fix	npx eslint --fix . --ext .ts,.tsx	Команда для запуска ESLint с исправлением найденных проблем в файлах .ts и .tsx, используя автоматическое

		исправление (флаг --fix).

Таблица 4 – ESLint команды

Команды "lint" и "lint-fix" предназначены для запуска ESLint с различными настройками, включая проверку и исправление проблем в файлах с расширениями ts и tsx.

Рефакторинг включает в себя также тестирование изменений для обеспечения их корректной работы и отсутствия новых ошибок. Это помогает предотвратить непредвиденные проблемы после внесения изменений и поддерживать стабильность работы приложения.

В целом, рефакторинг является важным этапом в развитии веб-приложения, который помогает сохранить его актуальность, улучшить качество кода и обеспечить более эффективную работу.

2.5 Тестирование

Тестирование веб-приложения играет ключевую роль в обеспечении его качества и стабильной работы. Оно включает в себя несколько видов тестов, каждый из которых направлен на проверку определенных аспектов приложения.

Основная часть тестирования в начальных этапах разработки была проведена вручную. Это было обусловлено необходимостью быстрого реагирования на изменения в требованиях, а также быстрым выпуском новых функций в соответствии с установленными сроками. В ходе ручного тестирования проверялись основные функциональные возможности приложения, внешний вид и пользовательский опыт.

Ручное тестирование в ходе разработки приложения играло важную роль, обеспечивая ряд значительных преимуществ и поддерживая высокий уровень качества продукта.

1. **Гибкость и оперативность**: Ручное тестирование позволило быстро реагировать на изменения в требованиях и оперативно вносить исправления и

улучшения в приложение. Это было особенно важно на начальных этапах разработки, когда требования могли меняться быстро, и нужно было быстро адаптироваться к новым условиям.

2. **Проверка пользовательского опыта**: Ручное тестирование позволило оценить пользовательский опыт и убедиться, что приложение работает удобно и интуитивно для пользователей. Была активно проверена внешний вид, работа элементов управления, взаимодействие с пользователем и общее впечатление от использования приложения.

3. **Выявление скрытых проблем**: Ручное тестирование помогло выявить скрытые проблемы, которые могли быть упущены при автоматизированном тестировании. Были активно протестированы различные сценарии использования, взаимодействие компонентов и обработка ошибок, что позволило предотвратить потенциальные проблемы и улучшить общую надежность приложения.

4. **Тестирование нестандартных ситуаций**: Ручное тестирование дало возможность проверить работу приложения в различных нестандартных ситуациях, которые могли возникнуть в реальной эксплуатации. Была проведена проверка обработки ошибок, поведения при непредвиденных действиях пользователя и других аспектов, которые могли повлиять на работоспособность приложения.

Исходя из этих факторов, ручное тестирование оказался эффективным инструментом для обеспечения качества и надежности приложения на ранних стадиях разработки. Однако, с увеличением сложности проекта и объема функциональности, автоматизированные тесты становятся неотъемлемой частью процесса тестирования. В дальнейшем планируется расширение набора автоматизированных тестов, включая модульное, интеграционное и сквозное тестирование, что позволит еще более эффективно обеспечить качество и надежность разрабатываемого продукта.

2.6 Развертывание и публикация веб-приложения

Для начала, необходимо подготовить и настроить сервер на платформе Selectel для развертывания backend части вашего веб-приложения. В первую очередь, убедитесь, что у вас есть учетная запись на Selectel, и что вы ознакомились с основными функциями и возможностями платформы. Загрузите и настройте ваш backend код на сервере Selectel, убедитесь, что все зависимости установлены и сервер запущен. Возможно, вам придется настроить базу данных, а также убедиться, что сервер имеет доступ к необходимым ресурсам и службам.

После этого, переходите к следующему шагу – деплою frontend части вашего приложения на Netlify. Для этого войдите в вашу учетную запись Netlify, если у вас ее еще нет, зарегистрируйтесь. Netlify предоставляет удобный и интуитивно понятный интерфейс, который значительно упрощает процесс развертывания. Создайте новый сайт, следуя инструкциям по развертыванию вашего frontend кода. Netlify автоматически обработает и развернет ваш проект, обеспечив его доступность через предоставленный URL.

После успешного деплоя, Netlify предоставит вам URL для вашего frontend приложения. Этот URL можно использовать для доступа к вашему сайту, а также для интеграции с другими сервисами. Возьмите этот URL и перейдите в Mini Apps. В Mini Apps найдите соответствующее поле для интеграции вашего веб-приложения и вставьте туда URL вашего фронтенд приложения. Это позволит вашему веб-приложению быть доступным через Mini Apps и обеспечит его полноценное функционирование.

Следуя этим шагам, вы сможете развернуть backend на Selectel, frontend на Netlify и интегрировать ваше веб-приложение с Mini Apps, обеспечивая его надежную и стабильную работу.

2.7 Общий вывод

Разработка веб-приложения включает множество этапов, начиная с создания макета и заканчивая развертыванием и публикацией. Каждый из этих этапов играет ключевую роль в формировании конечного продукта. Последовательное выполнение всех этапов разработки и тщательное тестирование гарантируют

создание качественного, надежного и удобного обучающего приложения для Telegram-платформы. Этот процесс также обеспечивает доступность приложения для пользователей и его интеграцию с другими сервисами, такими как Mini Apps, соответствуя требованиям пользователей и техническим стандартам.

ЗАКЛЮЧЕНИЕ

Целью данной выпускной квалификационной работы было создание обучающего приложения на Telegram-платформе с использованием Mini Apps, соответствующего современным требованиям и стандартам образовательных технологий. В процессе разработки были последовательно выполнены все этапы, включая анализ требований пользователей, разработку макета, создание фронтенда и бэкенда, рефакторинг, тестирование, развертывание и публикацию.

Каждый этап разработки был тщательно проработан, что позволило создать качественное, надежное и удобное в использовании обучающее приложение. Разработка макета обеспечила ясное представление о конечном продукте, а создание фронтенда и бэкенда позволило реализовать все необходимые функциональные возможности. Рефакторинг и тестирование способствовали повышению качества кода и стабильности работы приложения. Процесс развертывания и публикации обеспечил доступность приложения для пользователей и его интеграцию с другими сервисами, такими как Mini Apps.

В результате выполненной работы было создано инновационное обучающее приложение, предоставляющее пользователям доступ к качественному обучающему контенту и интерактивным инструментам обучения на Telegram-платформе. Полученный опыт и результаты исследования могут быть использованы в дальнейшем развитии образовательных технологий и создании новых образовательных продуктов и сервисов.

Таким образом, все поставленные цели были успешно достигнуты, и разработка обучающего приложения на Telegram-платформу завершена с положительным результатом.

ЛИТЕРАТУРА

1. Флэнаган, Д. JavaScript. Подробное руководство, 6-е издание. — Пер. с англ. — СПб: Символ-Плюс, 2012. — 1080 с., ил. ISBN 978-5-93286-215-5. URL: resources.oreilly.com/examples/9780596805531/tree/master/examples

2. Флэнаган, Дэвид. JavaScript. Полное руководство, 7-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2021. — 720 с.: ил. — Парал. тит. англ. ISBN 978-5-907203-79-2.

3. Закас, Н. ECMAScript 6 для разработчиков. — СПб.: Питер, 2017. — 352 с.: ил. — (Серия «Библиотека программиста»). ISBN 978-5-496-03037-3. URL: github.com/GossJS/understandinges6, leanpub.com/understandinges6/read, nostarch.com/ecmascript6

4. Черный, Б. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений. — СПб.: Питер, 2021. — 352 с. ISBN 978-5-4461-1651-5.

5. Файн, Я., Моисеев, А. TypeScript быстро. — СПб.: Питер, 2021. — 528 с.

6. Ришкуция, В. Программируй & типизируй. — СПб.: Питер, 2021. — 352 с.: ил. — (Серия «Библиотека программиста»). ISBN 978-5-4461-1692-8.

7. Фрисби, М. JavaScript для профессиональных веб-разработчиков. 4-е международное изд. — СПб.: Питер, 2022. — 1168 с.

8. Государев, И. Б. Введение в веб-разработку на языке JavaScript: учебное пособие / И. Б. Государев. — Санкт-Петербург: Лань, 2022. — 144 с. — ISBN 978-5-8114-3539-5. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/206588 (дата обращения: 10.05.2022). — Режим доступа: для авториз. пользователей.

9. Симпсон, К. Вы пока еще не знаете JS: Познакомьтесь, JavaScript. 2-е изд. — СПб.: Питер, 2022. — 192 с. ISBN 978-5-4461-1875-5.

10. Симпсон, К. Вы не знаете JS: Замыкания и объекты. — СПб.: Питер, 2019. — 336 с.: ил. — (Серия «Бестселлеры O'Reilly»). ISBN 978-5-4461-1255-5. [уже есть 2-е издание].

11. Симпсон, К. Вы не знаете JS: Типы и грамматические конструкции. — СПб.: Питер, 2019. — 240 с.: ISBN 978-5-4461-1266-1.

12. Rauschmayer, Axel. Exploring ES6. — URL: leanpub.com/exploring-es6, 2ality.com.

13. Крокфорд, Д. Как устроен JavaScript. — СПб.: Питер, 2019. — 304 с.

14. Розенталс, Н. Изучаем TypeScript 3. – М.: ДМК Пресс, 2019. – 624 с.

15. Краудер, Т. Джей. Новые возможности JavaScript: как написать чистый код по всем правилам современного языка / Ти Джей Краудер. — Москва: Эксмо, 2023. — 640 с.

16. Фаулер, М. Рефакторинг кода на JavaScript: улучшение проекта существующего кода, 2-е изд.: Пер. с англ. — СПб.: ООО "Диалектика", 2019. — 464 с.

17. Хавербеке, М. Выразительный JavaScript. Современное веб-программирование. 3-е изд. — СПб.: Питер, 2019. — 480 с.: ил. — (Серия «Для профессионалов»). ISBN 978-5-4461-1226-5.

18. Бэнкс, А., Порселло, Е. GraphQL: язык запросов для современных веб-приложений. — СПб.: Питер, 2019. — 240 с.: ил. — (Серия «Бестселлеры O'Reilly»). ISBN 978-5-4461-1143-5.

19. Скотт, А. Д. Разработка на JavaScript. Построение кроссплатформенных приложений с помощью GraphQL, React, React Native и Electron. — СПб.: Питер, 2021. — 320 с.: ил. — (Серия «Бестселлеры O'Reilly»). ISBN 978-5-4461-1462-7.