МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЕЖИ РЕСПУБЛИКИ КРЫМ
Государственное бюджетное образовательное учреждение высшего образования Республики Крым
«Крымский инженерно-педагогический университет имени Февзи Якубова»
(ГБОУВО РК КИПУ имени Февзи Якубова)

Факультет экономики, менеджмента и информационных технологий
Кафедра прикладной информатики

Направление подготовки 09.03.03 Прикладная информатика
Профиль «Прикладная информатика в информационной сфере»

КУРСОВОЙ ПРОЕКТ

по дисциплине «Разработка мобильных приложений»
на тему:
«Разработка мобильного приложения для организации учебных материалов и составления расписания репетиторов»

Студента III курса
группы И-2-21
очной формы обучения
Ганиев Ибраим Ризаевич
_____
(подпись)
Научный руководитель:
преподаватель Танишева С.С.
_____
(подпись)

Симферополь – 2024

АННОТАЦИЯ

Ганиев И.Р. Разработка мобильного приложения для организации учебных материалов и составления расписания репетиторов

Мобильное приложение для организации учебных материалов и составления расписания репетиторов. Проект включает в себя анализ существующих решений, выбор оптимальных технологий и реализацию приложения. Приложение предназначено для помощи студентам и преподавателям в управлении учебными материалами и планировании занятий. Работа содержит анализ объектно-ориентированных языков программирования, рассмотрение популярных IDE и описание предметной области. Основное внимание уделяется удобству использования и мультиплатформенной совместимости.

Ключевые слова: мобильное приложение, Flutter, занятие, IDE, кроссплатформенная разработка, образовательные технологии.

ANNOTATION

Ganiev I.R. Development of a mobile application for organizing educational materials and scheduling tutors

A mobile application for organizing educational materials and scheduling tutors. The project includes an analysis of existing solutions, the choice of optimal technologies and the implementation of the application. The app is designed to help students and teachers manage learning materials and schedule classes. The work contains an analysis of object-oriented programming languages, a review of popular IDEs and a description of the subject area. The main focus is on usability and multiplatform compatibility.

Keywords: mobile application, Flutter, lesson, IDE, cross-platform development, educational technologies.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. Современные образовательные процессы все чаще интегрируют цифровые технологии, что делает актуальной разработку мобильных приложений, способных организовать и оптимизировать учебную деятельность. В частности, создание мобильного приложения для организации учебных материалов и составления расписания репетиторов обусловлено потребностью в улучшении доступности, интерактивности и персонализации образовательного процесса. Эффективная организация учебных материалов и расписаний поможет учащимся и преподавателям лучше планировать своё время, повысить уровень вовлеченности и улучшить образовательные результаты.

Разработанное приложение позволит пользователям (студентам и преподавателям) повысить эффективность учебного процесса благодаря быстрому доступу к необходимым учебным материалам и удобному расписанию занятий. Это способствует лучшему планированию времени и улучшает качество образования.

Целью данного курсового проекта является разработка мобильного приложения, которое позволит учащимся и репетиторам эффективно управлять учебными ресурсами и расписанием. Для достижения этой цели были поставлены следующие **задачи**:

1. проанализировать существующие аналоги и выявить их недостатки;
2. определить функциональные требования к разрабатываемому приложению;
3. выбрать технологический стек для разработки приложения;
4. разработать прототип приложения;
5. собрать приложение.

Объектом исследования является процесс управления учебными материалами и расписаниями в образовательных учреждениях.

Предметом исследования выступает мобильное приложение, предназначенное для оптимизации данных процессов.

Структура курсового проекта. Курсовой проект состоит из введения, двух глав, заключения, списка использованных источников и приложений. Общий объем курсового проекта составляет 35 страниц печатного текста, 12 рисунков и 2 приложения. В первой главе проведён анализ существующих решений и сформулированы требования к разрабатываемому приложению. Вторая глава посвящена технологическим аспектам разработки, описанию архитектуры приложения и разработке. В заключении сформулированы основные выводы и рекомендации по дальнейшему развитию проекта.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

1.1. Общая информация о мобильных приложениях и их категориях

Мобильные приложения — это программные продукты, предназначенные для работы на смартфонах, планшетах и других мобильных устройствах. Они оказывают значительное влияние на современное общество, предоставляя широкий спектр функциональных возможностей от общения и развлечений до бизнеса и образования. Согласно исследованию Statista, на 2021 год в магазинах приложений доступно более 5 миллионов приложений, что свидетельствует об их значительном разнообразии и распространённости [1].

Мобильные приложения можно классифицировать по нескольким основным категориям. Во-первых, по типу платформы: iOS от Apple, Android от Google и реже Windows Phone от Microsoft. Во-вторых, по способу разработки: нативные приложения создаются специально для одной платформы с использованием специфических языков программирования и инструментов (например, Swift для iOS или Kotlin для Android); кроссплатформенные приложения, которые могут работать на нескольких операционных системах и разрабатываются с помощью таких технологий, как React Native или Flutter; веб-приложения, которые загружаются через интернет-браузер и не устанавливаются на устройство как отдельное приложение.

В зависимости от предоставляемых функций мобильные приложения также могут быть разделены на несколько категорий, таких как игры, социальные сети, образовательные приложения, бизнес-приложения, приложения для путешествий, инструменты для работы и личной продуктивности, а также приложения для онлайн-шоппинга. Каждая из этих категорий предлагает пользователям различные виды взаимодействия и сервисов, что делает мобильные приложения неотъемлемой частью повседневной жизни многих людей.

Такое разнообразие и функциональность мобильных приложений делает их ценным инструментом в сфере образования, где они могут служить для организации учебного материала, управления временем и взаимодействия между учащимися и образовательными учреждениями. Развитие мобильных технологий способствует трансформации образовательного процесса, делая его более доступным, интерактивным и персонализированным.

1.2. Сравнительный анализ мобильных приложений

Сравнительный анализ мобильных приложений для организации учебных материалов и составления расписания репетиторов включает в себя оценку основных функциональных возможностей, интерфейса пользователя, адаптивности к различным устройствам и операционным системам, а также удобства использования и доступности дополнительных сервисов. Основными критериями анализа являются: функциональность, удобство интерфейса, мультиплатформенность, интеграция с другими сервисами и стоимость.

В данном анализе основное внимание уделено популярным приложениям для изучения языков и обучения с использованием карточек, таким как Duolingo, Quizlet, Anki и Memrise. Duolingo выделяется своей игровой моделью обучения, позволяющей изучать множество языков в интерактивной форме. Quizlet предлагает широкий спектр инструментов для создания образовательных карточек и викторин, что делает его полезным для запоминания и повторения материала. Anki, известное своим методом интервальных повторений, помогает пользователю улучшать запоминание информации на долгий срок, а Memrise использует уникальный подход с включением видео от носителей языка для улучшения процесса изучения [2].

Каждое из этих приложений предоставляет различные подходы к образованию, и их выбор зависит от специфических потребностей пользователя. Например, для тех, кто ищет более структурированный и игровой подход к изучению языков, Duolingo будет предпочтительнее, в то

время как для серьёзных академических занятий подойдет Anki с его научно обоснованным методом интервальных повторений.

Проведя анализ, стоит упомянуть их общие недостатки, которые могут повлиять на выбор пользователя:

1. Несмотря на широкий спектр функций, многие приложения не предлагают достаточно глубоких настроек персонализации под индивидуальные предпочтения пользователя, что может снизить их образовательную эффективность.

2. Большинство приложений требует постоянного подключения к интернету для доступа ко всем функциям, что ограничивает их использование в условиях с ограниченным доступом к сети.

3. Многие бесплатные версии приложений содержат рекламу, которая может отвлекать от обучения и снижать концентрацию пользователя.

4. Полный доступ к функциям некоторых приложений возможен только при оформлении подписки, стоимость которой может быть довольно высока, особенно для студентов или учащихся.

Учитывая эти недостатки, выбор конкретного приложения должен базироваться на тщательном анализе его возможностей и ограничений в контексте специфических учебных или образовательных потребностей пользователя.

1.3. Обзор языков программирования для мобильных приложений

Разработка мобильных приложений часто опирается на использование объектно-ориентированных языков программирования, которые обеспечивают высокую модульность, переиспользуемость кода и легкость в поддержке. Среди наиболее популярных языков, используемых в мобильной разработке, можно выделить Java, Swift, Kotlin и Dart. Java долгое время был стандартом для разработки приложений под Android благодаря своей надежности и широкой поддержке [3]. Swift, являясь продуктом компании Apple, оптимизирован для разработки приложений под iOS, предлагая улучшенные возможности по сравнению с его предшественником Objective-C

[4]. Kotlin, новый язык от JetBrains, используется для Android-разработки и считается более безопасным и современным вариантом по сравнению с Java.

В контексте данной работы особое внимание уделяется языку Dart, который лежит в основе фреймворка Flutter от Google. Dart — это объектно-ориентированный язык программирования, который был специально адаптирован для создания высокопроизводительных мобильных и веб-приложений. Особенности Dart, такие как компиляция в нативный код, горячая перезагрузка и обширная система виджетов, делают его особенно привлекательным для разработки кроссплатформенных приложений с помощью Flutter. Это позволяет разработчикам создавать высококачественные интерфейсы с нативной производительностью на iOS и Android из единой кодовой базы [5].

Использование Flutter и Dart в разработке мобильных приложений обусловлено рядом преимуществ, включая сокращение времени и ресурсов, необходимых для разработки, а также упрощение процесса тестирования и поддержки приложений. Flutter предлагает богатый набор предварительно сконфигурированных виджетов, которые значительно ускоряют процесс разработки и улучшают визуальную составляющую приложений.

В заключение, анализ объектно-ориентированных языков программирования, применяемых в мобильной разработке, показывает, что выбор языка и технологического стека сильно зависит от специфических требований к приложению, целевой платформы и предпочтений разработчика. Dart в сочетании с Flutter выделяется как особенно мощное средство для создания кроссплатформенных мобильных приложений, сочетающее в себе производительность, эффективность и гибкость.

1.4. Описание основных этапов и методологий разработки мобильных приложений

Разработка мобильных приложений — это комплексный процесс, включающий в себя несколько ключевых этапов: планирование, проектирование, разработку, тестирование и развертывание. На каждом из

этих этапов используются различные технологии и методологии, обеспечивающие создание качественного конечного продукта.

На этапе планирования осуществляется анализ требований, определение целевой аудитории и формулировка функциональных и нефункциональных требований к приложению. Здесь же происходит выбор подходящей платформы (iOS, Android или кроссплатформенные решения) и технологического стека.

Проектирование включает в себя разработку архитектуры приложения и пользовательского интерфейса. На этом этапе создаются макеты экранов и навигационная схема, что помогает визуализировать конечный продукт. Для проектирования интерфейса часто используются инструменты, такие как Sketch, Adobe XD или Figma, которые позволяют дизайнерам создавать интерактивные прототипы.

Этап разработки включает непосредственное программирование приложения. В зависимости от выбранной платформы могут использоваться языки программирования Swift или Kotlin для нативной разработки, а также JavaScript или Dart в случае кроссплатформенных решений, таких как React Native или Flutter. В этот период также активно используются системы контроля версий, такие как Git, для отслеживания изменений в коде и сотрудничества в команде [6].

Таким образом, процесс разработки мобильного приложения является многоэтапным и требует комплексного подхода к каждой детали проекта, начиная от идеи и заканчивая тестированием и доработками. В нашем случае, завершающий этап не включает публикацию приложения в магазинах приложений, а ориентирован на демонстрацию работоспособности и исполнения функциональных требований в контролируемой среде. После завершения тестирования приложение может быть представлено заинтересованным сторонам для оценки, но дальнейшее его развитие и поддержка будут зависеть от полученной обратной связи и потребностей заказчика.

1.5. Рассмотрение интегрированных визуальных сред разработки

Интегрированные визуальные среды разработки (Integrated Development Environments, IDEs) являются ключевым инструментом для разработчиков мобильных приложений, предоставляя функции для написания, тестирования и отладки кода. В разработке мобильных приложений выделяются несколько основных IDE: Visual Studio Code (VS Code), Android Studio, Xcode и IntelliJ IDEA [7].

VS Code, разработанный Microsoft, является лёгким, но мощным редактором кода, поддерживающим множество языков программирования и платформ. Его гибкость и настраиваемость через плагины делают его популярным выбором среди разработчиков кроссплатформенных приложений.

Android Studio, официальная IDE от Google для разработки на Android, предлагает комплексные инструменты для создания, тестирования и профилирования приложений. Она включает встроенный эмулятор, который позволяет тестировать приложения без использования реального устройства.

Xcode — это IDE от Apple, предназначенная для разработки приложений под iOS и macOS. Xcode предоставляет все необходимые инструменты для разработки под устройства Apple, включая симуляторы различных устройств iOS и интегрированный интерфейс для проектирования пользовательских интерфейсов.

IntelliJ IDEA от JetBrains также заслуживает внимания как мощное средство для разработки, особенно с плагином Android, который превращает её в полнофункциональную среду для Android-разработки. IntelliJ IDEA известна своей эффективностью в управлении кодом и интеграцией с различными системами сборки и версионного контроля [8].

Использование различных IDE в зависимости от платформы и предпочтений разработчика позволяет создавать мобильные приложения, оптимизированные для конкретных устройств и операционных систем. Это сочетание инструментов позволяет разработчикам эффективно

реализовывать, тестировать и оптимизировать приложения, улучшая их качество и производительность.

Выводы по главе 1

В заключение, анализ всех аспектов разработки мобильного приложения для организации учебных материалов и составления расписания репетиторов показывает, что успешная реализация проекта зависит не только от технологического выбора, но и от глубокого понимания предметной области и потребностей пользователей. Важность мобильных приложений в образовательной сфере, обсужденная в первом разделе, подчеркивает их роль в улучшении доступности и качества обучения. Сравнительный анализ различных приложений выявил необходимость интеграции успешных функциональных возможностей в разрабатываемое приложение, что подтверждает важность тщательного отбора функций для включения. Выбор Flutter и Dart, как показано в анализе объектно-ориентированных языков и технологий, обоснован их способностью к созданию производительных и мультиплатформенных решений, которые могут значительно оптимизировать процесс разработки и обеспечить высокую скорость работы приложения. Освещение возможностей интегрированных сред разработки, таких как Visual Studio Code и Android Studio, дополнительно подтверждает выбор инструментов, способствующих упрощению и оптимизации разработки.

Таким образом, объединяя анализированные аспекты, проект ориентирован на использование платформы Flutter и языка программирования Dart. Это не только соответствует техническим требованиям проекта, но и обеспечивает его кроссплатформенность, удобство интерфейса и возможность интеграции с внешними системами.

ГЛАВА 2. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ОРГАНИЗАЦИИ УЧЕБНЫХ МАТЕРИАЛОВ И СОСТАВЛЕНИЯ РАСПИСАНИЯ РЕПЕТИТОРОВ

2.1. Проектирование программного продукта

Раздел проектирования программного продукта подробно описывает техническую архитектуру и основные компоненты мобильного приложения для организации учебных материалов, и составления расписания репетиторов. Этот раздел курсового проекта разъясняет, как различные элементы приложения взаимодействуют между собой для обеспечения его функциональности и пользовательской эффективности.

Проект разработан с использованием фреймворка Flutter, который позволяет создавать кроссплатформенные мобильные приложения. Это решение было выбрано благодаря его высокой производительности, гибкости и широкой поддержке сообщества разработчиков. Flutter использует язык программирования Dart, который обладает современными возможностями, такими как асинхронное программирование, что критически важно для разработки отзывчивых пользовательских интерфейсов.

Клиент-серверная архитектура — Приложение разделено на клиентскую и серверную части. Серверная часть обрабатывает запросы, управляет базой данных и выполняет аутентификацию пользователей, используя Node.js и Express.js для обработки API запросов. Данные хранятся в базе данных MySQL, что обеспечивает их доступность и масштабируемость.

Основные компоненты приложения включают в себя:

Экраны приложения. Различные экраны (главная страница, экран блока, группы и т.д.) представляют пользовательский интерфейс, через который пользователи взаимодействуют с функционалом приложения.

Классы данных. Определяют структуру данных приложения, такие как пользователи, блоки, карточки и группы. Эти классы используются для сериализации и десериализации данных, а также для обеспечения взаимодействия с API.

Сервисы для работы с API. Отвечают за отправку и прием данных от сервера. Эти сервисы обрабатывают запросы к серверу и интегрируют полученные данные в логику приложения.

2.2. Используемые библиотеки и зависимости

Клиентская часть приложения использует следующие библиотеки и зависимости:

— Provider для управления состоянием и зависимостями между компонентами.

— HTTP для выполнения сетевых запросов к API. Для выполнения сетевых запросов к API и обработки данных используется протокол HTTP, который является стандартом в обмене данными между клиентом и сервером в интернете. В нашем проекте HTTP-запросы реализованы с помощью пакета http, который предоставляет простой и эффективный способ для отправки данных на сервер и получения ответов от него. Протокол HTTP позволяет разрабатывать структурированные запросы, включающие методы как GET, POST, PUT и DELETE, что соответствует различным операциям CRUD (создание, чтение, обновление, удаление данных) [9].

— Animate_do и flutter_svg для анимации и работы с графикой. Для визуальной привлекательности и плавности интерфейса, были использованы библиотеки animate_do и flutter_svg. Библиотека animate_do обеспечивает простую реализацию множества анимаций, таких как отскоки, затухания, скольжения и другие, что позволяет значительно улучшить визуальное восприятие приложения. Параллельно, flutter_svg используется для встраивания и отображения векторных изображений формата SVG, что гарантирует высокое качество графики на всех типах устройств, обеспечивая четкость и масштабируемость без потери качества.

— Swipe_cards для интерактивных элементов пользовательского интерфейса. Этот пакет обеспечивает функциональность, позволяющую пользователям перелистывать карточки с учебными материалами с помощью жестов «свайпа», что делает процесс обучения более динамичным и

вовлекающим. swipe_cards активно используется в модуле для репетиции материалов, где пользователи просматривают вопросы и ответы, выбирая «знаю» или «не знаю» путем свайпа вправо или влево. Такой подход упрощает интеракцию с приложением и способствует глубокому запоминанию информации за счет повторения материала в интерактивном формате. В техническом плане использование swipe_cards потребовало настройки параметров для оптимизации пользовательского интерфейса и обеспечения плавности анимаций свайпа. Карточки были настроены для легкого перелистывания, с добавлением функции автоматической загрузки новых карточек по мере продвижения пользователя через учебный блок. Интеграция swipe_cards значительно повысила удобство использования приложения, делая процесс обучения не только эффективным, но и интересным, и привнося элемент игровой механики, что особенно ценно для современных пользователей мобильных приложений.

— Локализация и интернационализация поддерживаются через flutter_localizations. Пакет flutter_localizations предоставляет необходимые инструменты и библиотеки для добавления поддержки различных языков в приложение. Это включает в себя стандартные переведённые строки и форматы для множества языков, что позволяет приложению автоматически адаптировать тексты интерфейса, форматы дат, времени и чисел в соответствии с локалью устройства пользователя. Применение flutter_localizations начинается с активации пакета в файле конфигурации проекта pubspec.yaml, после чего разработчик может использовать предоставленные классы и методы для инициализации и управления языковыми настройками.

В этом разделе курсового проекта представлена архитектура мобильного приложения, разработанного на базе фреймворка Flutter. Приложение характеризуется прямым использованием стандартных компонентов и классов Flutter, что обеспечивает гибкость, упрощение

поддержки и ускорение разработки благодаря отсутствию сложной иерархии классов.

2.3. Архитектура программы: иерархия классов, интерфейсы и взаимодействие

Корнем приложения является класс MaterialApp, который управляет ключевыми аспектами приложения, такими как маршрутизация, локализация и тематизация. Маршруты в приложении связывают различные экраны с их функциональными точками доступа:

— HomePageScreen — главный экран, служащий центральным узлом для доступа к основным функциям.

— LogInPage и SignUpPage — экраны для аутентификации и регистрации пользователей.

— SettingsPageScreen — экран настроек, предоставляющий возможности кастомизации приложения.

— CoursePage, CreateCardPage, CreateCoursePage — экраны для управления курсами и учебными материалами.

— GroupPage и SessionPage — новые экраны, предназначенные для управления группами и организации занятий соответственно, что позволяет улучшить взаимодействие и координацию между пользователями в учебном процессе.

Состояние приложения управляется с помощью паттерна Provider, обеспечивающего внедрение зависимостей и реактивное обновление UI в ответ на изменения в данных. Этот метод упрощает разделение логики управления состоянием и пользовательского интерфейса, повышая модульность и упрощая масштабирование приложения.

Приложение интегрировано с различными внешними сервисами и API, что позволяет обеспечивать разнообразные функциональные возможности, включая аутентификацию, управление данными и взаимодействие с пользователем. Стандартные и сторонние библиотеки Flutter используются для работы с сетью, базами данных и элементами интерфейса, что

способствует повышению производительности и удобства использования приложения. Архитектурный подход, основанный на модульности и использовании стандартных решений Flutter, делает приложение легко адаптируемым и масштабируемым. Простота архитектуры в сочетании с эффективным управлением состоянием и взаимодействием с внешними сервисами обеспечивает высокую производительность и удобство использования, делая приложение удобным для разработчиков.

2.4. UML диаграммы

Для наглядного представления структуры и взаимодействий в приложении разработаны следующие UML-диаграммы:

Диаграмма классов показывает структуру основных классов данных (User, Block, Card, Group, Session) и их взаимосвязи.

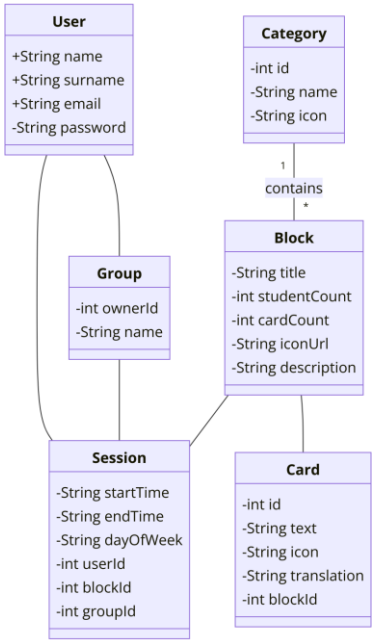

Рис. 2.1. Диаграмма классов.

Диаграмма вариантов использования иллюстрирует функциональные возможности приложения, такие как создание и прохождение блоков, управление группами и расписаниями.

Рис. 2.2. Диаграмма вариантов использования.

Диаграмма активности описывает процессы создания блоков, регистрации в группах и управления учебным процессом.

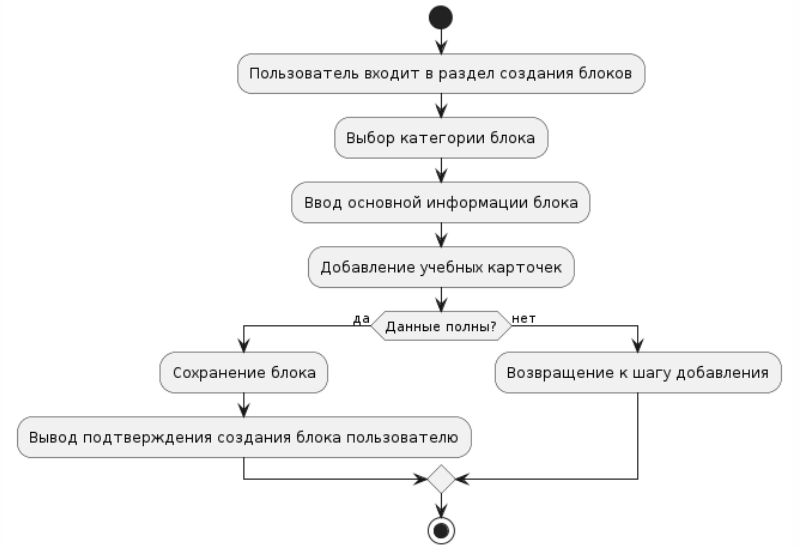

Рис. 2.3. Диаграмма процесса создания учебных блоков.

Диаграмма активности изображает процесс создания учебных блоков в рамках мобильного приложения, предназначенного для организации образовательного процесса. Процесс начинается, когда пользователь входит в специализированный раздел для создания блоков, после чего осуществляется выбор категории, что предшествует вводу основной информации блока, включая его название и описание. Следующий шаг предусматривает добавление учебных карточек, необходимых для составления блока. В случае, если данные введены не полностью, процесс предписывает возвращение к шагу добавления для их завершения. По

завершении добавления карточек блок сохраняется, и пользователь получает подтверждение о создании блока. Диаграмма детально отображает последовательность шагов и действий, подчеркивая итеративный характер процесса создания блока и важность полноты данных для успешного его завершения.

Рис. 2.4. Диаграмма процесса регистрации пользователя в группу.

На второй диаграмме активности изображён процесс регистрации пользователя в группах внутри образовательного приложения. Пользовательский путь начинается с перехода в раздел групп, где он может просмотреть список доступных групп для вступления. После выбора подходящей группы пользователь отправляет запрос на вступление, который включает в себя необходимую информацию для рассмотрения кандидатуры. Затем следует этап ожидания, в течение которого руководитель группы рассматривает запрос и выносит решение. Процесс завершается получением пользователем уведомления о результатах рассмотрения его запроса. Эта диаграмма отражает линейный поток действий, предпринимаемых для включения в образовательное сообщество, подчёркивая значимость каждого этапа в формировании учебной группы.

Рис. 2.5. Диаграмма процесса назначения занятия.

Диаграмма активности, представленная на рисунке, иллюстрирует последовательность действий, которые выполняет преподаватель или руководитель группы для управления учебным процессом в образовательном приложении. Процесс начинается с этапа планирования учебного расписания, где определяются временные рамки и содержание предстоящих занятий. Затем следует назначение конкретного занятия, что включает выбор темы, материалов для изучения и задач для выполнения. Последним шагом является отслеживание прогресса учащихся и анализ результатов, что позволяет преподавателю оценивать эффективность обучения и при необходимости вносить корректировки в учебный план. Эта диаграмма ясно демонстрирует ключевые этапы в администрировании учебного процесса, подчеркивая роль преподавателя как координатора образовательной деятельности.

2.5. Описание и разработка программного продукта

В рамках курсового проекта выполнена разработка мобильного приложения, предназначенного для организации учебных материалов и составления расписания репетиторов. Проект реализован с целью упрощения и оптимизации учебного процесса как для индивидуального обучения, так и для групповых занятий. Основная задача приложения — предоставление эффективного и интуитивно понятного интерфейса для управления учебными блоками, расписанием, а также взаимодействия между учащимися и преподавателями.

Функциональные возможности приложения:

— **Создание учебных блоков.** Приложение позволяет пользователям создавать учебные блоки на различные академические темы. Каждый блок может содержать множество карточек, которые включают теоретический материал, практические задания или вопросы для самопроверки. Это позволяет преподавателям систематизировать учебный материал и делать его доступным для студентов в удобной и структурированной форме.

— **Интерактивное прохождение блоков.** Пользователи могут самостоятельно изучать материалы или принимать участие в групповых обучающих сессиях под руководством репетиторов. Интерактивные элементы, такие как тесты и квизы, интегрированы непосредственно в учебные блоки, что способствует активному взаимодействию учащихся с материалом и повышает их мотивацию.

— **Организация групп.** Преподаватели или учащиеся могут создавать группы для совместного прохождения курсов. Эта функция включает возможность планирования групповых занятий, обсуждений и совместной работы над проектами, что значительно улучшает процесс обучения и делает его более взаимодейственным и социально ориентированным.

— **Расписание занятий.** Интеграция календаря в приложение позволяет репетиторам и студентам легко планировать, отслеживать и изменять расписание занятий. Это обеспечивает чёткую организацию учебного процесса и позволяет всем участникам быть в курсе предстоящих событий.

Приложение разработано с акцентом на удобство и простоту использования, что делает его доступным для студентов и преподавателей всех возрастных категорий. Интуитивно понятный интерфейс, возможность персонализации учебного процесса. Приложение также поддерживает

синхронизацию данных между устройствами, что позволяет пользователям доступ к учебным материалам с любого устройства и в любое время.

Разработанное мобильное приложение является комплексным решением для обучения и управления учебными ресурсами, предлагая различные функциональные возможности.

Разработка мобильного приложения осуществлялась на платформе Flutter, что позволило использовать единый кодовый базис для создания кроссплатформенного приложения, доступного как на Android, так и на iOS. Flutter выбран ввиду его высокой производительности, обширной поддержки анимаций и интерактивных элементов, а также удобства в построении сложных пользовательских интерфейсов с консистентным визуальным стилем.

Для управления состоянием приложения использовалась библиотека Provider, которая позволяет эффективно управлять состоянием в реактивном стиле, обеспечивая надёжную передачу данных между компонентами приложения. Сетевые запросы реализованы с помощью пакета http, который поддерживает как базовую работу с HTTP-запросами, так и более сложные сценарии обработки данных.

Пользовательский интерфейс приложения был разработан с использованием мощных инструментов и библиотек Flutter. Использование виджетов «MaterialApp» позволило создать нативно выглядящий интерфейс для основных мобильных платформ. Анимации были реализованы с использованием пакета «animate_do», что позволило сделать интерактивные элементы более живыми и отзывчивыми.

В данном разделе приводится описание нескольких ключевых экранов и классов данных мобильного приложения, предназначенного для организации учебных материалов и составления расписания репетиторов.

2.6. Описание экранов приложения

На примере экранов HomePageScreen, LoginPageScreen, CardPageScreen и CoursePageScreen рассмотрим, как устроены экраны мобильного

приложения. Описание каждого экрана поможет понять, как пользователи взаимодействуют с приложением и какие задачи могут быть выполнены в каждом из разделов.

Главный экран (HomePageScreen) служит центральным узлом для доступа к основным функциям приложения. Этот экран разделён на три части: верхняя отображает информацию о пользователе, центральная часть представляет список блоков, которые пользователь ещё не завершил, а также список блоков от сообщества. Нижняя навигационная панель, включающая элементы навигации для доступа к курсам, карточкам и группам через виджет «BottomNavigationBar», обеспечивает удобное и интуитивно понятное переключение между разделами. На рисунке 2.1 представлен код виджета, отображающего навигационную панель, где вложенный виджет «BottomNavigationBarItem» представляет кнопку на панели.

```
bottomNavigationBar: BottomNavigationBar(
  items: const <BottomNavigationBarItem>[
    BottomNavigationBarItem(
      icon: Icon(Icons.home),
      label: 'Home',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon(Icons.add_circle_outline),
      label: 'Create',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon(Icons.school),
      label: 'Education',
    ), // BottomNavigationBarItem
  ], // <BottomNavigationBarItem>[]
  currentIndex: _selectedIndex,
  onTap: _onItemTapped,
  selectedItemColor: Color.fromARGB(255, 255, 255, 255),
), // BottomNavigationBar
```

Рис. 2.6. Виджет «BottomNavigationBarItem».

Экран входа (LoginPageScreen) обеспечивает функции аутентификации пользователя с помощью формы, включающей поля для ввода логина и пароля. На этом экране расположены два поля ввода: «Email» и «Пароль». На рисунке 2.2 представлены фрагменты кода, отображающие эти поля. Они необходимы для ввода данных, которые затем отправляются на сервер посредством API. Для усиления безопасности применяются механизмы

валидации введённых данных, а также осуществляется интеграция с серверными API для проверки учётных данных.

```
child: Column(
  children: <Widget>[
    Container(
      padding: EdgeInsets.all(10),
      decoration: BoxDecoration(
          border: Border(
              bottom: BorderSide(
                  color: Colors.grey.shade300))), // BorderSide // Border // BoxDecoration
      child: TextField(
        decoration: InputDecoration(
            hintText: "Email",
            hintStyle: TextStyle(
                color: Color.fromARGB(255, 75, 91, 99)), // TextStyle
            border: InputBorder.none), // InputDecoration
      ), // TextField
    ), // Container
    Container(
      padding: EdgeInsets.all(10),
      decoration: BoxDecoration(
          border: Border(
              bottom: BorderSide(
                  color: Colors.grey.shade200))), // BorderSide // Border // BoxDecoration
      child: PasswordTextField(
        label: 'Пароль',
      ), // PasswordTextField
    ), // Container
  ], // <Widget>[]
), // Column
```

Рис. 2.7. Поля ввода данных .

Экраны карточек (CardPageScreen) предназначен для изучения информации по карточкам. CardPageScreen позволяет пользователю просматривать и проходить карточки с учебным материалом. На рисунке 2.3 представлен виджет «Expanded», который содержит потомка «SwipeCards». Этот виджет, взятый из пакета «swipe_cards», реализует функцию смахивания карточки, при котором смахивание влево откладывает карточку, а вправо сообщает программе, что карточка с информацией выучена.

```
Expanded(
  child: SwipeCards(
    matchEngine: matchEngine,
    itemBuilder: (BuildContext context, int index) {
      return Card(
        child: Padding(
          padding: const EdgeInsets.all(24.0),
          child: _cardInfo(context, course[index]),
        ), // Padding
      ); // Card
    },
    onStackFinished: () {
      print('Stack Finished');
      setState(() {
        _showButton = true;
      });
    },
  ), // SwipeCards
), // Expanded
```

Рис. 2.8. Виджет «SwipeCards».

Экран курсов (CoursePageScreen), представленный на рисунке 2.13, позволяет ознакомиться с блоком. Этот экран состоит из различных

компонентов: верхняя часть предоставляет информацию, такую как название блока, количество карточек в блоке и количество зачисленных студентов. В центральной части находится описание блока, а в нижней части — кнопка для поступления на курс. Эти экраны поддерживают структурированное изучение тем и облегчают процесс обучения за счёт геймификации. Кроме того, для улучшения пользовательского опыта, каждый элемент интерфейса оптимизирован для различных устройств, будь то смартфоны, планшеты или настольные компьютеры. Дизайн экрана также включает адаптивные шрифты и цветовые схемы, которые повышают читаемость и визуальную привлекательность.

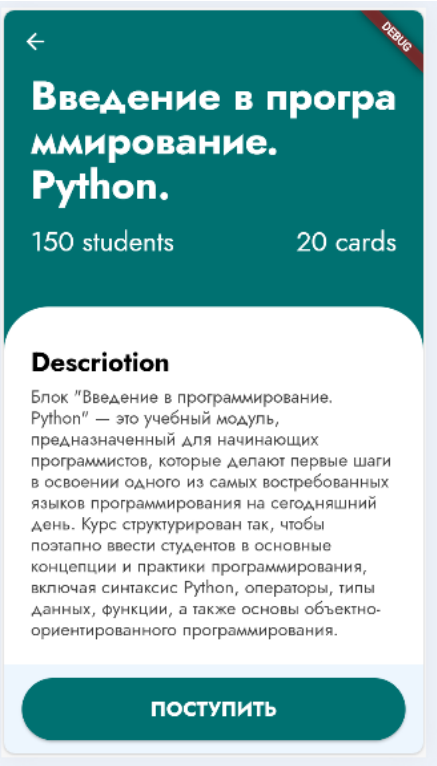

Рис. 2.9. Экран CoursePageScreen.

Это описание экранов приложения демонстрирует, как тщательно продуманный интерфейс может способствовать улучшению образовательного процесса, делая его более интерактивным и доступным для пользователей всех уровней.

2.7. Описание классов данных

В данном разделе представлены основные классы данных, используемые в приложении на языке Dart. Рассмотрим структуру и функциональность некоторых ключевых классов:

Класс «Card» представляет собой учебную карточку, которая содержит информацию, необходимую для запоминания определённого материала. В его структуру входят следующие поля:

1. **Идентификатор (ID).** Уникальный идентификатор карточки для её идентификации в системе.
2. **Название.** Название карточки, обозначающее тему или содержание материала.
3. **Описание.** Дополнительное описание или пояснение к содержанию карточки.
4. **Содержимое.** Текстовое или мультимедийное содержимое карточки, которое предназначено для изучения или запоминания.

```dart
// Класс "Card"
class Card {
  final int id;
  final String title;
  final String description;
  final String content;

  Card({
    required this.id,
    required this.title,
    required this.description,
    required this.content,
  });
}
```

Рис. 2.10. Класс Card.

Класс «Block» представляет собой учебный курс, который состоит из нескольких учебных карточек. Его основные поля включают:

1. **Идентификатор курса (ID).** Уникальный идентификатор курса, позволяющий однозначно его идентифицировать.
2. **Название.** Название курса, которое описывает его тему или предметную область.
3. **Описание.** Краткое описание курса, содержащее основную информацию о его содержании и целях.

4. **Список карточек.** Коллекция учебных карточек, входящих в состав курса, которая позволяет организовать и структурировать учебный материал.

```
// Класс "Block"
class Block {
  final int id;
  final String title;
  final String description;
  final List<Card> cards;

  Block({
    required this.id,
    required this.title,
    required this.description,
    required this.cards,
  });
}
```

Рис. 2.11. Класс Block.

Класс «Lesson» представляет собой сущность, которая инкапсулирует информацию о занятии в рамках учебного процесса. Он содержит следующие поля:

1. **Время начала и окончания.** Точное время начала и завершения занятия.

2. **День недели.** День недели, на который запланировано проведение занятия.

3. **Идентификаторы пользователя, учебного блока и группы.** Идентификаторы, которые связывают занятие с конкретным пользователем, учебным блоком и учебной группой.

```
// Класс "Lesson"
class Lesson {
  final DateTime startTime;
  final DateTime endTime;
  final String dayOfWeek;
  final int userId;
  final int blockId;
  final int groupId;

  Lesson({
    required this.startTime,
    required this.endTime,
    required this.dayOfWeek,
    required this.userId,
    required this.blockId,
    required this.groupId,
  });
}
```

Рис. 2.12. Класс Lesson.

Эти классы обеспечивают структурирование и управление учебным материалом, позволяя эффективно организовывать учебный процесс и облегчать взаимодействие между пользователями и учебным контентом.

2.8. Описание системы автоматической сборки проекта

В качестве системы автоматической сборки проекта в данном приложении используется инструмент Flutter SDK, который включает в себя собственный механизм сборки и управления зависимостями. Flutter SDK обеспечивает полноценную среду разработки для создания кроссплатформенных мобильных приложений с использованием языка программирования Dart.

Основные особенности системы сборки в Flutter включают в себя следующее:

1. Flutter CLI (Command Line Interface). Интерфейс командной строки Flutter предоставляет разработчикам удобные инструменты для управления проектами, сборки приложений и выполнения различных задач, таких как добавление новых пакетов, запуск и тестирование приложений [10].

2. Dart и Flutter Packages. Для управления зависимостями в проекте используется файл `pubspec.yaml`, в котором указываются все используемые пакеты Dart и Flutter. После добавления или изменения зависимостей проекта необходимо запустить команду «flutter pub get», чтобы обновить локальные зависимости [11].

3. Средства отладки и профилирования. Flutter SDK предоставляет инструменты для отладки и профилирования приложений, такие как Flutter DevTools, которые позволяют анализировать производительность приложения, выявлять и исправлять ошибки и улучшать пользовательский опыт.

4. Интеграция с средами разработки. Flutter поддерживает интеграцию с различными средами разработки, такими как Android Studio, Visual Studio Code и IntelliJ IDEA. Эти интегрированные среды разработки предоставляют дополнительные инструменты и функции для удобства разработки и отладки.

5. Кроссплатформенная сборка. Система сборки Flutter позволяет создавать кроссплатформенные приложения для Android и iOS с использованием единого кодовой базы. При сборке проекта для конкретной платформы Flutter автоматически оптимизирует и собирает приложение, генерируя необходимые бинарные файлы для запуска на целевой платформе.

Использование Flutter SDK в проекте обеспечивает удобную и эффективную среду разработки, позволяя разработчикам создавать высококачественные кроссплатформенные мобильные приложения с минимальными затратами времени и ресурсов.

Процесс сборки программного проекта является одним из ключевых этапов разработки, направленных на превращение исходного кода в исполняемые файлы, готовые к развёртыванию и использованию конечными пользователями. В данном разделе рассматривается процесс сборки приложения, реализованного с использованием Flutter SDK.

Компиляция исходного кода в Flutter проекте выполняется с помощью команды «flutter build», которая преобразует код на языке Dart в исполняемые файлы для целевой платформы. Примеры команд сборки:

1. Команда «flutter build apk» - сборка APK-файла для установки на устройства с операционной системой Android.
2. Команда «flutter build ios» - сборка приложения для устройств с операционной системой iOS.

После компиляции исходного кода применяются различные оптимизации для уменьшения размера и повышения производительности приложения. В рамках Flutter SDK автоматически применяются такие оптимизации, как удаление неиспользуемого кода, минификация ресурсов и оптимизация графических элементов.

Перед публикацией в магазине приложений каждый исполняемый файл подписывается цифровой подписью, обеспечивающей его целостность и подтверждающей авторство. Для сборки подписанной версии приложения в

Flutter используются команды с ключами, содержащими информацию о ключе подписи и другие параметры.

После сборки приложения проводится тестирование на соответствие требованиям функциональности и дизайна. Для отладки приложения можно использовать инструменты, предоставляемые Flutter SDK, такие как Flutter DevTools, который позволяет анализировать производительность и ошибки приложения.

После успешного завершения всех предыдущих этапов приложение готово для публикации и распространения среди конечных пользователей. Для этого разработчики могут загрузить приложение в магазины приложений (Google Play Store, Apple App Store), развернуть его на веб-сервере или распространить среди ограниченной аудитории через другие каналы.

Для упрощения и автоматизации процесса сборки приложения разработчики могут использовать специализированные инструменты и сценарии сборки. Например, можно настроить непрерывную интеграцию и доставку с использованием таких инструментов, как Jenkins, Fastlane или GitHub Actions, чтобы автоматически выполнять сборку, тестирование и публикацию приложения при каждом обновлении исходного кода.

В рамках курсового проекта предполагается выполнение только базового этапа сборки приложения без последующих действий, таких как подпись, публикация в магазины приложений или автоматизация процесса сборки.

Таким образом, выполнение сборки проекта в рамках курсового проекта позволит демонстрировать основные навыки разработки мобильных приложений с использованием Flutter SDK.

Выводы по главе 2

В заключение, анализ практической реализации мобильного приложения для организации учебных материалов и составления расписания занятий демонстрирует важность правильного выбора архитектуры программы и технологических решений. Глава 2 предоставила обзор

ключевых компонентов приложения, включая его структуру, основные экраны и классы данных. Рассмотрены основные принципы проектирования и взаимодействия между компонентами приложения, что является ключевым для обеспечения его функциональности и эффективности.

Анализ структуры приложения позволяет увидеть, как каждый его компонент взаимодействует с другими, что способствует пониманию его работы и потенциальных улучшений. Важно отметить, что разработка приложения в рамках данного курсового проекта была сосредоточена на реализации основной функциональности, такой как вход в систему, просмотр и создание учебных материалов и управление курсами и группами. Однако, для полноценного использования приложения в реальных условиях, необходимо провести дальнейшую работу по его доработке, тестированию и оптимизации.

Исходя из анализа, можно заключить, что успешная реализация проекта зависит от грамотного выбора технологий, архитектуры и методов разработки. Дальнейшее развитие приложения предполагает интеграцию новых функциональных возможностей, улучшение пользовательского опыта и обеспечение его стабильной работы на различных устройствах.

ЗАКЛЮЧЕНИЕ

В заключение курсового проекта по разработке мобильного приложения для организации учебных материалов и составления расписания репетиторов можно подчеркнуть, что успешная реализация такого рода проектов требует комплексного подхода. Научный анализ всех этапов разработки, начиная от теоретического обоснования выбора платформы и программных инструментов до практической реализации и тестирования программного продукта, позволяет выявить наиболее эффективные решения и подходы.

Использование кроссплатформенного фреймворка Flutter и языка программирования Dart в проекте оправдано их способностью к созданию производительных и функционально богатых приложений, что существенно оптимизирует процесс разработки и поддержки. Рассмотрение различных аспектов мобильных приложений, включая классификацию, методологии разработки и интеграцию с внешними сервисами, подтверждает выбор технологического стека и обеспечивает гибкость в адаптации приложения под разнообразные пользовательские потребности.

Подведение итогов проведённого сравнительного анализа мобильных приложений аналогичного назначения позволило определить ключевые функциональные возможности, которые были успешно интегрированы в разрабатываемое приложение, учитывая специфические требования и ожидания целевой аудитории. Такой подход способствует созданию качественного продукта, который не только отвечает современным требованиям к мобильным образовательным ресурсам, но и предоставляет пространство для дальнейшего развития и совершенствования.

В результате реализации данного курсового проекта были приобретены ценные навыки и опыт в области проектирования, разработки и анализа мобильных приложений, что является важным вкладом в профессиональное развитие автора в сфере информационных технологий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Statista. Число мобильных приложений в различных магазинах. [Электронный ресурс]. URL: www.statista.com - Дата обращения: 20.04.2024.

2. Сравнительный анализ платформ для дистанционного обучения в образовательной среде [Электронный ресурс] // CyberLeninka. - Режим доступа: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-platform-dlya-distantsionnogo-obucheniya-v-obrazovatelnoy-srede, свободный. - Дата обращения: 21.04.2024.

3. Белоусов А. И., Ткачев С. Б. Java. Методы программирования. – М.: БИНОМ. Лаборатория знаний, 2014. – 912 с.

4. Симонов К. Ю. Разработка приложений на Swift для iOS и macOS. – М.: ДМК Пресс, 2020. – 480 с.

5. Официальная документация Flutter [Электронный ресурс]. – Режим доступа: https://flutter.dev. – Дата обращения: 21.04.2024.

6. AppMaster. Полное руководство по разработке мобильных приложений [Электронный ресурс]. – Режим доступа: https://appmaster.io - Дата обращения: 21.04.2024.

7. AppVerticals. Лучшие интегрированные среды разработки (IDE) для разработки мобильных приложений [Электронный ресурс]. – Режим доступа: https://www.appverticals.com, свободный. – Дата обращения: 22.04.2024.

8. JetBrains. Интегрированные среды разработки JetBrains: наслаждайтесь исключительным опытом разработчика [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/ides/, свободный. – Дата обращения: 22.04.2024.

9. Flutter Documentation. Полное руководство по использованию библиотеки Provider в Flutter для управления состоянием [Электронный ресурс]. – Режим доступа: https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple, свободный. – Дата обращения: 25.04.2024.

10. Официальная документация Flutter [Электронный ресурс]. - Режим доступа: https://flutter.dev/docs, свободный. – Дата обращения: 25.04.2024.

11. Dart Packages. Управление пакетами в Dart и Flutter [Электронный ресурс]. - Режим доступа: https://pub.dev, свободный. – Дата обращения: 25.04.2024.

ПРИЛОЖЕНИЯ

Приложение А

Листинг файла CreateCoursePage.dart

```
import 'package:flutter/material.dart';
import '../../../models/course.dart';
import '../../../models/card.dart';
import '../../data.dart';

class CreateCoursePage extends StatelessWidget {
  final course = CourseData.coursesList;
  var blockNameInputController = "";
  var blockDescriptionInputController = "";
  var cardTextInputController = "";
  var cardAnswerInputController = "";
  var cardCountBlock = 0;
  List<BlockCard> createdCards = [];

  void showCustomDialog(BuildContext context) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return Dialog(
          backgroundColor: Colors.white,
          child: _cardInfo(context),
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.background,
        iconTheme: IconThemeData(color: Colors.white),
        leading: IconButton(
          icon: Icon(Icons.arrow_back, color: Colors.white),
          onPressed: () {
            Navigator.of(context).pushNamed('/');
          },
        ),
```

```
      ),
      body: Container(
        width: double.infinity,
        decoration:
            BoxDecoration(color: Theme.of(context).colorScheme.background),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Padding(
              padding: EdgeInsets.only(top: 0, right: 24, left: 24, bottom:
24),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: <Widget>[
                  Text(
                    "Создание блока",
                    style: Theme.of(context)
                        .textTheme
                        .displayLarge!
                        .copyWith(color: Colors.white, fontSize: 36),
                  ),
                  SizedBox(
                    height: 16,
                  ),
                  Text(
                    "${createdCards.length} cards",
                    style: Theme.of(context).textTheme.titleLarge!.copyWith(
                        color: Colors.white,
                        fontSize: 24,
                        fontWeight: FontWeight.normal),
                  ),
                ],
              ),
            ),
            SizedBox(height: 20),
            Expanded(
              child: Container(
                padding:
                     const EdgeInsets.only(right: 24.0, left: 24.0, top: 24.0),
                decoration: BoxDecoration(
                    color: Colors.white,
                    borderRadius: BorderRadius.only(
```

```
                    topLeft: Radius.circular(40),
                    topRight: Radius.circular(40))),
              child: SingleChildScrollView(
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    _buildField(
                         context, 'Название', "Название курса", "nameField"),
                    _buildField(context, 'Описание',
                        "Введите информацию о блоке", "descField"),
                    // _buildActionButton("Готово", "white", context,
                    //     "white", "createBlock"),
                    _createCards(context)
                  ],
                ),
              ),
            ),
          ),
        ],
      ),
    ),
    resizeToAvoidBottomInset: false,
    bottomNavigationBar: BottomAppBar(
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Expanded(
            child: _buildActionButton(
                "Удалить блок", "red", context, "white", "none"),
          ),
          SizedBox(width: 8),
          Expanded(
            child: _buildActionButton(
                "Сохранить блок", "green", context, "white", "save"),
          ),
        ],
      ),
    ),
  );
}

Widget _cardInfo(BuildContext context) {
```

```
    return Row(
      children: [
        Expanded(
            child: Container(
          height: 240,
          padding: EdgeInsets.symmetric(horizontal: 24),
          decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.all(Radius.circular(40)),
              border:
                  Border.all(color: const Color.fromARGB(255, 255, 255,
255))),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              _buildField(
                  context, "Вопрос, слово, утверждение:", "Текст",
"textCard"),
              _buildField(context, "Ответ:", "Ответ", "answerCard"),
              Row(
                mainAxisAlignment: MainAxisAlignment.end,
                children: [
                  ElevatedButton(
                    onPressed: () {
                      _createCard();
                      cardCountBlock += 1;
                      print("save");
                      print(createdCards.length);
                      print(createdCards.last.text);
                      for (var element in createdCards) {
                        print(element.text);
                      }
                      Navigator.of(context).pop();
                    },
                    child: Text("Сохранить"),
                  ),
                ],
              ),
            ],
          ),
        )),
      ],
```

```
    );
  }

  Widget _buildField(
      BuildContext context, String title, String hintText, String role) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          title,
          textDirection: TextDirection.ltr,
          style: Theme.of(context).textTheme.titleLarge!.copyWith(fontSize:
18),
        ),
        TextField(
          onChanged: (text) {
            if (role == 'nameField') {
              blockNameInputController = text;
            } else if (role == 'descField') {
              blockDescriptionInputController = text;
            } else if (role == 'answerCard') {
              cardAnswerInputController = text;
            } else if (role == 'textCard') {
              cardTextInputController = text;
            }
          },
          style: TextStyle(fontSize: 16),
          decoration: InputDecoration(
              hintText: hintText,
              hintStyle: TextStyle(color: Color.fromARGB(255, 75, 91, 99)),
              border: InputBorder.none),
        ),
      ],
    );
  }

  Widget _createCards(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 12.0),
```

```
          child: Text(
            'Карточки',
            textDirection: TextDirection.ltr,
            style:
                Theme.of(context).textTheme.titleLarge!.copyWith(fontSize:
18),
          ),
        ),
        InkWell(
          onTap: () {
            showCustomDialog(context);
          },
          child: Container(
            height: 64,
            decoration: BoxDecoration(
                border: Border.all(
                  width: 4,
                  color: Color.fromARGB(136, 12, 134, 134),
                ),
                borderRadius: BorderRadius.all(Radius.circular(24))),
            child: Center(
              child: Icon(
                size: 36,
                Icons.add_circle_outline,
                color: Color.fromARGB(136, 12, 134, 134),
              ),
            ),
          ),
        ),
      ],
    );
  }

  Widget _buildActionButton(String title, String bgColor, BuildContext
context,
      String textColor, String buttinAction) {
    Color styleBgColor = Color(0xFF007171);
    Color styleTextColor = Colors.black;

    if (textColor == "white") {
      styleTextColor = Colors.white;
    } else {
```

```
  styleTextColor = Colors.black;
}

if (bgColor == 'red') {
  styleBgColor = const Color.fromARGB(255, 244, 76, 54);
} else if (bgColor == 'green') {
  styleBgColor = const Color.fromARGB(255, 76, 175, 125);
}
return ElevatedButton(
  onPressed: () {
    var lastCourse = CourseData.coursesList.last;
    if (buttinAction == "save") {
      if (blockNameInputController == "" ||
          blockDescriptionInputController == "") {
        // TO DO: дописать всплывающее окно о пустых полях
      } else {
        if (cardCountBlock > 0) {
          CourseData.coursesList.add(Course(
              id: lastCourse.id + 1,
              title: blockNameInputController,
              studentCount: 0,
              cardCount: createdCards.length,
              iconUrl: "none",
              description: blockDescriptionInputController));

          Course newCourse = CourseData.coursesList.last;
          print(newCourse);
          Navigator.of(context).pushNamed("/");
        } else {
          print("Вы не создали карточку");
        }
      }
    }
  },
  style: ElevatedButton.styleFrom(
    elevation: 3,
    shadowColor: styleBgColor,
    minimumSize: Size(
        double.infinity, 68
        ),
    padding: EdgeInsets.symmetric(horizontal: 10),
    backgroundColor: styleBgColor,
```

```
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(24),
        ),
      ),
      child: Text(
        title,
        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.displayMedium!.copyWith(
              color: styleTextColor,
            ),
      ),
    );
  }

  Future<Course> _createCourse() async {
    Course newCourse = Course(
        id: 0,
        title: blockNameInputController,
        studentCount: 0,
        cardCount: cardCountBlock,
        iconUrl: "none",
        description: blockDescriptionInputController);
    CourseData.coursesList.add(newCourse);
    return newCourse;
  }
  Future<void> _createCard() async {
    BlockCard newCard = BlockCard(
        id: 10,
        text: cardTextInputController,
        icon: "icon",
        translation: cardAnswerInputController,
        blockId: 20,
        updatedAt: DateTime.now(),
        createdAt: DateTime.now());
    createdCards.add(newCard);
  }

  Future<void> _createCourseAndCards() async {
    Course course = await _createCourse();
    // await _createCard();
  }
}
```

Листинг файла CreateCardPage.dart

```
import 'package:flutter/material.dart';
import 'package:education_app/features/data.dart';
import '../../../models/card.dart';
import 'package:swipe_cards/swipe_cards.dart';
import '../../data.dart';

class CardPage extends StatefulWidget {
  @override
  State<CardPage> createState() => _CardPageState();
}

class _CardPageState extends State<CardPage>
    with SingleTickerProviderStateMixin {
  late MatchEngine matchEngine;
  bool _showButton = false;
  final course = CourseData.cardsList;

  @override
  void initState() {
    super.initState();

    List<SwipeItem> swipeItems = course.map((item) {
      return SwipeItem(
        content: item,
        likeAction: () {
          print("Выучено: ${item.text}");
        },
        nopeAction: () {
          print("Отложить: ${item.text}");
        },
      );
    }).toList();

    matchEngine = MatchEngine(swipeItems: swipeItems);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
```

```
        backgroundColor: Color.fromARGB(255, 3, 100, 100),
        title: Padding(
          padding: const EdgeInsets.only(left: 10.0),
          child: Text("Course name", style: TextStyle(color: Colors.white)),
        ),
      ),
      backgroundColor: Color.fromARGB(255, 3, 100, 100),
      body: Stack(
        children: <Widget>[
          Padding(
            padding:
                const EdgeInsets.only(bottom: 36, top: 0, left: 24, right:
24),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                Text('Выучено 0 из 20',
                    style: TextStyle(fontSize: 24, color: Colors.white)),
                SizedBox(height: 10),
                Expanded(
                  child: SwipeCards(
                    matchEngine: matchEngine,
                    itemBuilder: (BuildContext context, int index) {
                      return Card(
                        child: Padding(
                          padding: const EdgeInsets.all(24.0),
                          child: _cardInfo(context, course[index]),
                        ),
                      );
                    },
                    onStackFinished: () {
                      print('Stack Finished');
                      setState(() {
                        _showButton = true;
                      });
                    },
                  ),
                ),
                SizedBox(height: 20),
                Row(
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: [
```

```
                    Expanded(
                        child: _buildActionButton('Отложить и повторить',
'red',
                            context, "white", "repeat")),
                    SizedBox(width: 8),
                    Expanded(
                        child: _buildActionButton('Запомнить', 'green',
context,
                            "white", "remember")),
                  ],
                ),
              ],
            ),
          ),
          if (_showButton)
            Positioned(
              child: Container(
                decoration: BoxDecoration(
                  color: Color.fromARGB(255, 3, 100, 100),
                ),
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    Row(),
                    Container(
                      decoration: BoxDecoration(
                        boxShadow: [
                          BoxShadow(
                            color: Colors.grey.withOpacity(0.5),
                            spreadRadius: 3,
                            blurRadius: 6,
                            offset: Offset(0, 3), // changes position of
shadow
                          ),
                        ],
                        borderRadius: BorderRadius.circular(25),
                        color: const Color.fromARGB(255, 255, 255, 255),
                      ),
                      height: 240,
                      margin: EdgeInsets.all(48),
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.spaceAround,
```

```
children: [
  Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Center(
        child: Icon(
          Icons.check,
          size: 50, // размер иконки
          color: Color.fromARGB(
              255, 12, 134, 134), // цвет иконки
        ),
      ),
    ],
  ),
  Text(
    "Пройдено",
    style: TextStyle(
        color: Color.fromARGB(255, 12, 134, 134),
        fontSize: 24,
        fontWeight: FontWeight.bold),
  ),
  Text(
    "Поздравляю! Вы завершили курс.",
    style: TextStyle(
        color: Color.fromARGB(255, 31, 31, 31),
        fontSize: 16),
  ),
  Container(
    padding: EdgeInsets.only(bottom: 24),
    width: 200,
    child: ElevatedButton(
      onPressed: () {
        Navigator.of(context).pushNamed("/");
      },
      style: ElevatedButton.styleFrom(
        padding: EdgeInsets.symmetric(vertical: 10),
        minimumSize: Size(double.infinity, 48),
        backgroundColor:
            Color.fromARGB(255, 12, 134, 134),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(24),
        ),
```

```
                            ),
                            child: Text(
                              "На главную",
                              textAlign: TextAlign.center,
                              style: Theme.of(context)
                                  .textTheme
                                  .displayMedium!
                                  .copyWith(
                                    fontWeight: FontWeight.bold,
                                    color: const Color.fromARGB(
                                        255, 255, 255, 255),
                                  ),
                            ),
                          ),
                        )
                      ],
                    ),
                  )
                ],
              ),
            ),
          ),
        ],
      ),
    );
}

Widget _buildCustomContainer({required String title}) {
  return Column(
    children: [
      Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(24),
          color: Color.fromARGB(255, 122, 122, 122),
        ),
        height: 100,
        width: 150,
      ),
    ],
  );
}
```

```
  Widget _buildActionButton(String title, String bgColor, BuildContext
context,
      String textColor, String actionCard) {
    Color styleBgColor = Color(0xFF007171);
    Color styleTextColor = Colors.black;

    if (textColor == "white") {
      styleTextColor = Colors.white;
    } else {
      styleTextColor = Colors.black;
    }

    if (bgColor == 'red') {
      styleBgColor = const Color.fromARGB(255, 244, 76, 54);
    } else if (bgColor == 'green') {
      styleBgColor = const Color.fromARGB(255, 76, 175, 125);
    }
    return ElevatedButton(
      onPressed: () {
        if (actionCard == "repeat") {
          matchEngine.currentItem?.nope();
        } else if (actionCard == "remember") {
          matchEngine.currentItem?.like();
        } else {}
      },
      style: ElevatedButton.styleFrom(
        elevation: 3,
        shadowColor: styleBgColor,
        padding: EdgeInsets.symmetric(horizontal: 10),
        minimumSize: Size(double.infinity, 68),
        backgroundColor: styleBgColor,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(24),
        ),
      ),
      child: Text(
        title,
        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.displayMedium!.copyWith(
              color: styleTextColor,
            ),
      ),
```

```
  );
}

Widget _cardInfo(BuildContext context, BlockCard card) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.start,
    children: [
      _buildCustomContainer(title: 'Картинка'),
      SizedBox(height: 20),
      Row(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Expanded(
              child: Container(
            decoration: BoxDecoration(
                border: Border.all(
                    color: const Color.fromARGB(255, 255, 255, 255))),
            child: Text(
              card.text,
              style: Theme.of(context)
                  .textTheme
                  .titleLarge!
                  .copyWith(fontWeight: FontWeight.normal, fontSize: 18),
            ),
          )),
        ],
      ),

      SizedBox(height: 20),
      // Кнопки "Написать ответ", "Увидеть ответ"
      Row(
        children: [
          Expanded(
              child: _buildActionButton(
                  'Написать ответ', 'white', context, "white", "none")),
          SizedBox(width: 8),
          Expanded(
              child: _buildActionButton(
                  'Увидеть ответ', 'white', context, "white", "none")),
        ],
      ),
    ],
```

```
    );
  }
}
```

Приложение Б

Листинг файла LoginPageScreen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import '../../sign_up_page/widgets/password_text_field_widgets.dart';

class LogInPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: Container(
        width: double.infinity,
        decoration:
            BoxDecoration(color: Theme.of(context).colorScheme.background
                // gradient: LinearGradient(
                //   begin: Alignment.topCenter,
                //   end: Alignment.bottomCenter,
                //   colors: [
                //     Theme.of(context).colorScheme.background,
                //     const Color.fromARGB(255, 126, 191, 244)
                //   ],
                // ),
                ),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            SizedBox(
              height: 60,
            ),
            Padding(
              padding:
                  EdgeInsets.only(top: 10, right: 20, left: 20, bottom: 20),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
```

```
      children: <Widget>[
        Text(
          "Вход",
          style: TextStyle(color: Colors.white, fontSize: 40),
        ),
        SizedBox(
          height: 10,
        ),
        Text(
          "С возвращением",
          style: TextStyle(color: Colors.white, fontSize: 18),
        ),
      ],
    ),
  ),
  SizedBox(height: 20),
  Expanded(
    child: Container(
      decoration: BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.only(
              topLeft: Radius.circular(60),
              topRight: Radius.circular(60))),
      child: Padding(
        padding: EdgeInsets.all(30),
        child: Column(
          children: <Widget>[
            Center(
              child: SvgPicture.asset(
                'assets/logo.svg',
                semanticsLabel: 'Logo',
                width: 100.0,
                color: Theme.of(context).colorScheme.background,
              ),
            ),
            Container(
              decoration: BoxDecoration(
                  color: Colors.white,
                  borderRadius: BorderRadius.circular(10),
                  boxShadow: [
                    BoxShadow(
                        color: Color.fromRGBO(0, 124, 6, 0.29),
```

```
                    blurRadius: 20,
                    offset: Offset(0, 10))
              ]),
          child: Column(
            children: <Widget>[
              Container(
                padding: EdgeInsets.all(10),
                decoration: BoxDecoration(
                    border: Border(
                        bottom: BorderSide(
                            color: Colors.grey.shade300))),
                child: TextField(
                  decoration: InputDecoration(
                      hintText: "Email",
                      hintStyle: TextStyle(
                          color: Color.fromARGB(255, 75, 91,
99)),

                      border: InputBorder.none),
                ),
              ),
              Container(
                padding: EdgeInsets.all(10),
                decoration: BoxDecoration(
                    border: Border(
                        bottom: BorderSide(
                            color: Colors.grey.shade200))),
                child: PasswordTextField(
                  label: 'Пароль',
                ),
              ),
            ],
          ),
        ),
        SizedBox(
          height: 40,
        ),
        GestureDetector(
          onTap: () {
            Navigator.pushNamed(context, '/sign-up');
          },
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
```

```
                    children: [
                      Text(
                        "Нет аккаунта? ",
                        style: TextStyle(
                            color: Color.fromARGB(255, 75, 91, 99)),
                      ),
                      Container(
                        padding: EdgeInsets.only(
                          bottom: 2, // Space between underline and
text
                        ),
                        decoration: BoxDecoration(
                            border: Border(
                                bottom: BorderSide(
                          color: Colors.blue,
                          width: 1.0, // Underline thickness
                        ))),
                        child: Text(
                          "Зарегистрироваться?",
                          style: TextStyle(
                            color: Colors.blue,
                          ),
                        ),
                      )
                    ],
                  ),
                ),
                SizedBox(
                  height: 40,
                ),
                MaterialButton(
                  onPressed: () {
                    Navigator.of(context).pushNamed('/');
                  },
                  height: 50,
                  // margin: EdgeInsets.symmetric(horizontal: 50),
                  color: Theme.of(context).colorScheme.background,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(50),
                  ),
                  // decoration: BoxDecoration(
                  // ),
```

```
                        child: Center(
                          child: Text(
                            "Войти",
                            style: TextStyle(
                                color: Colors.white,
                                fontWeight: FontWeight.bold),
                          ),
                        ),
                      ),
                    ],
                  ),
                ),
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

Листинг файла main.dart.

```
import 'package:education_app/features/card_page/view/card_page_screen.dart';
import
'package:education_app/features/course_page/view/create_course_page_screen.da
rt';
import 'package:education_app/generated/l10n.dart';
import 'package:education_app/models/course_arguments.dart';
import 'package:education_app/ui/theme/theme.dart';
import 'package:flutter/material.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:provider/provider.dart';

import 'features/card_page/view/create_card_page_screen.dart';
import 'features/course_page/course_page.dart';
import 'features/courses_list_page/view/courses_list_page_sceen.dart';
import 'features/home_page/home_page.dart';
import 'features/reviews_page/reviews_page.dart';
import 'features/settings_page/settings_page.dart';
import 'features/login_page/login_page.dart';
import 'features/sign_up_page/sign_up_page.dart';
```

```
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (BuildContext context) {},
      child: MaterialApp(
        localizationsDelegates: const [
          S.delegate,
          GlobalMaterialLocalizations.delegate,
          GlobalWidgetsLocalizations.delegate,
          GlobalCupertinoLocalizations.delegate,
        ],
        supportedLocales: S.delegate.supportedLocales,
        locale: Locale("ru"),
        title: 'Education App',
        theme: themeData,
        routes: {
          // '/': (context) => CardPage(),
          // '/': (context) => CourseCompletionScreen(),
          '/': (context) => HomePageScreen(),
          '/course': (context) {
            final courseId = ModalRoute.of(context)?.settings.arguments as
int;
            return CoursePage(courseId: courseId);
          },
          '/reviews-list': (context) => ReviewsListPageScreen(),
          '/log-in': (context) => LogInPage(),
          '/sign-up': (context) => SignPage(),
          '/card': (context) => CardPage(),
          '/create-card': (context) {
            final courseId = ModalRoute.of(context)?.settings.arguments as
int;
            return CreateCardPage(courseId: courseId);
          },
          '/create-course': (context) => CreateCoursePage(),
          '/settings': (context) => SettingsPageScreen(),
```

```
          '/courses-list': (context) {
            final args =
                ModalRoute.of(context)!.settings.arguments as CourseArguments;
            return CoursesListScreen(args);
          },
        },
      ),
    );
  }
}
```