МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ПОД IOS С ОБРАЗОВАТЕЛЬНЫМИ ИГРАМИ ДЛЯ ДЕТЕЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Петриченко Артёма Максимовича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во времена, когда компьютерные технологии развиваются с огромной скоростью, появляются новые и эффективные методы обучения. Один из таких методов — это использование игровых форм для запоминания образовательного материала. Дидактические компьютерные игры создают привлекательную среду обучения, которая позволяет ученикам активно взаимодействовать с учебным материалом и улучшает процесс запоминания и понимания. Кроме того, игровые формы обучения делают процесс более интересным и увлекательным, что способствует удержанию внимания учеников и повышает мотивацию к обучению.

Дидактические компьютерные игры являются мощным инструментом в обучении, который помогает ученикам легко и быстро усваивать новый материал. Они предлагают интерактивный и увлекательный способ обучения, который может быть адаптирован к индивидуальным потребностям каждого ученика. Кроме того, такие игры могут быть использованы для развития различных навыков, таких как логическое мышление, внимание и память. В целом, дидактические компьютерные игры являются отличным дополнением к традиционным методам обучения и могут значительно повысить эффективность образовательного процесса.

Цель бакалаврской работы — разработать iOS–приложение с образовательными играми для детей, которое будет соответствовать высоким стандартам удобства использования, интерактивности и эстетической привлекательности, работающее как самостоятельное приложение.

Поставленная цель определила следующие задачи:

1. Изучить лучшие практики в разработке мобильных образовательных приложений для iOS.
2. Проанализировать технологический стек для iOS-разработки, включая Swift, Objective-C, UIKit и другие актуальные фреймворки.
3. Спроектировать архитектуру приложения, оптимизированную под устройства Apple.
4. Реализовать интерактивные элементы с использованием последних возможностей iOS SDK.
5. Обеспечить интеграцию с серверной частью для синхронизации данных и сохранения прогресса пользователя.

1 Основное содержание работы

1.1 Языки программирования для iOS

Swift — это мощный и интуитивно понятный язык программирования от Apple, который был впервые представлен в 2014 году. Он был разработан с целью упростить процесс разработки приложений и сделать его более доступным. Swift обладает современным синтаксисом, который легко читается и пишется, и включает в себя ряд функций, которые делают код более безопасным и менее подверженным ошибкам. Кроме того, Swift предлагает мощные возможности для работы с функциональным программированием и обработкой ошибок.

Objective-C был основным языком программирования для разработки приложений на iOS до появления Swift. Он был создан в начале 1980-х годов и является расширением языка программирования C с добавлением возможностей объектно-ориентированного программирования. Objective-C имеет более сложный синтаксис по сравнению со Swift и может быть более сложным для изучения для новых разработчиков. Однако он все еще активно используется, особенно в старых и больших проектах.

Важно отметить, что, несмотря на различия между Swift и Objective-C, оба языка могут быть использованы вместе в одном проекте, благодаря мосту между Swift и Objective-C, предоставляемому Apple. Это позволяет разработчикам постепенно переходить на Swift, сохраняя существующий код на Objective-C. Это также означает, что разработчики могут использовать существующие библиотеки и фреймворки Objective-C в своих проектах на Swift.

1.2 Использование UIKit и SwiftUI для создания пользовательского интерфейса

UIKit — это набор библиотек и инструментов, который был основным фреймворком для разработки приложений на iOS на протяжении многих лет. UIKit предоставляет разработчикам все необходимые инструменты для создания графического интерфейса приложения, включая элементы управления, такие как кнопки, слайдеры и переключатели, а также более сложные элементы, такие как таблицы, коллекции и навигационные контроллеры. UIKit также предоставляет поддержку многозадачности, анимации, графики и работы с текстом. Однако, несмотря на свою мощь и гибкость, UIKit имеет свои сложности. Он основан на объектно-ориентированном подходе и требует от разработчиков

написания большого количества шаблонного кода. Кроме того, UIKit не предоставляет прямой поддержки декларативного стиля программирования, который становится все более популярным.

SwiftUI — это совершенно новый фреймворк для создания пользовательских интерфейсов, который использует декларативный подход. Вместо того чтобы указывать, как именно должен выглядеть интерфейс и как он должен изменяться в ответ на различные действия пользователя, с SwiftUI разработчик просто описываете, как должен выглядеть UI в зависимости от текущего состояния приложения, и фреймворк берет на себя остальную работу. SwiftUI значительно упрощает процесс создания пользовательского интерфейса, делая его более прямым и менее подверженным ошибкам. Он также включает в себя ряд современных функций, таких как поддержка темного режима, анимации, доступности и межъязыковой поддержки прямо из коробки.

1.3 Работа с сетью и HTTP

HTTP (HyperText Transfer Protocol) — это основной протокол, используемый для передачи данных в интернете. Он определяет, как сообщения форматируются и передаются, а также как веб-серверы и браузеры должны реагировать на различные команды.

В контексте мобильных приложений, HTTP играет важную роль в обмене данными между приложением и сервером. Большинство мобильных приложений являются «клиент–серверными», что означает, что они взаимодействуют с веб–сервером для получения, отправки и обновления данных. Это включает в себя все, от загрузки новых контентов и синхронизации данных пользователя до отправки уведомлений и выполнения транзакций. В iOS есть несколько инструментов, которые можно использовать для работы с HTTP и выполнения сетевых запросов:

1. URLSession: Это основной класс, который используется для отправки сетевых запросов в iOS. URLSession предоставляет API для выполнения HTTP и HTTPS запросов, загрузки и загрузки файлов, а также для работы с веб-сокетами. URLSession поддерживает как синхронные, так и асинхронные запросы, а также предоставляет возможности для настройки поведения сессии, такие как политики кэширования, таймауты и другие.
2. Alamofire: Это сторонняя библиотека, которая предоставляет более высокоуровневый и удобный интерфейс для выполнения сетевых запросов.

Alamofire облегчает выполнение сложных сетевых операций, таких как многопоточная загрузка, обработка ошибок и преобразование данных. Однако, использование Alamofire требует добавления сторонней зависимости в ваш проект.

3. Combine: Это фреймворк от Apple, который предоставляет функциональность для обработки асинхронных событий с использованием концепций из реактивного программирования. Combine может быть использован вместе с URLSession для создания более чистого и управляемого кода при работе с асинхронными сетевыми запросами.

1.4 Стартап как диплом

Пользовательская часть приложения, разработанная в рамках данной бакалаврской работы, предназначалась для университетского проекта «EduPlay», который участвует в программе «Стартап как диплом».

«Стартап как диплом» — это программа, которая направлена на вовлечение талантливых студентов в развитие экосистемы технологического предпринимательства, а также на поддержку бизнеса, находящегося на начальной стадии. Разработка и реализация программы обучения абитуриентов и преподавателей университетов в подготовке стартапов в качестве ВКР предусмотрены программой «Цифровая экономика». По итогам защиты проекта было получено экспертное заключение, приложенное к бакалаврской работе, в котором высоко оценён инновационный подход и потенциал проекта в сфере образовательных технологий.

EduPlay — это веб–портал, который разрабатывается командой студентов для обучения детей дошкольного возраста, учеников младших классов и родителей, желающих, чтобы обучение приносило их детям удовольствие.

1.5 Метки

Описывается процесс создания специализированного класса для меток, наследуемого от базового класса UILabel. Это делается с целью упрощения повторного использования данного элемента в приложении и минимизации дублирования кода.

UILabel — стандартный класс в UIKit, используемый для отображения статического текста. Однако, в больших приложениях часто возникает необходимость в более сложном поведении или внешнем виде меток. Вместо того

чтобы каждый раз заново настраивать эти метки, создается специализированный класс, наследуемый от UILabel и включающий в себя все необходимые настройки.

В классе меток предусмотрены различные инициализаторы, которые позволяют создавать метки с различными параметрами, такими как выравнивание текста, размер шрифта и толщина шрифта. Это обеспечивает большую гибкость при создании меток и позволяет легко создавать метки, соответствующие общему стилю приложения. Также в классе меток есть метод для настройки различных свойств метки, таких как цвет текста, минимальный масштаб шрифта, режим переноса строк и автоматическое изменение размера шрифта. Это позволяет обеспечить единообразие и согласованность внешнего вида всех меток в приложении.

В целом, создание специализированного класса для меток - это эффективный способ управления метками в приложении. Это обеспечивает единообразие, сокращает дублирование кода и делает код более чистым и легче для понимания. Это также упрощает процесс изменения внешнего вида или поведения всех меток в приложении, так как все изменения можно внести в одном месте - в специализированном классе меток.

1.6 Кнопки

Создание двух типов кастомных классов для кнопок в iOS, наследуемых от базового класса UIButton: filled и plain.

Кнопки filled и plain имеют различные характеристики и внешний вид. Filled кнопки имеют заполненный фон, в то время как Plain кнопки не имеют фона, но имеют границу. Это позволяет создавать разнообразные кнопки, которые соответствуют различным требованиям дизайна приложения.

Каждый из этих классов кнопок имеет свои собственные методы и свойства, которые позволяют настраивать внешний вид и поведение кнопок. Например, можно настроить цвет текста, цвет фона, текст кнопки, а также обработку событий, таких как нажатие на кнопку.

Важной особенностью этих классов является то, что они используют автоматическое размещение для установки ограничений. Это позволяет легко управлять положением и размером кнопок на экране.

В целом, создание кастомных классов для кнопок — это эффективный способ управления кнопками в приложении. Это обеспечивает единообразие,

сокращает дублирование кода и делает код более чистым и легче для понимания. Это также упрощает процесс изменения внешнего вида или поведения всех кнопок в приложении, так как все изменения можно внести в одном месте - в кастомных классах кнопок.

Также стоит отметить, что кнопка на стартовом экране обрабатывает события перехода на следующий экран, добавляя навигационный контроллер с экраном Логина. Это обеспечивает удобство использования и единообразие интерфейса.

1.7 Экран логина

В разделе «Экран логина» описывается создание интерфейса для входа в приложение. Экран входа состоит из различных элементов, включая текстовые поля, плоскости (вью) и кнопки. Снимок экрана приложения изображен на рисунке 1.

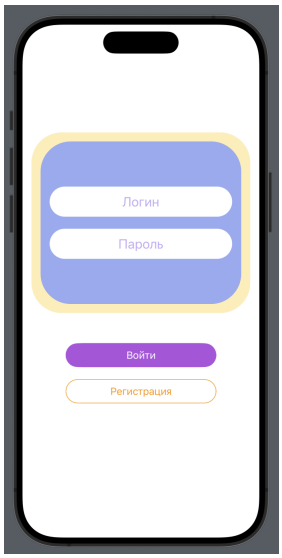

Рисунок 1 – Снимок экрана входа

Описывается создание двух видов вью: beigeView и blueView. Оба добавляются на главное представление и настраиваются с определенными цветами фона и радиусами углов. Затем устанавливаются ограничения для каждого представления, определяющие их положение и размер на экране.

Также описывается создание кастомного класса для текстового поля, EPTextField, который позволяет избежать повторения кода при создании текстовых полей с одинаковыми атрибутами в приложении.

1.8 Авторизация

В разделе «Авторизация» описывается процесс входа в приложение. При нажатии на кнопку входа, извлекаются текстовые значения из полей для ввода логина и пароля. Затем создается URL–запрос к серверу авторизации. При получении ответа от сервера, если в ответе присутствует токен, он сохраняется в UserDefaults. После успешной авторизации происходит смена корневого контроллера на EPTabBarController.

1.9 Навигация приложения

В разделе «Навигация приложения» описывается управление переключением между различными экранами приложения. Класс EPTabBarController управляет переключением между различными экранами приложения (игры, лидерборды, профиль). Класс AppCoordinator управляет переключением между экранами авторизации и основным интерфейсом приложения.

1.10 Экран с играми

В разделе «Игра» описывается создание интерфейса для отображения веб–страницы игры. Экран игры состоит из веб–просмотрщика WKWebView, который позволяет отображать веб–страницы прямо в приложении.

В классе GameSelectedVC, который является подклассом UIViewController, создается новый экземпляр WKWebView и добавляется на главное представление контроллера. Затем загружается URL игры, если он существует.

Также в этом классе создается кнопка «closeButton», которая добавляется в навигационную панель контроллера. При нажатии на эту кнопку вызывается метод «closeButtonTapped()», который закрывает контроллер представления.

Таким образом, GameSelectedVC представляет собой контроллер представления, который отвечает за отображение веб–страницы игры в WKWebView. Пользователь может просматривать веб–страницу игры и закрыть контроллер представления, нажав на кнопку закрытия. Это обеспечивает удобный и интуитивно понятный интерфейс для пользователя, позволяя ему легко просматривать информацию об игре и возвращаться к предыдущему экрану при необходимости. Это также демонстрирует, как можно интегрировать веб–контент в приложение, используя WKWebView, и управлять этим контентом с помощью встроенных средств управления интерфейсом пользователя iOS, таких как

UIBarButtonItem. Это позволяет создавать богатые и интерактивные пользовательские интерфейсы, которые могут включать в себя как нативные элементы пользовательского интерфейса iOS, так и веб–контент.

1.11 Экран с таблицей лидеров

В разделе «Экран с таблицей лидеров» описывается создание интерфейса для отображения таблицы лидеров. Экран лидерборда представляет собой уникальный и важный элемент пользовательского интерфейса, который служит зеркалом достижений и прогресса игроков.

В отличие от экрана игр, где основной акцент делается на представлении различных игр, экран лидерборда сосредоточен на отображении результатов игроков. При нажатии на ячейку таблицы лидерборда, пользователь перенаправляется на детальное описание игры, где, помимо основной информации об игре, представлены результаты игроков.

Класс «LeaderboardTableVC», который является подклассом «UIViewController», отвечает за отображение таблицы лидеров для определенной игры. В этом классе используется объект «UITableView» для отображения таблицы лидеров. Он конфигурируется в методе «configureTableView()», где устанавливаются его делегат, источник данных, идентификаторы повторного использования для ячеек и цвет разделителя.

Метод «fetchLeaderboardData(gameLinkName:)» выполняет запрос на сервер для получения данных таблицы лидеров для игры с указанным именем ссылки. Полученные данные затем декодируются в массив объектов «LeaderBoard» и сохраняются в свойстве «leaderboardData». Затем таблица перезагружается, чтобы отобразить новые данные.

Метод «tableView(:numberOfRowsInSection:)» возвращает количество строк в таблице, которое равно количеству записей в таблице лидеров плюс одна для ячейки игры.

Метод «tableView(:cellForRowAt:)» возвращает ячейку для указанного индекса строки. Если индекс строки равен нулю, то возвращается ячейка игры, которая конфигурируется с информацией об игре. В противном случае возвращается ячейка таблицы лидеров, которая конфигурируется с записью таблицы лидеров для данной строки.

Метод «tableView(:didSelectRowAt:)» вызывается при выборе строки в таблице. Если выбрана строка, отличная от нулевой, то создается новый экземпляр

«ProfileOtherVC» и показывается с информацией о пользователе, соответствующему выбранной записи таблицы лидеров.

Таким образом, «LeaderboardTableVC» представляет собой контроллер представления, который отвечает за отображение таблицы лидеров для определенной игры. Он загружает данные таблицы лидеров с сервера, отображает их в таблице и обрабатывает выбор строки в таблице. Это обеспечивает удобный и интуитивно понятный интерфейс для пользователя, позволяя ему легко просматривать информацию об игре и возвращаться к предыдущему экрану при необходимости. Это также демонстрирует, как можно интегрировать веб-контент в приложение, используя WKWebView, и управлять этим контентом с помощью встроенных средств управления интерфейсом пользователя iOS, таких как UIBarButtonItem. Это позволяет создавать богатые и интерактивные пользовательские интерфейсы, которые могут включать в себя как нативные элементы пользовательского интерфейса iOS, так и веб-контент.

1.12 Экран с профилем пользователя

В разделе «Экран с профилем пользователя» описывается создание интерфейса для отображения информации о пользователе и его прогрессе в играх. Экран профиля пользователя играет центральную и важную роль, служа персональной доской достижений и прогресса пользователя.

Основным элементом на этом экране является UITableView, который отображает различные данные в зависимости от выбранного сегмента контроллера в UISegmentedControl. Когда выбран сегмент «Ваш профиль», таблица отображает данные о пользователе и его прогрессе в играх. Когда выбран сегмент «Ваш компаньон», таблица отображает информацию о маскоте пользователя.

В разделе «Экран с друзьями пользователя» описывается создание интерфейса для отображения списка друзей пользователя. Экран друзей играет существенную и центральную роль, служа персональной платформой для взаимодействия и социализации пользователей.

Основным элементом на этом экране является UITableView, который отображает список друзей пользователя. Этот список может быть отфильтрован с помощью UISegmentedControl, который содержит три сегмента: «Все», «Входящие» и «Исходящие». В зависимости от выбранного сегмента, в таблице отображаются все друзья пользователя, входящие заявки в друзья или исходя-

щие заявки в друзья. Каждая ячейка в таблице представляет собой друга или заявку в друзья и содержит кнопки, которые позволяют принять или отклонить заявку в друзья или удалить друга.

Таким образом, экран друзей предоставляет удобный и интуитивно понятный интерфейс для управления списком друзей пользователя. Он позволяет пользователям просматривать своих друзей, принимать или отклонять заявки в друзья и удалять друзей, обеспечивая гибкость и контроль над их социальными взаимодействиями в игре.

ЗАКЛЮЧЕНИЕ

В ходе данной работы была разработана пользовательская часть для приложения с образовательными играми для детей.

Для достижения поставленной цели в рамках работы:

1. Изучены лучшие практики в разработке мобильных образовательных приложений для iOS.
2. Проанализирован технологический стек для iOS-разработки, включая Swift, Objective-C, UIKit и другие актуальные фреймворки.
3. Спроектирована архитектура приложения, оптимизированная под устройства Apple.
4. Реализованы интерактивные элементы с использованием последних возможностей iOS SDK.
5. Обеспечена интеграция с серверной частью для синхронизации данных и сохранения прогресса пользователя.

Таким образом, все поставленные в рамках работы задачи были выполнены. Завершающим этапом разработки стала реализация основных экранов приложения и функциональных возможностей, что привело к созданию готового продукта, готового к использованию. Итоговое приложение не только соответствует образовательным стандартам, но и предоставляет увлекательный способ обучения для детей, что делает его ценным инструментом в образовательной среде. Таким образом, работа не только достигла поставленных целей, но и продемонстрировала потенциал использования мобильных технологий для улучшения образовательного процесса. Это делает ее неотъемлемой частью общего игрового опыта, подчеркивая важность персонализации и индивидуального подхода в современных образовательных приложениях. Это подчеркивает важность интеграции веб–технологий в мобильные приложения для обеспечения богатого и интерактивного пользовательского опыта.