Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **ФИКТ**

Направление подготовки **45.03.04 Интеллектуальные системы в гумманитарной сфере**

Образовательная программа**: Языковые модели и искусственный интеллект**

К У Р С О В О Й П Р О Е К Т

Тема: «iOS приложение CryptoTracker»

Обучающийся: Мелихов Андрей Юрьевич, K3162

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире криптовалюты стали неотъемлемой частью глобальной экономики и инвестиций. Многие пользователи нуждаются в удобном инструменте для отслеживания актуальных данных о рынке криптовалют. CryptoTracker решает эту задачу, предлагая простой интерфейс и возможность изучения подробной информации о любой монете.

Целью проекта является разработка приложения, которое отображает список криптовалют с актуальными ценами, изменением за сутки и предоставляет доступ к подробной информации о выбранной монете.

Основные задачи проекта:

- построение базовой архитектуры приложения [1].
- реализация функционала получения данных через REST API [2].
- создание экрана списка криптовалют и экрана подробной информации о монете.
- интеграция сетевого слоя с пользовательским интерфейсом.

1 Описание проекта

Проект CryptoTracker представляет собой мобильное приложение для iOS [3], которое позволяет пользователям отслеживать актуальную информацию о криптовалютах.

Приложение отображает список криптовалют с их текущими ценами, процентными изменениями за последние 24 часа и предоставляет возможность детального просмотра данных о конкретной монете. Интерфейс приложения интуитивно понятен и разработан с учетом современных требований UX/UI. В основе проекта лежит использование REST API [2] для получения данных о рынке криптовалют в реальном времени.

Целью проекта является упрощение доступа к информации о криптовалютах для пользователей, что делает приложение полезным как для начинающих инвесторов, так и для профессионалов.

2 Процессы работы над проектом

Работа над проектом началась с этапа планирования и распределения задач между участниками команды. Основной архитектурной моделью была выбрана MVVM (Model-View-ViewModel) [1], так как она позволяет эффективно разделять бизнес-логику и логику представления данных, что упрощает поддержку и расширение функционала приложения.

Для реализации пользовательского интерфейса использовались UIKit [4] и SnapKit [5], а для взаимодействия с API и обработки данных был выбран URLSession с использованием Combine [6] для управления асинхронными процессами. Командная работа была организована через систему задач, каждая из которых имела четкие временные рамки. Регулярные обсуждения и код-ревью позволили выявлять возможные ошибки и находить оптимальные решения.

3 Задачи, поставленные передо мной

Разработать систему авторизации с использованием шаблона проектирования MVC (Model-View-Controller), включающую в себя механизм навигации между контроллерами и сохранение состояния авторизации пользователя.

Интеграция `Coordinator` и `StorageService`: Необходимо интегрировать разработанные классы в архитектуру приложения, чтобы обеспечить взаимодействие между контроллерами и механизмом сохранения состояния авторизации. Это включает в себя модификацию контроллеров авторизации и главного экрана для использования `Coordinator` для переходов и `StorageService` для работы с данными авторизации.

Реализация механизма авторизации: Необходимо разработать логику авторизации, которая будет взаимодействовать с бэкендом (если используется) и `StorageService` для сохранения токена доступа. Включить обработку ошибок авторизации.

Тестирование: Необходимо провести тщательное тестирование разработанных компонентов, включая модульные тесты для `Coordinator` и `StorageService`, а также интеграционные тесты для проверки корректной работы всей системы авторизации. Включая тестирование сценариев успешной и неуспешной авторизации, а также обработку ошибок.

4 Задача экрана подробной информации о монете

Экран подробной информации был реализован в классе CoinViewController. Этот экран предоставляет пользователю данные о конкретной криптовалюте, включая начальную и конечную цены за сутки, процентное изменение стоимости, текущую цену и дату последнего обновления, рисунок 1.

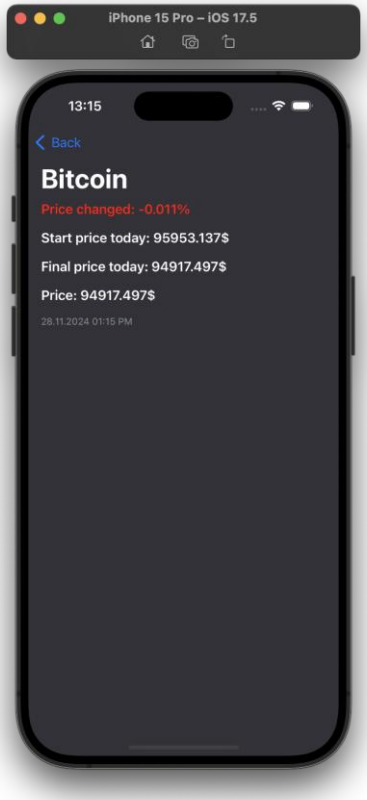

Рисунок 1 – Экран CoinViewController

Реализация началась с проектирования интерфейса с использованием стека UIStackView, который обеспечил удобное размещение элементов. Для получения данных о выбранной монете был разработан метод передачи данных через CoinViewModel. Это позволило обработать запросы к API и обеспечить своевременное обновление информации на экране.

Для обработки ошибок, возникающих при получении данных, был предусмотрен алерт-контроллер, который информирует пользователя о сбоях в работе API. Это улучшило пользовательский опыт и сделало приложение надежнее.

5 Решение задачи авторизации

Разработка системы авторизации была выполнена с использованием архитектурного шаблона MVC (Model-View-Controller) и включала в себя механизм навигации между контроллерами с помощью Coordinator, а также сохранение состояния авторизации пользователя через StorageService.

В процессе реализации возникли следующие основные сложности: разработка механизма авторизации с взаимодействием с бэкендом для проверки учетных данных пользователя и получения токена доступа, интеграция Coordinator и StorageService в общую архитектуру приложения для обеспечения гибкой навигации между экранами и сохранения состояния авторизации, а также проведение тестирования компонентов и интеграции для проверки корректной работы всей системы.

Реализация механизма авторизации была выполнена в соответствии с принципами MVC. На уровне модели был создан класс AuthModel, который отвечает за отправку учетных данных пользователя на сервер, обработку ответа от бэкенда, включая получение и сохранение токена доступа, а также за возврат результата авторизации (успех или ошибка). На уровне контроллера был реализован AuthViewController, обрабатывающий действия пользователя, такие как нажатие кнопки "Войти", и взаимодействующий с моделью для выполнения авторизации. В случае успешной авторизации контроллер передает управление Coordinator для перехода на главный экран. При возникновении ошибок авторизации контроллер отображает пользователю соответствующие уведомления. Интерфейс пользователя (View) был создан с учетом удобства взаимодействия, включая текстовые поля для ввода логина и пароля, а также кнопку "Войти".

Интеграция Coordinator и StorageService была реализована следующим образом. Coordinator отвечает за управление навигацией между экранами. Основной координатор (AppCoordinator) определяет логику переходов, включая запуск экрана авторизации при отсутствии токена доступа или главного экрана при успешной авторизации. Специализированный AuthCoordinator управляет процессом авторизации и уведомляет AppCoordinator о необходимости перехода на другой экран. StorageService используется для сохранения токена доступа и проверки состояния авторизации. Для сохранения данных использовался UserDefaults. При успешной авторизации токен доступа сохраняется в StorageService, а при разлогинивании токен удаляется, и пользователь перенаправляется обратно на экран авторизации.

Тестирование системы авторизации включало как модульные, так и интеграционные тесты. Модульные тесты покрывали функциональность StorageService (сценарии сохранения, получения и удаления токена) и Coordinator (сценарии навигации между экранами, такие как переход на главный экран после успешной авторизации и возврат на экран авторизации при разлогинивании). Интеграционные тесты проверяли связность всех компонентов: корректную передачу данных между моделью, контроллером и координатором, а также обработку различных сценариев, включая успешную авторизацию с переходом на главный экран, неуспешную авторизацию с отображением сообщения об ошибке и разлогинивание с возвратом на экран авторизации.

Таким образом, разработанная система авторизации соответствует архитектурным принципам MVC, обладает гибкостью и масштабируемостью благодаря использованию Coordinator и StorageService, а также прошла тщательное тестирование для минимизации ошибок и обеспечения стабильной работы.

6 Взаимодействие с командой и руководителем проекта

Взаимодействие с командой было организовано четко: задачи распределялись равномерно, и каждый член команды отвечал за свою область. Регулярные обсуждения помогали согласовывать изменения и решать возникающие проблемы.

Руководитель проекта оказал значительную поддержку, предлагая решения сложных вопросов и помогая с проверкой архитектурных решений. Его вклад способствовал успешной реализации проекта и достижению поставленных целей.

7 Оценка руководителя команды

Руководитель команды продемонстрировал себя как уверенный и компетентный лидер, который успешно организовывал работу всех участников проекта. Он хорошо распределял задачи, предоставлял своевременную обратную связь и следил за тем, чтобы проект двигался в направлении поставленных целей.

Благодаря его подходу удалось поддерживать продуктивную атмосферу в команде, где каждый осознавал свою роль и ответственность. Руководитель умело находил баланс между индивидуальными инициативами и общей стратегией, что способствовало эффективному решению возникающих проблем.

Особенное внимание хочется уделить его способности видеть проект в целом, предлагая оптимальные решения для интеграции различных модулей. Это позволило членам команды сосредоточиться на своих обязанностях, не теряя при этом общего направления.

В целом, руководитель команды проявил высокий уровень профессионализма, и его вклад стал ключевым для успешной реализации проекта.

ЗАКЛЮЧЕНИЕ

Подводя итог, можно сказать, что цель проекта была достигнута. Основные задачи, включая создание сетевого слоя, экрана списка криптовалют и экрана подробной информации, успешно выполнены. Оставшиеся мелкие доработки связаны с улучшением пользовательского интерфейса и производительности приложения. Лично я внёс значительный вклад в реализацию проекта, разработав экран подробной информации о монете и универсальный сетевой слой. Работа над проектом дала мне ценный опыт в использовании современных технологий, таких как Combine, SnapKit, и применении архитектуры MVVM. Также я научился более эффективно планировать свои задачи и решать сложные технические проблемы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. MVVM архитектруа приложения - https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel
2. API приложения: Messary IO - https://messari.io/
3. Официальная документация Apple - https://developer.apple.com/documentation/
4. Официальная документация UIKit - https://developer.apple.com/documentation/uikit/
5. Официальная документация Snapkit - https://snapkit.github.io/SnapKit/docs/
6. Фреймворк Combine и реактивное программирование - https://developer.apple.com/documentation/combine

ПРИЛОЖЕНИЕ

Этапы задач:

- Анализ предметной области,
- Проектирование,
- Разработка,
- Ручное тестирование.

Задачи:

- Название: Разработать экран авторизации
  Ответственный: Баташов Богдан Александрович
  Описание: Первый вью контроллер для авторизации (AuthViewController), просто 2 текс Филда (UITextField) и кнопка (UIButton).

- Название: Разработать экран списка монет
  Ответственный: Востров Илья Анатольевич
  Описание: Список всех монет сделанный через tableview.
  На ячейке укажите название монеты, ее стоимость, и ее изменение за сутки или час. (Данные брать из API)
  Пока список монет грузится, отображается спинер (UIActivityIndicator).
  После тапа на ячейку, осуществляется переход на 3-тий вью контроллер.

- Название: Разработка экрана подробной информации о монете
  Ответственный: Титор Матвей Андреевич
  Описание: 3 - ий вью контроллер. Просто детальная информация о монете.
  (Можно в ряд добавить лейблы с любой дополнительной информацией о монете)
  При нажатии на ячейку доставать название монеты и передавать на этот экран. Дальше использовать название для запроса

- Название: Внедрить координатор для навигации между контроллерами
  Ответственный: Мелихов Андрей Юрьевич
  Описание: Нужно добавить Координатор
  - может пушить и попать вьюконтроллер

- Название: Разработать класс StorageService для сохранения состояния авторизации пользователя

Ответственный: Мелихов Андрей Юрьевич

Описание: Класс который инкапсулирует в себе логику (методы) для сохранения состояния isAuth: Bool переменной в UserDefaults

- Название: Разработать класс NetworkLayer, инкапсулирующий логику взаимодействия с сетью

  Ответственный: Титор Матвей Андреевич

  Описание: - Построить сетевой слой. (Network Layer in Swift)

  - Разобраться как использовать Codable = Encodable & Decodable

  - GET/POST/PATCH - REST API

  - URLSeesion, URL, URLRequest

  - JSONDecoder

- Название: Разработать сервис икапсулирующий логику работы с CoreData (Опционально)

  Ответственный: Востров Илья Анатольевич

  Описание: Создание класса CoreDataManager. Если юзер не был авторизован

- Название: Добавить кнопки фильтрации в NavigationBar

  Ответственный: Попов Артемий Альбертович

  Описание: В навигейшен бар слева добавьте кнопку с возможностью сортировки изменения цены за сутки или за час

- Название: Настроить проект

  Ответственный: Востров Илья Анатольевич

  Описание: Интегрировать .gitignore файл, а также установить зависимости в проект (SnapKit), через cocoapods

- Название: Создать кастомные UI элементы

  Ответственный: Попов Артемий Альбертович

  Описание: - Наследник UITextField, который умеет менять бордер, если произошла ошибка и отображать внизу error message (по сути надо сделать наследника CustomTextField: UITextField, настроить его делегаты, а потом его вместе с CustomLabel: UILabel поместить в класс TextFieldView: UIView и при верстке использовать только его)