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

Факультет **Прикладной информатики**

Направление подготовки **09.03.03 Прикладная информатика**

Образовательная программа **Мобильные и сетевые технологии**

К У Р С О В О Й П Р О Е К Т

Тема: «iOS приложение CryptoTracker»

Обучающийся: Титор Матвей Андреевич, K3139

Санкт-Петербург 2024

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ 2
ВВЕДЕНИЕ 3
1 Описание проекта 4
2 Процессы работы над проектом 5
3 Суть проблемы, поставленной передо мной 6
4 Решение задачи экрана подробной информации о монете 7
5 Решение задачи сетевого слоя 9
6 Анализ работы и выводы 10
7 Взаимодействие с командой и руководителем проекта 11
8 Оценка руководителя команды 12
ЗАКЛЮЧЕНИЕ 13
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14
ПРИЛОЖЕНИЕ 15

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 Суть проблемы, поставленной передо мной

Моя задача включала создание экрана подробной информации о монете и разработка сетевого слоя, чтобы обеспечить взаимодействие приложения с внешними API.

Главные сложности заключались в:

- построении универсального и легко расширяемого сетевого слоя,
- интеграции данных с REST API в пользовательский интерфейс,
- обеспечении корректного отображения данных при ошибках сети или API.

Проблема в деталях:

- экран подробной информации о монете:

  Нужно было передавать данные от списка криптовалют на другой экран, динамически подгружая дополнительную информацию,
- сетевой слой:

  Нужно было создать универсальное решение, подходящее для различных API-запросов (например, получения списка монет, их исторических данных и т. д.). Также возникали сложности с асинхронным управлением данными.

4 Решение задачи 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.

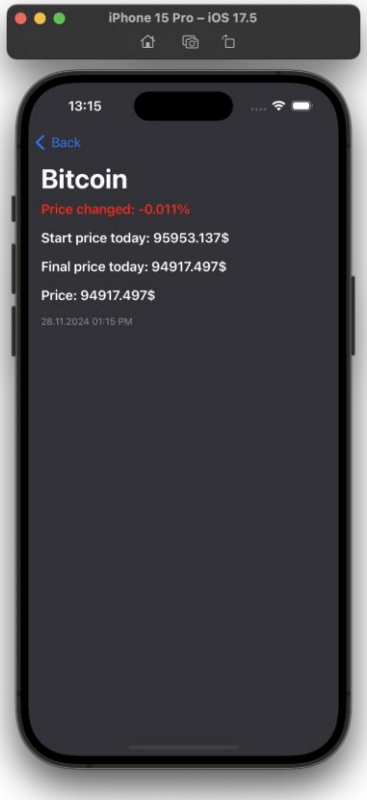

Рисунок 1 – Экран CoinViewController

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 сетевого слоя

Сетевой слой был реализован в классе NetworkService, инкапсулирующем всю логику взаимодействия с API. Для формирования запросов использовались компоненты URLComponents, что позволило легко изменять параметры запросов и добавлять новые эндпоинты.

Основная функциональность включала отправку запросов к API, обработку полученных данных и их декодирование с помощью JSONDecoder. Для обработки ошибок была реализована система проверки кода ответа сервера, которая классифицировала ошибки на клиентские, серверные и сетевые.

Особенностью реализации стало использование протокола INetworkService, что позволило упростить тестирование и обеспечило гибкость при масштабировании сетевого слоя.

6 Анализ работы и выводы

Проект позволил мне углубить знания в области работы с сетевыми запросами, использования архитектуры MVVM и библиотек Swift, таких как Combine. В процессе работы возникли трудности, связанные с обработкой данных API и корректным отображением информации при отсутствии подключения к сети. Эти проблемы удалось решить за счет тестирования и применения асинхронных механизмов обработки данных.

Несмотря на возникающие трудности, удалось планомерно выполнять поставленные задачи, следуя дедлайнам. Работа над проектом стала ценным опытом, позволившим не только освоить новые технологии, но и улучшить навыки работы в команде.

7 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.

8 Оценка руководителя команды

Руководитель команды показал себя как компетентный и уверенный лидер, который умело координировал работу всех участников проекта. Он чётко распределял задачи, обеспечивал своевременную обратную связь и следил за тем, чтобы проект развивался в соответствии с поставленными целями.

Благодаря его подходу удалось сохранить рабочую атмосферу в команде, где каждый понимал свою роль и ответственность. Руководитель поддерживал баланс между индивидуальной инициативой и общей стратегией, что помогало эффективно решать возникающие проблемы.

Особенно хочется отметить его умение видеть проект в целом, предлагая лучшие решения для интеграции отдельных модулей. Это позволило каждому члену команды сосредоточиться на своей зоне ответственности, не теряя общего фокуса.

В целом руководитель команды проявил высокий профессионализм, и его вклад был ключевым фактором 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)