Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по курсовой работе «IOS приложение CryptoTracker»

по дисциплине «Инфокоммуникационные системы и технологии»

Автор: Баташов Богдан Александрович

Факультет: ФИКТ

Группа: К3162

Преподаватель: Горлушкина Н.Н.

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время рынок криптовалют активно развивается, привлекая всё большее количество инвесторов. Для успешной работы на этом рынке необходим удобный и эффективный инструмент для отслеживания цен, изменения рыночной капитализации и других важных параметров криптовалют. CryptoTracker iOS призван решить эту задачу, предоставляя пользователям мобильное приложение для комфортного мониторинга криптовалютного рынка. Полученный результат (рабочее приложение) будет полезен как опытным трейдерам, так и начинающим инвесторам, стремящимся к эффективному управлению своими активами.

Целью данного проекта является разработка мобильного приложения CryptoTracker для iOS, обеспечивающего удобный и информативный мониторинг криптовалютного рынка.

Основные задачи проекта:

Разработка экрана авторизации

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

1 Описание проекта

CryptoTracker – это iOS-приложение, позволяющее пользователям отслеживать цены различных криптовалют в режиме реального времени. Ключевой функционал включает в себя: отображение текущих цен с процентным изменением цен за последние сутки.

Главная задача проекта - предоставить пользователям удобный и надежный инструмент для мониторинга курсов криптовалют в режиме реального времени.

2 Работа над проектом

Работа над проектом началась с этапа планирования и распределения задач между участниками команды. Основной архитектурной моделью была выбрана MVVM (Model-View-ViewModel) [2], 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 [3] и SnapKit [4], а для взаимодействия с API и обработки данных был выбран URLSession с использованием Combine [5] для управления асинхронными процессами. Командная работа была организована через систему задач, каждая из которых имела четкие временные рамки.

3 Суть проблемы, поставленной передо мной

Моей основной задачей была разработка экрана авторизации. Это включало в себя разработку первого вью контроллера для авторизации (AuthViewController), т. е. 2 текс Филда (UITextField) и кнопка (UIButton). Реализация 2 строк для проверки что логин пароль введён верно (логин/пароль 1234/1234). Если пароль или логин не верны, показывать красную обводку вокруг тексфилд и подсказку с названием ошибки (кастомный view который содержим Филд и лейбл). После успешного ввода логина и пароля сделать переход на второй вью контроллер. Переход не просто через пуш вью контроллер, а смену рутового вьюконтроллера у окна или у навигейшен контроллера. Сделать так чтобы после успешного ввода логина/пароля есть возможность разлогиниться со 2/3 включения, и вернуться на экран авторизации. Или после закрытия приложения новый запуск начинался сразу со второго включения. Сохранение состояния авторизаци в (UserDefauls) при закрытии приложения.

4 Разработка экрана авторизации

1. Запуск приложения: В AppDelegat проверяется значение isLoggedIn в UserDefaults. Если true, то загружается MainViewController; иначе — AuthViewController.

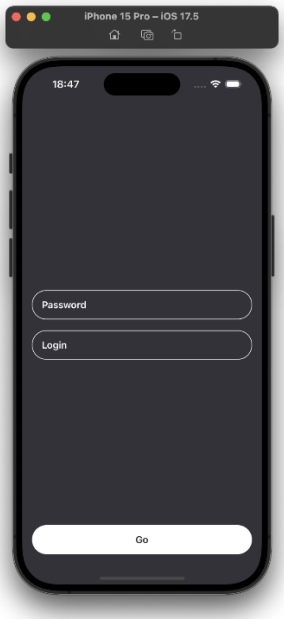

Рисунок 1 - Экран авторизации

2. Пользователь вводит данные в usernameTextField и passwordTextField.

3. Вход в систему осуществляется нажатием на loginButton:

AuthViewModel вызывается метод authenticate. AuthViewController подписывается на изменения isLoading и error в AuthViewModel.

Пока выполняется аутентификация (isLoading == true), отображается activityIndicator.

При успешной аутентификации:

isLoggedIn устанавливается в true в UserDefaults и AuthViewModel.

AuthViewController переходит на MainViewController, заменяя rootViewController.

При ошибке аутентификации:

Отображается сообщение об ошибке из error в errorLabel.

4. Выход из системы: В MainViewController, при нажатии на logoutButton:

Вызывается метод logout() в AuthViewModel.

isLoggedIn в UserDefaults устанавливается в false.

MainViewController переходит на AuthViewController, заменяя rootViewController.

5 Анализ работы и выводы

Проект позволил мне углубить знания в области работы с сетевыми запросами, использования архитектуры MVVM и библиотек Swift. 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.

6 Взаимодействие с командой и руководителем проекта

Взаимодействие с командой было хорошо организованно. Мы использовали telegram-звонки для оперативного общения и обмена информацией. Задачи распределялись руководителем проекта с учетом навыков каждого участника команды. Ежедневно проводились короткие планерки, где каждый член команды рассказывал о своем прогрессе за предыдущий день, планировал задачи на текущий день и сообщал о любых проблемах, требующих внимания. Общий позитивный настрой и взаимная поддержка способствовали эффективной работе и успешному завершению проекта.

Руководитель проекта сыграл важную роль, поддерживая команду в решении сложных задач и проверке архитектурных решений. Его участие существенно способствовало успешному выполнению проекта и достижению целей.

7 Оценка руководителя проекта

Руководитель продемонстрировал высокий уровень компетентности в управлении проектом и оказал значительную помощь в решении сложных задач. Его советы по реализации функций приложения и анализу обратной связи от пользователей значительно повысили качество конечного продукта. Таким образом, его руководство было эффективным, что способствовало успешному завершению проекта.

ЗАКЛЮЧЕНИЕ

Подводя итоги, можно сказать, что цель была успешно достигнута. Основные задачи были выполнены в срок. Лично я внес значительный вклад в проект, разработав экран авторизации. Работа над проектом дала мне ценный опыт в создании IOS-приложений, а также в работе с командой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация об использовании API приложения: Messary IO -https://messari.io/api
2. MVVM архитектруа приложения - https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel
3. Официальная документация UIKit - https://developer.apple.com/documentation/uikit/
4. Официальная документация Snapkit - https://snapkit.github.io/SnapKit/docs/
5. 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)