Санкт-Петербургский государственный университет

Математическое обеспечение и АИС
Кафедра системного программирования

Ференц Данила Святославович

Проектирование и разработка приложения для портфельного менеджмента на платформе iOS

Курсовая работа

Научный руководитель:
ст. преп. Я. А. Кириленко

Консультант:
Ф. П. Долголев

Санкт-Петербург
2020

Оглавление

Введение

На данных момент, из-за развития онлайн бирж, набирают популярность онлайн инвестиции. В связи с доступностью информации, всё больше людей становятся финансово подкованными и из-за того, что онлайн инвестиции не требуют обширного начального капитала, начинают ими заниматься. Получается, что растет и спрос на приложения, которые позволяют грамотно анализировать процесс торговли и тем самым помогать инвесторам. Данная работа и посвящена разработке приложения, позволяющего проводить анализ деятельности инвестора на бирже, тем самым её упрощая.

Конечной целью любого инвестора является получение прибыли, поэтому одна из их задач - купить ценные бумаги подешевле и продать подороже. Для этого инвесторы создают «инвестиционный портфель» - это совокупность капиталовложений, которые были инвестированы в различные направления для того, чтобы снизить риск получения убытков к минимуму. Выбор портфеля зависит напрямую от конкретных целей инвестора и его поведения на рынке. Он бывает агрессивным, то есть нацеленным на рискованные акции с быстрым ростом, консервативным или умеренным. Причём у одного инвестора их может быть несколько. Необходимо грамотно регулировать портфели, иными словами диверсифицировать - снижать риски и уменьшать зависимость от колебаний рынка. Это одна из компонент портфельного менеджмента - процесса управления инвестиционным портфелем. Также в него входит анализ доходности, результативности и корреляции портфеля. Этот процесс можно упростить, если предложить пользователю продукт, позволяющий отобразить его операции, построить на их основе графики, показывать на основе метрик успешность его торговли, что в конечном итоге позволит принимать более взвешенные решения.

Каким должно быть данное приложение? Понятным и удобным, чтобы люди, не разбирающиеся глубоко в финансовой сфере, могли его использовать. Включать в себя необходимые метрики, чтобы предоставить полную информацию о портфеле.

В рамках студенческого проекта в компании DSX Technologies была собрана наша команда, целью работы которой является разработка такого приложения.

1. Цели и задачи

В данном разделе сформулированы цели и задачи работы.

1.1. Цель работы

Проектирование и разработка на платформе iOS приложения для портфельного менеджмента.

1.2. Поставленные задачи

Для выполнения поставленной цели были выделены следующие задачи:

- Исследование предметной области: разобраться с основными принципами биржевой торговли и на какие аспекты при анализе стоит обратить внимание
- Проанализировать существующие на рынке продукты
- Выделить обязательную и дополнительную функциональность в приложении, а также изучить языки и технологии для реализации приложения
- Проанализировать библиотеки для построения графиков на платформе iOS и выделить подходящие под мои цели. Реализовать с помошью них заявленную функциональность
- Проектирование и реализация приложения: на этапе проектирования описать логику работы приложения, продумать дизайн. В конце этапа реализации обеспечить покрытие юнит тестами

2. Обзор существующих решений

Рассмотрим представленные на рынке решения для портфельного менеджмента, с позиции пользователя выделим положительные аспекты каждого и их недостатки.

2.1. Существующие решения

Для анализа были выбраны следующие приложения:

- Intelinvest - приложение для учета и контроля инвестиций. По словам разработчиков оно подходит как для новичков, так и для опытных инвесторов [5]
  Плюсы:
  - Небольшой порог вхождения
  - Возможность автоматической загрузки данных
  - Возможность посмотреть распределение активов в портфеле и график его стоимости

  Минусы:
  - Приложение является платным
  - Отсутствует поддержка криптовалютных активов
- Blockfolio - приложение позволяет агрегировать информацию о своих активах (например Bitcoin, Litecoin и пр.) в разных криптовалютах, которые могут хранится на разных торговых площадках, кошельках и других имеющихся у пользователя средств хранения [3]
  Плюсы:
  - Современный дизайн
  - Ручное добавление балансов

Минусы:

- Представлены только криптовалюты

- Personal Capital - бесплатное приложение для мониторинга финансовых активностей [11]
  Плюсы:
  - Классификация активов по основным категориям и их представление
  - Безопасность данных, двухфакторная аутентификация

  Минусы:
  - Приложение не представлено на российском рынке
  - Нет возможности изменять категории активов
  - Перегружено дополнительной функциональностью (счета, кредитные карты, планирование бюджета)

- Investment Account Manager - Windows приложение для управления инвестиционным портфелем [6]
  Плюсы:
  - Детальное описание портфеля
  - Возможность объединять портфели для составления отчетов
  - Ручная загрузка данных

  Минусы:
  - Отсутствие мобильной версии
  - Устаревший интерфейс

- Mint - популярное приложения для управления финансами [10]
  Плюсы:
  - Автоматическая подгрузка информации с бирж

– Безопасность, двухфакторная аутентификация

Минусы:

– Нет возможности ручного добавления сделок

– Ориентировано на Американских и Канадских пользователей

– Сервис ориентирован больше именно на бюджет

Также были рассмотрены и другие подобные продукты, однако, оказалось, что они направлены на планирование бюджета или на другие аспекты, связанные с финансовой деятельностью, а не на порфтельный менеджмент.

2.2. Сравнение продуктов

Для сравнения представленных на рынке решений были выделены следующие критерии:

- Мультиплат. - Мультиплатформенность
- Сложность - Высокий порог вхождения
- Метрики - Наличие необычных метрик по портфелю
- Доступность - Коммерциализированность продукта

Продукт	Мультиплат.	Сложность	Метрики	Доступность
Intelinvest	+	-	-	-
Blockfolio	-	-	-	+
Personal Capital	+	-	-	-
Inv.Acc.Manager	-	+	+	+

Таблица 1: Сравнение существующих решений

Таким образом, существование большого количества приложений на рынке подтверждает востребованность и актуальность заявленной темы. Однако исходя из таблицы можно сделать вывод, что на данный

момент на российском рынке нет доступного для пользователя мультиплатформенного приложения, обладающего необходимой функциональностью. Это объясняется тем, что некоторые из приложений направлены также на планирование бюджета, другие ориентированы на работу только с криптовалютными портфелями или вовсе нацелены на Американский рынок.

3. Выделение требований

В данном разделе рассказано о выделенных функциональных и нефункциональных требованиях к приложению.

3.1. Функциональные требования

После проделанного анализа существующих решений, а также написания user stories мною решено было реализовывать следующую функциональность:

Авторизация и регистрация. Каждый пользователь загружает информацию о своих порфтелях, поэтому необходимо было предусмотреть:

- Создание учетной записи пользователя
- Авторизацию пользователя

Портфели пользователя. У пользователя может быть несколько портфелей, с которыми он может производить различные действия. Вследствие этого надлежало предусмотреть:

- Создание нового портфеля
- Редактирование портфеля
- Удаление портфеля

Добавление операций пользователя. Портфели не могут существовать без транзакций и сделок, совершенных в рамках них. Вследствие чего надобно было продумать:

- Добавление сделок и транзакций в интерфейсе
- Импорт данных о торговой деятельности из csv и xlsx файлов

Метрики. Для реализации возможности оценки успешности деятельности пользователя на бирже необходимо рассчитывать следующие метрики:

- Стоимость актива, пересчитанную в базовой валюте за определённый промежуток времени
- Стоимость портфеля, пересчитанную в базовой валюте во времени [1]
- Состав портфеля за промежуток времени и в настоящий момент
- Корреляция активов
- Доход портфеля и актива за выбранный промежуток времени

Визуализация. На основе представленных метрик предполагается строить графики и диаграммы:

- Диаграмму активов в портфеле на данный момент
- Линейный график стоимости актива за выбранный промежуток времени с заданным интервалом
- Кластерную диаграмму стоимости и состава портфеля за выбранный промежуток с заданным интервалом с возможностью пересчёта состава портфеля в процентах
- Таблицу корреляции активов с возможностью построения линейного графика корреляции активов во времени
- Линейный график сравнения котировок двух активов, нормированных относительно друг друга, во времени
- Линейный график дохода актива и портфеля за выбранный промежуток времени
- Столбчатую диаграмму ввода/вывода средств за выбранный промежуток времени

[1] Базовая валюта выбирается пользователем

3.2. Нефункциональные требования

После выделения необходимой функциональности были продуманы остальные требования:

- Язык разрабоки - Swift. В разработке приложений на платформе iOS используется два языка: Swift[1] и Objective-C.[2] Однако на данный момент компания Apple активно развивает и поддерживает только Swift, а также вынуждает компании активно мигрировать на него. Подробнее про преимущества языка [12]

- Решено было обеспечить возможность масштабирования приложения, удобного добавления дополнительной функциональности

4. Проектирование и анализ

В данном разделе рассказано о проектировании приложения и анализе библиотек.

4.1. Анализ библиотек для визуализации

Для визуализации метрик было решено использовать готовые библиотеки, поэтому необходимо было их проанализировать.
Анализ был проведен по следующим критериям:

- Настраиваемость - возможность изменять и адаптировать графики
- Разнообразие графиков - наличие круговой, столбцовой диаграммы и линейного графика
- Поддерживаемость - библиотека развивается и адаптируется к изменениям в языке Swift
- Документация - наличие документации

Библиотека	Настр-сть	Наличие	Поддер-сть	Документация
AAChartKit	-	+	+	+-
Charts	+	+	+	+-
SwiftCharts	+	-	+	+
PNChart	+	-	-	-
XJYChart	+	-	-	-

Таблица 2: Сравнение библиотек

Таким образом, после анализа сначала было решено использовать библиотеку AAChartKit из-за того, что она показалась мне более подходящей по дизайну, но впоследствии из-за отсутствия должной документации (она имелась, но на китайском языке) было решено перейти на библиотеку Charts.

4.2. Разработка архитектуры

Мною был выбран архитектурный паттерн MVC - Model View Controller (рис.1), который предлагает использовать Apple[9]. В ней View отвечает за действия пользователя, сообщая о них контроллеру. В Model происходит рассчёт метрик и через неё происходит взаимодействие с серверной частью приложения.

Такой подход позволяет исключить скопление ответственности за вы-

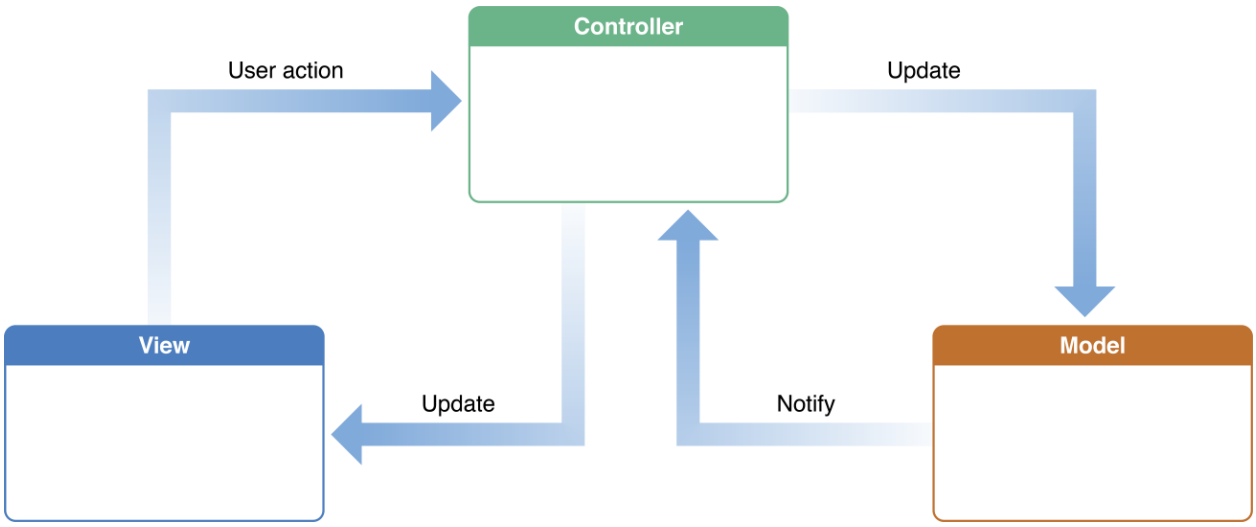

Рис. 1: MVC
`https://developer.apple.com`

числение метрик в одном классе и разделить их на разные классы, ответственные за одну или за несколько (в случае переиспользования вычислений) метрик.

Например: вычисление стоимости портфеля во времени и на данных момент, стоимость актива во времени решено было объединить в одну компоненту из-за того, что в них используется одинаковый подход в виде подсчёта балансов. (Рис. 2)

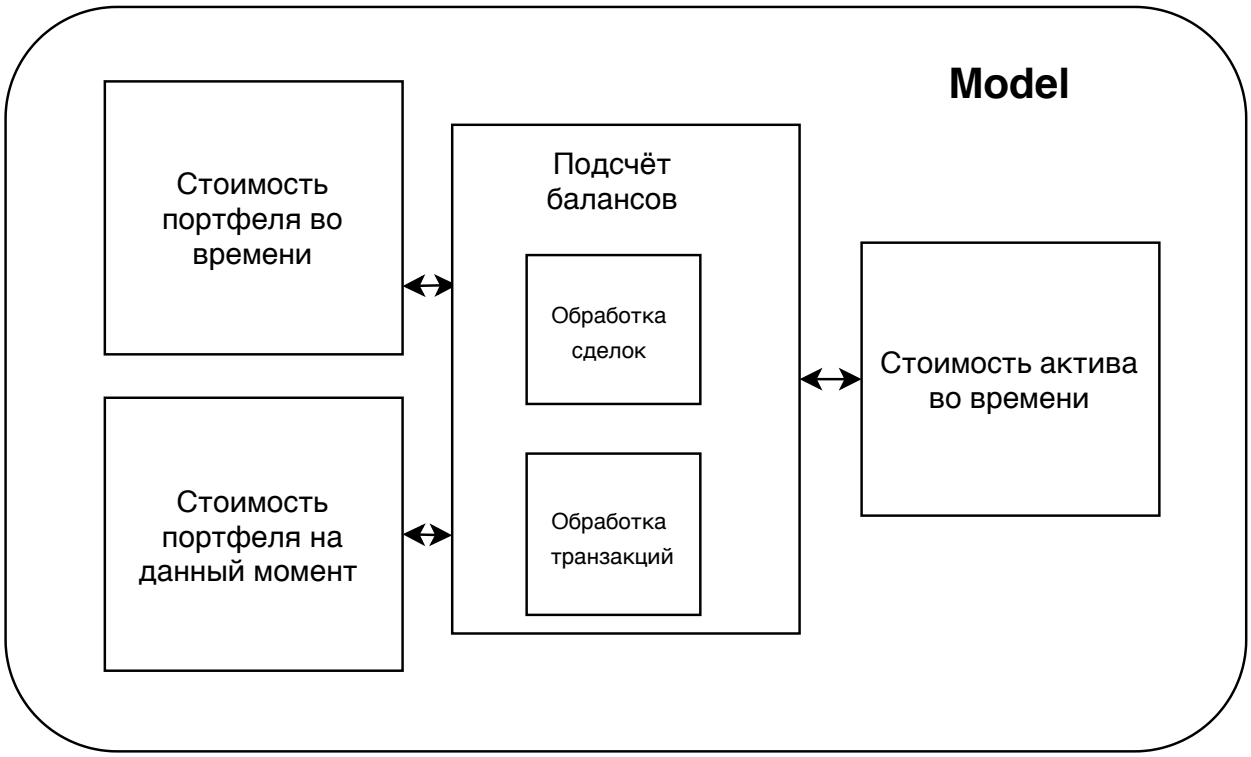

Рис. 2: Model

За взаимодействие с сервером отвечают три сервиса:

- Сервис идентификации - отвечает за регистрацию и аутентификацию пользователей
- Сервис портфелей - отвечает за добавление, изменение, удаление портфелей, ...
- Сервис данных - отвечает за транзакции, сделки и котировки

Модель совершает ассинхронные вызовы через данные сервисы, которые отправляют http запрос на сервер, обрабатывают ошибки и данные, полученные с сервера. (рис. 3)

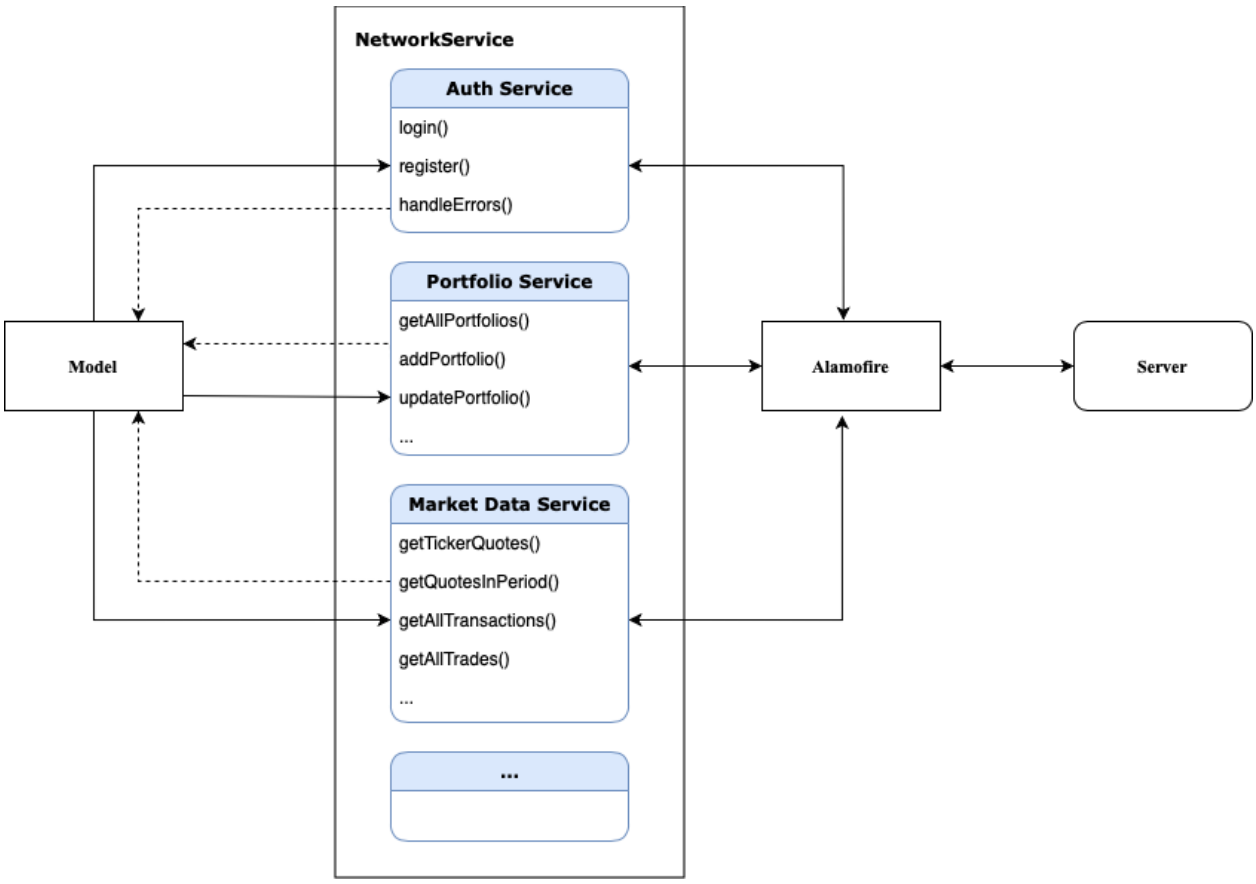

Рис. 3: Взаимодействие с сервером

5. Реализация

В данном разделе рассказывается об этапе реализации приложения.

5.1. Аутентификация

Для регистрации пользователю необходимо ввести имя, почту и пароль. (рис.I), а для аутентификации только имя и пароль. (рис.II) В приложении используется технология JSON Web Token.[7] (рис. 4)
На первом этапе данные отправляются на сервер. Затем пользователю возвращается закодированный и подписанный секретный ключ, имеющий срок действия - час. Этот ключ и используется пользователем на третьем шаге для доступа к данным.
Однако так как JWT не зашифрована, а только закодирована и подписана, то для безопасности приложения необходимо было продумать хранение ключа в приложении. Были изучены способы хранения ин-

Рис. 4: Технология JWT

формации в iOS приложениях:

- UserDefaults - способ хранения небольших данных и настроек приложения. Не является приемлемым из-за отсутствия шифрования
- File Manager - позволяет хранить файлы. Мне нужно было хранить одну строку
- Core Data - взаимодействие с базой данных. Заводить базу данных для токена было избыточно
- Keychain - сервис, созданный для хранения паролей и других секретных данных. Идеально подходил под мои нужды

Таким образом, для хранения токена была выбрана бибилотека KeychainAccess, использующая Keychain через Keychain Api от Apple.[8]

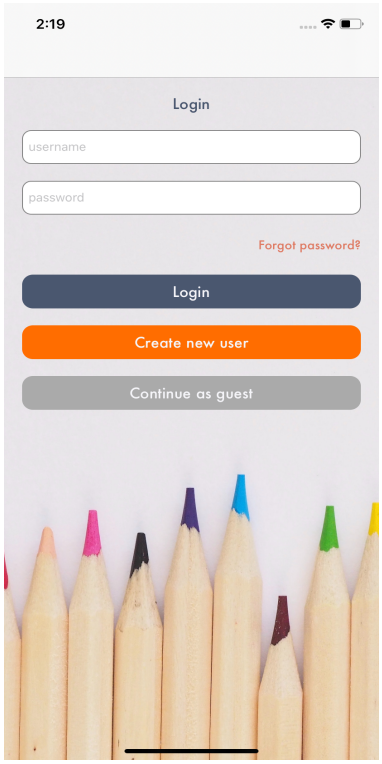

Рис.I: Аутентификация

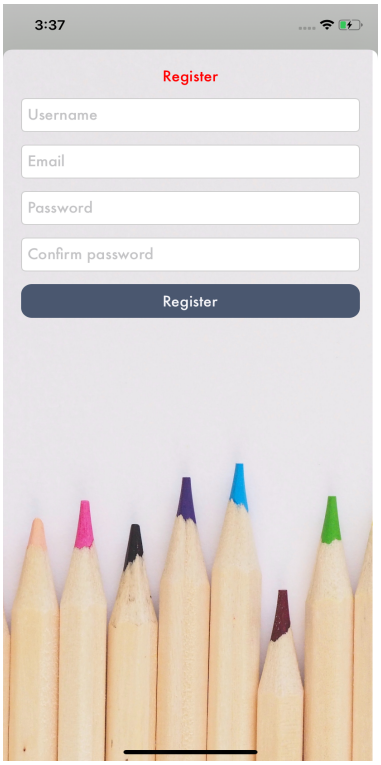

Рис.II: Регистрация

5.2. Добавление операций

Для того, чтобы строить по портфелю задуманные метрики, необходима информация об операциях пользователя. Была воплощена возможность добавления сделок и транзакций несколькими способами:

- Добавление одной сделки и транзакции вручную
- Загрузка из csv файла в формате бирж: Tinkoff[13], DSX [4]

Для удобства была добавлена возможность загружать операции сразу в несколько порфтелей. Так как в дальнейшем планируется увеличить количество воспринимаемых форматов, то работа с файлами была организована в отдельном контроллере. Для работы с csv файлами использовался Document Picker, позволяющий получить доступ к файлам пользователя на устройстве и загрузить их.
При ручной загрузке сделок и транзакций необходимо было предусмотреть, чтобы пользователь не ввёл несуществующие активы, неверную дату или количество актива в неверном формате.

5.3. Расчёт корреляции

Была изучена корреляция активов. Оказалось, что в финансовом мире используется Индикатор корреляции Пирсона (ИКП).
Иными словами индикатор, устанавливающий меру и направление взаимосвязи (обратная или прямая) выбранной пары инструментов на участке, определяемом пользователем.
Для рассчета корреляции двух активов необходимо иметь котировки двух активов за промежуток времени. Первым делом считается их средняя цена за этот промежуток:

$$\overline{\text{Цена актива}} = \frac{1}{n} \sum_{i=1}^{n} \text{Ценаi}$$

Затем считается среднеквадратичное отклонение:

$$\delta = \sqrt{\sum_{i=1}^{n} (\text{Цена} - \overline{\text{Цена}})^2}$$

Сам коэффициент Пирсона можно вычислить по формуле:

$$\text{Кпирсона} = \frac{cov(\text{Цена1,2})}{\delta\text{цена1} \times \delta\text{цена2}}$$

Коэффициент корреляции принимает значение от -1 до +1. Отрицательные значения коэффициента говорят о существовании обратной связи между рассматриваемыми активами. Положительные значения говорят о прямой связи.
Визуализировать индикатор корреляции Пирсона решено было следующим образом. Пользователю предлагается выбрать два актива и период подсчёта. Если один из активов совпадает с базовым [2], то пользователю предлагается либо поменять его, либо базовый. Результат представляется в виде таблички. В строке фиксируется первый актив, а в столбцы пользователь может добавлять активы, зависимость которых от зафиксированного он хочет проанализировать. (Рис. I) Также, зафиксировав строку, пользователь может построить график зависимостей по выбранным в столбцах активам. (Рис. II)

[2] Базовый выбирается пользователем в настройках (по умолчанию: доллары)

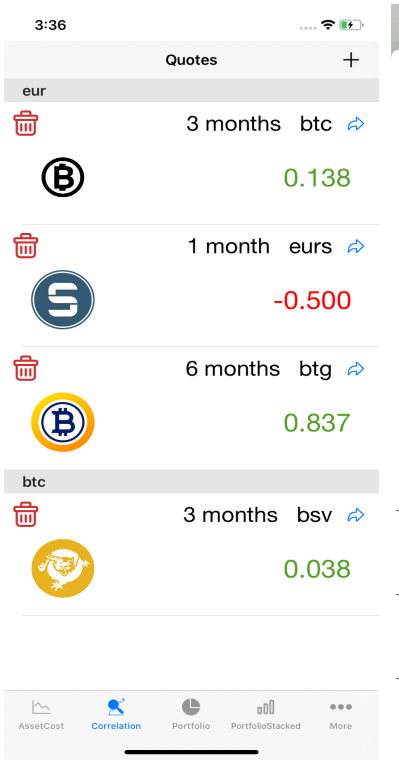

Рис.I: Таблица корреляции

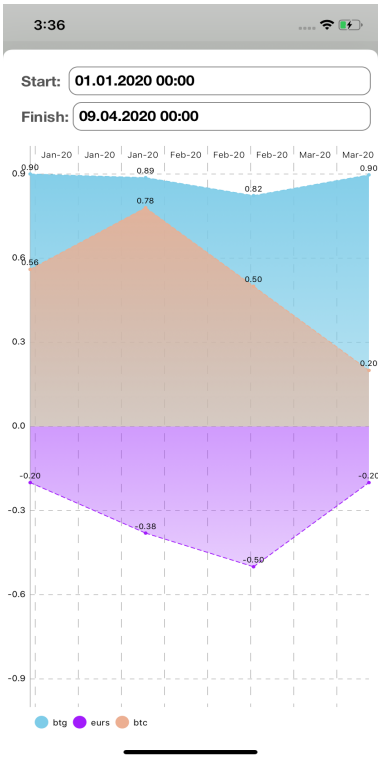

Рис.II: График корреляции

5.4. Подсчёт балансов

Для реализации метрик стоимости и состава портфеля, стоимости актива необходимо рассчитывать баланс. Первым делом запрашиваются сделки и транзакции пользователя в выбранном портфеле. Сделки сортируются по дате и идентификатору (необходимо учитывать в хронологическом порядке), а транзакции проверяются на успешность и сортируются. Затем они последовательно обрабатываются и формируют баланс пользователя. В случае, если необходимо рассчитывать баланс за определенный промежуток с заданным интервалов, то это промежуток разбивается на интервалы и в процессе рассчета проверяются временные границы.

Так как стоимость представляется в базовой валюте, то необходимо запросить с сервера котировки для каждого актива и пересчитать балансы в базовой валюте.

На основе балансов строится кластерная диаграмма стоимости и состава портфеля во времени. Пользователю предлагается выбрать времен-

ной промежуток, интервал (месяц или год). Также можно пересчитать диаграмму в процентах, чтобы было проще оценить состав портфеля. Помимо этого балансы позволяют построить диаграмму активов в портфеле и линейный график стоимости актива.

5.5. Подсчёт введённых и выведенных активов

Аналогичным образом считается объём введённых и выведенных активов. По запрошенным с сервера транзакциям в портфеле производится подсчёт. На этой основе строится горизонтальная столбчатая диаграмма.

5.6. Подсчёт дохода

Изначально решено было считать доход в виде разности балансов на конец и на начало интервала. Однако оказалось, из-за недостоверности результатов, что данный подход, выбранный мною являлся неверным. После этого было изучено как считается доход в финансовом анализе. Оказалось, что тремя способами:

- методом средневзвешанных цен
- методом FIFO
- методом LIFO

5.6.1. Метод средневзвешенных цен:

В данном методе доход считается как разность стоимости покупки и продажи актива. Стоимость покупки актива же считается как среднее арифметическое стоимостей его покупок.

5.6.2. Метод FIFO:

Метод FIFO (от англ. First In First Out - Первый вошёл, первый вышел) удобен при многократных покупках и продажах. Метод FIFO

утверждает, что при расчёте дохода первыми на продажу должны браться те сделки, которые были совершены первыми.
Например: были куплены 5 акций Сбербанка по 120 рублей, затем они подешевели и было докуплено еще 10 лотов по 110 рублей, Потом акции Сбербанка подорожали до 130 рублей и было решено продать 5 штук. В таком случае при подсчёте дохода используется цена 120 рублей:
доход = 5 × (130 рублей − 120 рублей) = 50 рублей
Из-за того, что существуют специальные инструменты для торговли на бирже, которые могут совершать несколько десятков сделок в секунду, то объём расчётов, которые необходимо совершать в данном методе велик. Однако, решено было использовать данный метод при подсчёте дохода, но вычисления перенести на серверную часть приложения, чтобы не нагружать клиентскую.

5.6.3. Метод LIFO:

Существует также метод LIFO (от англ. Last In First Out - Последний вошёл, первый вышел). Если вернуться к примеру в предыдущем параграфе, то в данном методе:
доход = 5 × (130 рублей − 110 рублей) = 60 рублей
Данный метод используется только в США и считается менее точным.

5.7. Сравнение котировок

Для сравнения котировок двух активов необходимо их нормировать друг относительно друга.
Для этого был использован следующий подход:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Данный подход позволяет наглядно отобразить на графике тенденции даже тех котировок, которые сильно отличаются диапазонами значений. (Рис. 7)

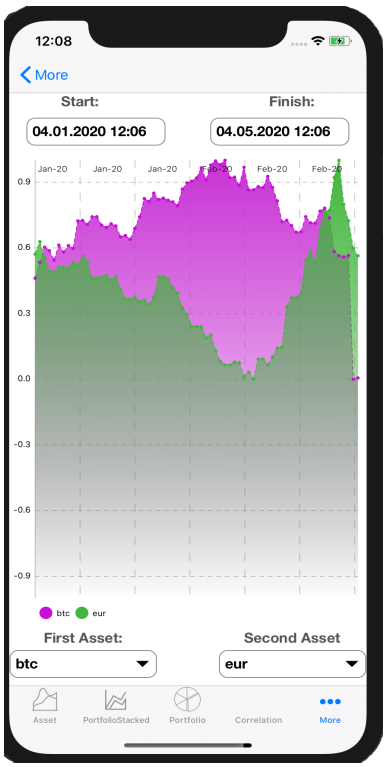

Рис. 7: Нормированный график

6. Трудности

В процессе реализации я столкнулся с несколькими проблемами, которые необходимо было решить.

6.1. Отрицательные балансы

Из-за того, что пользователю даётся возможность загружать сделки и транзакции за определённый период, то может так получится, что на момент начала периода у него уже имеются активы на бирже, о которых нет информации в приложении. Соотвественно, при подсчёте баланс уходит в минус (например совершена продажа актива, о наличии которого приложению не известно или этот актив был выведен со счёта). Данную проблему решено было обходить двумя путями. Во-первых при построении графиков делается смещение оси стоимости так, чтобы отрицательный баланс ушёл, а также при подсчёте высчитывается место ухода в отрицательный баланс и пользователю показывается, о каких активах была загружена неверная информация.

6.2. Медленная подгрузка данных

Как упоминалось в секции архитектуры - взаимодействие с сервером происходит через http запросы. Таких запросов необходимо делать большое количество, поэтому при последовательном подходе приложение работало медленно. Для этого решено было внедрить многопоточность и кеширование котировок.

1. Мною были изучены подходы к многопоточности в iOS приложениях. На практике используются:

   - Grand Central Dispatch - мощный фреймворк, оперирующий очередями, выполняющими задачи параллельно
   - Operations - класс, задающий операции, которые выполняются на очереди операций.

   Обе технологии являются мощнейшими инструментами и обе полезны, однако каждая из них предназначена для разных вещей:

Возможности	GCD	Operations
Ожидание полного завершения	+	-
Зависимости	-	+
Барьеры	+	-
Удаление всех операций	-	+

Таблица 3: Сравнение технологий многопоточности

Operations удобны, когда нужно строить зависимости или иметь возможность отменять задачи, а GCD даёт возможность ставить барьеры для преодоления состояния гонки.
Однако GCD позволяет с помощью Dispatch Group ждать полного завершения запущенных параллельно заданий, что и нужно реализовать в приложении. Запросы посылаются параллельно на

севрер, а когда вся информация придёт и обработается, то будет асинхронно возвращена.

2. Для внедрения кеширования решено было использовать базу данных SQLite через фреймворк CoreData. (Рис. 8).
   Контроллер запрашивает требуемые котировки у Sync Coordinator, который определяет есть ли они в базе данных. Если они есть, то происходит возврат в контроллер. В противном случае, котировки запрашивася на сервере, отсутствующие сохраняются в базу данных и затем возвращаются в контроллер.

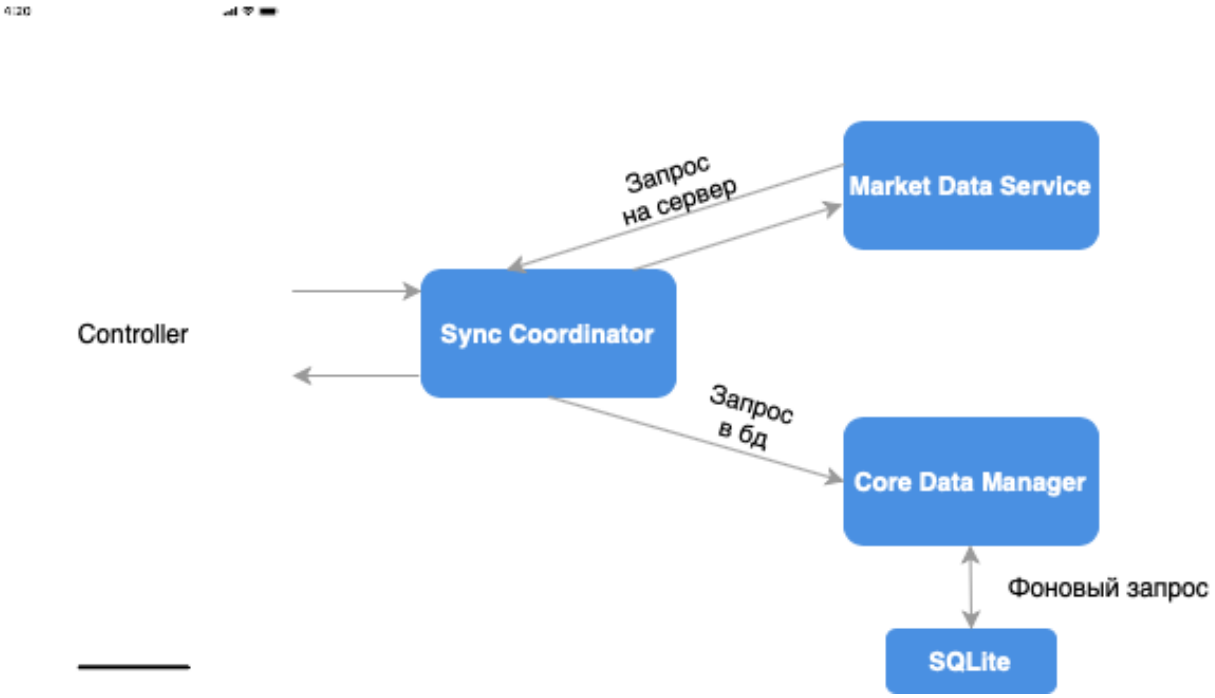

Рис. 8: Кеширование котировок

7. Тестирование

На основе реальных данных, предоставленных компанией DSX[4], было проведено юнит тестирование вычислительных модулей метрик. Было покрыто 94 % вычислительного модуля тестами.
В ходе тестирования также проверялись экстремальные данные:

- проверка на обработку только успешных транзакций
- полночь в датах
- различный порядок сделок и транзакций
- наличие только сделок и только транзакий
- нулевые значения в котировках
- и другие

8. Используемые технологии

Таким образом в процессе реализации были изучены и использованы следующие технологии:

- Язык разработки Swift
- iOS SDK
  - UIKit MVC - архитектура приложения
  - Grand Central Dispatch - фреймворк для внедрения многопоточности
- Keychain - библиотека для шифрования и хранения данных
- Charts - библиотека для построения графиков
- Alamofire - библиотека для конструирования и отправления http запросов
- Core Data - фреймворк, позволяющий разработчику взаимодействовать с хранилищем данных (с базой данных)
- iOS Unit Tests - библиотека для юнит тестирования приложения

Заключение

8.1. Текущие результаты

В данной курсовой работе были достигнуты следующие результаты:

1. Проанализированы существующие решения
2. Выявлены требования для клиентской части
3. Спроектирована клиентская часть приложения
4. Реализована клиентская часть приложения с заявленной функциональностью и интегрирована с серверной частью
5. Приложение протестировано

8.2. Дальнейшие планы

Ближайшими шагами будут:

1. выделение вычислительного модуля в библиотеку
2. добавление новых метрик и их визуализации
3. изучения того, как публикуются приложения в App Store [3]
4. добавление поддержки форматов данных новых бирж

Репозиторий, где ведётся разработка:
https://github.com/dsx-tech/student-project-balances-ios

[3] магазин приложений от компании Apple

Список литературы

[1] Apple. Swift // https://developer.apple.com/swift/. — 2020. — URL: https://developer.apple.com/swift/ (online; accessed: 01.05.2020).

[2] Apple. Swift // https://developer.apple.com/library/archive/documentation/ 2020. — URL: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html (online; accessed: 01.05.2020).

[3] Blockfolio // https://blockfolio.com. — 2020. — URL: https://blockfolio.com (online; accessed: 01.05.2020).

[4] Dsx format // https://api.dsxglobal.com. — 2020. — URL: https://api.dsxglobal.com (online; accessed: 01.05.2020).

[5] Intelinvest // https://intelinvest.ru. — 2020. — URL: https://intelinvest.ru (online; accessed: 01.05.2020).

[6] Investment Account Manager // https://www.investmentaccountmanager.com 2020. — URL: https://www.investmentaccountmanager.com (online; accessed: 01.05.2020).

[7] Json Web Token Технология // https://jwt.io/introduction/, year = 2020, url = https://jwt.io/introduction/, urldate =.

[8] KeychainAccess // https://github.com/kishikawakatsumi/KeychainAccess. — 2020. — URL: https://github.com/kishikawakatsumi/KeychainAccess (online; accessed: 01.05.2020).

[9] MVC // https://developer.apple.com/library/archive/documentation/Genera CocoaCore/MVC.html. — 2020. — URL: https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html (online; accessed: 01.05.2020).

[10] Mint // https://www.mint.com. — 2020. — URL: https://www.mint.com (online; accessed: 01.05.2020).

[11] Personal Capital // https://www.personalcapital.com. — 2020. — URL: https://www.personalcapital.com (online; accessed: 01.05.2020).

[12] Swift vs. Objective-C: A New Programming Language / Cristian González García, Jordán Espada, B. Pelayo García-Bustelo, Juan Cueva Lovelle // International Journal of Artificial Intelligence and Interactive Multimedia. — 2015. — 01. — Vol. 3. — P. 74–81.

[13] Tinkoff format // https://tinkoffcreditsystems.github.io/invest-openapi/. — 2020. — URL: https://tinkoffcreditsystems.github.io/invest-openapi/ (online; accessed: 01.05.2020).