Санкт-Петербургский государственный университет

Математическое обеспечение и АИС
Кафедра информационно-аналитических систем

Бородкин Вячеслав Павлович

Проектирование и разработка мобильного приложения для портфельного менеджмента на платформе Android

Курсовая работа

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

Санкт-Петербург
2020

Оглавление

Введение

В настоящее время существует множество бирж. Ежедневно на них проходит огромное количество сделок по покупке или продаже различных активов.

Приобретая тот или иной актив, человек вкладывает в него свои средства. Набор таких вложений называется "инвестиционным портфелем". Стоит заметить, что один человек может иметь больше одного портфеля, каждый из которых будет характеризоваться своим набором активов. Портфель требует тщательного анализа, качество которого сильно зависит от актуальности и удобного представления информации об истории торговли и состоянии рынка.

Есть большое количество метрик, позволяющих сделать те или иные предположения для дальнейшего выбора наиболее выгодной стратегии по увеличению прибыли. Соответственно, важно уметь выделять самые важные из этих метрик, чтобы не перегрузить человека информацией и при этом дать ему все необходимые сведения для комфортного анализа рынка.

Процесс управления инвестиционным портфелем, направленный на увеличение прибыли, получаемой с активов, входящих в него, называется портфельным менеджментом. Он включает в себя анализ доходности портфеля, расчёт рисков и принятие решений, по их уменьшению.

Важно заметить, что всё больше и больше людей по всему миру начинают пользоваться мобильными устройствами в повседневной жизни для получения актуальной информации и новостей. Тем самым, телевидение и персональные компьютеры отодвигаются на второй план.

Чтобы дать возможность людям, принимающим участие в торгах на биржах, получать необходимую им информацию и следить за состоянием их портфелей, существуют приложения для портфельного менеджмента. Их главной целью как раз и является удобное представление данных. Мобильные версии подобных приложений позволяют не только получать нужную информацию, но и делать это быстро.

На данный момент, выбор подобных программ для мобильных устройств

довольно мал, причём, на российском рынке счёт идёт на единицы.

Цель этой работы - создать полноценное приложение для портфельного менеджмента, предназначенное для мобильных устройств.

По данным за 2019 год[21], доля мобильных устройств под управлением операционной системы Android[2], составляет 74.13%, что делает её самой распространённой на современном рынке. Поэтому разработка была ориентирована именно под эту операциаонную систему.

1. Цели и задачи

Ниже приведены цели и выделены задачи, необходимые для их достижения.

1.1. Цель работы

Создать приложение для портфельного менеджмента, отличающееся удобной визуализацией предоставляемой информации, с расчётом на дальнейшее выделение функциональности приложения в отдельную библиотеку.

1.2. Поставленные задачи

- Исследовать предметную область: понять основные принципы биржевой торговли для корректного анализа инвестиционных портфелей
- Рассмотреть аналогичные приложения конкурентов с выделением ключевых особенностей каждого из них. Выделить функциональность, необходимую для портфельного менеджмента
- Реализовать расчёт выбранных метрик
- Разработать способы их визуализации
- Проанализировать существующие библиотеки для отрисовки графиков на мобильных устройствах под управлением операционной системы Android. Выбрать из них те, которые подходят для наших целей
- Спроектировать и реализовать приложение с учётом необходимой функциональности
- Протестировать работу приложения на реальных данных

2. Сбор требований и аналитика

2.1. Обзор существующих решений

Для сравнения существующих решений были выбраны следующие сервисы:

1. Blockfolio[5]
2. Future Adviser[8]
3. Mint[18]
4. SigFig Portfolio Tracker[20]
5. Investment Account Manager[12]

Таблица 1: Сравнение продуктов

Критерии	1	2	3	4	5
подробная визуализация портфеля	-	+	+	+	+
расчёт корреляции	-	-	-	-	+
выбор временных промежутков отображения данных	+	+	-	+	+
визуализация доходов	-	+	+	+	+
доступность российскому пользователю	+	-	-	-	-
мобильная версия приложения	+	-	+	-	-
интеграция с брокерами и биржами	+	+	+	+	+

Получается, что на данный момент нет продукта, отвечающего всем вышеперечисленным требованиям.

2.2. Выделение необходимой функциональности

Для того, чтобы обозначить необходимую для портфельного менеджмента функциональность, были написаны пользовательские истории, позволившие выделить следующие пункты:

- Регистрация и дальнейшая авторизация пользователя
- Отображение состава портфеля

- Отображение текущего баланса
- Добавление транзакций на ввод/вывод средств вручную
- Добавление сделок вручную
- Загрузка данных из csv-файла
- Расчёт различных метрик, а именно:
  - Корреляция актива
  - Доход конкретного актива за выбранный промежуток времени
  - Доход портфеля за выбранный промежуток времени
  - Стоимость портфеля в базовой валюте[1]
  - Стоимость конкретного актива в базовой валюте
  - Объём введённых и выведенных средств

Для визуализации метрик, указанных выше, были выбраны следующие графики:

Диаграмма распределения активов в портфеле. Предоставляет информацию об актуальном состоянии инвестиционного портфеля, отображая его стоимость и активы, входящие в него.

Кластерная диаграмма стоимости портфеля от времени. С её помощью пользователь может узнать, как со временем менялась капитализация портфеля и объёмы активов, входящих в него.

График дохода конкретного актива от времени. Показывает, какой доход приносит выбранный актив, давая возможность сделать выводы о его прибыльности.

[1]Базовая валюта - валюта, в которую пересчитывается стоимость

График дохода портфеля от времени. Даёт представление о доходе всего портфеля.

График стоимости конкретного актива в базовой валюте от времени. Отображает стоимость выбранного актива в портфеле.

График стоимости портфеля в базовой валюте от времени. Предоставляет информацию о том, как изменяется капитализация портфеля с течением времени.

График изменения цены актива от времени. Позволяет следить за изменением стоимости актива на рынке.

График корреляции актива от времени. Даёт возможность делать предположения о зависимости стоимости одного актива от стоимости другого.

Нормированный график сравнения котировок двух активов. С его помощью можно изобразить изменение цен на два актива, сильно отличающихся друг от друга по стоимости, на одном графике. Это позволяет (как и в предыдущем случае) обнаруживать зависимости между ценами двух активов.

Свечной график ввода/вывода средств за выбранный период. Даёт представление о количестве затраченных на торговлю средств и объём выведенных.

2.3. Анализ существующих библиотек по визуализации графиков

ЖГ Для визуализации метрик были рассмотрены следующие библиотеки: AndroidCharts[3], GraphView[10], JGraph[13], AnimatedPieView[4], MPAndroidChart[17], HelloChart[11], AACharts[1].
Сравнение происходило по ряду криетериев, таких как:

1. Открытая лицензия
2. Поддерживаемые графики
3. Вариативность изменений параметров графиков
4. Возможность комбинирования графиков

Таблица 2: Аналитика библиотек

Библиотека	1	2	3	4
AndroidCharts	+	Line Chart, Bar Chart, Clock Pie Chart, Pie Chart	+	+
GraphView	+	Line Chart, Bar Chart, Points Chart	-	+
JGraph	+	Line Chart, Bar Chart, Points Chart	-	-
AnimatedPieView	+	Pie Chart	+	-
MPAndroidChart	+	Line Chart, Bar Chart, Points Chart, Pie Chart, Bubble Chart	+	+
HelloChart	+	Line Chart, Bar Chart, Points Chart, Pie Chart, Bubble Chart	-	+
AACharts	+	Column Chart, Bar Chart, Area Chart, Areaspline Chart, Line Chart, Spline Chart, Radar Chart, Polar Chart, Pie Chart, Bubble Chart, Pyramid Chart, Funnel Chart, Columnrange Chart	+	+

После анализа библиотек, было выбрано 2 для дальнейшего использования, а конкретно:

- MPAndroidChart
- AACharts

3. Проектирование архитектуры

Приложение подразумевает клиент-серверное взаимодействие: сторона сервера должна отвечать за хранение информации о пользователях, котировках, сделках и транзакциях, сторона клиента - за расчёт необходимых метрик и предоставление информации в удобном виде. Как следствие, клиентская часть требует строгого разграничения вычислений и визуализации результатов, поэтому выбор пал на шаблон Model-View-ViewModel.

Чтобы не производить все вычисления для разных метрик в одном классе и не делать другой класс ответсвенным за отображение всех графиков, было принято решение для каждой отдельной метрики и соответствующего графика завести свою связку MVVM-типа, делая исключение для метрик, значения которых можно получить напрямую из резуультатов вычислений других. Для таких особых случаев, части, отвечающие за вычисления могут быть объёденены. Пример приведён на рисунке 1).

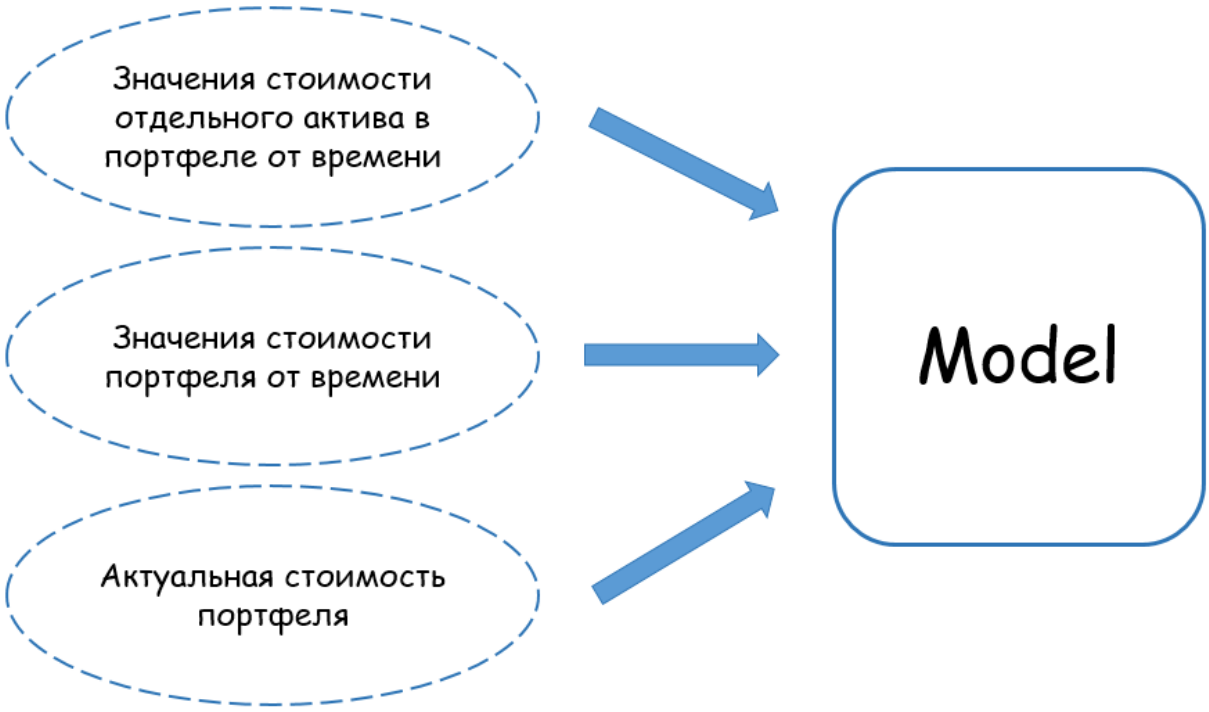

Рис. 1: Пример объёдинения расчётов нескольких метрик.

Такая архитектура позволила для каждой метрики получать дан-

ные с помощью запросов и производить расчёты в частях типа Model. Затем, передавая через промежуточную ViewModel-составляющую, визуализировать полученные результаты во View (рис. 2). Также, именно выделение расчётов метрик в отдельные классы позволит в дальнейшем вынести эти вычисления в отдельную библиотеку.

Рис. 2: Схема архитектуры.

Стоит также учесть, что получение ответа от сервера, вычисления и визуализация данных могут занять некоторое время, в течение которого приложение должно продолжать функционировать. Для этого, указанные действия должны производиться в отдельных асинхронных потоках, уведомляя остальные части приложения об окончании того или иного этапа.

Для того, чтобы пользователь получил доступ к расчёту метрик, необходимо прохождение им регистрации/авторизации. Для этих целей были созданы отдельные классы, отвечающие за взаимодействие с сервером при аутентификации.

Так как один человек может иметь несколько портфелей, необходимо было предусмотрель возможность переключения между уже существующими портфелями, их удаления, а также создания новых. Чтобы различать портфели, каждому из них присваивался уникальный номер, а для удаления и создания портфелей были написаны соответсвующие запросы.

3.1. Технологии

Языком разработки был выбран Kotlin[16], как самый приоритетный язык для написания android-приложений. Об этом было объявлено компанией Google[9] на своей конференции Google I/O 2019. В дополнение к этому, статическая типизация этого языка предотвращает некоторое количество возможных ошибок. Полная совместимость с языком Java[15], позволяет с лёгкостью интегрировать уже проверенные библиотеки, написанные на ней.

Для отправки запросов серверу использовалась библиотека Retrofit2[19], позволяющая с помощью корутин легко отправлять асинхронные запросы.

С выходом Android 6, получение разрешений на доступ к тем или иным функциональностям телефона необходимо в явном виде. Для удобной реализации получения этих разрешений использовалась библиотека Dexter[7].

Идентификатор пользователя с помощью Shared Preferences записывался в хранилище телефона.

4. Реализация

4.1. Взаимодействие классов

Для того, чтобы приложение могло продолжать работать во время обработки данных и расчёта метрик, вычисления и получение данных происходило в отдельных корутинах, доступных для реализации языком Kotlin. Данные же хранились с помощью компонента LiveData. Это позволило подписать классы, отвечающие за визуализацию, на изменения переменных, хранящих значения метрик.

4.2. Идентификация пользователя

Аутентификация пользователя является обязательным этапом, для дальнейшего использования приложения. Регистрация происходит путём заполнения формы, имеющей следующие поля:

Почта. На данном этапе не используемое поле, но необходимое в дальнейшем, для возможности восстановления пароля.

Логин.

Пароль.

Повторение пароля. Для уменьшения вероятности появления опечаток в указанном пароле.

Идентификация происходит с использованием стандарта JWT - JSON Web Token[14].

4.3. Загрузка данных

Чтобы анализировать портфель, необходимо добавить в него сделки и транзакции. Эта возможность была реализована двумя способами:

1. Добавление конкретной сделки или транзакции вручную

2. Загрузка из csv-файла

Чтобы иметь доступ к файлам мобильного устройства под управлением Android, необходимо получить явное согласие пользователя. Для этого, с помощью библиотеки Dexter, при первой попытке загрузить файл, программа спрашивает разрешение на доступ к хранилищу телефона. Такой вопрос задаётся единожды. В случае, если пользователь решит в дальнейшем изменить права доступа, это можно будет сделать в специальном разделе, через непосредственно, настройки телефона. На данный момент csv-файл принимается в формате двух бирж: DSX[6], Tinkoff[22]. Самостоятельное же добавление сделок и транзакций подразумевает заполнение специальных форм, включающих в себя все необходимые данные для хранения. Для транзакций это:

- id - идентификатор транзакции для хранения на сервере
- dateTime - дата и время совершения транзакции
- transactionType - тип транзакции
- currency - валюта, в которой производилась транзакция
- amount - объём
- commission - комиссия за выполнение транзакции
- transactionStatus - статус выполнения
- transactionValueId - идентификатор транзакции на бирже

Для сделок:

- id - идентификатор сделки для хранения на сервере
- instrument - валютная пара
- tradeType - тип транзакции
- tradedQuantity - объём первой валюты из инструмента

- tradedQuantityCurrency - первая валюта
- tradedPrice - объём второй валюты, обмениваемый на первую
- tradedPriceCurrency - вторая валюта инструмента
- commission - комиссия за выполнение сделки
- commissionCurrency - валюта, в которой взимается комиссия
- tradeValueId - идентификатор сделки на бирже

Для данной задачи очень важна точность вычисений. Поэтому хранение числовых значений, связанных с активами, происходило с помощью типа данных BigDecimal.

4.4. Проблема отрицательных балансов

При загрузке неполной истории тогровли может получиться так, что сделки проходят на средства, данных о которых нет в приложении.

Пример Пусть из всех загруженных сделок и транзакций первой в хронологическом порядке оказалась следующая сделка:

- id: 0,
- dateTime: ”2019-05-02T16:25:14”,
- instrument: ”btcusd”,
- tradeType: ”Buy”,
- tradedQuantity: 0.00308,
- tradedQuantityCurrency: ”BTC”,
- tradedPrice: 5392.81999,
- tradedPriceCurrency: ”USD”,

- commission: 0.0000077,
- commissionCurrency: ”BTC”,
- tradeValueId: ”36831516”

Получается, что пользователь совершил покупку 0.00308 BTC за USD, дополнительно уплатив при этом комиссию, в рамере 0.0000077 BTC, из расчёта 5392.81999 USD за 1 BTC, где BTC – биткоин, USD – американский доллар. Это означает, что на момент проведения этой сделки у пользователя на счету уже была некоторая сумма, а именно: $(0.00308 + 0.0000077) \cdot 5392.82 = 20.534240714$ USD. Но так как данных об этом активе не было, получится, что баланс USD у пользователя после учёта этой сделки станет отрицательным, что невозможно при торговле на бирже.

Чтобы предупредить пользователя о существовании этой проблемы, была создана специальная форма, выводящая информацию об объёмах таких отрицательных балансов, что даёт возможность отслеживать эти значения и добавлять необходимые сделки или транзакции вручную.

4.5. Котировки

Для каждой из необходимых метрик нужно хранить информацию о котировках по ряду инструментов. Для этих целей использовался ассоциативный массив, где в качестве ключа выступал инструмент, а в качестве значения – список котировок, каждая из которых имеет следующие поля:

- timestamp - момент времени, в который была актуальна данная котировка
- exchangeRate - отношение стоимостей первого и второго активов из инструмента

Так как в контексте данной задачи расчёт метрик производился для цельных временных промежутков, котировки с сервера запрашивались

только для необходимых на момент расчёта инструментов и только для нужного промежутка времени, не сохраняя их в дальнейшем. Такой подход исключил необходимость поиска котировки для нужного момента времени, позволяя последовательно обрабатывать каждый день торговли и просто проходя по принимаемому массиву котировок.

4.6. Стоимость портфеля

Одно из значений, которое необходимо пользователю – его текущий баланс. Чтобы его получить нужно учесть каждую из совершённых сделок и каждую успешно проведённую транзакцию, в результате чего получается массив значений "Валюта - Объём в портфеле". Перевод в базовую валюту происходит путём умножения объёма актива в конкретный день на актуальную на тот момент котировку. Таким образом, сумма полученных произведений и будет искомой величиной.

Пример Пусть, после учёта всех сделок и транзакций, получилось, что 1 января 2020 года инвестиционный портфель состоял из 3 активов: биткоин, евро,рубль, а в качестве базовой валюты выбран американский доллар. Тогда S – суммарная стоимость портфеля будет получаться путём умножения объёмов на соответствующие котировки(см. рис. 3).

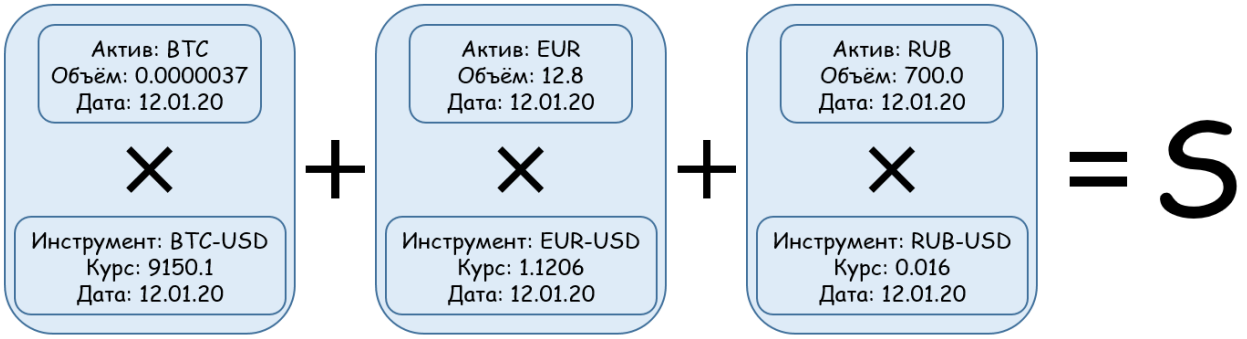

Рис. 3: Расчёт стоимости портфеля.

Можно заметить, что если при расчёте конечной стоимости портфеля все сделки и транзакции учитываются в хронологическом порядке, то промежуточные значения можно использовать для ещё одного из

необходимых графиков - кластерной диаграммы стоимости портфеля от времени. Это следует из того, что каждый отдельный столбец такой диаграммы будет хранить в себе распределение активов внутри портфеля в базовой валюте(см. рис. 4).

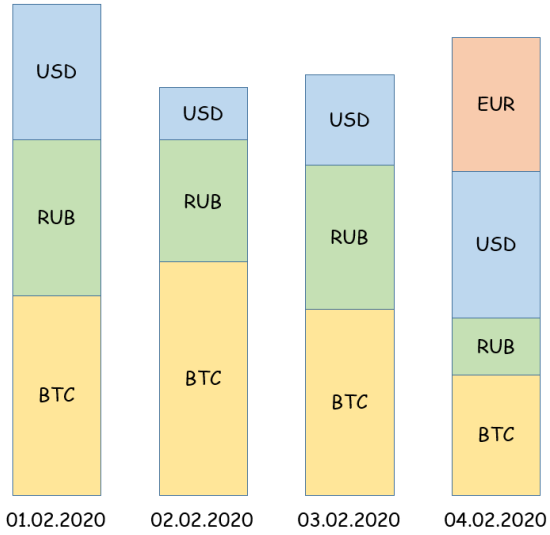

Рис. 4: Расчёт стоимости портфеля.

При этом, разбиение такой диаграммы на отдельные активы, даёт значения для графика стоимости конкретных активов в портфеле(см. рис. 5).

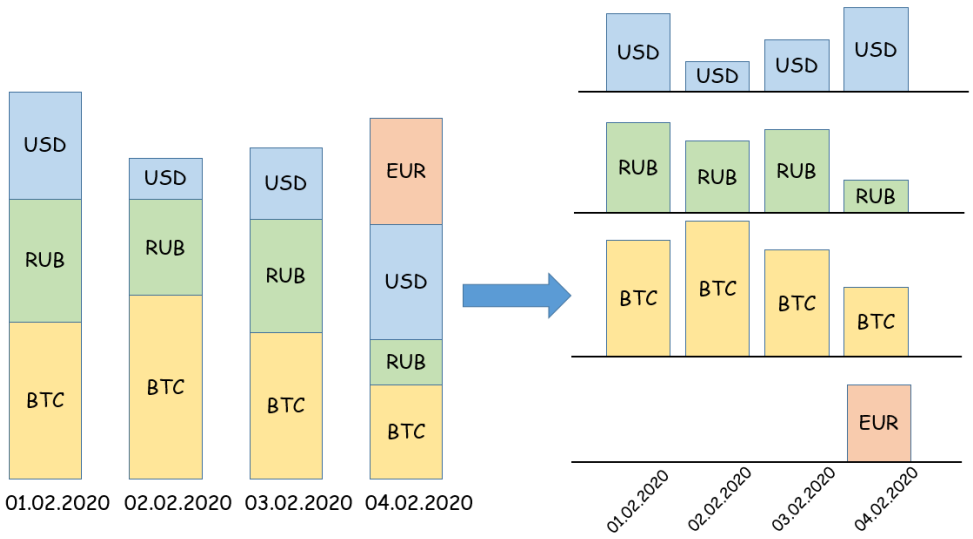

Рис. 5: Расчёт стоимости портфеля.

4.7. Ввод/вывод средств

Для этого графика нужен отсортированный по времени список всех транзакций за выбранный промежуток времени. Для получения необходимых значений нужно вычислить суммарный объём всех транзакций на ввод и на вывод средств, соответсвенно. При этом, объём каждой из транзакций переводится в базовую валюту по курсу, актуальному на момент её проведения.

4.8. Доход

Изначально подсчёт дохода был реализован как расчёт разности между балансами на конец интерсующего промежутка времени и начало.

В процессе разработки приложения выяснилось, что доход на биржах считается одним из трёх ниже представленных способов:

1. LIFO-метод
2. FIFO-метод
3. Метод средневзвешенных цен

LIFO-метод LIFO - Last In, First Out. Идея подсчёта дохода данным методом заключается в том, чтобы первыми продавать активы, купленные последними. Таким образом доход будет равен разнице между ценой продажи и ценой покупки именно последних активов. Данный метод используется в очень маленьком количестве стран, поэтому было принято решение не высчитывать доход таким способом.

FIFO-метод FIFO - First In, First Out.Идея этого метода прямо противоположна предыдущему. Доход будет получаться из разницы стоимости проданного и купленного актива, причём купленного самым первым из имеющихся в портфеле. Этот метод широко применяется во всем мире, но трудно реализуем на клиентской стороне приложения,

поэтому было принято решение перенести расчёт дохода этим методом на серверную часть.

Метод средневзвешенных цен В данном случае цена, по которой покупался актив усредняется, путём вычисления среднего арифметического из всех стоимостей его покупки. Доход же будет равен разнице между стоимостью продажи и покупке актива, предполагая, что покупали его по средней стоимости.

Сложности в реализации вычисления дохода появляются при работе с большими наборами данных. Например, если человек торгует на бирже с помощью специализированных программ, способных проводить несколько сделок в минуту, суммарное количество проведённых в месяц операций может превышать сотни тысяч. Каждую из этих операций необходимо учитывать. При таких объёмах данных высокой точности, вычисление дохода по методам описанным выше, становится очень не эффективно на мобильном устройстве.

Доходом портфеля складывается из суммы доходов активов, входящих в него, а как следствие, все рассуждения выше относятся и к этой метрике.

4.9. Корреляция

Корреляция - величина, принимающая значение от -1 до 1, характеризующая зависимость котировок первого инструмента от котировок второго. Расчётная формула выглядит следующим образом:

$$r_{xy} = \frac{\sum(x_i - \overline{x}) \cdot (y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \cdot \sum(y_i - \overline{y})^2}} \tag{1}$$

где r_{xy} - корреляция,

x_i, y_i - котировки первого и второго инструментов,

$\overline{x}, \overline{y}$ - средние значение котировок.

Для упрощения, пользователю предлагается выбрать всего два актива A и B, зависимость которых необходимо узнать. Отпраляется за-

прос на получения котировок по двум инструментам: ”A-базовая валюта” и ”B-базовая валюта”. Если A или B совпадает с базовой валютой, то в знаменателе расчётной формулы получается 0. Чтобы этого избежать, пользователю будет предложено поменять актив или выбрать другую базовую валюту. После получения котировок вычисляется, непосредственно, значение корреляции.

4.10. Нормированный график сравнения котировок двух активов

Для стандартизации данных использовалась следующая формула:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2}$$

Линейное преобразование было выбрано потому что максимумы и минимумы в наборах данных принимают конечные значения.

Таким образом на координатной плоскости возможно отобразить несколько графиков, сильно отличающихся друг от друга диапазонами значений (рис. 6).

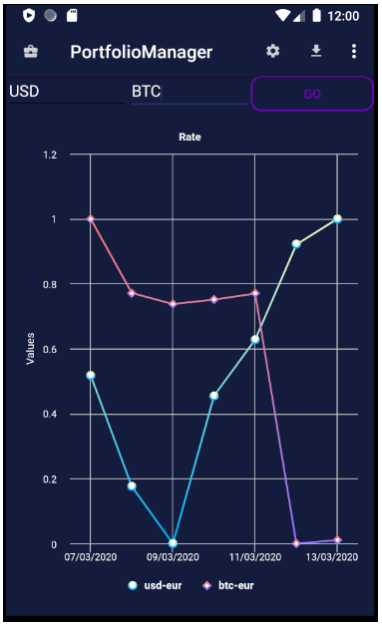

Рис. 6: График сравнения курсов.

4.11. Визуализация

Для визуализации состава портфеля с помощью круговой диаграммы на первом этапе была выбрана библиотека AACharts, но как оказалось на практике, этот вариант был неудобен в плане реализации и внешнего вида, поэтому было принято решение использовать библиотеку MPAndroidChart. Оба варианта представлены на рисунке 7.

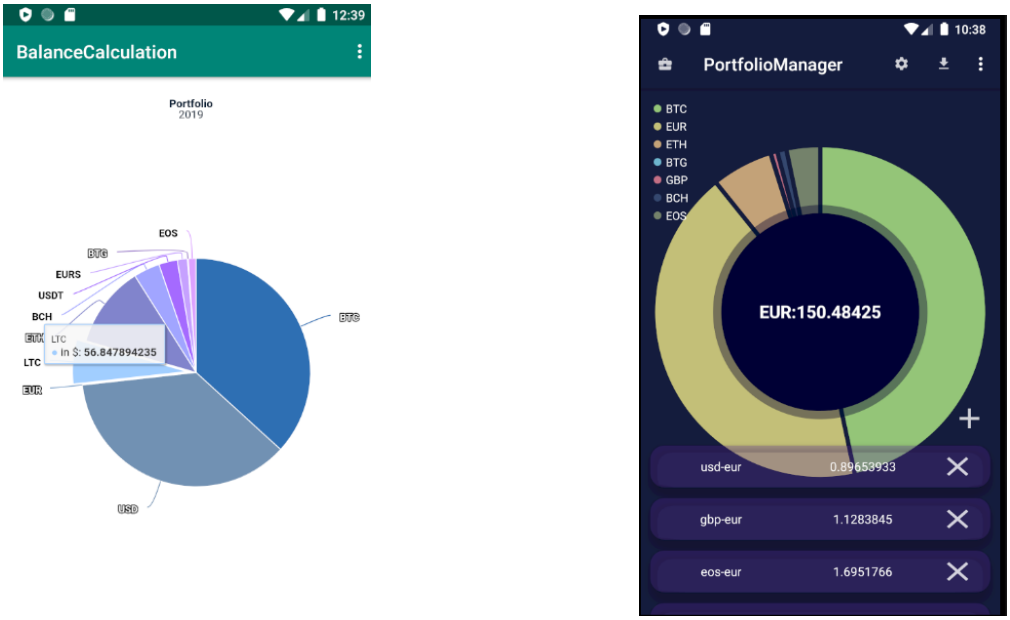

Рис. 7: Диаграммы состава портфеля.

На начальный экран также были добавлены котировки по активам, входящим в портфель к базовой валюте. При нажатии на любую из них открывается график котировок по выбранному инструменту. Для визуализации этого и остальных криволинейных графиков, а именно:

1. Доход актива от времени
2. Доход портфеля от времени
3. Объём выбранного актива в портфеле от времени
4. Нормированный график котировок двух инструментов
5. График корреляции

использовалась библиотека AACharts. К некоторым из них, со временем, была добавлена возможность выбора периода отображения данных: 1, 2 или 3 месяца (рис. 8).

Заметим, что в зависимости от визуализируемой метрики, площадь под графиком либо закрашивается (см. график котировок), либо остаётся прозрачной, как в случае нормированного графика котировок, приведённого выше. Это объясняется тем, что при отображении нескольких, возможно пересекающихся, графиков на одной плоскости, закрашенная площадь одного может полностью перекрыть другой. Тем самым, получение необходимой информации становится невозможным.

Следующий тип графиков - свечной, использовался для визуализации распределения активов от времени и ввода средств. В последнем, для удобства, была добавлена возможность менять параметры отображения: по дням или по месяцам (рис. 9).

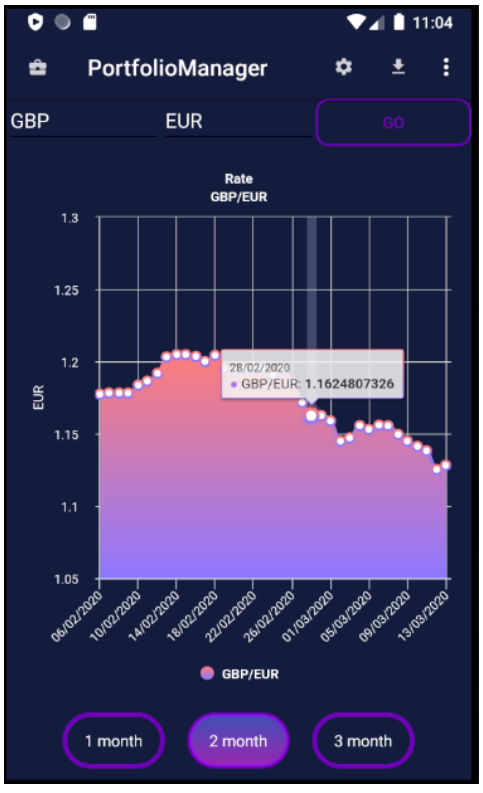

Рис. 8: График актуальных котировок.

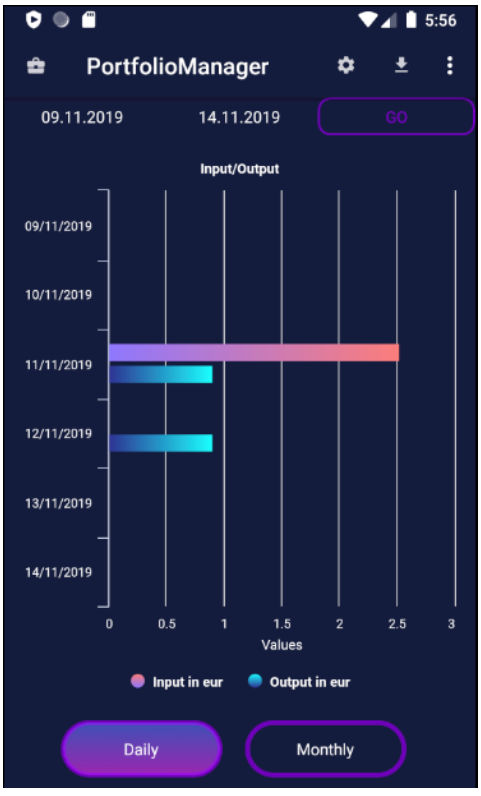

Рис. 9: График ввода/вывода средств.

5. Тестирование

Для каждой из реализованных метрик были написаны юнит-тесты, основанные на реальных данных, выгруженных с биржи DSX. Набор состоял из 1200 транзакций и 560 сделок. История торговли была взята за промежуток времени в 4 года, имея как периоды активной торговли, так и периоды полного отсутсвия деятельности. При этом среди транзакций были:

- успешно проведённые/отменённые
- без комиссии/с комиссией
- задействующие различные активы

Среди сделок также была большая вариативность, благодаря которой расчёты производились для большого набора инструментов, учитывая как сделки по покупке, так и по продаже активов со всевозможными значениями комиссий.

В дополнение к ним, тесты проводились на самостоятельно созданном наборе. Он был сделан для проверки корректной работы с краевыми значениями временных промежутков и работы с данными, состоящими только из одного типа операций: сделок/транзаций.

Такой подход позволил проверить работу приложения в условиях, максимально приближенных к реальным, не упуская при этом редко возникающих в торговле случаев.

Заключение

В процессе выполнения данной курсовой работы, были получены следующие результаты:

- Изучена предметная область
- Произведён анализ существующих решений
- Выделены необходимые для портфельного менеджмента метрики. Рассмотрены возможные способы их визуализации.
- Продумана архитектура приложения
- Реализовано и протестировано мобильное приложение, реализующее расчёт всех выделенных метрик

В дальнейшие планы развития приложения входят:

- Реализация поддержки большего количества брокеров и бирж
- Добавление новых метрик
- Улучшение качества визуализации

Ссылка на репозиторий:
https://github.com/dsx-tech/student-project-balances-android

Благодарность

Выражаю благодарность компании DSX, которой была предоставлена тема работы и тестовые данные, а конкретно Филиппу Петровичу Долголеву, Сергею Сергеевичу Крамскому и Анне Валерьевне Ивановой, курировавшим проект.

Список литературы

[1] AACharts. — 2020. — URL: https://github.com/AAChartModel/AAChartCore-Kotlin (дата обращения: 12.05.2020).

[2] Android. — 2020. — URL: https://www.android.com (дата обращения: 12.05.2020).

[3] AndroidCharts. — 2020. — URL: https://github.com/HackPlan/AndroidCharts (дата обращения: 12.05.2020).

[4] AnimatedPieView. — 2020. — URL: https://github.com/razerdp/AnimatedPieView (дата обращения: 12.05.2020).

[5] Blockfolio. — 2020. — URL: https://blockfolio.com (дата обращения: 12.05.2020).

[6] DSX. — 2020. — URL: https://dsxglobal.com (дата обращения: 12.05.2020).

[7] Dexter. — 2020. — URL: https://github.com/Karumi/Dexter (дата обращения: 12.05.2020).

[8] Future Adviser. — 2020. — URL: https://futureadvisor.com (дата обращения: 12.05.2020).

[9] Google. — 2020. — URL: https://www.google.com (дата обращения: 12.05.2020).

[10] GraphView. — 2020. — URL: https://github.com/jjoe64/GraphView (дата обращения: 12.05.2020).

[11] HelloCharts. — 2020. — URL: https://github.com/lecho/hellocharts-android (дата обращения: 12.05.2020).

[12] Investment Account Manager. — 2020. — URL: https://investmentaccountmanager.com (дата обращения: 12.05.2020).

[13] JGraph. — 2020. — URL: https://github.com/ZuYun/Jgraph?utm_source=android-arsenal.com (дата обращения: 12.05.2020).

[14] JSON Web Token. — 2020. — URL: https://jwt.io (дата обращения: 12.05.2020).

[15] Java. Programming language. — 2020. — URL: https://www.java.com (дата обращения: 12.05.2020).

[16] Kotlin. Programming language by JetBrains. — 2020. — URL: https://kotlinlang.org (дата обращения: 12.05.2020).

[17] MPAndroidChart. — 2020. — URL: https://github.com/PhilJay/MPAndroidChart (дата обращения: 12.05.2020).

[18] Mint. — 2020. — URL: https://mint.com (дата обращения: 12.05.2020).

[19] Retrofit2. — 2020. — URL: https://square.github.io/retrofit (дата обращения: 12.05.2020).

[20] SigFig Portfolio Tracker. — 2020. — URL: https://sigfig.com (дата обращения: 12.05.2020).

[21] Statistica. Mobile operating systems' market share worldwide from January 2012 to December 2019. — 2020. — URL: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009 (дата обращения: 12.05.2020).

[22] Tinkoff. — 2020. — URL: https://www.tinkoff.ru (дата обращения: 12.05.2020).