Содержание

1.Введение 3
2. Основная логика работы приложения. 4
2.1.Регистрация 4
2.2 Экран ввода PIN-кода пользователя 4
2.3 Экран портфель 5
2.4 Экран Обзор рынка 6
2.5 Экран Истории. 6
2.6 Экран профиль. 7
2.7 Экран Торгового пароля 8
2.8 Экран обзора актива 8
2.8.1 Обзор 8
2.8.2 История 9
3. Общие сведения о приложении 10
4. Стек технологий 12

Заключение. 13

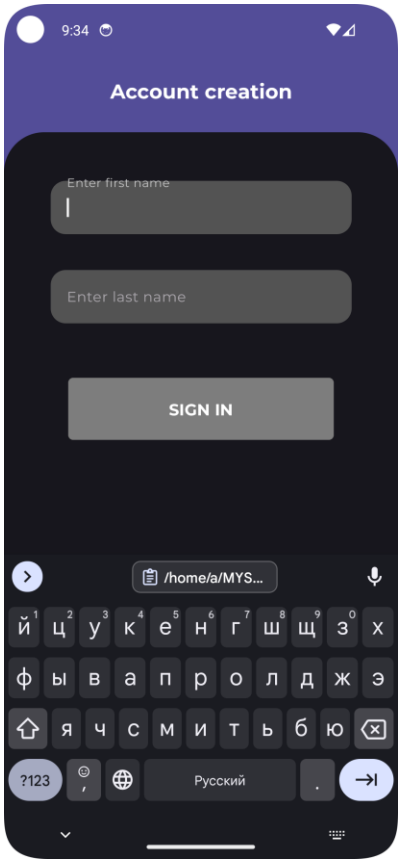
1. Введение

В рамках настоящей работы необходимо реализовать Android-приложение для инвестиций на языке программирования высокого уровня Kotlin с использованием выбранных API. Реализованное в ходе практики приложение отвечает следующим требованиям:

1. Безопасность данных пользователя
2. Стек технологий, использующийся в приложении, полностью реализует знания, полученные в ходе практики
3. Задействовать минимум одно API (в приложении задействовано 2 API)

2. Основная логика работы приложения

2.1 Экран регистрации пользователя

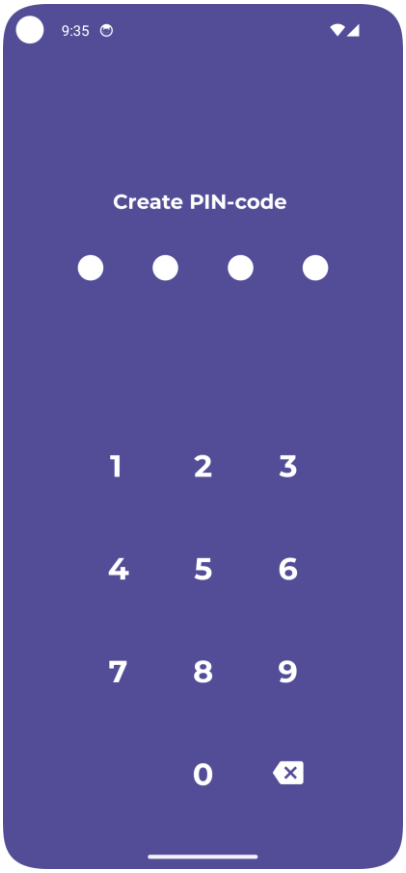


В поля вводятся Имя и Фамилия пользователя, производится валидация по правилам:
Поле может состоять из букв и цифр, не должно быть пустым и не заканчиваться пробелом. Двойные имена через дефис, а также имена с цифрами (имя 6-го сына Илона Маска - X Æ A-12)
Проверки на валидность:

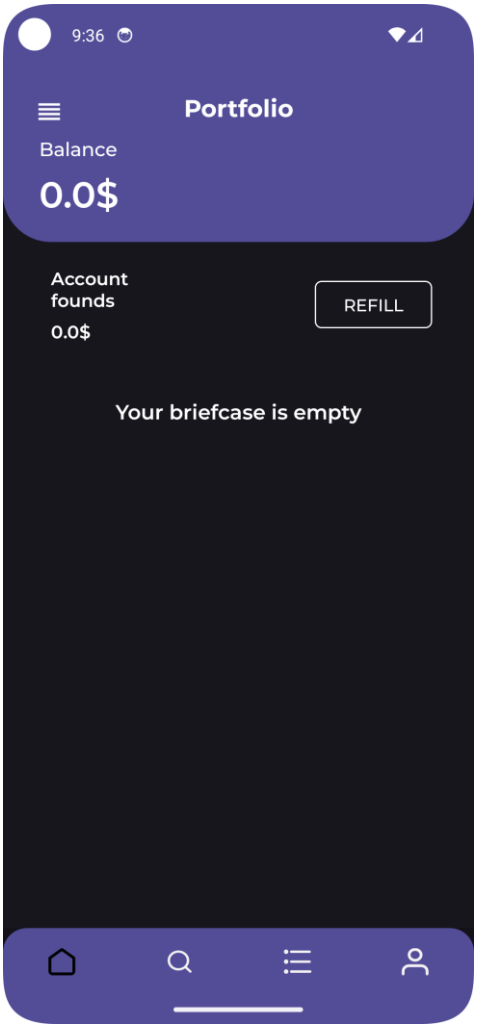
1. Если поле пустое, выводится красное сообщение под полем: “Поле не может быть пустым”.
2. Если поле содержит пробелы в начале или в конце: "Содержит пробелы в начале или конце”

2.2 Экран ввода PIN-кода пользователя

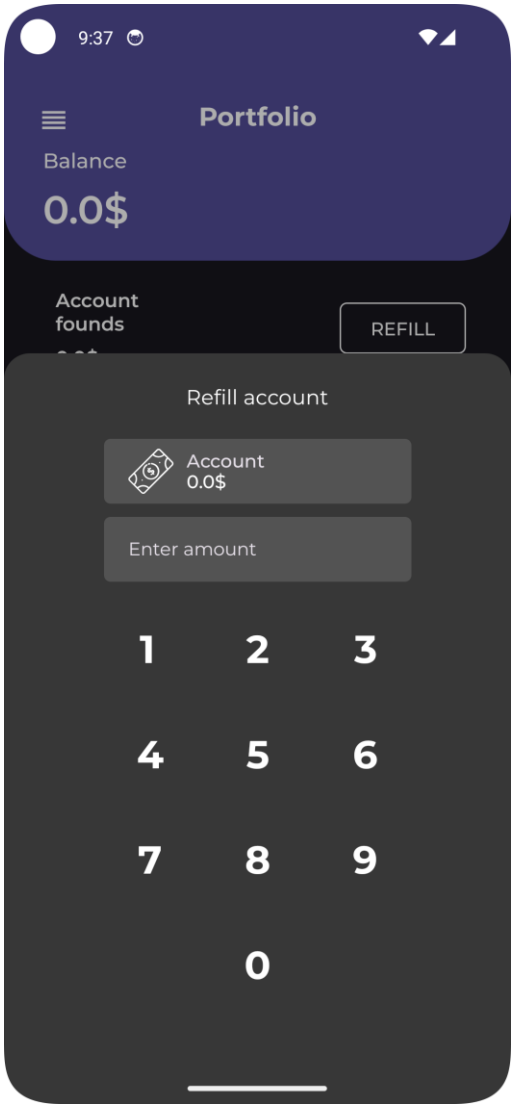
Этот экран используется для создания PIN-кода, а также для его последующей аутентификации при входе.



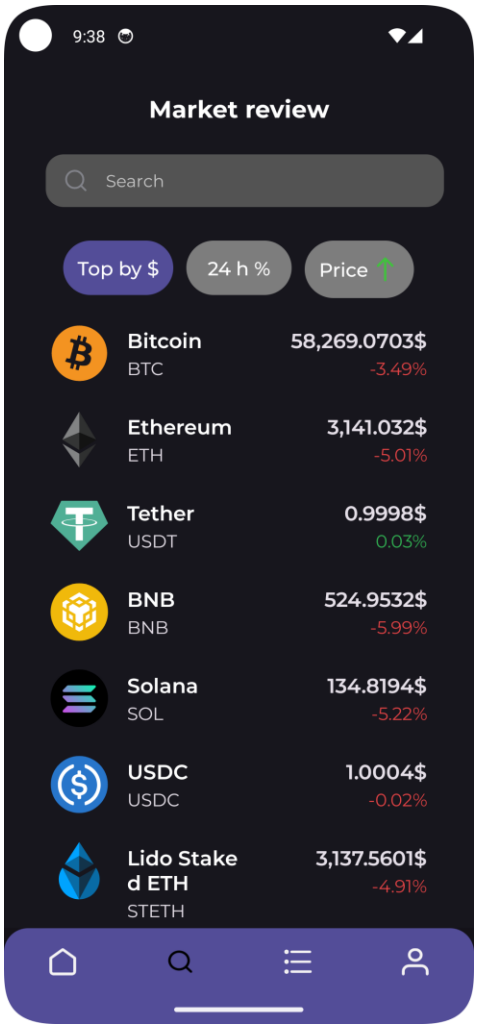
2.3 Экран портфель



По кнопке Refill открывается диалог пополнения счета,
По навигационной панели можно перемещаться к другим экранам.



2.4 Экран Обзор рынка

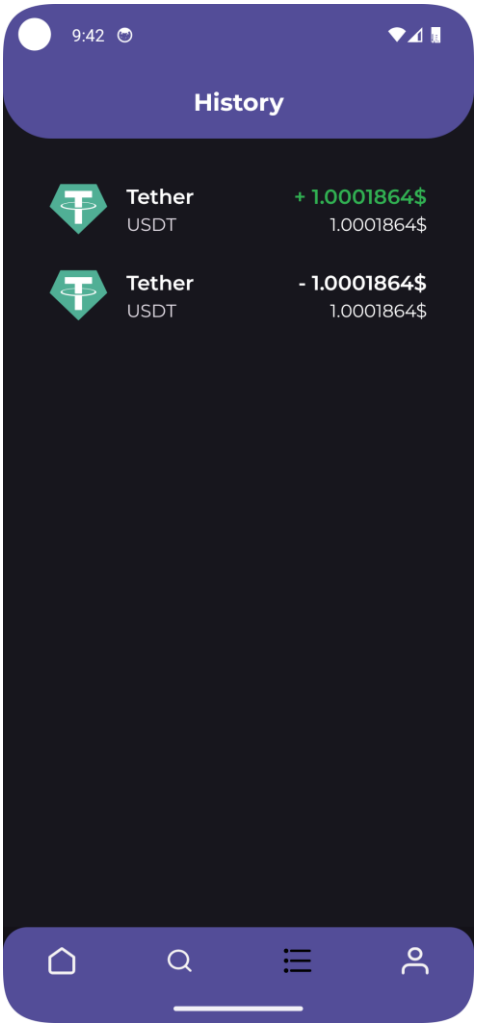


В поле поиска можно ввести какое-нибудь название Криптовалюты и на экране покажутся результаты поиска.
По трем вкладкам можно перемещаться, меняется сортировка списка: По капитализации, по росту за последние 24 часа и по цене (можно нажать дважды, сортировка по убыванию и возрастанию)

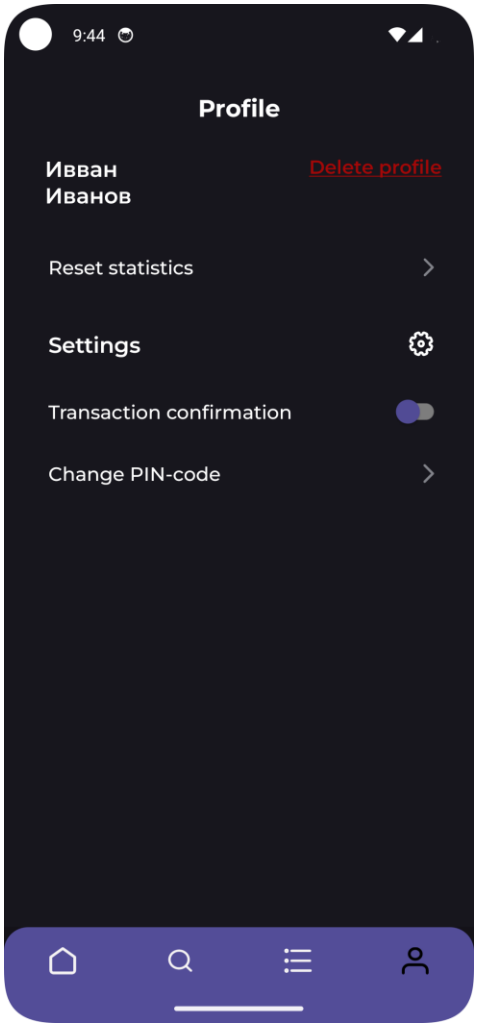
При клике на любой актив открывается экран Обзор актива

2.5 Экран Истории

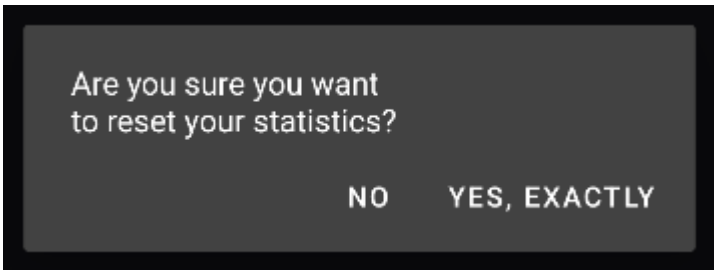
Здесь показана общая история покупок. При клике на любой актив открывается экран Обзор актива

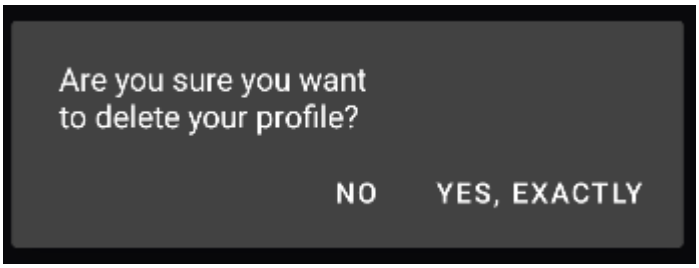


2.6 Экран профиль



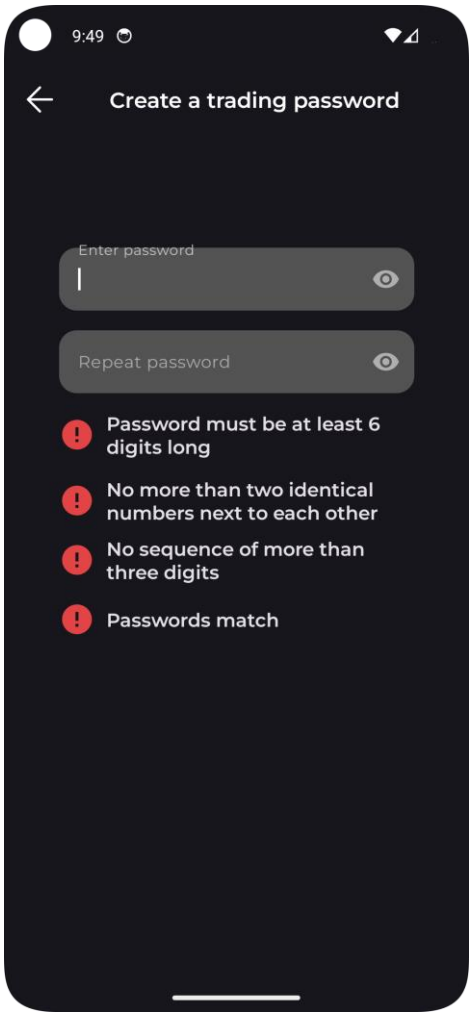
Здесь показана информация о пользователе, а также разные настройки: удаление профиля, сброс статистики, подтверждение сделок и смена PIN-кода.
Диалоги, показывающиеся при нажатии на «Delete profile» и «Reset statictics»





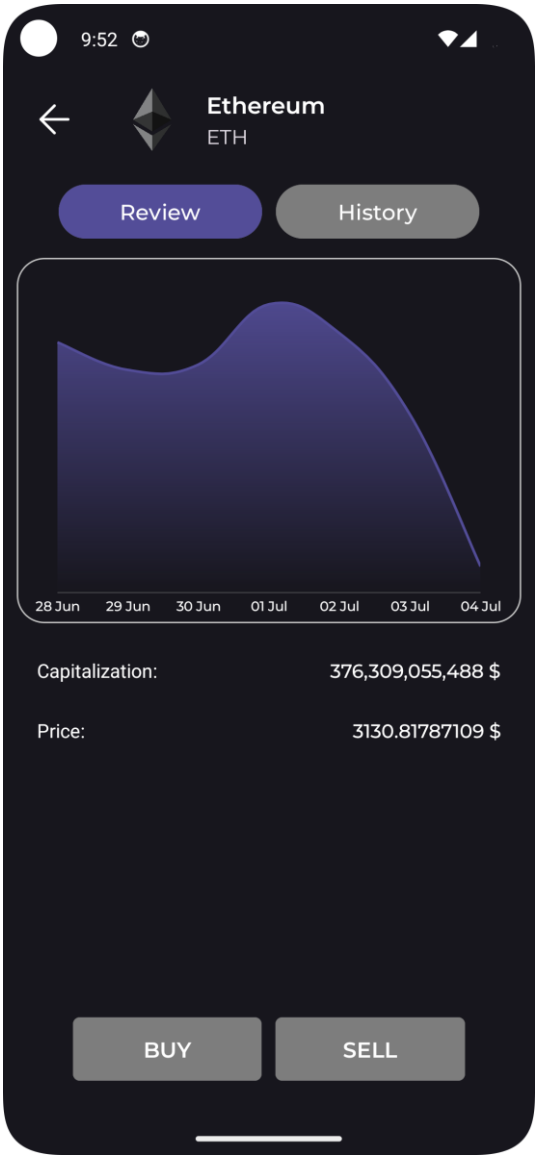
При клике на «Change PIN-code» и «Transaction confirmation» производится переход на экран Входа и экран Торгового пароля

2.7 Экран Торгового пароля



Здесь можно создать торговый пароль – пароль, который необходимо ввести, чтобы совершить сделку (покупку/продажу). Торговый пароль должен соответствовать указанным на скриншоте правилам:

1. Не менее 6 символов
2. Не более двух одинаковых цифр рядом
3. Нет последовательности более трех цифр (123, 234 – можно, 1234, 3456 – нельзя)
4. Пароля совпадают

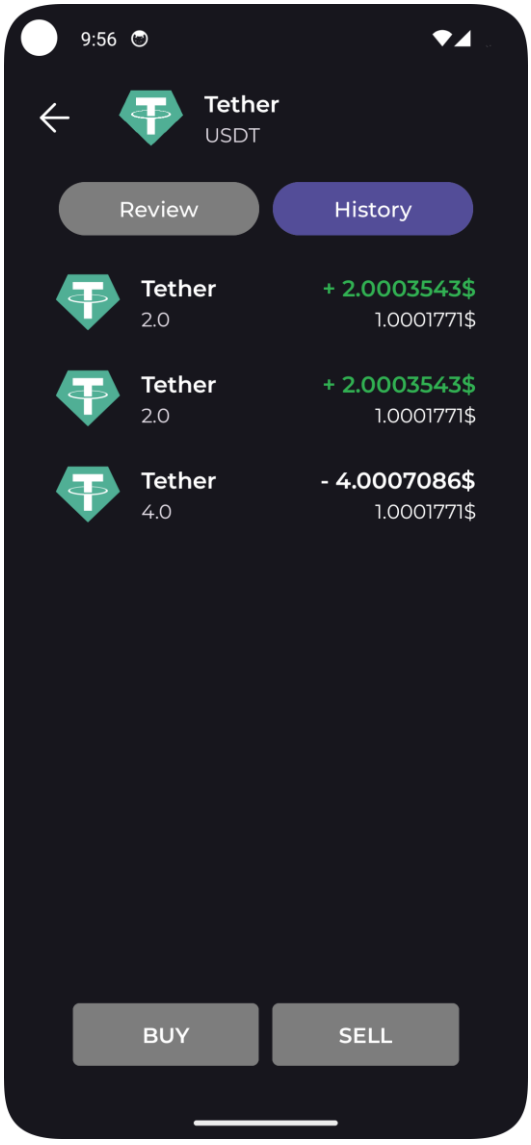


2.8 Экран обзора актива

Здесь 2 вкладки: Обзор и История.

2.8.1 Обзор

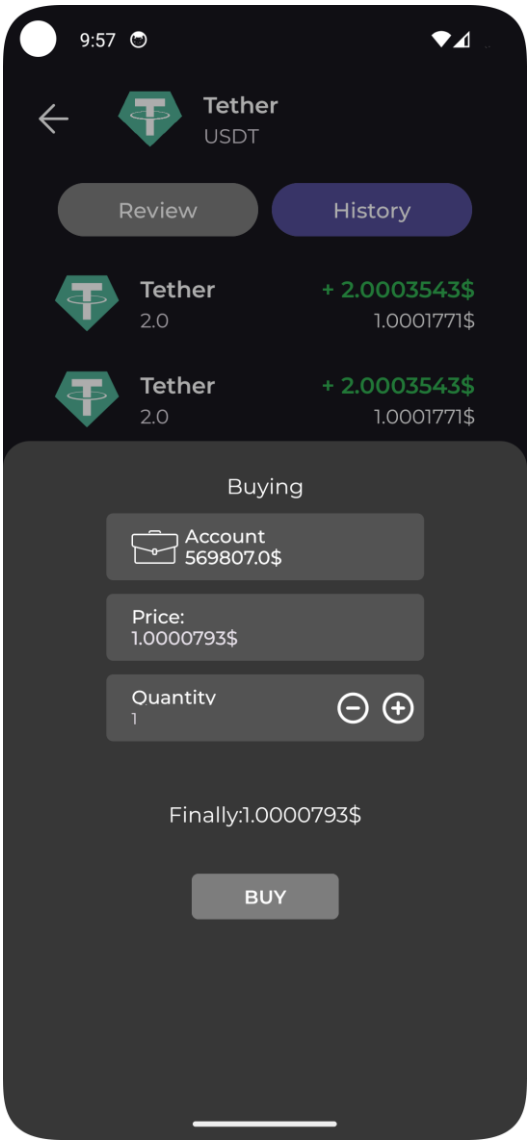
На обзоре актива можно увидеть информацию о нем: цену, обновляющуюся в реальном времени (как и на всех экранах) и капитализацию, а также купить/продать



2.8.2 История

Здесь можно увидеть историю сделок с одним активом

На кнопки купить/продать открываются диалоги покупки/продажи

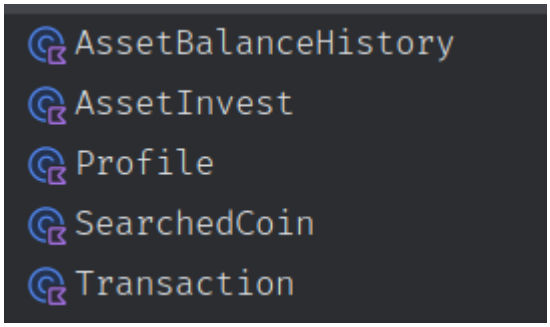


3. Общие сведения о приложении

Приложение написано на языке программирования Kotlin – ститически типизированный, объектно-ориентированный язык. Производный от Javaб Kotlin был объявлен приоритетным языком программирования для Android-разработки в 2019, до 2019-го года стандартом написания Android приложений был язык Java.

Среда программирования – Android Studio – IDE для написания Android-приложений. В качестве системы контроля версий при написании проекта используется git, а в качестве удаленного репозитория для фиксации изменений – GitHub.

В СУБД используется объектно-реляционная система. Ниже представлен список сущностей, которые используются для работы приложения. Концепция взаимодействия с базой данных написана на ORM Room – нативный, и с недавнего времени мультиплатформенный фреймворк для реализации работы с локальными базами данных. Работа с сущностями происходит через DAO – data access object, который содержит в себе набор оцераций на языке SQL. Ниже также представлено, как выглядит операция взаимодействия с сущностью AssetBalanceHistory.

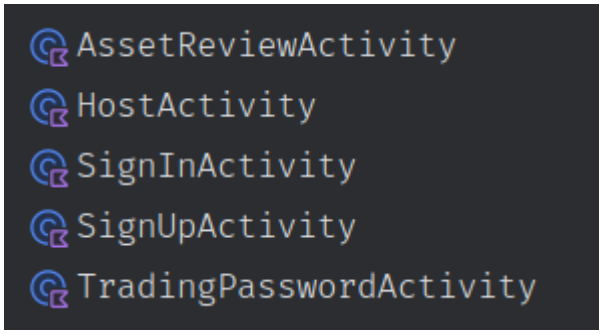


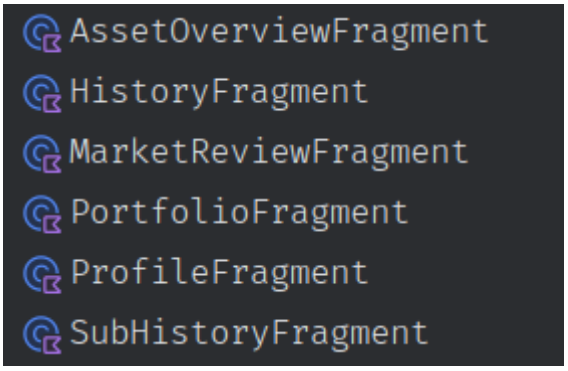

```
Vafeen

@Query("SELECT * FROM AssetBalanceHistory WHERE date = :date LIMIT 1")

suspend fun getByDate(date: Date): AssetBalanceHistory?
```

Подход к написанию программы – объектно ориентированное программирование. Главный класс — «MainActivity», с которого начинается запуск приложения. Данный класс наследуется от класса AppCompatActivity – класс, отвечающий за жизненный цикл экрана приложения, т.е. Activity.

```
23  class MainActivity : AppCompatActivity() {
```

За каждый экран приложения или его часть отвечает Activity, Fragment или Dialog. Почти все экраны, кроме экрана с нижним меню – отдельные Activity. Экран с нижним меню – Activity, которое содержит в себе элементы типа Fragment. Схема отношений Activity-Fragment: AssetReviewActivity {AssetOverviewFragment, SubHistoryFragment}, HostActivity {HistoryFragment, MarketReviewFragment, PortfolioFragment, ProfileFragment}. Остальные Activity являются свободными от Fragment'ов.





Структуры данных, использующиеся в приложении: List, Queue. List используется в библиотечном виде, а Queue реализована в виде очереди с постоянным размером: если при добавлении в очередь исходный размер данных превышает заданный, часть данных удаляется из очереди, пока не будет достигнут размер меньший или равен заданному.

Шаблон проектирования архитектуры приложения – MVVM (Model, View, ViewModel) — подход к написанию, при котором взаимодействие объектов с интерфейсом происходит через ViewModel. Справка:

- *Модель* (англ. *Model*) (так же, как в классической MVC) представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.
- *Представление* (англ. *View*) — графический интерфейс (окна, списки, кнопки и т. п.). Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновлённое значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.
- *Модель Представления* (англ. *ViewModel*) — с одной стороны, абстракция Представления, а с другой — обёртка данных из Модели, подлежащих связыванию. То есть, она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

4. Стек технологий, использующихся при написании приложения

Retrofit (сбор информации о цене и капитализации ЦБ в реальном времени)
Room (базы данных)
Coroutines (асинхронные операции)
Biometric (аутентификация пользователей по отпечатку пальца)
Constraintlayout (макет для написания интерфейса приложения)
Hilt (фреймворк для внедрения зависимостей)
Mpandroidchart (фреймворк для рисования графиков)
Coil (фреймворк для загрузки картинок ценных бумаг)
ValueAnimator (объект создания анимации изменения свойств объекта)

Заключение

В рамках настоящей работы было реализовано Android-приложение для инвестиций на языке программирования высокого уровня Kotlin с использованием выбранных API.