

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

Отчет

по курсовой работе

по дисциплине **«Разработка мобильных приложений»**

название приложения: Minvest

Автор: Динь Чыонг Лам

Факультет: ФПИиКТ

Группа: Р33202

Преподаватель: Ключев А.О.



Санкт-Петербург 2023

Оглавление

Введение.....	4
Краткое описание:.....	4
Цель:.....	4
Техническое задание	4
Функциональные требования	4
Макет интерфейса	5
Рисунок 1 Главная страница приложения:.....	9
Рисунок 2	9
Рисунок 3 Цена акции:	9
Рисунок 4 История инвестиций:	9
Рисунок 5 Детали инвестиции:	9
Рисунок 6 История сделок:	9
Рисунок 7 Покупка акций:	9
Палитра цветов	9
Архитектура системы.....	10
Стек технологий	10
Слои системы	11
Взаимодействие с пользователем и контроллеры	11
Взаимодействие с БД.....	11
Бизнес-процессы	12
Требования к безопасности	12
Структура базы данных.....	12
Описание реализации.....	14
Тестирование	15
Заключение	16

Сылка: <https://github.com/kaitokid2302/Minvest>

Введение

Краткое описание:

Проект Minvest — это проект инвестирования в акции, позволяющий пользователям использовать виртуальные деньги для покупки акций по реальным ценам. Приложение предоставит пользователям информацию об акциях и позволит пользователям использовать его для создания инвестиций, содержащих множество различных акций, и отслеживать прибыльность этих инвестиций. Цены на акции взяты из источника API Rapid API — Twelve Data. Twelve Data — это источник API-интерфейсов цен на акции, предоставляющий множество фондовых бирж, таких как NASDAQ, CXA, NYSE и многие виртуальные валюты, такие как Биткойн, Ethereum... Но в основном основное внимание уделяется NASDAQ.

Цель:

Создать приложение, которое позволит пользователям инвестировать в акции виртуальными деньгами и отслеживать прибыль от этих инвестиций, практиковаться в реальном инвестировании, где пользователи могут потерять реальные деньги, если они инвестируют неэффективные инвестиции, а также улучшить понимание пользователями инвестиции в акции.

Техническое задание

Функциональные требования

1. Поиск акций по названию:

Пользователь может искать название акции или название компании и получать в ответ список компаний с названиями или акциями, содержащими введенные символы.

2. Сортировка акций по цене, по названию, по убыванию или возрастанию:

Пользователь может сортировать акции по цене, по названию, по убыванию или возрастанию, чтобы легче найти акции, которые он хочет купить.

3. Сортировка прибыли от истории сделок:

Пользователь может сортировать историю сделок по прибыли, по убыванию или возрастанию, чтобы легче отслеживать свою историю сделок.

4. Просмотр информации об акциях:

Нажав на конкретную компанию, пользователь может увидеть текущую цену акции в реальном времени.

5. Создание, удаление инвестиционного портфеля:

Пользователь может создать или удалить инвестиционный портфель, если создает, пользователь выберет акции, в которые он хочет инвестировать, и количество акций, которые он хочет купить. Количество покупаемых акций не ограничено, и пользователь может купить акции в минимальном количестве - одна акция.

6. Удаление сделки:

Пользователь может удалить сделку из своей истории сделок, и количество акций также будет удалено из соответствующего инвестиционного портфеля.

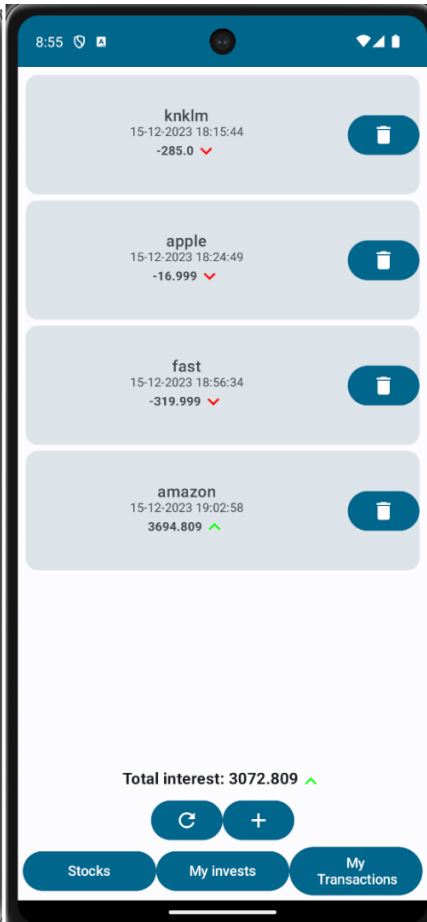
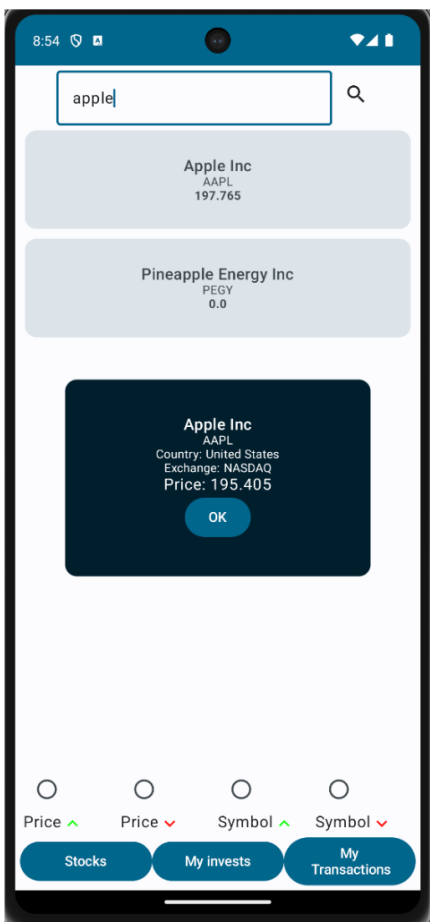
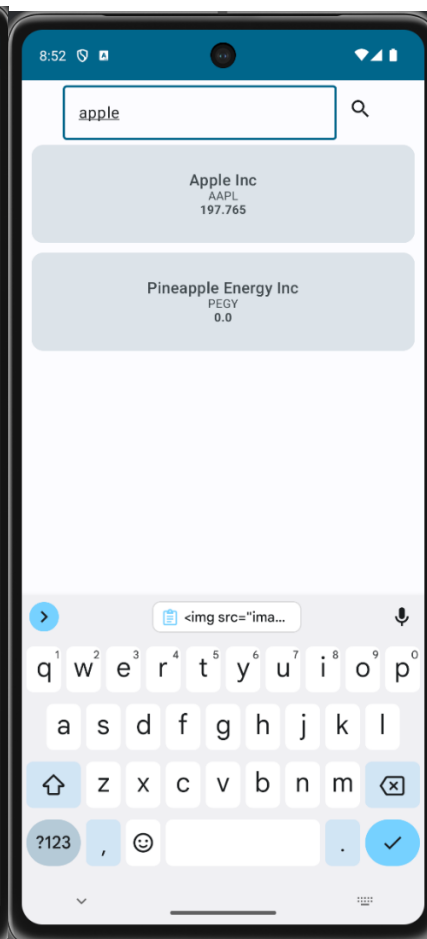
7. Просмотр прибыли от инвестиционного портфеля:

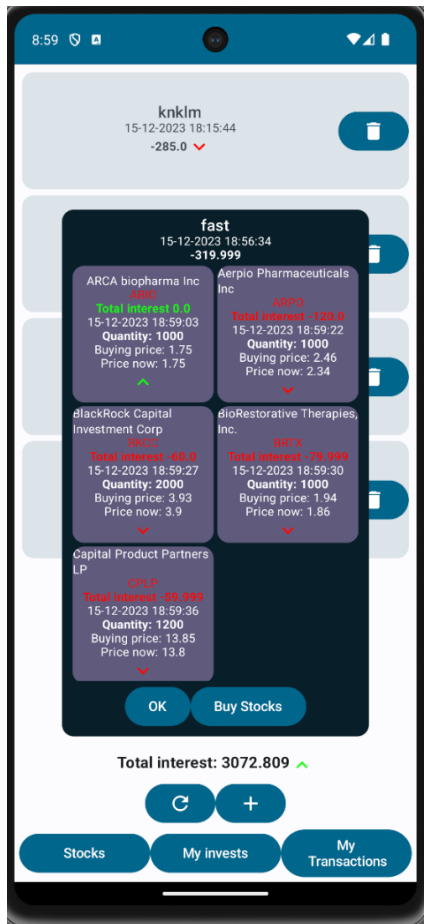
Пользователь может увидеть прибыль от своего инвестиционного портфеля и увидеть прибыль от каждой акции в этом инвестиционном портфеле в любое время, прибыль рассчитывается по разнице между текущей ценой акции и ценой при покупке этой акции, умноженной на количество этих акций. И обновляется в реальном времени.

8. Просмотр прибыли от сделки:

Пользователь может увидеть прибыль от своей сделки, прибыль рассчитывается по разнице между текущей ценой акции и ценой при покупке этой акции, умноженной на количество этих акций. И обновляется в реальном времени.

Макет интерфейса





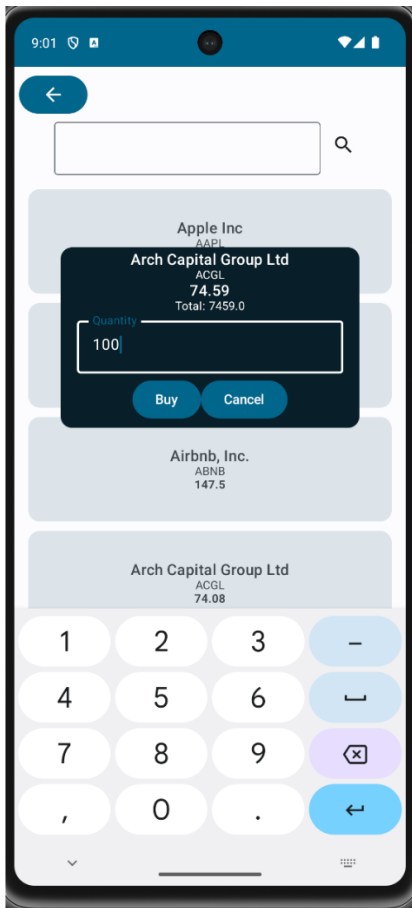


Рисунок 1 Главная страница приложения:

Пользователь увидит список акций и соответствующих компаний, здесь пользователь может сортировать по имени, цене или искать по названию компании или коду акции.

Рисунок 2

Поиск акций: Пользователь может искать акции по названию компании или коду акции и получит список компаний с названиями или кодами акций, содержащими введенные символы.

Рисунок 3 Цена акции:

После выбора компании пользователь может увидеть текущую цену акции этой компании в реальном времени.

Рисунок 4 История инвестиций:

Пользователь может просмотреть историю своих инвестиций и увидеть прибыль каждой из них. Зеленая стрелка, указывающая вверх, показывает рост прибыли, красная стрелка, указывающая вниз, показывает снижение прибыли. Пользователь может сортировать историю инвестиций по прибыли, по убыванию или возрастанию. Или может удалить инвестицию, есть кнопка обновления для обновления прибыли от инвестиции.

Рисунок 5 Детали инвестиции:

Пользователь может просмотреть детали инвестиции и увидеть прибыль каждой акции в этой инвестиции в любое время, прибыль рассчитывается по разнице между текущей ценой акции и ценой при покупке этой акции, умноженной на количество этих акций. И обновляется в реальном времени.







Рисунок 6 История сделок:

Пользователь может просмотреть историю своих сделок и увидеть прибыль каждой из них. Зеленая стрелка, указывающая вверх, показывает рост прибыли, красная стрелка, указывающая вниз, показывает снижение прибыли. Пользователь может сортировать историю сделок по прибыли, по убыванию или возрастанию. Или может удалить сделку, есть кнопка обновления для обновления прибыли от сделки.

Рисунок 7 Покупка акций:

Пользователь выбирает количество акций, которые хочет купить, минимум - одна акция, общая стоимость будет отображаться, и пользователь может купить акции, нажав на кнопку покупки. Если пользователь передумает, он может нажать на кнопку отмены, чтобы отменить сделку, или может нажать в любом месте, чтобы закрыть окно покупки акций.

Палитра цветов

Name of Color	RGB	Color Sample
Green	rgb(3, 255, 7)	 
Dark Blue	rgb(0, 102, 139)	
White	rgb(252, 252, 255)	
Light Grey	rgb(220, 227, 233)	
Red	rgb(255, 0, 0)	

Архитектура системы
Стек технологий

Kotlin - основной язык программирования

Android Studio - IDE

Retrofit - библиотека для вызова API

Room - библиотека для взаимодействия с базой данных

Moshi - библиотека для преобразования JSON в объект

Git - система управления версиями

Gradle - система управления зависимостями

Слои системы

Взаимодействие с пользователем и контроллеры

1. Нажмите на кнопку поиска
2. Нажмите на кнопку сортировки, по имени, прибыли, цене, по убыванию или возрастанию
3. Нажмите на компанию
4. Посмотрите цену акций этой компании
5. Посмотрите портфель инвестиций и данные этих инвестиций
6. Посмотрите историю сделок
7. Нажмите на кнопку создания инвестиционного портфеля
8. Нажмите на кнопку покупки акций
9. Удалите инвестицию, удалите сделку

Взаимодействие с БД

1. Обновление данных о портфеле инвестиций, истории сделок и прибыли от инвестиций, прибыли от сделок

2. Поиск акций по названию компании или коду акции, список будет обновляться во время ввода пользователя, не нужно нажимать enter или кнопку поиска
3. Удаление инвестиции, удаление сделки
4. Создание инвестиции, покупка акций

Бизнес-процессы

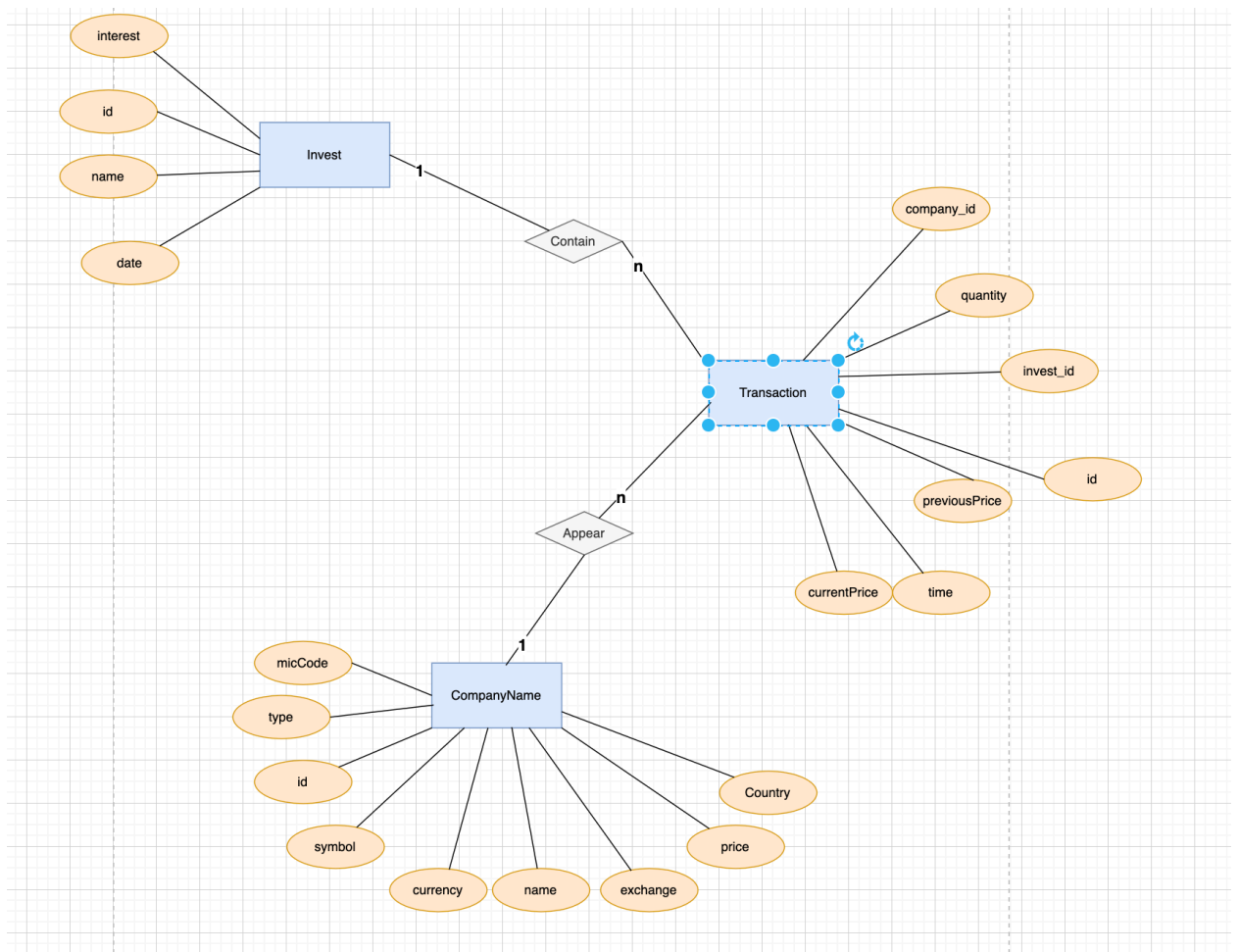
1. Обновление данных.
2. Анализ инвестиционного портфеля, сделок пользователя
3. Сводная информация о прибыли от инвестиций, прибыли от сделок по различным критериям

Требования к безопасности

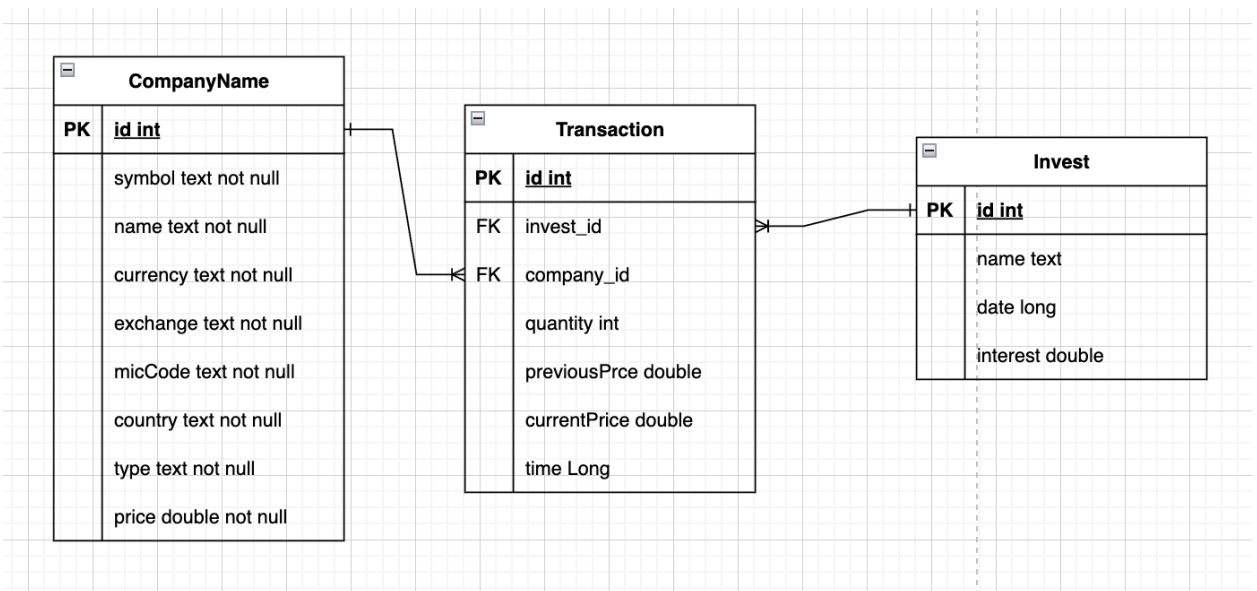
Приложение не будет собирать никакие данные пользователя, и все данные об инвестициях будут храниться только на устройстве пользователя и не будут передаваться никому другому.

Структура базы данных

Инфологическая модель



Даталогическая модель



Описание реализации

В этом проекте я использовал архитектуру MVVM и Jetpack Compose для реализации.

- Каталог Composable: здесь содержатся компоненты для пользовательского интерфейса. Каталог Data содержит сущности, методы Dao и базу данных.
- Каталог Network: здесь содержатся методы для вызова API. 1 класс для хранения информации, полученной из Json при вызове API, используется библиотека Retrofit для сетевых запросов и Moshi для преобразования Json в объект.
- Файл StockViewModel: это viewmodel в модели MVVM, где обрабатываются логика приложения и вызываются методы для запроса к базе данных и вызова API.
- Каталог res: здесь содержатся все ресурсы для приложения, включая изображения, строки, цвета, стили, ...
- Файл sortBy: это запечатанный класс, содержащий поля для сортировки акций по имени, цене, прибыли, по убыванию или возрастанию.
- Файл android manifest: это самый важный файл приложения, содержащий информацию о приложении, такую как имя приложения, версия, права доступа (сеть, база данных, ...), активности, службы, ...

Тестирование

Здесь я использую junit для тестирования, чтобы проверить правильность вызова API.

```
@Test
fun callAPI() {
    var cur = Service.companyName(Service.provideRetrofit())._getPrice(symbol = "AAPL")
    while(cur.isSuccessful==false){
        Credentials.changeToken()
        cur = Service.companyName(Service.provideRetrofit())._getPrice(symbol = "AAPL")
    }
    // assert price is not null
    assert(cur.body()?.price!=null)
}

new *
fun call10APIandFilter(){
    for(i in 1..10){
        var cur = Service.companyName(Service.provideRetrofit())._getPrice(symbol = "AAPL")
        while(cur.isSuccessful==false){
            Credentials.changeToken()
            cur = Service.companyName(Service.provideRetrofit())._getPrice(symbol = "AAPL")
        }
        // assert price is not null
        assert(cur.body()?.price!=null)
    }
}
```

```
fun findMaximumPrice(){
    var max = 0.0
    var maxSymbol = ""
    for(i in 1..10){
        var cur = Service.companyName(Service.provideRetrofit())._getPrice(symbol = "AAPL")
        while(cur.isSuccessful==false){
            Credentials.changeToken()
            cur = Service.companyName(Service.provideRetrofit())._getPrice(symbol = "AAPL")
        }
        // assert price is not null
        assert(cur.body()?.price!=null)
        if(cur.body()?.price!!.toDouble() > max){
            max = cur.body()?.price!!.toDouble()
            maxSymbol = cur.body()?.status.toString()
        }
    }
    assert(max > 0.0)
}
```

Заключение

Благодаря этому проекту я многому научился, такому как архитектура MVVM, Clean Architecture, Jetpack Compose, Room, Retrofit, Moshi, Junit, ... и смог завершить проект. Однако, из-за ограниченного времени, многие функции все еще находятся на базовом уровне. Тем не менее, я очень доволен результатами этого проекта, приложение работает хорошо и может удовлетворить потребности пользователей. Я смог удержать время для завершения проекта в течение 3 месяцев. Надеюсь, у меня будет возможность продолжить развитие этого проекта в будущем. Спасибо и до свидания.