

# BITCOIN SCAMMERS' WALLETS

Име:

Камен Трайков

Факултетен номер:

1801321018

Специалност:

Софтуерно Инженерство

# Кратко описание на проекта

- **Вдъхновение**

*Вдъхновението за проекта дойде след като прочетох за история за човек, който бива измамен от „крипто“ лъжец, обещаващ му бързи печалби от търгуване с крипто валути.*

Измамата се осъществява по следния начин:

Жертвата се захваща куката обикновено в секцията за коментари в YouTube, Facebook, Instagram и други. После тя изпраща малка сума пари във формата на *крипто* валута (в случая Bitcoin), за да може да не оставя следи, след което опитните измамници показват как само за няколко дни сте направили 80-100% възвращаемост на инвестицията си. Жертвата изгражда повече доверия и следващия път изпраща голяма сума пари отново във формата на Bitcoin, но този път вместо да види печалба, неговите пари просто изчезват някъде из “Block chain” системата.

- **Цел на проекта**

Целта на проекта е именно да защитава потребителите от подобни измами. Когато жертвата получи адреса на Bitcoin „портфейла“ (wallet), тя може лесно да го въведе в Bitcoin Scammers' Wallet и за броени секунди да получи ключова информация за въпросния портфейл. Потребителя ще получи *Scam Score*, с който лесно може

да прецени риска и да вземе решение дали да се предпази.

- **Предпоставки**

Въпреки добрите намерения и стремежа на приложението да бъде максимално точно в резултатите си, това на практика граничи с невъзможното. Резултата от всяка проверка не може да бъде взимана като абсолютна истина и не трябва да се приема като препоръка или съвет за последващи събития и решения.

Приложението работи само с крипто „портфейли“ съдържащи Bitcoin. Те се различават с това че адреса им започва винаги с „3“ или „1“.

## Дизайн

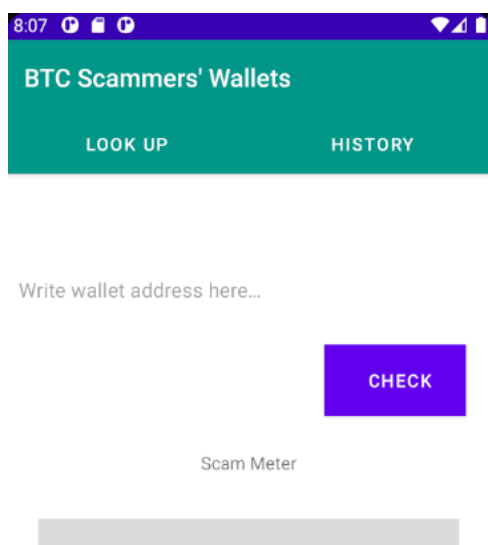
BTC Scammers' wallets използва Layout дизайн с два таба/страници (tabs):

За въвеждане на адреса на търсения „портфейл“, който се състои от текстово поле за писане, бутон за потвърждение и зареждащо „барче“, което се нарича Scam Score Bar.

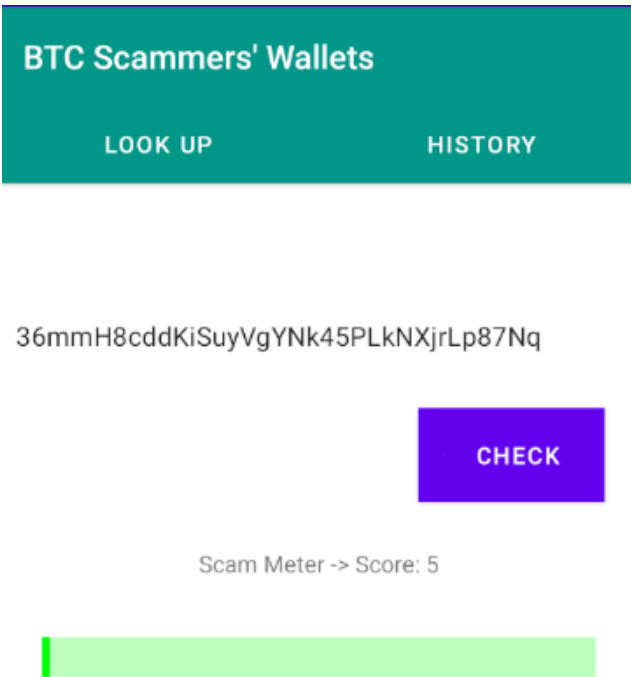
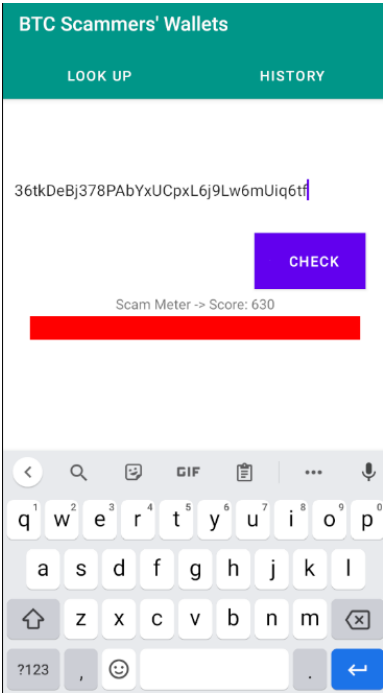
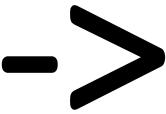
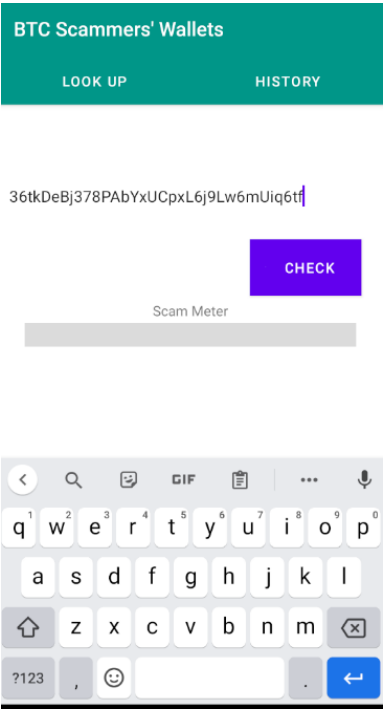
В полето за писане потребителя е подтикнат да въведе адреса на „портфейла“, към който има интерес, с така наречения Hint (подсказка). Под него се намира бутона “Check” (провери), чиято работа се състои в това да прочете адреса на потребителя и да го

прекара през алгоритъм за проверка дали този адрес е всъщност собственост на потенциален измамник. Мястото на бутона е избрано така, че да може потребителя да може по интуитивен начин да разбере той какво прави, с помощта също на текста в него „Check“, който също служи като подсказка.

Под всичко това се намира и Scam Score Bar, което е зареждащ „бар“, но в случай се използва с друга цел. Неговата задача е да изобрази и представи, по удобен начин на потребителя, резултата от неговото искане. Scam Score Bar получава стойност за всеки адрес, която представлява неговата вероятност „портфейла“ да е притежание на злонамерена личност. С начало 0 и максимална стойност 500, потребителя може бързо да получи смислен отговор с буквална стойност, както и визуална промяна на цвета на „барчето“ от зелено за стойностите близки до 0 към червено за стойностите приближаващи 500.

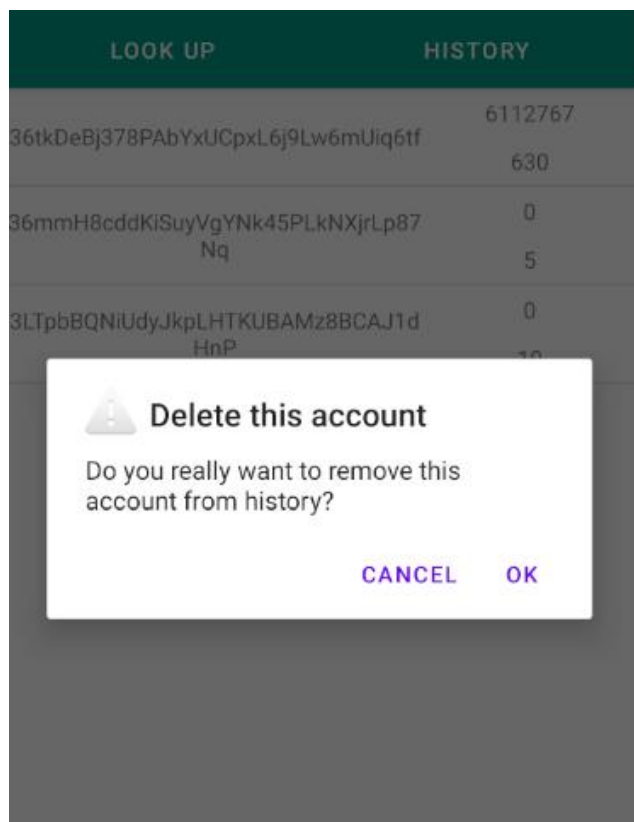


The screenshot shows a mobile application interface for "BTC Scammers' Wallets". At the top, there is a status bar with the time 8:07 and various icons. Below it, the app title "BTC Scammers' Wallets" is displayed in white text on a dark blue background. Underneath the title are two buttons: "LOOK UP" and "HISTORY". Below these buttons is a text input field with the placeholder text "Write wallet address here...". To the right of the input field is a red button labeled "CHECK". Below the input field and the "CHECK" button is a label "Scam Meter". At the bottom of the screen is a horizontal bar, which is currently grey, representing the Scam Score Bar.



Пример за „портфейл“ на измамник (горе) и нормален.

При плъзгане (swipe) на дясно се показва и втория таб/екран, който служи за съхранение на историята на търсения на потребителя и също малко допълнителна информация за „портфейлите“. Потребителя получава подреден поглед върху предишните му търсения, където за всеки „портфейл“ получава информация за адрес, баланс, и Scam Score (резултата от алгоритъма за проверка на „портфейл“). Потребителя може също да изтрие всяка част от историята си само с едно натиска върху колонката, която вече е нежелана и след бърз отговор на *pop-up* прозорче дали е сигурен че иска да изтрие адреса, той вече го няма. Също долу вдясно има “летящ” (floating) бутон, който има картинка на стрелки в кръг, за да подскаже на потребителя че този бутон ще инициира ъпдейт (refresh) на страницата, за да е сигурен че това което вижда е актуална информация.



Прозорец за потвърждение.

## BTC Scammers' Wallets

LOOK UP

HISTORY

36tkDeBj378PAbYxUCpxL6j9Lw6mUiq6tf	6112767
	630

36mmH8cddKiSuyVgYNk45PLkNXjrLp87	0
Nq	5

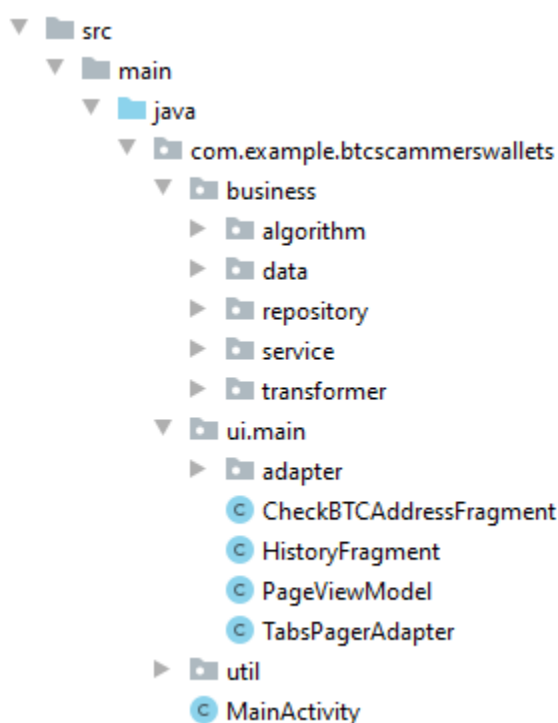
3LTpbBQNiUdyJkpLHTKUBAMz8BCAJ1d	0
HnP	10





# Функционалност

## Архитектура



За архитектура BTC Scammers' wallet използва нещо традиционно и добре познато. Всеки изглед/View си има свой собствен контролер, който се грижи за основната функционалност на елементите във екраните. В тях бизнес

логиката е лимитирана до максимум и вместо това тя е изместена в бизнес (business) слоя (layer).

В него намираме всички обекти, които стоят под **“data”** пакета и са съставени изцяло и само от полета (fields) – или така наречените POJO (plain old java objects). Интересното в случая е проекта има Lombok dependency, което позволява дата обектите да са още по простички и лесно четими, като автоматизира създаването на конструктори и „гетъри“ и „сетъри“ по време на компилация на кода (compile time).

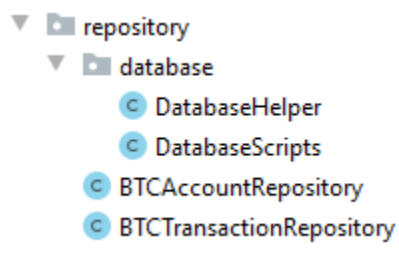
```
@Data
public class BTCTransaction {

    String senderAddress;
    String receiverAddress;
    Long amount;
    TransactionType transactionType;

    public enum TransactionType {
        SENDING,
        RECEIVING
    }
}
```

За извличането на тези данни се използва **“repository”**. То отговаря за основните (CRUD) операции по работа с всички обекти. Там също се намира и **“database”** пакета, който се грижи за достъпа до базата данни и SQL скриптовете които трябва да се изпълняват при инициализация или ъпдейт на базата. За **“repostory”**-тата

интересното е че някои от тях работят с базата, а други правят заявки към API.



“Service”- те се грижат за основната бизнес логика на приложението. Информацията получена от “data access layer” да е обработена и готова преди да премине за визуализация на екрана.

В помощ на “service” – те идват “transformer” – ите, които се грижат за трансформирането на данните от един вид в друг. В контекста на приложението това се състои в преобразуване на данните получени от заявка към API във JSON формат към Java Object и други.

“Algorithm” е последният основен компонент на приложението. Както става ясно това е алгоритъма отговорен за изчисленията нужни, за да получим Scam Score.

# Algorithm

## Scammer Score Algorithm

Scammer Score Algorithm е най-комплексния компонент на приложението, съответно носи и най-голяма отговорност. За входни данни алгоритъмът получава списък от транзакции и акаунта на заподозрения крипто „портфейл“ и връща резултат в формата на число. Колкото по-висок резултат толкова по „сигурен“ алгоритъмът, че „портфейла“ е притежание на злонамерена личност.

За да достигне до крайния резултат алгоритъмът използва помощта на няколко класа: Scammer Score, Certainty Level и Suspicion Level.

```

public class ScammerScore {

    int score;

    public ScammerScore(int initialScore) { score = initialScore; }

    public ScammerScore() { score = 0; }

    public void add(SuspicionLevel suspicionLevel) { score += suspicionLevel.getValue(); }

    public void addMultiple(int amount, SuspicionLevel suspicionLevel) {
        for (int i = 0; i < amount; i++) {
            add(suspicionLevel);
        }
    }

    public int getFinalScore() { return getFinalScore(CertaintyLevel.MEDIUM); }

    public int getFinalScore(CertaintyLevel certaintyLevel) {
        return (int) Math.ceil(score * certaintyLevel.getValue());
    }
}

```

```

public enum CertaintyLevel {
    CERTAIN( value: 3),
    HIGH( value: 1.5),
    LOW( value: 0.5),
    MEDIUM( value: 1),
    UNCERTAIN( value: 0.2);

    private final double value;

    CertaintyLevel(double value) { this.value = value; }

    public double getValue() { return this.value; }
}

```

```
public enum SuspicionLevel {  
  
    SMALL( value: 5),  
    MEDIUM( value: 10),  
    HIGH( value: 20);  
  
    private final int value;  
  
    SuspicionLevel(int value) { this.value = value; }  
  
    public int getValue() { return value; }  
}
```

**Scammer Score** се грижи за следенето на „резултата“ на всеки акаунт. Има два малки метода за добавяне към резултата и за пресмятане на финалния резултат. По този начин класа създава една абстракция, която не само помага за това да направи кода по четим, но и енкапсулира резултата на всеки акаунт в себе си, което намалява възможността от бъгове и прави кода по-лесен за тестване.

**Certainty Level** както се подразбира от името носи със себе си “увереността” на алгоритъма. До колко той е „убеден“ в резултата, който е получил. Има 5 нива на увереност, като всяко едно от тях е коефициент, по-който крайния резултат бива умножен. Certainty Level-а се пресмята от алгоритъма, като е базиран на наличните данни. При наличието на повече информация за акаунта „увереността“ на алгоритъма расте и обратното.

**Suspicion Level** - а е абстракция за количество недоверчивост. Представлява enum, който описва различните стойности за

недоверчивост, като отново ги енкапсулира и защитава от промяна.

## Алгоритъмът

```
public int calculateScammerScore(List<BTCTransaction> suspectedTransactions, BTCAccount suspectedAccount) {  
    scammerScore = new ScammerScore();  
    calculateCertainty(suspectedTransactions);  
    generalAccountCheck(suspectedAccount);  
    generalTransactionsCheck(suspectedTransactions);  
  
    return scammerScore.getFinalScore(certaintyLevel);  
}
```

Множество проверки прави алгоритъмът преди да върне крайния резултат.

Проверки върху транзакциите:

Транзакциите биват разделени на получени и изпратени. Алгоритъма си създава определение за „подозрителна“ транзакция. За всяка такава той добавя определено количество “Suspicion”. Такива са например двойка транзакции от тип изпратени и тип получени с една и съща сума. Причината затова е, защото много често измамниците изпращат получените суми в други акаунти. Друга проверка е дали няма голяма количество изпратени суми към един и същ акаунт по същата причина. Тези и подобни проверки се извършват и постепенно добавят към Scam Score-а на акаунта.

Проверки върху акаунта:

Проверка дали баланса на акаунта е 0. Особено подозрително е ако акаунта има под името си множество транзакции и въпреки това баланса е 0. Това може да означава че крипто валутата се източва към друг акаунт или монетизира в реална валута.

Подозрително и също и общата сума да е много близо до 0 при множество транзакции.