Barcoin

Titolo del progetto: Barcoin
Alunni: Nadir Barlozzo
Classe: I4AC
Anno scolastico: 2018/2019
Docente responsabile: Ugo Bernasconi

1 Introduzione

1.1 Informazioni sul progetto

Docente Responsabile: Ugo Bernasconi
Scuola: Scuola d'Arti Mestieri Trevano
Sezione: Informatica
Materia: Modulo 306
Data inizio progetto: 08.01.2019
Data consegna progetto: 10.05.2019

1.2 Abstract

Barcoin aims to be an open-source solution to provide a micro-credits fully featured platform, transactions get stored through Barcoin's blockchain technologies in order to maintain the users' credentials and transactions' information in a safer environment.

1.3 Scopo

Lo scopo di questo progetto consiste nel rendere l'ultima release del prodotto Barcoin integrato con tecnologia blockchain e supporto multiutenza, così da rendere non solo le transazioni immutabili e più sicure ma fornendo anche il possibile utilizzo in un gruppo di lavoro.
Barcoin subirà inoltre delle modifiche di maggiore importanza per quanto riguarda UI e Database.

Analisi

1.4 Analisi del dominio

Ad oggi esistono diversi software che permettono il micro-management di finanze, vengono usati specialmente nei paesi più poveri, dove le persone "*non riescono ad ottenere credito e altri servizi dalle istituzioni finanziarie tradizionali per due ragioni: vengono reputati non solvibili (unbanked) e/o i costi legati all'offerta di questi servizi sono eccessivi e rendono l'operazione non conveniente economicamente*." **Wikipedia**

Molti di questi applicativi sono però di tipo proprietario e quindi non aperti al dominio pubblico, quelli che invece vengono rilasciati al pubblico sono molte delle volte confusionari o poco trasparenti sul loro funzionamento.

Barcoin è un software sviluppato con tecnologie completamente open-source, comprende inoltre una componente blockchain per accertare l'integrità e sicurezza delle informazioni all'interno del suo sistema.
Questo progetto ha le basi necessarie per fornire al mercato un sistema di micro-management gratuito, più trasparente ed ottenibile da tutti.

1.5 Analisi dei costi e benefici

Categoria	**Costo**
Personale	174 ore * 1 persona * 60 CHF/h = 10440 CHF
TOT	**10440 CHF**

1.6 Analisi tecnologia blockchain

Un’analisi preliminare necessaria per comprendere i difficili fondamenti su cui blockchain è nata.

1.6.1 Abstract

Una blockchain è essenzialmente un database decentralizzato, distribuito pubblicamente e contenente tutte le transazioni o eventi digitali che sono stati eseguiti e condivisi tra le sue parti partecipanti. Ogni pacchetto di dati al suo interno viene definito blocco.
La rete blockchain si occupa di certificare l’integrità e la sicurezza del sistema complessivo tramite differenti tecniche crittografiche ed algoritmiche, per esempio effettuare un hash sul blocco stesso ed inserire una referenza al suo precedente.
I vari blocchi possono essere salvati in differenti catene, oltre alla principale è infatti possibile creare dei rami esterni e parzialmente indipendenti; la catena più lunga è definita determinante in quanto possiede la maggior parte di nodi attivi. Finché quest’ultima non viene attaccata non esistono virtualmente metodi per prendere controllo della rete, inoltre, i nodi possono unirsi e abbandonare la rete a proprio piacimento, a patto che al loro rientro rispettino la catena più lunga creatasi durante la loro mancanza.
Ogni transazione nel registro pubblico è verificata dal consenso dei partecipanti al sistema, questo consenso viene raggiunto in modi diversi a dipendenza dell’algoritmo adottato; una volta confermata le informazioni al suo interno non possono mai essere cancellate o modificate.
Per usare un'analogia di base, è facile rubare una caramella all’interno di un contenitore tenuto in un armadio ma risulta molto difficile se esposto in un mercato osservato da migliaia di persone.

1.6.2 Introduzione

L'attuale economia digitale si basa sulla dipendenza da una certa autorità attendibile.
Le nostre transazioni online si basano sulla fiducia di qualcuno/qualcosa, può essere la banca che ci offre il servizio di e-banking online oppure il nostro ISP che ci offre un servizio internet per casa e/o telefono.

In entrambi i casi ci affidiamo ad una terza entità per la sicurezza e la privacy delle nostre informazioni digitali, queste fonti di terze parti possono essere, teoricamente in ogni momento, violate, manipolate o compromesse.
La tecnologia blockchain è nata per prevenire il problema dei ‘middlemen’ e, comunque, offrire un sistema sicuro ed affidabile.

1.6.3 Algoritmi di consenso, POW (Proof of Work) vs POS (Proof of Stake)

Questi sono i due principali algoritmi tramite i quali le principali implementazioni di blockchain operano e sono tra i fattori decisivi da considerare se e quando si vuole investire in un progetto o meno, alcuni degli aspetti critici da considerare in questa scelta includono la velocità, le applicazioni tecniche e l'efficienza di calcolo (consumi elettrici).

POW

Proof of Work è l'algoritmo più utilizzato e conosciuto, molti dei capisaldi nel settore delle criptovalute lo adoperano come Bitcoin (BTC), Ethereum (ETH) e Dash (DASH).
Come minatore di criptovalute che utilizzano POW è necessario risolvere complessi problemi matematici alla ricerca della valuta virtuale, questo rende l'azione nel tempo esponenzialmente sempre più difficile.
Dopo aver risolto con successo diversi calcoli per varie transazioni, queste vengono verificate, raggruppate e memorizzate in un nuovo blocco sul registro pubblico.

Punti a favore di POW

- La verifica è interamente basata sulla neutralità di un risultato matematico, non esistono modi di avvantaggiarsi del sistema informatico.

Punti a sfavore di POW

- Spreco sia di potenza di calcolo che di elettricità nel generare presupposti casuali per risolvere i complicati problemi matematici.

POS

Visto l'esponenziale aumento in costi e diminuimento in velocità dell'algoritmo POW, l'algoritmo POS guadagna giornalmente terreno sempre in più campi d'applicazione. A differenza di POW che utilizza la potenza hardware del computer, POS capitalizza sulla carenza della valuta.
L'utente deve dimostrare di essere il proprietario di un certo valore in criptovaluta, come tale ottiene dei benefici e crescite del suo bene sull'avanzare della linea temporale (una sorta di interesse per capita).

Punti a favore di POS

- Non c'è bisogno di computer sovradimensionati
- L'intero processo è a basso costo elettrico

Punti a sfavore di POS

- POS offre una piattaforma che contraddice uno dei principi della tecnologia blockchain, quello di non avere un'autorità centrale, consentendo a coloro che possiedono più monete di dettare le modifiche da attuare sulla piattaforma.
- I possessori di ingenti somme di criptovalute subiscono un aumento maggiore rispetto a chi ne possiede in quantità minore.

POA (Extra)

In questo algoritmo, gli amministratori di una rete blockchain convalidano e approvano le transazioni. Si tratta di una soluzione altamente centralizzata, ma anche molto efficiente, potrebbe funzionare per reti private dove i membri conoscono e si fidano uno dell'altro.

1.6.4 Struttura funzionale

In seguito viene dimostrato insieme ad alcune immagini significative il funzionamento di una rete blockchain, l'algoritmo di consenso applicato a questa specifica situazione è il POW.

Indirizzo$_1$

Ogni utente all'interno della rete possiede un indirizzo univoco, questo viene usato per effettuare versamenti sul suo conto.
Si tratta di una stringa alfanumerica, solitamente lunga 32 caratteri.

Rif 1

Chiave privata$_2$

Una chiava privata univoca viene accoppiata con ogni account utente, viene utilizzata per effettuare pagamenti verso altri indirizzi e **deve** rimanere confidenziale.

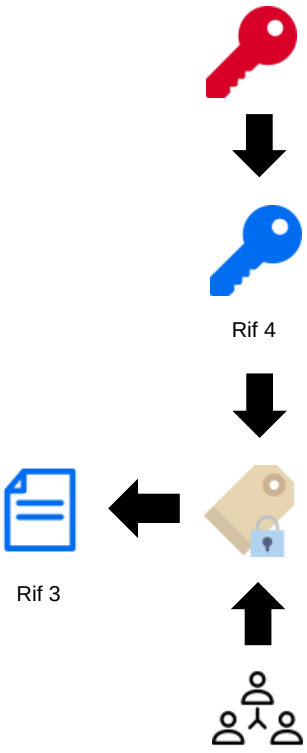
Rif 2

Chiave pubblica$_3$

Una chiave pubblica viene generata applicando una funziona crittografica unilaterale sulla propria chiave privata, questa risultante può essere utilizzata per scoprire il proprio indirizzo su una rete specifica.

Chiave pubblica di script$_4$

Questa chiave risulta essere una semplice conversione dell'indirizzo dalla sua forma "human friendly" ad una versione interpretabile dalla blockchain, la versione ottenuta dall'utente è solamente un'interfaccia di questo dato.



Rif 4

Rif 3

Transazione[5]

Una transazione può non avere destinatario, o può averne diversi. Lo stesso si può dire per i mittenti. Su una blockchain, il mittente e il destinatario sono sempre astratti con una chiave pubblica di script, come dimostrato nei sotto capitoli precedenti.
Il complesso di informazioni finali in uscita, chiamato TxOut, è definito dalla quantità di cripto valuta inviata e la chiave pubblica di script del destinatario.

Punto d'uscita

Ogni TxOut è riferito in modo univoco a livello di blockchain dall'ID della transazione e dall'indice al suo interno. Chiamiamo tale riferimento un punto d'uscita.

Firma digitale

Tramite la propria chiave privata è possibile mascherare un messaggio da inviare all'interno della rete, quest'ultimo sarà riconoscibile e traducibile da tutte le persone che conoscono la firma digitale dell'autore.

Mining

È detta mining l'azione che la comunità di una blockchain è sottoposta ad eseguire per ottenerne la sua cripto valuta.
I "miners" sono entità il cui unico obiettivo è quello di inserire una nuova transazione nella blockchain. Ognuno di loro tenta però di aggiungere un intero lotto di transazioni allo stesso tempo, andando così a formare un blocco invece che un record singolo. Altri nodi della rete (50%+1) confermano che il nuovo blocco obbedisca al protocollo di blockchain, se due minatori dovessero aggiungere un blocco allo stesso tempo viene data la priorità al ramo con l'indice di lavoro più alto, in altre parole, quello su cui sono stati eseguiti più cicli di calcolo. Se un minatore cercasse di includere una transazione non valida nel suo blocco, gli altri nodi non lo riconoscerebbero come valido e il minatore perderebbe l'investimento speso per la creazione del blocco, insieme alle possibili legittime transazioni.
Una volta che un minatore riesce a presentare un blocco valido, tutte le transazioni all'interno sono considerate confermate, quando questo accade tutti i minatori devono scartare il loro lavoro attuale e iniziare a lavorare su un nuovo blocco utilizzando nuove transazioni.

1.6.5 Ulteriori informazioni

Blockchain è una tecnologia in continua crescita, le sue applicazioni diventano ogni giorno più varie ed accettate sia dalle comunità interne o esterne al suo sviluppo.
Migliaia di professionisti in crittografia, algoritmica e matematica si sfidano ogni giorno per trovare nuove soluzioni, cercando di combinare efficienza, sicurezza e scalabilità.

1.7 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Form di registrazione
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Deve esserci un campo per il nome dell'utente di tipo testo
002	Deve esserci un campo per il cognome dell'utente di tipo testo
003	Deve esserci un campo per lo username dell'utente di tipo testo
004	Deve esserci un campo per la password dell'utente di tipo testo cifrato
005	Deve esserci un bottone per la conferma di registrazione dell'utente
006	Deve esserci un bottone per raggiungere il login
007	Deve apparire un messaggio di errore in caso di mancato inserimento di uno dei campi

ID: REQ-02	
Nome	Form di login
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Deve esserci un campo per lo username dell'utente di tipo testo
002	Deve esserci un campo per la password dell'utente di tipo testo cifrato
003	Deve esserci un bottone per la conferma di login dell'utente
004	Deve esserci un bottone per raggiungere la registrazione
005	Deve apparire un messaggio di errore in caso di mancato inserimento di uno dei campi

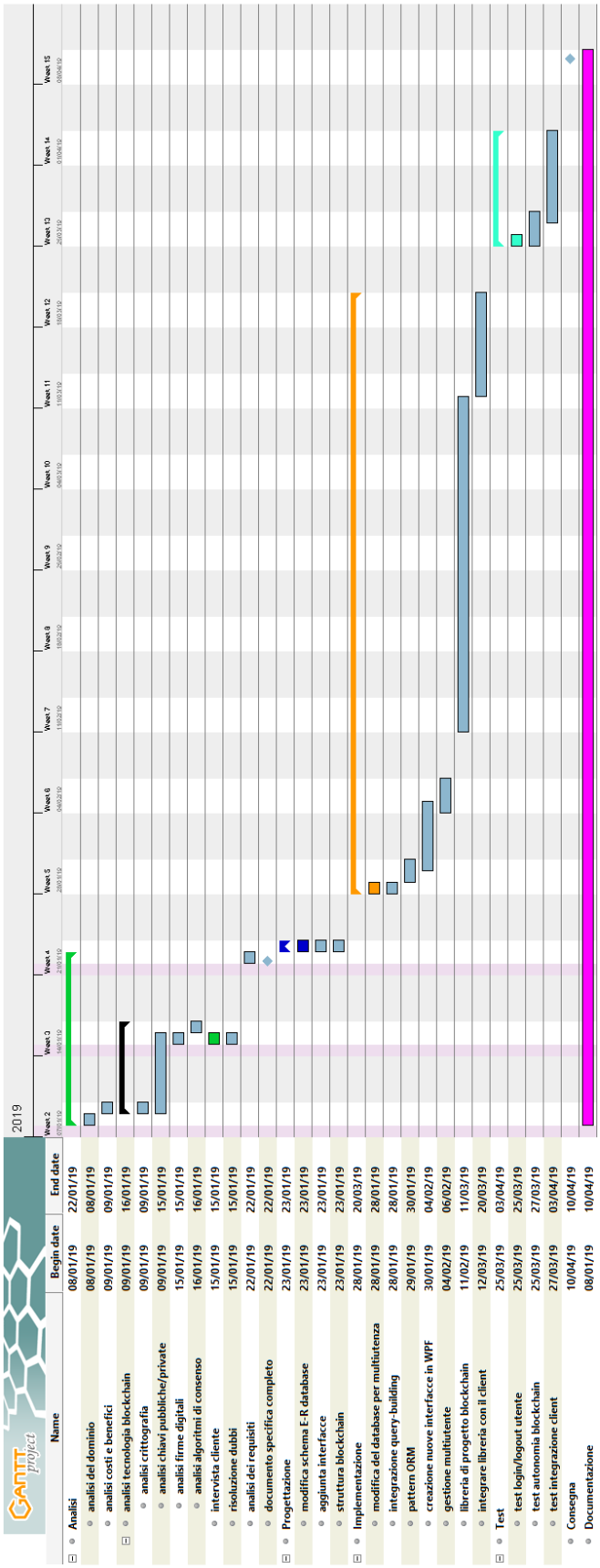
ID: REQ-03	
Nome	Utilità finestra
Priorità	2
Versione	1.1
Note	
Sotto requisiti	
001	Deve esserci un'etichetta per indicare lo stato di connessione con i server di Barcoin
002	Deve esserci un bottone per raggiungere la dashboard
003	Deve esserci un bottone per raggiungere le impostazioni
004	Deve esserci un bottone per raggiungere le informazioni

ID: REQ-04	
Nome	Dashboard
Priorità	1
Versione	1.3
Note	
Sotto requisiti	
001	Deve esserci un bottone per effettuare il log out
002	Deve esserci un'etichetta per il nome completo dell'utente
003	Deve esserci un'etichetta per il bilancio in BRC dell'utente
004	Deve esserci un bottone per raggiungere l'invio di BRC
005	Deve esserci una **Transaction List**
006	Deve esserci una **Chart Series**

ID: REQ-05	
Nome	Chart Series
Priorità	2
Versione	1.1
Note	
Sotto requisiti	
001	Deve contenere tre grafici statistici relativi al sistema di Barcoin
002	Il primo grafico deve visualizzare il numero di transazioni effettuate giornalmente dagli utenti
003	Il secondo grafico deve visualizzare l'ammontare di valuta transitato giornalmente dagli utenti
004	Il terzo grafico deve visualizzare in percentuale i BRC inviati e ricevuti giornalmente dallo stesso utente

ID: REQ-06	
Nome	Transaction List
Priorità	1
Versione	1.1
Note	
Sotto requisiti	
001	Deve contenere in ordine di data decrescente tutte le transazioni relative all'utente corrente
002	Deve esserci un'etichetta per l'id della transazione
003	Deve esserci un'etichetta per il mittente della transazione
004	Deve esserci un'etichetta per il destinatario della transazione
005	Deve esserci un'etichetta per l'hash della transazione in formato corto
006	Deve esserci un'etichetta per l'ammontare di valuta della transazione
007	Deve esserci un'etichetta per la data di creazione della transazione

1.8 Pianificazione



1.8.1 Software

Microsoft Word 2013
Gantt Project 2.8.5
Power Point 2013
Google Chrome 64.0
TortoiseGit 2.5.0
DB Schema 6.0.3
Visual Studio 2017 Community / Enterprise edition

1.8.2 Hardware

Laptop PC – Apple MacBook Air

2 Progettazione

2.1 Design dei dati e database

Lo schema del database è abbastanza semplice, composto da 4 tabelle (**user, transaction, pool** e **block**) sarà poi necessario implementarlo in MySQL e, possibilmente, sincronizzarlo su almeno due host differenti per garantire la continuità e decentralizzazione dei dati. Per scopi di statistica e di storico temporale ogni tabella contiene un campo per la data di creazione di ogni record (**timestamp**).

- La tabella **user** contiene alcuni dei dati anagrafici del creditore come nome e cognome (**firstname** e **lastname**), lo username scelto (**username**), l'hash della password con relativo salt (**password** e **salt**) ed infine l'indirizzo dell'account all'interno del sistema Barcoin (**address**).
- La tabella **transaction** contiene tutte le transazioni che sono state mai generate all'interno del sistema di Barcoin, possiamo trovare transazioni accettate, respinte o in attesa di decisione (**status**).
  Una transazione è definita inoltre da diversi campi come l'id del gruppo di appartenenza (**poolid**), gli id del mittente e mandante (**senderid** e **recipientid**) ed infine l'importo di valuta scambiato (**amount**).
- La tabella **pool** è molto sintetica e serve solamente allo scopo di fornire un riferimento di collezione.
- La tabella **block** chiude il cerchio logico della struttura di dati, include l'id del gruppo di transazioni relativo al blocco (**poolid**), la firma digitale del proprietario (**signature**), l'hash dell'intero blocco (**hash**) ed infine l'hash del blocco precedentemente inserito nella catena (**previoushash**).

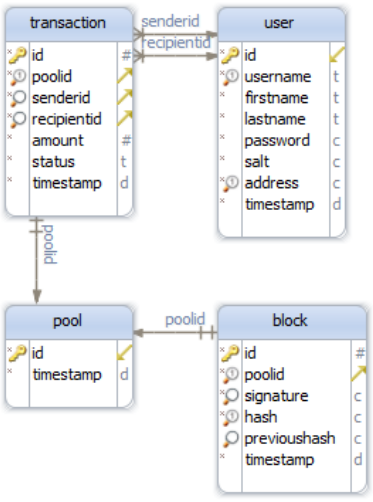


Figura 1 Schema database creato con DbSchema

2.2 Design delle interfacce

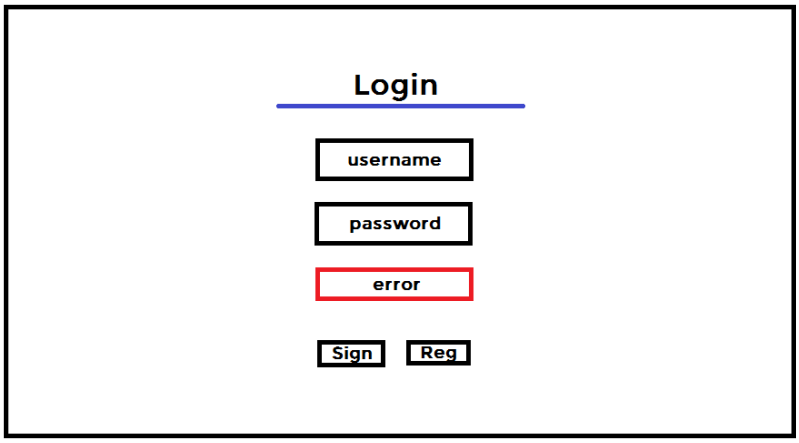


Figura 2 - Mockup di login

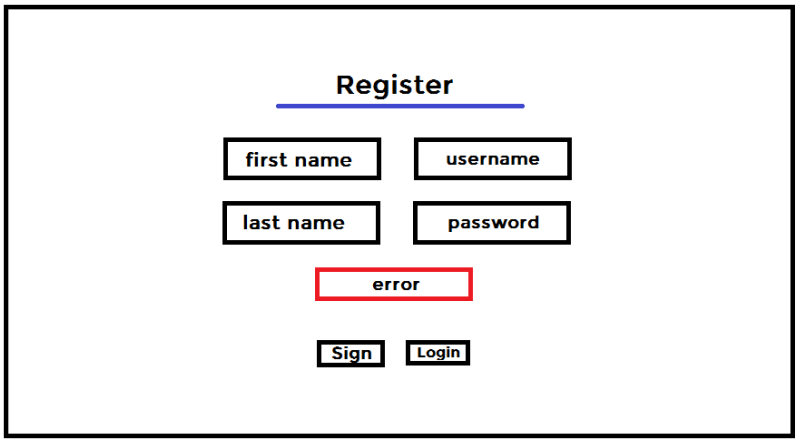


Figura 3 - Mockup di registrazione

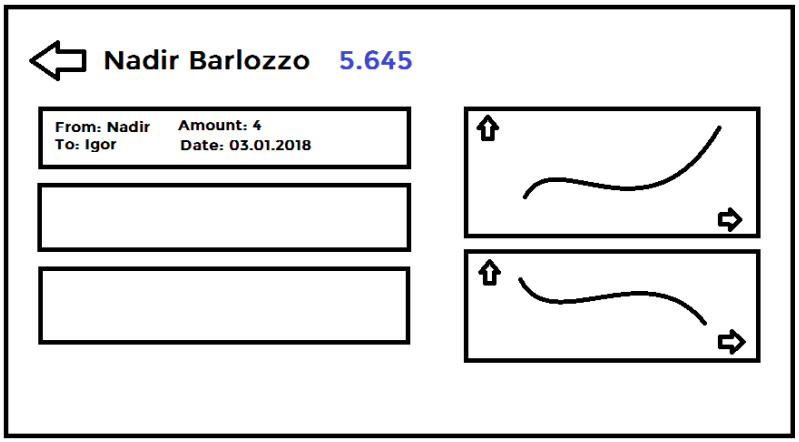


Figura 4 - Mockup di dashboard

2.3 Design Blockchain

La blockchain su cui si appoggia Barcoin si avvale di tecnologie, algoritmi e strutture logiche completamente open-source e raggiungibili da chiunque, sia per l'enorme quantità di supporti e documentazioni sia per seguire la filosofia dell'intero progetto.
Seguono pertanto le sue specifiche tecniche e le diverse applicazioni.

2.3.1 Tipologia

Barcoin opta per essere una piattaforma supportata su una blockchain POW (Proof of Work) e a codifica asimmetrica.
Seppure l'algoritmo di consenso POS (Proof of Stake) sia molto efficiente e a bassi consumi, la sua implementazione include un paio di problemi rilevanti:

- Un livello di mantenimento e sviluppo molto più avanzato.
- La possibile presenza di utenti con più potere decisivo degli altri, fattore che romperebbe le fondamenta di integrità, credibilità e decentralizzazione su cui è costruito Barcoin e che porterebbe il sistema più vicino a quello che non si vuole ottenere come risultato finale.

2.3.2 Utenti

Un BU (Barcoin User) è una qualsiasi persona che per conto suo o per conto di terzi crea un profilo utente all'interno del sistema, questi non possono essere sottoposti a rimozione ma possono subire un cambio di password e username.

2.3.2.1 Indirizzo

Un indirizzo di 32 caratteri alfanumerici generato tramite una libreria GUID, viene legato al proprio profilo utente durante la sua creazione ed è il mezzo utilizzato per inviare o ricevere importi di valuta.

2.3.2.2 Password & Salt

Un hash SHA256, in base 64 (44 caratteri), generato dalla password fornita in chiaro dall'utente combinata con un salt, quest'ultimo è invece creato combinando 32 byte casuali e poi convertito anch'esso in base 64.

2.3.3 Valuta

Un BRC (Barcoin, singolare/plurale) è un'unità di criptovaluta creata ad-hoc totalmente decentralizzata per l'utilizzo del solo intero sistema e che, a differenza di una moneta comune ufficiale come l'Euro (€), non viene coniata, distribuita e regolata da una sola entità centrale come la banca europea.
Sorvolando i discorsi alzati dall'economia e politica odierna sulle criptovalute, Barcoin trova un vantaggio enorme nell'utilizzo di una propria "moneta" (BRC), trae infatti l'autonomia, la sicurezza e la non rintracciabilità digitale che nessun'altra valuta di uso giornaliero può offrire.

2.3.4 Transazioni

Una BT (Barcoin Transaction) è l'evento digitale che accade ogni qual volta un utente effettua un movimento di valuta verso un'altra coordinata del sistema. Ogni transazione ha un suo stato di "condotta" che deve essere accordato dal 50%+1 dei nodi di rete, questo non influenza la sua permanenza all'interno del sistema ma può annullarne il trasferimento di valuta.
Ognuna di queste non circola all'interno della blockchain per conto proprio ma fa parte di un determinato gruppo, denominati "pool".

2.3.5 Chiavi & Firme

Come spiegato in precedenza il sistema si avvale di un sistema a chiavi asimmetriche, l'assegnazione di queste chiavi prende ispirazione dal metodo usato da Bitcoin, Litecoin, Ethereum e molte altre blockchain.
Alla creazione di un profilo utente viene generata una chiave privata dal servizio RSA di Microsoft e salvata immediatamente in un contenitore di sistema relativo solamente alla macchina fisica stessa, questo contenitore non è trasportabile a "mano" ma la piattaforma include una funzionalità per esportare la propria chiave in caso di necessità.
Questa metodologia assicura che anche se qualcuno dovesse sottrarre il profilo utente di una terza persona per obbiettivi malevoli non otterrebbe insieme le sue chiavi, rendendo quasi completamente inutili i suoi attacchi.
Per esportare la propria chiave privata riferirsi all'allegato "B", questa chiave non deve essere mai condivisa in quanto, tramite la chiave pubblica direttamente dipendente dalla privata, è possibile certificare che una determinata firma digitale sia stata effettuata dal corretto utente.

2.3.6 Blocchi

Una blockchain come probabilmente si può intuire è una catena formata da blocchi invece che anelli, ogni BB (Barcoin Block) è fondamentalmente una percentuale del valore complessivo del sistema dato dal gruppo di transazioni assegnato ad esso.
Ogni blocco deve essere in un qualche modo congiunto a quello precedente ed avere la possibilità di essere validato.

2.3.6.1 Proprietario

Ogni blocco viene inserito da un utente, questo diventa a tutti gli effetti il suo proprietario firmando digitalmente il risultante hash del blocco tramite la sua chiave privata.

2.3.6.2 Hash

Gli hash vengono utilizzati in campo informatico per rappresentare un grande quantitativo di informazioni in modo ridotto, nel caso dei BB (Barcoin Block) l'hash viene generato passando all'algoritmo SHA256 una combinazione di informazioni loro più il riferimento al relativo blocco precedente e convertito in base 64 (44 caratteri).

3 Implementazione

3.1 Database MySQL

Progettate le entità che avrei necessitato per l'intero sistema informatico di Barcoin mi sono messo immediatamente ad implementarle nella corrispondente struttura in MySQL.
Per auto-generare la stessa banca dati utilizzata nel progetto è possibile eseguire il file *barcoinv2.sql* all'interno di un qualsiasi DBMS.

3.2 Grafica XAML

Tutti i file creati in questa sezione servono la funzione di **View** all'interno del modello M**V**VM e come tali seguono la nomenclatura *NominazioneView.xaml*.
Ogni **View** ha una connessione logica ad un omonimo **ViewModel** (MV**VM**) il quale serve la funzione di "server" tra l'interfaccia e il modello di dati.
Quest'azione è possibile in parte grazie al *DataContext* e la sua implementazione avviene come segue:

```
DataContext="{Binding Source={StaticResource LocatorResource}, Path=LocatorInsideName}"
```

La parte relativa al funzionamento di un **Locator** è descritta nella sezione di implementazione logica.

I componenti grafici che rappresentano/visualizzano uno stato/informazione dell'utente sono collegati logicamente con una proprietà all'interno dei loro rispettivi **ViewModel**, nel caso si tratti di un campo d'inserimento è necessario specificare la modalità come bilaterale:

```
Caso 1: <ComponentName Property="{Binding Path=PropertyToBind}"/>
Caso 2: <TextBox Text="{Binding Path=Username, Mode=TwoWay}"/>
```

Per quanto riguarda i bottoni abbiamo invece un collegamento con un oggetto di archetipo *IDelegateCommand*, a questo sono date specifiche azioni da eseguire quando invocato:

```
<Button Command="{Binding Path=NameCommand}"/>
```

Alcuni degli stili applicati ai componenti grafici utilizzati nel progetto sono direttamente ereditati dalla libreria di MahApps.Metro.

3.2.1 Login

Ho iniziato l'implementazione grafica con la creazione dell'interfaccia di login, questa include un form con due componenti per l'inserimento delle credenziali e due pulsanti, uno per confermare l'azione e l'altro per raggiungere il form di registrazione.

Per quanto riguarda il campo *password*, è invece impossibile il collegamento logico per ragioni di sicurezza imposte da Microsoft, dobbiamo quindi passare l'intero oggetto come parametro durante l'azione di login.

```
<Button Command="{Binding Path=LoginCommand}" CommandParameter="{Binding ElementName=pass}"/>
```

Infine è stato inserito un ulteriore TextBox per visualizzare eventuali messaggi d'errore generati nel processo, di base è nascosto all'utente.

```
<TextBlock Text="{Binding Path=Error}" Visibility="{Binding Path=ErrorVisibility}"/>
```

Alla corretta autenticazione di un utente segue un dialogo per notificare la persona del login avvenuto con successo.

3.2.2 Register

Creata l'interfaccia di login ho poi implementato per coerenza quella di registrazione.
Questa form è molto simile al precedente in comportamento e disposizione di layout, l'unica differenza risiede nel numero di campi da compilare e in altri messaggi d'errore.

3.2.3 Dashboard

Seguendo l'ultimo mockup disegnato è stata poi costruita la struttura della dashboard/scrivania/homepage, interfaccia dove l'utente può visualizzare maggior parte delle informazioni relative al suo profilo e pure alcuni dati globali.

Nella parte superiore sono presenti due pulsanti per la navigazione, uno effettua il log out dell'utente mentre l'altro raggiunge **Send**. Inoltre ci sono due etichette per visualizzare rispettivamente nome completo dell'utente e bilancio monetario in BRC.

In centro è stato invece inserito inizialmente un componente lista per ordinare le transazioni inviate o ricevute dall'utente, ognuna di queste viene listata con data, valuta trasmessa, mittente/destinatario ed infine l'hash calcolato matematicamente. Questo hash viene accorciato tramite un *ValueConverter* per motivi di spazio e serve solamente come "preview" di quello che è stato generato:

```
<TextBlock Text="{Binding Path=Hash, Converter={StaticResource Shortener},
ConverterParameter=8, StringFormat={}{0}...}"/>
```

La lista viene invece generata tramite un *ItemTemplate* che applica ad ogni elemento di una collezione lo stile predefinito:

```
<ListBox ItemTemplate="{StaticResource TransactionTemplate}" ItemsSource="{Binding
Path=CustomTransactions}"/>

<DataTemplate x:Key="TransactionTemplate"> … </>
```

In un secondo momento sono stati infine aggiunti i tre grafici di *LiveCharts*, dividendo la parte centrale in due colonne e completando così la struttura disegnata inizialmente:

```
<lvc:CartesianChart>
       <lvc:CartesianChart.AxisY>
              <lvc:Axis Title="Count"></lvc:Axis>
       </lvc:CartesianChart.AxisY>
       <lvc:CartesianChart.AxisX>
              <lvc:Axis Title="Date"></lvc:Axis>
       </lvc:CartesianChart.AxisX>
</lvc:CartesianChart>
```

3.2.4 Send

Quest'ultima interfaccia è stata aggiunta in un secondo momento dopo i mockup ed è quindi stata progettata e scritta nello stesso momento.
Presenta un form per l'invio di crediti ad altri utenti, con due componenti per l'inserimento (indirizzo dell'account/ammontare di valuta) e una casella di conferma per i termini e condizioni.
Infine due bottoni per confermare la transazione o per cancellare il processo e tornare alla **Dashboard**.

3.2.5 Shortcuts

Sono stati aggiunti degli *shortcut* in ogni **View** per velocizzarne l'utilizzo d'uso e per migliorare l'esperienza utente:

```
<KeyBinding Command="{Binding Path=NameCommand}" Key="KeyCode"/>
```

Questi sono impostati come segue:

- ESC - Annullare operazione e/o ritornare alla **View** precedente
- ENTER – Confermare operazione e/o procedere alla **View** successiva

3.3 Logica C#

Questa sezione comprende un pacchetto relativamente grande di informazioni e diverse implementazioni, la seguente struttura illustra la divisione dei file in gruppi o funzioni:

Client	Helper
	Model
	Service
	Static
	ViewModel
Blockchain	Enum
	Helper
	Interface
	Model
	Service

I file che servono la funzione di **Model o ViewModel** all'interno del modello **M**V**VM** seguono le rispettive nomenclature *NominazioneModel.cs* e *NominazioneViewModel.cs*.
Ogni **Model** serve la funzione di modello di dati per un entità del nostro sistema, comprende tutte le informazioni relative ad essa tramite delle proprietà omonime e le rende accessibili dai **ViewModel** con metodi specifici. Così come per le **View**, ogni **Model** ha quindi un **ViewModel** relazionato ad esso.

Le classi di *Service* sono invece delle repository che offrono metodi utili per interrogare diverse basi di dati, queste possono essere vere e proprie banche dati come MySQL oppure più semplicemente una classe creata da noi con lo scopo di fornire dei dati locali. Questi file seguono la nomenclatura *NominazioneRepository.cs*.

Infine, per quanto riguarda le sezioni *Helper*, *Static*, *Enum* ed *Interface* sono tutte dedicate a contenere quelle classi che hanno la funzione di supporto nella struttura, magari tramite metodi matematici comuni oppure dati comuni tra tutti.

3.3.1 Client

Questo primo blocco dell'applicazione comprende tutte le componenti relative all'interfaccia utente, seppure sia teoricamente indipendente ha una dipendenza diretta dal secondo data dalla necessità di accedere alla rete blockchain e alle sue informazioni.

3.3.1.1 HashPrefixValueConverter

Si tratta di una classe per accorciare una lunghezza di un hash ad un determinato valore passato come argomento, implementa *IValueConverter* ed eredità quindi i metodi *Convert* e *ConvertBack*.
I valori vengono modificati mediante *substring*:

```
return s.Substring(0, prefixLength);
```

3.3.1.2 CustomTransaction

Estensione specifica della classe **Transaction** per il binding grafico.

3.3.1.3 ChartSeriesRepository

Repository per i dati statistici utilizzati nei grafici, vengono calcolati localmente e vengono salvati in delle *ObservableCollection* (liste senza permessi di modifica) oppure *SeriesCollection* (liste di LiveCharts):

```
public ObservableCollection<CustomTransaction> UserTransactions { get; set; }

public SeriesCollection DailyCurrencyTraded { get; set; }
public SeriesCollection DailyTransactions { get; set; }
public SeriesCollection DailyBalances { get; set; }
```

Per dettare l'ordine nel quale le transazioni vengono processate dal sistema, vengono inizialmente raggruppate in base al loro giorno di creazione e successivamente in base all'ora:

```
var dateGroups = UserTransactions
       .OrderBy(x => Convert.ToDateTime(x.Timestamp).Date)
       .GroupBy(x => Convert.ToDateTime(x.Timestamp).Date);
```

3.3.1.4 Splasher

Classe statica per controllare apertura e chiusura dello splash screen mostrato all'avvio dell'applicazione, si tratta solamente di collegare la proprietà *Splash* con il riferimento ad una *Window*:

```
Splasher.Splash = new SplashScreenWindow();
```

3.3.1.5 LoginViewModel

Il **ViewModel** del **Login** che si occupa di salvare le credenziali inserite dall'utente che vuole accedere al sistema e convalidarle, mostra dei dialoghi di errore o conferma a differenza del risultato ottenuto:

```
private readonly IDialogCoordinator dialogCoordinator;

dialogCoordinator
       .ShowMessageAsync(this, "Login Successful", "You are now signed into your profile.");
```

Durante l'azione di login viene anche controllato che la macchina in uso comprenda un paio di chiavi valide per l'utente corrente, in caso contrario verrà mostrato un dialogo specifico per una rottura di sicurezza:

```
bool validOwner = DigitalSignatureUtils.RetrieveKeyPair(user.Address);

if (validOwner)
{
	…
}
else
{
	dialogCoordinator.ShowMessageAsync(
		this,
		"Security Breach - 3",
		"A valid key pair associated to this account wasn't found on your machine."
	);
}
```

3.3.1.6 RegisterViewModel

Il **ViewModel** del **Register** che si occupa di salvare le informazioni inserite dall'utente per la registrazione del suo account e convalidarle, al successo di questa azione, un paio di chiavi asimmetriche crittografiche verranno generate in un contenitore di registro all'interno della macchina utilizzata:

```
DigitalSignatureUtils.AssignKeyPair(user.Address);
```

Verranno poi usate in un secondo momento per firmare digitalmente i propri blocchi nella blockchain.

Per quanto riguarda gli indirizzi degli account, sono generati come GUID, una sequenza di caratteri a 128bits utilizzata per identificare le risorse univocamente:

```
Guid gAddress = Guid.NewGuid(); //xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
gAddress.ToString("N"); //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

3.3.1.7 DashboardViewModel

Il **ViewModel** della **Dashboard** che si occupa di salvare i dati delle transazioni, le informazioni dell'account stesso e calcolare il bilancio monetario del relativo utente, questo viene fatto inizializzando e convalidando un'istanza della blockchain di Barcoin:

```
var barcoin = new Blockchain.Model.Blockchain();
```

Per creare la sessione e notificare la **Dashboard** del login di un determinato profilo viene implementato un *Messenger*, sistema simile ai socket web dove un nodo notifica tutti gli altri "iscritti" di un cambiamento:

```
Application.Current.Resources["SID"] = user.Id;
Messenger.Default.Register<User>(this, OnSentUser);

Messenger.Default.Send(user); //LoginViewModel.cs
```

Queste sessioni vengono chiuse quando l'utente effettua il logout:

```
SignedUser = null;
Application.Current.Resources["SID"] = null;
```

3.3.1.8 SendViewModel

Il **ViewModel** del **Send** che si occupa di salvare le informazioni inserite dall'utente per inviare una transazione verso un altro indirizzo, viene convalidata anche la conferma di aver letto i termini e condizioni (ancora da scrivere e dettare accuratamente) per tutelare in futuro se stessi e il sistema da futili reclami.

Il sistema è pensato in modo da:

- Bloccare il raggiungimento della **View** relativa a questa classe nel caso in cui il bilancio sia equivalente a zero.
- Avere un "tetto" massimo di importo equivalente al bilancio dell'utente, nel caso in cui la persona inserisca un valore maggiore verrà sostituito con il valore di default e notificato del cambiamento.

3.3.2 Blockchain

Questo secondo blocco dell'applicazione comprende tutte le componenti logiche che compongono l'intero sistema blockchain, si tratta di una libreria di progetto e come tale non è possibile lanciarlo indipendentemente.

3.3.2.1 TransactionStatus

Enum che rappresenta gli stati di una transazione: in attesa, accettata o respinta.

3.3.2.2 DbHelper

Classe per facilitare la connessione con il database *barcoinv2* ad altri componenti della struttura, una *QueryFactory* è l'istanza che tutti potranno usare con flessibilità senza dipendere da un driver specifico:

```
string options =
"Host=localhost;Port=3306;User=root;Password=;Database=barcoinv2;SslMode=None;";

using (MySqlConnection connection = new MySqlConnection(options))
{
	connection.Open();
	db = new QueryFactory(connection, new MySqlCompiler());
}
```

3.4 Algoritmo di credito

Fondamento logico principale su cui si basa Barcoin, il calcolo degli interessi e bilanci alla fine di ogni transazione per poter ottenere un sistema corretto e trasparente. Ha necessitato diversi ragionamenti ed analisi di alcuni fogli di calcolo Excel già esistenti, il risultato è il seguente:

B = Bilancio totale
C = Capitale accreditato totale
IA = Bilancio interesse
I = Interesse percentuale di una singola transazione
$\sum I$ = Interessi accumulati delle transazioni precedenti
T = Tasso giornaliero stabilito (360/365)
D_0 = Data accredito relativa
D_1 = Data corrente

$$B = C + IA$$
$$B = C + C * \left(\frac{I}{T}\right) + \sum_{I=0}^{n} I * (D_1 - D_0)$$

Esempio:

$$B = 120CHF + 120CHF * \left(\frac{6.5\%}{360}\right) + 0.34CHF * (23.03.18 - 18.01.18) + 0.12CHF * (23.03.18 - 29.10.17)$$

Per ottenere questo bilancio prendo il capitale totale di 120CHF, gli aggiungo il 6.5% di interesse diviso per il tasso giornaliero stabilito, infine sommo gli interessi calcolati nei bilanci precedenti moltiplicati per l'intervallo di giorni tra la data attuale e la data di accredito relativa.

3.5 Dipendenze

CommonServiceLocation v1.0.0 – Fornisce un livello astratto per l'inserimento di dipendenze logiche.

LiveCharts v0.9.7 – Visualizzazione dei dati semplice, flessibile ed interattiva per .NET.
LiveCharts.WPF v0.9.7 – Estensione di LiveCharts per WPF.

Interactivity.WPF v2.0.20525 – Pacchetto di Microsoft per favorire l'interattività WPF.

SqlKata v1.1.7 – Un potente query builder che supporta diverse versioni di SQL tra cui MySQL

Dapper v1.60.7 – Libreria micro-ORM per interfacciarsi con diverse versioni di SQL

MahApps.Metro v1.6.5 – Libreria grafica per componenti nativi o aggiunti in stile Metro

4 Test

Test Case: **Riferimento**:	TC-01 REQ-01	**Nome:**	Controllo anteprima
Descrizione:	Visualizzare i creditori registrati in versione anteprima.		
Prerequisiti:	La banca dati deve contenere almeno un creditore.		
Procedura:	1. Aprire l'applicazione ed aspettare per lo splash screen di finire		
Risultati attesi:	Una serie di bottoni verranno visualizzati quanti sono i creditori registrati nella banca dati.		

Test Case: **Riferimento**:	TC-02 REQ-02	**Nome:**	Controllo dettaglio
Descrizione:	Visualizzare i creditori registrati in versione dettaglio.		
Prerequisiti:	TC-01.		
Procedura:	1. Premere su uno dei bottoni presenti nell'interfaccia di anteprima		
Risultati attesi:	L'utente viene mandato ad una nuova schermata con due dati testuali, un form per inserire nuove transazioni, una tabella che elenca tutte le transazioni effettuate ed un grafico del bilancio monetario. Ogni sessanta secondi deve essere effettuato un aggiornamento automatico dei dati mostrati.		

Test Case: **Riferimento**:	TC-03 REQ-04	**Nome:**	Controllo inserimento creditore
Descrizione:	Inserimento nella banca dati di un nuovo creditore.		
Prerequisiti:	Nessuno.		
Procedura:	1. Premere il bottone in alto a destra con l'immagine di un "+" 2. Compilare tutti i campi presenti nel form con i dati appartenenti al nuovo creditore. 3. Premere il bottone "Create"		
Risultati attesi:	L'utente viene riportato indietro alla schermata in anteprima con un nuovo bottone per l'utente creato.		

Test Case: TC-04 **Riferimento**: REQ-05		**Nome:**	Controllo inserimento transazione
Descrizione:	Inserimento nella banca dati di una nuova transazione.		
Prerequisiti:	TC-02.		
Procedura:	1. Compilare il form nell'interfaccia di dettaglio con i dati della nuova transazione. 2. Premere su "Add Transaction"		
Risultati attesi:	La tabella delle transazioni e il grafico dei bilanci vengono aggiornati per comprendere il nuovo record, l'utente avrà una nuova riga di tabella ed una coppia di colonne in più.		

Test Case: TC-05 **Riferimento**: REQ-06		**Nome:**	Controllo esportazioni PDF e CSV
Descrizione:	Esportare i vari dati in formato PDF e CSV.		
Prerequisiti:	TC-02.		
Procedura:	1. Premere il tasto "e". 2. Premere il tasto "p".		
Risultati attesi:	Andando nella cartella di compilazione del progetto l'utente troverà le cartelle "ExportedTransactions" e "PDFTransactions", al loro interno ci saranno gli omonimi file esportati con l'identificativo di *NomeCognome*.		

4.1 Risultati test

- TC-01 ✓
- TC-02 ✓
- TC-03 ✓
- TC-04 ✓
- TC-05 ✓

4.2 Mancanze/limitazioni conosciute

Come spiegato in modo abbastanza dettagliato all'inizio del documento, Barcoin non si presenta neanche lontanamente come un software per livelli aziendali o in grande volume, presenta infatti gli strumenti e funzionalità limitate per un campione di persone ristretto.

L'utilizzo del linguaggio C# porta inoltre delle limitazioni a livello di software e hardware, a differenza dei suoi simili C e C++ è ad alto livello di astrazione, con una compilazione più pesante e meno performante.

5 Consuntivo

6 Conclusioni

6.1 Sviluppi futuri

Come sviluppi possibili da implementare ci sarebbe in primo punto la versione web, disponibile a tutti aventi una connessione internet e in secondo punto la creazione di una valuta proprietaria per sorpassare i problemi di conversione, inflazione e truffa.

6.2 Considerazioni personali

Ho trovato personalmente il progetto divertente e particolarmente interessante verso le mie conoscenze economiche, con il professore è già stata discussa la possibilità di legare questo progetto con uno degli sviluppi futuri e renderlo basato su una valuta virtuale.
L'obiettivo sarebbe quello di creare una struttura basata su tecnologia blockchain per mantenere e controllare l'autenticità di questa valuta usata in Barcoin.

7 Bibliografia

Le risorse utilizzate come supporto e fondamenta per lo sviluppo di Barcoin, alcune di queste sono propri componenti software mentre altre sono testi, documentazioni o media.

7.1 Sitografia

https://lvcharts.net/, Sito di documentazione LiveCharts WPF
https://docs.microsoft.com/en-us/dotnet/framework/wpf/, Sito di documentazione Microsoft WPF
https://en.wikipedia.org/wiki/Microfinance, Wikipedia sulla micro finanza
https://dev.mysql.com/doc/, Sito di documentazione MySQL
http://www3.weforum.org/docs/WEF_Realizing_Potential_Blockchain.pdf, Blockchain documento 1
https://en.wikipedia.org/wiki/Blockchain, Blockchain documento 2
https://programmingblockchain.gitbook.io/programmingblockchain/, Blockchain documento 3
https://blockgeeks.com/guides/what-is-blockchain-technology/, Blockchain documento 4
https://cryptodigestnews.com/ , Blockchain forum

8 Allegati

Allegato A: *I4_Diari_Barcoin.docx*

9 Glossario

Parola	Significato
WPF	Windows Presentation Foundation, modello di Microsoft per la costruzione di applicativi con interfaccia utente basata sul linguaggio XAML.
MVVM	Model View ViewModel, gruppo di regole e standard per definire una struttura dati da seguire negli sviluppi di applicativi.
Micro-Management	Stile di gestione in cui un manager osserva e / o controlla da vicino il lavoro dei suoi dipendenti o clienti.
Diem Rate	Tasso giornaliero prestabilito all'avvio di una linea di credito che indica quanti giorni effettivi vengono considerati dalla banca all'interno di un anno.
Unbanked	Che non ha la possibilità di pagare, non solvibile.
CSV	Comma-separated values, formato di dati separati da un carattere, solitamente dalla virgola.
PDF	Portable Document Format, formato di dati molto simile ad un documento Word ma senza la possibilità di modifica.