

CoinManager

Elaborato per il corso di basi di dati

Roberto Montalti matr. 0000915996
Juri Simoncini matr. 0000922072

Struttura dell'elaborato

La presente documentazione tratta nello specifico la progettazione e l'implementazione dell'elaborato "CoinManager" di Roberto Montalti e Juri Simoncini, ed è strutturata come segue:

1-Introduzione

2 -Analisi dei requisiti

2.1- Requisiti in linguaggio naturale

2.2 – Estrazione dei concetti fondamentali

3-Progettazione Concettuale

3.1 - Anteprima sviluppo dei "Movimenti Bancari"

3.2 – Anteprima sviluppo delle "Transazioni"

3.3 – Anteprima sviluppo degli "Utenti"

3.4 – Anteprima sviluppo delle "Amicizie"

3.5 – Anteprima sviluppo "Portafoglio"

3.6 – Schema Generale

4 – Progettazione Logica

4.2 – Descrizione delle operazioni principali e stima della loro frequenza

4.3 – Schemi di navigazione e tabelle degli accessi

(4.3.1) Acquisto, vendita o prestito Cripto

(4.3.2) Transazioni tra utenti

(4.3.3) Visualizzazione di una transazione

(4.3.4) Visualizzazione cripto valute

(4.3.5) Miner conferma una transazione

(4.3.6) Visualizzazione amici da parte di un utente

(4.3.7) Invio richieste di amicizie

(4.3.8) Accettare una richiesta di amicizia

(4.3.9) Visualizzazione transazione in base allo stato

(4.3.10) Visualizzazione cripto nel portafoglio

4.4 – Raffinamento dello schema

4.4.1 – Eliminazione delle gerarchie

4.6 – Traduzioni di entità e associazioni in relazioni

5 – Progettazione Fisica

6 – Progettazione dell'Applicazione

6.2 – Descrizione dell'architettura

6.3 – Interfaccia Utente

6.3.2 - Lista di cripto

6.3.3 - Portafoglio

6.3.4 - Transazioni

6.3.5 - Profilo

1-Introduzione

Il progetto consiste nella realizzazione di un sistema database a supporto di una piattaforma di compravendita di criptovalute chiamata "CoinManager". La piattaforma permette all'utente loggato di comprare, vendere o chiedere un prestito per la criptovaluta scelta.

Per le informazioni sulle criptovalute ci colleghiamo alla piattaforma di analisi CoinGecko che ci permette di avere dati aggiornati ogni ora.

Inoltre, grazie a un sistema di amicizie consente di effettuare transazioni verso un amico, queste transazioni poi possono essere completate più velocemente grazie all'intervento di un utente speciale, chiamato Miner.

2 -Analisi dei requisiti

La seguente descrizione riporta in linguaggio naturale i requisiti per il nostro sistema informativo, per poi poterne estrarre i principali concetti fondamentali:

2.1- Requisiti in linguaggio naturale

"La piattaforma di cripto analisi *CoinGecko* vuole ampliare le proprie funzionalità e ha deciso di commissionare una piattaforma per lo scambio di criptovalute sfruttando i propri dati sulle cripto.

Si vuole creare un' applicazione che permetta agli utenti di effettuare uno o più movimenti bancari: Acquisto, Prestito o Vendita di criptovalute.

Una volta effettuato un movimento bancario la cripto viene aggiunta al Portafoglio personale dell'utente dal quale l'utente potrà avviare delle Transazioni verso un altro Portafoglio di un utente amico.

Si vuole tenere traccia di tutte le Transazioni effettuate sia quelle completate che quelle in corso.

Perciò dev'essere presente anche un sistema di amicizie attraverso il quale un utente può avere più amici e può inviare richieste di amicizie.

Per quanto riguarda sempre l'utente è di massima importanza fare una distinzione fra Utente Standard e Utente Miner con la differenza che l'utente miner può completare le Transazioni in corso”

2.2 – Estrazione dei concetti fondamentali

Soggetto	Descrizione	Sinonimi
Utente	Colui che effettua tutte le transazioni e i movimenti bancari	Manager
Acquisto	Movimento bancario che permette l'aggiunta al portafoglio di crypto	-
Prestito	Movimento bancario che permette l'aggiunta al portafoglio di crypto in prestito	-
Transazione	Invio di crypto fra utenti	Invio
Vendita	Movimento bancario che permette la vendita di crypto	-
Portafoglio	Lista delle crypto possedute da un utente	-

Cripto	Moneta di compravendita	Moneta
Miner	Utente speciale che permette il completamento più veloce delle transazioni	-
Sessione Miner	Storicizza tutte le transazioni a cui ha partecipato un miner	-
Transazioni in corso	Transazioni non ancora completate	Transazioni Pendenti
Amicizia	Amicizia tra due utenti che permette le transazioni tra i due	-
Richieste di amicizia	Richiesta di amicizia non ancora accettata dall' invitato	-

A seguito della lettura e comprensione dei requisiti si procede redigendo un testo che ne riassume tutti i concetti e in particolare ne estragga quelli principali eliminando le ambiguità:

Un **Utente** può essere **UtenteStandard** oppure **Miner**. Per ogni **UtenteStandard** e **Miner** si memorizzano : Id, Username e Password. Per il **Miner** si memorizza anche la potenza di mining

Ogni **Utente** può effettuare più **Acquisti**, **Prestiti** o **Vendite**. Per ogni **Acquisto** e **Vendita** si memorizzano: Id utente, Id cripto, Id cripto base, quantità di cripto base, quantità di acquisto.

Per ogni **Prestito** si memorizzano: Id utente, Id cripto, la data di scadenza, la quantità di valuta anticipata e l'id della valuta anticipata. Quando un prestito scade, questo viene detratto automaticamente dal portafoglio dell'utente che ha richiesto il prestito.

Ogni volta che una **Cripto** viene acquistata o venduta o presa in prestito viene aggiunta al **Portafoglio** dell'**Utente**.

Per ogni **Portafoglio** viene memorizzato: Id utente, Id cripto, Quantità di cripto posseduta.

Per ogni **Cripto** viene memorizzato : Nome, Simbolo, Prezzo corrente, Immagine della cripto, volume di mercato, volume totale e la quantità di valuta in circolo.

Dal **Portafoglio** può essere effettuata una **Transazione** verso il **Portafoglio** di un altro **Utente** e quindi la Transazione viene memorizzata anche nelle **Transazioni in corso**

Per ogni **Transazione in corso** viene memorizzato: Id transazione, data inizio, tempo totale. Quando una transazione in corso viene completata, per tempo scaduto o da un miner, un task c# eliminerà la tupla corrispondente e aggiornerà il valore dell'attributo *Stato* di una tupla della tabella *Transazione*.

Per ogni **Transazione** viene memorizzato: Id utente che avvia la transazione, Id utente che riceve, Id cripto, Data di inizio, Data di fine, Quantità di cripto inviata, Id di un possibile miner, Stato (1 = Completata, 2 = In corso).

Un **Utente** può avviare una **Transazione** solo verso un **Utente** che è presente nelle sue **Amicizie** attraverso una **Richiesta di amicizia**

Per ogni **Amicizia** si memorizzano: Id utente, Id amico.

Per ogni **Richiesta di amicizia** viene memorizzato: Id utente che ha inviato la richiesta, Id invitato.

Segue un elenco delle principali azioni richieste:

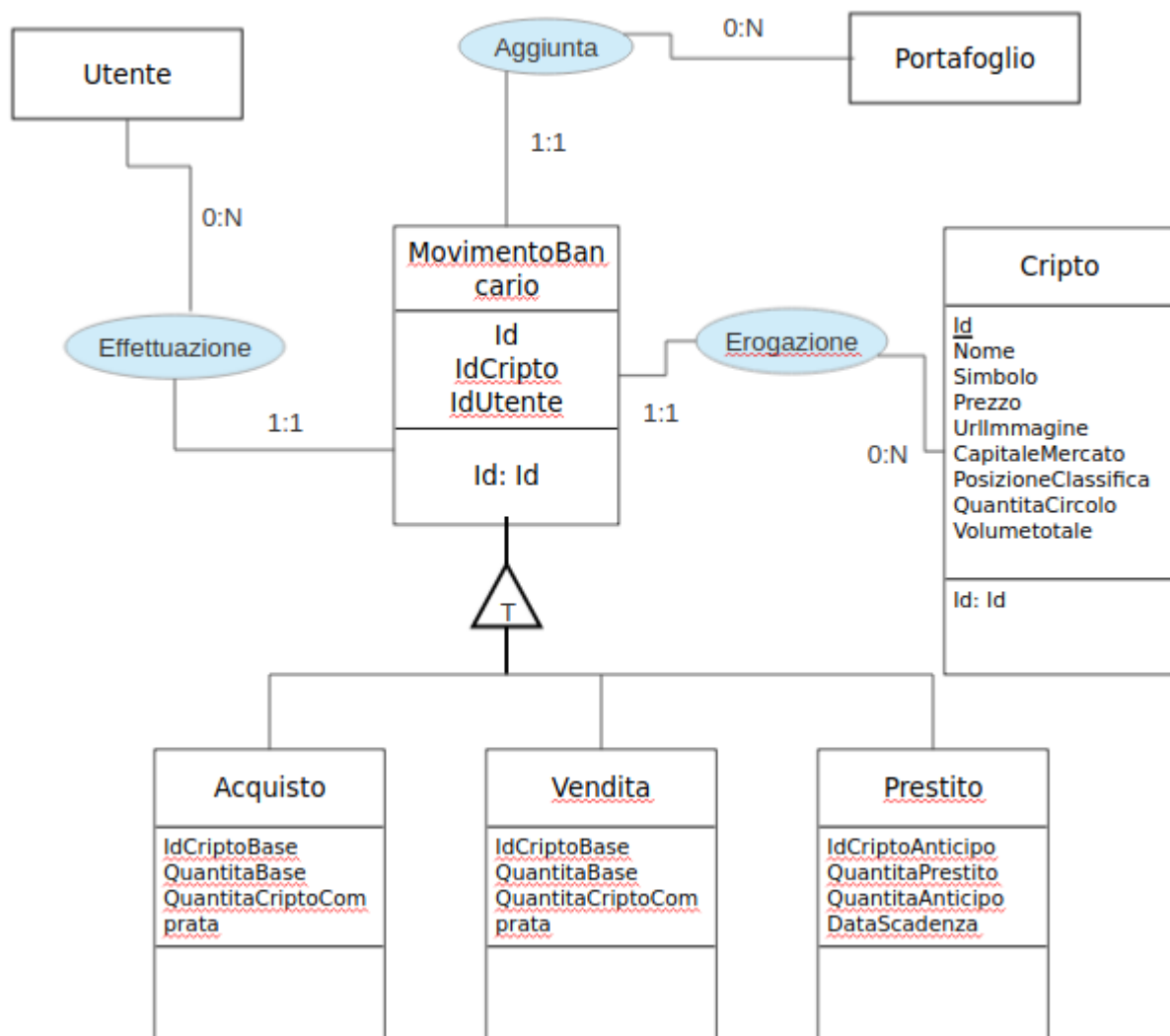
- Acquisto, Vendita o Prestito cripto
- Transazioni tra utenti
- Visualizzazione Transazione
- Visualizzazione dati cripto
- Completamento Transazioni in corso da parte del Miner
- Visualizzazione Amicizie dell'Utente loggato
- Invio Richieste di amicizie
- Accettare Richiesta di amicizia
- Visualizzazione Transazioni filtrate in base allo stato
- Visualizzazione Cripto possedute nel Portafoglio

3-Progettazione Concettuale

3.1 - Anteprima sviluppo dei “Movimenti Bancari”

Si decide di dividere i vari aspetti del **Movimento Bancario** in più entità: **Acquisto**, **Vendita**, **Prestito**.

Ognuna di queste entità è figlia della classe padre **Movimento Bancario**. La gerarchia è stata decisa di renderla totale ed esclusiva.

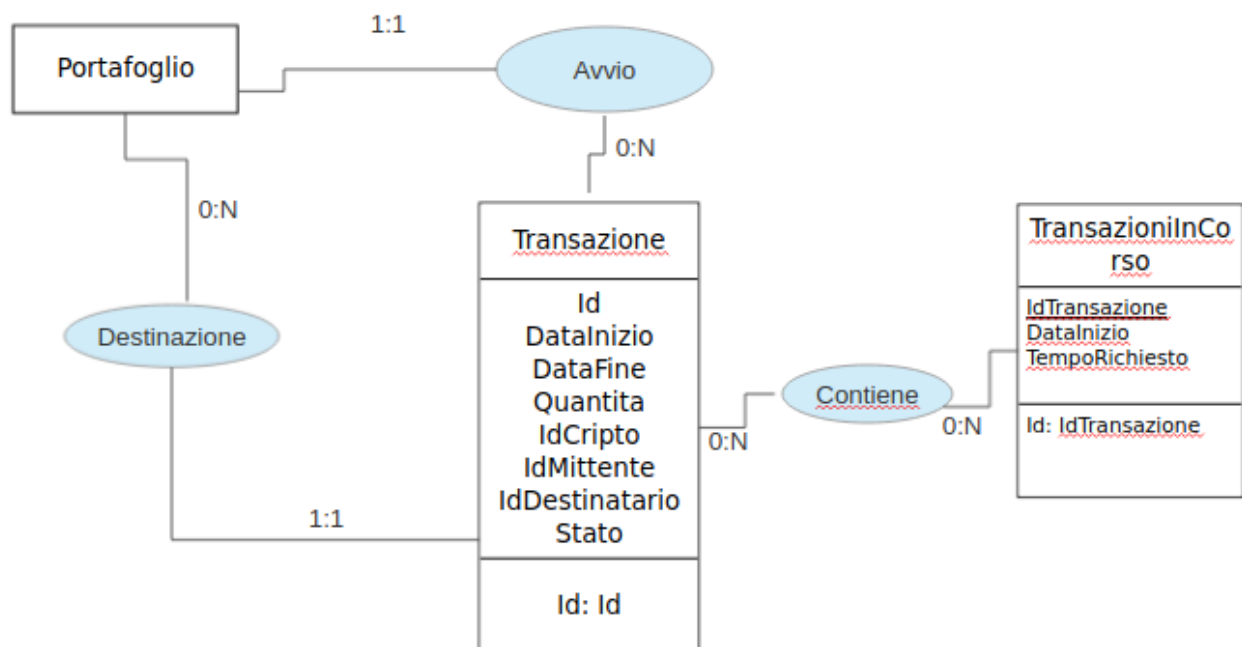


3.2 – Anteprima sviluppo delle “Transazioni”

La **Transazione** si è deciso di modellarla in modo che sia presente un record che presenta tutte le Transazioni effettuate, quindi sia quelle in corso sia quelle completate, e un record che presenta solo le **Transazioni in corso**. E' stato scelto questo metodo sia per storicizzare tutte le Transazioni sia per la questione del miner che così può accedere direttamente alle Transazioni in corso

Una **Transazione** è identificata da un *IdTransazione*; possiede un id sia per chi ha avviato la transazione sia per il destinatario; è presente sia una data di inizio transazione sia una di fine se questa è stata completata, un *IdMiner* se il Miner è intervenuto e uno Stato se la transazione è stata completata o in corso.

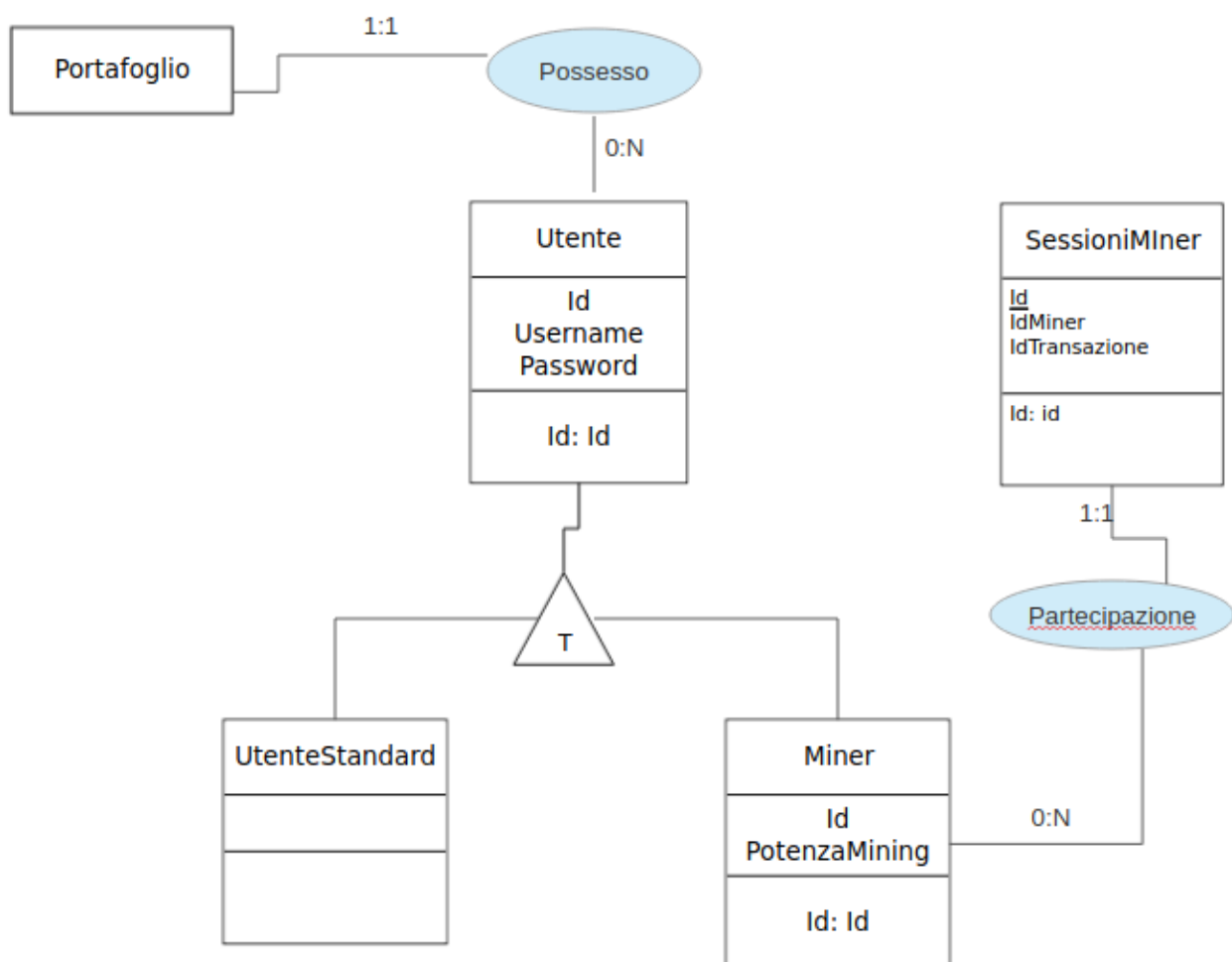
Una **Transazione in corso**, invece, ha bisogno solo dell'*IdTransazione*, della Data di inizio e del Tempo totale che impiega la Transazione prima di essere completata.



3.3 – Anteprima sviluppo degli “Utenti”

Gli **Utenti** abbiamo pensato di dividerli in **UtentiStandard** e **Miner** attraverso una gerarchia totale ed esclusiva poiché un utente non può essere sia standard che miner.

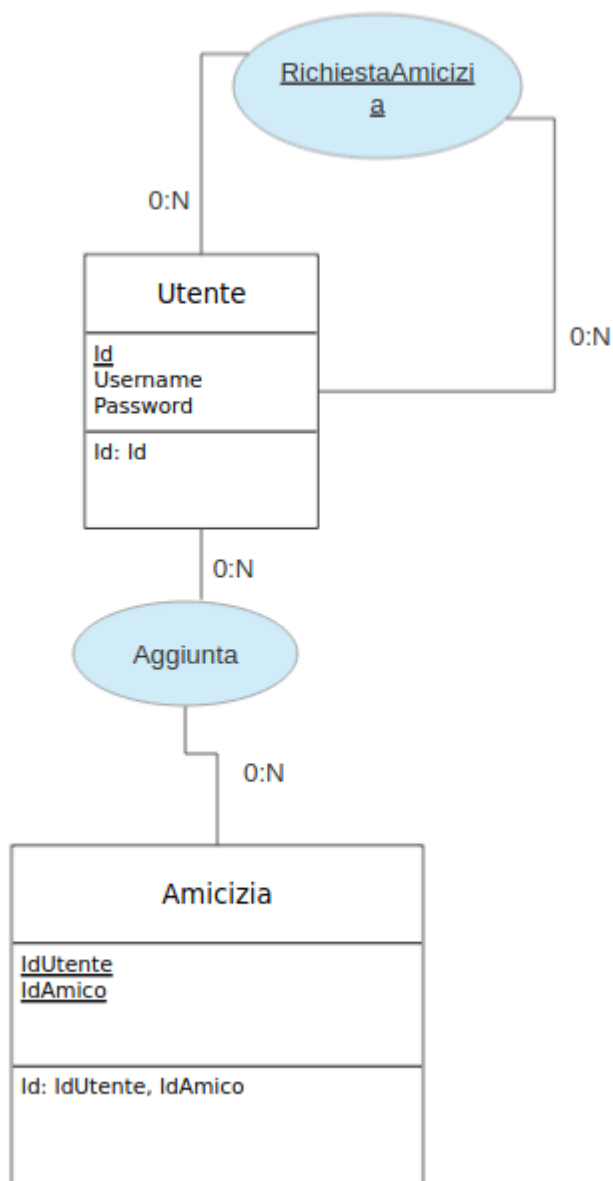
Le gerarchia è stata pensata anche per una questione di funzionalità dell'applicativo poiché il Miner possiede feature che l'*UtenteStandard* non ha. Dell'**Utente** voglio conoscere l'Id lo Username e la password per il login. Il **Miner** possiede anche la potenza di mining.



3.4 – Anteprima sviluppo delle “Amicizie”

Le **Amicizie** si è deciso di gestirle in maniera molto semplice attraverso un *IdUtente* e un *IdAmico* ed è identificata da entrambi gli attributi

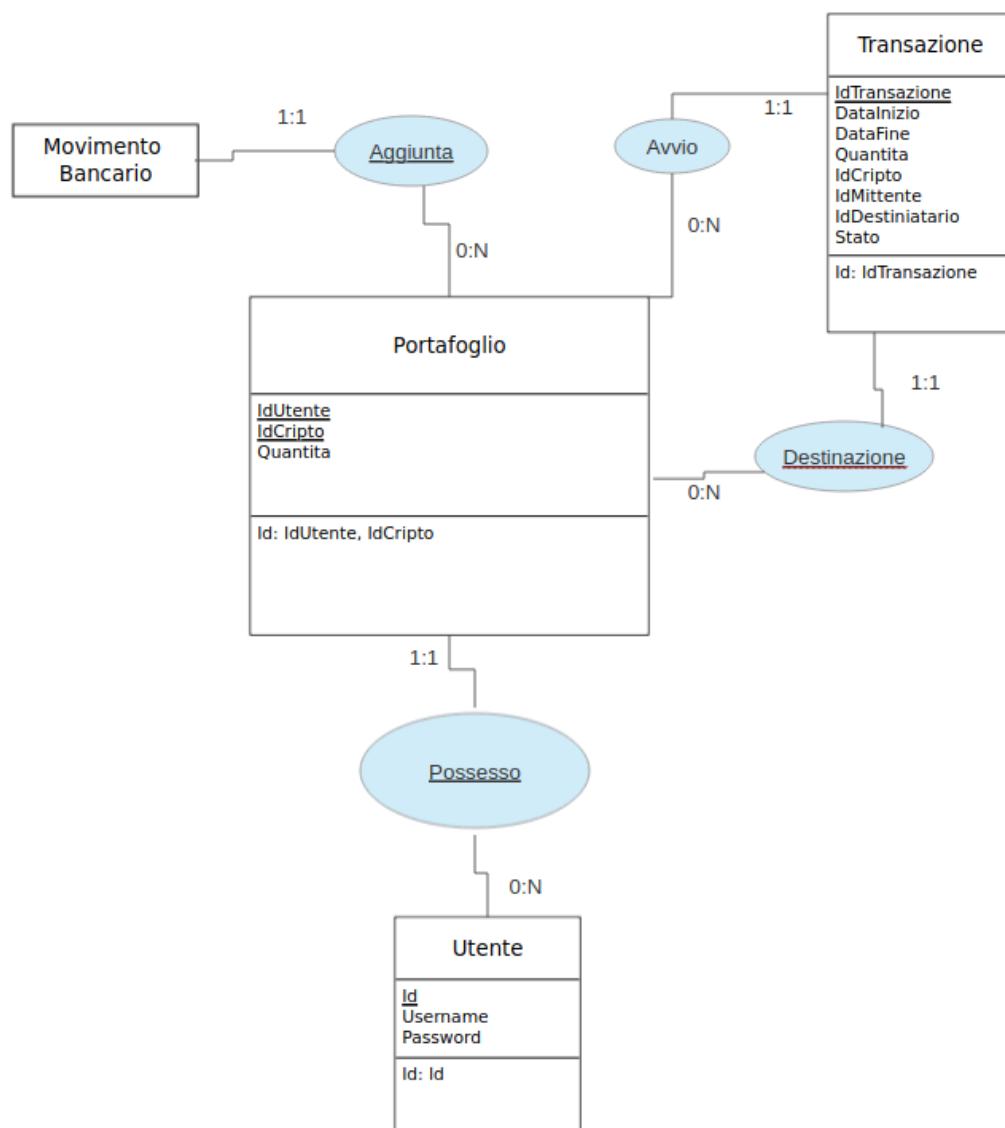
Se l'Amicizia non è ancora stata accettata dall'invitato è presente la **Richiesta di amicizia** che come l'Amicizia è identificata da *Id utente* di chi ha invitato e *Id invitato*.



3.5 – Anteprima sviluppo “Portafoglio”

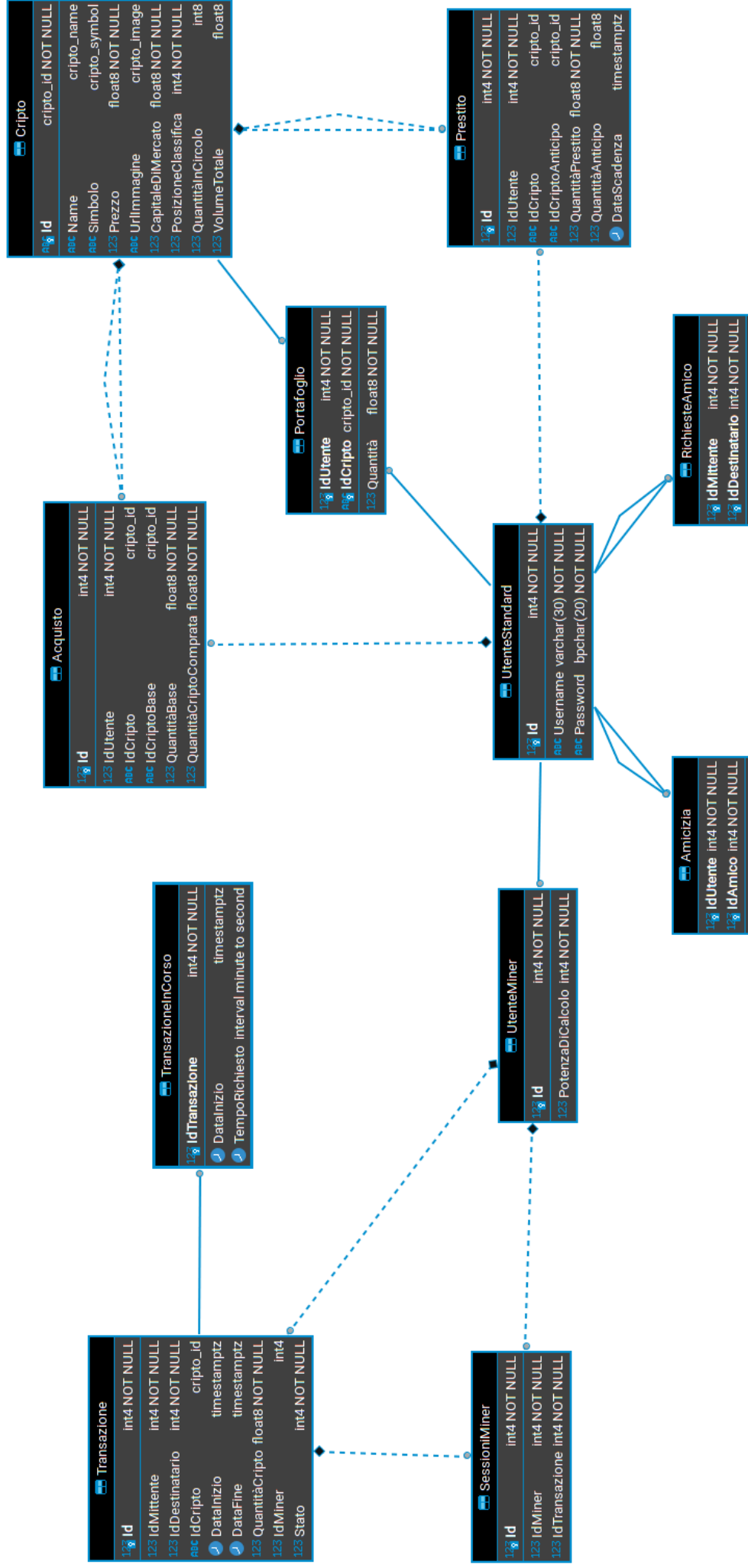
Il **Portafoglio** presenta tutte le cripto possedute è personale per ogni Utente e permette le Transazioni tra Utenti.

E' identificato dall' Id utente e dall' *Id* delle cripto, inoltre contiene la quantità di cripto posseduta.



3.6 – Schema Generale

Di seguito verrà riportato lo schema concettuale generale, contenente tutte le entità e associazioni prima citate nelle varie sezioni superiori collegate tra loro e con altre entità non spiegate nel dettaglio



4 – Progettazione Logica

4.1 – Stima del volume dei dati

Soggetto	Tipo	Volume
UtenteStandard	E	500 000
Acquisto	E	10 000
Prestito	E	3 000
Vendita	E	10 000
Cripto	E	300
Portafoglio	E	10 000 000
SessioneMiner	E	10 000
Miner	E	1 000
Transazione	E	1 000 000
TransazioneInCorso	E	10 000
Amicizia	E	300
RichiestaAmicizia	E	10
ErogazioneCripto	A	400000
AggiuntaPortafoglio	A	400000
AvvioTransazione	A	1 000 000
DestinazioneTransazione	A	1 000 000
ContributoMiner	A	500
PartecipazioneMiner	A	1 000
EffettuazioneMovimento	A	100 000

4.2 – Descrizione delle operazioni principali e stima della loro frequenza

*Le operazioni da effettuare sono quelle già elencate nella fase di analisi.
Segue una tabella riportante la loro descrizione e relativa frequenza:*

Codice	Descrizione Operazione	Frequenza
1	Acquisto, Vendita o Prestito crypto	1 000 000 a settimana
2	Transazioni tra utenti	50 al giorno
3	Visualizzazione Transazione	200 000 al giorno
4	Visualizzazione dati crypto	500 000 al giorno
5	Completamento Transazioni in corso da parte del Miner	50 al giorno
6	Visualizzazione Amicizie dell'Utente loggato	500 000 al giorno
7	Invio Richieste di amicizie	50 al giorno
8	Accettare Richiesta di amicizia	10 a settimana
9	Visualizzazione Transazioni filtrate in base allo stato	200 000 al giorno

10	Visualizzazione Cripto possedute nel Portafoglio	500 000 al giorno
----	--	-------------------

4.3 – Schemi di navigazione e tabelle degli accessi

Sono riportate in seguito una serie di tabelle le quali descrivono, in maniera semplice, i costi delle operazioni di lettura e scrittura in base alla loro frequenza, si considerano di valore doppio gli accessi di tipo *scrittura*.

(4.3.1) Acquisto, vendita o prestito Cripto

Nelle tabelle *Acquisto* (che tiene conto anche per vendita) e *Prestito* viene aggiunta 1 tupla rispettivamente.

Concetto	Costrutto	Accessi	Tipo
Acquisto	E	1	S
Prestito	E	1	S

Siccome si contano 1'000'000 di accessi settimanali come totale delle due tabelle segue come costo:

$$1S * 1'000'000 \text{ a settimana} = 2'000'000/\text{settimana}$$

(4.3.2) Transazioni tra utenti

Inserimento di una tupla nella tabella *Transazione*.

Concetto	Costrutto	Accessi	Tipo
Transazione	E	1	S

Un utente avvia una transazione, questo succede circa 50 volte al giorno il costo assumerà valore:

$$1S * 50 \text{ al giorno} = 100/\text{giorno}$$

(4.3.3) Visualizzazione di una transazione

Un utente decide di vedere lo stato di una determinata transazione.

Concetto	Costrutto	Accessi	Tipo
Transazione	E	1	L

Costo:

$$1L * 200'000 \text{ al giorno} = 200'000/\text{giorno}$$

(4.3.4) Visualizzazione cripto valute

Ogni utente visualizza dati su cripto valute con molta frequenza dato che sono la colonna portante della piattaforma e compaiono in evidenza all'avvio dell'applicazione.

Concetto	Costrutto	Accessi	Tipo
Cripto	E	1	L

Costo:

$$1L * 500'000 \text{ al giorno} = 500'000/\text{giorno}$$

(4.3.5) Miner conferma una transazione

Un miner può decidere di contribuire ad una transazione in corso offrendo capacità di calcolo, questo fa sì che la transazione si completi immediatamente, cancellando una tupla da *TransazioneInCorso* e in seguito aggiornando una tupla nella tabella *Transazione*.

Concetto	Costrutto	Accessi	Tipo
TransazioneInCorso	E	1	S
Transazione	E	1	S

Costo:

$$(1S + 1S) * 50 \text{ al giorno} = 200/\text{giorno}$$

(4.3.6) Visualizzazione amici da parte di un utente

Un utente online sulla piattaforma interagisce con altri utenti, suoi amici, tramite transazioni o semplice revisione dei loro dati.

Concetto	Costrutto	Accessi	Tipo
Amicizia	E	1	L

Costo:

$$1L * 500'000 \text{ al giorno} = 500'000/\text{giorno}$$

(4.3.7) Invio richieste di amicizie

Per poter interagire con un certo utente, ogni utente deve averlo come amico, questo si fa mandando una richiesta di amicizia a quest'ultimo.

Concetto	Costrutto	Accessi	Tipo
RichiesteAmico	E	1	S

Costo:

$$1S * 50 \text{ al giorno} = 100/\text{giorno}$$

(4.3.8) Accettare una richiesta di amicizia

Quando un utente accetta una richiesta in arrivo questo comporta che una tupla venga eliminata da *RichiesteAmico* e 1 tupla venga aggiunta a *Amicizia*.

Concetto	Costrutto	Accessi	Tipo
RichiesteAmico	E	1	S
Amicizia	E	1	S

Costo:

$$(1S + 1S) * 10 \text{ a settimana} = 40/\text{settimana}$$

(4.3.9) Visualizzazione transazione in base allo stato

Un utente per vedere, ad esempio, tutte le transazioni in corso sulla piattaforma deve farlo per mezzo dei filtri che sono implementati nella applicazione, in questo modo accede ad una vista esclusiva alle transazioni in corso.

Concetto	Costrutto	Accessi	Tipo
Transazione	E	1	L

Costo:

$$1L * 200'000 \text{ al giorno} = 200'000/\text{giorno}$$

(4.3.10) Visualizzazione cripto nel portafoglio

Ogni utente possiede un portafoglio che consulta con regolarità.

Concetto	Costrutto	Accessi	Tipo
Portafoglio	E	1	L

Costo:

$$1L * 500'000 \text{ al giorno} = 500'000/\text{giorno}$$

4.4 – Raffinamento dello schema

4.4.1 – Eliminazione delle gerarchie

Come visibile nella raffigurazioni di schemi ER, compare una gerarchia in corrispondenza dell'entità *MovimentoBancario*, questa presenta delle entità figlie (Acquisto, Vendita e Prestito) le quali sono diventate poi delle vere e proprie entità senza tener conto di nessuna entità *MovimentoBancario*.

Passando alla progettazione logica è stato deciso di rendere Acquisto una tabella risultante dalla combinazione del concetto di acquisto e di vendita, questo risultato si ottiene facilmente facendo sì che quando avviene una

vendita, piuttosto che un acquisto, vengono scambiati i valori di *Cripto* e *CriptoBase* e di *QuantitàBase* e *QuantitàCriptoComprata*.

4.6 – Traduzioni di entità e associazioni in relazioni

UTENTE(Id, Nome, Username, Password)

PORTAFOGLIO(IdUtente, IdCripto : CRIPTO, Quantità)

CRIPTO(Id, Simbolo, Nome, Prezzo, UrlImmagine, CapitaleDiMercato, PosizioneClassifica, QuantitaCircolo, VolumeTotale)

ACQUISTO(Id, IdUtente : UTENTE, IdCripto : CRIPTO, QuantitaBase, QuantitaAcquisto, IdCriptoBase : CRIPTO)

PRESTITO(Id, IdUtente : UTENTE, IdCripto : CRIPTO, QuantitaPrestito, Anticipo, Scadenza, IdCriptoAnticipo : CRIPTO)

MINER(Id : UTENTE, PotenzaMining)

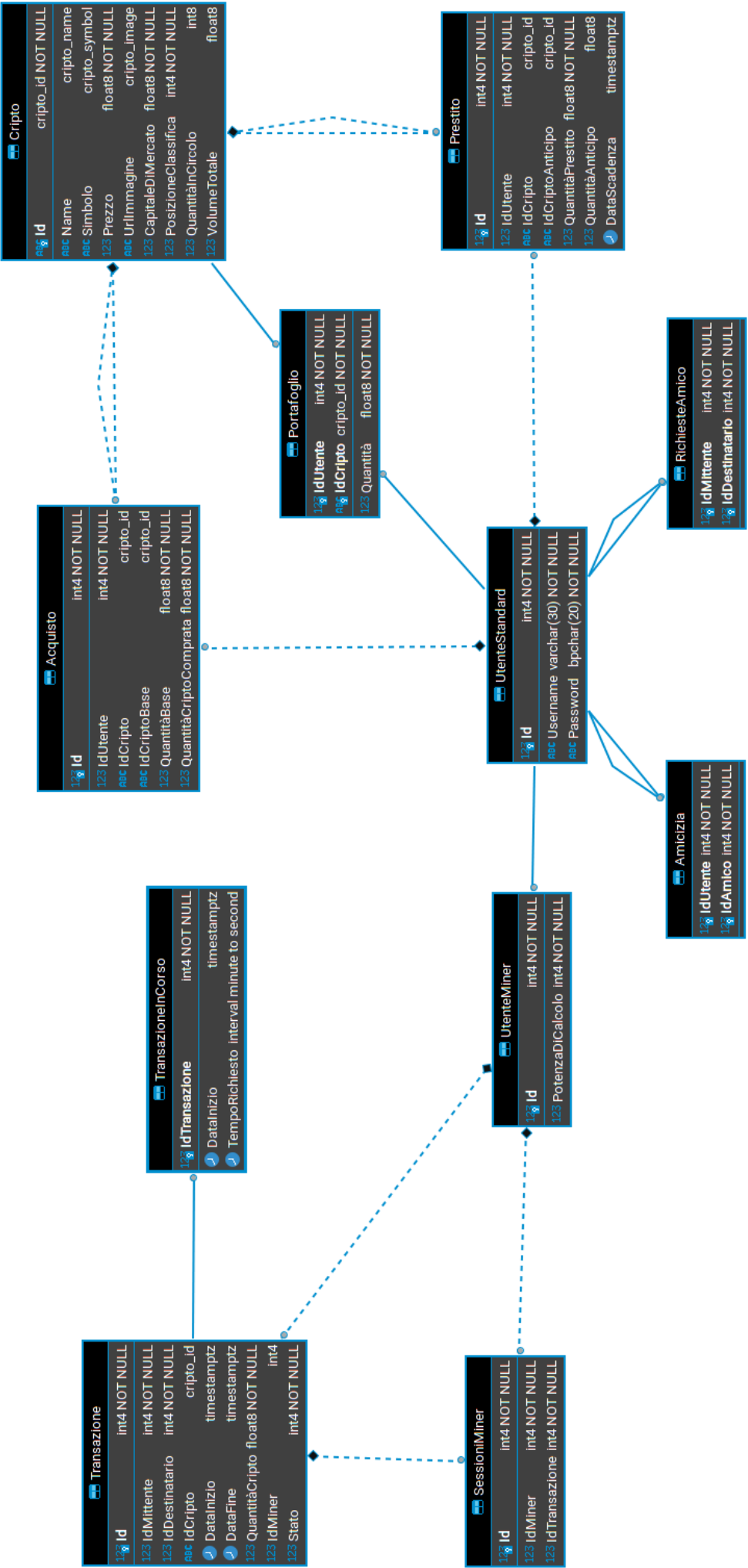
TRANSAZIONI_IN_CORSO(IdTransazione : TRANSAZIONE, DataInizio, TempoTotale)

TRANSAZIONE(Id, IdMittente, : UTENTE, IdDestinazione : UTENTE, IdCripto : CRIPTO, DataInizio, DataFine*, QuantitaCripto, IdMiner* : MINER, Stato)

AMICIZIA(IdUtente : UTENTE, IdAmico : UTENTE)

RICHIESTA_AMICIZIA(IdMittente : UTENTE, IdDestinatario : UTENTE)

4.7 – Schema Relazione finale



5 – Progettazione Fisica

5.1 – Traduzione in SQL

1) Acquisto, Vendita o Prestito cripto.

```
INSERT INTO Acquisto(Id, IdUtente, IdCripto, IdCriptoBase,  
QuantitaBase, QuantitaCriptoComprata) VALUES (?, ?, ?, ?, ?, ?);
```

```
INSERT INTO Prestito(Id, IdUtente, IdCripto,  
IdCriptoAnticipo, QuantitaPrestito, QuantitaCriptoAnticipo,  
DataScadenza) VALUES (?, ?, ?, ?, ?, ?);
```

2) Transazioni tra utenti.

```
INSERT INTO Transazione(Id, IdMittente, IdDestinatario,  
IdCripto, DataInizio, DataFine, QuantitaCripto, IdMiner)  
VALUES (?, ?, ?, ?, ?, ?, ?);
```

3) Visualizzazione Transazioni.

```
SELECT  
Id, IdMittente, IdDestinatario, IdCripto, DataInizio,  
DataFine, QuantitaCripto, IdMiner  
FROM Transazione;
```

4) Visualizzazione dati cripto.

```
SELECT Id, Nome, Simbolo  
FROM Cripto;
```

5) Completamento Transazioni in corso da parte del Miner. (213 = Id del miner)

```
UPDATE Transazione
SET Stato = 1, IdMiner = 213
FROM (
    SELECT IdTransazione
    FROM SessioniMiner
    WHERE IdMiner = 213 ) as Confermate
WHERE Transazione.Id = Confermate.IdTransazione
```

6) Visualizzazione Amicizie dell'Utente loggato (212 = Id utente loggato).

```
SELECT Id, Username
FROM UserStandard
WHERE Id = (
    SELECT IdAmico
    FROM Amicizia
    WHERE IdUtente = 212 );
```

7) Invio Richieste di amicizie (212 = Id utente loggato)

```
INSERT INTO RichiesteAmico(IdMittente, IdDestinatario) VALUES
(212,?);
```

8) Visualizzazione Transazioni filtrate in base allo stato.

```
SELECT *
FROM Transazione
WHERE Stato = ??
```

9) Accettare Richiesta di amicizia.

```
INSERT INTO Amicizia(IdUtente, IdAmico) VALUES (?,?);
```

10) Visualizzazione Cripto possedute nel Portafoglio (212 = Id utente loggato).

```
SELECT *  
FROM Portafoglio  
WHERE IdUtente = 212
```

6 – Progettazione dell'Applicazione

6.1 – Descrizione della scelta del linguaggio e del DBMS

L'applicazione è realizzata in C#, utilizzando per l'UI la libreria ETO Forms che ci permette di realizzare interfacce cross-platforms e molto pulite e intuitive.

Per l'interrogazione del database dall'applicazione abbiamo sfruttato una versione di Entity Framework adattata per PostgreSQL che, tramite sintassi Linq, ci permette di effettuare interrogazioni al database in maniera veloce e intuitiva.

Il database risiede in locale e il DBMS utilizzato è PostgreSQL

6.2 – Descrizione dell'architettura

L'applicazione presenta all'avvio una finestra di login, una volta autenticati sono presenti 4 finestre navigabili attraverso un drawer: Lista di cripto, Portafoglio, Transazioni e Profilo.

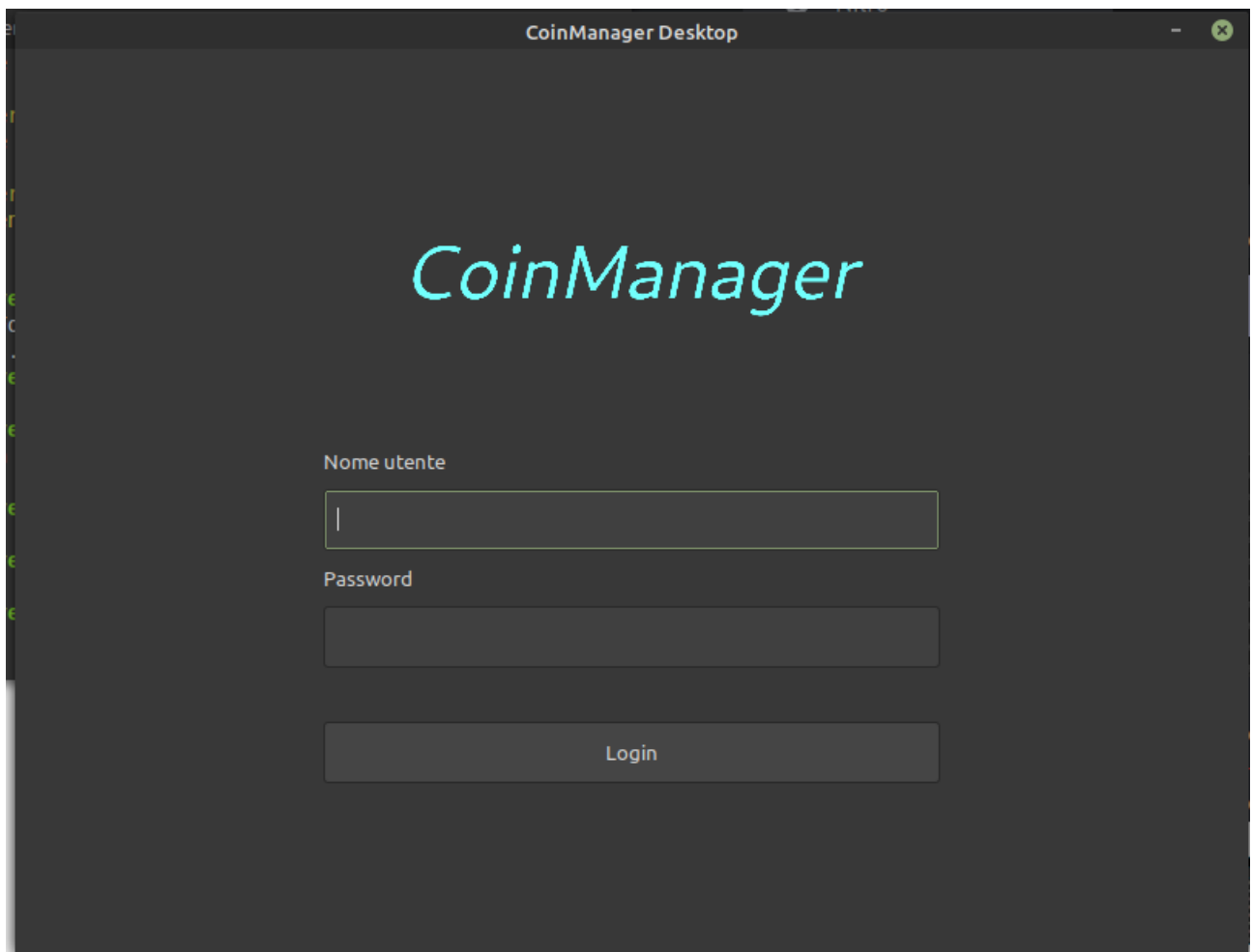
Gli utenti disponibili sono: Utente Standard e Miner

6.3 – Interfaccia Utente

6.3.1 – Autenticazione

La pagina di autenticazione permette di accedere all'applicativo attraverso credenziali , per la versione utilizzata gli utenti sono già stati creati e non è presente un metodo per registrarsi e aggiungere nuovi utenti.

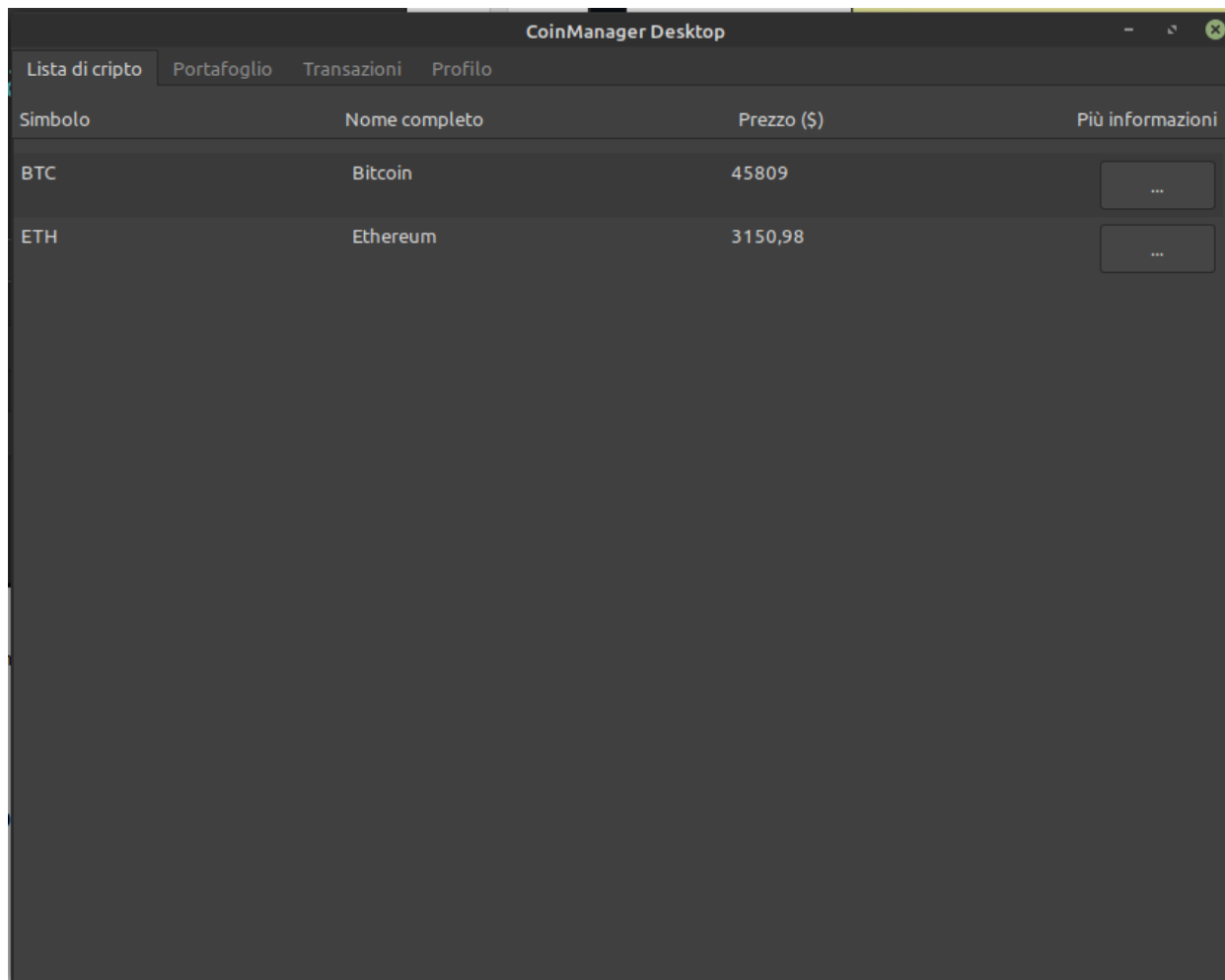
Una volta premuto il pulsante di login al primo avvio viene chiamato un metodo che popola il db con le criptovalute e tutti i suoi dati presi da CoinGecko



In figura è mostrata la schermata di login

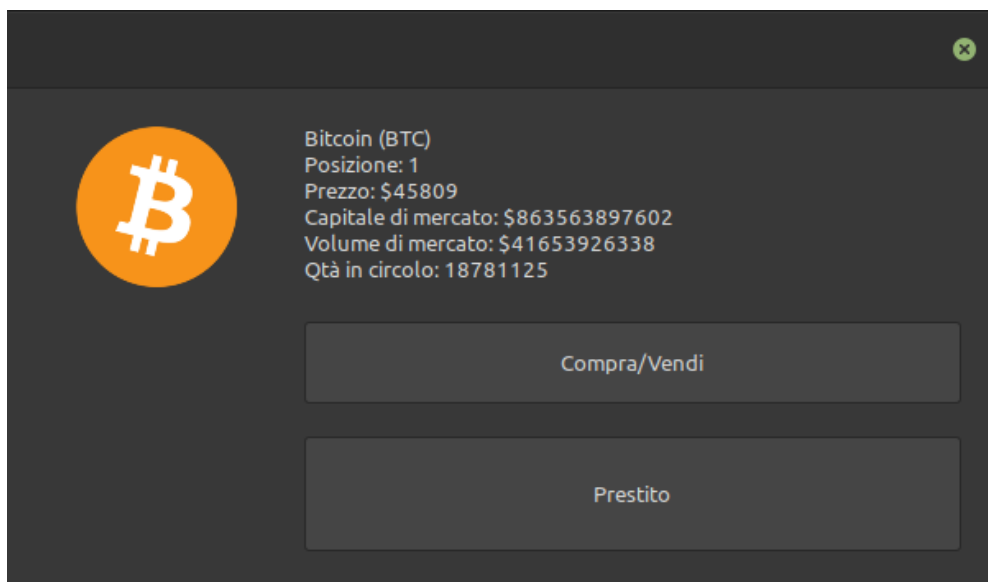
6.3.2 - Lista di crypto

La finestra presenta una lista con tutte le crypto disponibili, qualche informazione utile e un bottone che permette di effettuare operazioni sulle crypto.

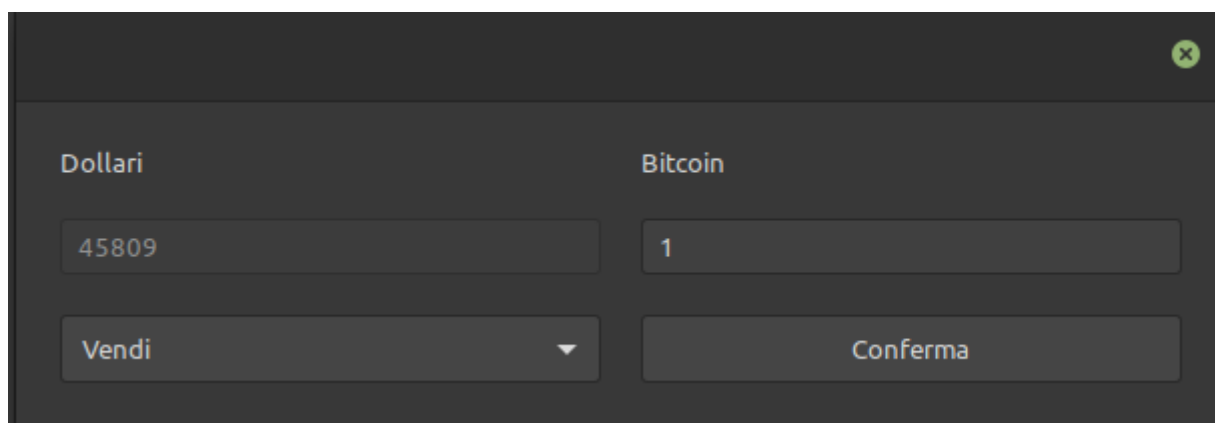


CoinManager Desktop			
Lista di crypto Portafoglio Transazioni Profilo			
Simbolo	Nome completo	Prezzo (\$)	Più informazioni
BTC	Bitcoin	45809	...
ETH	Ethereum	3150,98	...

In figura la schermata principale delle crypto



In figura la schermata di compravendita di crypto



In figura la schermata di acquisto/vendita crypto

6.3.3 - Portafoglio

La finestra presenta la lista di crypto di un utente, si ha la possibilità di effettuare transazioni attraverso il pulsante avvia
 E' presente una tabella con tutte le ultime transazioni

CoinManager Desktop

Lista di criptoPortafoglioTransazioniProfilo

Le tue criptovalute

Cripto Id	Quantità
bitcoin	1
ethereum	17

Invia

Ultime transazioni

Id	Emittente	Destinatario	Cryptoid	Stato	Inizio	Fine	Miner	Quantità	
1	1	2	bitcoin	1	03/09/2021 14:37:32	03/09/2021 14:42:25	0	20	
2	1	2	ethereum	1	03/09/2021 15:17:02	03/09/2021 15:21:48	0	6	
3	1	2	ethereum	1	03/09/2021 16:06:49	03/09/2021 16:10:51	0	2	
4	1	2	bitcoin	1	03/09/2021 16:11:01	03/09/2021 16:14:01	0	1	
5	1	2	ethereum	1	03/09/2021 16:14:12	03/09/2021 16:15:24	0	4	
6	1	2	ethereum	1	03/09/2021 16:19:40	03/09/2021 16:23:41	0	1	
7	1	2	ethereum	1	03/09/2021 16:23:52	03/09/2021 16:28:31	0	1	
8	1	2	ethereum	1	03/09/2021 16:28:42	03/09/2021 16:31:48	0	1	
9	1	2	ethereum	1	03/09/2021 16:31:57	03/09/2021 16:41:00	0	1	
10	1	2	ethereum	1	03/09/2021 16:41:16	03/09/2021 16:42:00	0	1	
12	1	2	ethereum	1	03/09/2021 16:50:35	03/09/2021 16:53:47	0	3	
11	1	2	ethereum	1	03/09/2021 16:50:20	03/09/2021 16:53:51	0	2	
13	1	2	ethereum	1	03/09/2021 16:54:03	03/09/2021 16:54:53	0	1	
14	1	2	ethereum	1	03/09/2021 16:55:06	03/09/2021 17:13:30	0	1	

In figura la schermata del portafoglio

Destinatario

Critpovaluta da inviare

Quantità

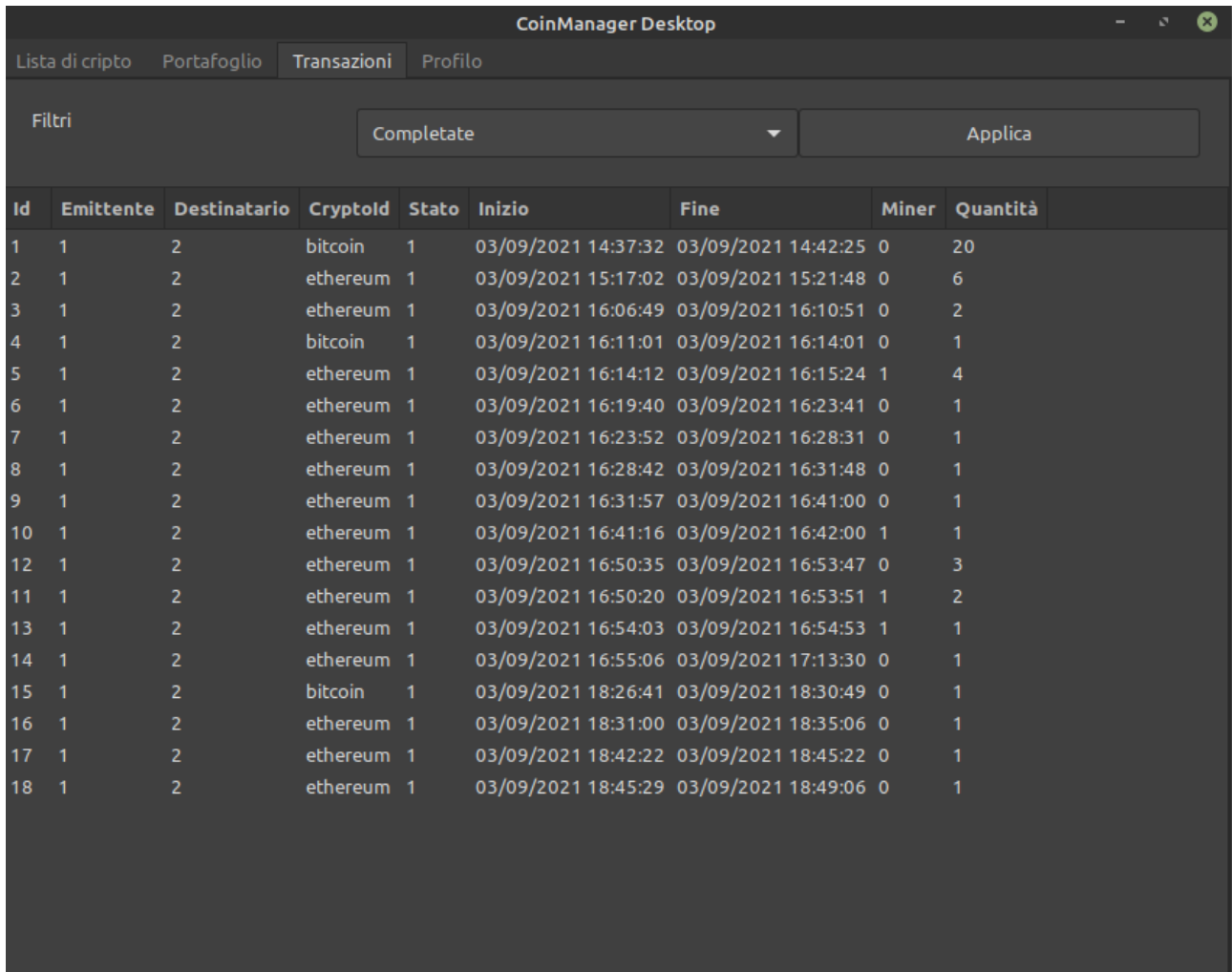
Quantità che rimarrà

Avvia

In figura la schermata una volta premuto il pulsante Invia

6.3.4 - Transazioni

La schermata consente di visualizzare le transazioni filtrate per stato



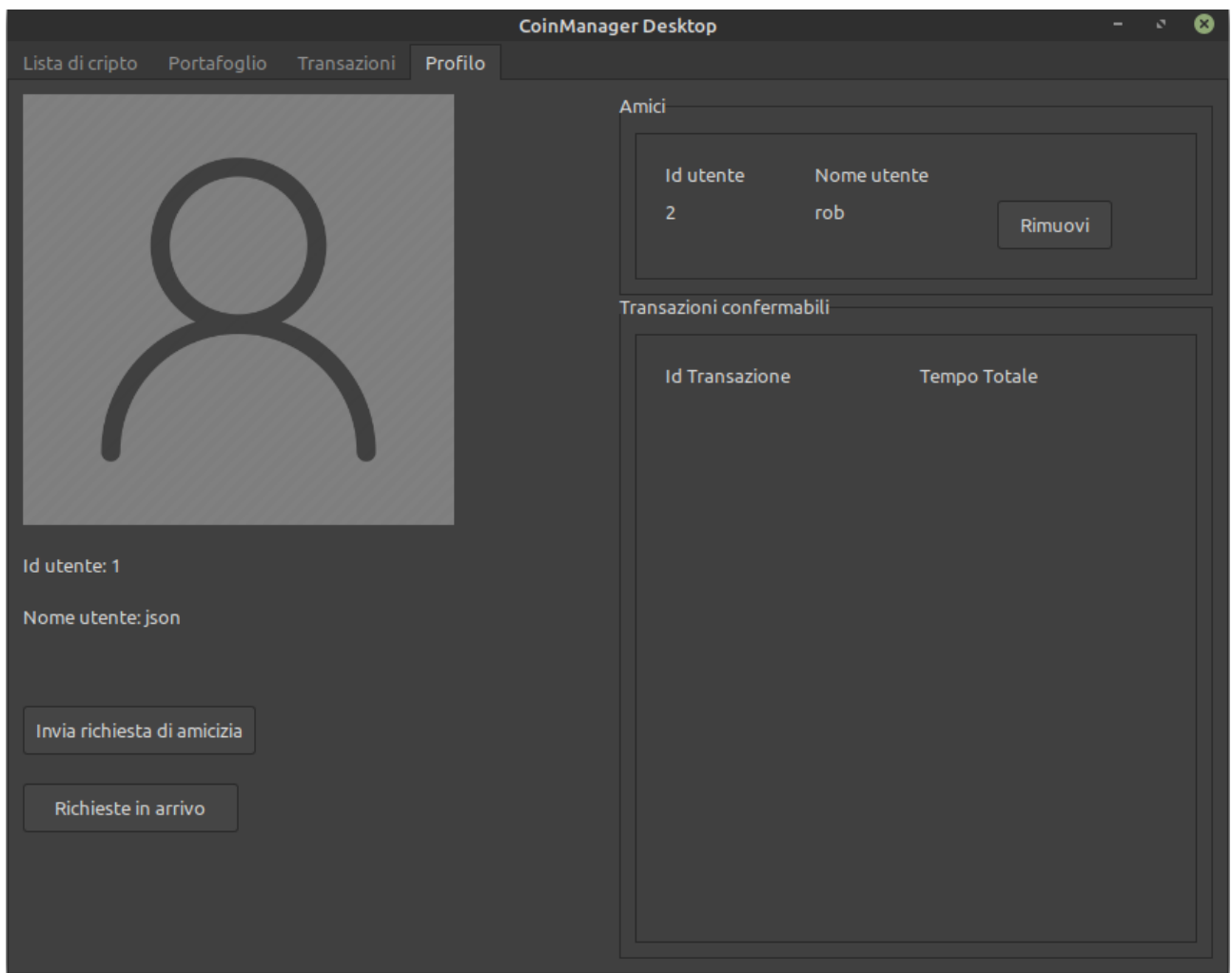
CoinManager Desktop									
Lista di cripto Portafoglio Transazioni Profilo									
Filtri									
Completate						Applica			
Id	Emittente	Destinatario	Cryptoid	Stato	Inizio	Fine	Miner	Quantità	
1	1	2	bitcoin	1	03/09/2021 14:37:32	03/09/2021 14:42:25	0	20	
2	1	2	ethereum	1	03/09/2021 15:17:02	03/09/2021 15:21:48	0	6	
3	1	2	ethereum	1	03/09/2021 16:06:49	03/09/2021 16:10:51	0	2	
4	1	2	bitcoin	1	03/09/2021 16:11:01	03/09/2021 16:14:01	0	1	
5	1	2	ethereum	1	03/09/2021 16:14:12	03/09/2021 16:15:24	1	4	
6	1	2	ethereum	1	03/09/2021 16:19:40	03/09/2021 16:23:41	0	1	
7	1	2	ethereum	1	03/09/2021 16:23:52	03/09/2021 16:28:31	0	1	
8	1	2	ethereum	1	03/09/2021 16:28:42	03/09/2021 16:31:48	0	1	
9	1	2	ethereum	1	03/09/2021 16:31:57	03/09/2021 16:41:00	0	1	
10	1	2	ethereum	1	03/09/2021 16:41:16	03/09/2021 16:42:00	1	1	
12	1	2	ethereum	1	03/09/2021 16:50:35	03/09/2021 16:53:47	0	3	
11	1	2	ethereum	1	03/09/2021 16:50:20	03/09/2021 16:53:51	1	2	
13	1	2	ethereum	1	03/09/2021 16:54:03	03/09/2021 16:54:53	1	1	
14	1	2	ethereum	1	03/09/2021 16:55:06	03/09/2021 17:13:30	0	1	
15	1	2	bitcoin	1	03/09/2021 18:26:41	03/09/2021 18:30:49	0	1	
16	1	2	ethereum	1	03/09/2021 18:31:00	03/09/2021 18:35:06	0	1	
17	1	2	ethereum	1	03/09/2021 18:42:22	03/09/2021 18:45:22	0	1	
18	1	2	ethereum	1	03/09/2021 18:45:29	03/09/2021 18:49:06	0	1	

In figura la schermata delle transazioni completate

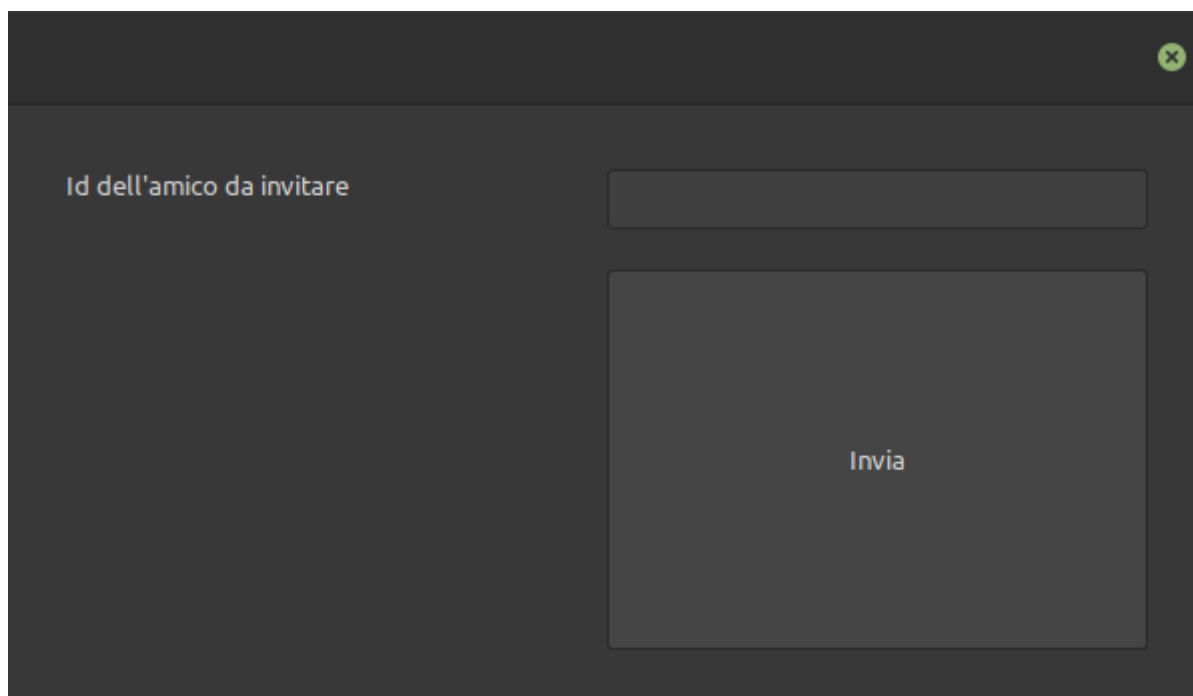
6.3.5 - Profilo

La finestra del Profilo contiene tutte le info dell'utente, la possibilità di inviare e accettare amicizie e la lista di amici.

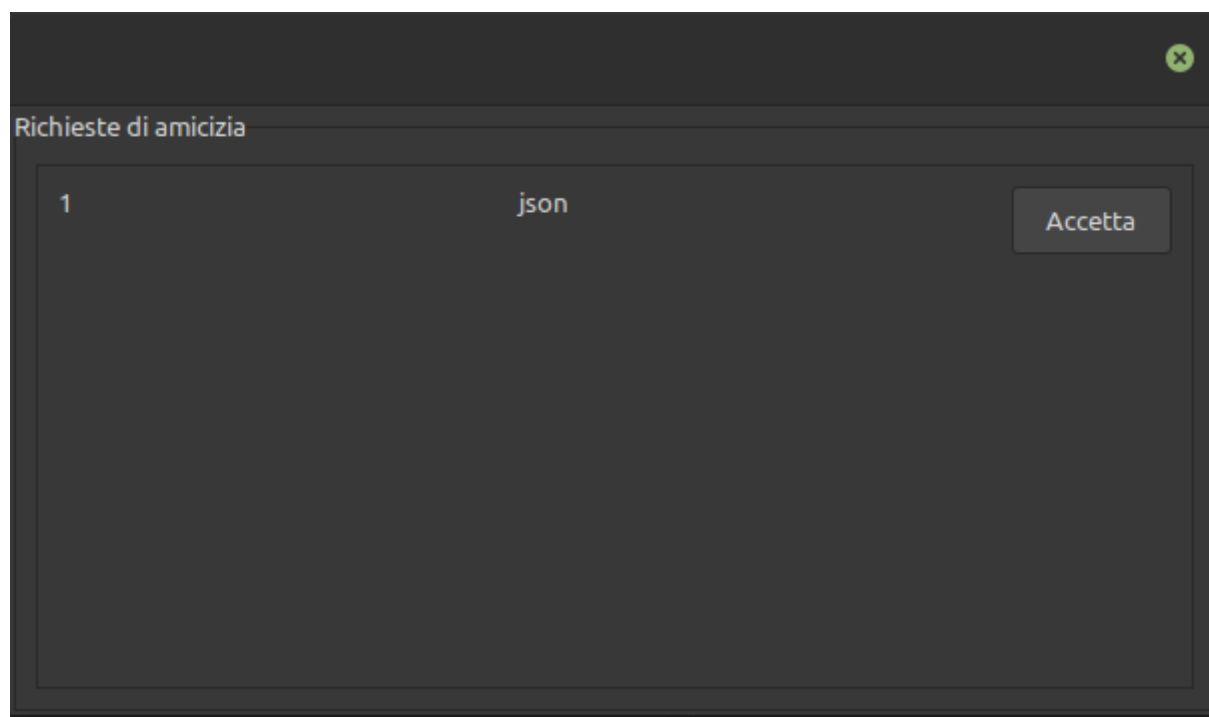
Inoltre se l'utente è Miner è presente una lista di transazioni in corso.



In figura è mostrato il profilo di un utente MIner



In figura finestra di invito amici



In figura la finestra di Richieste di amicizia