# BitMoneto

Daniele Piscaglia 0000766586

## Sommario

[BitMoneto 1](#_Toc522480408)

[Sommario 2](#_Toc522480409)

[Abstract 3](#_Toc522480410)

[Analisi dei requisiti 4](#_Toc522480411)

[Requisiti del sistema 4](#_Toc522480412)

[Analisi del dominio 4](#_Toc522480413)

[Analisi dei requisiti 6](#_Toc522480414)

[Analisi del rischio 10](#_Toc522480415)

[Analisi del problema 15](#_Toc522480416)

[Analisi documento dei requisiti 15](#_Toc522480417)

[Analisi ruoli e responsabilità 17](#_Toc522480418)

[Scomposizione del problema 18](#_Toc522480419)

[Modello del dominio 18](#_Toc522480420)

[Architettura logica: Struttura 18](#_Toc522480421)

[Architettura logica: Interazione 20](#_Toc522480422)

[Architettura logica: Comportamento 24](#_Toc522480423)

[Piano di lavoro 24](#_Toc522480424)

[Piano del collaudo 25](#_Toc522480425)

[Progetto 28](#_Toc522480426)

[Architettura del sistema: Struttura 28](#_Toc522480427)

[Architettura del sistema: Interazione 33](#_Toc522480428)

[Architettura del sistema: Comportamento 38](#_Toc522480429)

[Implementazione 39](#_Toc522480430)

[Collaudo 40](#_Toc522480431)

[Deployment 41](#_Toc522480432)

## Abstract

Si intende progettare un applicazione desktop, che permetta all’utilizzatore di tenere traccia di tutti suoi fondi posseduti in criptovalute in varie posizioni in modo da riassumerli in una sola vista. A tale scopo, per rendere il tutto automatizzato, una volta inseriti i dati necessari per l’autenticazione, l’applicazione dovrà scaricare e mantenere aggiornati i dati dei più famosi siti di exchange e analizzatori di blockchain per gli account/portafogli collegati direttamente senza richiedere l’intervento dell’utente per ogni movimento effettuato. Per ogni fondo posseduto sarà richiesto di mostrare oltre ai dati della criptovaluta anche il valore corrispondente in altre valute fiat e criptovalute.

## Analisi dei requisiti

### Requisiti del sistema

La pagina principale dell’applicazione sarà la tabella che mostra tutti gli attuali fondi posseduti dall’utente, tali fondi verranno scaricati ed aggiornati dai relativi exchange/portafogli all’apertura dell’applicazione ed aggiornati quando espressamente richiesto dall’utente. Per ogni fondo dell’utente deve essere mostrato:

* Nome della valuta
* Simbolo della valuta
* Quantità nel fondo
* Valore corrispondente in Euro, Dollari, Bitcoin ed Ethereum

L’inserimento dei dati necessari per il collegamento ai vari siti dovrà essere effettuato tramite un’apposita scheda nell’ applicazione dove sarà presente una lista di tutte le blockchain o exchange supportati.

### Analisi del dominio

|  |  |  |
| --- | --- | --- |
| Voce | Definizione | Sinonimi |
| Valuta fiat | Normale moneta distribuita e garantita da una nazione o banca | Moneta |
| Criptovaluta | Bene digitale che può essere scambiato con denaro fiat, bene materiale o altra criptovaluta |  |
| Movimento | Compravendita o scambio di criptovaluta | Trade, scambio |
| Exchange | Sito sul quale è possibile comprare e vendere criptovalute con altre o valute fiat dove possono risiedere una parte dei fondi dell’utente |  |
| Account | Si intende un account su un sito di exchange dove si possiedono dei fondi |  |
| Blockchain | Catena di blocchi salvata nella rete dove sono mantenute tutte le transazioni effettuate per una determinata criptovaluta |  |
| Portafoglio | Indirizzo associato e appartenente all’utente che ne identifica la identità e movimenti su una determinata blockchain, al quale è associata una quantità di fondi posseduti | Indirizzo |
| Fondi | Quantità posseduta di ogni criptovaluta su un determinato exchange, portafoglio o in generale |  |
| Esploratore di blockchain | Applicazione che analizza una determinata blockchain una volta fornito un indirizzo di un portafoglio per risalire alla quantità di fondi attualmente posseduta da tale indirizzo | Analizzatore di blockchain |
| Chiavi API | Coppia di due chiavi alfanumeriche pubblica e privata necessarie per collegarsi ai servizi per lo scaricamento dei dati fondi da siti di exchange o esploratori di blockchain | Credenziali |
| REST API | Protocollo di comunicazione utilizzato per reperire i dati aggiornati di exchange e portafogli |  |
| CryptoCompare | Il sito cryptocompare.com verrà utilizzato come riferimento per ottenere i dati aggiornati riguardo il valore di mercato associato di ogni criptovaluta | Sito di cambi |

#### Sistemi esterni

Per collegarsi, ricevere e mantenere aggiornati i dati si utilizzeranno, nel caso dei siti di exchange le chiamate REST API proprietarie, mentre per i portafogli saranno utilizzati gli esploratori di blockchain di terze parti più famosi (come Block Explorer per Bitcoin e Etherscan per Ethereum) sempre tramite le REST API da loro fornite.

Per ogni criptovaluta, verrà anche mostrato il valore convertito nelle più importanti valute fiat e criptovalute (Euro, Dollaro, Bitcoin, Ethereum). Tal tasso di conversione deve essere reperito dalla rete, a tale scopo, è possibile utilizzare le API gratuitamente dal sito cryptocompare.com che fornisce tutte le informazioni di una criptovaluta (nome, simbolo e attuale cambio facendo una media dei valori sui principali exchange mondiali).

### Analisi dei requisiti

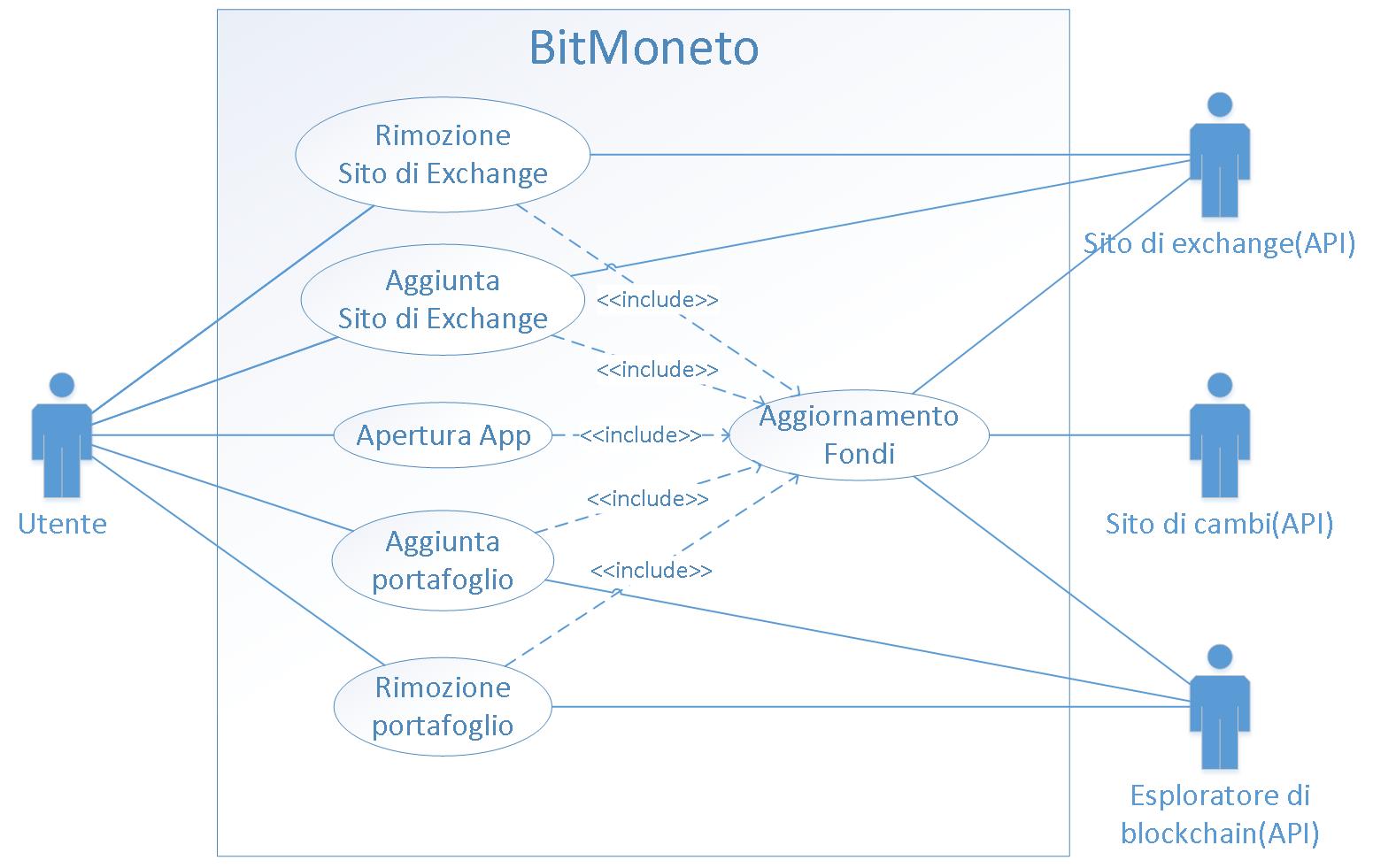
#### Requisiti funzionali

* Appena aperta l’applicazione deve essere avviato lo scaricamento dei fondi aggiornati
* Tramite una scheda di impostazioni apposita deve essere possibile aggiungere (o rimuovere) una coppia di chiavi API necessarie per il collegamento ad ogni sito di exchange per lo scaricamento dei dati dei fondi. Ogni volta che vengono aggiunte delle nuove chiavi deve partire in automatico il relativo aggiornamento dei fondi.
* Tramite la stessa scheda per l’aggiunta delle chiavi API deve essere possibile aggiungere (o rimuovere) un indirizzo di portafoglio da monitorare per ogni blockchain supportata. Anche in questo caso, appena un portafoglio viene aggiunto si devono riaggiornare i fondi.
* L’utente deve poter richiedere un aggiornamento dei dati in qualsiasi momento desideri tramite l’interfaccia grafica
* L’interfaccia grafica per ogni valuta deve mostrare il valore corrispondente in euro, dollari, bitcoin ed ethereum a tale scopo bisogna scaricare il relativo tasso di conversione dal sito cryptocompare.com

#### Requisiti non funzionali

* I dati che vengono scaricati dai siti di exchange o esploratori di blockchain vanno reperiti dalla rete quindi è comprensibile che vi siano ritardi notevoli, per questo mentre è attiva la sincronizzazione dei dati deve essere mostrato in maniera non invasiva e non bloccante nell’ applicazione un animazione di caricamento.
* I dati devono essere ricevuti tramite delle API le quali potrebbero avere un numero di richieste limitato effettuabili in un arco di tempo limitato.
* Tutti i dati dell’utente relativi ad exchange e portafogli dovranno essere salvati in locale sulla macchina per essere ricaricati alla prossima apertura.

#### Casi d’uso



#### Scenari

|  |  |
| --- | --- |
| Titolo | Apertura App |
| Descrizione | L’utente avvia l’applicazione |
| Attori | Utente |
| Relazioni | Aggiornamento Fondi |
| Precondizioni |  |
| Postcondizioni |  |
| Scenario principale | 1. Viene caricata la schermata principale 2. Viene avviato un aggiornamento dei fondi |
| Scenari alternativi |  |
| Requisiti non funzionali |  |
| Punti aperti |  |

|  |  |
| --- | --- |
| Titolo | Aggiornamento fondi |
| Descrizione | Viene avviato lo scaricamento o sincronizzazione dei fondi dalla rete |
| Attori | Utente, Sito di exchange(API), Esploratore di blockchain(API), Sito di cambi(API) |
| Relazioni | Apertura App, Aggiunta sito Exchange, Aggiunta portafoglio |
| Precondizioni | Ci sono delle chiavi API o dei portafogli memorizzati |
| Postcondizioni |  |
| Scenario principale | 1. Viene avviato lo scaricamento dei dati 2. Finché la sincronizzazione dei fondi non è completata viene mostrata una icona di caricamento ma l’applicazione continua a essere utilizzabile 3. Per ogni valuta che è contenuta in un portafoglio o in un sito di exchange sarà anche necessario utilizzare il sito di cambi per scaricare il cambio nelle rispettive valute principali richieste |
| Scenari alternativi | Scenario a: Non vi sono chiavi API o portafogli memorizzati   1. Si mostra all’utente un messaggio che lo invita ad aggiungere delle chiavi API o dei portafogli   Scenario b: Errore durante lo scaricamento dei dati   1. Viene mostrato un errore all’utente |
| Requisiti non funzionali | Velocità di scaricamento dei dati e utilizzo dell’applicazione durante l’operazione |
| Punti aperti |  |

|  |  |
| --- | --- |
| Titolo | Aggiunta sito di exchange |
| Descrizione | L’utente immette le chiavi API di un nuovo sito di exchange |
| Attori | Utente, Sito di Exchange(API) |
| Relazioni | Aggiornamento Fondi |
| Precondizioni | L’utente ha immesso delle nuove chiavi API |
| Postcondizioni | Le chiavi API vengono registrate nell’applicazione |
| Scenario principale | 1. Vengono lette le chiavi API inserite dall’utente 2. Si controllano le chiavi API e che non siano già state associate delle chiavi API per lo stesso exchange 3. Le chiavi vengono registrate nell’applicazione 4. Si avvia un aggiornamento dei fondi |
| Scenari alternativi | Scenario a: Chiavi API errate o già presenti  3. Viene mostrato un messaggio di errore |
| Requisiti non funzionali |  |
| Punti aperti |  |

|  |  |
| --- | --- |
| Titolo | Aggiunta portafoglio |
| Descrizione | L’utente immette l’indirizzo di un nuovo portafoglio |
| Attori | Utente, Esploratore di blockchain(API) |
| Relazioni | Aggiornamento Fondi |
| Precondizioni | L’utente ha immesso un nuovo indirizzo di portafoglio |
| Postcondizioni | Il nuovo portafoglio viene registrato nell’applicazione |
| Scenario principale | 1. Viene letto l’indirizzo inserito dall’utente 2. Si effettua un controllo dell’indirizzo e che non sia già stato aggiunto un indirizzo per quella stessa blockchain 3. L’indirizzo viene registrato nell’applicazione 4. Si avvia un aggiornamento dei fondi |
| Scenari alternativi | Scenario a: Indirizzo non corretto o già presente per quella blockchain  3. Viene mostrato un messaggio di errore |
| Requisiti non funzionali |  |
| Punti aperti |  |

|  |  |
| --- | --- |
| Titolo | Rimozione sito di exchange |
| Descrizione | L’utente rimuove delle chiavi API di un sito di exchange |
| Attori | Utente, Sito di Exchange(API) |
| Relazioni | Aggiornamento Fondi |
| Precondizioni |  |
| Postcondizioni | Le chiavi API sono state rimosse dall’applicazione |
| Scenario principale | 1. Le chiavi vengono cancellate dall’applicazione 2. Si avvia un aggiornamento dei fondi |
| Scenari alternativi |  |
| Requisiti non funzionali |  |
| Punti aperti |  |

|  |  |
| --- | --- |
| Titolo | Rimozione portafoglio |
| Descrizione | L’utente rimuove l’indirizzo di un nuovo portafoglio |
| Attori | Utente, Esploratore di blockchain(API) |
| Relazioni | Aggiornamento Fondi |
| Precondizioni |  |
| Postcondizioni | Il portafoglio è stato rimosso dall’applicazione |
| Scenario principale | 1. L’indirizzo viene rimosso dall’applicazione 2. Si avvia un aggiornamento dei fondi |
| Scenari alternativi |  |
| Requisiti non funzionali |  |
| Punti aperti |  |

### Analisi del rischio

#### Tabella valutazione dei beni

|  |  |  |
| --- | --- | --- |
| **Bene** | **Valore** | **Esposizione** |
| Chiavi API | Alto. Forniscono potenzialmente l’accesso a tutti i fondi posseduti sull’Exchange associato | Molto Alta. Perdita di immagine |
| Indirizzo di portafoglio | Medio. Informazione che non dà accesso a ai fondi posseduti ma può comunque essere considerata confidenziale dal cliente | Media. Perdita di immagine |
| File dei dati dell’applicazione | Molto Alto. Contiene tutte le informazioni | Molto Alta. Perdita di immagine |

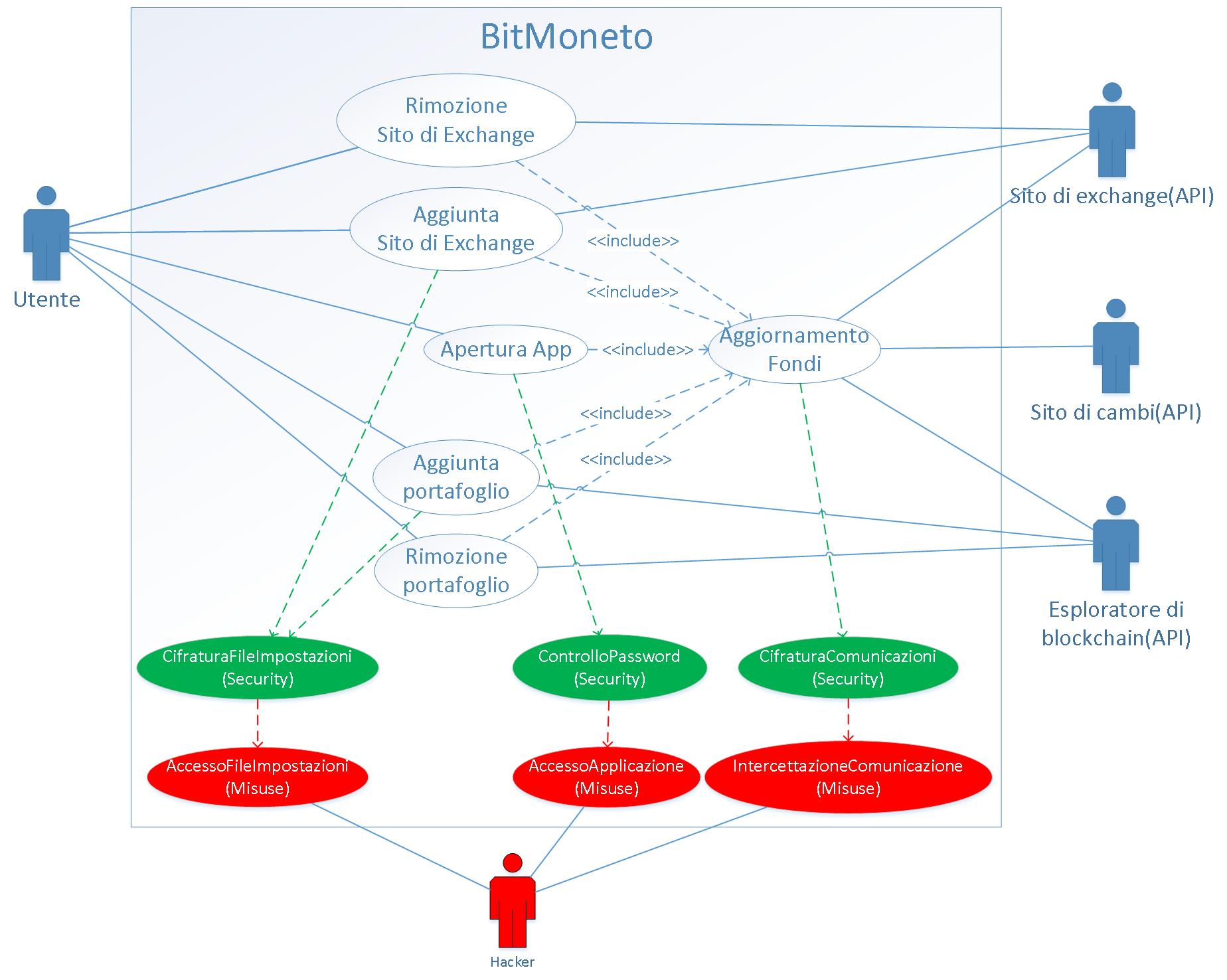
#### Tabella Minacce/Controlli

|  |  |  |  |
| --- | --- | --- | --- |
| **Minaccia** | **Probabilità** | **Controllo** | **Fattibilità** |
| Furto delle chiavi API | Alta. | Non mostrare le chiavi API una volta inserite | Basso costo, una volta inserite dall’utente non sarà possibile visualizzarle ma solo cancellarle |
| Invitare l’utente ad utilizzare chiavi API in sola lettura | Basso costo, inserire un semplice messaggio all’utente dove lo si invita a impostare le chiavi API in modo da non poter effettuare operazioni sui fondi, ma rimarrà comunque libero di utilizzare chiavi con tutti i permessi |
| Proteggere l’applicazione con una password | Basso costo di realizzazione |
| Furto degli indirizzi di portafoglio | Alta. Visualizzabili direttamente nell’applicazione | Proteggere l’applicazione con una password | Basso costo di realizzazione |
| Furto del file di dati delle applicazioni | Molto Alta. Il file andrà salvato sul disco fisso della macchina dove verrà installata l’applicazione | Cifrare le informazioni nel file | Medio costo di realizzazione |
| Intercettazione comunicazioni nelle chiamate API | Alta. Ad ogni aggiornamento è necessario effettuare numerose chiamate API | Cifratura delle comunicazioni | Medio costo di realizzazione, utilizzare comunicazioni sicure cifrate end-to-end quando si trasmettono dati sensibili |

#### Analisi Tecnologica della Sicurezza

|  |  |
| --- | --- |
| **Tecnologia** | **Vulnerabilità** |
| Autenticazione password | * Password banale * L’utente rivela la password volontariamente o involontariamente in seguito a un attacco di ingegneria sociale |
| Cifratura file impostazione | In base al tipo di cifratura.  Cifratura simmetrica:   * Tempo di vita della chiave * Memorizzazione della chiave * Stessa chiave per tutte le installazioni dell’applicazione   Cifratura asimmetrica:   * Memorizzazione chiave privata |
| Cifratura comunicazioni |  |

#### Security Use Case e Misuse Case



#### Security Use Case e Misuse Case Scenari

|  |  |  |
| --- | --- | --- |
| **Titolo** | CifraturaFileImpostazioni | |
| **Descrizione** | Le informazioni nel file vengono cifrate | |
| **Misuse case** | AccessoFileInformazioni | |
| **Relazioni** |  | |
| **Precondizioni** | L’attaccante ha accesso alla macchina dove l’applicazione è installata o almeno ai suoi file | |
| **Postcondizioni** | L’applicazione mantiene sotto controllo il file di impostazioni annullando il caricamento dei dati se il file viene compromesso | |
| **Scenario principale** | *Sistema* | *Attaccante* |
|  | Accede al file senza poterlo leggere non potendolo decifrare |
| **Scenario di attacco avvenuto con successo** | *Sistema* | *Attaccante* |
|  | Modifica i valori nel file |
| L’applicazione alla prossima lettura se riconosce delle inconsistenze nel file ne interrompe il caricamento |  |

|  |  |  |
| --- | --- | --- |
| **Titolo** | ControlloPassword | |
| **Descrizione** | L’accesso all’applicazione è protetto da una password | |
| **Misuse case** | AccessoApplicazione | |
| **Relazioni** |  | |
| **Precondizioni** | L’attaccante ha accesso alla macchina dove l’applicazione è installata | |
| **Postcondizioni** | L’applicazione viene terminata | |
| **Scenario principale** | *Sistema* | *Attaccante* |
|  | L’attaccante tenta di accedere al sistema inserendo password con un attacco con dizionario |
|  | Dopo un numero di tentativi limitati il sistema impedisce l’accesso per una finestra di tempo |  |
| **Scenario di attacco avvenuto con successo** |  | |

|  |  |  |
| --- | --- | --- |
| **Titolo** | CifraturaComunicazioni | |
| **Descrizione** | Le comunicazioni verso l’esterno avvengono tramite un canale sicuro | |
| **Misuse case** | IntercettazioneComunicazione | |
| **Relazioni** |  | |
| **Precondizioni** | L’attaccante riesce ad intercettare i messaggi del sistema | |
| **Postcondizioni** | La comunicazione viene interrotta | |
| **Scenario principale** | *Sistema* | *Attaccante* |
| Effettua un tentativo di comunicazione verso l’esterno tramite un canale cifrato |  |
|  |  | Riesce ad intercettare/modificare il messaggio |
| **Scenario di attacco avvenuto con successo** | *Sistema* | *Attaccante* |
| Si accorge che il messaggio è stato modificato e interrompe la comunicazione |  |

#### Requisiti di Protezione dei Dati

Dall’analisi del rischio si ricavano i seguenti requisiti:

Funzionali:

* All’ avvio l’applicazione deve richiedere una password per l’accesso se questa è stata impostata, aprirsi direttamente altrimenti. Nel caso la password sia sbagliata l’applicazione deve chiedere il reinserimento. Nel caso la password venga dimenticata, essendo completamente offline e non salvata altrove non è previsto alcun meccanismo di recupero. Per evitare che l’utente inserisca una password troppo banale bisogna obbligarlo ad inserirne una di almeno 8 caratteri

Non funzionali:

* Non potendo garantire la sicurezza della macchina e dei file dell’utente sarà necessario cifrare tutte le informazioni importanti prima di salvarle su disco fisso.
* Per ogni chiamata REST API dove verranno trasmesse informazioni che si considerano confidenziali è necessario utilizzare un canale sicuro che garantisca riservatezza, integrità e autenticità.

## Analisi del problema

Dall’analisi dei requisiti si distinguono due principali entità tramite le quali i dati dovranno essere reperiti: Portafoglio (analizzatori di blockchain) ed Exchange (chiavi API). Il compito della gestione dei metadati delle valute e dei cambi da una valuta a un'altra va delegato ad un altro componente, il convertitore.

### Analisi documento dei requisiti

#### Tabella Funzionalità

|  |  |  |
| --- | --- | --- |
| **Funzionalità** | **Tipo** | **Grado Complessità** |
| Apertura App | Gestione dati | Semplice |
| Aggiunta Sito di exchange | Gestione dati | Semplice |
| Rimozione Sito di exchange | Gestione dati | Semplice |
| Aggiunta portafoglio | Gestione dati | Semplice |
| Rimozioni portafoglio | Gestione dati | Semplice |
| Aggiornamento fondi | Interazione verso l’esterno | Semplice |

##### Apertura App: Tabella informazioni/flusso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Informazione** | **Tipo** | **Livello protezione/privacy** | **Input/Output** | **Vincoli** |
| Password | Semplice | Protezione alta | Input | Almeno 8 caratteri |

##### Aggiunta Sito di Exchange: Tabella informazioni/flusso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Informazione** | **Tipo** | **Livello protezione/privacy** | **Input/Output** | **Vincoli** |
| Tipo exchange | Complessa | Protezione bassa | Input |  |
| Chiave pubblica API | Semplice | Protezione molto alta | Input | Dai 30 ai 64 caratteri |
| Chiave privata API | Semplice | Protezione molto alta | Input | Dai 30 ai 64 caratteri |

##### Rimozione Sito di Exchange: Tabella informazioni/flusso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Informazione** | **Tipo** | **Livello protezione/privacy** | **Input/Output** | **Vincoli** |
| Tipo exchange | Complessa | Protezione bassa | Input |  |

##### Aggiunta portafoglio: Tabella informazioni/flusso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Informazione** | **Tipo** | **Livello protezione/privacy** | **Input/Output** | **Vincoli** |
| Tipo blockchain | Complessa | Protezione bassa | Input |  |
| Indirizzo portafoglio | Semplice | Protezione alta | Input |  |

##### Rimozione portafoglio: Tabella informazioni/flusso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Informazione** | **Tipo** | **Livello protezione/privacy** | **Input/Output** | **Vincoli** |
| Tipo blockchain | Complessa | Protezione bassa | Input |  |

##### Aggiornamento fondi: Tabella informazioni/flusso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Informazione** | **Tipo** | **Livello protezione/privacy** | **Input/Output** | **Vincoli** |
| Chiavi API | Complessa | Protezione molto alta | Output |  |
| Indirizzo portafoglio | Semplice | Protezione alta | Output |  |
| Fondi | Complessa | Protezione alta | Output |  |

#### Tabella vincoli

|  |  |  |  |
| --- | --- | --- | --- |
| **Requisito** | **Categorie** | **Impatto** | **Funzionalità** |
| Cifratura dati salvati sulla macchina | Usabilità | Peggiorano l’usabilità | Apertura App, Aggiunta/Rimozione Exchange e portafogli |
| Cifratura comunicazioni verso l’esterno | Tempo di risposta | Peggiorano leggermente il tempo di risposta | Aggiornamento fondi |

#### Tabella Maschere

|  |  |  |
| --- | --- | --- |
| Maschera | Informazioni | Funzionalità |
| Interfaccia login | Richiesta password | Apertura App |
| Interfaccia Impostazioni | Modifica password, chiavi API e indirizzi portafoglio | Aggiunta/Rimozione Exchange e portafogli |
| Interfaccia fondi | Elenco dei fondi | Aggiornamento fondi |

#### Tabella sistemi esterni

|  |  |  |  |
| --- | --- | --- | --- |
| Sistema | Descrizione | Protocollo di Interazione | Livello di Sicurezza |
| Sito di exchange | Fornisce i dati relativi ai fondi posseduti sullo specifico exchange | Viene richiesta una coppia di chiavi API (pubblica/privata) ed attraverso delle chiamate http specifiche fornisce la lista dei fondi posseduti sull’Exchange | Alto livello di sicurezza perché una volta che si conoscono le chiavi API chiunque ha pieno accesso alle funzionalità associate |
| Esploratore di blockchain | Analizza una blockchain e fornisce i dati relativi a un indirizzo di portafoglio | A partire da un indirizzo di portafoglio esistente su una blockchain restituisce i fondi associati a tale portafoglio sulla specifica blockchain | Medio, unica informazione coinvolta è l’indirizzo di portafoglio che non fornisce alcuna autorità |
| Sito di cambi | Fornisce informazioni generali sui valori delle varie valute | Tramite chiamate API fornisce informazioni relative a una specifica valuta e dei relativi cambi in relazione alle altre | Basso, non gestisce alcuna informazione privata |

### Analisi ruoli e responsabilità

#### Tabella ruoli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ruolo | Responsabilità | Maschere | Riservatezza | Numerosità |
| Utente | Gestione di tutte le impostazioni (password, chiavi API, portafogli) | Interfaccia login, Interfaccia fondi, Interfaccia impostazioni | E’ richiesto un alto grado di riservatezza | Un solo utente previsto |
| Sito di exchange | Gestione dei fondi relativi a uno specifico exchange | Interfaccia fondi | E’ richiesto un alto grado di riservatezza | Un collegamento verso il sistema esterno per ogni tipo di exchange supportato dall’applicazione |
| Esploratore di blockchain | Gestione dei fondi relativi a un portafoglio in una specifica blockchain | Interfaccia fondi | E’ richiesto un alto grado di riservatezza | Un collegamento verso il sistema esterno per ogni tipo di blockchain supportato dall’applicazione |
| Sito di cambi | Gestione dei cambi tra le varie valute | Interfaccia fondi | E’ richiesto un basso grado di riservatezza | Un solo collegamento verso l’esterno che gestisce tutte le informazioni |

#### Utente: Tabella ruolo-informazioni

|  |  |
| --- | --- |
| **Informazione** | **Tipo di Accesso** |
| Password | Scrittura |
| Chiavi API | Scrittura |
| Indirizzo portafoglio | Lettura/Scrittura |

#### Sito di exchange: Tabella ruolo-informazioni

|  |  |
| --- | --- |
| **Informazione** | **Tipo di accesso** |
| Chiavi API | Lettura |

#### Esploratore di blockchain: Tabella ruolo-informazioni

|  |  |
| --- | --- |
| **Informazione** | **Tipo di accesso** |
| Indirizzo portafoglio | Lettura |

#### Sito di cambi: Tabella ruolo-informazioni

|  |  |
| --- | --- |
| **Informazione** | **Tipo di accesso** |
| Fondi utente | Lettura |

### Scomposizione del problema

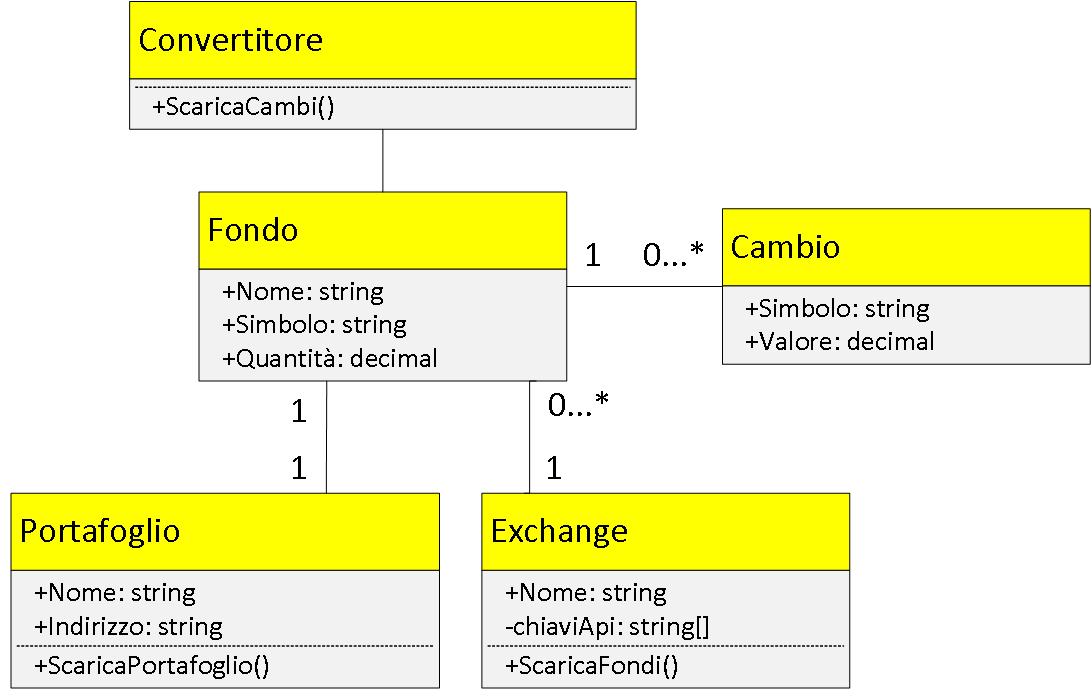
Non è stata prevista alcuna funzionalità complessa scomponibile in sotto-funzionalità

### Modello del dominio

Per organizzare l’architettura del sistema viene utilizzato il pattern BCE(Boundary-Control-Entity) suddividendolo in tre sottoinsiemi:

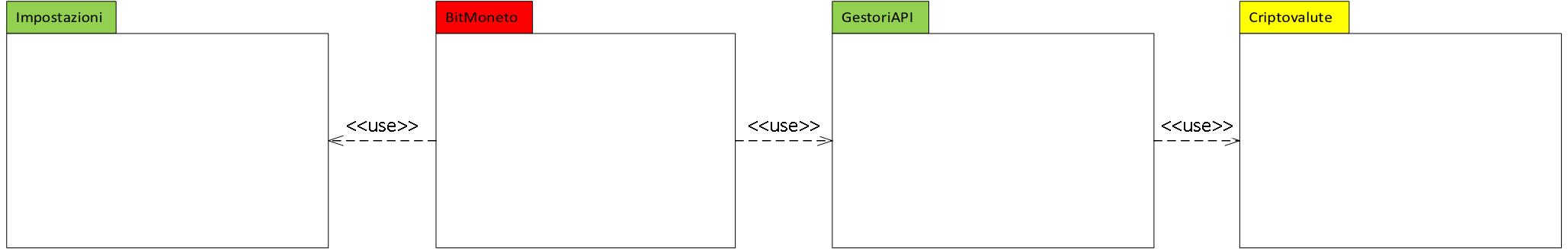
* Entity(informazione): Insieme di quelle entità che gestiscono le informazioni caratterizzanti del problema, principalmente composto dal modello del dominio.
* Boundary(presentazione): funzionalità che comunicano con il mondo esterno (utente o sistemi esterni)
* Control(applicazione): logica che gestisce i dati e agisce da tramite per le altre due parti

Per indicare la classe di appartenenza di ogni componente della struttura nei successivi diagrammi si utilizzerà come sfondo il colore associato sopra

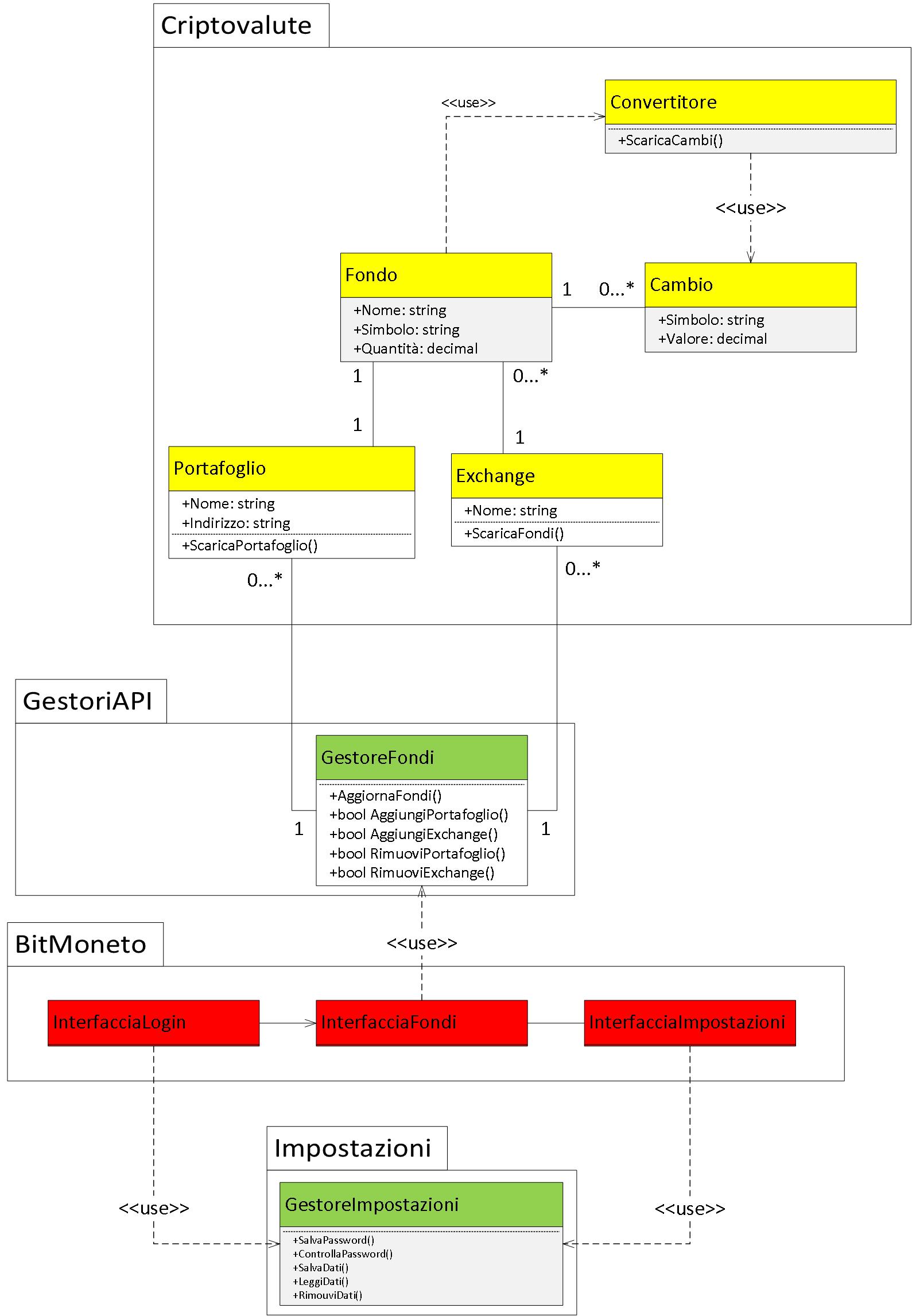


### Architettura logica: Struttura

#### Diagramma dei package

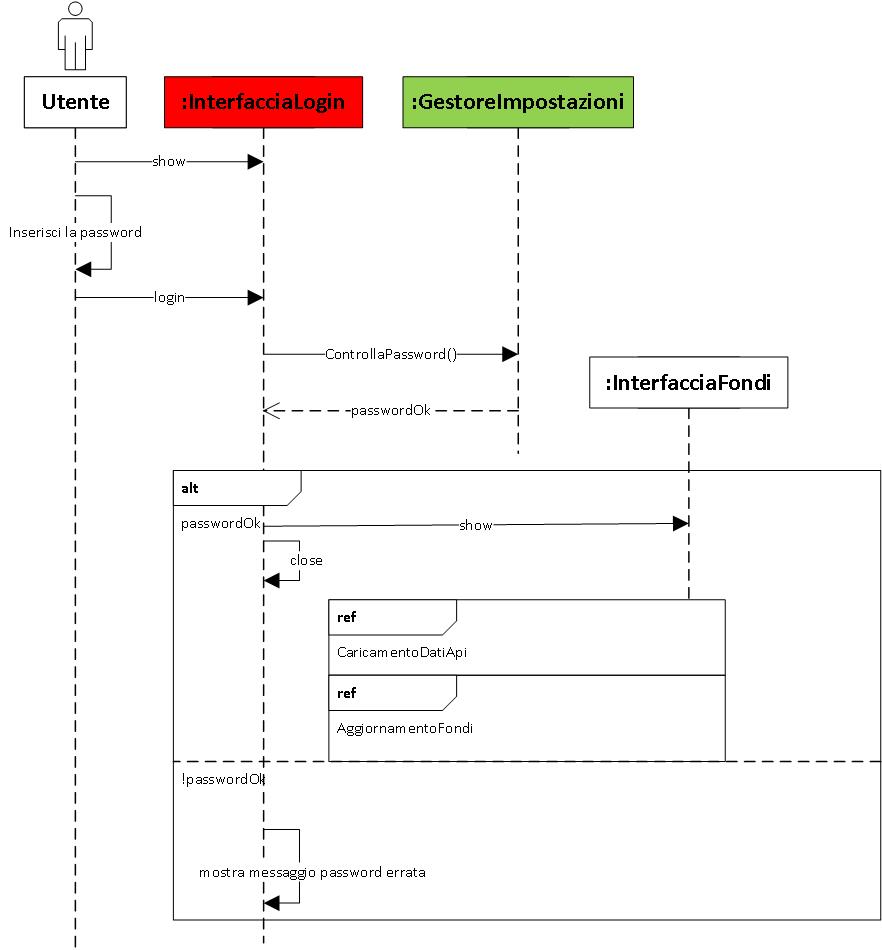


#### Diagramma delle classi

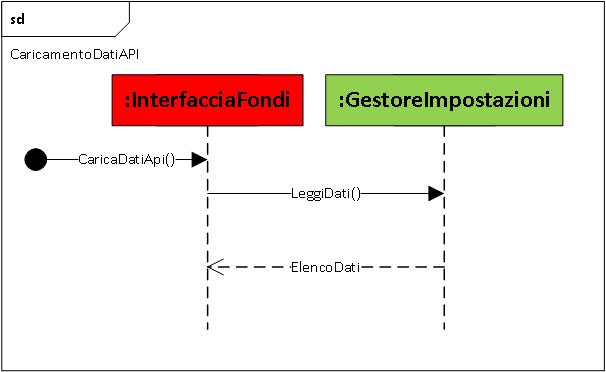


### Architettura logica: Interazione

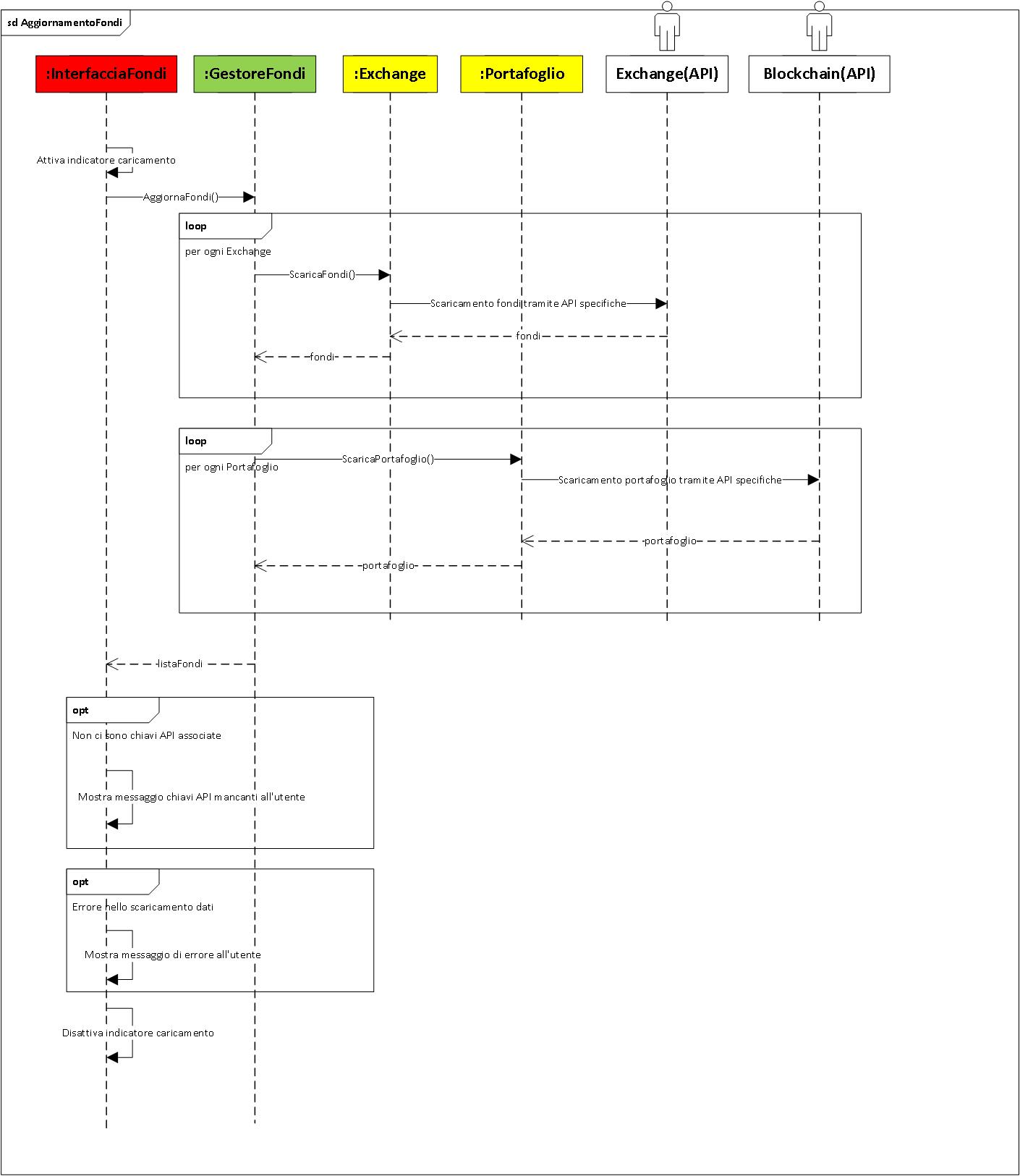
#### Diagramma di sequenza: Login



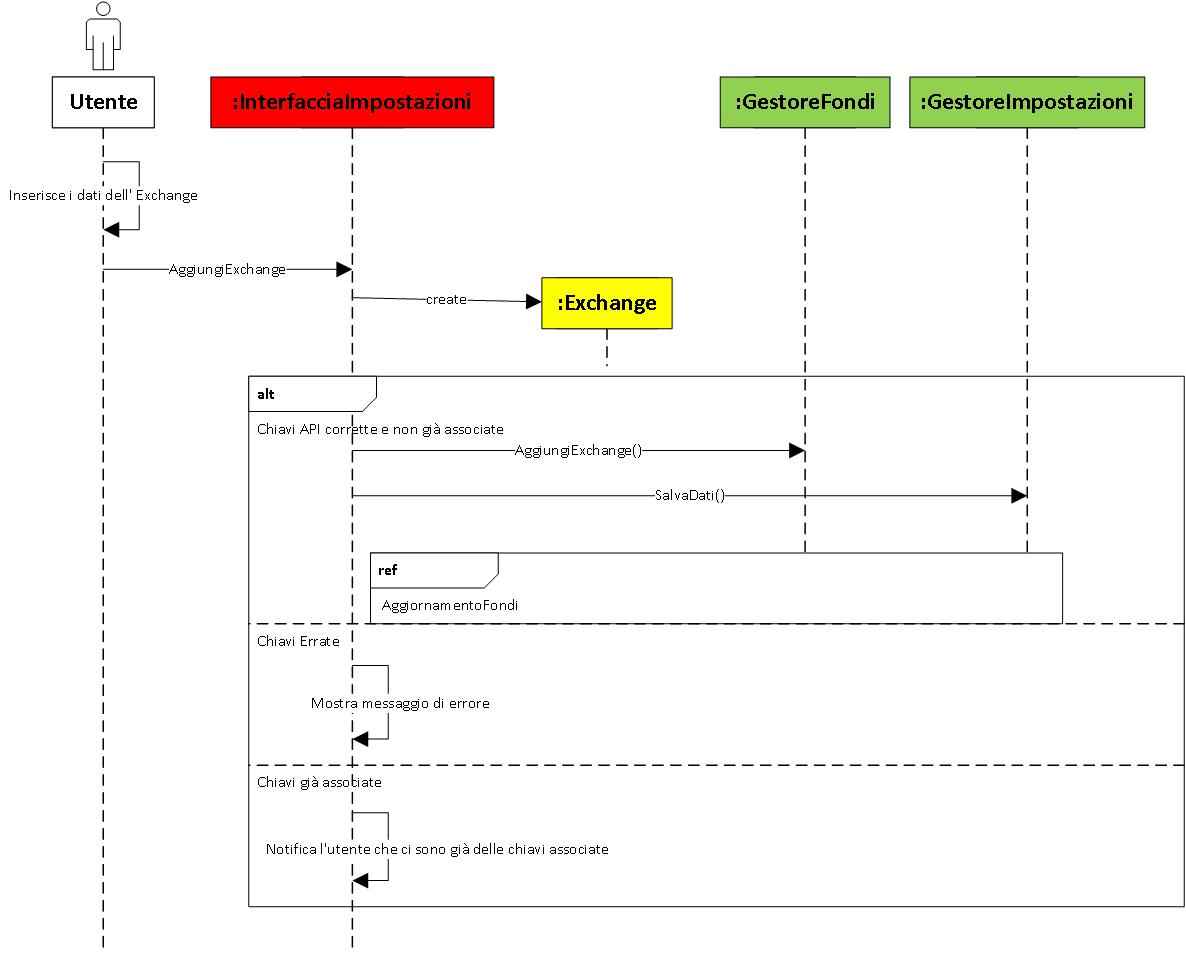
#### Diagramma di sequenza: Carica dati API



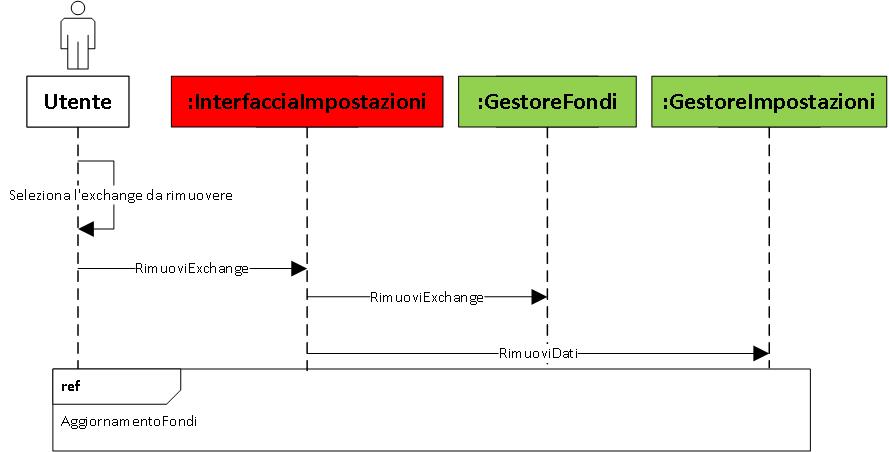
#### Diagramma di sequenza: Aggiornamento fondi



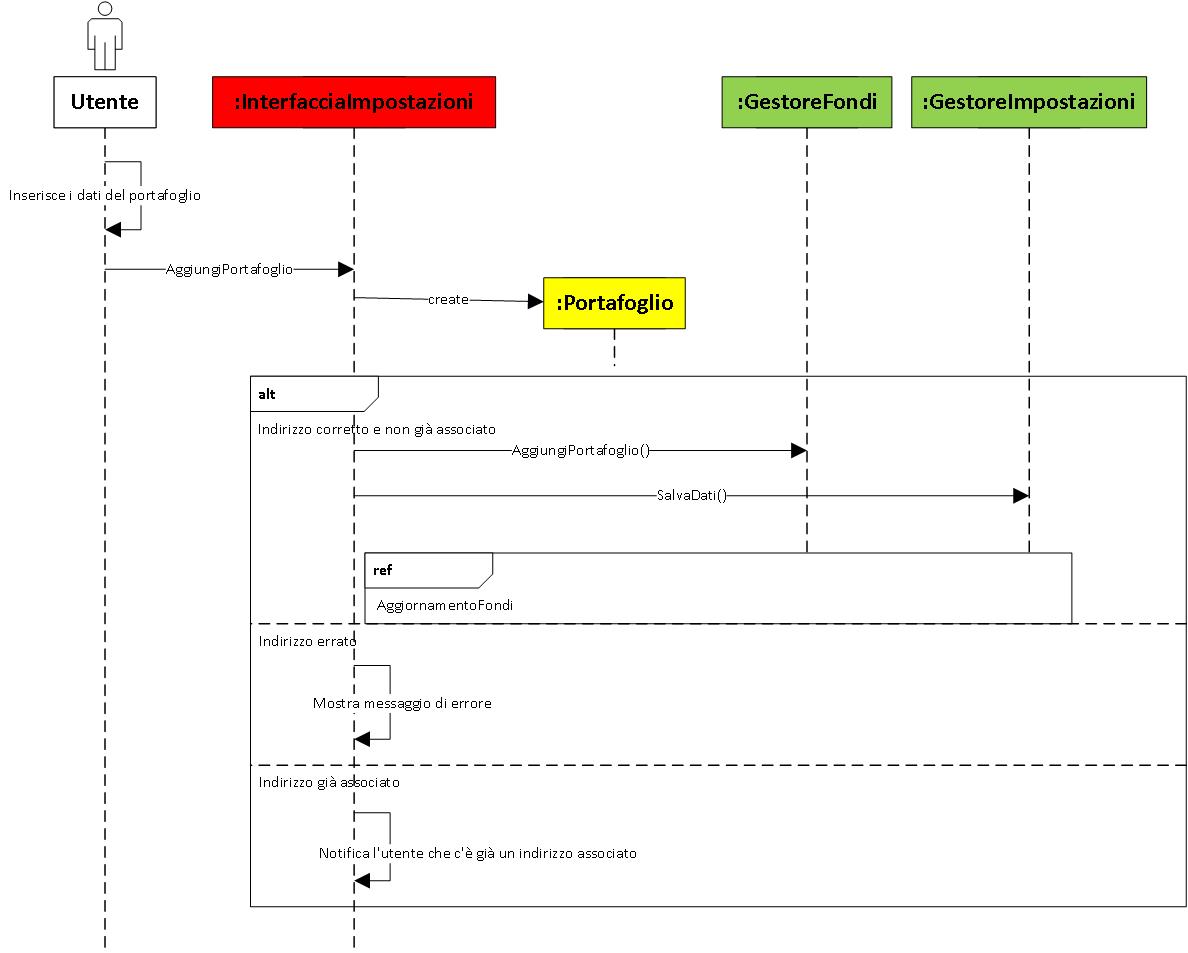
#### Diagramma di sequenza: Aggiunta di un exchange



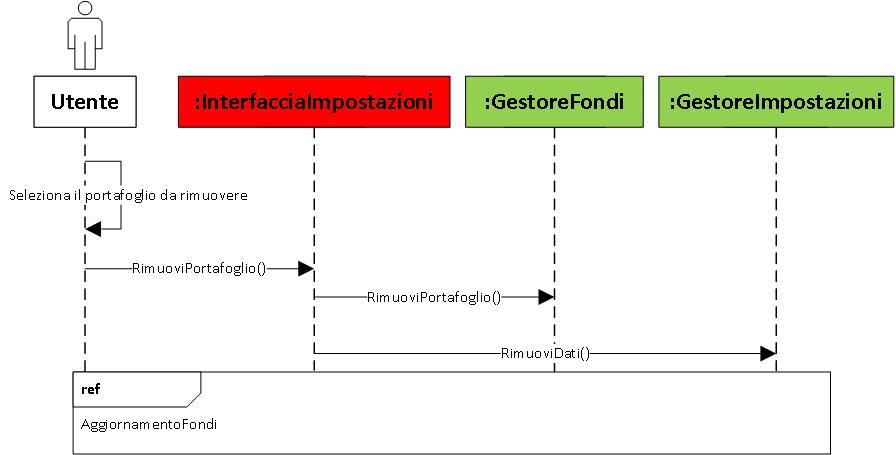
#### Diagramma di sequenza: Rimozione di un exchange



#### Diagramma di sequenza: Aggiunta di un portafoglio

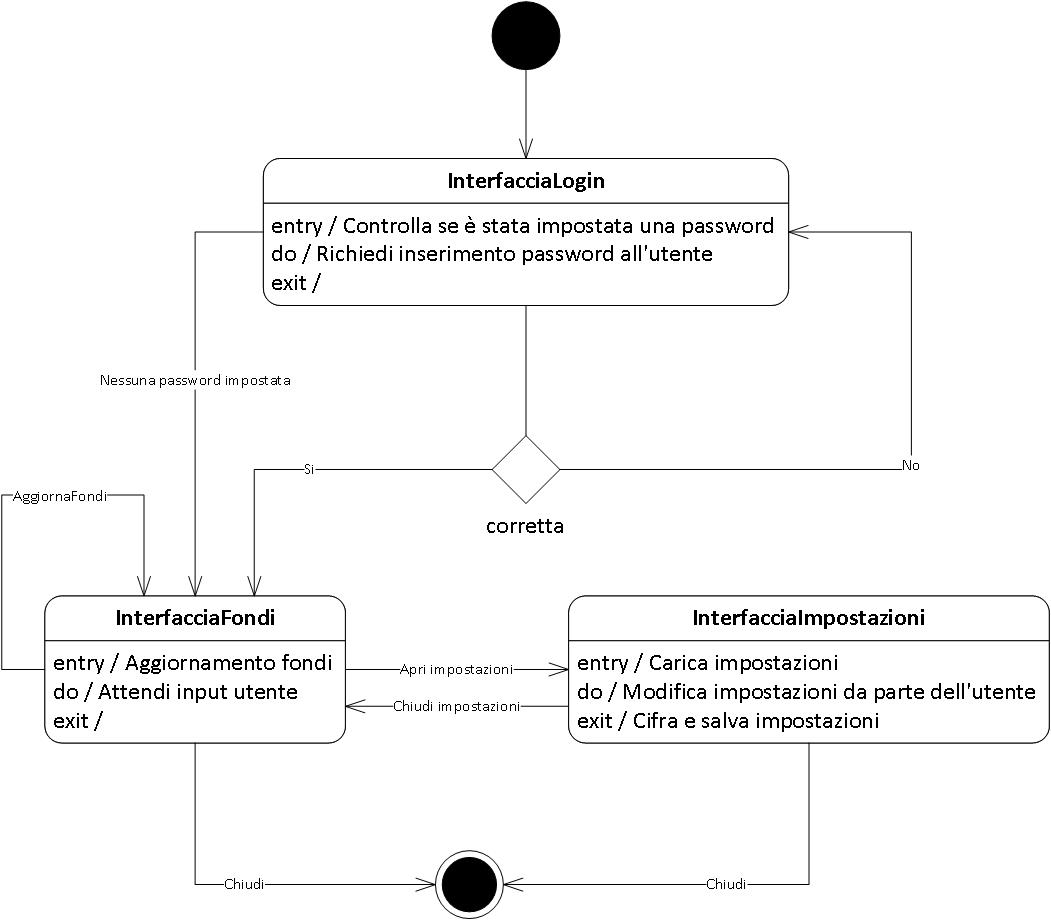


#### Diagramma di sequenza: Rimozione di un portafoglio



### Architettura logica: Comportamento

Il diagramma di stato mostra come l’applicazione si comporta per chiarire come vengono gestite le varie interfacce in base alle interazioni dell’utente.



Per quanto riguarda i singoli componenti non è stato prodotto alcun diagramma di comportamento dato che non prevedono uno stato interno particolarmente complicato.

### Piano di lavoro

In base alle dipendenze dettate dai componenti dell’applicazione si è deciso di procedere con il seguente ordine per la progettazione e il successivo sviluppo:

1. Criptovalute
2. GestoriAPI
3. Impostazioni
4. BitMoneto

Con i seguenti tempi previsti:

* Progettazione entro una settimana
* Sviluppo dei componenti (con analogo test unitario) entro 1 settimana dalla progettazione
* Integrazione e test dell’intero sistema in una settimana rispetto la fine dello sviluppo

#### Sviluppi futuri

Il primo prototipo di release includerà il seguente insieme di exchange supportati:

* Binance
* Bitfinex

E potranno essere aggiunti i seguenti tipi di portafogli:

* Bitcoin (tramite le API di blockexplorer.com)
* Ethereum (tramite le API di etherscan.com)

In futuro però queste liste sono destinate ad arricchirsi e saranno in continuo cambiamento.

Per quanto riguarda il convertitore al momento, come già detto è stato scelto CryptoCompare, ad ogni modo in futuro questo potrebbe essere sostituito o affiancato da altri convertitori per dare più scelte all’utente.

Al momento per ogni blockchain sarà possibile monitorare solamente un indirizzo, ma nelle release successive dovrà essere prevista la possibilità di aggiungere più indirizzi appartenenti ad una blockchain. Mentre nel caso degli exchange la coppia di chiavi api associabili sarà sempre una (una persona non può possedere più di un account su un exchange).

In fase di progettazione si richiede di tenere conto di tutti questi sviluppi futuri.

### Piano del collaudo

Tutto l’interfacciamento dell’applicazione che effettua chiamate API esterne andrà sottoposta ad un accurata fase di test. Ad ogni modo un test accurato sarà reso più complicato dal fatto che i dati che verranno ricevuti dall’esterno non saranno stabili (valori delle valute variabili) e quindi risulterà complicato scrivere dei test precisi ed affidabili.

Nel caso degli exchange verranno utilizzati degli account privati dove non vengono effettuati trasferimenti in modo da poter confrontare i fondi che si hanno effettivamente sugli exchange con quelli che il sistema riesce a scaricare.

Nel caso degli indirizzi di portafogli alla stessa maniera si farà riferimento a portafogli stabili che non hanno movimenti.

#### PortafoglioTests

         [TestMethod]

        [ExpectedException(typeof(ArgumentException))]

        public void Costruttore\_SenzaIndirizzo\_Eccezione()

        {

            Portafoglio explorer = new Portafoglio(null);

        }

        [TestMethod]

        [ExpectedException(typeof(ArgumentException))]

        public void Costruttore\_IndirizzoVuoto\_Eccezione()

        {

            Portafoglio explorer = new Portafoglio("");

        }

         [TestMethod]

        [ExpectedException(typeof(Exception))]

        public void ScaricaPortafoglio\_IndirizzoErrato\_Eccezione()

        {

            Portafoglio explorer = new Portafoglio ("blabla");

            explorer.ScaricaPortafoglio();

        }

         [TestMethod]

        public void ScaricaPortafoglio\_IndirizzoStabile\_Valore()

        {

            String indirizzoStabile = "1JaPNwMXt2AuVkWmkUHbsw78MbGorTfmm2";

           decimal valore = (decimal)2194.51101);

            Portafoglio atteso = new Portafoglio(indirizzoStabile, fondo);

            Portafoglio explorer = new Portafoglio(indirizzoStabile);

            explorer.ScaricaPortafoglio();

            Assert.AreEqual<string>(explorer.Fondo.Nome, "Bitcoin");

            Assert.AreEqual<decimal>(explorer.Fondo.Quantità, valore);

        }

#### GestioneFondi

Per il test del gestore dei fondi, per evitare di inserire una dipendenza verso gli exchange e i portafogli verranno scritte due classi di test apposite:

##### Classe di test: TestExchange

class TestExchange

    {

        public TestExchange()

        {

            Nome = "TestExchange";

            ChiavePrivata = "APIPRIVATA";

            ChiavePubblica = "APIPUBBLICA";

        }

        public string Nome { get; }

        public string ChiavePubblica { get; }

        public string ChiavePrivata { get; }

        private List<Fondo> \_fondi = null;

        public List<Fondo> Fondi { get { return \_fondi; } }

        public void ScaricaFondi()

        {

            \_fondi = new List<Fondo>();

            Fondo tmp = new Fondo(new Valuta("Bitcoin","BTC"), 1);

            \_fondi.Add(tmp);

            tmp = new Fondo(new Valuta("Ethereum", "ETH"), 10);

            \_fondi.Add(tmp);

        }

    }

##### Classe di test: TestPortafoglio

class TestPortafoglio

    {

        public TestPortafoglio()

        {

            Nome = "TestPortafoglio";

            Indirizzo = "TestIndirizzo";

        }

        public string Nome { get; }

        public string Indirizzo { get; }

        private Fondo \_fondo = null;

        public Fondo fondo { get { return \_fondo; } }

        public void ScaricaPortafoglio()

        {

            \_fondo = new Fondo(new Valuta("Bitcoin", "BTC"), 1);

        }

    }

##### GestioneFondiTests

[TestClass]

    public class GestoreFondiTests

    {

        GestoreFondi gestoreFondi;

private TestExchange exchange;

private TestPortafoglio portafoglio;

        public GestoreFondiTests()

        {

            gestoreFondi = new GestoreFondi();

exchange = new TestExchange();

            portafoglio = new TestPortafoglio();

        }

        [TestMethod]

        public void AggiungiExchange\_TestExchange\_True()

        {

            Assert.IsTrue(gestoreFondi.AggiungiExchange(exchange));

        }

        [TestMethod]

        public void AggiungiPortafoglio\_TestPortafoglio\_True()

        {

            Assert.IsTrue(gestoreFondi.AggiungiPortafoglio(portafoglio));

        }

        [TestMethod]

        public void AggiornaFondi\_Test\_2()

        {

            Assert.IsTrue(gestoreFondi.AggiungiExchange(exchange));

            Assert.IsTrue(gestoreFondi.AggiungiPortafoglio(portafoglio));

            gestoreFondi.AggiornaFondi();

            Assert.AreEqual<int>(gestoreFondi.Portafogli.Count, 1);

            Assert.AreEqual<string>(gestoreFondi.Portafogli.ElementAt(0).Nome, “TestPortafoglio”);

            Assert.AreEqual<decimal>(gestoreFondi.Portafogli.ElementAt(0).Fondo.Quantità, 1);

            Assert.AreEqual<int>(gestoreFondi.Exchange.Count, 1);

            Assert.AreEqual<string>(gestoreFondi.Exchange.ElementAt(0).Nome, “TestExchange”);

            Assert.AreEqual<int>(gestoreFondi.Exchange.ElementAt(0).Fondi.Count,2);

            Assert.AreEqual<decimal>(gestoreFondi.Exchange.ElementAt(0).Fondi.ElementAt(0).Quantità,1);

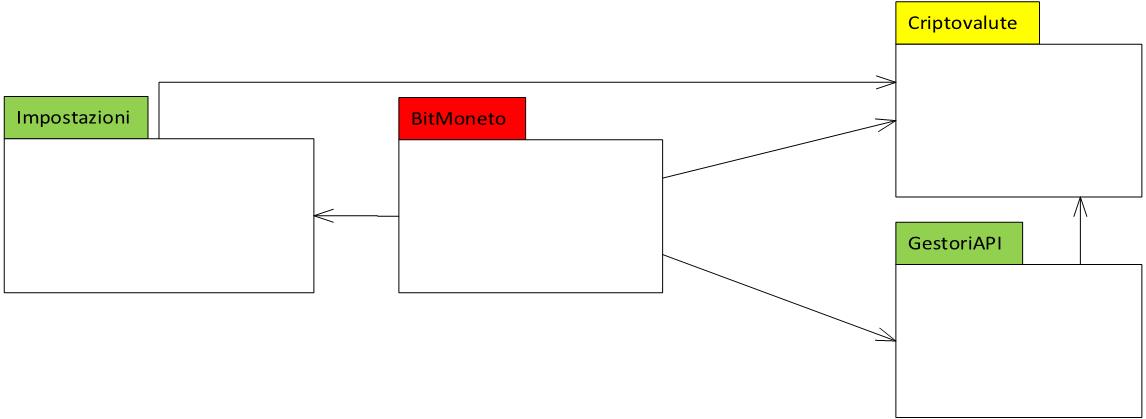
            Assert.AreEqual<decimal>(gestoreFondi.Exchange.ElementAt(0).Fondi.ElementAt(1).Quantità,10);

        }

    }

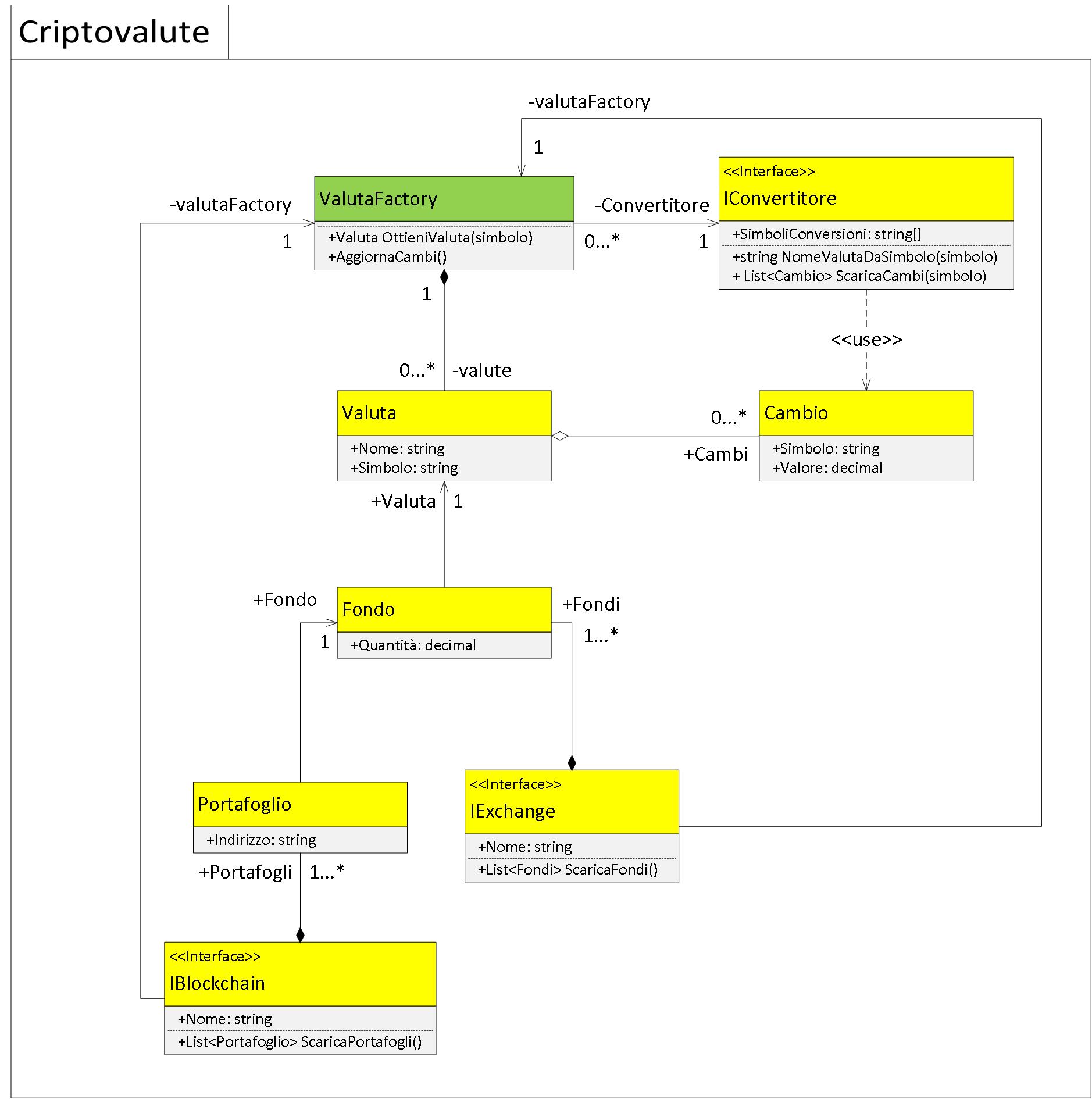
## Progetto

### Architettura del sistema: Struttura



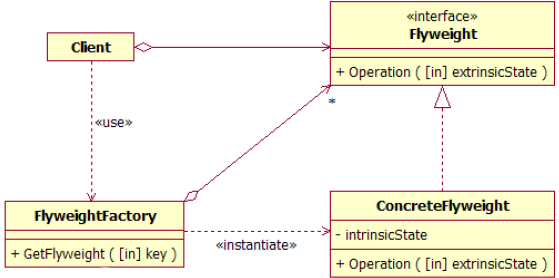
#### Diagramma di dettaglio: Criptovalute

Rispetto all’analisi del problema l’entità portafoglio è stata suddivisa in Blockchain (recupera i dati tramite le chiamate API) e Portafoglio (salva i dati del portafoglio) dato che in futuro potrebbero esserci più portafogli monitorati appartenenti alla stessa blockchain.



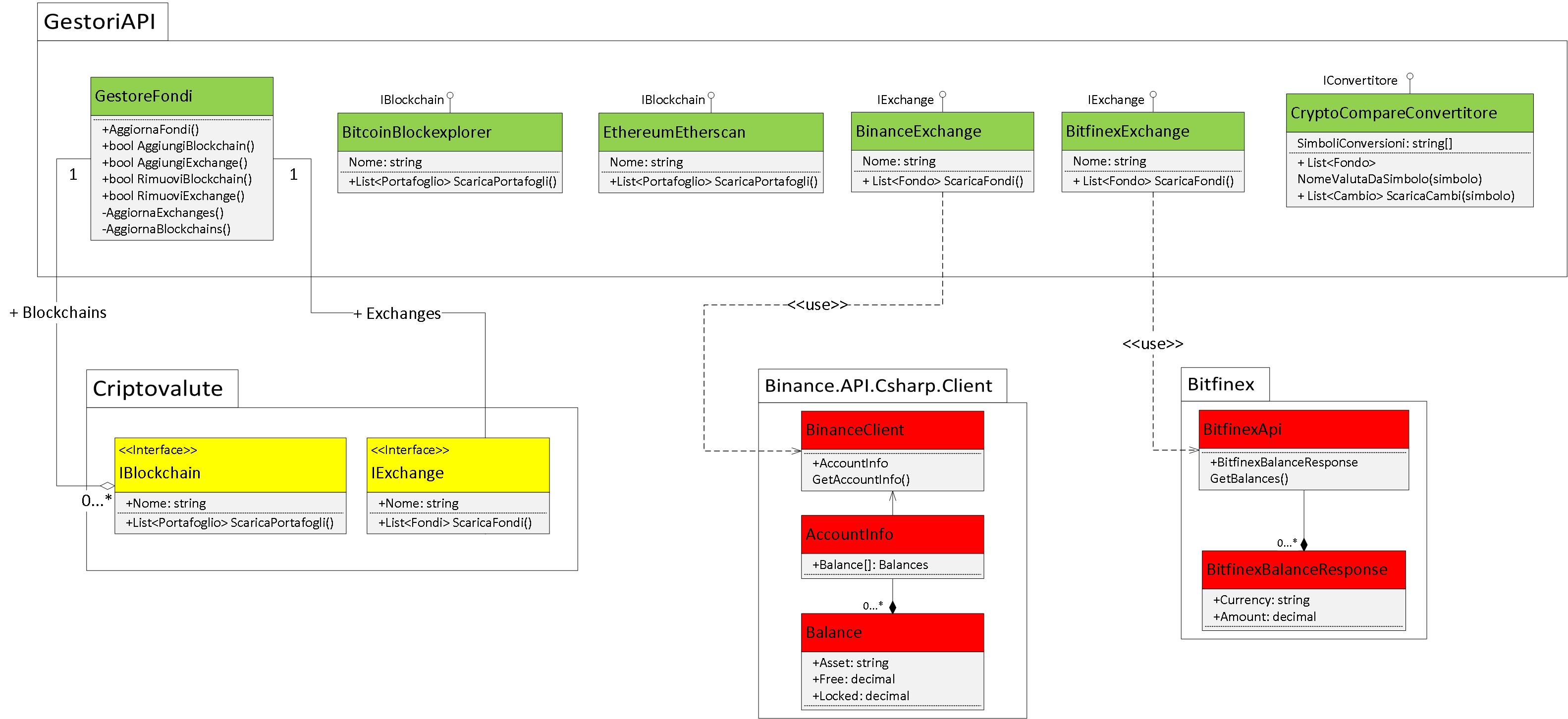
Avendo deciso di separare l’implementazione di tutti i componenti che fanno uso di chiamate API (Dependency Inversion Principle), dato che potrebbero essere modificati nel tempo, le entità Blockchain, Exchange e Convertitore sono delle interfacce in modo che la loro implementazione sia rimandata al pacchetto GestoriAPI.

Per la gestione delle valute e dei relativi cambi è stata utilizzata una soluzione basata sul pattern “Flyweight”:

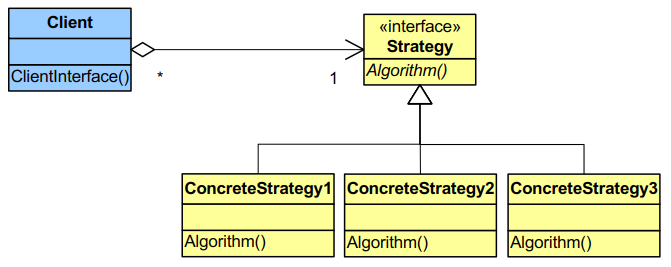


In questo modo ogni volta che utilizzo una determinata valuta, anziché crearla da capo (riscaricando anche tutti i cambi e le informazioni associate) ne utilizzo una versione che è già stata scaricata e viene condivisa da tutta l’applicazione tramite la factory, evitando inutili chiamate API verso il convertitore e tassi di cambio incongruenti. Inoltre nella maggior parte dei casi, le informazioni restituite dalle API dei siti di exchange identificano il tipo di valuta solamente tramite il simbolo, sarà quindi necessario reperire il nome completo della valuta tramite il convertitore, a questo scopo è stato inserito il metodo NomeValutaDaSimbolo.

#### Diagramma di dettaglio: GestoriAPI



In questo package ci sono le sole implementazioni di tutti i componenti che effettuano chiamate API verso l’esterno. In GestoreFondi vengono gestite tutte le istanze dei vari exchange e blockchain del programma, ad ogni modo utilizzando una soluzione basata sul pattern STRATEGY:

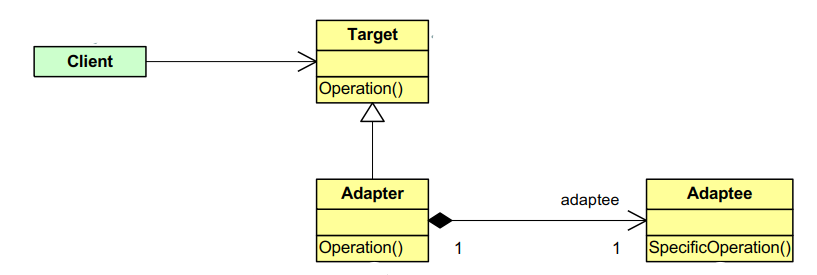


Le implementazioni di ogni piattaforma rimangono separate dall’interfaccia di base in modo da essere gestite dinamicamente a runtime.

Nei casi degli exchange (Binance e Bitfinex) si è deciso di utilizzare librerie già esistenti:

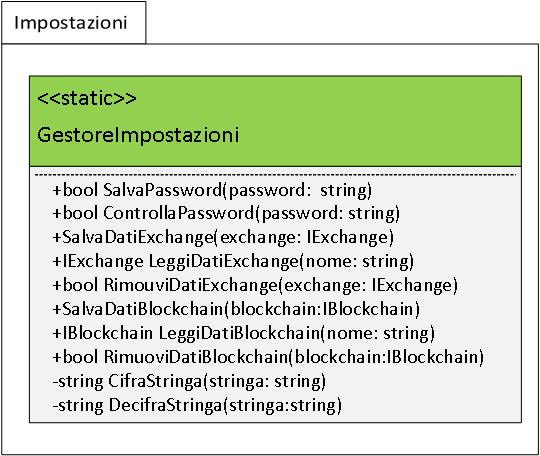
* Binance.API.Csharp.Client (<https://github.com/morpheums/Binance.API.Csharp.Client>)
* BitfinexAPI (<https://github.com/twobeeb/BitfinexAPI>)

Entrambe le librerie effettuano le chiamate API tramite una connessione TLS e sono quindi da considerare affidabili per il nostro livello di sicurezza. Le due classi BinanceExchange e BitfinexExchange diventano quindi delle semplici implementazioni del pattern “Adapter”:



Che utilizzano le librerie per reperire i dati e poi non fanno altro che “adattarli” all’interfaccia IExchange.

#### Diagramma di dettaglio: Impostazioni

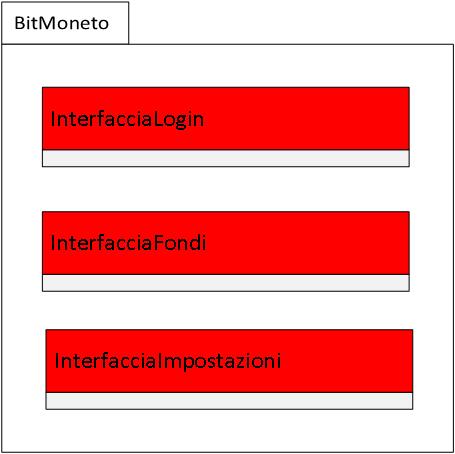


La classe del GestoreImpostazioni è stata resa statica in quanto è prevista una sola configurazione per l’applicazione condivisa da tutti i suoi componenti (senza stato interno).

Inoltre i metodi per il salvataggio/lettura/rimozione di exchange e blockchain sono stati separati per fornire implementazioni più specifiche.

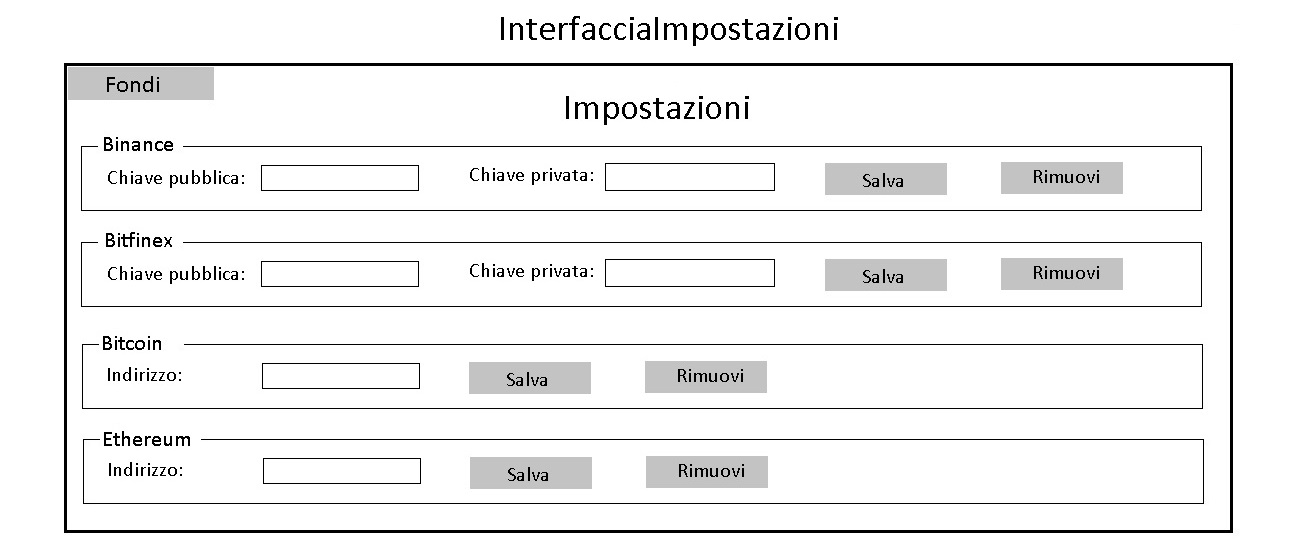
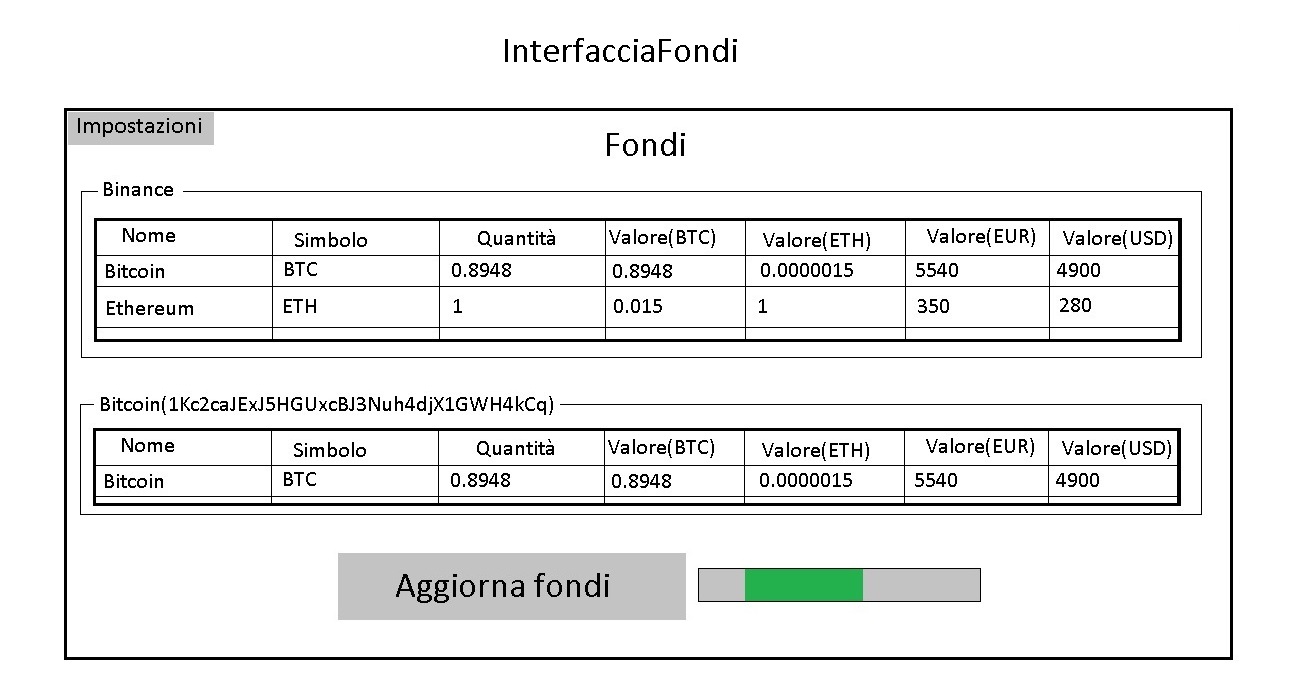
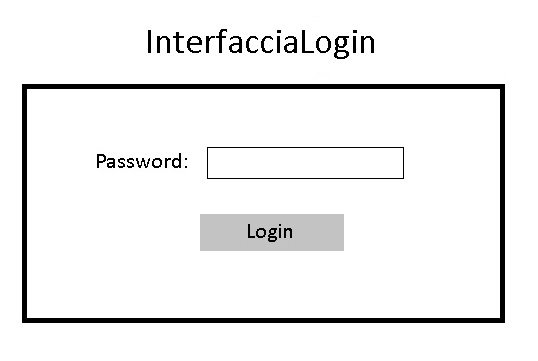
I due metodi privati CifraStringa e DecifraStringa sono i metodi che verranno utilizzati per ogni scrittura o lettura sul file di configurazione.

#### Diagramma di dettaglio: BitMoneto



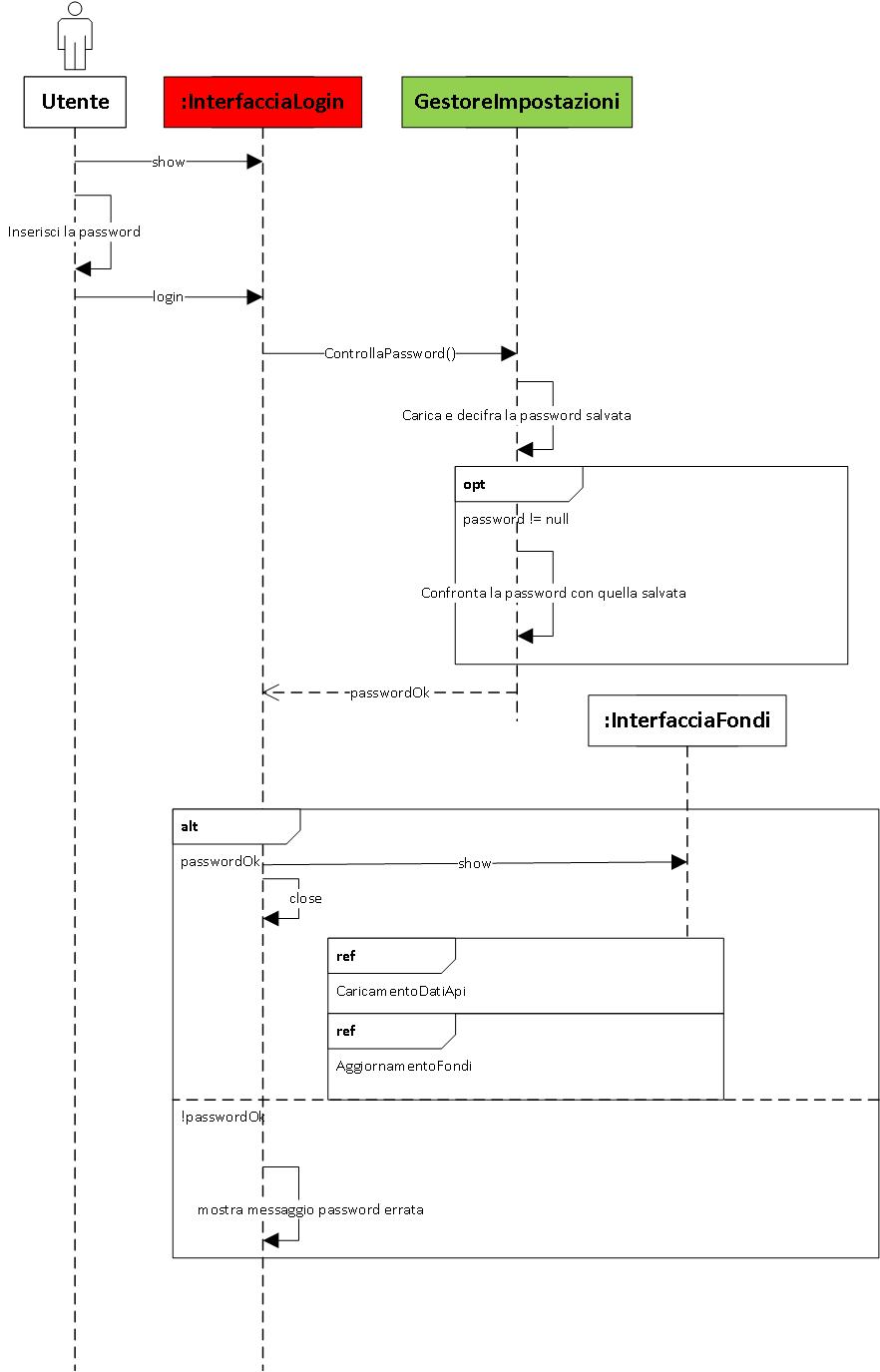
Questo pacchetto contiene le sole interfacce grafiche dell’applicazione, appositamente mantenute separate dal resto dell’applicazione per renderla il più versatile possibile nel caso in futuro si decida di rendere l’applicazione “cross-platform” e trasportarla su altri dispositivi oltre a quello desktop.

#### Interfacce grafiche

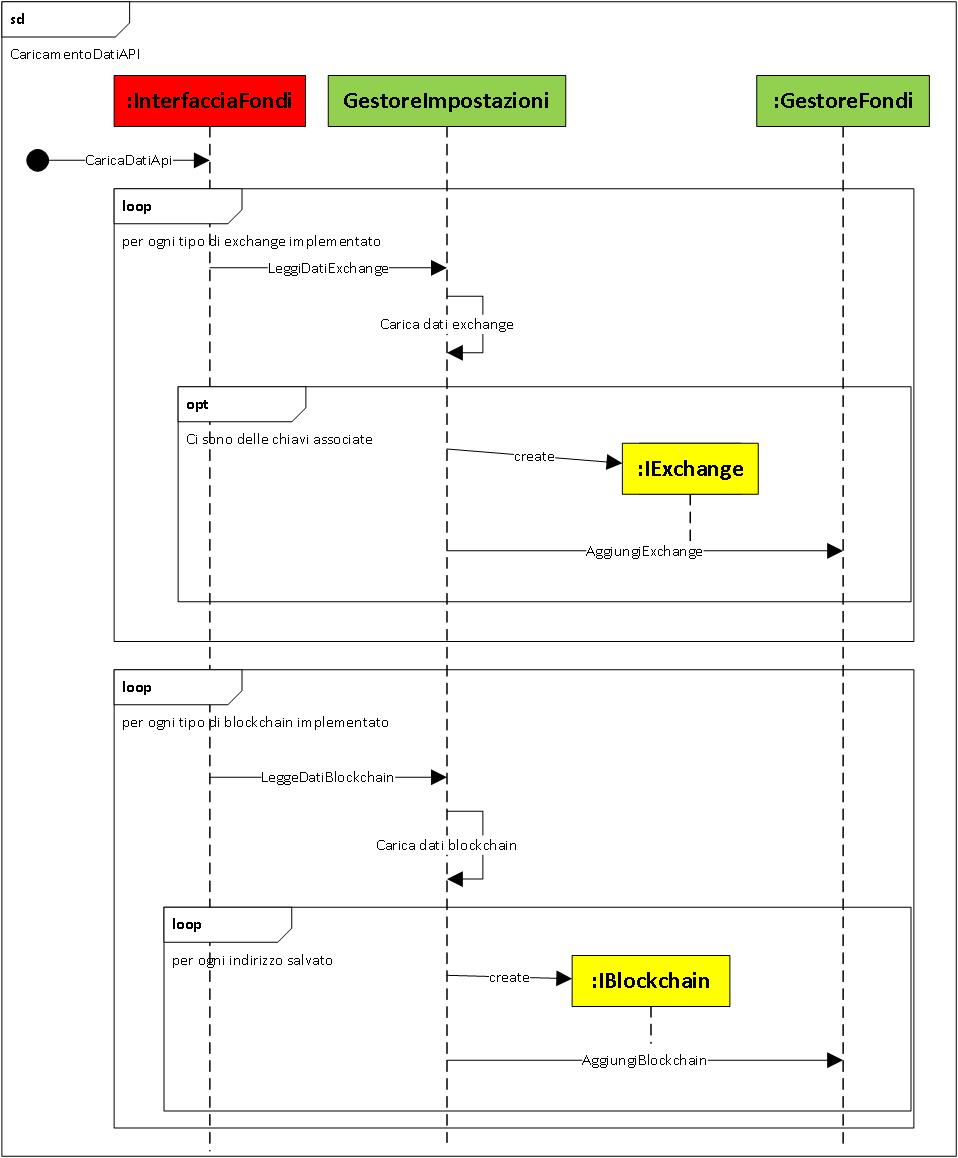


### Architettura del sistema: Interazione

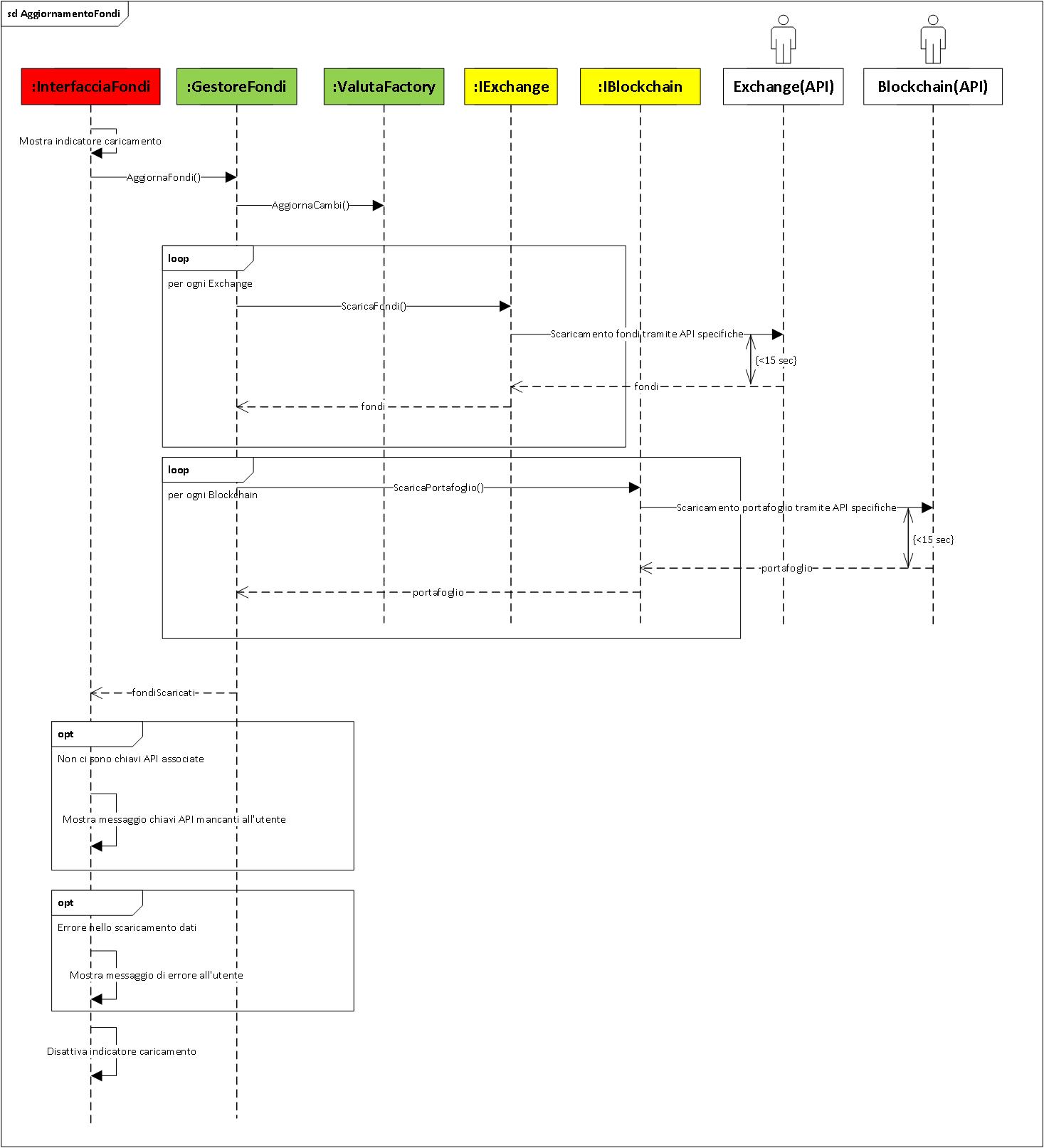
#### Diagramma di sequenza: Login



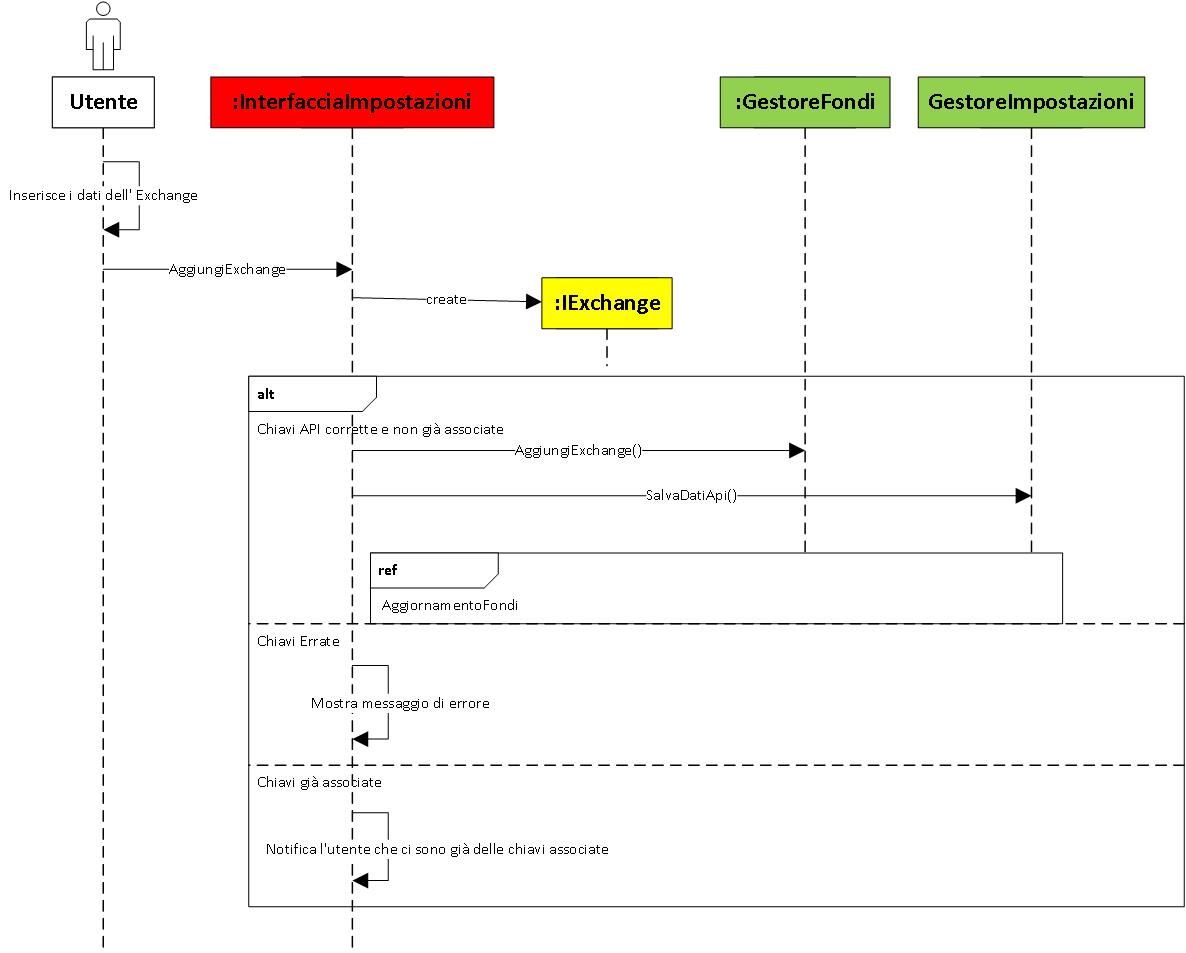
#### Diagramma di sequenza: Carica dati API



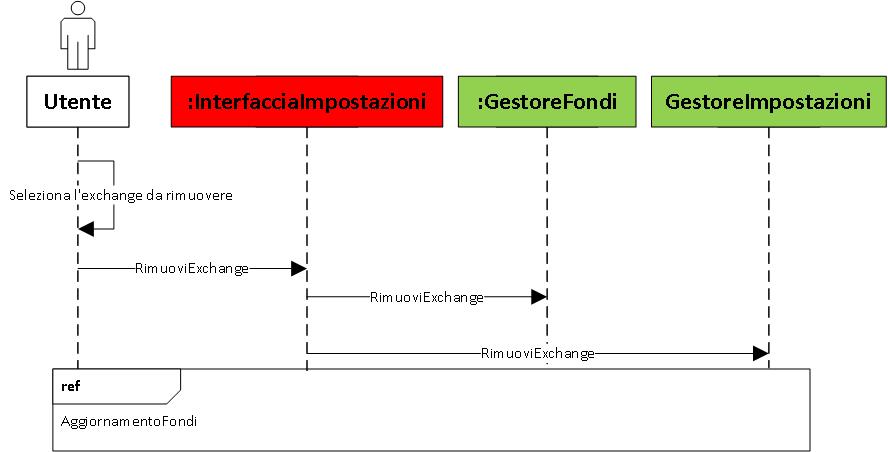
#### Diagramma di sequenza: Aggiornamento fondi



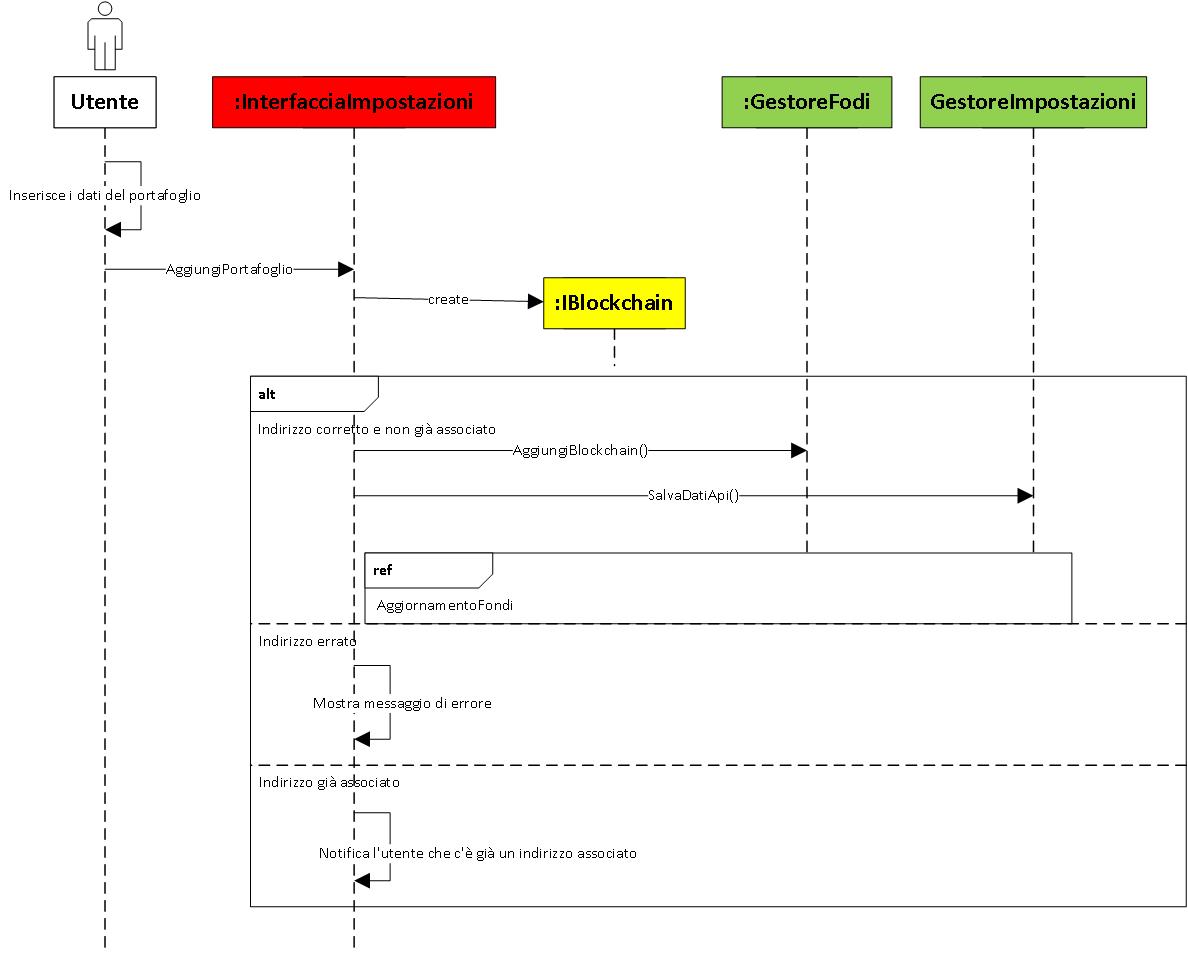
#### Diagramma di sequenza: Aggiunta di un exchange



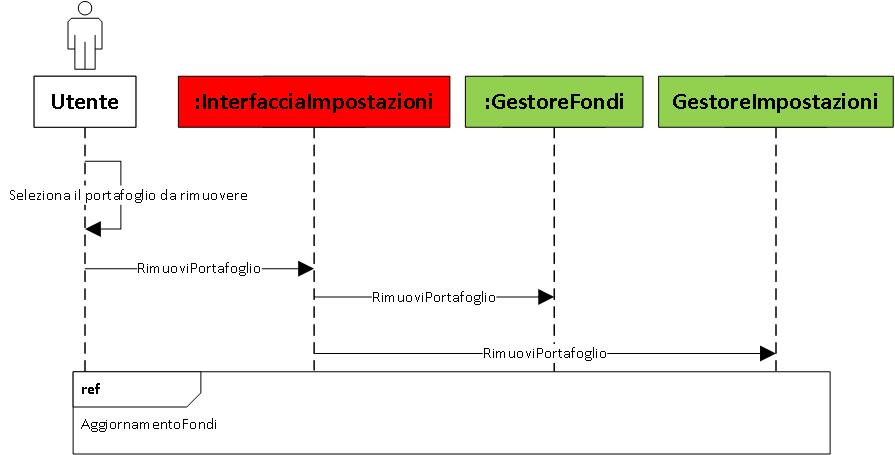
#### Diagramma di sequenza: Rimozione di un exchange



#### Diagramma di sequenza: Aggiunta di un portafoglio

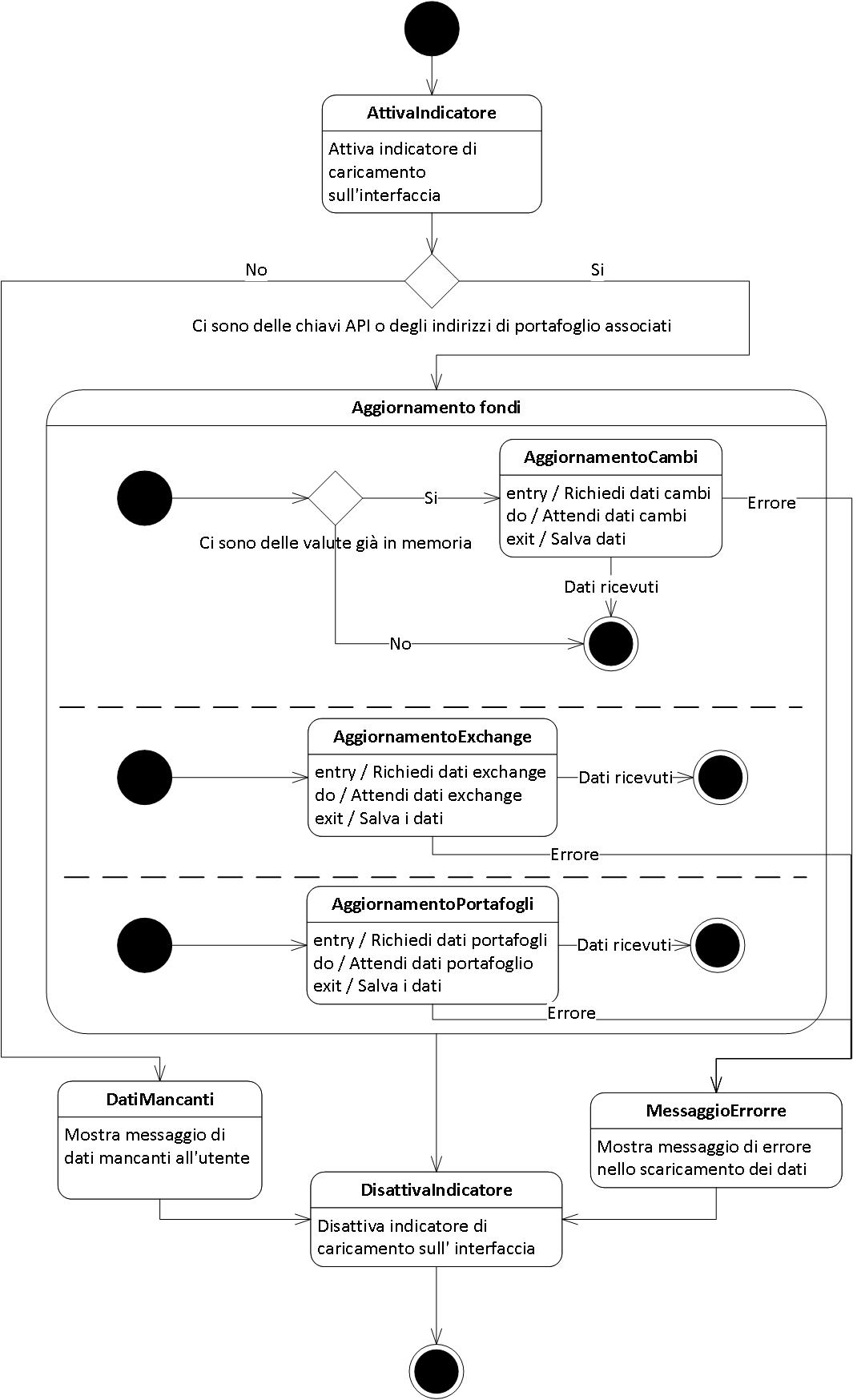


#### Diagramma di sequenza: Rimozione di un portafoglio



### Architettura del sistema: Comportamento

#### Aggiornamento Fondi



Nel diagramma di stato viene descritto il processo di aggiornamento dei fondi. L’aggiornamento dei cambi si riferisce a tutte quelle valute che sono state scaricate in un precedente aggiornamento dei fondi e sono attualmente salvate in ValutaFactory. Nel caso non siano stati effettuati aggiornamenti dei fondi precedentemente questa fase verrà semplicemente saltata non essendoci cambi da aggiornare. Come anticipato tutte le operazioni saranno asincrone quindi verranno avviate in parallelo in modo da ottimizzare i tempi di attesa.

## Implementazione

#### Tecnologie utilizzate

Le librerie dell’applicazione sono state realizzate in codice C# e per il design dell’interfaccia grafica è stata utilizzata la libreria WPF (Windows Presentation Foundation).

#### Dizionari

Durante l’implementazione, grazie all’utilizzo dei dizionari è stato possibile eliminare l’entità cambio inserendo i valori direttamente in un dizionario (chiavi: simbolo valute(string), valori: cambio(decimal)) all’interno di Valuta.

Anche le valute già scaricate in ValutaFactory sono salvate in un dizionario (chiave: simbolo valuta(string), valore: valuta(Valuta)) in questo modo è facile e veloce accedere alla lista delle valute per controllare quando la valuta cercata è già presente nell’ elenco e invece quando bisogna aggiungerla.

#### Chiamate asincrone

Per migliorare l’esperienza utente tutte le operazioni che richiedevano un collegamento verso l’esterno tramite chiamate API sono state implementate asincrone tramite l’utilizzo dei costrutti Task<>, async e await di c#. In questo modo è possibile avviare l’aggiornamento dei vari dati da tutte le sorgenti e poi restituire il controllo dell’applicazione all’utente mentre i dati vengono scaricati e poi mostrarli non appena questi sono disponibili. Velocizzando il tempo di attesa totale grazie al parallelismo delle chiamate.

#### Impostazioni

Le impostazioni vengono cifrate con l’algoritmo aes(Advanced Encryption Standard), convertite in Base64 e salvate in un file di configurazione con stesso nome e cartella del file eseguibile con estensione .config.

## Collaudo

Il sistema è stato collaudato seguendo la seguente strategia:

1. Durante lo sviluppo di tutte le funzioni interne di gestione del dominio e durante lo sviluppo delle parti di interfacciamento verso l’esterno tramite chiamate API tramite l’utilizzo di test di unità che venivano eseguiti di pari passo con le classi che andavano ad testare per verificare il comportamento atteso delle singole parti secondo le indicazioni specificate nel piano del collaudo (utilizzando le classi di test TestExchange TestPortafoglio e TestConvertitore).

A tal fine è stato creato un package Bitmoneto.UnitTests contenente tutti i test, tra cui i più significativi:

* 1. Creazione exchange con chiavi API non corrette o mancanti
  2. Scaricamento fondi da exchange con chiavi API corrette (e fondi stabili)
  3. Creazione portafoglio con indirizzo mancante
  4. Scaricamento fondi da portafoglio con indirizzo corretto (e fondi stabili)
  5. Scaricamento tassi di conversione dal convertitore per varie valute
  6. Conversione da simbolo a nome dal convertitore per varie valute

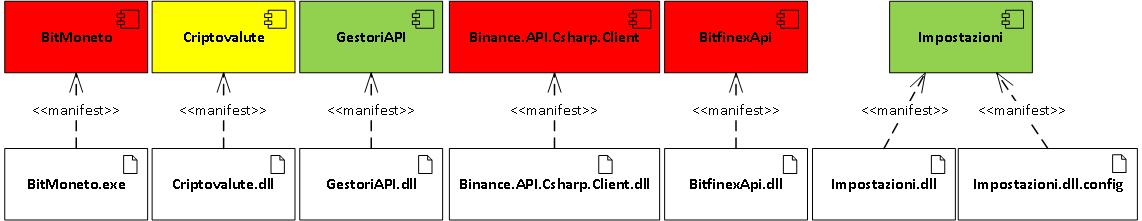
1. Successivamente, tramite test di integrazione dove le classi di test sono state sostituite con le reali implementazioni per assicurarsi che il comportamento del sistema nel complesso fosse quello atteso.

A tal fine è stato creato un package Bitmoneto.IntegrationTests dove viene istanziato un GestoreFondi al quale vengono associate varie implementazioni utilizzando chiavi API e indirizzi validi per gestire contemporaneamente fondi provenienti da sorgenti differenti (utilizzando sempre chiavi API e indirizzi conosciuti)

Come anticipato per il testing del sistema è necessario avere a disposizione degli account sui vari siti di exchange in modo da effettuare uno scaricamento fondi per poi confrontare i fondi scaricati con quelli attesi. Lo stesso problema si pone con gli analizzatori di blockchain. A questo scopo per ognuna delle soluzioni implementate è stato previsto un metodo per il relativo test (chiavi API per exchange o indirizzo di portafoglio per blockchain).

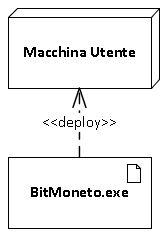
## Deployment

Diagramma dei componenti prodotti:



L’applicazione prodotta è composta dai file che realizzano le librerie di codice (.dll), l’eseguibile principale dell’applicazione (.exe) e il file di configurazione dove vengono salvate le impostazioni dell’utente.

Diagramma di deployment:



Per semplificare la struttura del programma è stato utilizzato lo strumento Fody di Costura. Grazie a Fody è possibile effettuare una fusione di tutti gli assembly in fase di release evitando che il file .exe dell’applicazione principale necessiti di tutti i file .dll delle librerie utilizzate nella stessa cartella, infatti grazie a Fody tutti questi assembly vengono inseriti direttamente nel file eseguibile riducendo il numero di file per il quale è necessario effettuare il deployment a uno.

Al primo utilizzo dell’applicazione il file di configurazione risulterà assente, ma non appena l’utente inserirà le proprie impostazioni il file associato BitMoneto.exe.config verrà automaticamente creato sulla macchina utente nella stessa cartella dell’ eseguibile.