roundabout

Roundabout - Etherless

Analisi dei requisiti

Versione	0.1.0
Approvazione	
Redazione	Egon Galvani
	Antonio Zlatkovski
Verifica	Feim Jakupi
Stato	Non approvato
Uso	Esterno
Destinato a	Roundabout
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin

Descrizione
Analisi dei requisiti del gruppo Roundabout per la realizzazione del progetto Etherless

team.roundabout.13@gmail.com

Registo delle modifche

Versione	Data	Nominativo	Ruolo	Descrizione
0.1.6	2020-03-28	Antonio Zlatkovski	*Analista*	Stesura UC14, UC17, UC19
0.1.5	2020-03-28	Egon Galvani	*Analista*	Stesura UC15, UC16, UC18
0.1.4	2020-03-27	Antonio Zlatkovski	*Analista*	Stesura UC11, UC12, UC13
0.1.3	2020-03-27	Egon Galvani	*Analista*	Stesura UC5, UC6, UC7
0.1.2	2020-03-26	Antonio Zlatkovski	*Analista*	Stesura UC8, UC9, UC10
0.1.1	2020-03-26	Egon Galvani	*Analista*	Stesura UC1, UC2, UC3, UC4
0.1.0	2020-03-25	Feim Jakupi	*Verificatore*	Verifica sezione §1, §2
0.0.3	2020-03-18	Egon Galvani	*Analista*	Stesura §2, §3.1
0.0.2	2020-03-18	Antonio Zlatkovski	*Analista*	Stesura §1
0.0.1	2020-03-14	Egon Galvani	*Analista*	Creata struttura del documento in LaTeX

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il presente documento ha l'obiettivo di individuare le funzionalità e i casi d'uso previsti dal progetto Etherless, proposto dall'azienda Red Babel. Le informazioni qui riportate, individuate da un'approfondita analisi del capitolato stesso e dai successivi incontri con il proponente, rappresentano la base di partenza per la successiva fase di progettazione.

1.2 Scopo del prodotto

Si svuole sviluppare una piattaforma cloud che consenta agli sviluppatori di fare il deploy di funzioni Javascript e gestisca il pagamento per la loro esecuzione tramite la piattaforma Ethereum. Il prodotto finale prevede quindi l'integrazione di due tecnologie, Serverless ed Ethereum.
Il lato Serverless si occupa dell'esecuzione delle funzioni fornite dagli sviluppatori. Tali funzioni vengono salvate ed eseguite in un servizio cloud esterno, quale Amazon Web Services.
La richiesta di utilizzo di una funzione e il successivo pagamento vengono invece gestiti tramite la piattaforma Ethereum sfruttando gli smart contracts. Il pagamento viene effettuato in ETH. Una percetuale significativa di ogni pagamento viene riservata agli amministratori del servizio.
Lo sviluppatore e l'utente finale interagiscono con il prodotto tramite una CLI che prevede alcuni comandi intuitivi.

1.3 Documenti complementari

1.3.1 Glossario

I termini tecnici utilizzati in questo documento sono contrassegnati da una 'G' a pedice ed ulteriormente approfonditi nel Glossario denominato "Glossario Esterno 1.0.0", disponibile in allegato al presente documento.

1.4 Riferimenti

1.4.1 Normativi

- **Norme di progetto**: *Norme di progetto v1.0.0*;
- **Verbale esterno con il proponente**: *VE_2020_03_18*;
- **Capitolato d'appalto Etherless**: https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C2.pdf.

1.4.2 Informativi

- **Studio di fattibilità**: *Studio di fattibilità v1.0.0*;
- **Sito ufficiale Ethereum**: https://ethereum.org/;
- **Sito del framework Serverless**: https://serverless.com/;
- **Amazon Web Services**: https://aws.amazon.com/.

2 Descrizione generale

2.1 Obiettivi del prodotto

Il progetto *Etherless* ha come obiettivo finale creare una piattaforma per permettere agli sviluppatori di fare il deploy di funzioni JavaScript, preoccupandosi solamente della relativa codifica e non dell'architettura sottostante. Allo stesso tempo, tali funzioni vengono messe a disposizione agli altri utenti, che possono eseguirle e pagare secondo un modello Caas (Computation As a Service), cioè solamente per il tempo e le risorse richieste dalla loro esecuzione.

2.2 Funzionalità del prodotto

L'applicativo deve fornire la possibilità agli sviluppatori di caricare nel cloud le proprie funzioni e renderle disponibili secondo un'ideologia Faas (Function As A Service). Gli utenti finali possono usufruire di tali servizi pagando; in questo modo gli sviluppatori non si devono preoccupare della gestione dell'infrastruttura alla base dei servizi e possono guadagnare ad ogni esecuzione di una funzione da loro caricata.
Nello specifico:

- gli utenti finali e gli sviluppatori possono:
  a. autenticarsi all'interno della rete Ethereum;
  b. eseguire una funzione presente della piattaforma e visualizzarne i risultati;
  c. elencare tutte le funzioni disponibili nella piattaforma;
  d. visualizzare i dettagli di una determinata funzione;
  e. visualizzare la propria cronologia di esecuzione di funzioni;
  f. ricercare funzioni in base ad un termine di ricerca.
- gli sviluppatori possono:
  a. caricare all'interno della piattaforma delle proprie funzioni JavaScript;
  b. eliminare una funzione da loro precedentemente caricata;
  c. modificare le informazioni e il codice relativo ad una loro funzione.

2.3 Analisi della struttura

Il prodotto si divide nelle seguenti parti:

- **Etherless-cli:** interfaccia a linea di comando tramite cui i vari utenti della piattaforma interagiscono con Etherless;
- **Etherless-smart:** insieme di smart-contract che si occupano della gestione della comunicazione tra *Etherless-cli* ed *Etherless-server* e del pagamento richiesto per l'esecuzione delle funzioni;
- **Etherless-server:** si occupa di ascoltare gli eventi trasmessi da *Etherless-smart* e di avviare l'esecuzione delle funzioni così richieste. I risultati ottenuti vengono inviati tramite un ulteriore evento nella blockchain e mostrati all'utente attraverso *Etherless-cli*;

2.3.1 Comandi disponibili

Etherless-cli mette a disposizione dell'utente i seguenti comandi:

- **etherless-init:** avvio dell'applicativo;
- **etherless-login:** esecuzione della procedura di autenticazione all'interno della rete Ethereum;
- **etherless-signup:** creazione di un nuovo wallet all'interno della rete Ethereum;
- **etherless-logout:** esecuzione del logout dalla propria utenza;
- **etherless-whoami:** visualizzazione dell'indirizzo associato all'utenza corrente;
- **etherless-list:** elenco di tutte le funzioni disponibili all'interno della piattaforma *Etherless*;
- **etherless-deploy:** esecuzione del deploy di una determinata funzione;
- **etherless-run:** esecuzione di una funzione e visualizzazione del relativo risultato;
- **etherless-mylist:** visualizzazione del nome delle funzioni di cui l'utente corrente ha eseguito il deploy;
- **etherless-info:** visualizzazione di informazioni più dettagliate di una determinata funzione;
- **etherless-search:** elenco delle funzioni con nome simile ad un termine inserito;
- **etherless-delete:** eliminazione di una determinata funzione;
- **etherless-logs:** visualizzazione dei log di esecuzione di una specifica funzione;
- **etherless-history:** visualizzazione della cronologia di esecuzione dell'utente corrente;
- **etherless-exit:** chiusura dell'applicativo.

2.3.2 Vincoli implementativi

Viene richiesto che:

- gli smart-contract possano essere aggiornati;
- *Etherless* sia sviluppato utilizzando Typescript 3.6, in particolare facendo riferimento all'approccio di promise/async-await;
- typescript-eslint deve essere utilizzato come rinforzo di ESLint durante il processo di sviluppo;
- *Etherless-server* deve essere implementato usando il framework Serverless.

2.3.3 Ambienti di esecuzione

Viene inoltre richiesto che il progetto possa funzionare nei seguenti ambienti di esecuzione:

- **Locale:** viene simulata una rete Ethereum all'interno della macchina locale, per tale scopo può essere utilizzata la rete Ethereum testrpc messa a disposizione da Truffle;
- **Test:** ambiente di test, in cui vengono eseguiti i test di verifica; la rete può coincidere con quella usata in ambiente locale;
- **Staging:** ambiente pubblicamente accessibile, per tale scopo può essere usata la testnet Ropsten di Ethereum;
- **Produzione:** non richiesto, ma il progetto deve essere pronto per la produzione. In questo caso si fa riferimento alla Ethereum mainnet.

2.3.4 Vincoli generali

L'utente, per usufruire del servizio, deve possedere una connessione internet e aver installato Node.js e il modulo relativo a *Etherless-cli*.
Per la gestione del servizio, e quindi l'esecuzione di *Etherless-server*, oltre ai requisiti già indicati è necessario avere un'utenza AWS e aver completato correttamente la relativa configurazione.

3 Casi d'uso

3.1 Attori dei casi d'uso

3.1.1 Attori primari

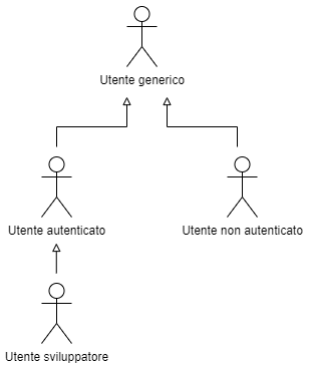

Figura 3.1.1: Gerarchia degli attori

Utente generico
Si riferisce ad un utente generico che ha avviato l'applicativo.

Utente non autenticato
Si riferisce ad un utente generico che non ha ancora effettuato l'autenticazione all'interno della rete Ethereum.

Utente autenticato
Si riferisce ad un utente generico che si è autenticato nel sistema tramite il comando di login. Ciò implica che sia in possesso di una chiave o una frase mnemonica valida all'interno della rete Ethereum.

Utente sviuppatore
Si riferisce ad un utente autenticato che ha svolto il deploy di almeno una sua funzione.

3.1.2 Attori secondari

Ethereum network
Piattaforma decentralizzata utilizzata per la gestione dell'autenticazione, dei pagamenti e della comunicazione tra i vari moduli della piattaforma *Etherless*.

3.2 Elenco dei casi d'uso

In questa sezione vengono elencati tutti i casi d'uso individuati. Ogni caso d'uso rappresenta uno scenario per uno o più attori, ed è descritto tramite degli appositi diagrammi dei casi d'uso.

Operazioni utenti

Di seguito sono riportati tutti i casi d'uso aventi come attore primario l'utente generico, l'utente non autenticato o l'utente autenticato.

3.2.1 UC1 - Visualizzazione guida introduttiva

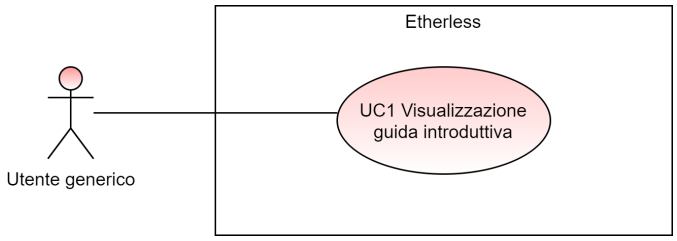

Figura 3.2.1: UC1 - Visualizzazione guida introduttiva

- **Attori primari:** utente generico;
- **Descrizione:** dopo aver eseguito il comando *init* il sistema mostra una guida contenente una breve spiegazione riguardante i comandi di *login* e *signup*. Tale guida ha lo scopo di aiutare l'utente a procedere nell'utilizzo dell'applicativo;
- **Scenario principale:**
  - l'utente inserisce il comando *init*;
  - viene visualizzata la guida introduttiva.
- **Precondizione:** l'applicativo viene avviato correttamente e il sistema è raggiungibile;
- **Postcondizione:** vengono fornite all'utente tutte le informazioni necessarie per procedere con l'utilizzo dell'applicativo.

3.2.2 UC2 - Aiuto comandi

- **Attori primari:** utente generico;
- **Descrizione:** dopo aver inserito il comando *help*, l'utente visualizza le informazioni dettagliate riguardo al comando desiderato;
- **Scenario principale:**
  - l'utente inserisce il comando *help command_name*;
  - vengono visualizzate le informazioni dettagliate del comando richiesto.
- **Precondizione:** l'utente vuole ottenere maggiori informazioni riguardo ad un comando;
- **Postcondizione:** vengono visualizzate le informazioni dettagliate del comando di interesse.

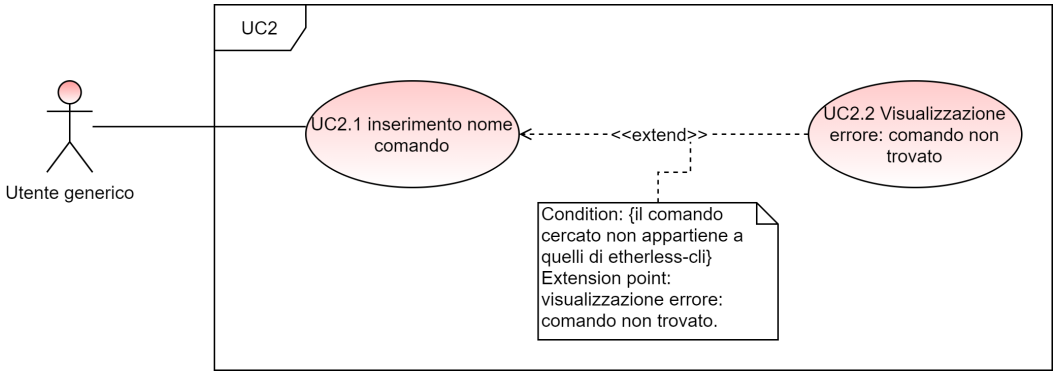

Figura 3.2.2: UC2 - Aiuto comandi

3.2.3 UC2.1 - Inserimento nome comando

- **Attori primari:** utente generico;
- **Descrizione:** l'utente digita il nome del comando a cui è interessato;
- **Scenario principale:** l'utente inserisce il nome del comando;
- **Estensioni:**
  - **UC2.2:** se il comando di cui si vogliono ottenere informazioni non appartiene a quelli messi a disposizione da *etherless-cli* viene mostrato un messaggio di errore.
- **Precondizione:** l'utente vuole ottenere maggiori informazioni riguardo ad un comando messo a disposizione da *etherless-cli*;
- **Postcondizione:** l'utente ha inserito il comando di interesse.

3.2.4 UC2.2 - Visualizzazione errore: comando non trovato

- **Attori primari:** utente generico;
- **Descrizione:** l'utente inserisce un comando che non è tra quelli messi a disposizione da *etherless-cli* e il sistema provvede a mostrare un messaggio di errore;
- **Scenario principale:** l'utente visualizza un messaggio di errore relativo all'assenza del comando richiesto tra quelli messi a disposizione da *etherless-cli*;
- **Estensioni:**
  - **UC4:** tramite l'utilizzo del flag *–s* l'utente può richiedere che venga creato nella cartella corrente un file contenente le informazioni del nuovo account Ethereum creato.
- **Precondizione:** l'utente richiede di visualizzare maggiori informazioni relative ad un comando non presente in *etherless-cli*;
- **Postcondizione:** viene visualizzato un messaggio di errore.

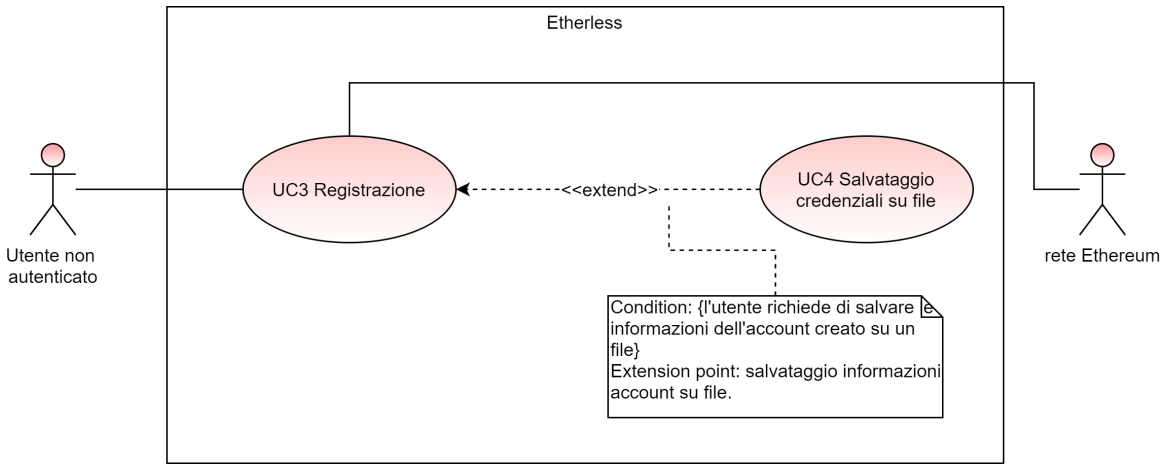

Figura 3.2.3: UC3 - Registrazione: schema generale

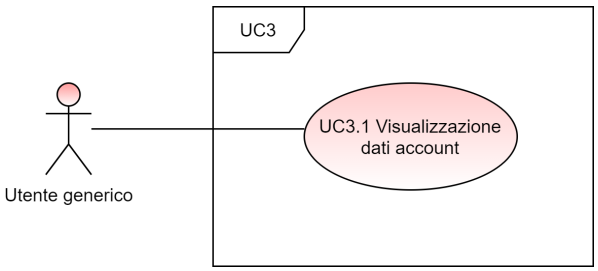

Figura 3.2.4: UC3 - Registrazione

3.2.5 UC3 - Registrazione

- **Attori primari:** utente non autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** dopo aver richiesto la creazione di un nuovo account Ethereum tramite l'utilizzo del comando *signup*, l'utente visualizza a video le credenziali del nuovo account creato;
- **Scenario principale:**
  - l'utente richiede la creazione di un account all'interno della rete Ethereum mediante il comando *signup*;
  - vengono visualizzate le credenziali del nuovo account creato.
- **Estensioni:**
  - **UC4:** tramite l'utilizzo del flag *–s* l'utente può richiedere che venga creato nella cartella corrente un file contenente le informazioni del nuovo account Ethereum creato.
- **Precondizione:** l'utente vuole creare un nuovo account;
- **Postcondizione:** l'account è stato creato correttamente.

3.2.6 UC3.1 - Visualizzazione dati account

- **Attori primari:** utente non autenticato;
- **Descrizione:** dopo la creazione di un nuovo account all'interno della rete Ethereum il sistema mostra nella CLI le relative credenziali; vengono mostrate sia la private key sia la mnemonic phrase; in questo modo l'utente potrà decidere autonomamente quale usare durante la fase di autenticazione;
- **Scenario principale:** l'utente visualizza le credenziali relative al nuovo account creato;
- **Precondizione:** è stato creato correttamente un nuovo account sulla rete Ethereum;
- **Postcondizione:** vengono mostrate a video le credenziali del nuovo account.

3.2.7 UC4 - Salvataggio credenziali su file

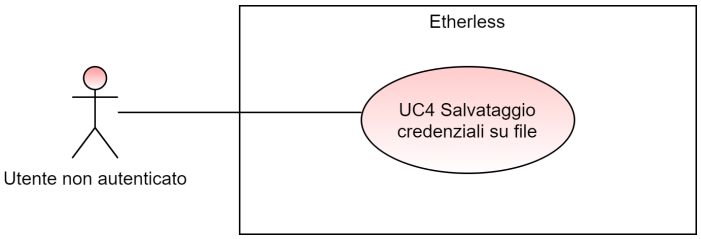

Figura 3.2.5: UC4 - Salvataggio credenziali su file: schema generale

- **Attori primari:** utente non autenticato;
- **Descrizione:** a seguito della creazione di una nuova utenza Ethereum l'utente può richiedere al sistema il salvataggio su file delle credenziali dell'account creato. Tali credenziali includono address, private key e mnemonic phrase.
- **Scenario principale:**
  - l'utente richiede di salvare su file le informazioni del nuovo account creato;
  - il sistema si occupa del corretto salvataggio delle credenziali.
- **Precondizione:** l'utente richiede di salvare le informazioni relative al nuovo account tramite l'utilizzo del flag *-s*;
- **Postcondizione:** le credenziali sono state salvate con successo all'interno del file.

3.2.8 UC5 - Login manuale

- **Attori primari:** utente non autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente può utilizzare il comando *login* per autenticarsi all'interno della rete Ethereum;

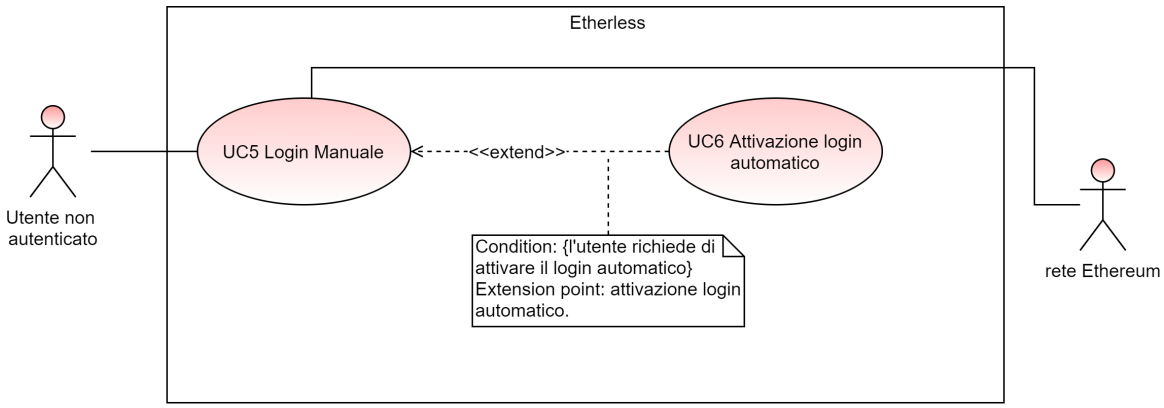

Figura 3.2.6: UC5 - Login manuale: schema generale

- **Scenario principale:** l'utente esegue il comando *login* indicando manualmente le credenziali necessarie;
- **Estensioni:**
  - **UC6:** tramite l'apposito flag *-r* l'utente può richiedere che le credenziali inserite, se corrette, vengano memorizzate e usate per l'autenticazione automatica in caso di accessi futuri;
- **Precondizione:** l'utente tenta di autenticarsi alla piattaforma;
- **Postcondizione:** l'utente si è autenticato correttamente.

3.2.9 UC5.1 - Inserimento credenziali di accesso

- **Attori primari:** utente non autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente procede all'inserimento delle credenziali necessarie per l'autenticazione. Oltre all'inserimento obbligatorio dell'address viene richiesta, a scelta dell'utente, la private key o mnemonic phrase;
- **Scenario principale:** l'utente provvede ad inserire le credenziali necessarie per l'accesso.
- **Specializzazioni:**
  - **UC5.2:** l'utente decide di eseguire il login tramite private key;
  - **UC5.3:** 'utente decide di eseguire il login tramite mnemonic phrase.
- **Estensioni:**
  - **UC5.6:** se le credenziali inserite sono errate allora il sistema mostra un messaggio di errore.
- **Precondizione:** l'utente ha inserito il comando *login*;
- **Postcondizione:** l'utente ha inserito correttamente le credenziali per effettuare l'accesso.

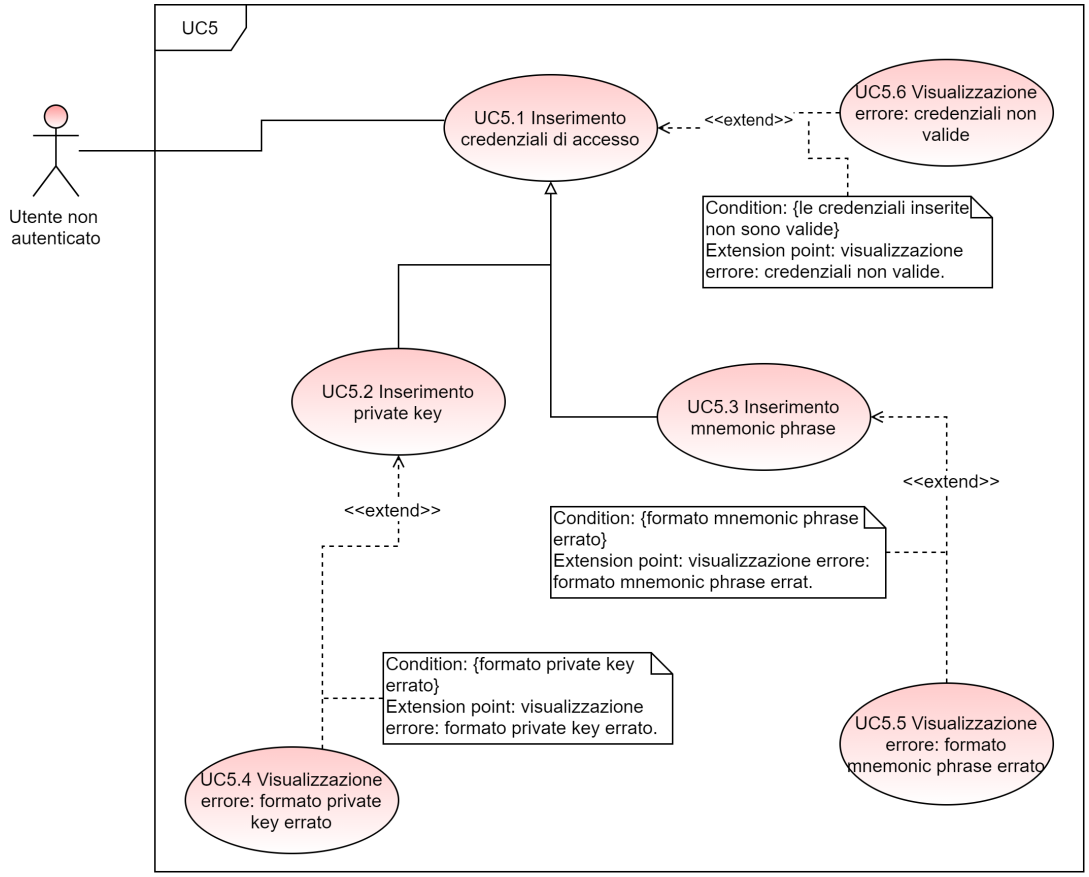

Figura 3.2.7: UC5 - Login manuale

3.2.10 UC5.1.1 - Inserimento address

- **Attori primari:** utente non autenticato;
- **Descrizione:** al fine di portare a termine il processo di autenticazione l'utente deve inserire il proprio address;
- **Scenario principale:** l'utente inserisce il proprio address;
- **Estensioni:**
  - **UC5.1.2:** se l'indirizzo inserito non rispetta il formato adeguato viene mostrato un messaggio di errore a riguardo;
- **Precondizione:** l'utente ha inserito il comando *login*;
- **Postcondizione:** l'utente ha inserito correttamente il proprio address.

3.2.11 UC5.1.2 - Visualizzazione errore: formato address errato

- **Attori primari:** utente non autenticato;

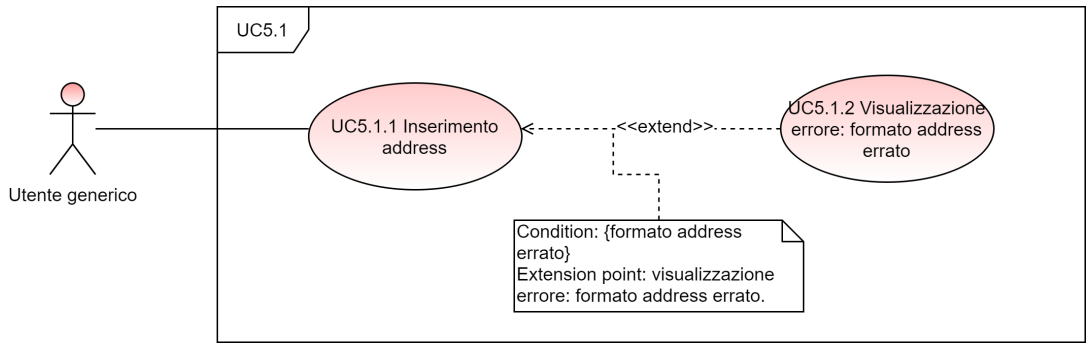

Figura 3.2.8: UC5.1 - Inserimento credenziali di accesso

- **Descrizione:** l'utente tenta di eseguire la procedura di login con un address non conforme al formato richiesto da Ethereum;
- **Scenario principale:** il sistema mostra un messaggio di errore;
- **Precondizione:** l'utente vuole eseguire la procedura di login manuale;
- **Postcondizione:** il sistema avvisa l'utente dell'errato formato dell'address tramite un messaggio di errore.

3.2.12 UC5.2 - Inserimento private key

- **Attori primari:** utente non autenticato;
- **Descrizione:** al fine di terminare la procedura di autenticazione l'utente deve inserire la propria private key;
- **Scenario principale:** dopo aver deciso di volersi autenticare tramite l'utilizzo della propria private key, l'utente procede con l'inserimento di quest'ultima;
- **Estensioni:**
  - **UC5.4:** nel caso di inserimento di una private key in formato errato viene mostrato un messaggio di errore.
- **Precondizione:** l'utente ha deciso di autenticarsi tramite uso della propria private key e ha già inserito nella CLI il comando login seguito dal suo address;
- **Postcondizione:** l'utente ha inserito correttamente la propria private key.

3.2.13 UC5.3 - Inserimento mnemonic phrase

- **Attori primari:** utente non autenticato;
- **Descrizione:** al fine di terminare la procedura di autenticazione l'utente deve inserire la propria mnemonic phrase;
- **Scenario principale:** dopo aver deciso di volersi autenticare all'interno della rete Ethereum tramite mnemonic phrase, l'utente procede all'inserimento di quest'ultima;

- **Estensioni:**
  - **UC5.5:** nel caso di inserimento di una mnemonic phrase in formato errato viene mostrato un messaggio di errore.
- **Precondizione:** l'utente ha deciso di autenticarsi tramite l'uso della propria mnemonic phrase e ha già inserito nella CLI il comando di login seguito dal suo address;
- **Postcondizione:** l'utente ha inserito correttamente la propria mnemonic phrase.

3.2.14 UC5.4 - Visualizzazione errore: formato private key errato

- **Attori primari:** utente non autenticato;
- **Descrizione:** l'utente tenta di eseguire la procedura di login con una private key non conforme al formato richiesto da Ethereum;
- **Scenario principale:** il sistema mostra un messaggio di errore relativo al formato della private key inserita;
- **Precondizione:** l'utente ha inserito una private key in formato errato durante la procedura di autenticazione;
- **Postcondizione:** viene mostrato all'utente un messaggio di errore.

3.2.15 UC5.5 - Visualizzazione errore: formato mnemonic phrase errato

- **Attori primari:** utente non autenticato;
- **Descrizione:** l'utente tenta di eseguire la procedura di login con una mnemonic phrase che non rispetta il formato richiesto;
- **Scenario principale:** il sistema mostra un messaggio di errore relativo al formato della menmonic phrase inserita;
- **Precondizione:** l'utente ha inserito una mnemonic phrase in formato errato durante la procedura di autenticazione;
- **Postcondizione:** viene visualizzato nella CLI un messaggio relativo all'errore considerato.

3.2.16 UC5.6 - Visualizzazione errore: credenziali non valide

- **Attori primari:** utente non autenticato;
- **Descrizione:** l'utente tenta di eseguire la procedura di autenticazione all'interno della rete Ethereum tramite delle credenziali non corrette; il sistema mostra quindi un messaggio di errore;
- **Scenario principale:** le credenziali inserite non sono corrette e non permettono quindi la corretta autenticazione dell'utente, che verrà avvisato tramite un messaggio di errore;
- **Precondizione:** l'utente tenta di autenticarsi tramite l'utilizzo del comando login seguito dalle credenziali richieste;
- **Postcondizione:** il sistema mostra all'utente un errore.

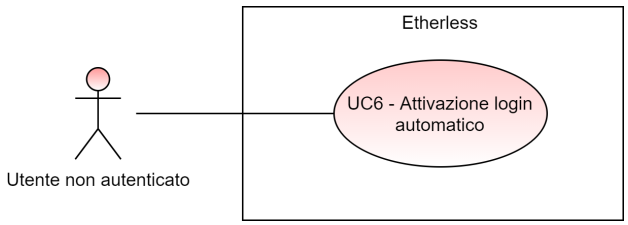

Figura 3.2.9: UC6 - Attivazione login automatico: schema generale

3.2.17 UC6 - Attivazione login automatico

- **Attori primari:** utente non autenticato;
- **Descrizione:** dopo aver inserito le credenziali di accesso l'utente specifica, tramite l'apposito flag *-r*, la richiesta di essere ricordato anche per eventuali accessi futuri;
- **Scenario principale:**
  - l'utente inserisce il comando di login seguito dalle credenziali necessarie e dal flag *-r*;
  - a seguito di una corretta autenticazione le credenziali dell'utente vengono salvate per gli accessi futuri.
- **Precondizione:** l'utente richiede di attivare il login automatico tramite il flag *-r*;
- **Postcondizione:** le informazioni necessarie all'autenticazione dell'utente vengono salvate correttamente in vista di accessi futuri.

3.2.18 UC7 - Login automatico

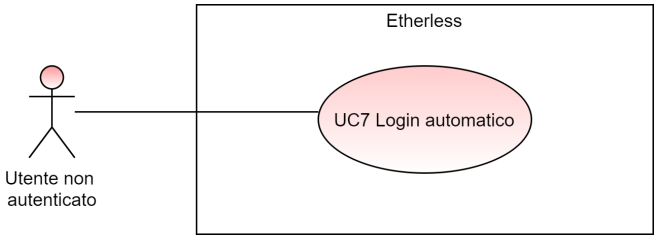

Figura 3.2.10: UC7 - Login automatico: schema generale

- **Attori primari:** utente non autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** in maniera automatica il sistema si occupa dell'autenticazione dell'utente;
- **Scenario principale:** l'utente avvia l'applicativo tramite il comando *init* e viene autenticato in maniera automatica;
- **Precondizione:** l'utente ha eseguito il login manuale almeno una volta, indicando esplicitamente la volontà di essere ricordato [**UC6**];
- **Postcondizione:** l'utente si è autenticato con successo

3.2.19 UC8 - Disconnessione dal servizio

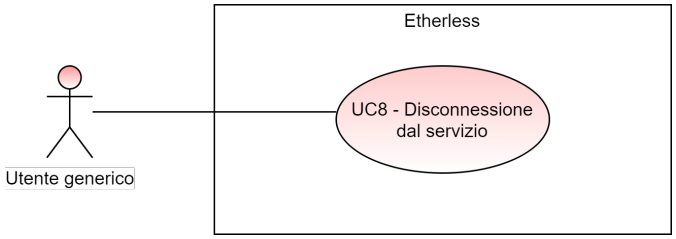

Figura 3.2.11: UC8 - Disconnessione dal servizio: schema generale

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente richiede la disconnessione dal servizio eseguendo il comando *logout*. Il sistema effettua la disconnessione;
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *logout*;
  - l'utente viene disconnesso dal servizio.
- **Precondizione:** l'utente è stato autenticato correttamente e richiede di essere disconnesso;
- **Postcondizione:** l'utente viene disconnesso con successo.

3.2.20 UC9 - Visualizzazione address utente

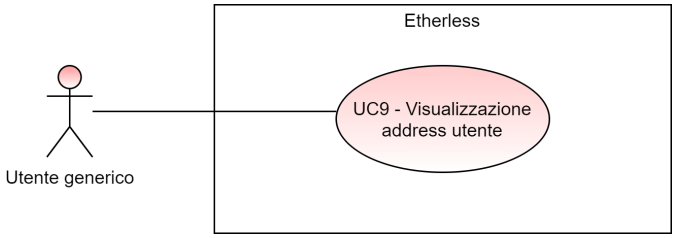

Figura 3.2.12: UC9 - Visualizzazione address utente: schema generale

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente richiede la visualizzazione del campo address associato al proprio account eseguendo il comando *whoami*. Il sistema stampa a video tale informazione;
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *whoami*;
  - il sistema visualizza il campo address associato all'utente

- **Precondizione:** l'utente è stato autenticato correttamente e richiede di visualizzare l'indirizzo associato alla propria utenza;
- **Postcondizione:** la CLI riporta il campo address associato all'account dell'utente.

3.2.21 UC10 - Informazioni dettagliate funzione

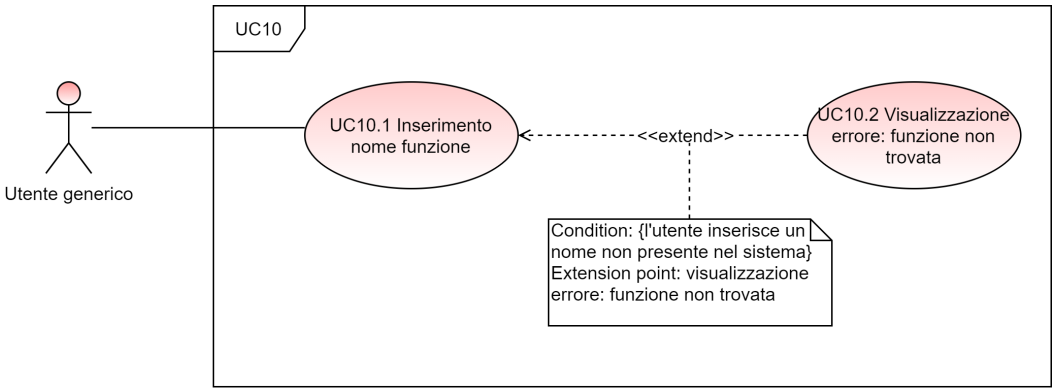

Figura 3.2.13: UC10 - Informazioni dettagliate funzione

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede la visualizzazione della descrizione completa di una funzione di cui conosce il nome eseguendo il comando *info function_name*. Il sistema riporta le seguenti informazioni:
  - firma della funzione;
  - descrizione completa della funzione.
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *info function_name*;
  - vengono visualizzate le informazioni relative alla funzione in questione.
- **Precondizione:** l'utente inserisce correttamente ed esegue il comando *info*;
- **Postcondizione:** la CLI riporta la descrizione completa della funzione in questione.

3.2.22 UC10.1 - Inserimento nome funzione

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente inserisce il nome della funzione della quale desidera visualizzare le informazioni generali;
- **Scenario principale:** l'utente inserisce il comando *info* seguito dal nome della funzione;

- **Estensioni:**
  - **UC10.2:** l'utente inserisce un nome non presente nel sistema, viene quindi visualizzato un messaggio di errore.
- **Precondizione:** l'utente ha digitato il comando *info*;
- **Postcondizione:** l'utente ha inserito correttamente il nome della funzione.

3.2.23 UC10.2 - Visualizzazione errore: funzione non trovata

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente richiede la visualizzazione della descrizione completa di una funzione specificando un identificativo non presente nel sistema. Il sistema riporta un messaggio di errore relativo alla mancata presenza della funzione;
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *info function_name*;
  - viene visualizzato un messaggio di errore relativo alla mancata presenza della funzione.
- **Precondizione:** l'utente esegue il comando info specificando l'identificativo di una funzione non esistente.
- **Postcondizione:** la CLI riporta un messaggio di errore.

3.2.24 UC11 - Ricerca funzione per nome

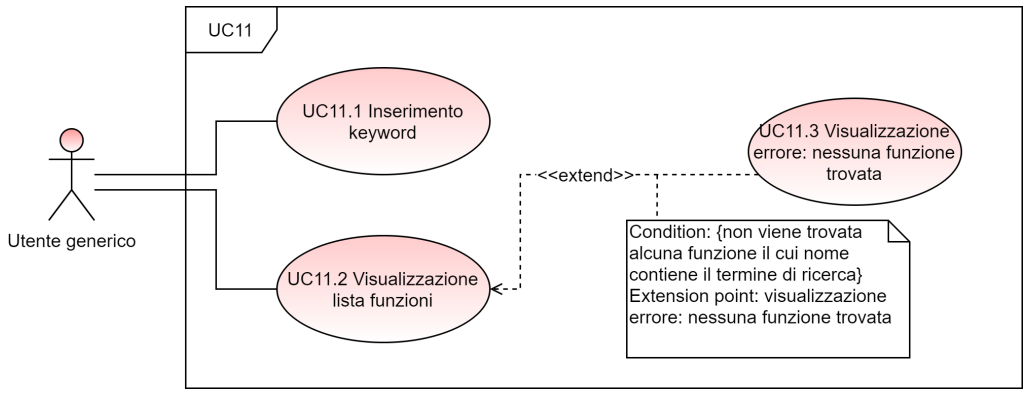

Figura 3.2.14: UC11 - Ricerca funzione per nome

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede la visualizzazione della lista di tutte le funzioni il cui nome contiene un certo termine di ricerca eseguendo il comando *search keyword*. Il sistema riporta la lista di tali funzioni.

- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *search keyword*
  - il sistema stampa la lista di tutte le funzioni il cui nome contiene il termine di ricerca.
- **Precondizione:** l'utente desidera individuare tutte le funzioni correlate ad un certo termine;
- **Postcondizione:** la CLI riporta la lista di tutte le funzioni il cui nome contiene il termine di ricerca specificato nel comando *search*.

3.2.25 UC11.1 - Inserimento keyword

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente inserisce, nel campo *keyword*, il termine di ricerca che desidera individuare all'interno dei nomi delle funzioni disponibili nella piattaforma Etherless;
- **Scenario principale:** l'utente inserisce il termine di ricerca;
- **Precondizione:** l'utente ha digitato nella CLI il comando *search*;
- **Postcondizione:** l'utente inserisce correttamente il termine di ricerca.

3.2.26 UC11.2 - Visualizzazione lista funzioni

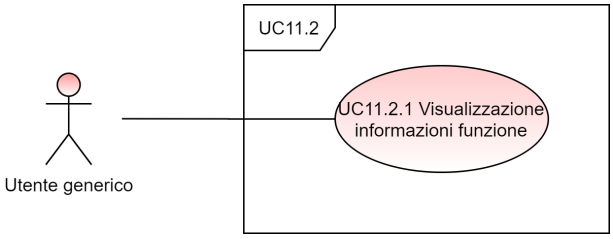

Figura 3.2.15: UC11 - Ricerca funzione per nome

- **Attori primari:** utente autenticato;
- **Descrizione:** viene visualizzata la lista ritornata dal comando search
- **Scenario principale:** viene visualizzata una lista di tutte le funzioni che nel nome contengono il termine di ricerca specificato;
- **Estensioni:**
  - **UC11.3:** non viene trovata alcuna funzione il cui nome contiene il termine di ricerca. Viene di conseguenza visualizzato un messaggio di errore;
- **Precondizione:** l'utente ha inserito ed eseguito correttamente il comando *search*;
- **Postcondizione:** la CLI riporta la lista di funzioni ritornata dal comando *search*.

3.2.27 UC11.2.1 - Visualizzazione informazioni funzione

- **Attori primari:** utente autenticato;
- **Descrizione:** vengono visualizzate le informazioni rilevanti della funzione, ovvero:
  - firma della funzione;
  - descrizione della funzione;
  - proprietario della funzione;
- **Scenario principale:** vengono visualizzate le informazioni rilevanti della funzione.
- **Precondizione:** l'utente ha inserito ed eseguito correttamente il comando di ricerca;
- **Postcondizione:** la CLI riporta le informazioni rilevanti della funzione.

3.2.28 UC11.3 - Visualizzazione errore: "Nessuna funzione trovata

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede la visualizzazione della lista di tutte le funzioni il cui nome contiene un certo termine di ricerca eseguendo il comando *search keyword*. Nel sistema non è presente alcuna funzione che soddisfi tale criterio, di conseguenza viene visualizzato un messaggio di errore.
- **Scenario principale:** viene visualizzato un messaggio di errore relativo alla mancata presenza di funzioni il cui nome contiene il termine di ricerca specificato.
- **Precondizione:** l'utente ha inserito una keyword che non ha portato ad alcun risultato;
- **Postcondizione:** viene visualizzato un messaggio che descrive l'errore.

3.2.29 UC12 - Esecuzione funzione

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede l'esecuzione di una delle funzioni presenti nel sistema eseguendo il comando *run function_id [parameters_list]*. Il sistema stampa a video il risultato di tale chiamata.
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *run function_id [parameters_list]*;
  - viene visualizzato il risultato dell'esecuzione.
- **Estensioni:**
  - **UC19:** l'utente non dispone di credito sufficiente per portare a termine l'esecuzione della funzione, di conseguenza viene visualizzato un messaggio di errore;
- **Precondizione:** l'utente si è autenticato all'interno della piattaforma e vuole eseguire una delle funzioni disponibili;
- **Postcondizione:** la CLI riporta il risultato dell'esecuzione della funzione.

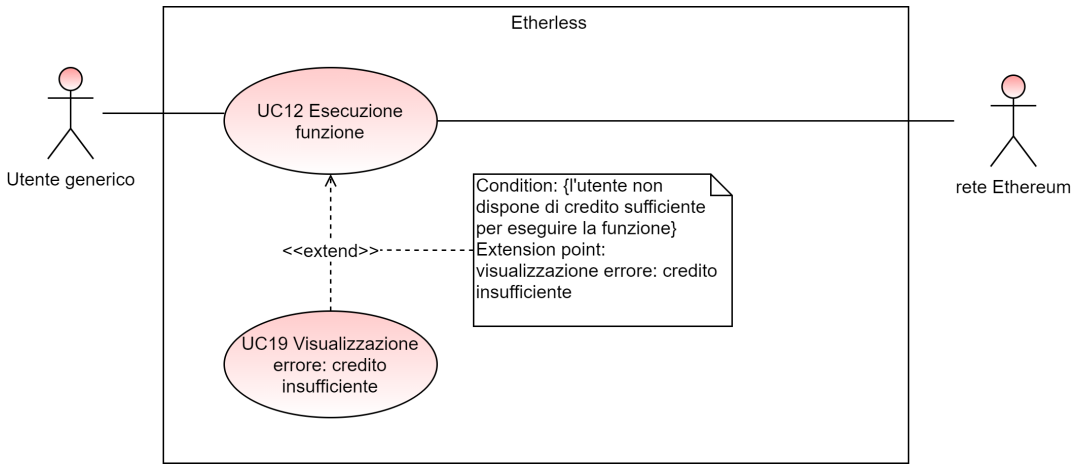

Figura 3.2.16: UC12 - Esecuzione funzione: schema generale

3.2.30 UC12.1 - Inserimento *function_name*

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente inserisce il nome della funzione che desidera eseguire nel campo *function_name*;
- **Scenario principale:** l'utente inserisce il nome della funzione.
- **Estensioni:**
  - **UC12.4:** l'utente inserisce un nome non presente nel sistema, di conseguenza viene visualizzato un messaggio di errore.
- **Precondizione:** l'utente ha digitato all'interno della CLI il comando *run*;
- **Postcondizione:** il campo *function_name* contiene il nome della funzione.

3.2.31 UC12.2 - Inserimento lista parametri

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente inserisce la lista dei parametri richiesti per la corretta esecuzione della funzione nel campo *parameters_list*;
- **Scenario principale:** l'utente inserisce la lista dei parametri della funzione.
- **Estensioni:**
  - **UC12.5:** l'utente inserisce un numero di parametri diverso dal numero di parametri richiesto, di conseguenza viene visualizzato un messaggio di errore.
- **Precondizione:** l'utente ha digitato il comando run seguito dal campo *function_name*;
- **Postcondizione:** il campo *parameters_list* contiene la lista dei parametri.

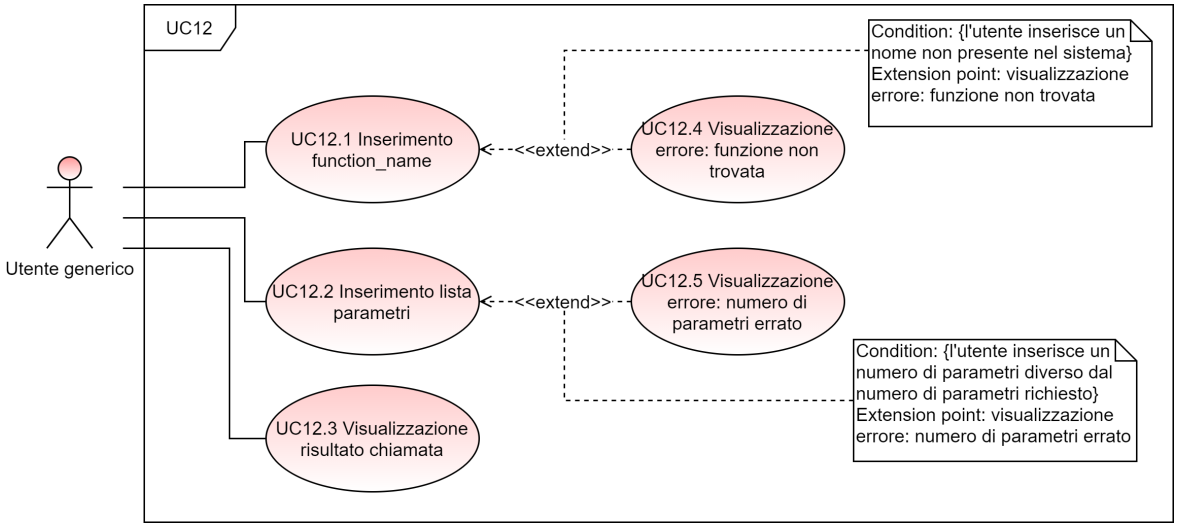

Figura 3.2.17: UC12 - Esecuzione funzione

3.2.32 UC12.3 - Visualizzazione risultato della chiamata

- **Attori primari:** utente autenticato;
- **Descrizione:** Viene visualizzato il risultato dell'esecuzione della funzione.
- **Scenario principale:** viene visualizzato il risultato della chiamata;
- **Precondizione:** l'esecuzione della funzione è andata a buon fine.
- **Postcondizione:** la CLI riporta il risultato della chiamata.

3.2.33 UC12.4 - Visualizzazione errore: funzione non trovata

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente richiede l'esecuzione di una funzione specificando un identificativo non presente nel sistema. Il sistema riporta un messaggio di errore relativo alla mancata presenza della funzione.
- **Scenario principale:** viene visualizzato un messaggio di errore relativo alla mancata presenza di una funzione con il nome specificato;
- **Precondizione:** il nome inserito dall'utente non si riferisce ad alcuna funzione presente all'interno del sistema;
- **Postcondizione:** viene visualizzato un messaggio che descrive l'errore considerato.

3.2.34 UC12.5 - Visualizzazione errore: numero di parametri errato

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;

- **Descrizione:** l'utente richiede l'esecuzione di una funzione inserendo un numero di parametri diverso dal numero di parametri richiesto. Il sistema riporta un messaggio di errore relativo al numero errato di parametri;
- **Scenario principale:** viene visualizzato un messaggio di errore relativo al numero errato di parametri;
- **Precondizione:** il numero di parametri inseriti dall'utente non corrisponde a quelli indicati nella firma della funzione;
- **Postcondizione:** la CLI riporta un messaggio di errore.

3.2.35 UC13 - Elenco funzioni

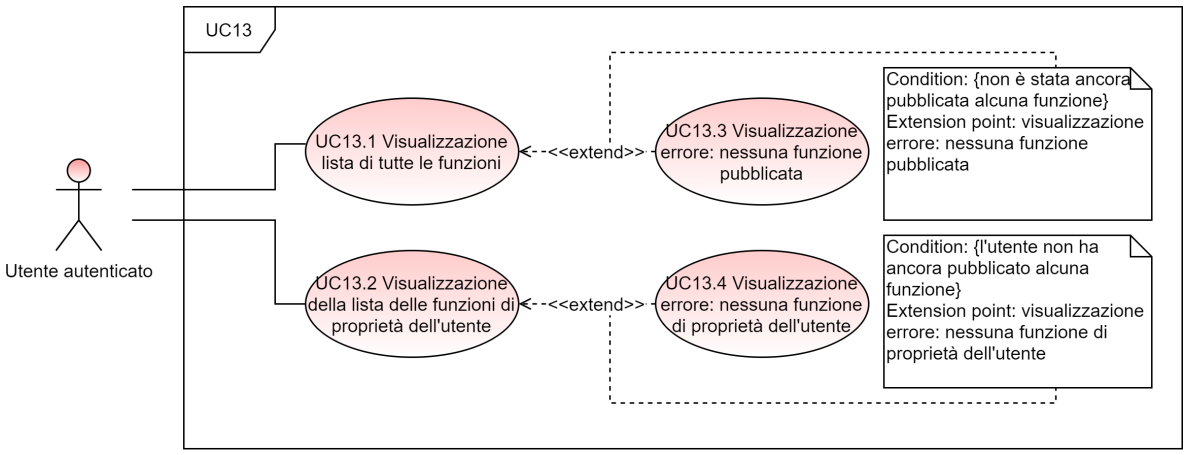

Figura 3.2.18: UC13 - Elenco funzioni

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede la visualizzazione della lista di tutte le funzioni disponibili presso il servizio oppure della lista delle funzioni di sua proprietà. Il sistema stampa a video tale lista.
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *list* oppure il comando *list -m*
  - viene visualizzata la lista completa di tutte le funzioni disponibili oppure soltanto di quelle di proprietà dell'utente.
- **Precondizione:** l'utente desidera visualizzare la lista di tutte le funzioni oppure la lista delle funzioni di sua proprietà;
- **Postcondizione:** la CLI riporta la lista di tutte le funzioni disponibili oppure la lista delle funzioni di proprietà dell'utente.

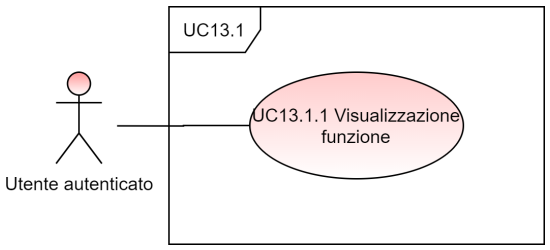

Figura 3.2.19: UC13.1 - Visualizzazione lista di tutte le funzioni

3.2.36 UC13.1 - Visualizzazione lista di tutte le funzioni

- **Attori primari:** utente autenticato;
- **Descrizione:** l'utente richiede la visualizzazione della lista di tutte le funzioni fornite dal servizio eseguendo il comando list. Il sistema stampa a video tale lista;
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *list*.
  - viene visualizzata la lista di tutte le funzioni disponibili presso il servizio;
- **Estensioni:**
  - **UC13.3:** non è stata ancora pubblicata alcuna funzione, di conseguenza viene visualizzato un messaggio di notifica.
- **Precondizione:** l'utente inserisce correttamente ed esegue il comando *list*;
- **Postcondizione:** la CLI riporta la lista di tutte le funzioni fornite dal servizio.

3.2.37 UC13.1.1 - Visualizzazione funzione

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** vengono visualizzate le informazioni rilevanti della funzione, ovvero:
  - firma della funzione;
  - descrizione della funzione;
- **Scenario principale:** vengono visualizzate le informazioni rilevanti della funzione;
- **Precondizione:** l'utente inserisce correttamente ed esegue il comando *list*.
- **Postcondizione:** la CLI riporta le informazioni rilevanti della funzione.

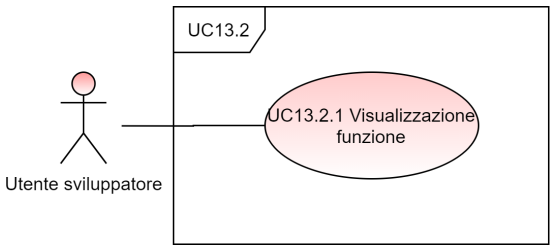

Figura 3.2.20: UC13.2 - Visualizzazione lista funzioni di proprietà dell'utente

3.2.38 UC13.2 - Visualizzazione lista funzioni di proprietà dell'utente

- **Attori primari:** utente sviluppatore;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede la visualizzazione della lista di tutte le funzioni di sua proprietà eseguendo il comando *list -m*. Il sistema stampa a video tale lista;
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *list -m*
  - viene visualizzata la lista di tutte le funzioni di proprietà dell'utente.
- **Estensioni:**
  - **UC13.4:** l'utente non ha ancora pubblicato alcuna funzione, di conseguenza viene visualizzato un messaggio di notifica.
- **Precondizione:** l'utente inserisce correttamente ed esegue il comando *list -m*.
- **Postcondizione:** la CLI riporta la lista di tutte le funzioni di proprietà dell'utente.

3.2.39 UC13.2.1 - Visualizzazione funzione

- **Attori primari:** utente sviluppatore;
- **Descrizione:** vengono visualizzate le informazioni rilevanti della funzione, ovvero:
  - firma della funzione;
  - descrizione della funzione;
- **Scenario principale:** vengono visualizzate le informazioni rilevanti della funzione;
- **Precondizione:** l'utente inserisce correttamente ed esegue il comando *list -m*.
- **Postcondizione:** la CLI riporta le informazioni rilevanti della funzione.

3.2.40 UC13.3 - Visualizzazione errore: nessuna funzione pubblicata

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede la visualizzazione della lista di tutte le funzioni fornite dal servizio ma non viene trovata alcuna funzione. Il sistema visualizza un messaggio di errore relativo alla mancata presenza di funzioni pubblicate;
- **Scenario principale:** viene visualizzato un messaggio di errore relativo all'assenza di funzioni pubblicate;
- **Precondizione:** nessun utente ha provveduto al caricamento di una propria funzione;
- **Postcondizione:** l'utente visualizza un messaggio che spiega l'errore considerato.

3.2.41 UC13.4 - Visualizzazione errore: nessuna funzione di proprietà dell'utente

- **Attori primari:** utente sviluppatore;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede la visualizzazione della lista di tutte le funzioni di sua proprietà non avendo prima pubblicato alcuna funzione. Il sistema visualizza un messaggio di errore relativo alla mancata presenza di funzioni di sua proprietà;
- **Scenario principale:** viene visualizzato un messaggio di errore relativo alla mancata presenza di funzioni di proprietà dell'utente;
- **Precondizione:** l'utente corrente non ha funzioni di sua proprietà;
- **Postcondizione:** il sistema mostra un messaggio di errore.

3.2.42 UC14 - Deploy di funzione

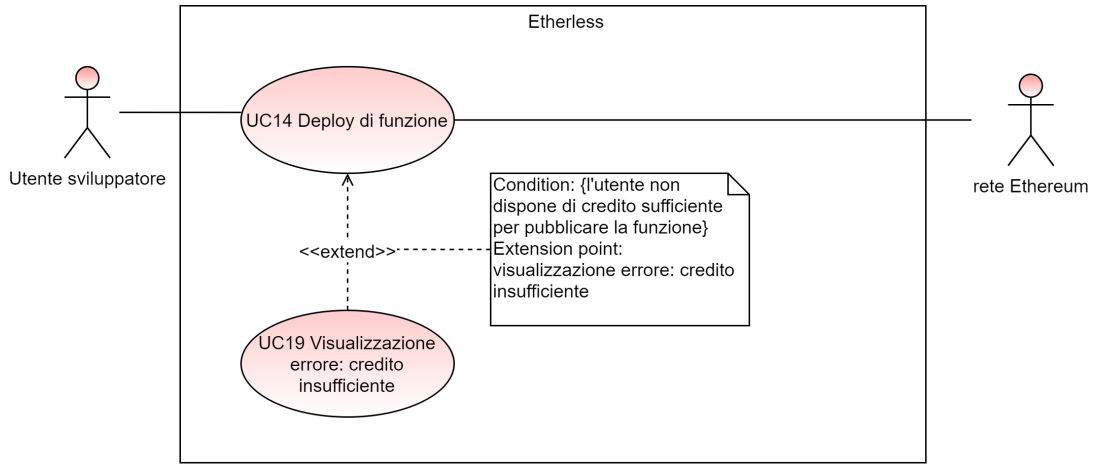

Figura 3.2.21: UC14 - Deploy di funzione: schema generale

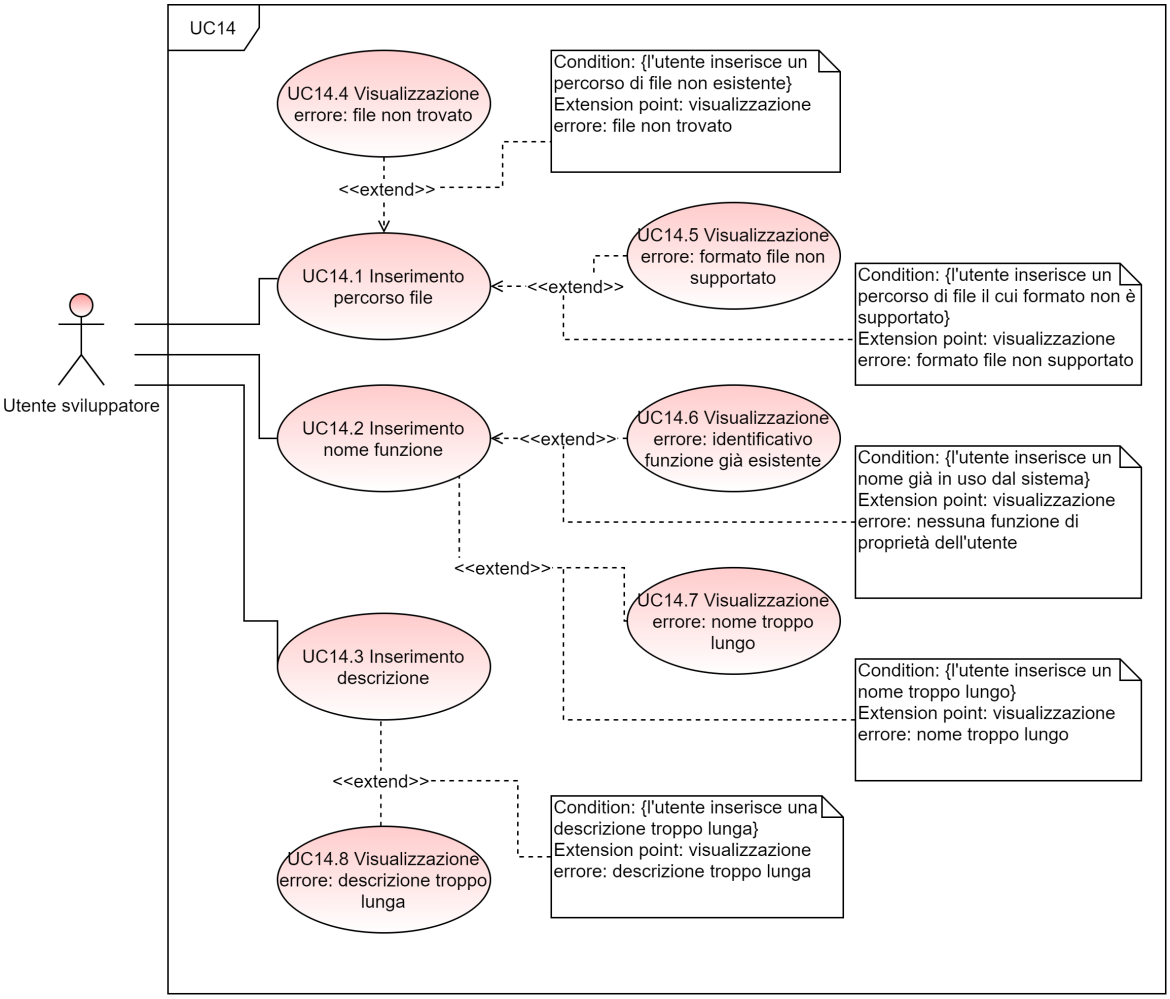

Figura 3.2.22: UC14 - Deploy di funzione

- **Attori primari:** utente sviluppatore;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede di pubblicare una funzione eseguendo il comando *deploy file_path function_name brief_desc*. Il sistema pubblica la funzione;
- **Scenario principale:**
  - l'utente inserisce correttamente ed esegue il comando *deploy file_path function_name brief_desc*;
  - la funzione viene pubblicata.
- **Estensioni:**
  - **UC19:** l'utente non dispone di credito sufficiente per pubblicare la funzione, di conseguenza viene visualizzato un messaggio di errore.

- **Precondizione:** l'utente ha avviato correttamente l'applicativo e desidera eseguire il deploy di una funzione;
- **Postcondizione:** la nuova funzione inserita è disponibile presso il servizio.

3.2.43 UC14.1 - Inserimento percorso file

- **Attori primari:** utente sviluppatore;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** al fine di portare a termine il processo di deploy l'utente inserisce il percorso del file contenente la funzione che si vuole pubblicare;
- **Scenario principale:** l'utente inserisce il percorso del file contenente la funzione che si vuole pubblicare;
- **Estensioni:**
  - **UC14.4:** l'utente inserisce un percorso di file non esistente. Viene di conseguenza visualizzato un messaggio di errore;
  - **UC14.5:** l'utente inserisce un percorso di file il cui formato non è supportato dal sistema. Viene di conseguenza visualizzato un messaggio di errore.
- **Precondizione:** l'utente ha digitato il comando *deploy*;
- **Postcondizione:** il campo *file_path* contiene il percorso del file contenente la funzione che si vuole pubblicare.

3.2.44 UC14.2 - Inserimento nome funzione

- **Attori primari:** utente sviluppatore;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** al fine di portare a termine il processo di deploy l'utente inserisce il nome della funzione che si vuole pubblicare.
- **Scenario principale:** l'utente inserisce il nome della funzione che vuole pubblicare;
- **Estensioni:**
  - **UC14.6:** l'utente inserisce un nome già in uso dal sistema. Viene di conseguenza visualizzato un messaggio di errore.
  - **UC14.7:** l'utente inserisce un nome troppo lungo. Viene di conseguenza visualizzato un messaggio di errore.
- **Precondizione:** L'utente ha digitato il comando deploy seguito dal campo *file_path*;
- **Postcondizione:** il campo *function_name* contiene il nome della funzione che si vuole pubblicare.

3.2.45 UC14.3 - Inserimento descrizione

- **Attori primari:** utente sviluppatore;
- **Descrizione:** al fine di portare a termine il processo di deploy l'utente inserisce una breve descrizione della funzione che si vuole pubblicare;
- **Scenario principale:** l'utente inserisce una breve descrizione della funzione che si vuole pubblicare;
- **Estensioni:**
  - **UC14.8:** l'utente inserisce una descrizione troppo lunga. Viene di conseguenza visualizzato un messaggio di errore.
- **Precondizione:** l'utente ha digitato il comando *deploy* seguito dai campi *file_path* e *function_name*;
- **Postcondizione:** il campo *brief_desc* contiene una descrizione breve della funzione che si vuole pubblicare.

3.2.46 UC14.4 - Visualizzazione errore: file non trovato

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente inserisce un percorso di file non esistente. Il sistema visualizza un messaggio di errore relativo alla mancata presenza del file;
- **Scenario principale:** viene visualizzato un messaggio di errore relativo alla mancata presenza del file;
- **Precondizione:** l'utente inserisce il percorso di un file non esistente;
- **Postcondizione:** la CLI riporta un messaggio di errore.

3.2.47 UC14.5 - Visualizzazione errore: formato non supportato

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente inserisce un percorso di file di formato non supportato dal servizio. Il sistema visualizza un messaggio di errore relativo al formato errato.
- **Scenario principale:** viene visualizzato un messaggio di errore relativo al formato errato.
- **Estensioni:**
  - **UC:**
- **Precondizione:** l'utente inserisce un percorso di file di formato non supportato;
- **Postcondizione:** viene mostrato un messaggio di errore.

3.2.48 UC14.6 - Visualizzazione errore: nome funzione già esistente

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente inserisce un nome di funzione già presente nel servizio. Il sistema visualizza un messaggio di errore relativo alla presenza del nome specificato;
- **Scenario principale:** viene visualizzato un messaggio di errore relativo alla presenza del nome specificato;
- **Precondizione:** il nome inserito dall'utente è già in uso all'interno del sistema;
- **Postcondizione:** la CLI riporta un messaggio di errore.

3.2.49 UC14.7 - Visualizzazione errore: nome troppo lungo

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente inserisce un nome la cui lunghezza supera la lunghezza massima prevista. Il sistema visualizza un messaggio di errore specificando qual è la lunghezza massima consentita.
- **Scenario principale:** viene visualizzato un messaggio di errore relativo all'eccessiva lunghezza del nome inserito;
- **Precondizione:** l'utente inserisce un nome troppo lungo;
- **Postcondizione:** l'utente visualizza un messaggio riguardante l'errore considerato.

3.2.50 UC14.8 - Visualizzazione errore: descrizione troppo lunga

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente inserisce una descrizione la cui lunghezza supera la lunghezza massima prevista. Il sistema visualizza un messaggio di errore specificando qual è la lunghezza massima consentita;
- **Scenario principale:** viene visualizzato un messaggio di errore relativo all'eccessiva lunghezza della descrizione inserita;
- **Precondizione:** l'utente inserisce una descrizione troppo lunga;
- **Postcondizione:** la CLI riporta un messaggio di errore.

3.2.51 UC15 - Modifica funzione

- **Attori primari:** utente sviluppatore;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede di modificare delle informazioni relative ad una sua funzione tramite il comando: *edit function_name*;
- **Scenario principale:**
  - l'utente richiede la modifica di un'informazione relativa ad una propria funzione;

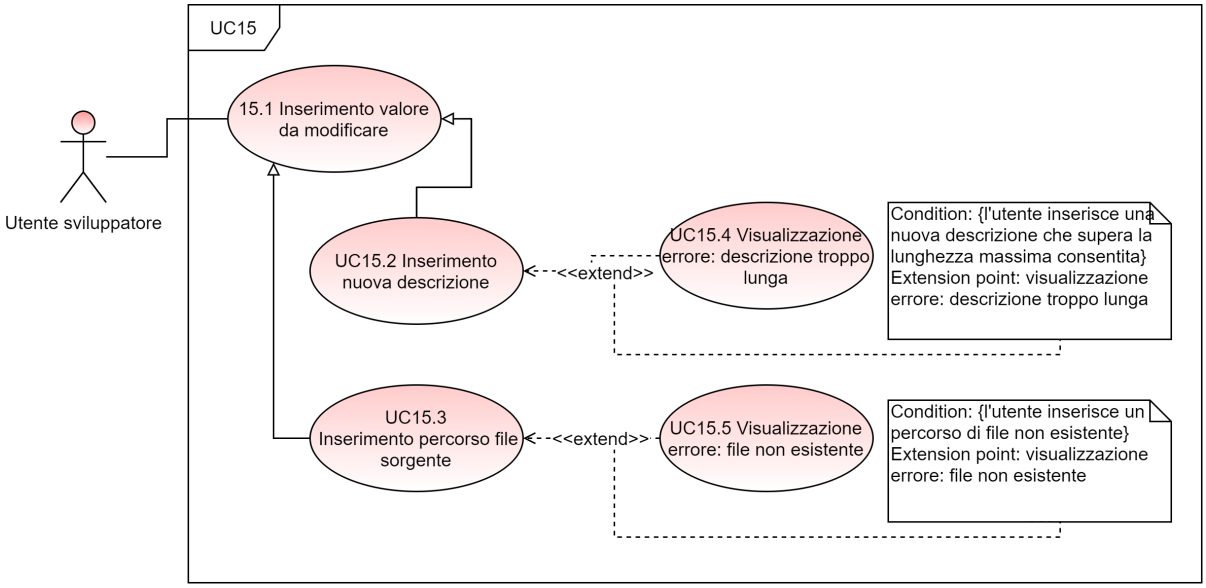

Figura 3.2.23: UC15 - Modifica funzione

– il sistema modifica correttamente tale informazione.

- **Precondizione:** l'utente ha eseguito il deploy di almeno una funzione
- **Postcondizione:** l'utente modifica correttamente la funzione.

3.2.52 UC15.1 - Inserimento valore da modificare

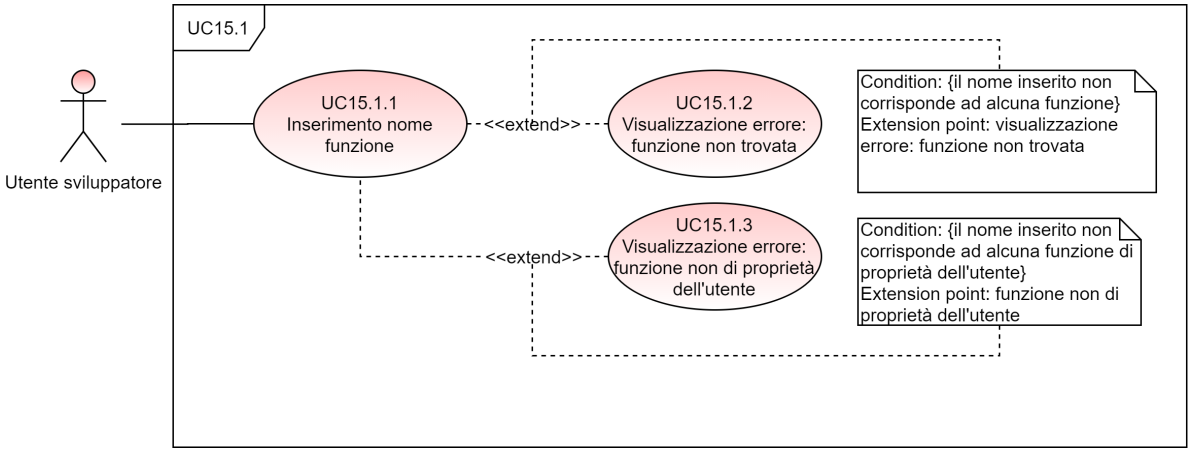

Figura 3.2.24: UC15.1 - Inserimento valore da modificare

- **Attori primari:** utente sviluppatore;

- **Descrizione:** a seguito dell'inserimento del comando *edit* l'utente procede con l'inserimento del nome della funzione considerata e delle modifiche da apportare;
- **Scenario principale:** l'utente inserisce il comando *edit* seguito dal nome della funzione e dalle infrormazioni da aggiornare.
- **Specializzazioni:**
  - **UC15.2:** l'utente vuole modificare la descrizione della funzione considerata;
  - **UC15.3:** l'utente vuole modificare il codice della funzione.
- **Precondizione:** l'utente ha inserito all'interno della CLI il comando *edit*;
- **Postcondizione:** l'utente ha inserito correttamente le nuove informazioni relative alla funzione.

3.2.53 UC15.1.1 - Inserimento nome funzione

- **Attori primari:** utente sviluppatore;
- **Descrizione:** al fine di portare a termine il processo di modifica l'utente deve inserire il nome della funzione da modificare;
- **Scenario principale:** l'utente inserisce il nome della funzione da considerare;
- **Estensioni:**
  - **UC15.1.2:** nel caso in cui il nome inserito non corrisponda ad alcuna funzione viene mostrato un messaggio di errore;
  - **UC15.1.3:** se il nome inserito non corrisponde ad una funzione di proprietà dell'utente, esso viene avvisato tramite un apposito messaggio di errore.
- **Precondizione:** l'utente ha inserito all'interno della CLI il comando *edit*;
- **Postcondizione:** l'utente ha inserito correttamente il nome della funzione considerata.

3.2.54 UC15.1.2 - Visualizzazione errore: funzione non trovata

- **Attori primari:** utente sviluppatore;
- **Descrizione:** dopo aver inserito il nome di una funzione non presente all'interno della piattaforma *Etherless*, l'utente visualizza un messaggio di errore;
- **Scenario principale:** viene mostrato un messaggio di errore che informa l'utente dell'assenza della funzione precedentemente indicata;
- **Precondizione:** il nome inserito dall'utente non corrisponde ad alcuna funzione;
- **Postcondizione:** la CLI riporta un messaggio che descrive l'errore considerato.

3.2.55 UC15.1.3 - Visualizzazione errore: funzione non di proprietà dell'utente

- **Attori primari:** utente sviluppatore;
- **Descrizione:** a seguito dell'inserimento di un nome relativo a una funzione non di proprietà dell'utente, viene visualizzato un errore;
- **Scenario principale:** viene visualizzato a schermo un messaggio di errore che informa l'utente che il nome inserito si riferisce ad una funzione non di sua proprietà;
- **Precondizione:** l'utente ha inserito il nome di una funzione non di sua proprietà;
- **Postcondizione:** viene visualizzato un messaggio di errore.

3.2.56 UC15.2 - Inserimento nuova descrizione

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente inserisce il comando *edit function_name –d new_desc* indicando la volontà di voler modificare la descrizione associata alla funzione tramite il flag *-d*, e inserendo successivamente la nuova descrizione nel campo new_desc; Scenario principale: l'utente inserisce la nuova descrizione;
- **Scenario principale:** l'utente inserisce la nuova descrizione;
- **Estensioni:**
  - **UC15.4:** se l'utente inserisce una nuova descrizione che supera la lunghezza massima consentita, viene visualizzato un apposito messaggio di errore.
- **Precondizione:** l'utente ha inserito all'interno della CLI il comando *edit*;
- **Postcondizione:** l'utente ha inserito correttamente la nuova descrizione.

3.2.57 UC15.3 - Inserimento percorso file sorgente

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente inserisce il comando *edit function_name –c file_path* indicando la volontà di voler modificare il codice associato alla funzione tramite il flag *-c*, e inserendo successivamente il percorso del file sorgente nel campo *file_path*;
- **Scenario principale:** l'utente inserisce il percorso del file contente il codice aggiornato dalla funzione.
- **Estensioni:**
  - **UC15.5:** se l'utente inserisce il percorso di un file non presente viene visualizzato un apposito messaggio di errore.
- **Precondizione:** l'utente ha inserito all'interno della CLI il comando *edit*;
- **Postcondizione:** l'utente ha inserito il percorso del file contenente il codice aggiornato della funzione.

3.2.58 UC15.4 - Visualizzazione errore: descrizione troppo lunga

- **Attori primari:** utente sviluppatore;
- **Descrizione:** dopo aver inserito una descrizione di lunghezza maggiore rispetto a quella massima consentita l'utente visualizza un messaggio di errore;
- **Scenario principale:** viene visualizzato un messaggio di errore che indica l'eccessiva lunghezza della descrizione inserita;
- **Precondizione:** l'utente ha inserito la nuova descrizione della funzione;
- **Postcondizione:** viene mostrato all'utente un errore che descrive l'errore considerato.

3.2.59 UC15.5 - Visualizzazione errore: file non esistente

- **Attori primari:** utente sviluppatore;
- **Descrizione:** a seguito dell'inserimento del percorso di un file non esistente, viene visualizzato un messaggio di errore;
- **Scenario principale:** viene mostrato un messaggio di errore che informa l'utente dell'assenza del file indicato;
- **Precondizione:** l'utente ha inserito il percorso del file contenente il codice aggiornato della funzione considerata;
- **Postcondizione:** la CLI riporta un messaggio di errore.

3.2.60 UC16 - Cronologia di esecuzione

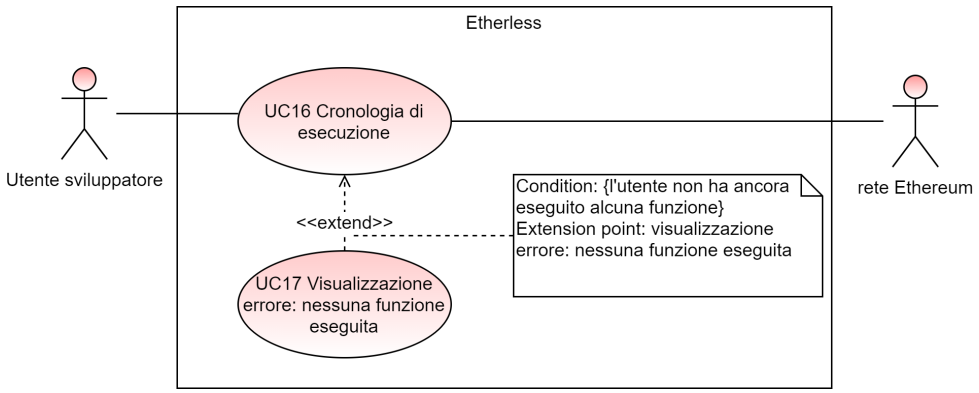

Figura 3.2.25: UC16 - Cronologia di esecuzione: schema generale

- **Attori primari:** utente sviluppatore;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente ottiene le informazioni relative alla propria cronologia di invocazione di funzioni;

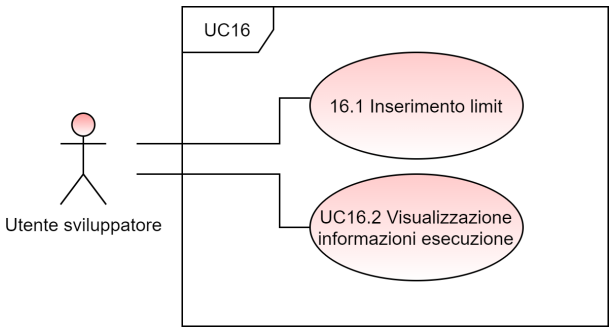

Figura 3.2.26: UC16 - Cronologia di esecuzione

- **Scenario principale:**
  - l'utente inserisce il comando history;
  - vengono visualizzate le informazioni relative alla cronologia delle chiamate dell'utente.
- **Estensioni:**
  - **UC17:** nel caso l'utente non abbia mai eseguito alcuna funzione all'interno della piattaforma viene mostrato un apposito avviso.
- **Precondizione:** l'utente desidera visualizzare la propria cronologia di utilizzo della piattaforma *Etherless*;
- **Postcondizione:** viene visualizzata la cronologia di chiamate dell'utente.

3.2.61 UC16.1 - Inserimento limit

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente può inserire un numero massimo di elementi da visualizzare tramite il flag *-l*;
- **Scenario principale:** l'utente inserisce il comando history seguito dal flag *-l* e il relativo valore;
- **Precondizione:** l'utente ha inserito il comando "history" nella CLI;
- **Postcondizione:** è stato inserito correttamente il numero di massimo di risultati.

3.2.62 UC16.2 - Visualizzazione informazioni esecuzione

- **Attori primari:** utente sviluppatore;
- **Descrizione:** Per ogni elemento della cronologia di esecuzione vengono visualizzati:
  - identificativo della richiesta;
  - nome della funzione richiamata;
  - eventuali parametri passati;
  - risultato dell'esecuzione;

  - data e orario della richiesta.

- **Scenario principale:** vengono visualizzate le informazioni rilevanti della singola esecuzione.
- **Precondizione:** l'utente ha inserito ed eseguito correttamente il comando *history*;
- **Postcondizione:** la CLI contiene le informazioni rilevanti della singola esecuzione;

3.2.63 UC17 - Visualizzazione avviso: nessuna funzione eseguita

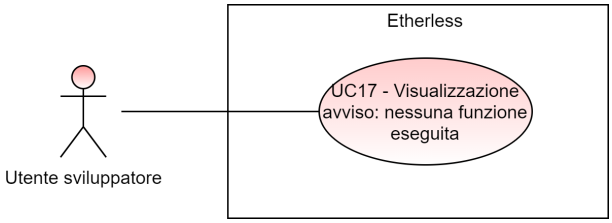

Figura 3.2.27: UC17 - Visualizzazione avviso: nessuna funzione eseguita - schema generale

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente dopo aver eseguito il comando history visualizza un avviso relativo alla mancanza di passate esecuzioni di funzioni all'interno della piattaforma *Etherless*;
- **Scenario principale:**
  - l'utente richiede la visualizzazione della propria cronologia di esecuzione tramite il comando *history*;
  - viene visualizzato un avviso relativo alla mancanza di passate richieste di esecuzione.
- **Precondizione:** l'utente ha richiesto la visualizzazione della propria cronologia di esecuzione di funzioni non avendo prima eseguito alcuna funzione;
- **Postcondizione:** la CLI riporta un avviso che descrive la situazione considerata.

3.2.64 UC18 - Rimozione funzione

- **Attori primari:** utente sviluppatore;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** attraverso l'utilizzo del comando *delete* l'utente può procedere alla rimozione di una funzione;
- **Scenario principale:**
  - l'utente inserisce il comando *delete* seguito dal nome della funzione da rimuovere;
  - la funzione viene rimossa con successo;
- **Precondizione:** l'utente vuole rimuovere una funzione dal sistema;
- **Postcondizione:** la funzione è stata rimossa con successo dal sistema.

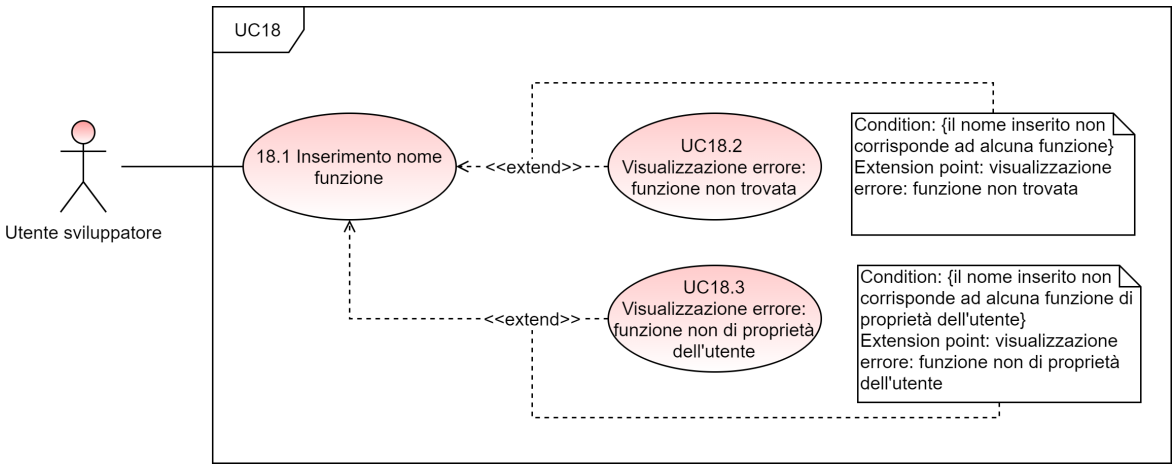

Figura 3.2.28: UC18 - Rimozione funzione

3.2.65 UC18.1 - Inserimento nome funzione

- **Attori primari:** utente sviluppatore;
- **Descrizione:** al fine di eseguire la procedura di rimozione di una funzione è richiesto l'inserimento del relativo nome;
- **Scenario principale:** dopo aver deciso di eliminare una funzione l'utente inserisce nella CLI il relativo nome;
- **Estensioni:**
  - **UC18.2:** se l'utente inserisce un nome che non si riferisce ad alcuna funzione della piattaforma *Etherless* viene mostrato un errore adeguato;
  - **UC18.3:** se viene inserito un nome relativo ad una funzione non appartenente all'utente considerato, viene mostrato un relativo messaggio di errore.
- **Precondizione:** l'utente vuole rimuovere una determinata funzione dal sistema e ha già inserito il comando *delete* nella CLI;
- **Postcondizione:** l'utente ha inserito correttamente il nome della funzione che vuole rimuovere.

3.2.66 UC18.2 - Visualizzazione errore: funzione non trovata

- **Attori primari:** utente sviluppatore;
- **Descrizione:** a seguito del tentativo di eliminazione di una funzione non presente all'interno del sistema, l'utente visualizza un messaggio di errore;
- **Scenario principale:** l'utente viene avvisato dell'assenza della funzione indicata tramite un messaggio di errore;
- **Precondizione:** l'utente ha inserito un nome che non si riferisce ad alcuna funzione presente all'interno del sistema;
- **Postcondizione:** la CLI riporta un messaggio che descrive l'errore considerato.

3.2.67 UC18.3 - Visualizzazione errore: funzione non di proprietà dell'utente

- **Attori primari:** utente sviluppatore;
- **Descrizione:** l'utente tenta di eliminare una funzione che non è di sua proprietà; il sistema rileva tale situazione e mostra un messaggio di errore;
- **Scenario principale:** l'utente viene avvisato che la funzione non è di sua proprietà tramite la visualizzazione di un messaggio di errore;
- **Precondizione:** l'utente ha inserito il nome di una funzione che non è di sua proprietà;
- **Postcondizione:** viene mostrato un messaggio di errore.

3.2.68 UC19 - Visualizzazione avviso: credito insufficiente

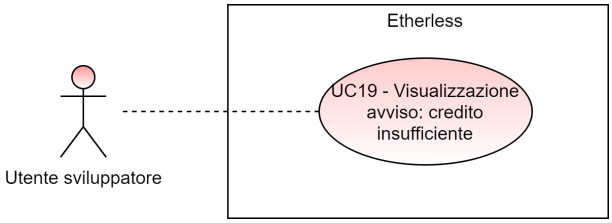

Figura 3.2.29: UC19 - Visualizzazione avviso: credito insufficiente - schema generale

- **Attori primari:** utente autenticato;
- **Attori secondari:** rete Ethereum;
- **Descrizione:** l'utente richiede di eseguire un'operazione non avendo abbastanza credito a disposizione. Il sistema riporta un messaggio di errore relativo alla mancanza di credito;
- **Scenario principale:** viene visualizzato un messaggio di errore relativo alla mancanza di credito necessario per portare a termine l'operazione;
- **Precondizione:** l'utente ha tentato di eseguire un'operazione a pagamento non avendo abbastanza credito a disposizione;
- **Postcondizione:** la CLI riporta un messaggio di errore.

Operazioni sviluppatori

Di seguito sono riportati tutti i casi d'uso aventi come attore primario l'utente sviluppatore.

4 Requisiti

Ogni requisito è composto dai seguenti elementi:

- **Codice identificativo**: ogni codice identificativo è univoco e segue la seguente codifica:

  R[Importanza][Tipologia][Codice]

  Dove:

  - **Importanza**: indica il grado di importanza del requisito ai fini del progetto. Può assumere i valori:
    - **1**: requisito obbligatorio ai fini del progetto, irrinunciabile per gli stakeholders;
    - **2**: requisito desiderabile: non strettamente necessario ai fini del progetto ma che porta valore aggiunto;
    - **3**: requisito opzionale, contrattabile più avanti nel progetto.
  - **Tipologia**: classe a cui appartiene il requisito in questione. Può assumere i valori:
    - **F**: funzionale;
    - **P**: prestazionale;
    - **Q**: qualitativo;
    - **V**: vincolo.
  - **Codice**: identificatore univoco del requisito.

  Il codice stabilito secondo la convenzione precedente, una volta associato ad un requisito, non può più essere modificato.

- **descrizione**: breve descrizione del requisito, strutturata in maniera da evitare ambiguità;
- **classificazione**: indica il grado di importanza del requisito considerato. Sebbene tale informazione sia già presente nell'identificativo, la sua ripetizione rende la lettura più semplice e scorrevole;
- **fonti**:
  - *capitolato*: requisito indicato nel capitolato;
  - *interno*: requisito individuato dagli analisti;
  - *caso d'uso*: il requisito è stato estrapolato da uno o più casi d'uso. In questo caso vengono riportati gli identificativi dei casi d'uso considerati;
  - *verbale*: si tratta di un requisito individuato a seguito di un incontro tra i membri del gruppo o di una richiesta di chiarimento con il proponente. In questo caso è riportato il codice identificativo presente nella tabella delle decisioni dei verbali considerati.

4.1 Requisiti funzionali

Tabella 4.1.1: (continua)

Requisito	Descrizione	Classificazione	Fonti

4.2 Requisiti di qualità

Tabella 4.2.1: (continua)

Requisito	Descrizione	Classificazione	Fonti
R1Q1	La progettazione e la codifica devono rispettare le norme e le metriche definite nei documenti *Norme di Progetto v1.0.0* e *Piano di Qualifica v1.0.0*	Obbligatorio	Interno
R1Q2	Il sistema deve essere pubblicato con licenza MIT	Obbligatorio	Capitolato
R1Q3	Il codice sorgente di *Etherless* deve essere pubblicato e versionato usando Github o GitLab	Obbligatorio	Capitolato
R1Q4	Deve essere redatto un manuale sviluppatore	Obbligatorio	Capitolato
R1Q4.1	Il manuale sviluppatore deve contenere le informazioni per eseguire e fare il deploy dei moduli	Obbligatorio	Capitolato
R1Q5	Deve essere redatto un manuale utente	Obbligatorio	Capitolato
R1Q5.1	Il manuale utente deve contenere tutte le informazioni necessarie all'utente finale per utilizzare correttamente il sistema	Obbligatorio	Capitolato
R1Q6	La documentazione per l'utilizzo del software deve essere scritta in lingua inglese.	Obbligatorio	Verbale 2020-03-18, VE_1.2
R1Q7	Nella scrittura del codice JavaScript deve essere seguita la guida sullo stile di programmazione Airbnb JavaScript style guide	Obbligatorio	Verbale venerdì ?

Tabella 4.2.1: (continua)

Requisito	Descrizione	Classificazione	Fonti
R1Q8	Lo sviluppo del codice JavaScript deve essere supportato dal software di analisi statica del codice ESLint	Obbligatorio	Capitolato

4.3 Requisiti di vincolo

Tabella 4.3.1: (continua)

Requisito	Descrizione	Classificazione	Fonti
R1V1	Gli smart contract devono essere scritti in Solidity	Obbligatorio	Verbale
R1V2	Gli smart contract devono poter essere aggiornati	Obbligatorio	Capitolato
R1V3	L'applicativo deve essere sviluppato utilizzando TypeScript 3.6	Obbligatorio	Capitolato
R1V3.1	Deve essere utilizzato il meccanismo delle promise/async-await come approccio principale	Obbligatorio	Capitolato
R1V4	Il modulo *Etherless-server* deve essere implementato utilizzando il Framework Serverless	Obbligatorio	Capitolato
R1V5	Il progetto deve utilizzare i seguenti ambienti di sviluppo: ambiente di sviluppo locale, ambiente di testing e ambiente di staging	Obbligatorio	Capitolato
R2V5.1	Gli ambienti per la fase di sviluppo locale e testing possono fare utilizzo della rete testrpc fornita dal framework Truffle	Desiderabile	Capitolato
R2V5.2	Per la fase di staging è desiderabile l'utilizzo della rete Ethereum Ropsten	Desiderabile	Capitolato

Tabella 4.3.1: (continua)

Requisito	Descrizione	Classificazione	Fonti
R1V5.3	Durante la fase di staging l'applicativo deve essere pubblicamente accessibile	Obbligatorio	Capitolato
R1V5.4	Al termine del progetto il prodotto deve essere pronto per la produzione	Obbligatorio	Capitolato
R3V5.4.1	L'ambiente di produzione deve fare utilizzo dell'Ethereum main network	Opzionale	Capitolato
R3V6	Il sistema di pagamento deve seguire un meccanismo di escrow	Opzionale	Verbale

4.4 Requisiti funzionali

Non sono stati individuati requisiti prestazionali in quanto la maggior parte delle funzionalità messe a disposizione da *Etherless* necessita dell'interazione con una rete Ethereum. Le operazioni che avvengono all'interno di tale rete hanno un tempo di soddisfacimento che dipende dal carico della rete nel momento in cui viene fatta la richiesta. Nel caso di operazioni che portano ad una modifica nello stato del contratto, deve essere inoltre preso in considerazione anche il quantitativo di Ether pagati per unità di Gas.
Per le motivazioni appena descritte, i tempi di risposta della rete Ethereum e quindi di completamento delle operazioni messe a disposizione da *Etherless*, non risultano essere costanti o prevedibili con precisione.

4.5 Tracciamento

4.5.1 Fonte - Requisiti

4.5.2 Requisito - Fonti

4.6 Considerazioni