SWEmming Pool

swemming.pool@gmail.com

Specifica Tecnica

Informazioni sul documento

Responsabile	Nicolò Trinca
Redattori	Ennio Italiano
	Elia Pasquali
Verificatori	Fabio Pantaleo
Uso	Esterno
Destinatari	Prof. Cardin Riccardo

Sommario

Questo documento contiene le scelte architetturali e di progettazione del gruppo *SWEmming Pool* per il progetto *Trustify*.

Registro delle modifiche

Version	Data	Nominativo	Ruolo	Descrizione
1.0.1	2023-05-10	Nicolò Trinca	*Amministratore*	Verifica modifiche e documento approvato per il rilascio
1.0.0	2023-05-10	Elia Pasquali, Ennio Italiano	*Programmatore*	Modifiche in seguito a colloquio con Prof. Cardin
1.0.0	2023-04-29	Nicolò Trinca	*Amministratore*	Approvato per il rilascio
0.1.0	2023-04-28	Fabio Pantaleo	*Verificatore*	Verifica sezioni §2.4, §3 e documento
0.0.2	2023-04-27	Elia Pasquali, Ennio Italiano	*Programmatore*	Stesura sezioni §2.4, §3
0.0.2	2023-04-26	Fabio Pantaleo	*Verificatore*	Verifica sezioni §1, §2.1, §2.2 - Web app
0.0.1	2023-04-26	Ennio Italiano, Elia Pasquali	*Programmatore*	Stesura sezioni §2.1, §2.2 - Web app
0.0.1	2023-04-24	Ennio Italiano, Elia Pasquali	*Programmatore*	Creata struttura del documento in LaTeXe sezione §1

Contenuti

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Questo documento contiene le scelte architetturali e di progettazione del gruppo *SWEmming Pool* per il progetto *Trustify*. A tale scopo, si riportano i diagrammi delle classi per descrivere architettura e componenti del sistema, oltre ai design pattern utilizzati per la loro realizzazione.

È anche presente una sezione riguardante i requisiti soddisfatti, al fine di fornire una visione dello stato di avanzamento dei lavori.

1.2 Riferimenti

1.2.1 Riferimenti normativi

- Capitolato d'appalto C7
  [Ultimo accesso: 2023-04-29]
- **Norme di Progetto v2.0.0**
- **Analisi dei Requisiti v2.0.0**

1.2.2 Riferimenti informativi

- Materiale didattico IS: Diagrammi delle Classi
  [Ultimo accesso: 2023-04-29];
- Materiale didattico IS: Design Pattern Architetturali - Dependency Injection
  [Ultimo accesso: 2023-04-29];
- Materiale didattico SWE: Model-View Patterns
  [Ultimo accesso: 2023-04-29];

2 Architettura del prodotto

2.1 Architettura logica

Il sistema *Trustify* consiste di uno *smart contract* per gestire i pagamenti e le recensioni, e di una *web app* che permette l'interazione con esso tramite il *wallet MetaMask*. È anche presente un servizio *API REST* che permette agli *e-commerce* interessati di accedere alle recensioni per facilitarne l'integrazione nel loro sito web.

La web app interagisce direttamente con la *blockchain* tramite *Remote Procedure Call* utilizzando un nodo per dialogare con lo smart contract. D'altra parte, il servizio *API REST* è separato dalla logica dell'applicazione e ha come unico scopo quello di fornire un'interfaccia standard e facile da usare per gli *e-commerce*, che permette loro di visualizzare le recensioni sul proprio sito.

2.2 Architettura di deployment

L'architettura di *deployment* del prodotto prevede la suddivisione dei componenti in tre nodi distinti:

- *Server web app*: ospita la *web app* e comunica con lo *smart contract* attraverso l'utilizzo della libreria `web3js`;
- *Server API*: ospita l'*API REST* e comunica con il *smart contract* attraverso l'utilizzo della libreria `web3j`;
- *Nodo Ethereum*: nodo per dialogare direttamente com la blockchain *Ethereum* tramite *Remote Procedure Call*. È utilizzato sia dal *server web app* che dal *server API*. Fornito da *Infura*.

2.3 Dettaglio architettura

2.3.1 Smart contract

Lo *smart contract* è stato sviluppato in *Solidity* e pubblicato su *Sepolia*, una *testnet* di *Ethereum*, per permettere la verifica del suo funzionamento.

2.3.1.1 Diagramma delle classi

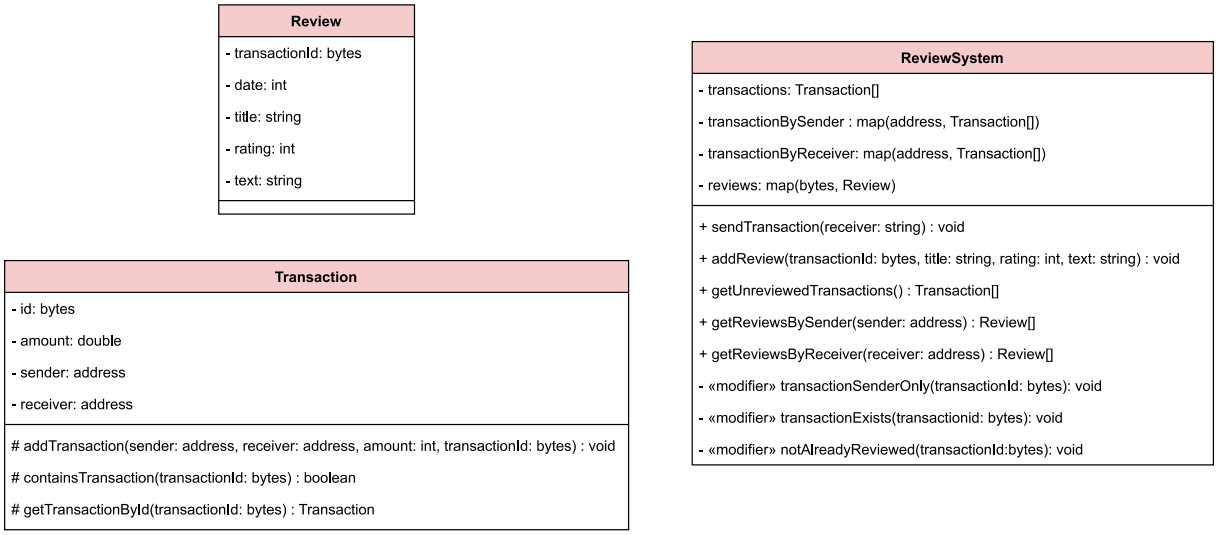


Figura 1: Diagramma delle classi dello *smart contract*

Commento sui `modifier`

In *Solidity* le funzioni possono essere modificate da particolari `modifier`, che permettono di aggiungere delle condizioni che devono essere verificate prima dell'esecuzione della funzione. Questi sono stati inseriti nel diagramma come delle funzioni private con tipo di ritorno nullo.

2.3.1.2 Design patterns

Access Restriction

Limita la chiamata di alcune funzioni ad una ristretta tipologia di utenti.

Viene utilizzato per limitare la possibilità di rilasciare una recensione per una determinata transazione solo all'effettivo cliente che l'ha effettuata. Avviene tramite `modifier` del linguaggio, particolari funzioni che permettono di bloccare l'esecuzione della funzione a cui vengono legati e annullare l'operazione se non viene rispettata la condizione.

Guard check

Valida i parametri passati ad una funzione, annullandola se non rispettano determinate condizioni.

Tramite `require`, un meccanismo di validazione offerto dal linguaggio, viene verificato che i parametri passati ad una funzione siano validi e che i vari attributi rispettino i vincoli di dominio imposti (ad esempio, che il voto sia compreso tra 1 e 5). Inoltre viene utilizzato per verificare che la transazione che si sta cercando di recensire sia effettivamente presente nel sistema e che non sia già stata recensita.

2.3.2 Web app

2.3.2.1 Diagramma delle classi

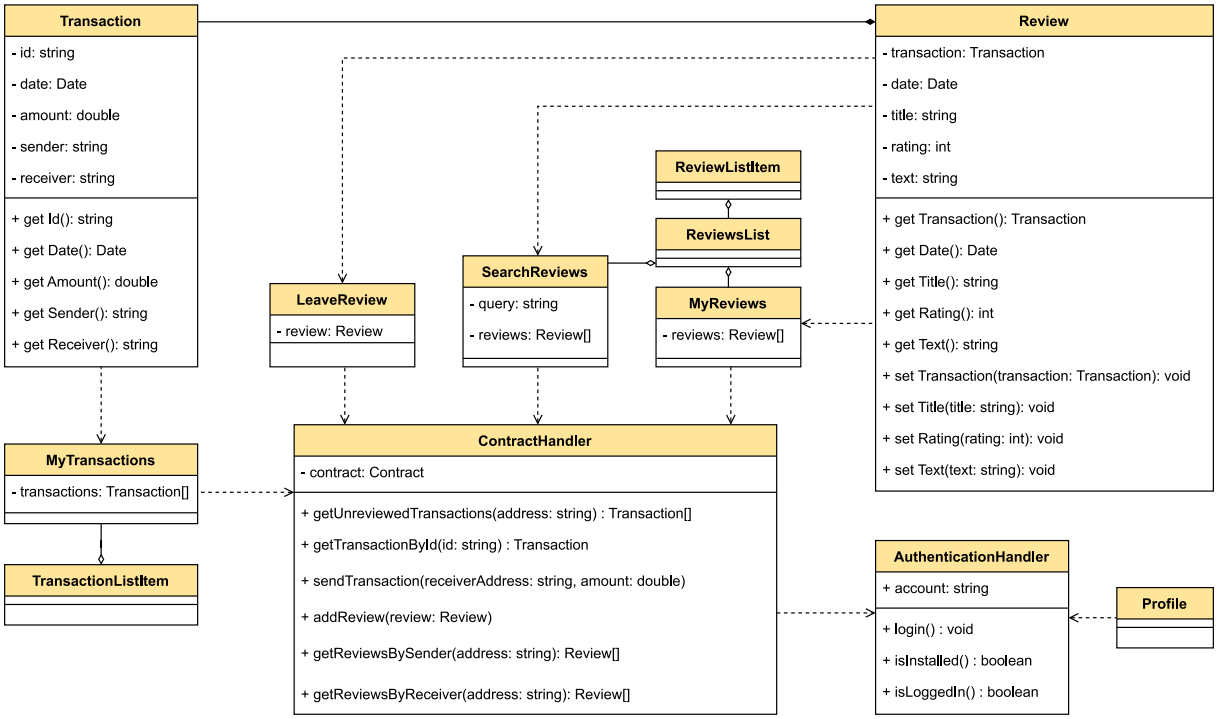


Figura 2: Diagramma delle classi della *web app*

2.3.2.2 Design patterns

Per la *web app* si utilizzano i pattern *MVVM* (*Model-View-ViewModel*) e *Dependency Injection*, per garantire un alto livello di separazione delle responsabilità e una maggiore modularità. Tali pattern costituiscono caratteristiche intrinseche di *Angular*, il *framework* utilizzato per lo sviluppo della *web app*.

MVVM (Model-View-ViewModel)

Il pattern *MVVM* divide il sistema in tre parti principali: il *Model*, la *View* e il *ViewModel*. All'interno della nostra *web app*:

- il *Model* è costituito dalle rappresentazioni di recensioni e transazioni (`Review` e `Transaction`), e dai servizi che gestiscono l'autenticazione e le interazioni con lo *smart contract* (`Contract Handler` e `AuthenticationHandler`);
- la *View* viene definita utilizzando i *template HTML*, che possono essere usati per definire la struttura della pagina e visualizzare i dati dal modello;
- il *ViewModel* viene rappresentato dalle classi *TypeScript* utilizzate per gestire gli eventi della vista e aggiornare il modello di conseguenza. Un'esempio di *ViewModel* nella *web app* è costituito dalla classe `LeaveReview`, che gestisce l'invio di una recensione al sistema in seguito all'inserimento dei dati da parte dell'utente.

Dependency Injection

In *Angular*, il pattern *Dependency Injection* viene applicato per fornire un meccanismo di interazione tra i consumatori e i fornitori di dipendenze mediante un'astrazione chiamata *Injector*. In pratica, un *injector* viene utilizzato per cercare se esiste già un'istanza disponibile di una

determinata dipendenza richiesta; se l'istanza non è già presente, viene creata e salvata. Un esempio di utilizzo di *Dependency Injection* si ha nel nostro caso con i servizi `ContractHandler` e `AuthenticationHandler`, che vengono iniettati nei componenti che li necessitano per gestire l'autenticazione e le interazioni con lo *smart contract*.

2.3.3 API REST

2.3.3.1 Diagramma delle classi

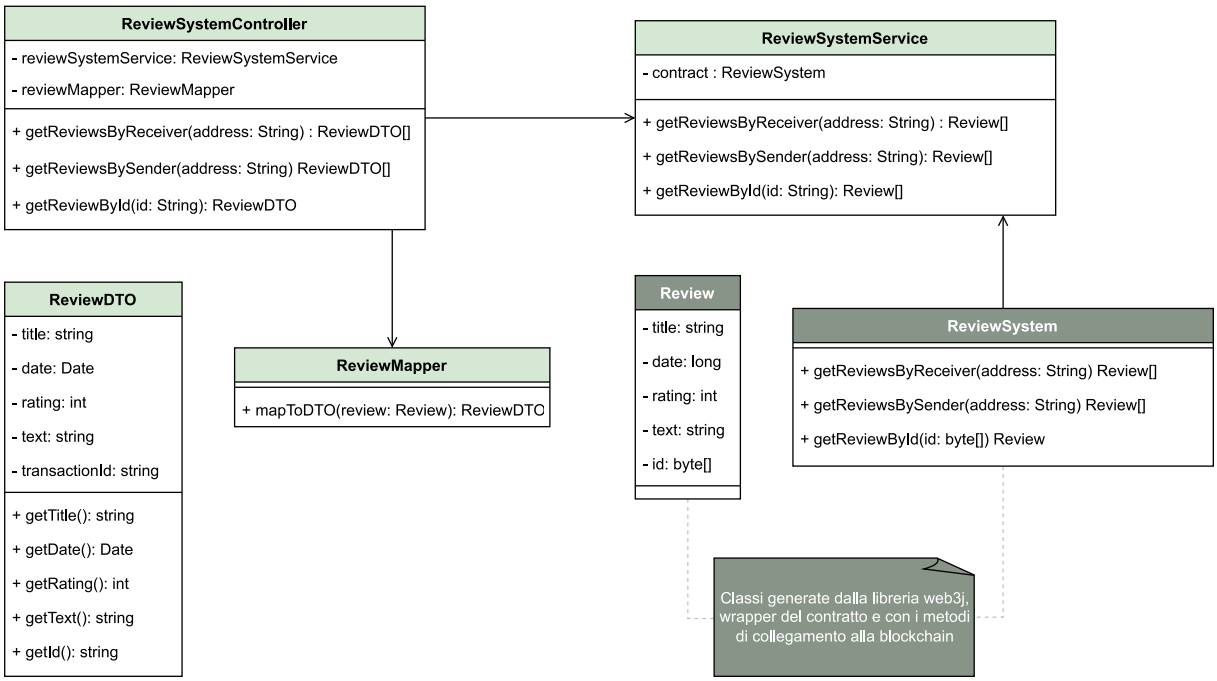


Figura 3: Diagramma delle classi delle *API REST*

2.3.3.2 Design patterns

Dependency Injection

In *Spring*, la *Dependency Injection* è implementata attraverso l'uso di un container di inversione di controllo (IoC), che si occupa di gestire la creazione e l'inizializzazione dei componenti dell'applicazione e di fornire le dipendenze tra essi. In particolare, *Spring* utilizza l'annotazione `@Autowired` per iniettare automaticamente le dipendenze dei componenti, come ad esempio i servizi che gestiscono l'accesso ai dati, all'interno dei *Controller*.

In pratica, questo significa che è possibile definire un *Controller* che dipende da un servizio di accesso ai dati, e automaticamente *Spring* si occuperà di creare e iniettare l'istanza del servizio all'interno di esso, senza che sia necessario gestire manualmente la creazione e l'inizializzazione delle dipendenze.

DTO (Data Transfer Object)

Il *DTO pattern (Data Transfer Object)* è un oggetto che viene utilizzato per semplificare la comunicazione con l'applicazione; contiene solo i dati necessari per il trasferimento e non ha alcuna logica o comportamento associato ad esso. La sua funzione principale è quella di incapsulare i dati in modo da trasferirli in maniera più efficiente.

Nelle nostre *API* viene utilizzato per ottenere un oggetto `ReviewDTO` che permetta una gestione semplificata delle recensioni.

3 Requisiti soddisfatti

Codice	Tipo	Descrizione	Stato
R1F1	Obbligatorio	La *web app* fornisce all'utente non autenticato la possibilità effettuare il login tramite *MetaMask*.	Soddisfatto
R1F1.1	Obbligatorio	La *web app* visualizza un errore in caso di autenticazione fallita.	Soddisfatto
R1F1.2	Obbligatorio	L'utente non autenticato visualizza un messaggio di errore in caso di assenza di *MetaMask* installato.	Soddisfatto
R1F2	Obbligatorio	L'utente autenticato deve poter eseguire un pagamento tramite *MetaMask* verso un altro indirizzo wallet.	Soddisfatto
R1F2.1	Obbligatorio	Il pagamento fallisce a causa di un errore e il sistema notifica l'utente.	Soddisfatto
R1F3	Obbligatorio	L'utente autenticato può visualizzare una lista dei pagamenti effettuati, senza una recensione, tra cui scegliere quale recensire.	Soddisfatto
R1F4	Obbligatorio	L'utente autenticato ha la possibilità di rilasciare una recensione nel caso abbia acquisti non ancora recensiti.	Soddisfatto
R1F4.1	Obbligatorio	Per ogni transazione effettuata può esistere al massimo una recensione ad esso collegata.	Soddisfatto
R1F4.2	Obbligatorio	L'utente deve scegliere la transazione, effettuata, da associare alla recensione.	Soddisfatto
R1F4.3	Obbligatorio	La recensione deve avere un titolo inserito dall'utente.	Soddisfatto
R1F4.4	Obbligatorio	La recensione deve avere un voto inserito dall'utente.	Soddisfatto
R1F4.5	Obbligatorio	La recensione deve avere un testo inserito dall'utente.	Soddisfatto
R1F4.6	Obbligatorio	La *web app* visualizza un errore se l'utente non rispetta la validità del formato di una recensione.	Soddisfatto
R1F4.6.1	Obbligatorio	La recensione deve avere un titolo, non superiore ai 50 caratteri.	Soddisfatto

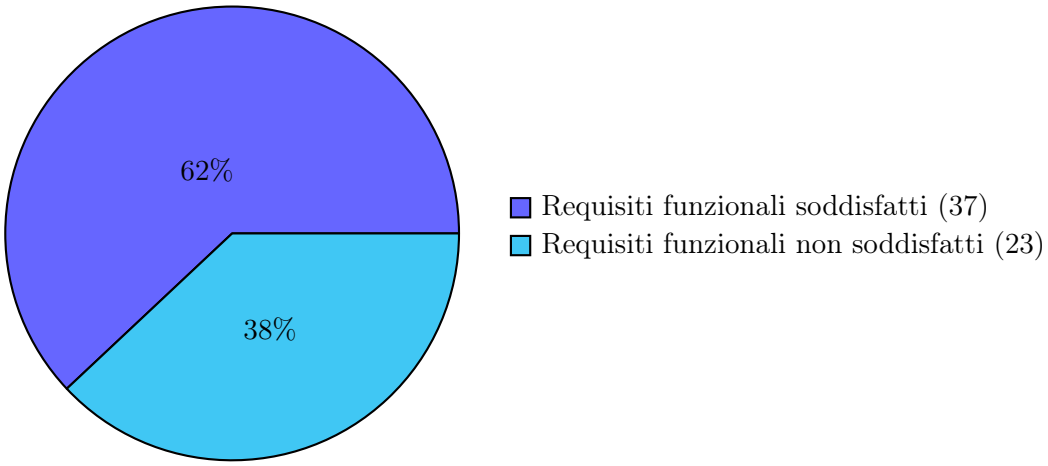
Codice	Tipo	Descrizione	Stato
R1F4.6.2	Obbligatorio	La recensione deve esprimere un voto, compreso tra 1(minimo) e 5 (massimo).	Soddisfatto
R1F4.6.3	Obbligatorio	La recensione deve avere un testo, di al massimo 500 caratteri.	Soddisfatto
R1F5	Obbligatorio	L'utente generico può ricercare recensioni.	Soddisfatto
R1F5.1	Obbligatorio	L'utente generico può ricercare recensioni per indirizzo wallet.	Soddisfatto
R1F5.1.1	Obbligatorio	L'utente generico può ricercare recensioni per indirizzo wallet appartenente all'autore.	Soddisfatto
R1F5.1.2	Obbligatorio	L'utente generico può ricercare recensioni per indirizzo wallet appartenente al destinatario.	Soddisfatto
R3F5.2	Opzionale	L'utente generico può ricercare recensioni per voto.	Non soddisfatto
R3F5.3	Opzionale	L'utente generico può ricercare recensioni per titolo.	Non soddisfatto
R3F5.4	Opzionale	L'utente generico può ricercare recensioni per data.	Non soddisfatto
R1F6	Obbligatorio	La *web app* fornisce all'utente autenticato la possibilità di ricercare le recensioni rilasciate.	Soddisfatto
R1F7	Obbligatorio	La *web app* fornisce all'utente autenticato la possibilità di ricercare le recensioni ricevute.	Soddisfatto
R1F8	Obbligatorio	La *web app* notifica all'utente un errore nella ricerca.	Soddisfatto
R1F8.1	Obbligatorio	La *web app* notifica all'utente un errore nella ricerca per indirizzo wallet.	Soddisfatto
R3F8.2	Opzionale	La *web app* notifica all'utente un errore nella ricerca per voto.	Non soddisfatto
R3F8.3	Opzionale	La *web app* notifica all'utente un errore nella ricerca per titolo.	Non soddisfatto
R3F8.4	Opzionale	La *web app* notifica all'utente un errore nella ricerca per data.	Non soddisfatto

Codice	Tipo	Descrizione	Stato
R1F9	Obbligatorio	La *web app* fornisce all'utente generico la possibilità di visualizzare una lista di recensioni.	Soddisfatto
R1F10	Obbligatorio	La *web app* fornisce all'utente generico la possibilità di visualizzare una singola recensione.	Soddisfatto
R1F10.1	Obbligatorio	La *web app* fornisce all'utente generico la possibilità di visualizzare l'autore di una singola recensione.	Soddisfatto
R1F10.2	Obbligatorio	La *web app* fornisce all'utente generico la possibilità di visualizzare il destinatario di una singola recensione.	Soddisfatto
R1F10.3	Obbligatorio	La *web app* fornisce all'utente generico la possibilità di visualizzare il titolo di una singola recensione.	Soddisfatto
R1F10.4	Obbligatorio	La *web app* fornisce all'utente generico la possibilità di visualizzare la data di una singola recensione.	Soddisfatto
R1F10.5	Obbligatorio	La *web app* fornisce all'utente generico la possibilità di visualizzare il voto di una singola recensione.	Soddisfatto
R1F10.6	Obbligatorio	La *web app* fornisce all'utente generico la possibilità di visualizzare il testo di una singola recensione.	Soddisfatto
R1F11	Obbligatorio	La *web app* fornisce all'utente autenticato la possibilità di visualizzare le recensioni rilasciate.	Soddisfatto
R1F12	Obbligatorio	La *web app* fornisce all'utente autenticato la possibilità di visualizzare le recensioni ricevute.	Soddisfatto
R1F13	Obbligatorio	La *web app* notifica all'utente un errore nella visualizzazione delle recensioni.	Soddisfatto
R1F14	Obbligatorio	Il servizio di *API REST* fornisce all'utente, utilizzatore dell'API, la possibilità di ottenere le recensioni ad esso riferite.	Soddisfatto
R1F15	Obbligatorio	Il servizio di *API REST* notifica all'utente un errore, nel caso non esistano recensioni a lui collegate.	Soddisfatto

Codice	Tipo	Descrizione	Stato
R3F16	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare la lista di tutti i pagamenti effettuati.	Non soddisfatto
R3F17	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare una singola transazione, selezionata dalla lista.	Non soddisfatto
R3F17.1	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare l'ID di una singola transazione.	Non soddisfatto
R3F17.2	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare la data di una singola transazione.	Non soddisfatto
R3F17.3	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare l'importo di una singola transazione.	Non soddisfatto
R3F17.4	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare l'utente pagante di una singola transazione.	Non soddisfatto
R3F17.5	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare l'utente ricevente di una singola transazione.	Non soddisfatto
R3F18	Opzionale	La *web app* notifica all'utente un errore nella visualizzazione dei pagamenti.	Non soddisfatto
R3F19	Opzionale	La *web app* fornisce all'utente generico la possibilità di visualizzare una lista di recensioni, ordinate.	Non soddisfatto
R3F19.1	Opzionale	La *web app* fornisce all'utente generico la possibilità di visualizzare una lista di recensioni, ordinate dal meno recente.	Non soddisfatto
R3F19.2	Opzionale	La *web app* fornisce all'utente generico la possibilità di visualizzare una lista di recensioni, ordinate dal più recente.	Non soddisfatto
R3F20	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare la lista di tutti i pagamenti effettuati in ordine in base alla data del pagamento.	Non soddisfatto

Codice	Tipo	Descrizione	Stato
R3F20.1	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare la lista di tutti i pagamenti effettuati, ordinandoli dal meno recente.	Non soddisfatto
R3F20.2	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare la lista di tutti i pagamenti effettuati, ordinandoli dal più recente.	Non soddisfatto
R3F21	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare la lista di tutti i pagamenti effettuati in ordine in base all'importo.	Non soddisfatto
R3F21.1	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare la lista di tutti i pagamenti effettuati, ordinandoli per importo più economico.	Non soddisfatto
R3F21.2	Opzionale	La *web app* fornisce all'utente autenticato la possibilità di visualizzare la lista di tutti i pagamenti effettuati, ordinandoli per importo meno economico.	Non soddisfatto

Tabella 1: Requisiti funzionali e relativo stato



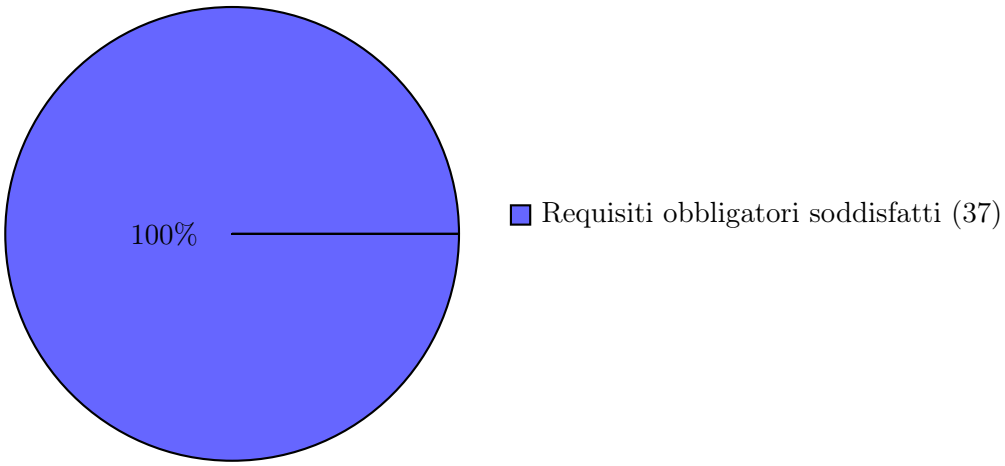


Figura 4: Grafici sullo stato dei requisiti funzionali