Winning Software Solution
winningsoftwaresolution@gmail.com

ShopChain
SyncLab

Analisi dei requisiti

Informazioni	
Redattori	Giovanni Cocco
Revisori	Federico Marchi
Responsabili	Giovanni Cocco
Versione	0.0.1
Uso	esterno

Descrizione

Architettura del progetto

Versione	Data	Persona	Attivtà	Descrizione
0.0.1	03/3/2022	Raffaele Oliviero	Redazione	Aggiornamento
0.0.0	23/2/2022	Giovanni Cocco	Redazione	Strutturazione del documento

1 Introduzone

1.1 Scopo del documento

Il documento illustra le scelte architetturali e illustra l'architettura tramite diagrammi delle classi e di sequenza.

2 Architettura generale

Il progetto si compone di 4 macro parti:

- Server
- Smart contract
- Web app
- Script di messa in vendita

2.1 Server

Realizzato in typescript con express come modulo http e MariaDB come database SQL. Si divide in 2 parti principali: la persistenza e il server web.

2.1.1 Persistenza

Si occupa di gestire i dati delle transazioni.

Si collega allo smart contract attraverso un websocket fornito da moralis.io.
Rimane in ascolto degli eventi dello smart contract e aggiorna il database SQL di conseguenza.
Tiene sempre traccia dell'ultimo blocco da cui ha ricevuto un evento e all'avvio recupera tutti gli eventi arretrati partendo da quest'ultimo blocco.

I dati nel database SQL vengono forniti al frontend.

Notare come è molto oneroso effettuare query sui dati in block chain in quanto non sono disponibili strutture dati adeguate.
Questa soluzione ci permette una maggiore flessibilità e apertura a modifiche future quali aggiungere query specifiche.

Inoltre il contratto una volta publicato non può essere modificabile al fine di garantire la trasparenza ed è quindi cruciale che il codice di quest'ultimo sia semplice ed affidabile.

2.1.2 Server Web

Riceve le richieste HTTP dalla rete e risponde con le pagine della Web app.

2.2 Smart contract

Realizzato in solidity e publicato su una rete Polygon tiene traccia delle transazioni; gestisce la logica e la sicurezza di esse.

Per gestire il timer che fa scadere le transazioni si usa il servizio Upkeeper di Chain-Link.
Uno smart contract non esegue operazioni se non viene chiamato, per realizzare un timer si realizzano delle funzioni che se è passato abbastanza tempo eseguono le operazioni. Upkeeper chiama automaticamente queste funzioni dall'esterno a intervalli di tempo prefissato.

2.3 Web app

Fornita all'utente tramite il server web fornisce la logica lato client.
Tramite MetaMask l'utente interagisce direttamente col contratto.

In caso di utenti mobile reindirizza tramite DeepLink per aprire la pagine sull'app di Metamask.

I deep link vengono usati anche per il QR code di ricezione del pacco.
Possono essere scansionati sia da dentro l'app di MetaMask che dalla fotocamera del cellulare.

2.4 Script di messa in vendita

Usato dall'ecommerce per inserire prodotti in vendita in blockchain al fine di verificare che il prezzo sia corretto al momento della vendita. Realizzato in python per flessibilità, si connette alla block chain tramite moralis.io.

3 Design pattern

Nonostante le tecnologie esoteriche del capitolato non si prestino a molti design pattern ne sono stati comunque utilizzati diversi.

3.1 Constructor injection

Molto usato nell'architettura del server permette di tener traccia delle dipendenze e rende più facile la realizzazione dei mock per i test di unità

3.2 Builder

La creazione del server è molto articolata. Richieda sia la classe Sever che la classe ShopContractEventManger. Queste due classi hanno alcuni argomenti di costruzione comuni. Per semplificare la costruzione e assicurare che sia tutto configurato la classe ServerManager si comporta come un builder. L'unica differenze è che al posto di un metodo build v'è il metodo start.

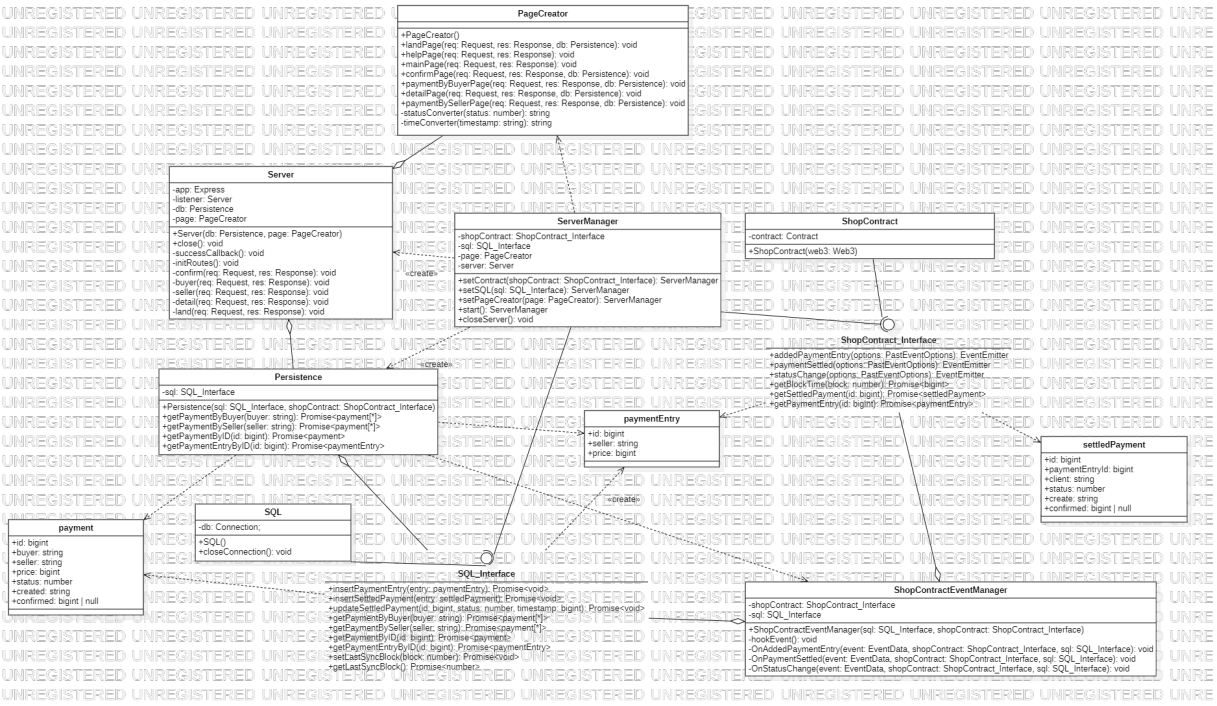
3.3 Singleton

L'utilizzo del pattern del singleton è stata inizialmente presa in considerazione, ma poi scartata in quanto complicava la fase di testing e mocking.

4 Diagramma delle classi

Notare che in solidity i metodi e le funzioni possono avere come visibilità **external**. Non essendo disponibile nello standard UML si userà la notazione **+{external}** per indicarla.

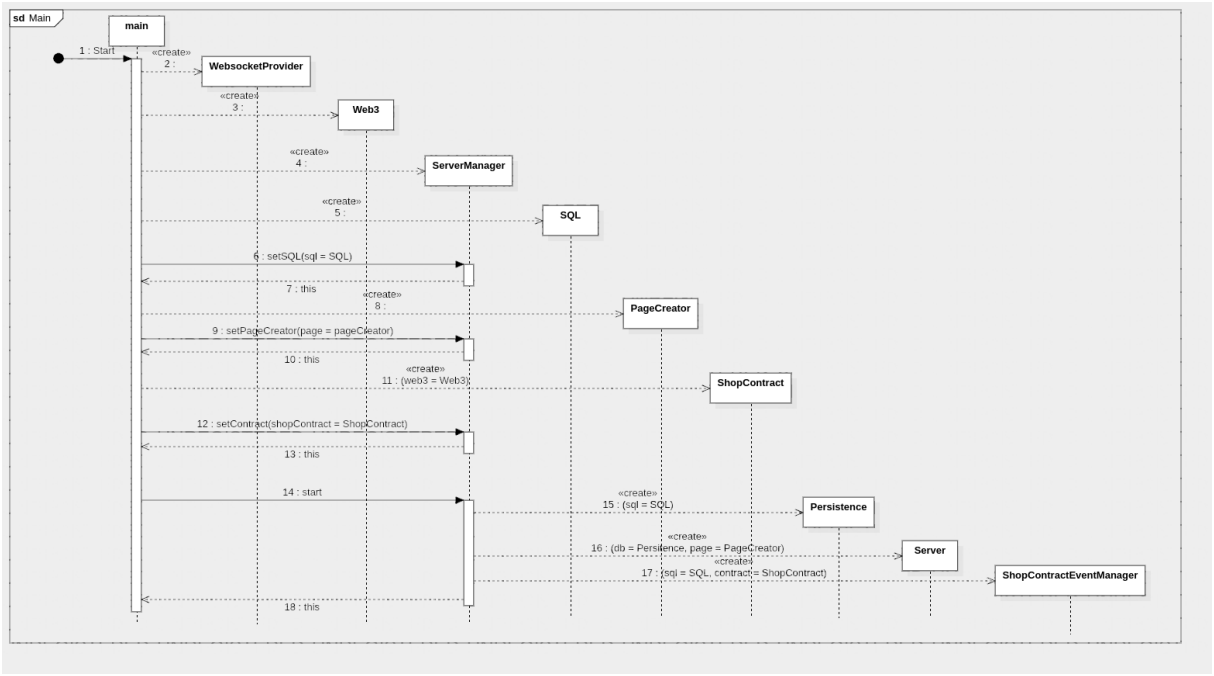
Server



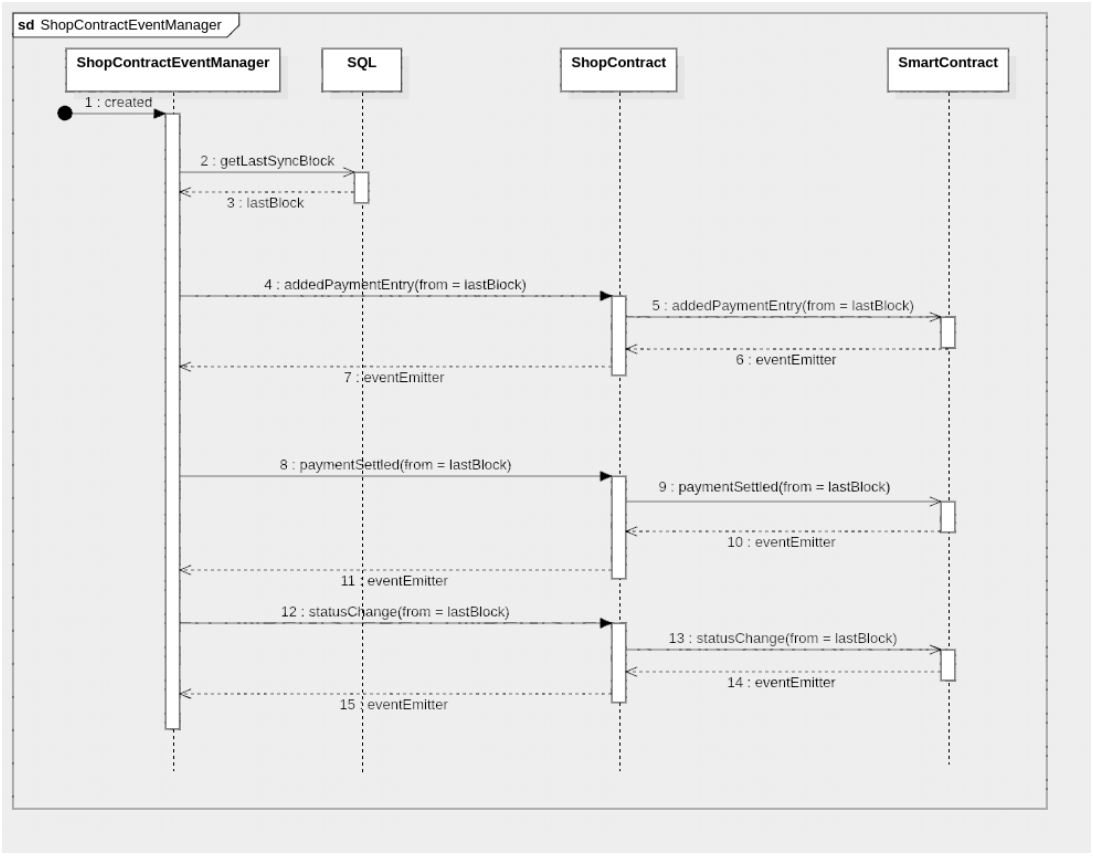
ShopContract

5 Diagrammi di sequenza

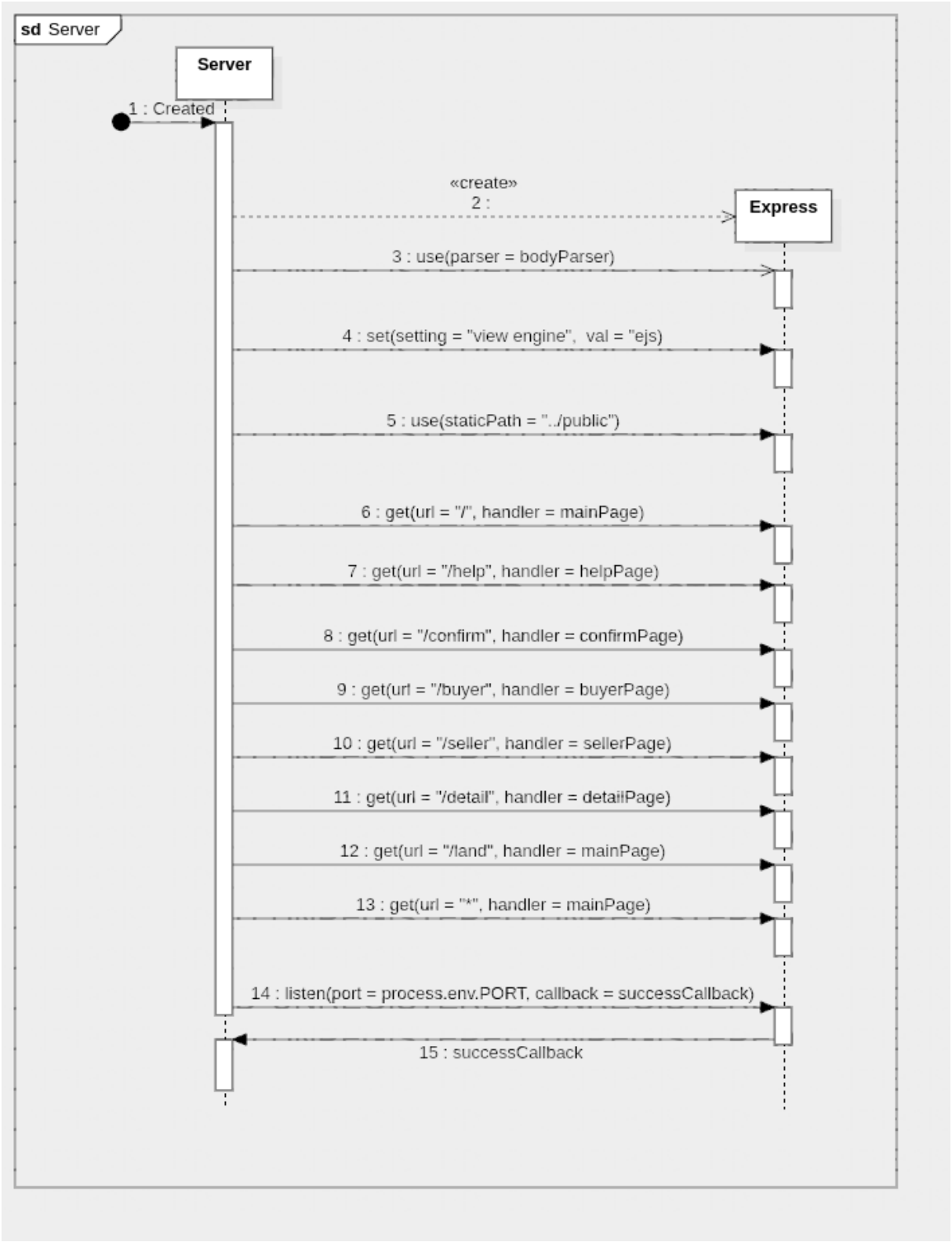
Avvio server



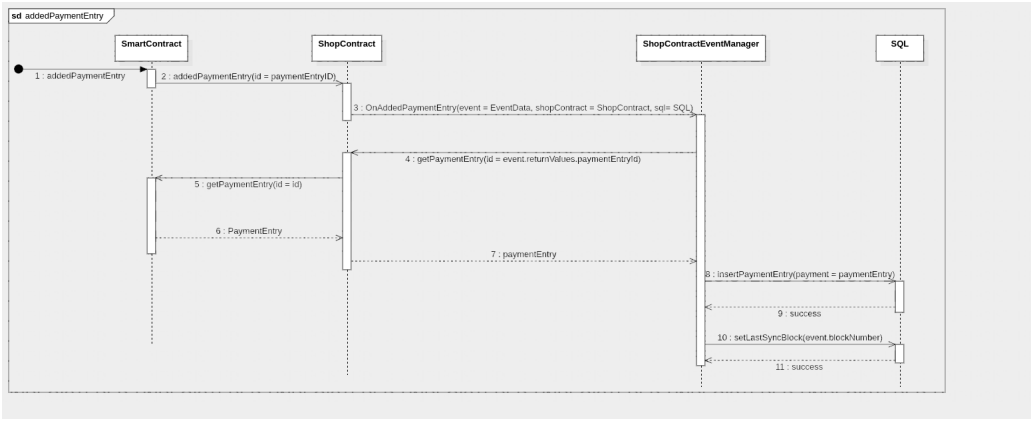
Ascolto eventi del contratto



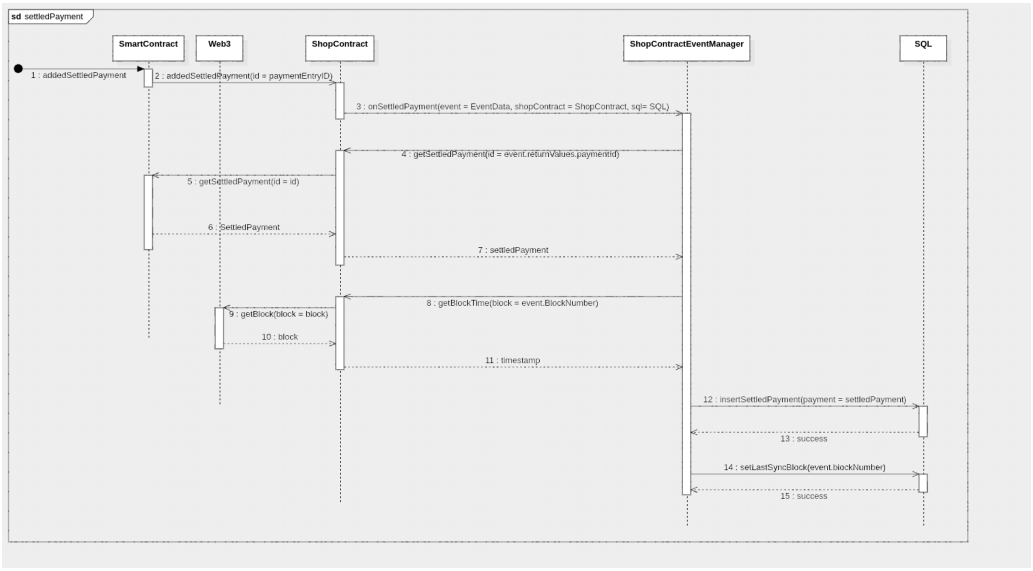
Inizializzazione server web



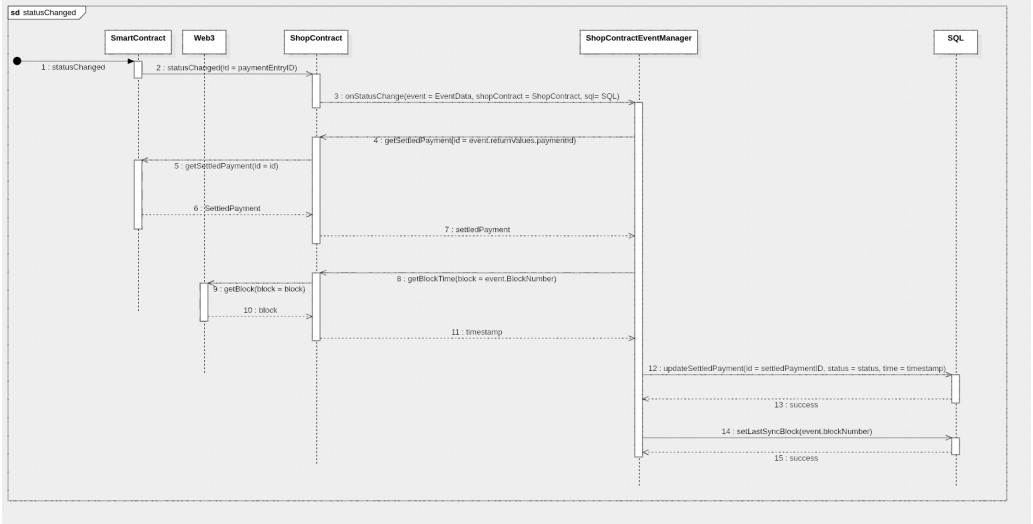
Nuovo oggetto in vendita



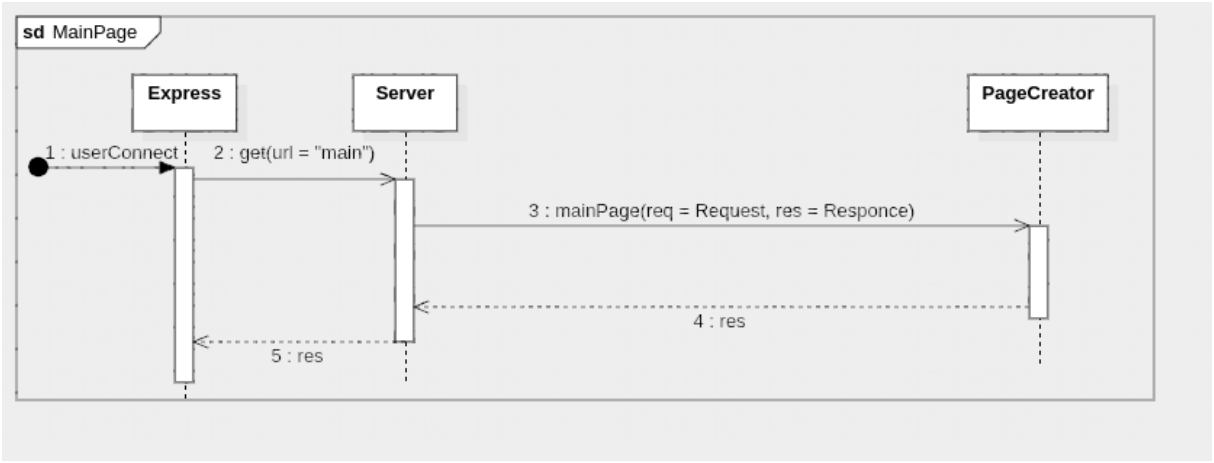
Nuova transazione



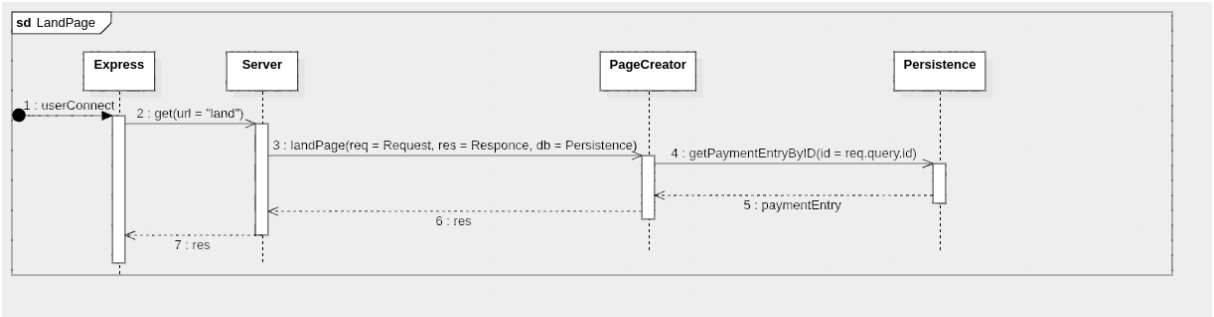
Cambio di stato di una transazione



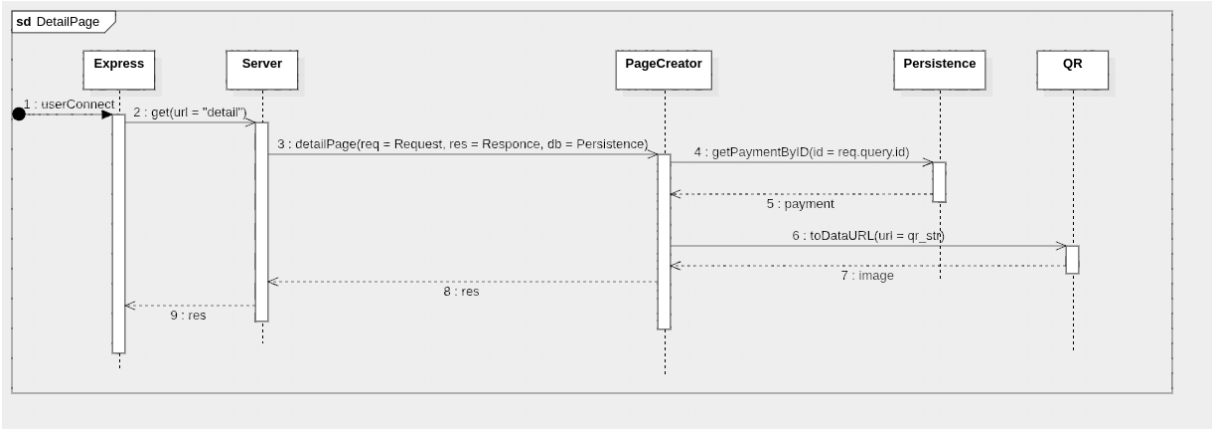
Home page



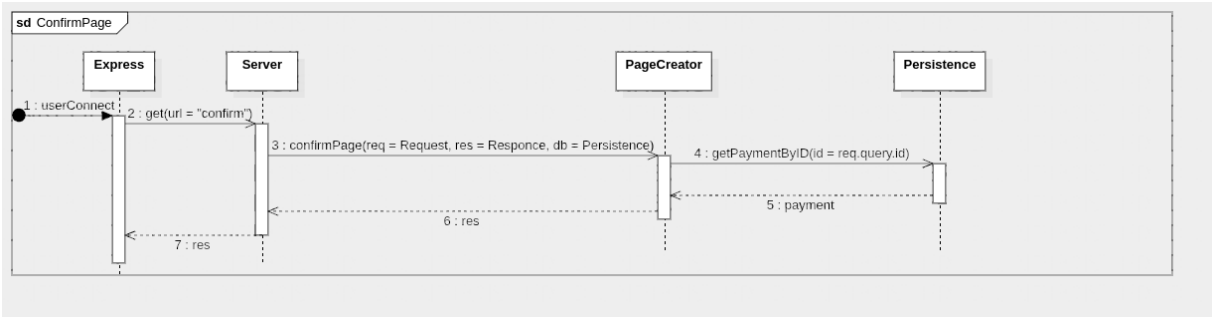
Landing page di vendita



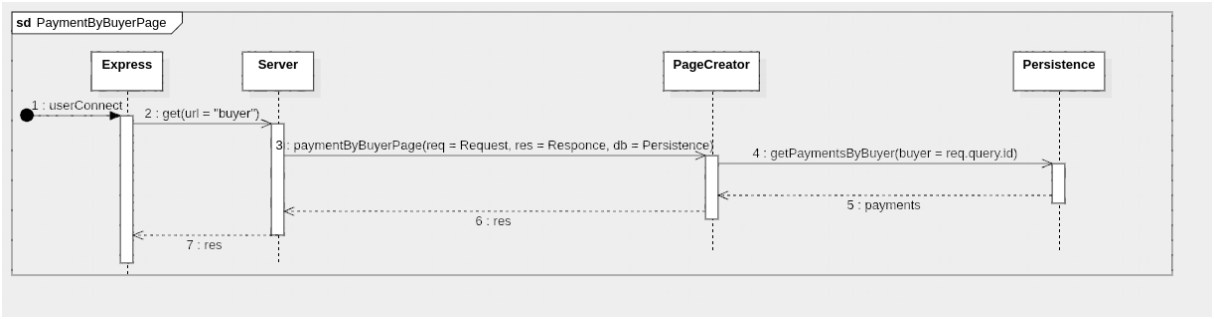
Dettagli transazione



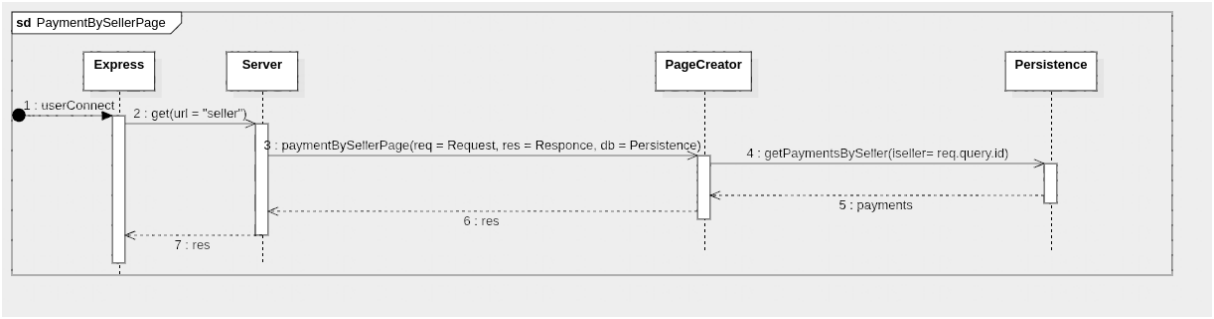
Pagina di sblocco fondi (arrivo del pacco)



Lista oggetti venduti



Lista oggetti comprati



Manuale utente