



Winning Software Solution

winningssoftwaresolution@gmail.com

ShopChain

SyncLab

## Specifiche Architettureali

### *Informazioni*

<i>Redattori</i>	Giovanni Cocco
<i>Revisori</i>	Federico Marchi
<i>Responsabili</i>	Giovanni Cocco
<i>Versione</i>	1.0.0
<i>Uso</i>	esterno

### Descrizione

---

Architettura del progetto

<b>Versione</b>	<b>Data</b>	<b>Persona</b>	<b>Attività</b>	<b>Descrizione</b>
1.0.0	23/2/2022	Giovanni Cocco	Redazione	Creazione del documento

# Contents

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Scopo del documento . . . . .	3
<b>2</b>	<b>Riferimenti</b>	<b>3</b>
2.1	Riferimenti normativi . . . . .	3
2.2	Riferimenti informativi . . . . .	3
<b>3</b>	<b>Tecnologie/Linguaggi/Librerie</b>	<b>3</b>
3.1	Solidity . . . . .	3
3.2	Typescript . . . . .	3
3.3	Express . . . . .	3
3.4	MariaDB . . . . .	3
3.5	Python . . . . .	3
3.6	Web3 . . . . .	4
3.7	MetaMask . . . . .	4
<b>4</b>	<b>Architettura</b>	<b>4</b>
4.1	Server . . . . .	4
4.1.1	Diagramma delle classi . . . . .	4
4.1.2	Design pattern: Constructor injection . . . . .	5
4.1.3	Schema DB . . . . .	5
4.2	Smart contract . . . . .	5
4.2.1	Diagramma delle classi . . . . .	5
4.2.2	Pattern architetturale adottato . . . . .	6
4.3	Web app . . . . .	6
4.3.1	Diagramma delle classi . . . . .	6
4.3.2	Pattern architetturale adottato . . . . .	6
4.4	Script di messa in vendita . . . . .	6
4.4.1	sell_item . . . . .	6
<b>5</b>	<b>Diagrammi di sequenza</b>	<b>7</b>
5.1	Inizializzazione server web . . . . .	7
5.2	Ascolto eventi del contratto . . . . .	8
5.3	Nuovo oggetto in vendita . . . . .	9
5.4	Nuova transazione . . . . .	9
5.5	Cambio di stato di una transazione . . . . .	10
5.6	Pagina transazioni in entrata . . . . .	10

# 1 Introduzione

## 1.1 Scopo del documento

Il documento illustra le scelte architettureali e illustra l'architettura.

# 2 Riferimenti

## 2.1 Riferimenti normativi

- Capitolato d'appalto C2;
- Norme di Progetto;
- Verbale esterno 2021/03/01.

## 2.2 Riferimenti informativi

- Progettazione Software - Materiale didattico del corso IS;
- Slide diagrammi di sequenza - Materiale didattico del corso IS;
- Slide design pattern architettureali - Materiale didattico del corso IS;
- Slide diagrammi delle classi - Materiale didattico del corso IS;
- Slide principi SOLID - Materiale didattico del corso IS.

# 3 Tecnologie/Linguaggi/Librerie

## 3.1 Solidity

- **Versione:** 0.8.13
- **Documentazione:** <https://docs.soliditylang.org/en/v0.8.13/>

## 3.2 Typescript

- **Versione:** 4.6.3
- **Documentazione:** <https://www.typescriptlang.org/docs/>

## 3.3 Express

- **Versione:** 4.17.2
- **Documentazione:** <https://devdocs.io/express/>

## 3.4 MariaDB

- **Versione:** 10.7.3
- **Documentazione:** <https://mariadb.com/kb/en/documentation/>

## 3.5 Python

- **Versione:** 3.8
- **Documentazione:** <https://docs.python.org/3.8/>

## 3.6 Web3

- **Versione:** 1.7.1
- **Documentazione:** <https://web3js.readthedocs.io/en/v1.7.1/>

## 3.7 MetaMask

- **Versione:** 10.11.3
- **Documentazione:** <https://docs.metamask.io/guide/>

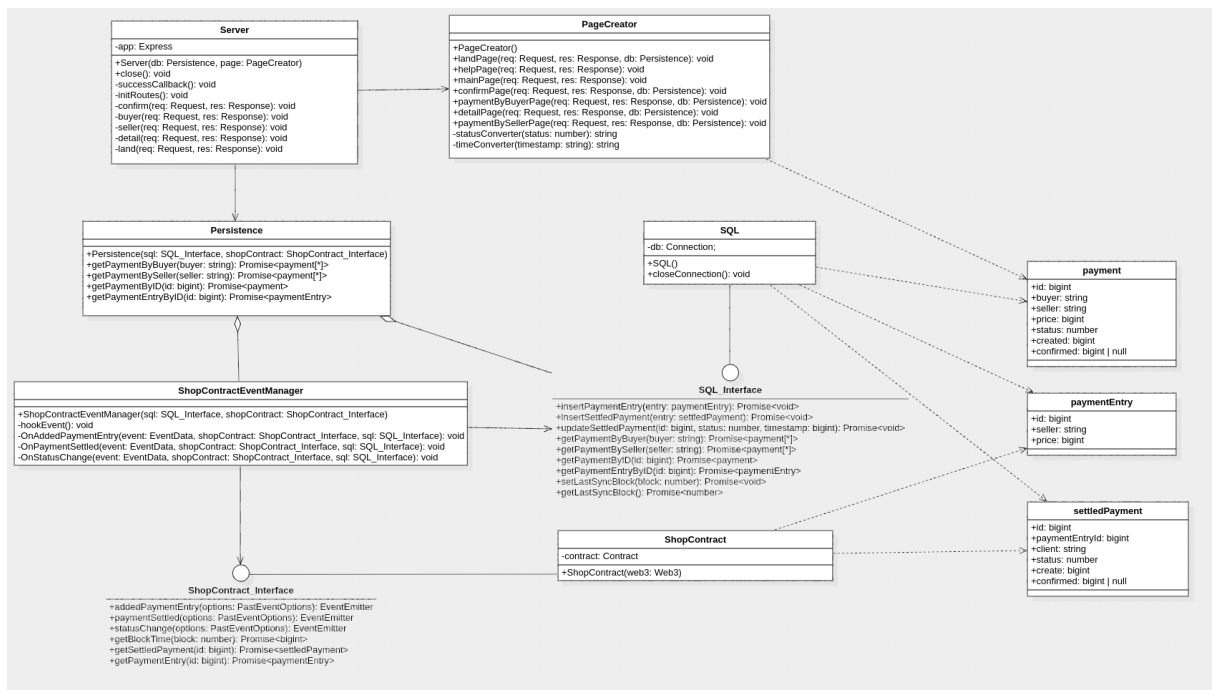
# 4 Architettura

Il progetto si compone di 4 macro parti:

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

## 4.1 Server

### 4.1.1 Diagramma delle classi



**Figure 1:** Diagramma delle classi del server

## Commenti

La classe ShopContract andrà a interfacciarsi con il contratto in blockchain.  
La classe PageCreator andrà a interfacciarsi con la WebApp.

### 4.1.2 Design pattern: Constructor injection

#### Descrizione

Le dipendenze sono tracciate e passate agli oggetti tramite il costruttore.

#### Motivazioni

Facilita il tracciamento delle dipendenze e agevola il mocking in fase di test.

### 4.1.3 Schema DB

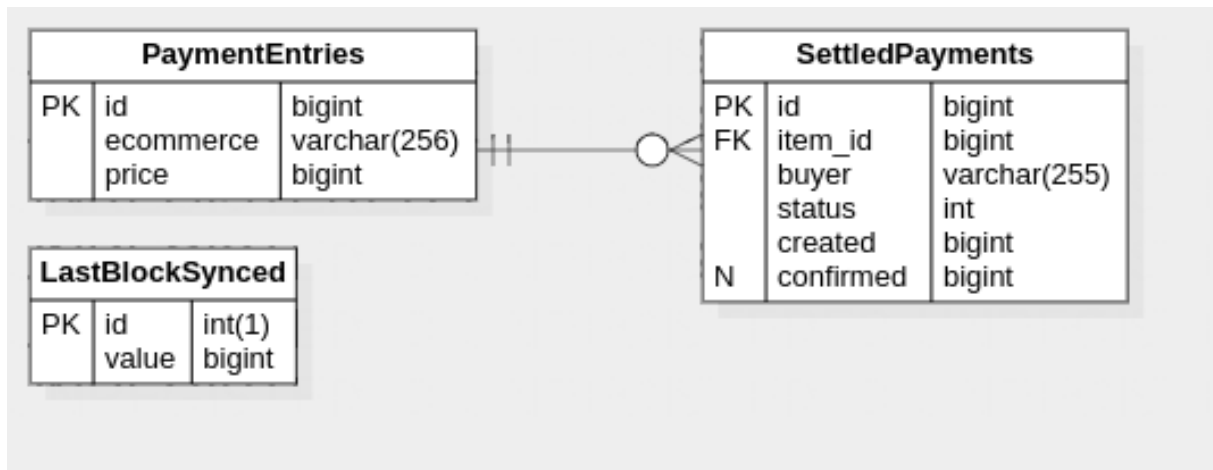


Figure 2: Schema del DB relazionale

#### Commenti

Schema delle tabelle del database.

**LastSyncedBlock** contiene una sola riga con *id* 0 con il valore dell'ultimo blocco sincronizzato.

## 4.2 Smart contract

### 4.2.1 Diagramma delle classi



Figure 3: Diagramma delle classi del contratto

## Commenti

### 4.2.2 Pattern architetturale adottato

#### Descrizione

#### Motivazioni

## 4.3 Web app

### 4.3.1 Diagramma delle classi

[grafico]

## Commenti

### 4.3.2 Pattern architetturale adottato

#### Descrizione

#### Motivazioni

## 4.4 Script di messa in vendita

### 4.4.1 `sell_item`

#### Parametri

price: float - il prezzo in dollari della entry di pagamento.

#### Valore Restituito

entry id: int - l'id dell'entry inserita in blockchain (-1 in caso di errore).

#### Comportamento

Il metodo inserisce una nuova entry di pagamento in blockchain con il prezzo specificato. Sia in caso di successo che di errore tutte le informazioni vengono salvate nel file *sell.log*.

## 5 Diagrammi di sequenza

### 5.1 Inizializzazione server web

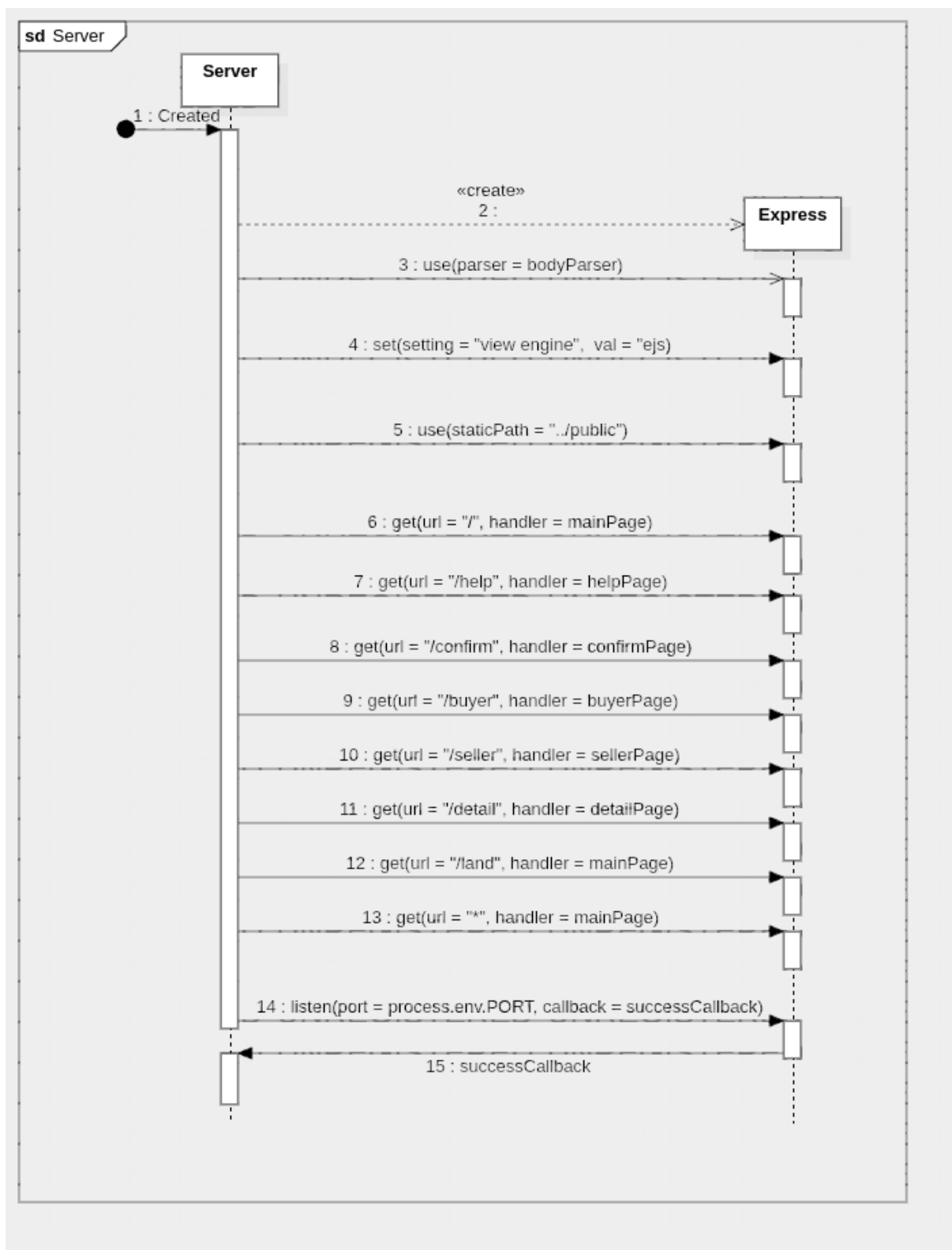


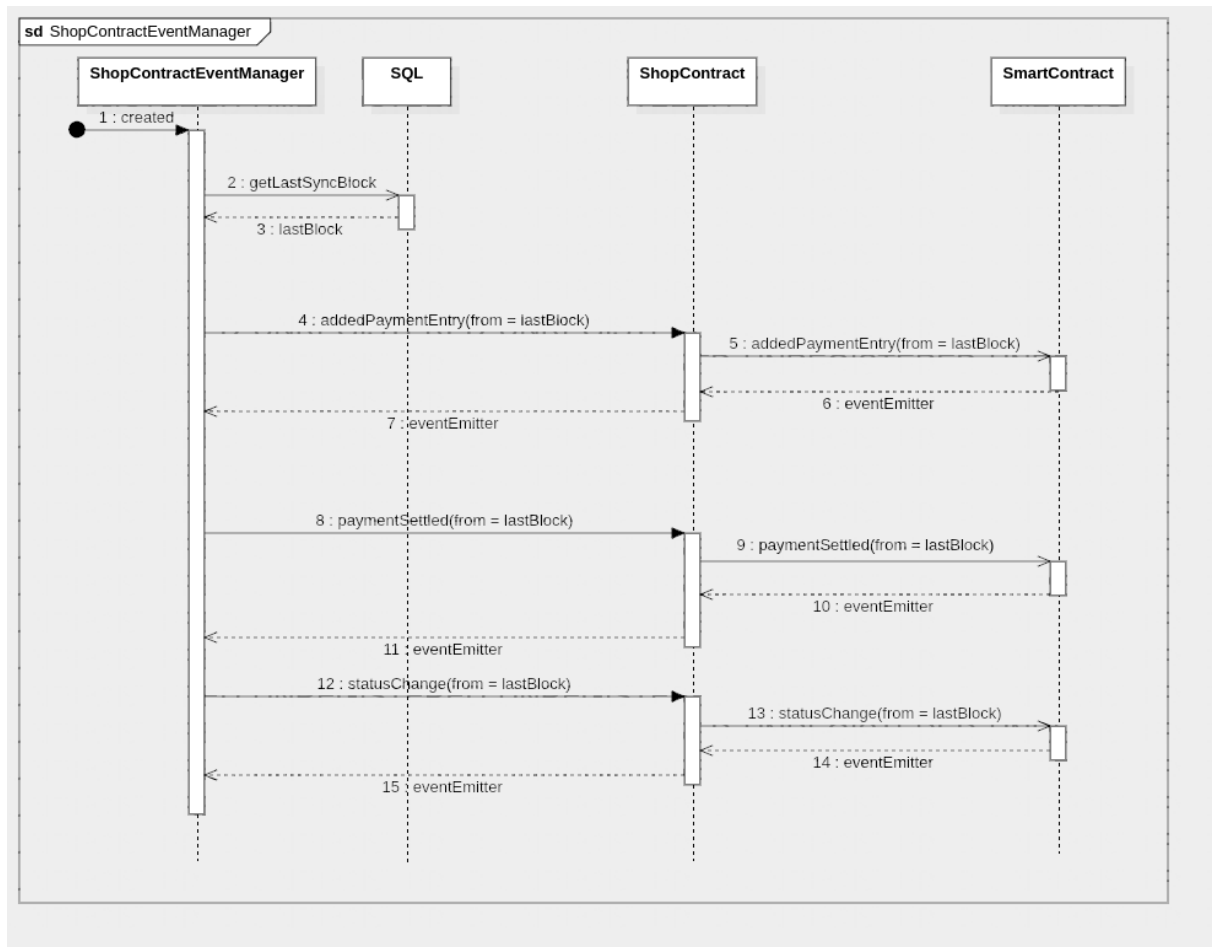
Figure 4: Diagramma di sequenza dell'inizializzazione del server



## Commenti

Mostra l'inizializzazione delle routes per express.

## 5.2 Ascolto eventi del contratto

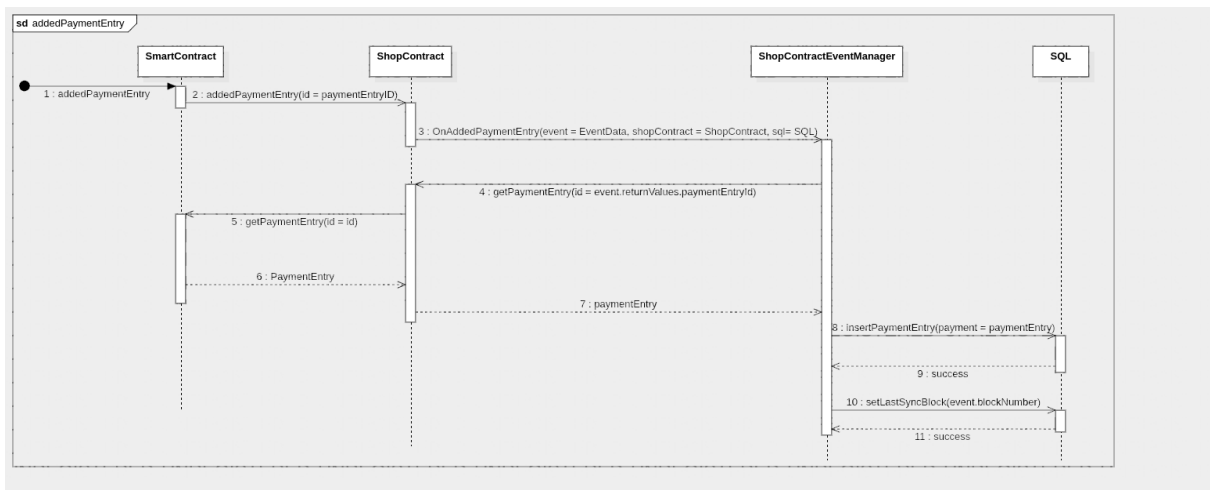


**Figure 5:** Diagramma di sequenza dell'ascolto degli eventi

## Commenti

Mostra la sottoscrizione degli eventi del contratto.

## 5.3 Nuovo oggetto in vendita

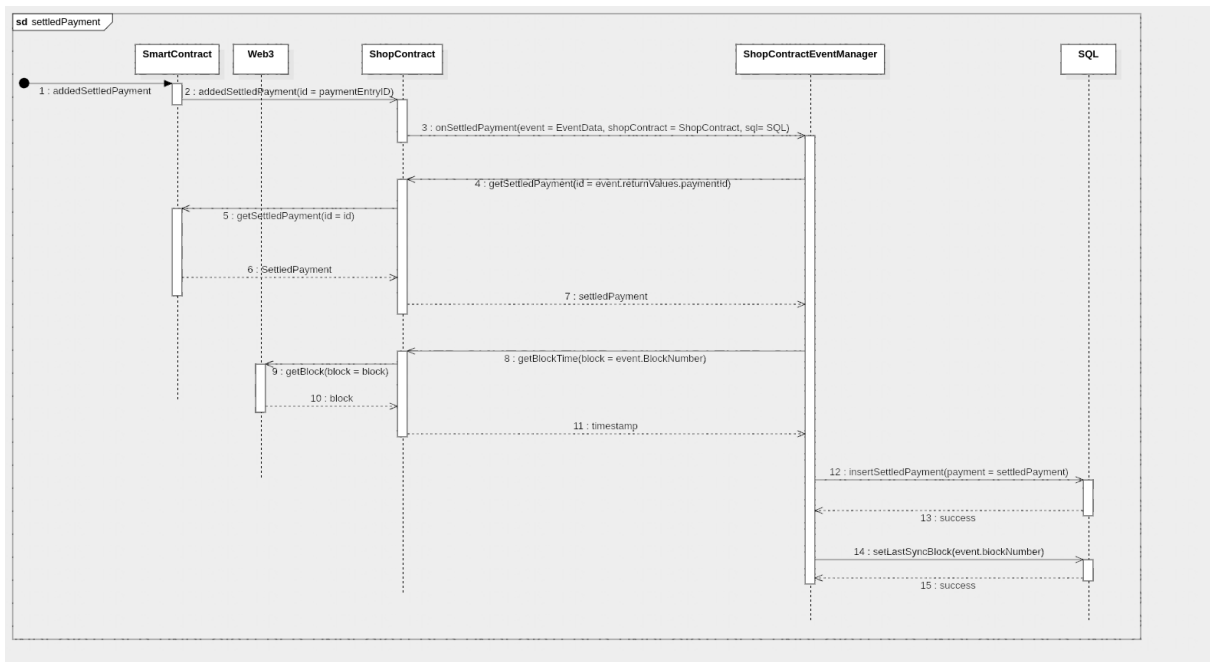


**Figure 6:** Diagramma di sequenza di un nuovo oggetto in vendita

### Commenti

Mostra cosa succede quando viene inserito una nuova entry di pagamento da parte di un e-commerce.

## 5.4 Nuova transazione



**Figure 7:** Diagramma di sequenza di una nuova transazione

### Commenti

Mostra cosa succede quando viene create una nuova transazione.

## 5.5 Cambio di stato di una transazione

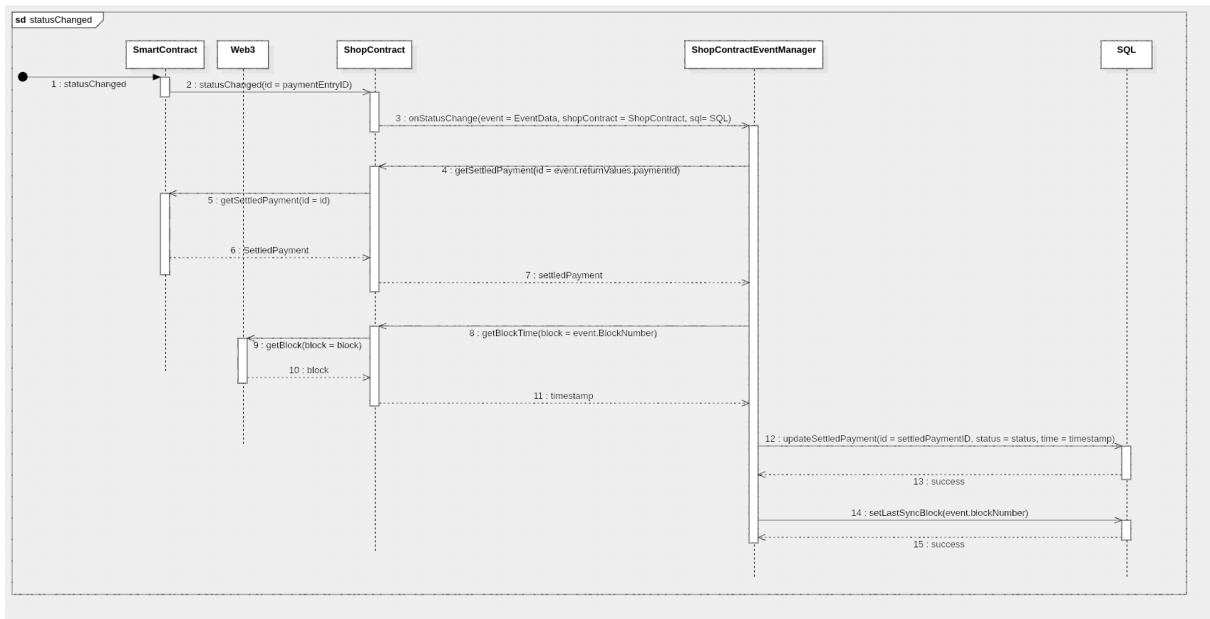


Figure 8: Diagramma di sequenza del cambio di stato di una transazione

### Commenti

Mostra cosa succede quando una transazione cambia di stato.

## 5.6 Pagina transazioni in entrata

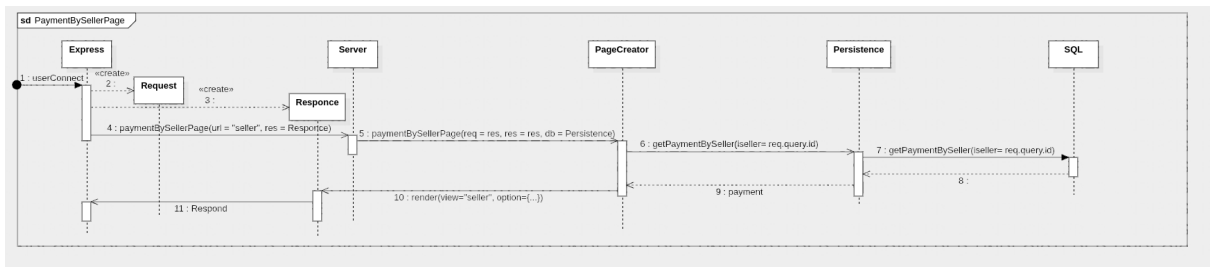


Figure 9: Diagramma di sequenza della richiesta della pagina delle transazioni in entrata

### Commenti

Mostra cosa succede quando un utente richiede la pagina delle transazione in entrata. Le altre pagine utilizzano lo stesso modello e dunque si preferisce evitare diagrammi di sequenza ridondanti.