OBERON

Manuale Sviluppatore v. 1.0.0

A.A. 2021-2022

Componenti del gruppo:

Casazza Domenico, matr. 1201136

Casonato Matteo, matr. 1227270

Chen Xida, matr. 1217780

Pavin Nicola, matr. 1193215

Poloni Alessandro, matr. 1224444

Scudeler Letizia, matr. 1193546

Stojkovic Danilo, matr. 1222399

Indirizzo repository GitHub:
https://github.com/TeamOberon07/ShopChain

Università degli Studi di Padova

Registro delle modifiche

v.	Data	Nominativo	Ruolo	Descrizione
1.0.1	25/05/2022	Casazza Domenico	Verificatore	Stesura Architettura §(5)
1.0.0	03/06/2022	Scudeler Letizia	Responsabile	Modifiche layout documento
1.0.0	25/05/2022	Casazza Domenico	Amministratore	Approvazione del documento
0.3.0	25/05/2022	Casonato Matteo	Verificatore	Verifica del documento
0.2.2	15/05/2022	Chen Xida / Casonato Matteo	Programmatore	Aggiornamento procedura d'installazione §(4)
0.2.1	13/04/2022	Chen Xida	Progettista	Stesura avvio mobile app §(4.4)
0.2.0	01/05/2022	Casonato Matteo	Verificatore	Verifica del documento
0.1.1	13/04/2022	Chen Xida	Progettista	Stesura installazione §(4)
0.1.0	05/04/2022	Casazza Domenico	Verificatore	Verifica del documento
0.0.2	05/04/2022	Chen Xida	Progettista	Stesura Tecnologie §(2), Configurazione §(3)
0.0.1	30/03/2022	Chen Xida	Progettista	Creazione bozza documento §(1), §(2)

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento fornisce le informazioni necessarie per l'estensione e la manutenzione del prodotto ShopChain, infatti sono state riportate le tecnologie e le scelte progettuali effettuate, in modo tale che uno sviluppatore in futuro sappia come è stato realizzato il prodotto e i prerequisiti necessari per usarlo.
Sono state illustrate anche le procedure per l'installazione per lo sviluppo in locale.

1.2 Riferimenti

1.2.1 Riferimenti informativi

- È stato creato il documento *Glossario_1.0.0.pdf* per chiarire il significato dei termini tecnici che possono creare dubbi e perplessità.
- La pianificazione è divisa in sprint, seguendo la metodologia agile. Le modalità e il modello di sviluppo sono riportate nel documento *NormeDiProgetto_3.0.0.pdf*

1.2.2 Riferimenti tecnici

- Javascript:
  https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Npm:
  https://www.npmjs.com/
- React:
  https://it.reactjs.org/
- Flutter:
  https://docs.flutter.dev/
- Metamask:
  https://metamask.io/
- Ethers.js:
  https://docs.ethers.io/v5/
- Solidity:
  https://docs.soliditylang.org/en/v0.8.13/
- Hardhat:
  https://hardhat.org/
- Dart:
  https://dart.dev/
- TraderJoe:
  https://docs.traderjoexyz.com

2 Tecnologie

Nella tabella seguente sono state riportate le tecnologie necessarie per lo sviluppo degli applicativi:

Tecnologia	Versione	Descrizione
Linguaggi		
JavaScript	ES6	Viene utilizzato per rendere le pagine dinamiche attraverso gli eventi invocati dall'utente
HTML	5	Fornisce una struttura semantica alle pagine della webApp
CSS	3	Fornisce una grafica alle pagine della webApp
Solidity	0.8.x	Linguaggio di programmazione per codificare gli SmartContract
Dart	2.16.x	Linguaggio di programmazione per lo sviluppo mobile
Librerie e framework		
Npm	8.x	Gestisce i pacchetti necessari per le operazioni di build.
React	18.x	Libreria grafica per facilitare lo sviluppo front-end e rendere migliore l'UX grazie al metodo di renderizzazione
Metamask	10.x	E' una estensione browser che funge da ponte tra le DApp e la blockchain
TraderJoe	-	Exchange decentralizzato sulla blockchain Avalanche usato per lo scambio in stablecoin
Flutter	2.10.x	Framework per la creazione di interfacce mobile cross-platform
Dart_web3	0.0.x	Libreria dart utilizzata per interagire con le blockchain basate su Ethereum

Wallet_connect	1.0.x	Libreria dart utilizzata per collegarsi al proprio wallet
Qr_code_scanner	0.6.x	Libreria dart utilizzata per scannerizzare QR code utilizzando la fotocamera del dispositivo mobile
Ethers.js	5.x	Libreria di utility per le interazioni con l'ecosistema delle blockchain

Nella tabella seguente sono state riportate le tecnologie necessarie per l'analisi e l'integrazione del codice:

Tecnologia	**Versione**	**Descrizione**
Analisi statica		
ESLint		Da vedere come utilizzare
Slither	0.8.x	Framework Solidity per la rilevazione di falle di sicurezza negli smart contract
Solidity Coverage	0.7.21	Tool (pacchetto NPM) per il report sulla copertura dei test relativi agli smart contract
Analisi dinamica		
Jest	27.x	Framework utilizzato per l'analisi dinamica del codice
React Testing Library	12.x	Libreria usata per testare le componenti di React
Hardhat	2.9.x	Ambiente di testing per lo sviluppo su blockchain

3 Configurazione

In questa sezione sono descritti i requisiti minimi di sistema per l'installazione del prodotto in locale da Github e l'avvio di ShopChain.

3.1 Requisiti hardware

Per avere buone prestazioni è preferibile avere almeno le seguenti specifiche hardware sui propri dispositivi:

Componente	Requisito
Processore	Quad-Core 1.8Ghz (Desktop) Octa-Core 1.8Ghz (Mobile)
RAM	8GB DDR4 (Desktop) 4GB LPDDR4 (Mobile)

3.2 Browser

La webApp è stata realizzata sulle seguenti versioni del browser:

Componente	Versione
Chrome	99
Firefox	98
Opera	83
Edge	99

4 Installazione

Per utilizzare l'applicazione web è necessario seguire questo procedimento:

1. Clonare la repository
2. Avviare il server e-commerce
3. Avviare il server webApp

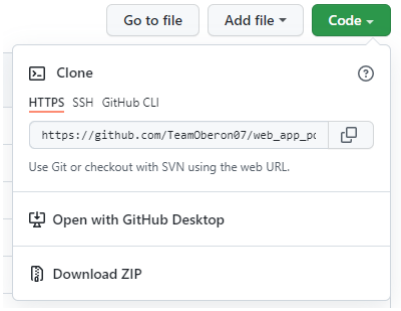
Per quanto riguarda l'avvio dell'app mobile, basterà:

1. Clonare la repository
2. Avviare l'applicazione

4.1 Clonazione repository

La clonazione della repository può avvenire in due modi:

1. Scaricare direttamente il codice in formato.zip



2. Clonare il repository usando il comando su un terminale:

```
git clone https://github.com/TeamOberon07/webApp.git
```

```
git clone https://github.com/TeamOberon07/mock_e-commerce.git
```

oppure

```
git clone https://github.com/TeamOberon07/mobileApp.git
```

Infine basterà accedere alla cartella dove si ha scaricato il prodotto.

4.2 Avvio server e-commerce

Per avviare il server che simula l'e-commerce:

1. aprire un terminale nella cartella dove è presente la webApp
2. usare il comando: `npm install` (primo avvio)
3. usare il comando: `npm run server`

```
matteo@matteo-PC:~/Desktop/GitHub/mock_e-commerce$ npm run server

> landingpage@0.1.0 server
> npx json-server --watch ./src/e-comm_db.json --port 8000


  \{^_^}/ hi!

  Loading ./src/e-comm_db.json
  Done

  Resources
  http://localhost:8000/orders

  Home
  http://localhost:8000

  Type s + enter at any time to create a snapshot of the database
  Watching...
```

4. per controllare gli ordini esistenti sull'e-commerce digitare l'url `http://localhost:8000/orders`


```
{
  "id": 1,
  "price": "4.99",
  "sellerAddress": "0x25EfE244b43036aF8915Aa9806a478f9405D31db",
  "buyerAddress": "0x90FC8a77E3a62A20f73CAcAaA04c3A2c22452B62",
  "confirmed": true,
  "hash": "0x09b430fbe46c03d8641fc318d985c9b9a6de1236006a9d1ff575a19c47025963"
},
{
  "id": 2,
  "price": "2.99",
  "sellerAddress": "0x25EfE244b43036aF8915Aa9806a478f9405D31db",
  "buyerAddress": "0x90FC8a77E3a62A20f73CAcAaA04c3A2c22452B62",
  "confirmed": true,
  "hash": "0x5f3f5c286dd0dcf0860b2c591b62c412dedb97bb042f26e9ba1943fde369754f"
},
```


5. per creare nuovi ordini dall'e-commerce bisogna avviare il server in locale che rappresenta il front-end dell'e-commerce con il comando `npm start`,

6. digitare l'url `http://localhost:3001/orders-page` per ottenere la schermata sottostante:

7. per creare l'ordine inserire il prezzo, selezionare il Seller, inserire l'indirizzo del Buyer e premere "Submit".

4.3 Avvio server webApp

Per avviare la webApp:

1. aprire un terminale nella cartella dove è presente la webApp
2. usare il comando: `npm install` (primo avvio)
3. usare il comando: `npm start`
4. si apre automaticamente una pagina sul browser predefinito dove è avviato la webApp

```
PS C:\Users\Xida\Documents\GitHub\web_app_poc> npm start
Compiled successfully!

You can now view reactapp in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000
```

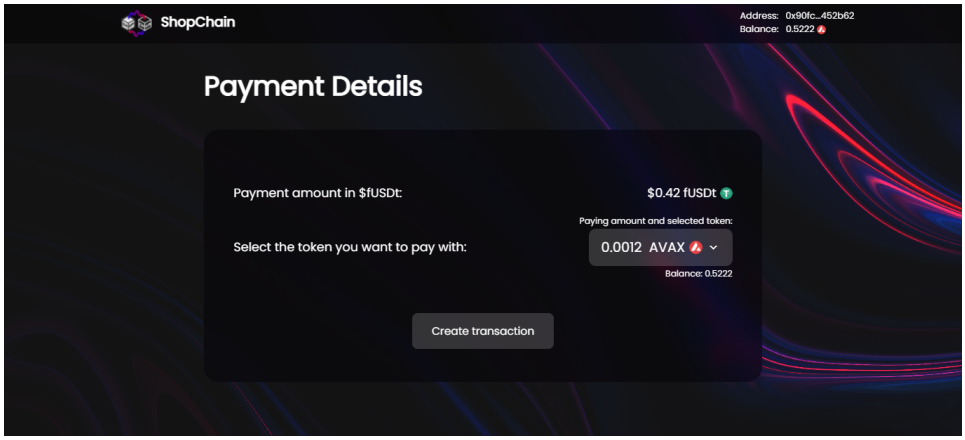
   Per creare una build ottimizzata per la produzione usare i seguenti comandi:

   (a) `npm run build`

   (b) `npm install -g serve` (se non è stato già installato)

   (c) `serve -s build`

5. Si può accedere alla vista principale (lista degli ordini) digitando l'url:
   `localhost:3000`
   mentre per accedere alla Landing Page all'url:
   `http://localhost:3000/landing-page?order=[numero dell'ordine]`



4.4 Avvio mobile app

4.4.1 Installazione flutter

Per utilizzare l'applicazione mobile è necessario avere installato Flutter e ciò richiede l'installazione di diversi componenti ossia:

1. Flutter SDK
2. Android Studio e Android SDK
3. Chrome
4. Visual Studio
5. VS Code (e le estensioni Dart e Flutter)

Per installare Flutter seguire la guida redatta dal team di Google:

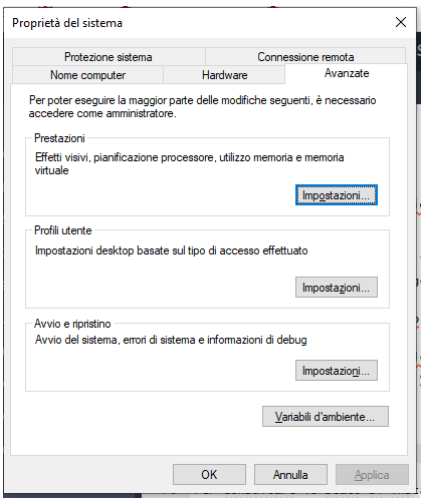
`https://docs.flutter.dev/get-started/install`

Durante l'installazione ci sono due punti critici dove porre attenzione:

1. Impostare la variabile ambiente per flutter
2. Scrivere su linea di comando `flutter` per assicurarsi che sia stato correttamente installato

Per impostare la variabile ambiente:

1. digitare "env" sulla barra di ricerca e cliccare su "Modifica le variabili d'ambiente relative al sistema" -> "Variabili d'ambiente" -> selezionare la variabile "Path" -> Modifica -> scrivere il percorso dove è stato installato flutter (es percorso C:Cartella di installazione/flutter/bin



Dopodichè se l'installazione è riuscita a seguito del comando `flutter` si otterrà la seguente schermata:

```
C:\Users\Xida>flutter
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

Global options:
-h, --help                  Print this usage information.
-v, --verbose               Noisy logging, including all shell commands executed.
                            If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                            diagnostic information. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id             Target device id or name (prefixes allowed).
    --version               Reports the version of this tool.
    --suppress-analytics    Suppress analytics reporting when this command runs.

Available commands:

Flutter SDK
  bash-completion   Output command line shell completion setup scripts.
  channel           List or switch Flutter channels.
```

Sono stati riportati i link per l'installazione dei componenti e la relativa documentazione nei siti di produzione ufficiali:

Flutter SDK:
`https://docs.flutter.dev/get-started/install`

Android Studio e Android SDK:
`https://developer.android.com/studio`

Chrome:
`https://www.google.it/intl/it/chrome/`

Visual Studio:
`https://visualstudio.microsoft.com/it/downloads/`

VS Code:
`https://code.visualstudio.com/`

Per accedere alla sezione delle estensioni di VSCode cliccare sull'icona più in basso dell'immagine riportata:

Per installare le estensioni Dart e Flutter su VSCode è sufficiente digitare i loro nomi sulla barra di ricerca e premere sul pulsante d'installazione.

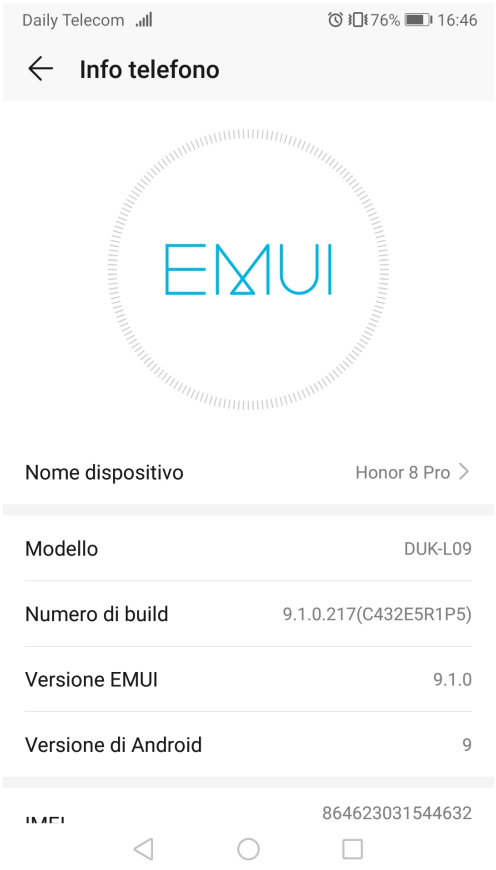
Per consultare lo stato di installazione dei singoli componenti basta scrivere nel terminale: `flutter doctor`

```
PS C:\Users\Xida\Documents\GitHub\mobile_app_poc> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 2.10.5, on Microsoft Windows [Versione 10.0.19044.1645], locale it-IT)
[√] Android toolchain - develop for Android devices (Android SDK version 32.0.0)
[√] Chrome - develop for the web
[√] Visual Studio - develop for Windows (Visual Studio Community 2022 17.1.5)
[√] Android Studio (version 2021.1)
[√] VS Code (version 1.66.2)
[√] Connected device (3 available)
[√] HTTP Host Availability

• No issues found!
PS C:\Users\Xida\Documents\GitHub\mobile_app_poc>
```

4.4.2 Run della build

1. Dopo aver installato flutter basterà aprire la cartella del progetto:; File -> Open Folder -> click sulla cartella corretta -> Select Folder;
2. Collegare il proprio dispositivo mobile in modalità sviluppatore con un cavo USB al computer e settare come funzionamento trasferisci file/multimedia.;
3. Per accedere alla modalità sviluppatore andare su Impostazioni -> Sistema -> Info telefono -> cliccare numero di build 7 volte;



4. installare/aggiornare i pacchetti delle dipendenze con `flutter pub get`;

5. Creare la build e avviare l'applicazione con `flutter run`;
6. Per avviare in modo più rapido dopo la prima build si può scegliere di usare la funzionalità hot reload o andare su main.dart e cliccare sull'icona di avvio in debug in alto a destra.

5 Architettura

5.1 Pattern architetturale

L'architettura di ShopChain è di tipo Fully Decentralized, o Pure DApp, cioè non fa uso di un backend centralizzato ma comunica esclusivamente con la blockchain tramite smart contract, prendendo da esso tutte le informazioni e funzionalità necessarie.

L'architettura di ShopChain fa riferimento al *"Pattern B – Self-Confirmed Transactions"* descritto nel paper *"Engineering Software Architectures of Blockchain-Oriented Applications"* di F. Wessling e V. Gruhn dell'Università di Duisburg-Essen.

In questo paper il Pattern B consiste nell'interazione dell'utente solo con un'applicazione web e/o un gestore di wallet (in questo caso MetaMask) per creare transazioni su una blockchain: le transazioni non vengono create direttamente dall'utente ma vengono generate dall'applicazione web e poi mandate manualmente al nodo della blockchain a cui l'utente è collegato.

Fonte:

- https://ieeexplore.ieee.org/abstract/document/8432174

5.2 Design patterns

Nell'applicativo ShopChain sono stati utilizzati i seguenti design patterns.

5.2.1 WebApp

- **React Hooks**: *useState, useEffect* e *useContext* per la visualizzazione dinamica della webApp: l'hook *useContext*,in particolare, viene utilizzato per istanziare la classe *StateContext* all'interno dei componenti che richiedono di comunicare con lo smart contract;
- **Provider**: applicato alla classe *StateContext* per fare in modo che sia accessibile globalmente all'interno della webApp in quanto è l'unica classe che si occupa di comunicare con lo smart contract;
- **Guard Check**: applicato al componente *Orders* per il controllo degli input inseriti dall'utente;
- **Access Restriction**: applicato al componente *OrderPage* per distinguere le funzionalità dell'utente venditore da quelle dell'utente compratore;
- **Presentational and Container Component**: applicati ai componenti "presentational", cioè *SwitchNetwork, Loading, NoWalletDetected, ConnectWallet, Log, Error*, e ai componenti "container", cioè *DApp e LandingPage)*.

5.2.2 Smart Contract

- **Oracle**: implementato per avere accesso ad informazioni esterne alla blockchain quale il prezzo della stablecoin;
- **Checks Effect Interactions**: implementato per evitare interazioni malevole da parte di altri smart contract;

- **Guard Check**: applicato agli input ricevuti dall'utente;
- **Access Restriction**: applicato alla maggior parte delle funzioni per distinguere le funzionalità dell'utente venditore da quelle dell'utente compratore.

5.2.3 App Mobile

- **Singleton**: applicato alla classe *stateContext* per averne un'unica istanza accessibile da molteplici classi;
- **Presentational and Container Component**: applicati alle classi "presentational", cioè *Log, Order, QRScanPage*, e ai componenti "container", cioè *MyHomePage, OrdersPage, MyApp*.