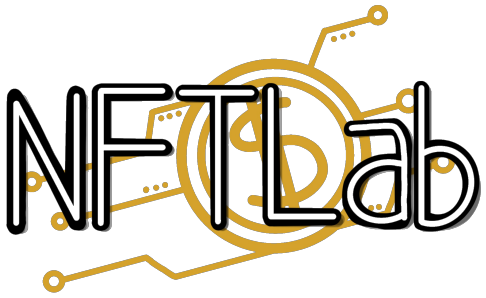

Progetto: ***NFT Lab***
f.pallaro@synclab.it

Allegato Tecnico

Informazioni sul documento	
Versione	1.0
Redattori	Baldisseri Michele
Uso	Interno
Distribuzione	*Sync Lab*

Descrizione
Questo documento racchiude tutte le informazioni necessarie per l'estensione e la manutenzione del prodotto *NFT Lab*.

Indice

1 Organizzazione del progetto

Nella directory principale del progetto sono presenti tutti i file di configurazione degli strumenti utilizzati, come le impostazioni dell'IDE, la configurazione della continuos integration e i file per il versionamento attraverso Git. Oltre a questo, è presente la cartella ***src*** al cui interno si trovano i file del prodotto vero e proprio. La struttura è organizzata nel seguente modo:

- **File principali**: *index.html*, *styles.css* e *main.ts*, utilizzati a livello globale dalla web app;
- **Cartella *environments***: contiene le variabili globali, sia per il prodotto in fase di sviluppo sia nel momento in cui entra in produzione (per esempio sono presenti le variabili utili per modificare le chiamate API);
- **Cartella *e2e***: contiene gli script dei test *end to end* creati con Taiko;
- **Cartella *assets***: è accessibile da qualsiasi file e contiene tutte le risorse che la web application utilizza, come immagini o documenti. Inoltre qualsiasi risorsa contenuta è raggiungibile via browser tramite richiesta HTTP;
- **Cartella *app***: comprende a sua volta il contenuto riportato di seguito.
  - **Cartella *mocks***: contiene file utili in fase di test per creare un *mock* di componenti o servizi;
  - **Cartella *models***: contiene le interfacce degli oggetti che rappresentano il dominio del progetto;
  - **Cartella *services***: contiene tutti i servizi utilizzati dai componenti per effettuare le chiamate API;
  - **Cartella *shared***: contiene tutti i componenti che sono stati riutilizzati.
  - Una cartella **per ogni pagina** del sito, che comprende tutti i componenti utilizzati in ciascuna.

2 Continuous integration

Sul repository di progetto è stato creato un workflow che viene attivato automaticamente ad ogni push. Questo permette di:

- Installare le dipendenze tramite *Node Package Manager*;
- Eseguire il processo di *build* dell'applicazione;
- Eseguire i test d'unità e calcolare la code coverage;
- Caricare i risultati dei test su *Coveralls*.

Il workflow in questione può essere modificato per estenderne le funzionalità o per collegare il repository ad altri servizi esterni.

3 Servizi

In questa sezione vengono riportati tutti i servizi creati, il loro compito e i componenti che utilizzano questa dipendenza.

- **AuthenticationService**: presenta il metodo per effettuare l'autenticazione dell'utente nel sistema. Se le informazioni, e-mail e password, sono corrette e individuano un'istanza nel database, viene ritornato un JSON contenente tutte le informazioni dell'utente, in modo da salvarle nel *local storage*. Il componente che dipende da questo servizio è *login-form-component*.

```
//user informations
{
    id: 12,
    name: "test",
    surname: "test",
    email: "test06@gmail.com",
    dob: "1998-06-01",
    wallet: "0xEd1bB395f00B22454c22B6c76b645657c739D3cc",
    age: 23
}
```

Esempio di codice 1: Esempio di JSON ritornato dal servizio di autenticazione

- **CategoriesService**: presenta il metodo per richiedere tutte le tipologie di categorie presenti nel database. I componenti che dipendono da questo servizio sono *filter-search*, *modify-opera-form*, *new-opera-form* ed *opera-management*.

```
//categories
{
    [
        {id: 1, name: "food"}, {id:2, name: "sport"}
    ]
}
```

Esempio di codice 2: Esempio di JSON ritornato dal servizio di richiesta delle categorie

- **OperaManagementService**: presenta tutti i metodi che riguardano la gestione delle opere dell'utente, quali le richiesta di tutte le sue opere, la modifica di un'opera e l'aggiunta di una di nuova. I componenti che dipendono da questo servizio sono *modify-opera-form*, *new-opera-form* ed *opera-management*.

- **OperasService**: presenta il metodo per richiedere tutte le opere in vendita. Il componente che dipende da questo servizio è *home*.

```
//opera informations
{
    id:"Qmb13ALEkqXtV5DT6jcrXuUGjat",
    title: "test",
    description: "test",
    authorId: 18,
    price: 52.0,
    currency: "ETH",
    status: false,
    path: "gallery/Qmb13ALEkqXtV5DT6jcrXuUGjat.jpg",
    type: "img",
    owner: "test",
    author: "test",
    categories: [{id: 2, name:"sport"}]
}
```

Esempio di codice 3: Esempio di JSON utilizzato per le opere

- **SignupService**: presenta il metodo per inviare la richiesta di registrazione dell'utente nel sistema. Il componente che dipende da questo servizio è *signup-form*.
- **UserManagementService**: presenta tutti i metodi che riguardano la gestione delle informazioni dell'utente, quali la modifica dei suoi dati e la modifica della password per accedere al sistema. I componenti che dipendono da questo servizio sono *modify-user-form* e *modify-psw-form*.

```
{
    oldPassword: "Test123@",
    newPassword: "Test123@123"
    email: "test@test.com"
}
```

Esempio di codice 4: Esempio di JSON utilizzato per la richiesta di modifica password

4 Implementazione

Per lo sviluppo dell'interfaccia grafica si è deciso di utilizzare la libreria *Angular Material*, per sfruttare componenti con proprio stile, e *Flex Layout*, per rendere il sito responsive e diminuire le righe di codice dei file CSS. Nel file *app.module.ts* sono importati solamente i moduli di Angular Material che sono stati effettivamente utilizzati; per una maggiore leggibilità i moduli sono contenuti all'interno dell'array *materialModules*.

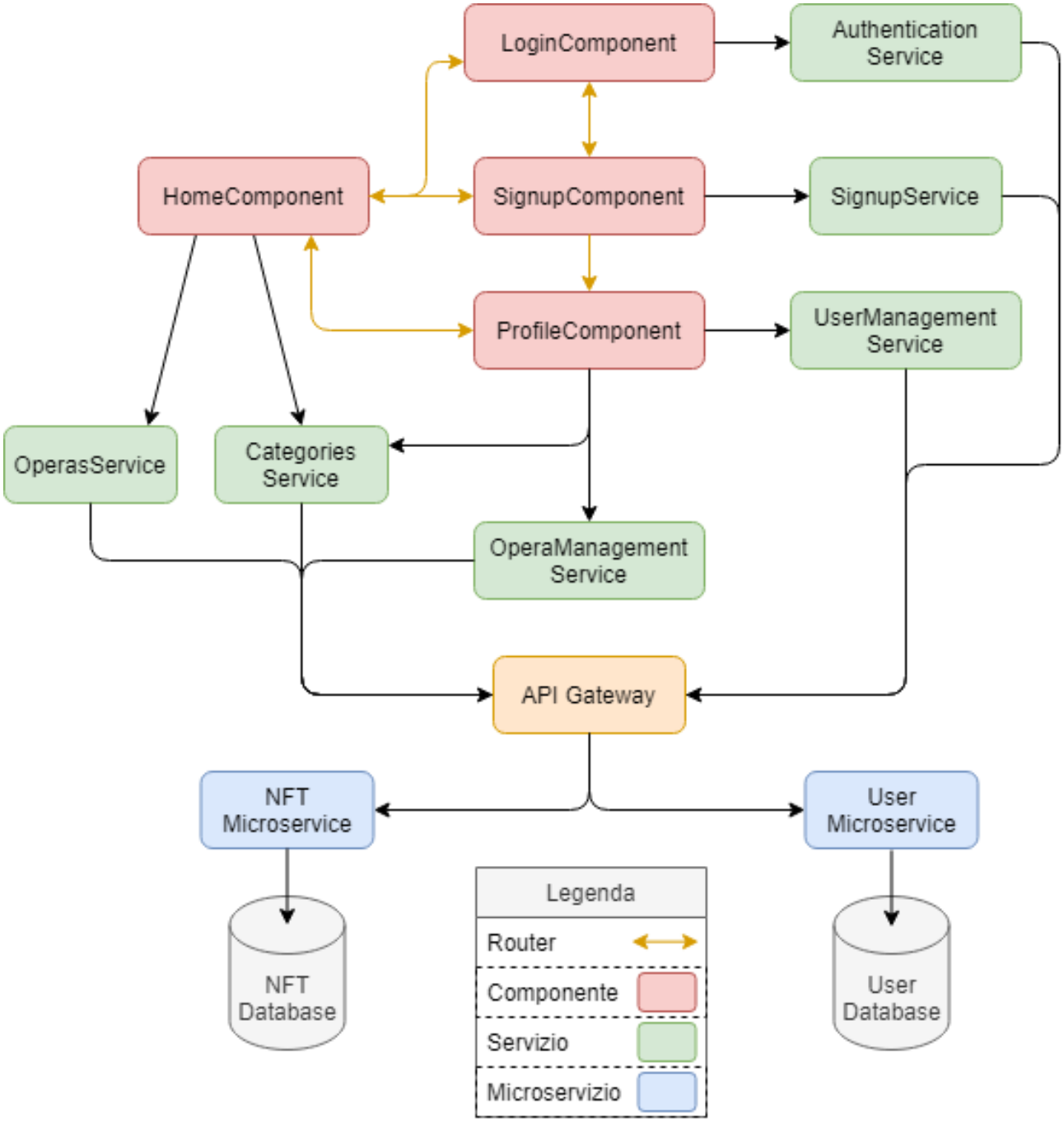

Figura 1: Componenti dell'applicativo e relativi collegamenti

La figura fa notare come l'applicativo sia sviluppato su cinque livelli. Al primo livello troviamo i componenti Angular con il compito di visualizzare e gestire i dati. I service stanno al secondo livello, e forniscono la gestione delle chiamate HTTP al back-end tramite l'API Gateway che, in base alla richiesta ricevuta dal service, chiama il microservizio dedicato a rispondere. In ultimo livello troviamo il database sul quale effettuare le operazioni richieste.

4.1 Registrazione e autenticazione dell'utente

La pagina di registrazione si presenta con un semplice form per la compilazione di tutti i campi (nonché FormControl soggetti a validazione) necessari all'inserimento dell'utente nel sistema. Al momento della conferma, nel caso in cui l'utente non compilasse correttamente il form, verranno presentati dei messaggi d'errore esplicativi, in modo da guidarlo durante questa fase. Se tutti i dati inseriti sono corretti viene richiamato *SignupService*, il quale effettuerà la POST. Se la risposta avrà esito positivo l'utente verrà reindirizzato nella *home*, altrimenti verrà visualizzato un errore in base all' *HTTP status code* ottenuto.

Esempio di password valida: Test123@
Esempio di un indirizzo di wallet valido: 0xEd1bB395f00B22454c22B6c76b645657c739D3cc

La pagina di autenticazione è molto simile (ma più semplice) e presenta le stesse caratteristiche della pagina precedente. In questo caso verrà chiamato *AuthenticationService*, che effettuerà la POST per recuperare le informazioni dell'utente. Queste vengono poi salvate nel *local storage*, in modo da poterle utilizzare dove necessario.

4.2 Pagina di profilo dell'utente

In questa pagina sono presenti sia le informazioni dell'utente che la parte di gestione delle opere in suo possesso. Tutti i pulsanti presenti in questa pagina apriranno un modale dedicato.
Per quanto riguarda la prima parte, le informazioni visualizzate vengono prelevate dal *local storage*, presenti in seguito al login. L'utente può inoltre cliccare sul proprio indirizzo del wallet per essere reindirizzato sul sito *www.blockchain.com* ed avere così ulteriori informazioni sullo stato del proprio conto.
Al di sotto sono visibili i pulsanti di modifica dei propri dati: anche in questo caso, entrambi i modali, presentano delle form soggette a validazione del contenuto inserito e l'utente sarà sempre avvisato in caso d'errore. Nonostante la modifica della password sia gestita separatamente rispetto agli altri dati, viene richiamato lo stesso servizio, *UserManagementService*, che si occupa di gestire entrambe le PUT. In caso di esito positivo, i dati presenti nel *local storage* verranno aggiornati; questa operazione non sarà effettuata per la modifica della password perché, essendo dato privato e sensibile, non può essere salvato localmente.
La seconda sezione è completamente dedicata alla gestione delle opere dell'utente e tutte le chiamate al back-end vengono effettuate mediante *OperaManagementService*. Al caricamento della pagina viene invocata la GET che recupera tutte le opere, per essere salvate e poi visualizzate. Selezionando un'opera tra quelle in lista verranno mostrate tutte le sue informazioni, mentre premendo il tasto di modifica si aprirà un modale dallo stile e funzionamento molto simile a quello dedicato ai dati dell'utente. Tuttavia, in questo caso, all'apertura del modale viene effettuata una GET da parte di *CategoriesService*, in modo che l'utente abbia a disposizione tutte le categorie presenti nel sistema; la richiesta di modifica sarà invece delineata da una PUT. Infine, per quanta riguarda l'inserimento di una nuova opera, viene utilizzato una ***multipart request***: dopo aver creato un oggetto *FormData*, viene effettuato l'*append* sia del file caricato che dell'oggetto contenente le informazioni dell'opera. Per questa POST non viene specificato l'*enctype* nell'header, in quanto la dimensione del file inviato viene calcolata automaticamente dal browser.
L'utente ha a disposizione anche un filtro di ricerca per recuperarle in modo più agevole.

4.3 Homepage

La homepage si serve di *OperasService* per ricevere tutte le opere in vendita nell'applicativo e la GET viene effettuata ad ogni nuovo caricamento della pagina. La barra di navigazione, presente nella parte superiore, varia in modo dinamico: se l'utente è autenticato sarà visibile un menù per accedere alle funzionalità disponibili, altrimenti saranno presenti i pulsanti per l'autenticazione e registrazione. In entrambi i casi è disponibile una barra di ricerca per scremare le opere presenti. Inoltre in questa pagina vengono riutilizzati i modali per la visualizzazione dei dettagli dell'opera e per il filtro di ricerca.