

Progetto di programmazione Web

e Mobile - A.A. 2023/24

Universe Heros: Album delle Figurine dei Super Eroi

Universe Hero è un'applicazione web per la creazione di un album, lo scambio e la compravendita di figurine virtuali.



Premesse

Tutte le didascalie, informazioni ecc. dell'applicazione web sono in **lingua inglese** per rendere le informazioni più uniformi con le informazioni passate dal server della Marvel (in lingua inglese). Tuttavia, la **documentazione**: i commenti delle funzioni all'interno del codice e lo swagger sono in **italiano** per fornire tutte le informazioni necessarie al fine di comprendere al meglio la struttura del progetto.

All'intero di questo progetto sono state progettate diverse **rarietà per le carte**, per rendere l'applicazione più coinvolgente e realistica. Il numero totale di carte all'interno dell'album sono 759 (Nella sezione dedicata all'album verrà approfondito e spiegato questo aspetto), ovvero i primi 1000 offset. Dopo varie analisi sono giunto alla conclusione che ogni eroe all'interno della Marvel ha un proprio numero di offset univoco (come se fossero degli indici). Sfruttando questa logica, ho creato delle **'zone di rarità'**, cioè:

- **Leggendarie**: Il range di **offset 0-79** rappresenta tutte le carte leggendarie. Le carte leggendarie rappresentano circa l'**8%** delle carte totali all'interno dell'album.
- **Eroiche**: Il range di **offset 80-199** rappresenta tutte le carte eroiche. Le carte eroiche rappresentano circa il **12%** delle carte totali all'interno dell'album.
- **Mitiche**: Il range di **offset 200-359** rappresenta tutte le carte mitiche. Le carte mitiche rappresentano circa il **16%** delle carte totali all'interno dell'album.
- **Super**: Il range di **offset 360-599** rappresenta tutte le carte super. Le carte super rappresentano circa il **24%** delle carte totali all'interno dell'album.
- **Comuni**: Il range di **offset 600-999** rappresenta tutte le carte comuni. Le carte comuni rappresentano circa il **40%** delle carte totali all'interno dell'album.

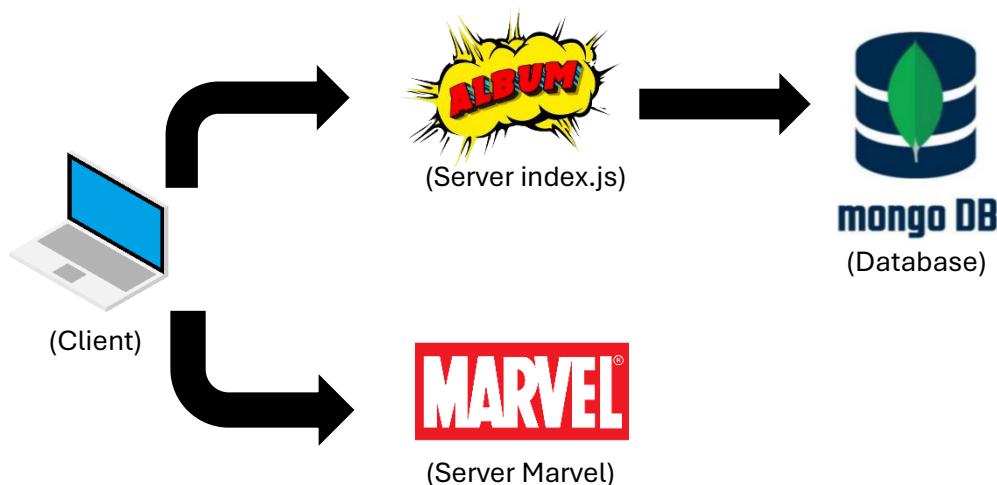
Queste percentuali sono indicative, questo perché durante la **generazione delle carte** nella fase di acquisto, il server controlla che la carta abbia un'immagine valida (ovvero che nel path non ci sia la stringa 'image_not_available'). Facendo delle analisi, le carte totali, contenenti un'immagine valida nei primi 1000 offset, sono 759. Se durante la generazione delle carte, il server genera un offset la cui immagine non è valida per il nostro applicativo (ovvero l'immagine non è disponibile), quest'ultimo procede con la generazione di un nuovo offset (indice).

Infine, per le **password** è stata applicata la tecnica del "**salting**", concatenando la stringa "\$Up3r_pR0j3c!-H3r@S" + password. Viene poi eseguito l'hash (**sha-256**) e memorizzata nel db.

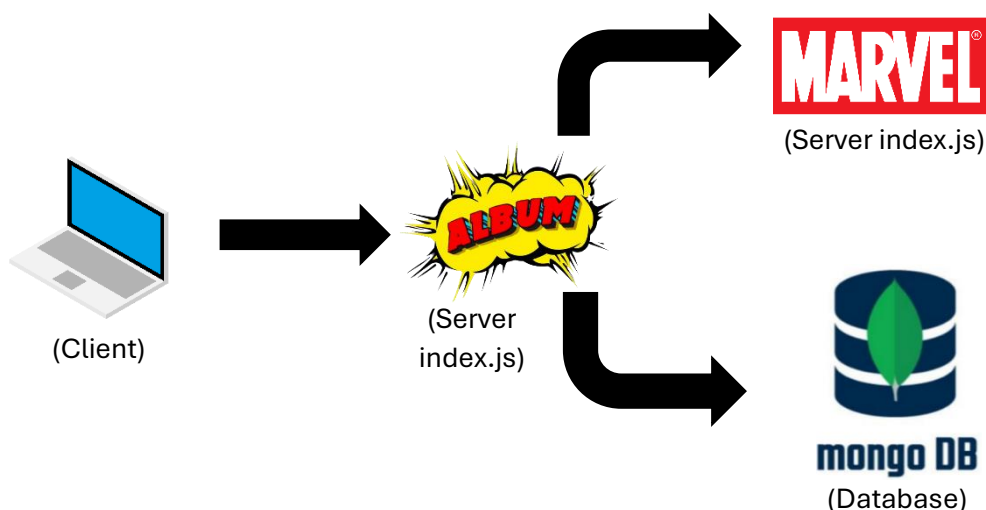
Architettura applicazione

Durante le scelte implementative per realizzare questo progetto, sono state analizzate due possibili architetture:

- Architettura con **front-end** che **comunica** con il server progettato per la gestione dell'**album** e il server della **Marvel**:



- Architettura **Server proxy**: Questa architettura prevede che il front-end **comunichi soltanto con il server**, il quale a sua volta comunicherà con altri servizi. Il server principale ('index.js'), farà da proxy verso tutte le chiamate esterne.



La scelta è stata la seconda architettura per i seguenti motivi:

- **Sicurezza**: La prima architettura presuppone che ogni utente abbia le chiavi per interagire con il server della Marvel. Nel nostro caso ogni utente avrebbe avuto la stessa coppia di chiavi (pubblica e privata), questo significa che se un'utente

commette un'azione malevola, o comunque effettua delle azioni che comportano il blocco delle chiavi, tutti gli altri utenti non potranno usufruire del servizio (perché le chiavi sono condivise tra i front-end).

- **Integrità dei dati:** Se si utilizzasse la prima infrastruttura, durante la fase di acquisto dei pacchetti delle figurine, il front-end dovrebbe interagire con il server della Marvel, per ottenere delle carte (eroi). Una volta finita questa fase, il front-end deve interagire con il server per comunicargli quali carte ha trovato. Tuttavia, il client potrebbe alterare le informazioni delle carte, e dichiarare di aver trovato altre carte (ad esempio delle carte importanti dalla quale può ottenere dei vantaggi scambiandole con altri utenti).

Durante la progettazione sono stati considerati maggiormente questi due scenari; tuttavia, le due infrastrutture hanno ulteriori vantaggi e svantaggi. Ad esempio, uno svantaggio dell'architettura scelta può essere la **latenza**. A causa della centralizzazione, il server potrebbe essere **più lento a livello computazionale**.

Per comprendere meglio la struttura del server si consiglia la lettura dello swagger in cui sono contenute le descrizioni di ogni API.

Struttura DB

Collections	Descrizione
Admin	All'interno di questa collection sono presenti tutti gli utenti admin. Durante il login viene controllata sia la collection admin che user per capire che tipo di utente sia.
Cards	All'interno di questa collection sono presenti tutte le carte degli utenti. Ogni carta contiene l'id dello user, per capire a chi appartiene, l'id della carta, e altre informazioni utili.
Draft_packets	All'interno di questa collection sono contenuti tutti i pacchetti creati dall'admin, non ancora pubblicati nello shop.
Packets	All'interno di questa collection sono presenti tutti i pacchetti disponibili nello shop. Sono tutti i pacchetti acquistabili.
Posts	All'interno di questa collection sono presenti le immagini e le didascalie create dall'admin da mostrare nell'homepage.
Trades	All'interno di questa collection sono presenti tutte le proposte di scambio create da utente. Sono presenti anche gli scambi già conclusi.
Users	All'interno di questa collection sono presenti tutti i dati inerenti agli utenti registrati. Questi dati possono essere utili durante la fase di login confrontando l'username e l'hash della password

Sicurezza chiamate API

Per proteggere l'applicazione dagli utenti che tentano di eseguire delle **chiamate senza** avere l'**autorizzazione**, sono state creati dei **middleware** in grado di controllare l'uid e accettare o meno l'azione richiesta. Ad esempio, in questa applicazione sono presenti chiamate che possono effettuare soltanto gli admin (ad esempio le api inerenti alla gestione dello shop), utenti registrati (ad esempio le api inerenti alle carte di un album), e utenti non registrati (ad esempio la visione dei pacchetti disponibili nello shop, o il login). Per questo motivo sono state create tre middleware in grado di eseguire questi controlli:

- **authAdmin**: Middleware usata per le funzionalità eseguibili solo dall'admin
- **authUser**: Middleware usata per le funzionalità eseguibili solo da utenti registrati non admin, che hanno eseguito l'accesso (utenti di base che gestiscono il proprio album)
- **auth**: Middleware usata per le funzionalità che possono svolgere tutti gli utenti che hanno eseguito un accesso (admin e utenti di base)

```
//Restituisce tutti i pacchetti in draft (visibili solo all'admin)
//Bisogna passare per forza l'uid dell'admin tramite query dell'url
app.get('/packets/drafts', authAdmin, (req, res) => {
  getDraftsPackets(res);
});
```

(Esempio utilizzo middleware 'authAdmin' durante la richiesta della lista dei pacchetti in draft).

```
//Prendo le carte di un utente
app.get('/cards/:uid', authUser, (req, res) => {
  getPageCards(req, res);
});
```

(Esempio utilizzo middleware 'authUser' durante la richiesta delle carte di un utente)

```
//Elimina un utente dal db
app.delete('/user/:uid', auth, (req, res) => {
  let uid = req.params.uid;
  deleteUser(uid, res);
});
```

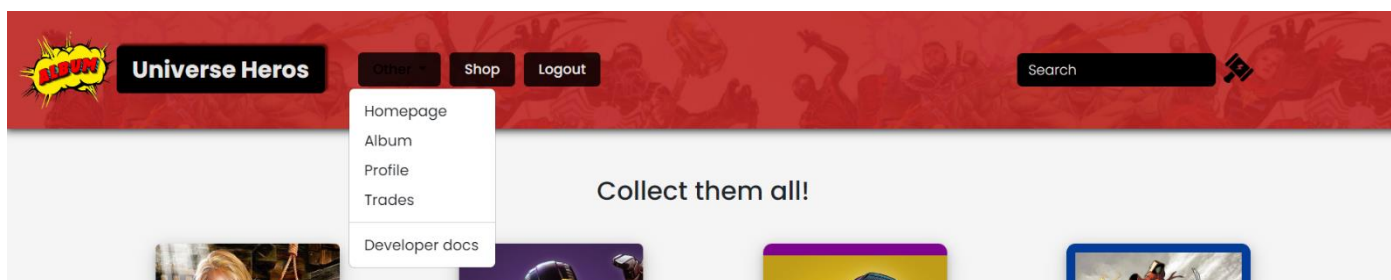
(Esempio utilizzo middleware 'auth' durante la richiesta per eliminare un'utente)

```
app.post('/login', (req, res) => {
  login(req.body, res);
});
```

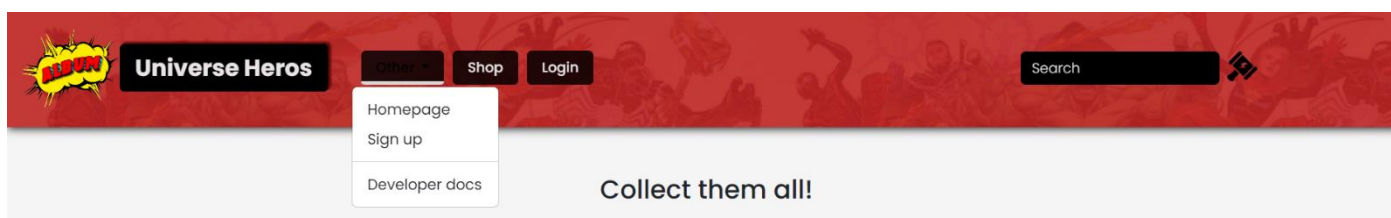
(Esempio chiamata senza controllo di sicurezza)

Gestione utenti

Il sito è **accessibile ad ogni utente**, sia utenti registrati, utenti senza un account e utenti che non hanno eseguito l'accesso. Gli utenti senza un profilo, o senza aver fatto l'accesso, potranno **navigare** in maniera molto **limitata** all'interno dell'applicazione web. Hanno la possibilità di visitare l'homepage, lo shop (senza poter acquistare i pacchetti), la documentazione per sviluppatori (lo swagger), la pagina per la registrazione e per l'accesso all'album. Il **menu** di queste pagine **si adatta** in base al tipo di utente che visiona la pagina. Ad esempio, un'utente senza un profilo (o senza aver fatto l'accesso) non sarà in grado di vedere nel menu il link per accedere alla gestione del profilo utente. Il menù viene personalizzato in base al tipo di utente, controllando se è presente un'uid nel localStorage.



(Schermata homepage, menù di un'utente all'interno dell'homepage dopo aver eseguito l'accesso).



(Schermata homepage, menù di un'utente all'interno dell'homepage senza aver eseguito l'accesso).

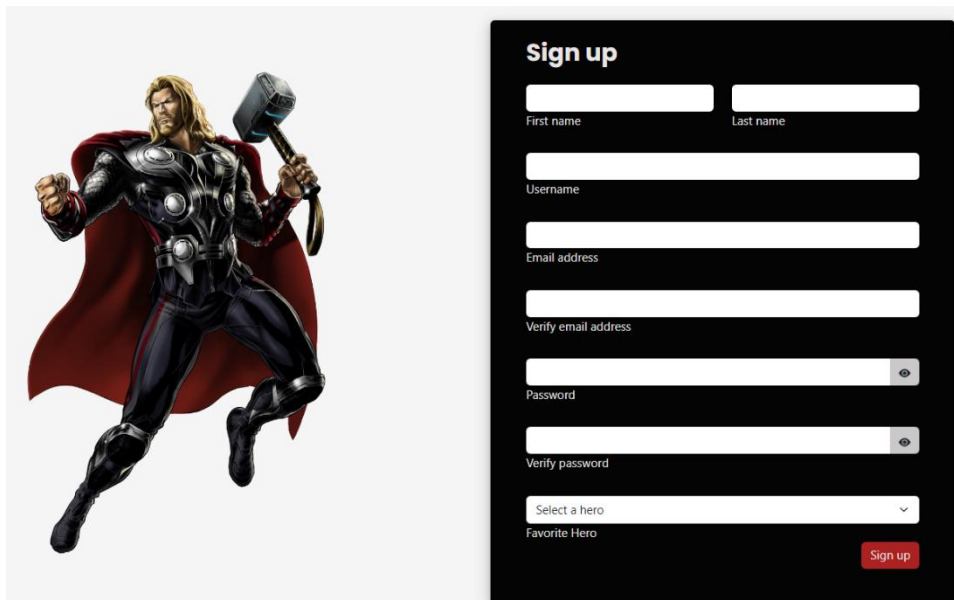
Inoltre, sono stati aggiunti ulteriori **controlli** in ogni pagina, in modo tale da evitare che un'utente senza un account (o senza aver fatto l'accesso) possa accedere a delle pagine alla quale non ha accesso. Al caricamento della pagina (evento 'onload()'), viene scatenata una funzione in grado di cercare **l'uid** nel localStorage, capire se è formattata correttamente, e nel caso proseguire con le chiamate al server. Se un utente senza l'uid tenta di navigare e accedere ad una pagina alla quale non ha accesso, modificando l'url, viene immediatamente reindirizzato nell'homepage. Infine, se un'utente tenta di inserire manualmente un'uid seguendo le regole di una regex, una volta fatta una chiamata al server, quest'ultimo risponderà con un 401 (per via dei middleware implementati).

```
function authAccess(){
  const checkID = /^[a-f0-9]{24}$/;

  if(!checkID.test(uidUser)){
    alert("Unauthorized access");
    return false;
  }
  return true;
}
```

(Funzione che controlla l'uid all'onload di una pagina).

Per poter utilizzare l'album, acquistare i pacchetti e utilizzare la maggior parte delle funzionalità offerte dal server, è necessario **registrarsi** e possedere un account. La registrazione prevede l'inserimento di una serie di informazioni, e al termine di questa fase verrà creato un ID (uid), in grado di identificare univocamente l'utente. L'applicazione sfrutterà il valore '_id' generato da MongoDB come uid per la gestione degli utenti. Non è possibile usare un'e-mail o username già usati.



(Schermata signup, form per la registrazione)

Il campo per l'eroe preferito serve per impostare una foto profilo ad ogni utente. Il campo 'Favorite Hero' è strutturato nel seguente modo: un'utente inserisce il nome o le iniziali del nome di un supereroe, clicca sul pulsante per eseguire la ricerca e parte la chiamata al server per ottenere la lista di tutti i supereroi con quel nome. Queste chiamate non sono state associate agli eventi come 'onkeypress', 'onchange' ecc. per evitare chiamate un numero elevato di chiamate superflue. Si ricorda che queste chiamate al server scatenano altre chiamate ai server della Marvel, aumentando la latenza per ogni chiamata. Per questo motivo è stata adottata questa soluzione.

Una volta terminata la fase di registrazione, il **server restituisce l'uid**.

Anche la login restituisce l'uid, dopo aver inserito le credenziali corrette.

Tutte le informazioni memorizzate in fase di registrazione possono essere modificate tramite un form apposito, nella pagina 'profile.html'.

Profile Settings Credit Card

General information

First name: Gianluca Last name: Calderon

Username: gianluca03 E-mail: gianluca.calderonrivera@studenti.unimi.it

Registration Date: 2024-09-05

Credits: 0

Total Cards: 0/759

Delete Account

Spider-Man (Noir) Favorite Hero

Security

New password Verify password

Save Changes Change Password

(Schermata profile, form per la gestione del profilo utente).

È inoltre possibile **eliminare** il proprio **profilo**. Questo comporta anche l'eliminazione **definitiva** di tutte le carte che possiede l'utente, e le proposte di scambio pubblicate.

Per eseguire il **logout** è sufficiente schiacciare sul pulsante 'Logout' all'interno del menù, e partirà una funzione che eliminerà l'uid dal localStorage, e reindirizzerà l'utente all'homepage (a questo punto la visione delle pagine sarà limitata, in quanto non è più 'loggato').

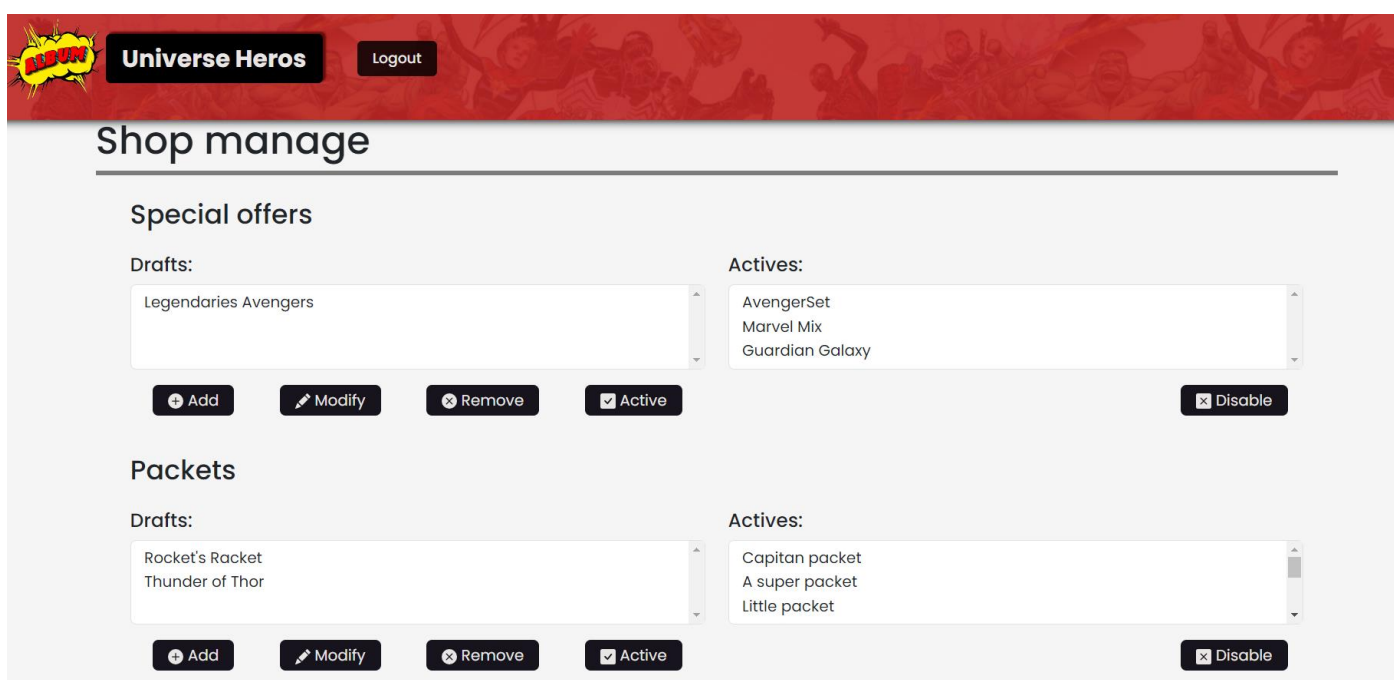
Infine, all'interno dell'applicazione si possono avere due tipi di utenti: utenti che utilizzano l'album che si registrano tramite form, e utenti admin.

Gli admin hanno un'altra visibilità dell'applicazione. Un'utente admin può solamente gestire i pacchetti di figurine dello shop, e pubblicare dei post (articoli) nell'homepage.

Admin

Gli admin sono degli utenti in grado di **gestire i pacchetti** dello shop, le **immagini** e gli **articoli** presenti nell'homepage.

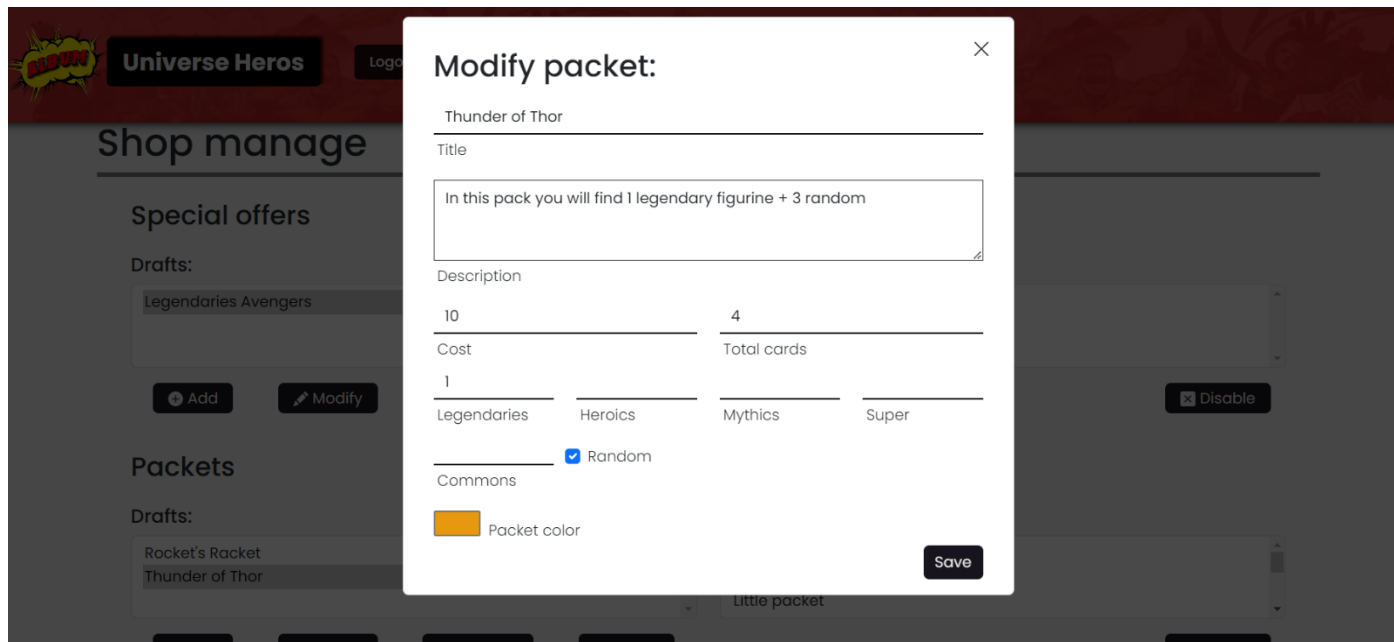
L'admin può creare **due tipi di pacchetti**, quelli **speciali**, ovvero dei pacchetti che contengono delle offerte speciali (decisi dall'admin), e i pacchetti **normali**. Tutti i pacchetti vengono gestiti nel seguente modo: prima che un pacchetto venga reso disponibile nello shop, viene memorizzato nella collection 'Drafts_packets'. In questo modo il pacchetto viene semplicemente creato ma non pubblicato. Quando un pacchetto si trova nello stato di **draft**, l'admin può **modificare** le proprietà di tale **pacchetto** (ad esempio cambiare il prezzo, il numero di carte o le rarità che si possono trovare). In questa fase è un admin può lasciare il pacchetto in draft, **attivarlo**, e di conseguenza **renderlo visibile nello shop**, oppure **eliminarlo definitivamente**. Una volta pubblicato il pacchetto è possibile disabilitarlo, eliminandolo dalla collection dei pacchetti attivi, e rimettendolo all'interno della collection dei pacchetti in draft.



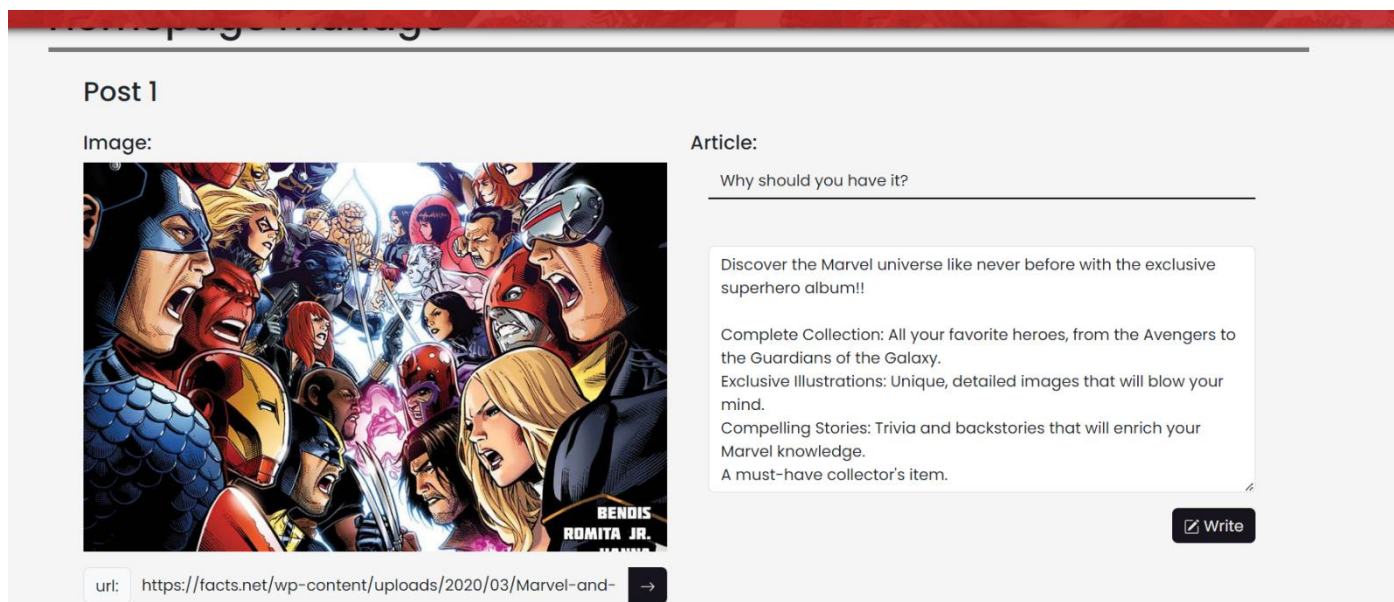
(Schermata admin, gestione dello shop e dell'homepage da parte dell'admin).

Anche la gestione dei singoli pacchetti è stata progettata per renderla il più **flessibile** possibile: un pacchetto ha un titolo, una descrizione in cui viene scritto che cosa si può trovare all'interno di un pacchetto, il costo, il numero totale di carte che si possono trovare e, il numero di carte trovabili per ogni tipo di rarità. Il **numero totale di carte indicato deve corrispondere alla somma di carte che è possibile trovare per ogni tipo di rarità**, ad esempio, se indico 5 carte totali, posso indicare 3 carte comuni, 1 eroica e 1 leggendaria. Tuttavia, abilitando il pulsante '**Random**', è possibile rendere la generazione delle carte mancanti in maniera casuale al server, nel momento

dell'acquisto di un pacchetto da parte di un'utente. Ad esempio, se indicassi 5 carte totali e 1 leggendaria, attivando l'opzione random, sarà il server, al momento dell'acquisto di un pacchetto da parte di un qualsiasi utente, a scegliere casualmente il tipo delle altre 4 carte.



(Schermata admin, modifica di un pacchetto presente nei 'drafts').



(Schermata admin, gestione dell'articolo numero 1 dell'homepage).

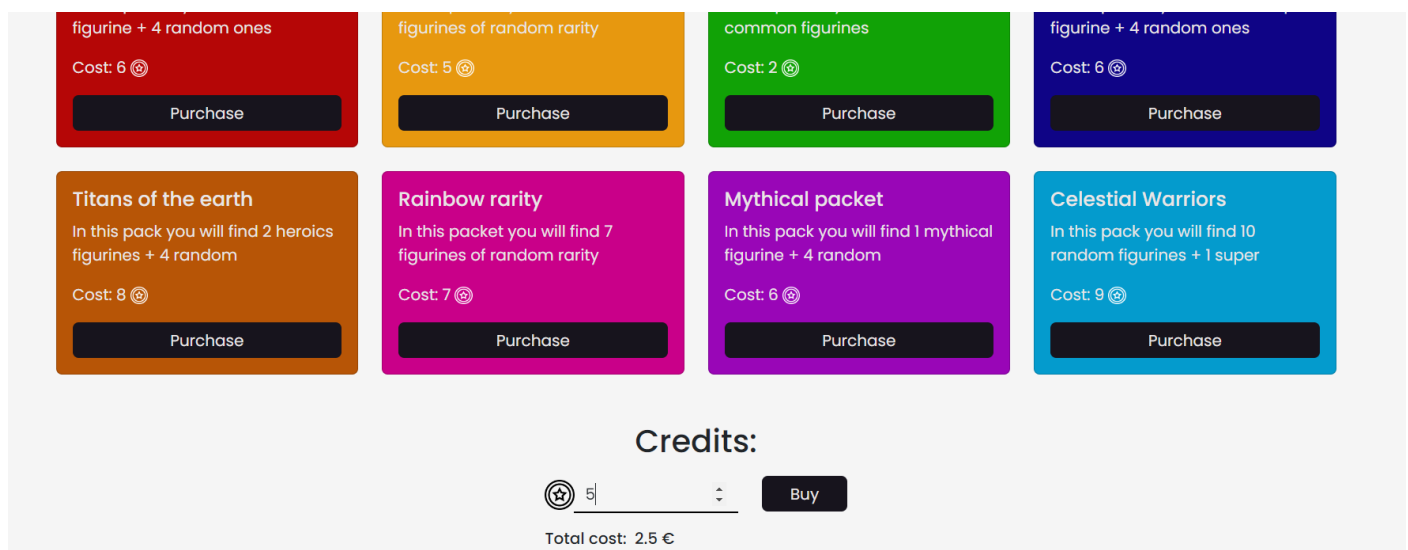
Al momento nell'applicazione web sono presenti due admin con le seguenti credenziali:

- username: admin password: &Admin123&
- username: adminGianluca password: &AdminGian123&

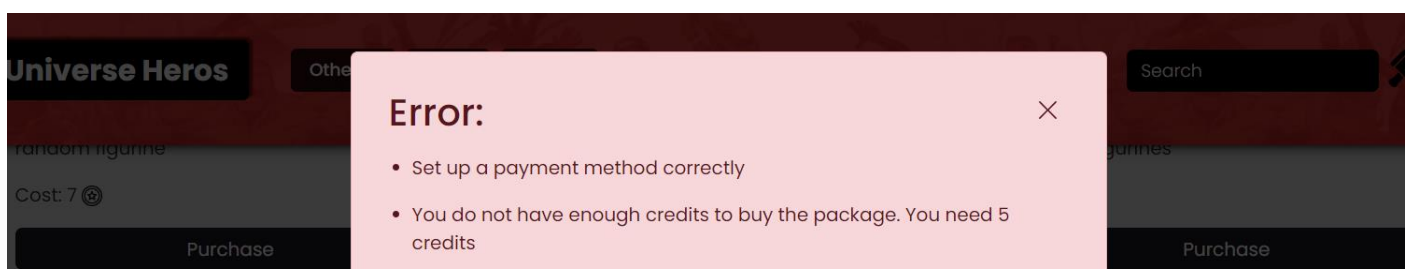
Acquisti

All'interno dell'applicazione è presente una pagina ('shop.html'), in cui è possibile **comprare** dei pacchetti di **figurine**, in cambio di crediti. È possibile acquistare i **crediti** sempre all'interno dello shop, ma è necessario che venga prima **definito un metodo di pagamento**. Le informazioni inerenti al metodo di pagamento si possono trovare all'interno del profilo ('profile.html'), dove è possibile aggiornare o aggiungere i dati per procedere agli acquisti. Un credito corrisponde a € 0,50. Si ricorda che è una simulazione, per cui queste informazioni verranno semplicemente aggiunte tra i dati di un'utente nella collection 'Users'. Nel momento dell'acquisto verrà semplicemente controllato che i campi non risultino vuoti, ma che siano formattati correttamente, per simulare il più possibile una transazione bancaria. Durante la **generazione delle carte** di un pacchetto il server si assicura che ogni carta abbia un'immagine valida, se così non fosse continua l'interazione con i server della Marvel per ottenere una carta che non contenga nel path la stringa 'image_not_available'.

Solo gli utenti con un account valido e che non siano admin, possono effettuare acquisti.



(Schermata shop, acquisto di crediti e/o pacchetti).



(Schermata shop, errore durante l'acquisto di un pacchetto per non aver impostato un metodo di pagamento. Questi messaggi d'errore vengono inviati dal server).

Profile Settings **Credit Card**

Payment settings

Username:
gianluca03

E-mail:
gianluca.calderonrivera@studenti.unimi.it

Registration Date:
2024-09-05

Credits:
0

Total Cards:
0/759

[Delete Account](#)

Name holder: Gianluca

Surname holder: Calderon

Card number: 1234567891023

Expire date: 04/28

CVV:

Logos: VISA, MasterCard

[Save Changes](#)

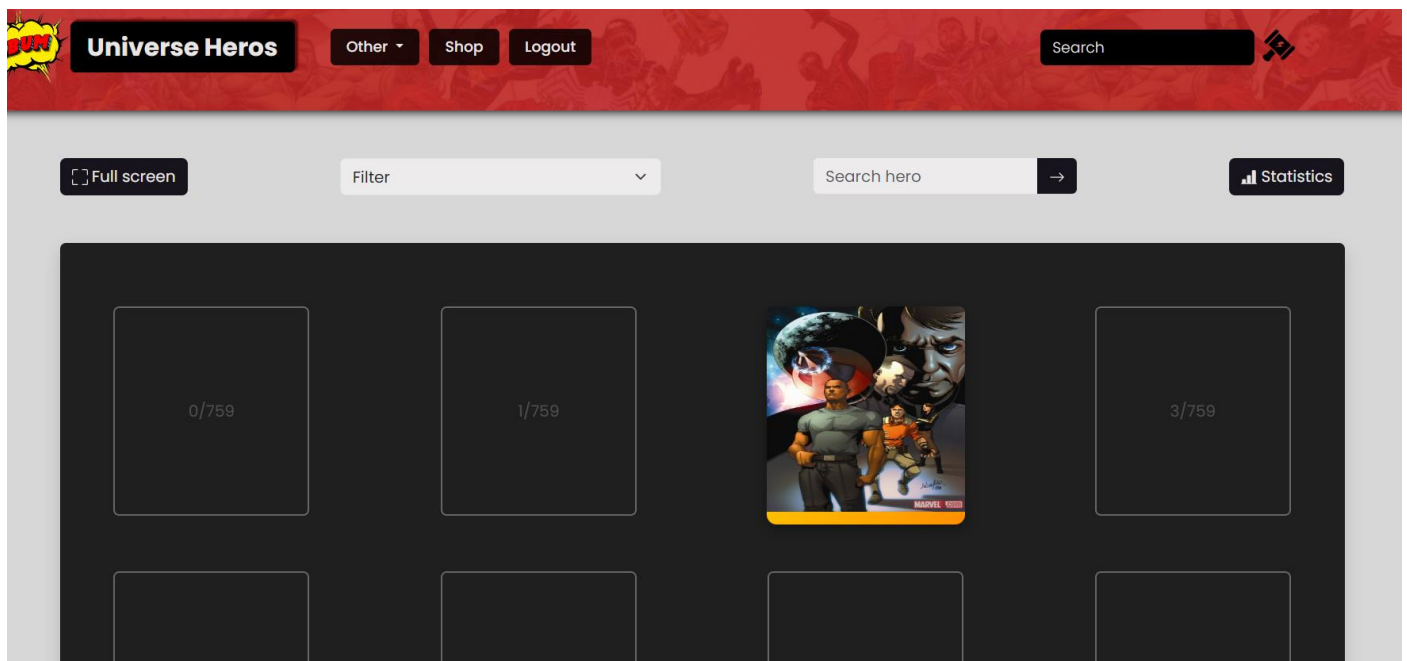
(Schermata profile, gestione metodo di pagamento).

Per rendere la gestione della carta di credito più realistica, sono stati progettati dei meccanismi per eseguire dei controlli reali sulla data di scadenza della carta di credito (controllando semplicemente il giorno corrente), il 'Card Number', 'CVV' ecc. tramite apposite regex.

Un utente può inoltre vendere delle carte che possiede, la cui quantità sia maggiore di 1 (doppioni). Questa azione, tuttavia, non viene gestita come se fossero acquisti normali, bensì nell'ambito dei 'Trades', una pagina dedicata (con dei meccanismi dedicati) in cui è possibile scambiare o vendere figurine. Successivamente verrà approfondito questo argomento.

Album

L'album è la parte principale dell'applicazione, ed è organizzato nel seguente modo: All'interno del server è presente una funzione (al momento commentata), in grado di prelevare tutte le carte dal server della Marvel fino ad un offset da me definito (1000), e contare solo le **carte valide**. Per carte valide si intendono le **carte con delle immagini disponibili**. Ho preferito dare un numero predefinito di carte, ovvero prendere i primi mille offset di eroi dal server della Marvel. Gli offset totali disponibili che offre la Marvel solo al momento 1560 circa. Al termine dell'esecuzione della funzione sopra menzionata, sono giunto alla conclusione che le carte valide, dei primi **1000 offset**, sono in totale **759**. Avendo così definito un numero massimo di carte, sono riuscito a simulare un vero e proprio album delle figurine, assegnando ad ogni carta una pagina, un indice all'interno della pagina, e definendo così il numero massimo di pagine. Tuttavia, il **client non è legato a questo numero specifico**. Ogni volta che il server invia dei dati inerenti alle carte, come nell'api `"/cards/:uid"` get, invio pure il numero totale di carte. Il client, una volta ricevuto questo parametro, **adatterà** l'applicazione nel modo



(Schermata album).

```
html_element+=`<div class="col-lg-3 col-md-6 d-flex justify-content-center ">
  <div class="mb-4 custom-w-lg-75 custom-w-md-50">
    <div class="custom-font-light-grey m-3 rounded custom-empty-card">${(page*20)+i}/${album_cards.maxTotalCards}</div>
  </div></div>`
```

(Esempio della funzione che crea i riquadri delle carte nell'album, che si adatta in base al numero totale di carte passato dal server, nel nostro caso 759).

più corretto.

Le carte vengono inviate dal server a **gruppi di 20**. Ad esempio, la pagina 1 corrisponde all'**offset 0** degli eroi che un'utente possiede, e di conseguenza, il server invia tutte le carte il cui **indice è compreso** tra 0 e 19. Cambiando pagina, l'offset verrà modificato.

```
//Prendo l'indice della prima carta di una pagina
firstIndexCards = offset*20;

const client = await client_to_db.connect();

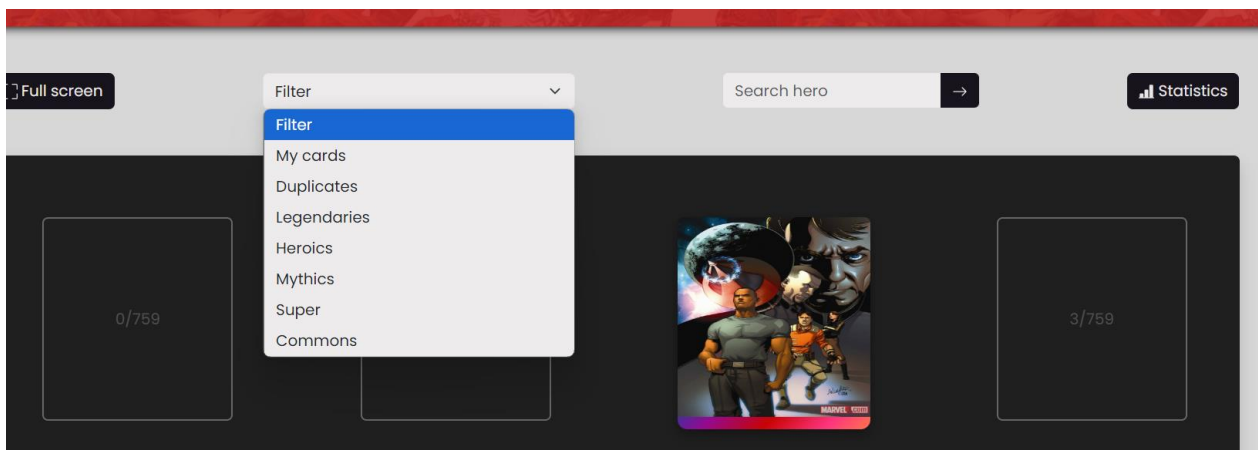
try{

    //Se non è indicato alcun filtro, procedo con l'estrazione delle carte normalmente,
    //restituendo le carte di una determinata pagina (offset)
    if((filter==null || filter ==undefined || filter=='') && (hero==null || hero==undefined || hero==''))
        cards = await client.db(db_name).collection("Cards").find({user:uid, index:{$gte:firstIndexCards, $lt:(firstIndexCards+20)}}).toArray();
    else if((filter==null || filter ==undefined || filter=='') && (hero!=null || hero!=undefined || hero!=''))
        allCards = await client.db(db_name).collection("Cards").find({user:uid, name: { $regex: new RegExp(hero, 'i') } }).toArray();
}
```

(Estratto di codice della funzione 'getPageCards()' del server contenuto nel file index.js. Queste righe di codice mostrano come funziona la logica implementata per estrarre le carte di un'utente dal DB, in base all'offset passato).

Inoltre, è possibile passare dei **filtri** per ottenere solo delle determinate carte:

- **my_cards**: Questo filtro restituisce tutte le carte che possiede un'utente. Nell'album non verranno mostrati i template di carte nere vuote.
- **duplicates**: Restituisce tutti i doppioni che possiede un'utente.
- **legendary**: Restituisce tutte le carte leggendarie che possiede un'utente
- **heroic**: Restituisce tutte le carte eroiche che possiede un'utente
- **mythic**: Restituisce tutte le carte mitiche che possiede un'utente
- **super**: Restituisce tutte le carte super che possiede un'utente
- **common**: Restituisce tutte le carte comuni che possiede un'utente

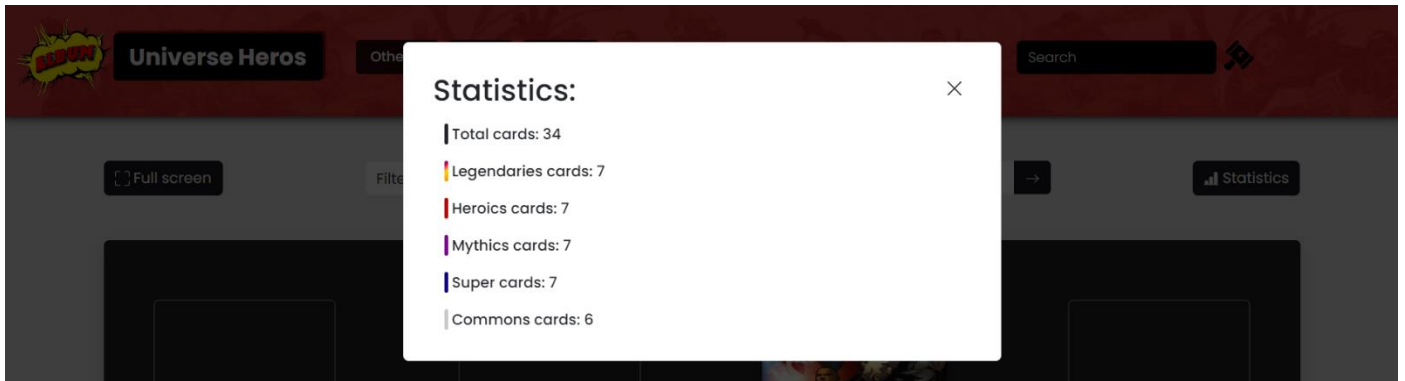


(Schermata album, scelta dei filtri da applicare).

Per rendere la visione delle carte più granulare, è presente anche una **barra di ricerca** che permette di **cercare delle sequenze di caratteri** contenute all'interno dei nomi degli eroi delle carte un'utente possiede. Ad esempio, se un'utente digitasse 'Spider-',

il server gli restituirà tutte le carte in suo possesso, il cui nome dell'eroe contiene la stringa 'Spider-'.

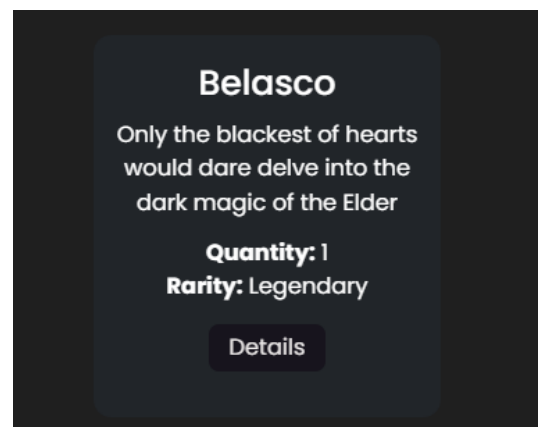
È presente anche un pulsante '**Statistics**' in grado di fornire tutte le statistiche riguardo alle carte che un'utente possiede, come ad esempio il numero totale di carte, il numero totale di carte leggendarie ecc.



(Schermata album, visione delle statistiche di un'utente).

Infine, per migliorare la visibilità dell'album, è stato creato un pulsante ('Full screen') in grado di mettere il "focus" sull'album, ovvero di far sparire il menu e il footer per posizionare l'**album a tutto schermo**.

Per vedere i **dettagli di una carta**, basta passarci su con il cursore, e la carta si capovolgerà mostrando alcuni dettagli, come una breve visione della descrizione, la rarità, il numero di carte che un'utente possiede per quel tipo di carta (cioè ci sono dei doppioni di quella carta), e un pulsante per aprire la pagina dedicata ai dettagli. Da dispositivi mobile basta cliccare sulla carta. Il pulsante dettagli effettua una chiamata al server, inviando l'id dell'eroe, e il server a sua volta effettuerà una chiamata al server della Marvel.



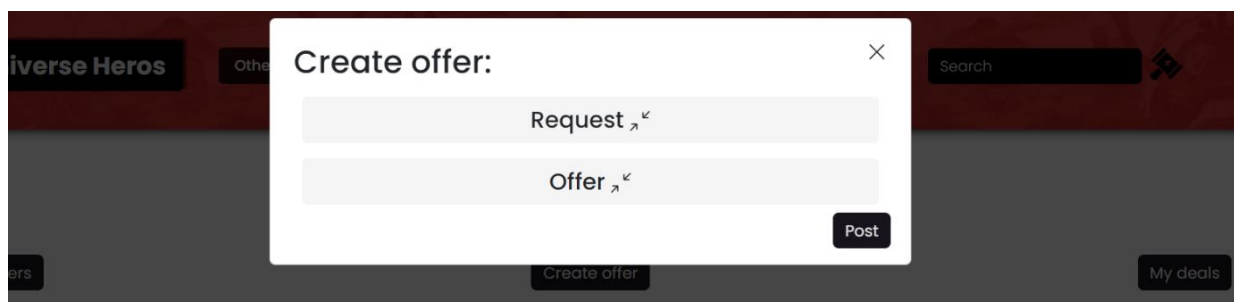
(Schermata album, parte posteriore di una carta).

Gestione scambi

Gli scambi vengono gestiti nella pagina 'Trades.html'. In questa sezione, oltre a proporre degli scambi, è possibile vendere le proprie carte ad altri utenti. Un'utente può creare delle **proposte di scambio**. Le proposte di scambio sono strutturate nel seguente modo:

- **Request:** In questa sezione vengono inserite le richieste dell'utente che crea la proposta di scambio. La richiesta può contenere:
 - **Crediti:** In questo caso si parla di vendita. Un'utente può richiedere dei crediti per una o più carte
 - **Carte generiche:** In questa sezione un'utente può scegliere in base alla rarità della carta, e non una carta specifica. Ad esempio, posso richiedere 1 carta leggendaria e una mitica, non importa quale.
 - **Carte specifiche:** In questa sezione un'utente indica l'id delle carte specifiche che desidera. Ad esempio, un utente può digitare 'Spider-Man (Noir)' nella barra di ricerca, e una volta premuto sul nome, verrà memorizzato l'id di quella specifica carta.
- **Offer:** In questa sezione viene inserita l'offerta dell'utente che crea la proposta di scambio. L'offerta può contenere:
 - **Crediti:** Si possono offrire dei crediti in cambio di carte.
 - **Carte doppie:** Si possono offrire i propri doppioni.

Per entrare in questa sezione basta cliccare il pulsante 'Create Offer'.

The image shows a web interface with a dark red header. A white modal box titled 'Create offer:' is centered on the screen. It contains two text input fields: the first is labeled 'Request' with a small icon to its right, and the second is labeled 'Offer' with a similar icon. A 'Post' button is located at the bottom right of the modal. In the background, parts of the website are visible, including a search bar, a 'Create offer' button, and a 'My deals' button.

(Schermata trade, sezione per creare una proposta di scambio).

Create offer: [Close]

Request ↕

Credits: _____

Generic cards: _____

Total cards Legendaries Heroics Mythics

Super Commons

Specific cards: _____

Select a card (hero) [Dropdown] [Trash]

Search... [Search]

[+]

[Post]

(Schermata trade, richiesta).

Create offer: [Close]

Offer ↕

Credits: _____

Yours duplicate cards: _____

Select a hero [Dropdown] [Trash]

Hulkling

[+]

[Post]

(Schermata trade, offerta).

Il server prima di pubblicare la richiesta, oltre a eseguire i controlli di base con le regex, per capire se è formattato tutto correttamente, controlla che l'utente non abbia indicato due carte uguali (sia tra le 'specifics' della richiesta, sia tra i doppietti dell'offerta), e che le carte indicate esistano effettivamente nel server della Marvel.

Nel momento in cui viene creata la proposta, vengono **decurtati** eventuali **crediti** offerti e le **quantità** di carte **offerte**.

Al **caricamento della pagina** 'Trade.html', vengono effettuate delle chiamate per richiedere al server tutte le proposte di scambio disponibili. Il server restituirà tutte le proposte di scambio aperte, con l'uid del creatore diverso dall'utente che esegue la chiamata. In poche parole, un'utente non vedrà nella schermata le proprie richieste create, ma tutte le richieste degli altri utenti.

Universe Heroes [Other] [Shop] [Logout] [Search] [Flag]

Available trades

[My offers] [Create offer] [My deals]

@calde003

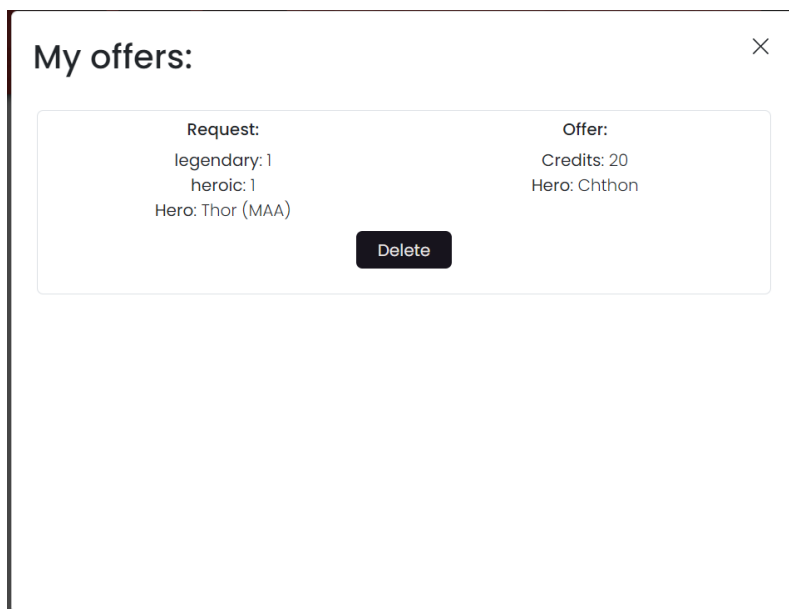
Request:
legendary: 1
heroic: 1
Hero: Thor (MAA)

Offer:
Credits: 20
Hero: Chthon

[Deal]

(Schermata trade con una proposta di scambio).

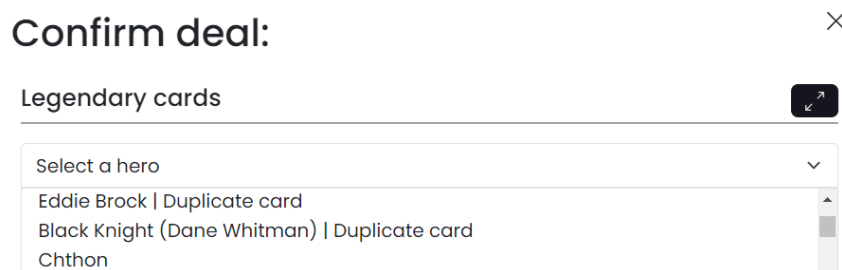
Nonostante non sia possibile vedere le proprie richieste all'interno della sezione principale, è possibile vedere le proprie offerte cliccando il pulsante 'My offers'. In questa scheda comparirà la lista di **tutte le proposte di un'utente**, e un pulsante se si desidera eliminarla.



(Schermata trade, richieste create da un'utente).

Se si elimina una proposta di scambio, i **crediti e/o le carte, verranno restituiti**. Ad esempio, se un'utente elimina una proposta in cui offre 10 crediti e una carta di Thor, una volta eliminata la proposta verranno aggiunti i 10 crediti, e incrementata la quantità della carta 'Thor' di 1.

Non è possibile offrire carte che non siano doppioni, tuttavia è possibile **scambiare** delle proprie carte (anche non doppioni), in una proposta di scambio aperta. Ad esempio, se all'interno della sezione Trades è presente una proposta di scambio aperta, in cui chiedono una carta qualsiasi leggendaria, è possibile offrire una carta anche se la quantità è pari ad 1. Per avvisare l'utente durante la conferma dell'affare, accanto ad ogni carta della lista è presente la voce 'Duplicate card' nel caso la carta abbia una quantità maggiore di 1 (doppione), altrimenti sarà presente solo il nome dell'eroe, come mostrato nella seguente figura.



Cliccando il pulsante 'MyDeals' è possibile vedere **tutti gli scambi effettuati** con gli utenti.

All'interno dell'applicazione è presente un'utente che possiede delle carte (inclusi i doppioni), e con delle proposte di scambio già create. Di seguito le credenziali:

- Username: gianluca03 Password: &Gian123&