

Progetto programmazione web e mobile

2023/2024

Questa relazione documenta la progettazione, sviluppo e implementazione di un sito web chiamato **marvel-album** con una separazione chiara tra Frontend e Backend. Il progetto è stato realizzato sulla base delle indicazioni date nel documento **PWM_project_23_24.pdf** pubblicato su Ariel.

Contents

Pubblicazione risorse	2
no-cookie branch	2
main branch	2
Ambiente locale.....	3
Backend.....	3
Frontend	3
Funzionamento applicazione web	5
Navbar	5
Login.....	5
Signin.....	6
Profile	7
Shop	8
Album	11
Trades.....	13
Card Details.....	18
Docs	19
Scelte implementative	20
Standardizzazione Alert	20
Estrazione di una carta casuale	20
Salvataggio delle carte.....	20
Struttura database	21
JWT Token	21
Cookie	22
Media Query.....	22
Sviluppi futuri	22

Pubblicazione risorse

Il progetto, con il codice consegnato, è stato implementato al seguente [link](#). Questo per rendere il sito web disponibile su un ambiente di produzione stabile e poter testare il funzionamento senza la necessità di creare un ambiente locale.

È importante sapere che il server potrebbe richiedere alcuni secondi per avviarsi in seguito alla prima richiesta ricevuta. Di conseguenza, la prima chiamata alle API potrebbe presentare un lieve ritardo, mentre le richieste successive saranno elaborate con tempi di risposta più rapidi.

Le API del sistema, che costituiscono il server per la gestione del backend, sono pubblicate al dominio <https://marvel-album.onrender.com> questo perché non è stata possibile la creazione di un sottodominio nell'url indicato prima dove al contrario è caricato il frontend.

Il database mongoDB è invece ospitato su un cluster Atlas.

Il codice del progetto è possibile trovarlo pubblicato anche su [Github](#) dove saranno visibili due branch, che indicano le due versioni del progetto.

no-cookie branch

Questa è la versione del progetto consegnata e quella rilasciata sull'ambiente di produzione.

main branch

In questo branch è presente una variante strutturale nella comunicazione tra backend e frontend che verrà descritta brevemente più avanti.

Ambiente locale

I prerequisiti per far avviare in ambiente locale il progetto sono:

- Creazione del proprio profilo nel Marvel Developer Portal
- Database mongoDB
- Node.js installato sul proprio dispositivo

Backend

I passaggi per implementare il backend dell'applicazione in localhost sulla porta 3001 sono:

per prima cosa vanno inseriti alcuni dati nel file con percorso **/backend/.env**.

- **JWT_SECRET** chiave con la quale verrà verificato il jwt token
- **DB_CONNECT** stringa di connessione al database MongoDB
- **ORIGIN** url del frontend, serve ad indicare l'url di provenienza delle chiamate api per le CORS policy. Per l'ambiente locale si può lasciare il valore di default (http://localhost:3000)
- **PORT** numero porta su cui verrà esposto il backend su localhost

successivamente da linea di comando del terminale, raggiungere percorso file del progetto ed eseguire i seguenti comandi:

- cd backend
- npm install
- npm start

Tecnologie principali utilizzate:

- express: per la creazione delle funzionalità del server
- jsonwebtoken: tecnologia per lo scambio di informazioni
- bcryptjs: per la crittografia dei dati
- swagger: per la documentazione e testing delle API
- mongoose: per semplificare l'interazione con i database MongoDB

Frontend

Per avviare il frontend dell'applicazione in localhost sulla porta 3000 i passaggi sono:

anche qui per prima cosa vanno inseriti alcuni dati nel file con percorso **/frontend/config.json** per permettere il corretto utilizzo delle Marvel API e del backend.

- **publicKey**: chiave pubblica fornita nel proprio profilo Marvel Developer Portal
- **hash**: stringa della funzione md5, maggiori informazioni [qui](#)

- **backendUrl**: stringa che specifica l'url del backend

successivamente da linea di comando del terminale, raggiungere percorso file del progetto ed eseguire i seguenti comandi:

- cd frontend
- npm install
- npm start

Tecnologie principali utilizzate:

- **React**: per la creazione di un'applicazione a pagina singola
- **react-bootstrap**: per l'utilizzo di alcune strutture come bottoni o griglie
- **scss**: per semplificare la gestione delle proprietà css
- **mui-material**: per l'utilizzo di alcune strutture come bottoni o griglie
- **axios**: per facilitare la gestione delle chiamate API

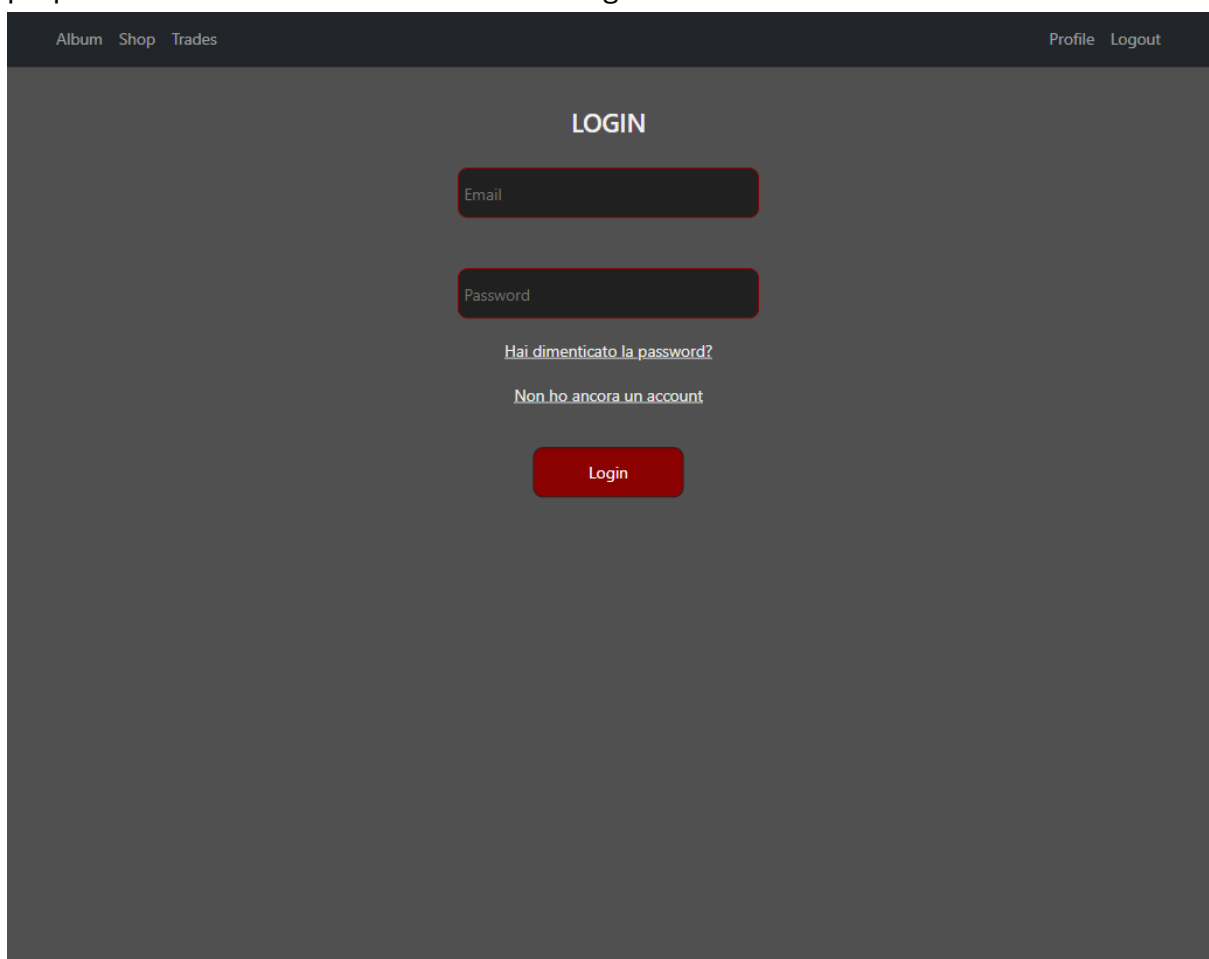
Funzionamento applicazione web

Navbar

Fissata in alto in ogni pagina è posizionata la navbar che gestisce la navigazione in tutte le pagine principali dell'applicazione: Album, Shop, Trades, Profile e Logout (che scollega l'utente dall'applicazione e riporta alla Login page).

Login

Una volta avviata l'applicazione si viene reindirizzati alla pagina di Login (path </login>). Per accedere al proprio album è necessario inserire la mail e password corrette del proprio account e cliccare sul bottone di Login rosso.

The screenshot shows a web application's login page. At the top, there is a dark navigation bar (navbar) with links for 'Album', 'Shop', and 'Trades' on the left, and 'Profile' and 'Logout' on the right. The main content area has a dark gray background. In the center, the word 'LOGIN' is displayed in white capital letters. Below it, there are two input fields: one for 'Email' and one for 'Password', both with dark borders and light gray text. Under the password field, there are two links: 'Hai dimenticato la password?' and 'Non ho ancora un account', both in a light blue color. At the bottom of the form, there is a prominent red button with the word 'Login' in white text.

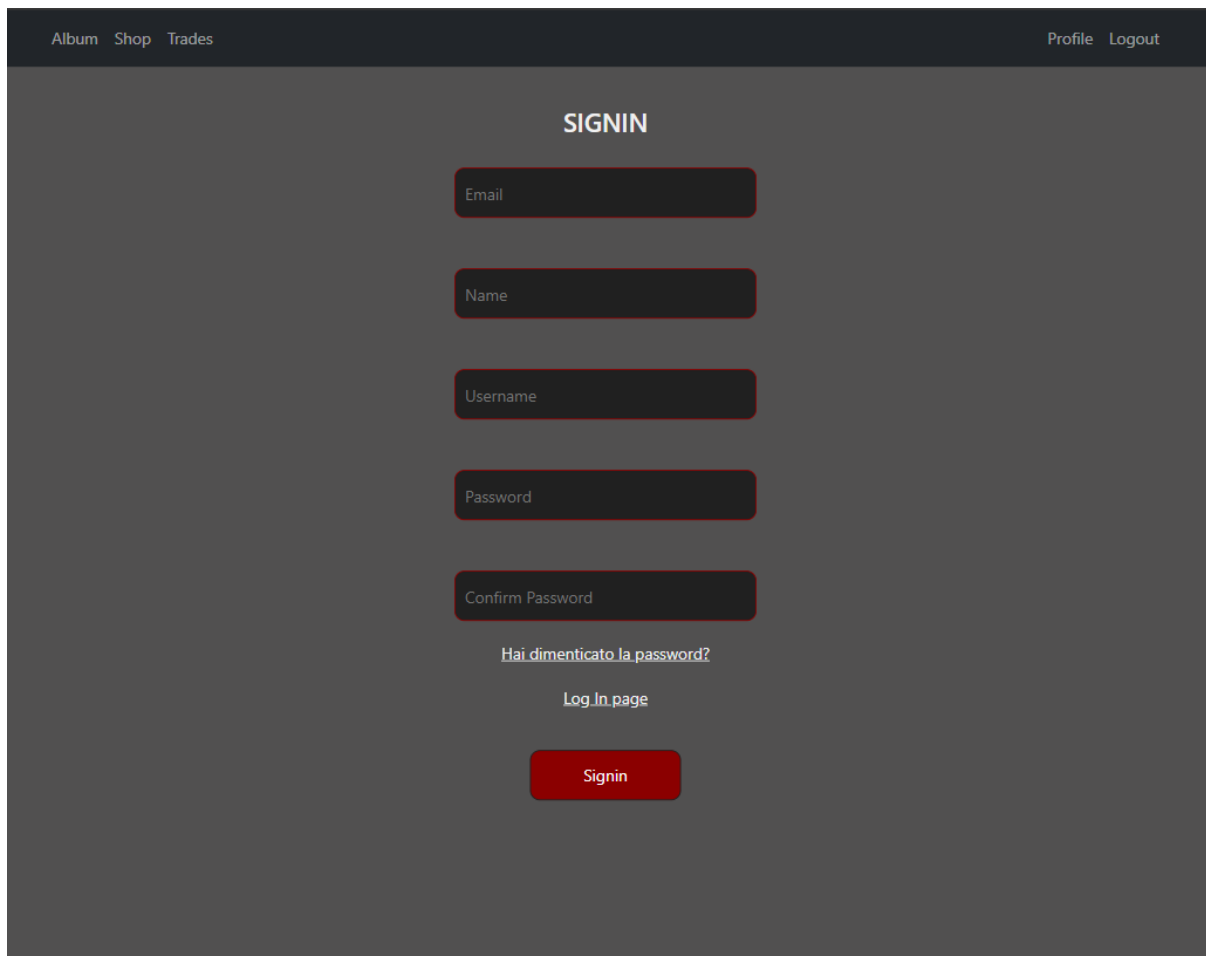
Lato frontend c'è il controllo dei campi vuoti, i controlli di verifica mail e password sono invece lato backend.

Se non si possiede un account è possibile cliccare sulla label “Non ho ancora un account” e si verrà riportati alla pagina di Signin per la creazione del profilo.

L'unica API in questa pagina è </api/users/login> utilizza il metodo POST per non passare i dati tramite parametri ma tramite il body della richiesta ed esegue i controlli backend precedentemente citati.

Signin

In questa pagina (path </signin>) è possibile creare un nuovo account, i dati necessari sono email, nome, username, password.

The image shows a web application's 'Signin' page. At the top, there is a dark navigation bar with links for 'Album', 'Shop', and 'Trades' on the left, and 'Profile' and 'Logout' on the right. The main content area has a dark gray background. In the center, the word 'SIGNIN' is displayed in white capital letters. Below it, there are five dark gray input fields with white placeholder text: 'Email', 'Name', 'Username', 'Password', and 'Confirm Password'. Each field has a thin red border. Below the 'Confirm Password' field, there is a link that says 'Hai dimenticato la password?'. Underneath that link is another link that says 'Log In page'. At the bottom of the form area is a red button with the word 'Signin' in white text.

Lato front-end i controlli fatti sono: tutti i campi riempiti, che password e la conferma siano uguali, che la mail abbia una struttura corretta e che lo username sia formato da sole lettere minuscole.

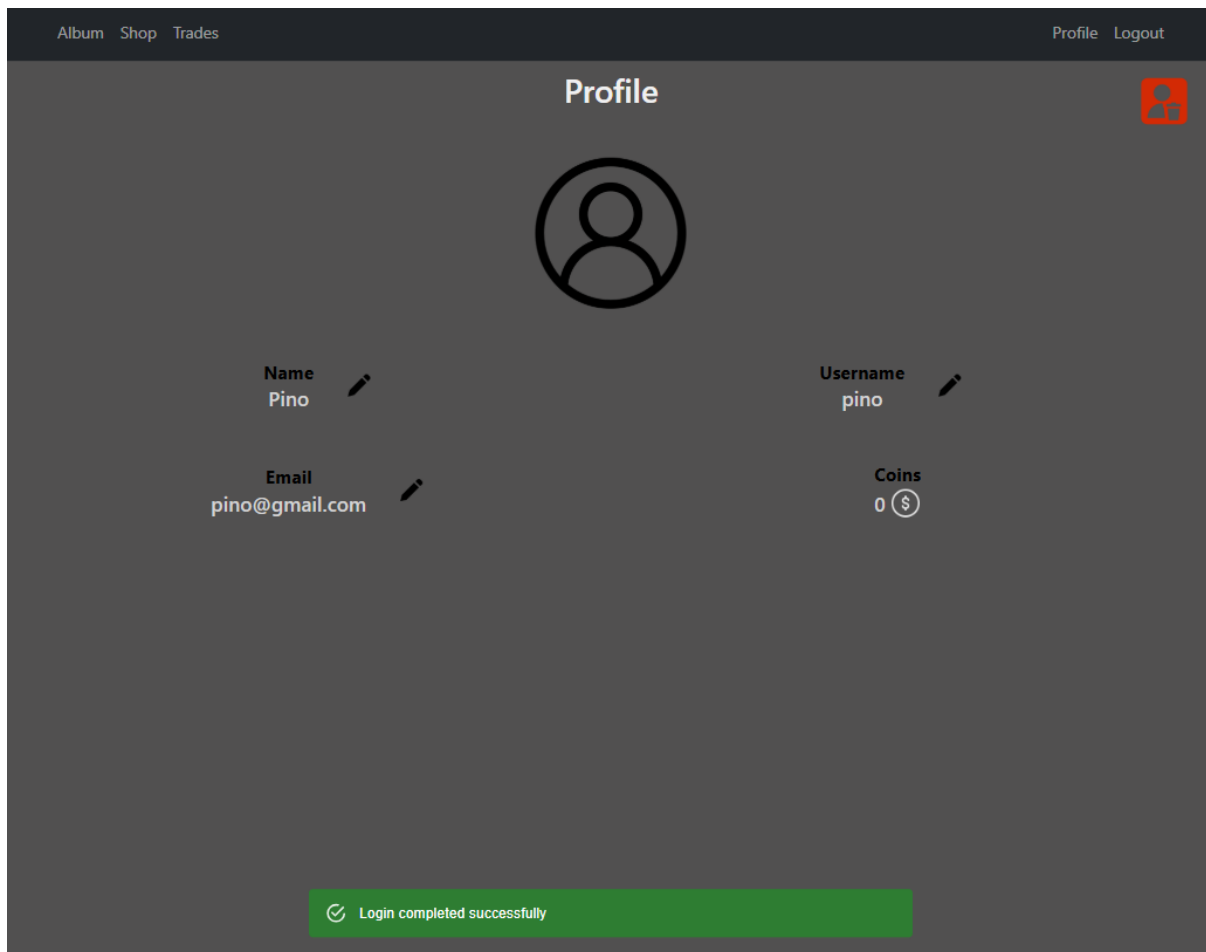
Lato back-end viene controllato che l'username passato non sia già utilizzato da altri account.

API utilizzata per la creazione sfrutta il metodo POST ed è </api/users> e serve per la creazione effettiva del profilo nel database.

La label "Log in page" se cliccata riporta alla pagina di Login dell'applicazione.

Profile

La pagina di profilo (path </profile>) permette di consultare ed eventualmente modificare le informazioni principali di un utente.



Tramite l'icona rossa in alto a destra è invece possibile eliminare l'account nel quale si è attualmente connessi, l'operazione andrà ad eliminare tutti i dati relativi ad esso nel database in modo irreversibile.

La modifica delle informazioni utente è regolamentata dagli stessi controlli effettuati nella creazione del profilo. In base al campo scelto le regole sono:

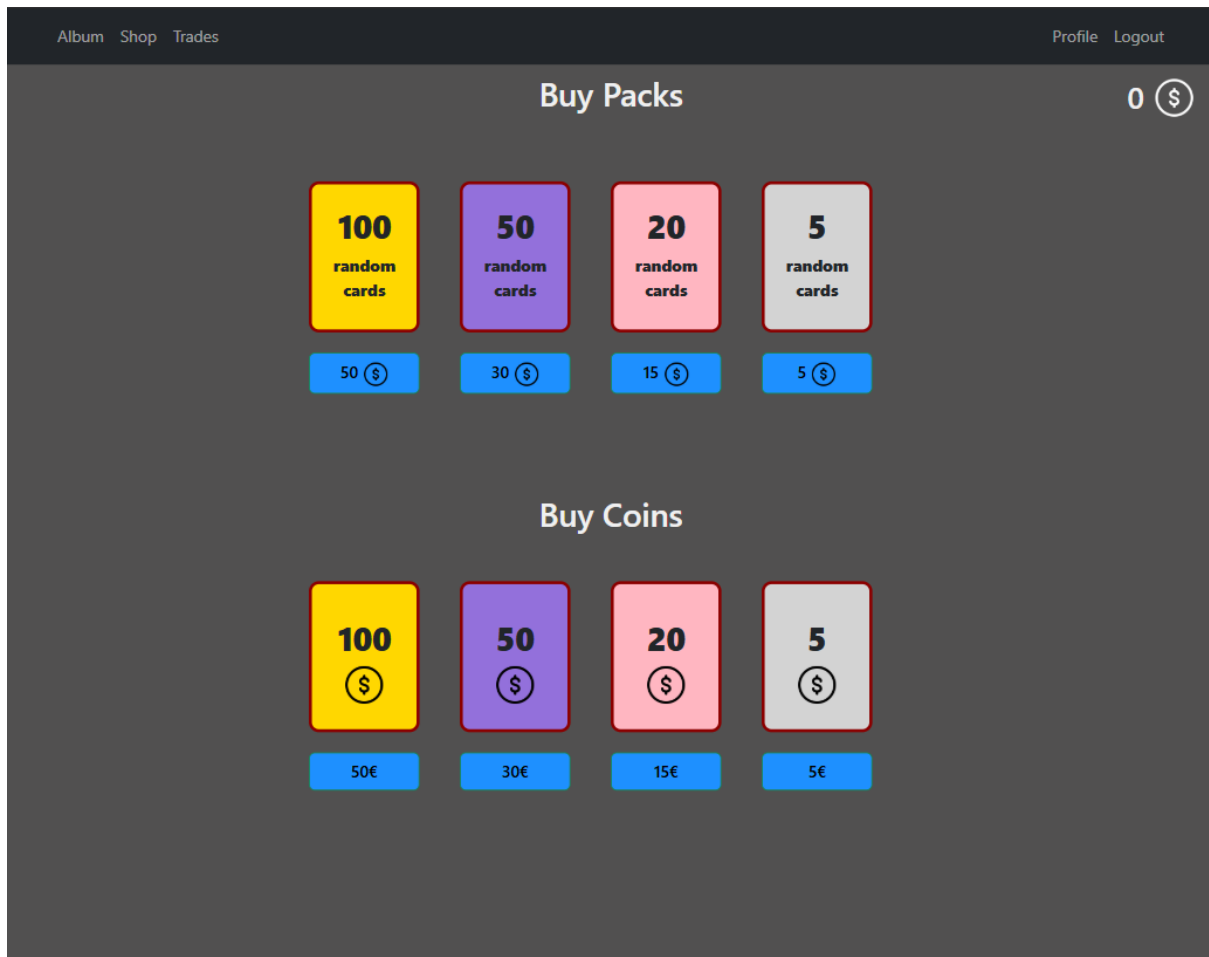
- Nome: solo lettere o spazi
- Username: solo lettere minuscole
- Email: controllo della struttura mail

Le API utilizzate sono:

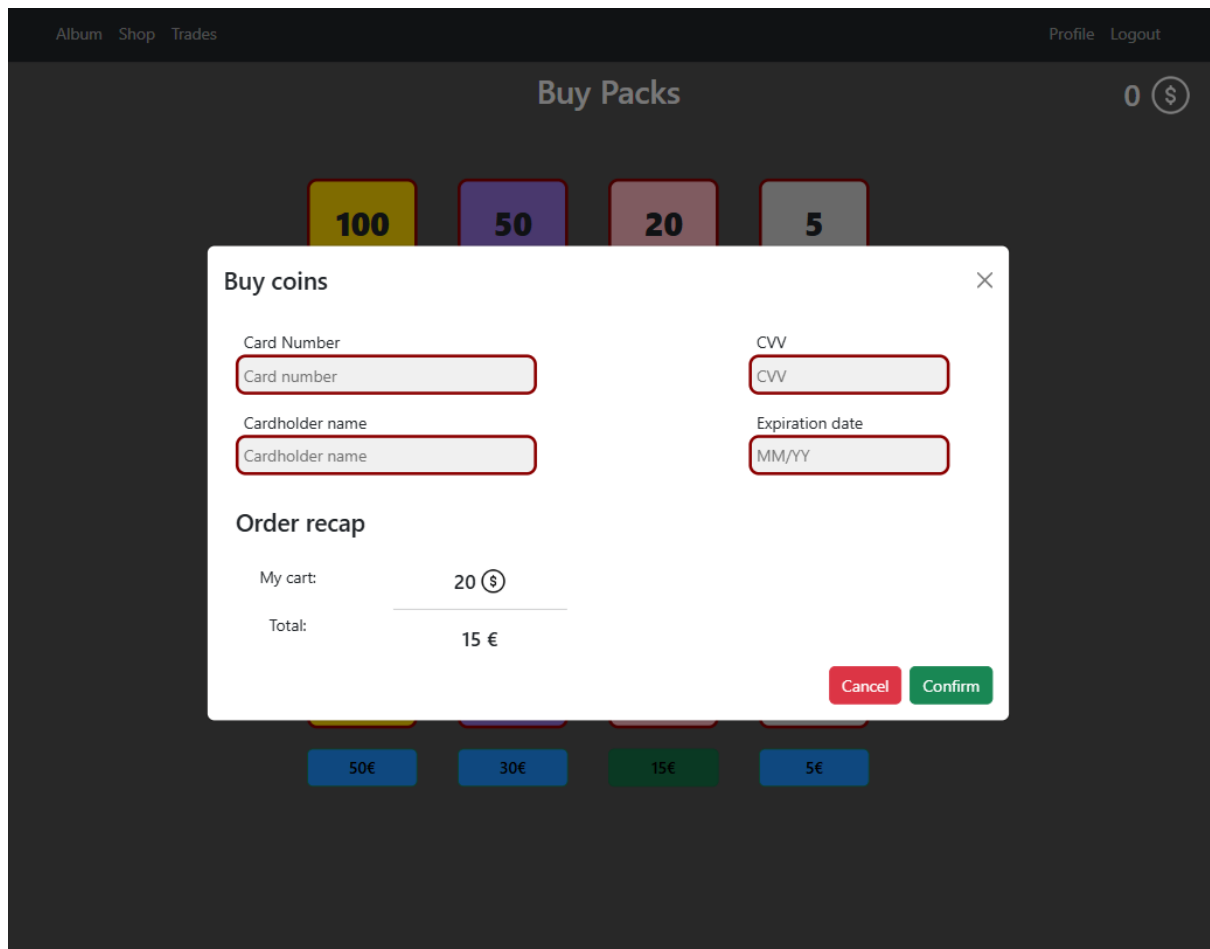
- </api/user> con metodo GET che restituisce tutti i dati principali di un utente
- </api/users> con metodo DELETE che elimina definitivamente un utente
- </api/users> con metodo PUT che va a modificare le informazioni di un utente

Shop

La pagina di Shop (path </shop>) mostra la possibilità di acquistare due tipi di pacchetti: i pacchetti di carte e i pacchetti di “Coins”. I Coins rappresentano la valuta dell’applicazione e si possono acquistare tramite l’utilizzo di una carta di credito o prepagata. I Coins che possiede l’utente sono visualizzabili in alto a destra.

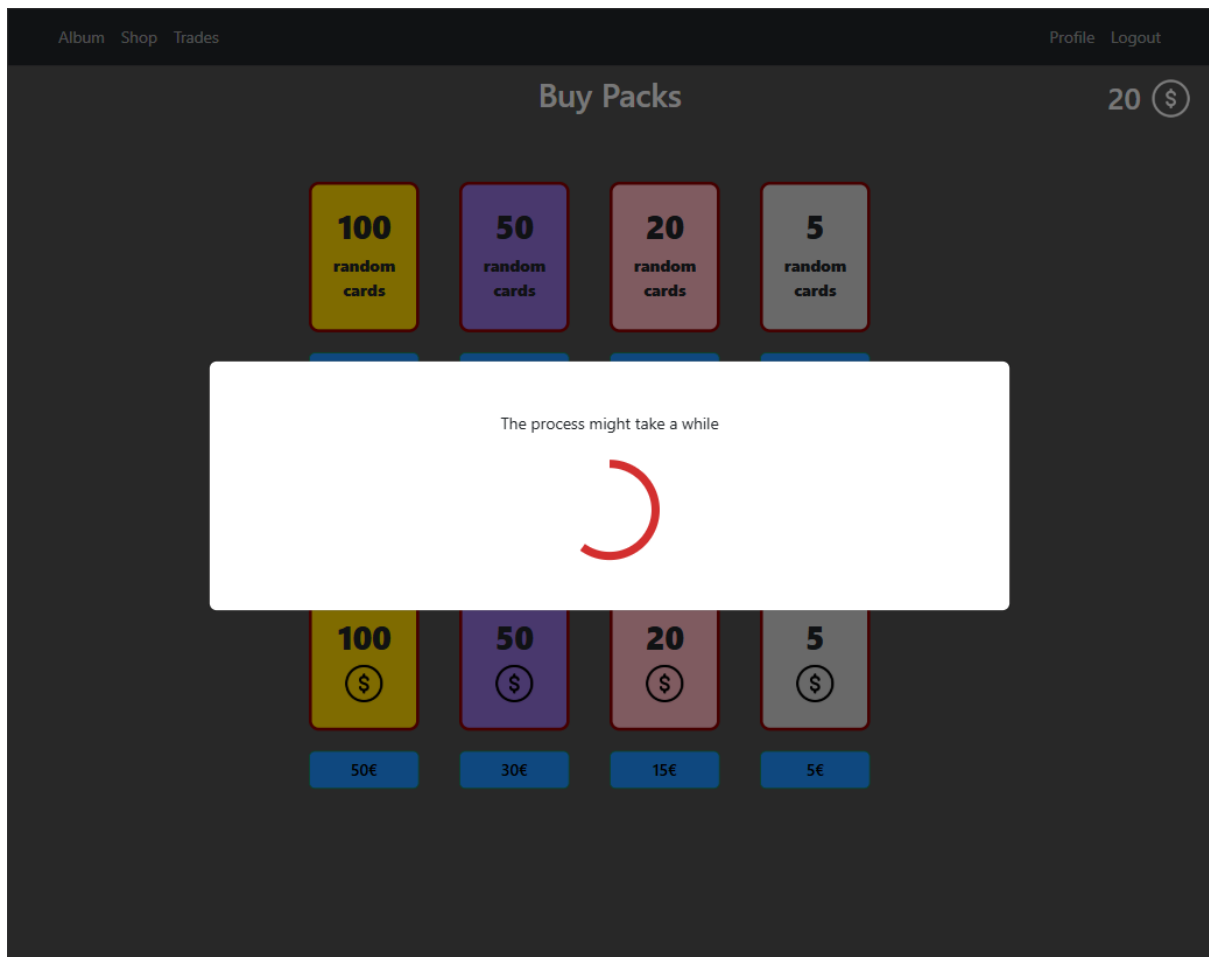


Nel progetto però non sarà necessario l'inserimento di alcun dato di carta, basterà solo il click del pulsante di successo nel modale di conferma del pacchetto Coins scelto per ottenere il numero corrispondente di Coins. Come mostrato nel seguente screenshot.



Il pacchetto carte, una volta data la conferma e controllato che l'utente abbia effettivamente il numero di Coins richiesti per l'apertura del pacchetto, avvia la procedura di estrazione casuale delle carte, che può richiedere anche diversi secondi per terminare.

Il completamento dello spinner indica lo stato percentuale dell'operazione in corso. Più alto sarà il numero di carte estratte più alto sarà il tempo di attesa per l'estrazione.

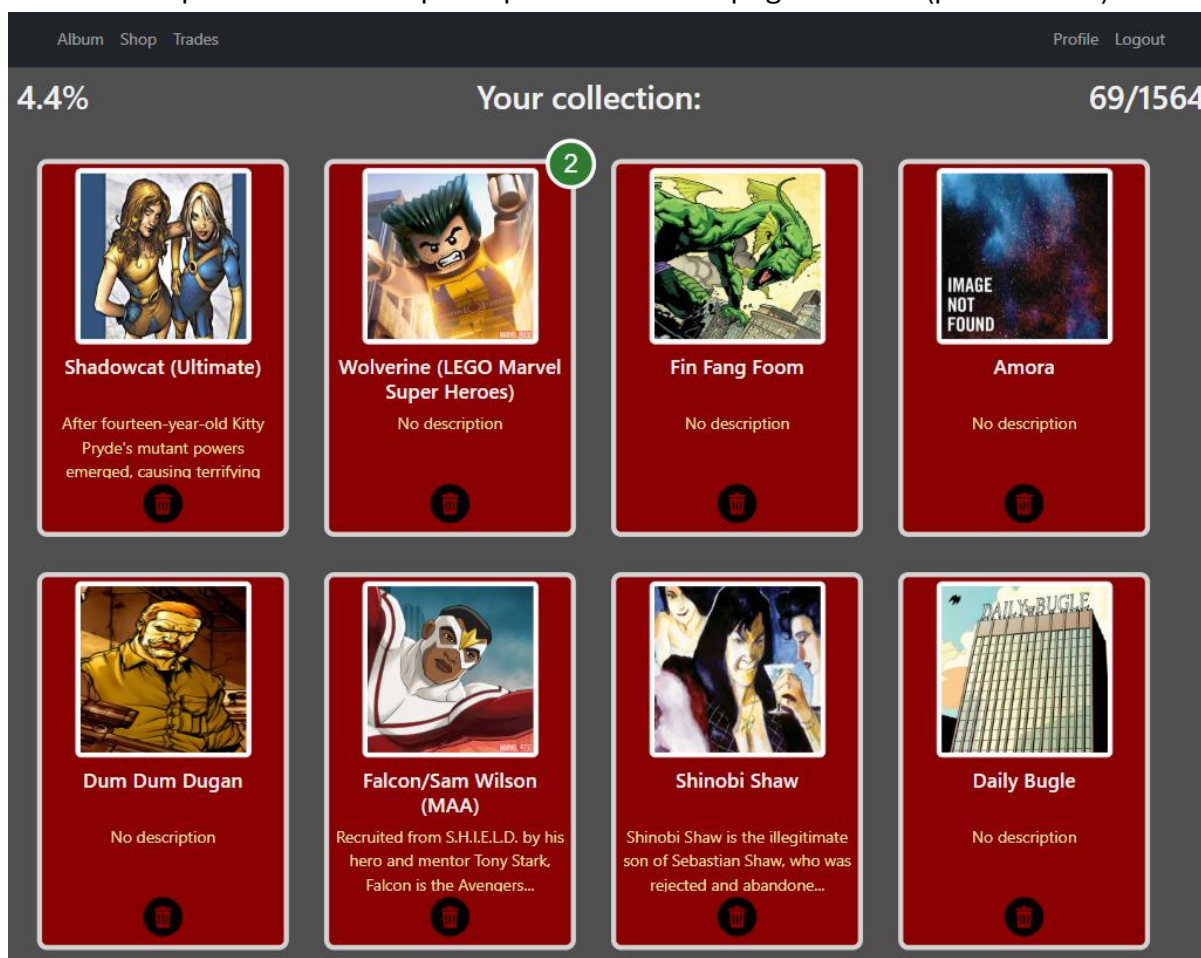


Le API utilizzate in questa pagina sono:

- [/api/user](#) con metodo GET che restituisce tutti i dati principali di un utente necessaria per reperire il numero dei coin dell'utente.
- gateway.marvel.com/v1/public/characters API marvel-portal con metodo GET per l'acquisizione dei dati delle varie carte estratte.
- [/api/users/coins](#) metodo POST per l'aggiunta o rimozione dei coins acquistati dall'utente
- [/api/users/add-card](#) metodo POST per l'aggiunta di un personaggio estratto all'album dell'utente

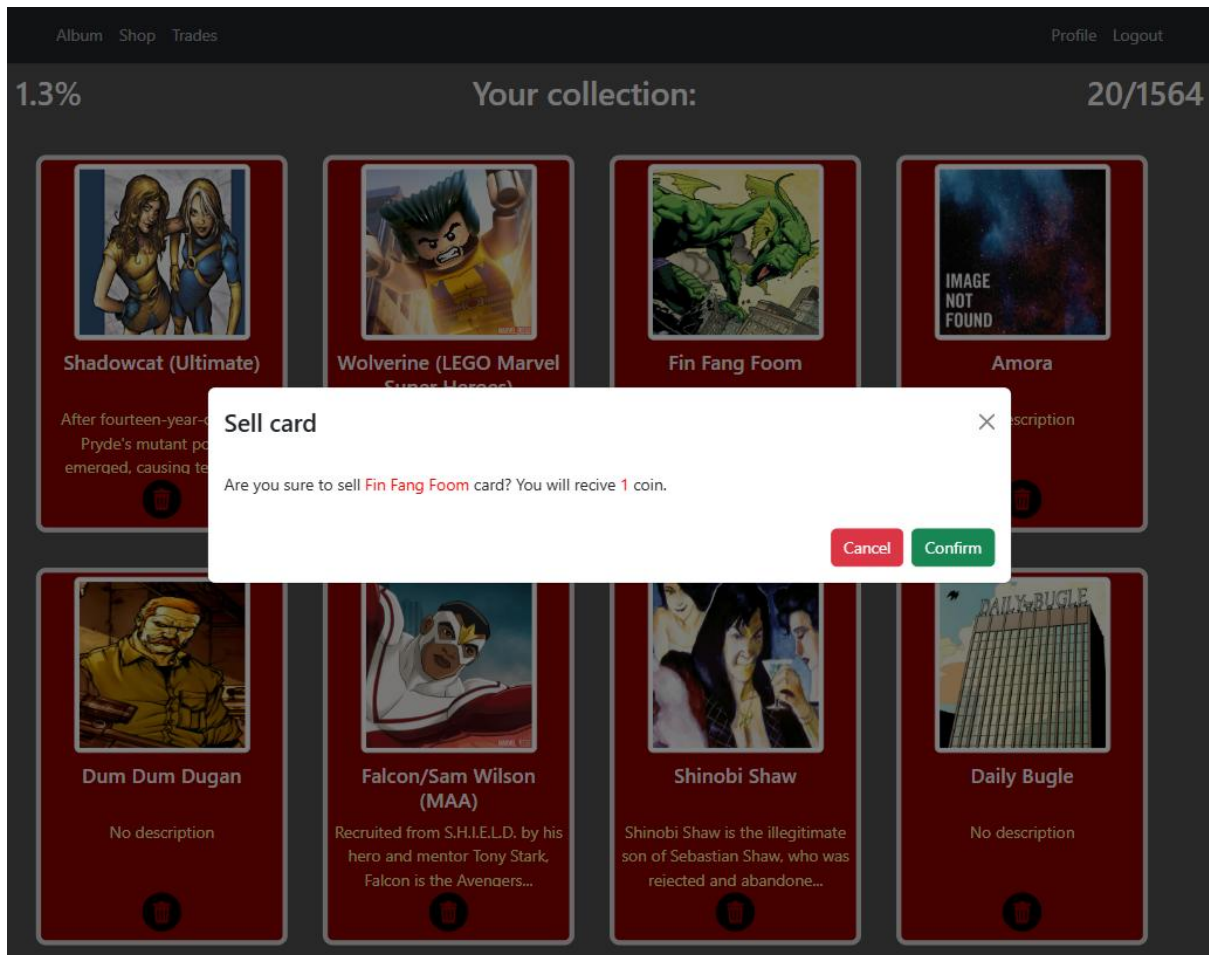
Album

Una volta che lo spaccettamento è andato a buon fine è possibile vedere le carte ricevute e in particolare tutte quelle possedute nella pagina album (path [/album](#)).



In alto a sinistra è visualizzabile la percentuale di carte possedute e in alto a destra invece il numero effettivo delle carte diverse possedute. Le carte mostrano le informazioni principali del personaggio: immagine, nome e descrizione (se esistente). In caso una carta sia stata trovata più volte in un pacchetto appare il componente Badge (che consiste in un cerchio verde con all'interno un numero) che indica il numero di carte di quel personaggio possedute. Nello screen presentato precedentemente si nota come il personaggio "Wolverine (Lego Marvel super heroes)" sia stato trovato due volte.

In ogni carta è anche presente in basso centrale un'icona rappresentabile un cestino, se cliccata e data conferma avvia la procedura di vendita della carta. La vendita di una qualsiasi carta restituisce all'utente un Coin.

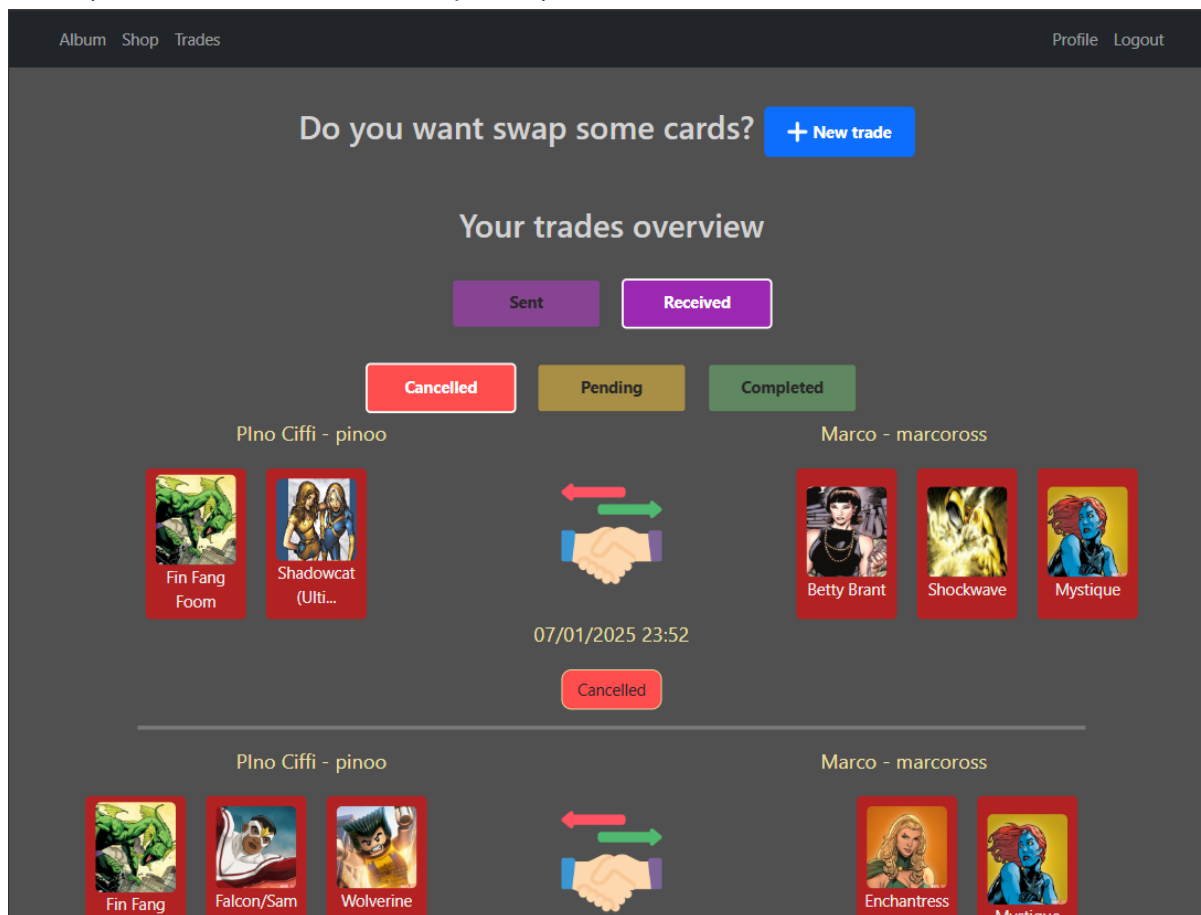


Le API utilizzate in questa pagina sono:

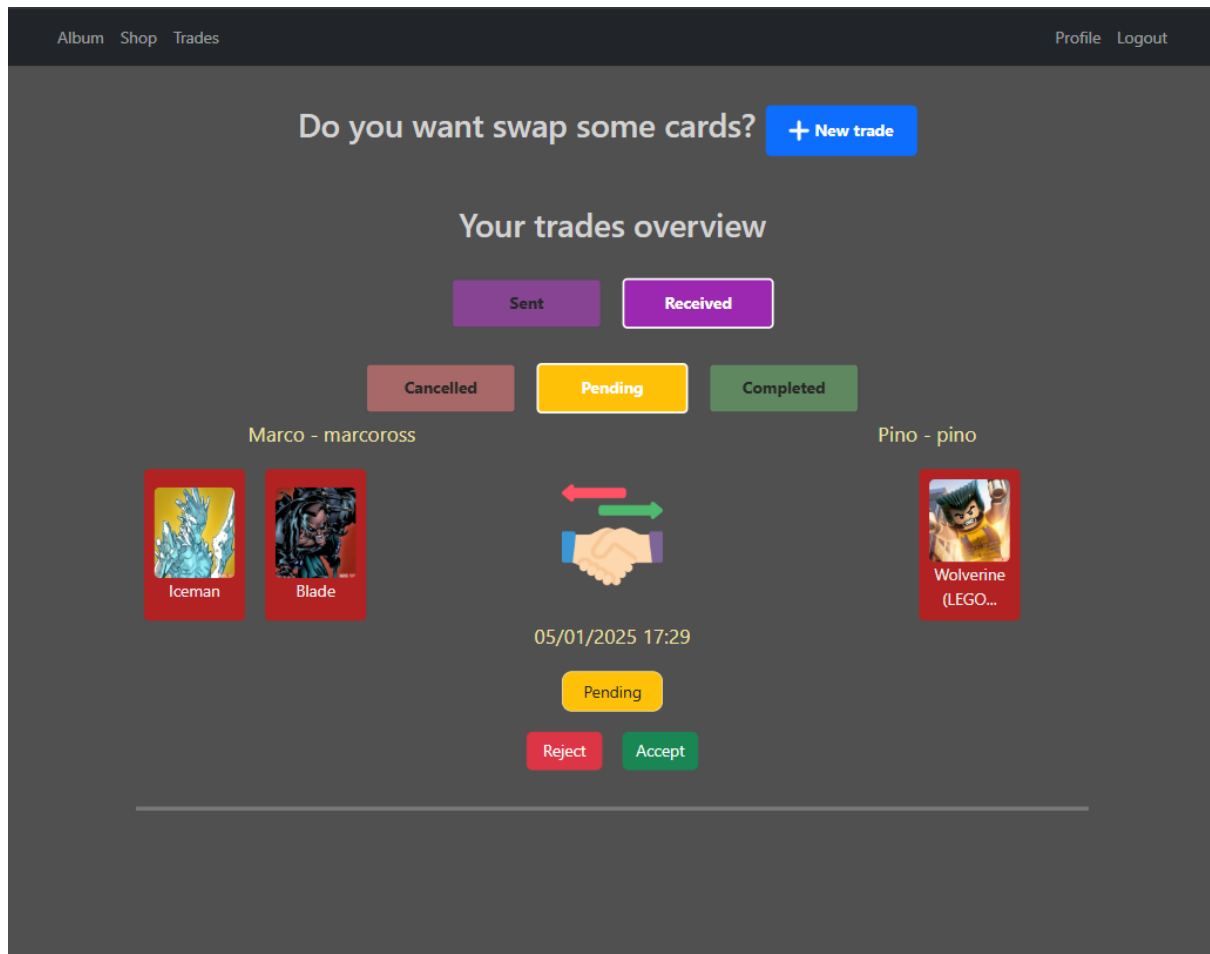
- [/api/users/cards](#) metodo GET per avere le informazioni relative a tutte le carte presenti nell'album di un utente
- gateway.marvel.com/v1/public/characters API marvel-portal con metodo GET per sapere il numero totale di personaggi nel Marvel-Portal
- [/api/cards/:CardId](#) metodo DELETE, questa API elimina la carta nell'album dell'utente e aggiunge un Coin all'utente

Trades

La pagina Trades (path </trades>) si divide in due parti, la prima permette la creazione di un nuovo trade invece la seconda mostra tutti i trades relativi all'utente in base alle proprietà selezionate. Si può filtrare per scambi inviati e ricevuti oppure per lo status del trade (cancellato, in attesa, completati).



Se vengono selezionati i trades ricevuti con status “pending” è anche possibile scegliere se accettare o rifiutare uno scambio. Se lo scambio viene accettato le carte tra gli utenti vengono effettivamente scambiate e lo status dello scambio muta in “completed”. Se lo scambio viene rifiutato è solamente modificato lo status dello scambio in “cancelled”. Prima di completare uno scambio vengono fatti alcuni controlli sull'effettiva disponibilità delle carte proposte e sull'integrità degli altri dati inviati. Se la carta non sarà effettivamente posseduta lo scambio verrà automaticamente rifiutato.



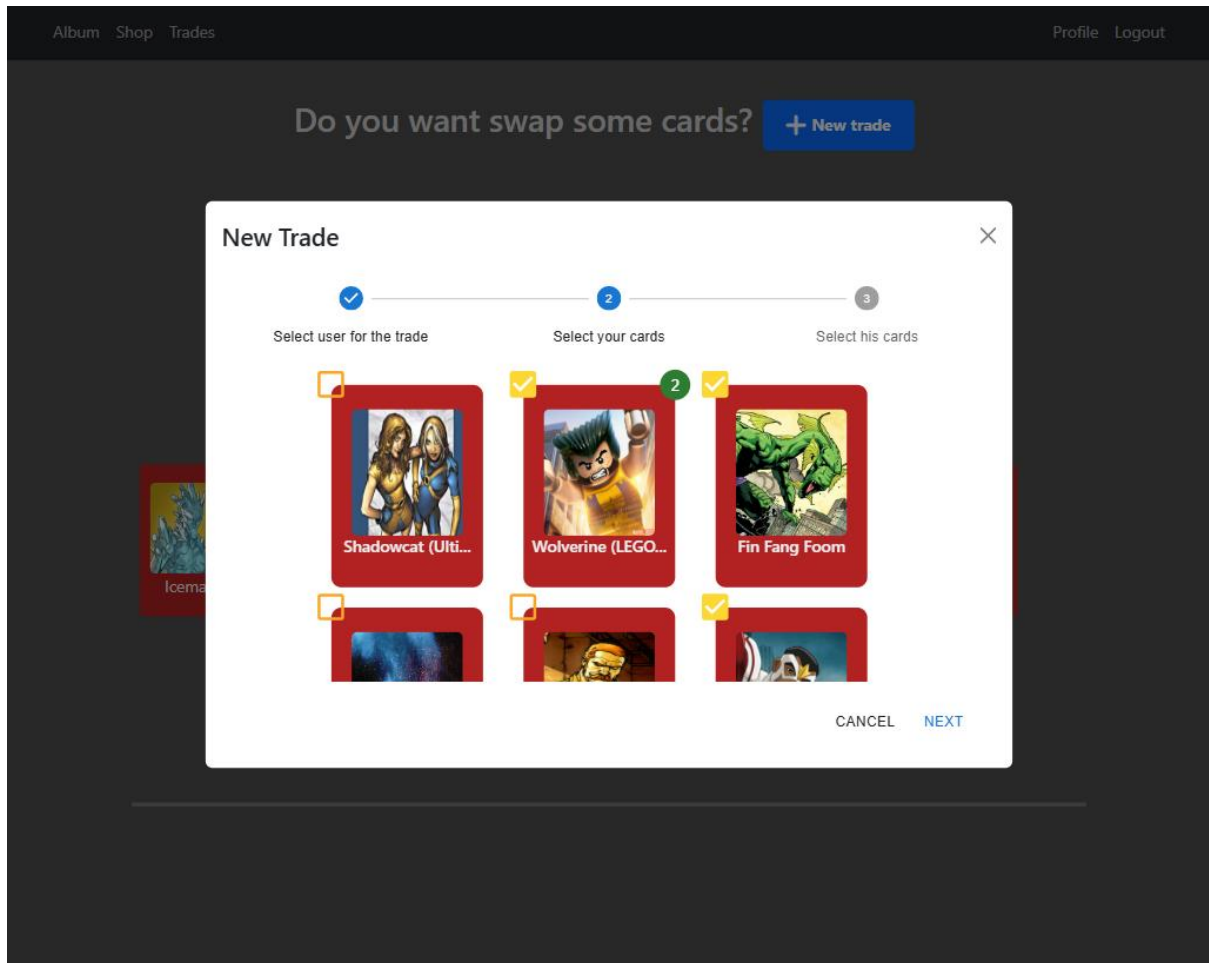
La creazione di un nuovo trade invece si attiva cliccando sul pulsante “New trade” a cui seguirà l’apertura di uno stepper.

Il primo step consiste nella ricerca dell’utente con cui si vuole creare uno scambio. Basterà cliccare su uno dei nomi utenti che compaiono dalla ricerca per selezionarlo. Per cambiare utente selezionato basterà cliccare su un altro utente, senza aver scelto alcun utente non sarà possibile passare al prossimo step.

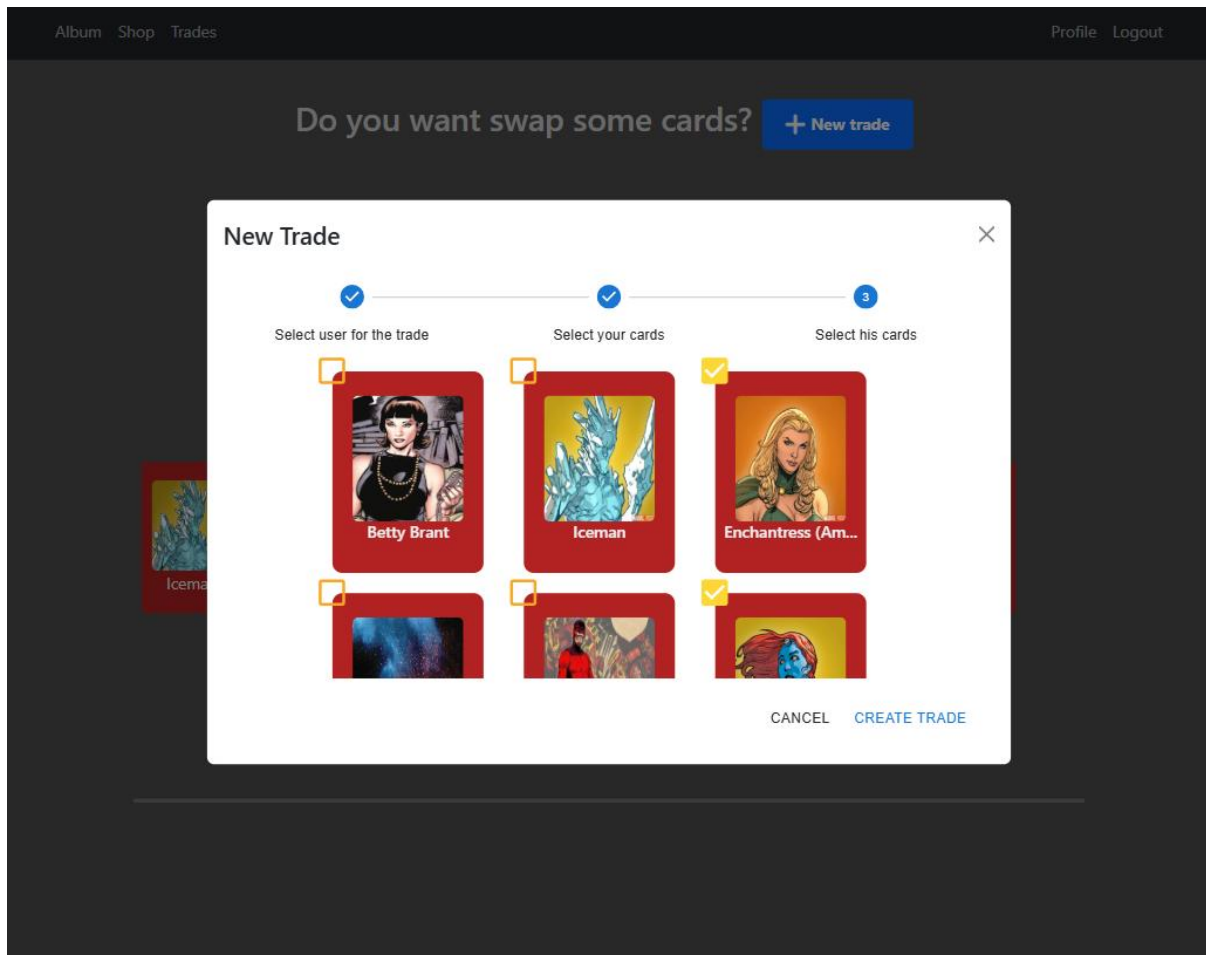
La ricerca non restituisce l’utente che crea lo scambio, così si evita la creazione di scambi con sé stesso.

The screenshot shows a web application interface with a dark theme. At the top, there is a navigation bar with links for 'Album', 'Shop', and 'Trades' on the left, and 'Profile' and 'Logout' on the right. Below the navigation bar, a main heading asks 'Do you want swap some cards?' with a blue button labeled '+ New trade'. A modal window titled 'New Trade' is open in the center. The modal has a close button (X) in the top right corner. It features a three-step progress indicator: Step 1 (active, blue circle) is 'Select user for the trade', Step 2 (grey circle) is 'Select your cards', and Step 3 (grey circle) is 'Select his cards'. In the first step, there is a search input field containing the text 'mar' and a dropdown menu showing the result 'marcoross - Marco'. Below the dropdown, it says 'User selected for the trade:' followed by a button labeled 'marcoross - Marco'. A 'NEXT' button is located at the bottom right of the modal. In the background, a card titled 'Iceberg' is partially visible.

Nel secondo step si vanno a selezionare le carte che l'utente vuole proporre per lo scambio, si selezionano tramite la checkbox presente su ogni carta in alto a sinistra.



Nel terzo step si vanno a selezionare le carte che si vogliono ricevere, sempre tramite la checkbox. Per terminare e creare effettivamente lo scambio si deve cliccare il pulsante “CREATE TRADE”.



Nel caso nessuna carta venga selezionata non sarà possibile passare al prossimo step o completare la proposta di scambio.

Le API utilizzate per questa pagina sono:


- [/api/trades](#) metodo PATCH per l'accettazione o rifiuto di uno scambio
- [/api/trades/:status](#) metodo GET per ottenere i trades appartenenti all'utente filtrati per status
- [/api/users/search?query=](#) metodo POST per ottenere l'elenco utenti
- [/api/trades](#) metodo POST per creare la proposta di un nuovo trade
- [/api/cards/userId](#) metodo GET per ottenere l'elenco delle carte possedute dall'utente loggato
- [/api/users/cards](#) metodo GET per ottenere le carte dell'utente scelto per la proposta di scambio

Card Details

La pagina di Card Details (path [/card/:cardName](#)) serve a dare tutte le informazioni aggiuntive che riguardano un singolo personaggio. Si raggiunge la pagina andando a cliccare sulle carte possedute nel proprio album.

[Album](#) [Shop](#) [Trades](#) [Profile](#) [Logout](#)

Iron Patriot (James Rhodes)



Description
U.S. Air Force pilot and Tony Stark's friend who has his own suit of Iron Man armor, nicknamed, "War Machine."

Urls
[detail](#) [wiki](#) [comiclink](#)

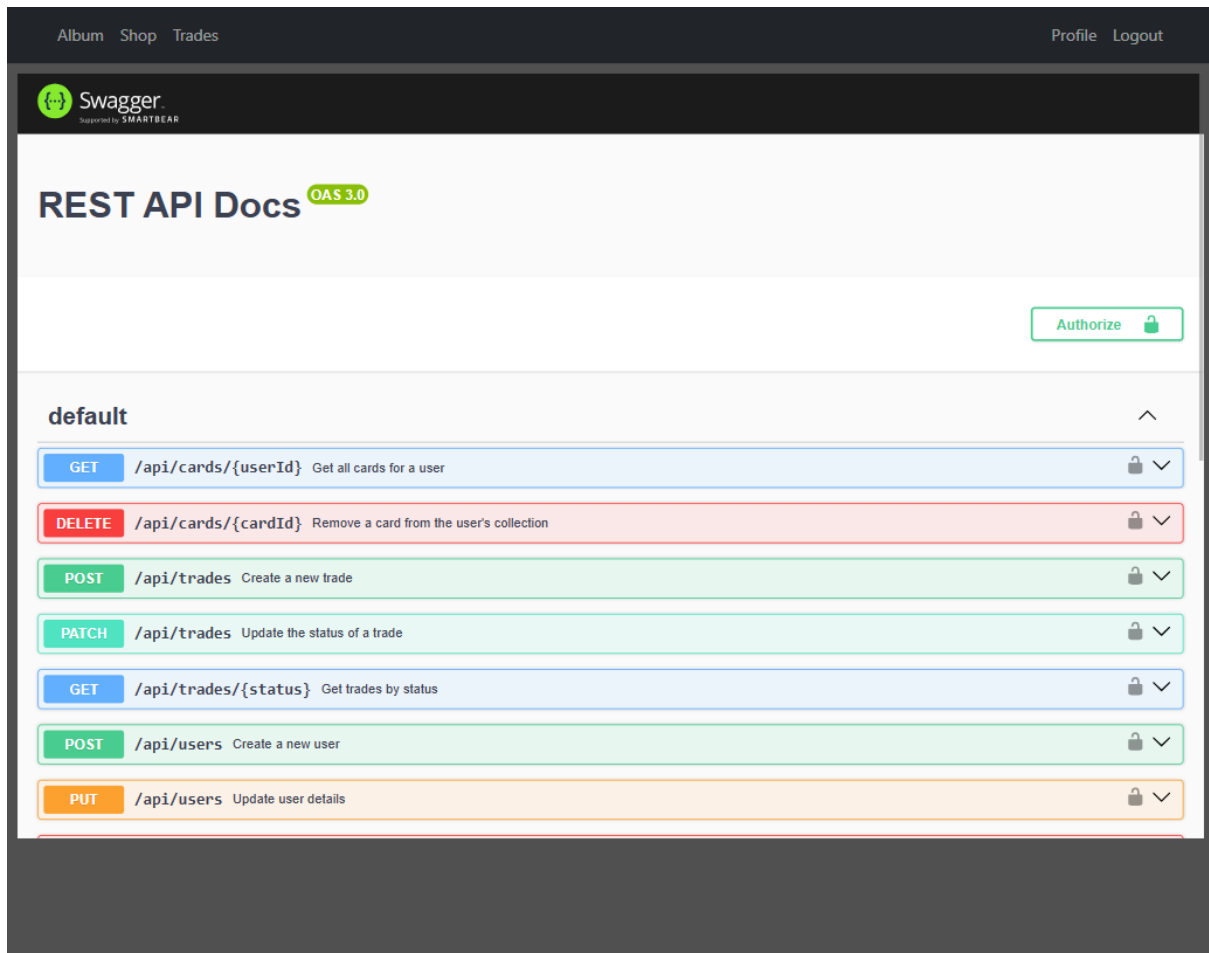
Series
Art of Marvel Studios TPB Slipcase (2011 - Present), Avengers: The Initiative (2007 - 2010), Avengers: The Terminatrix Objective (1993), Gambit (2012 - 2013), Incredible Hulks (2010 - 2011), Invincible Iron Man (2008 - 2012), Invincible Iron Man Vol. 1: The Five Nightmares (2009 - Present), Invincible Iron Man Vol. 4: Stark Disassembled (2011 - Present), Iron Man (1968 - 1996), Iron Man (1998 - 2004), Iron Man 2.0 (2011), Iron Man 2.0: Modern Warfare (2011), Iron Man Annual (1976 - 1994), Iron Man Legacy (2010 - 2011), Iron Man: Rapture (2010 - 2011), IRON MAN: RAPTURE TPB (2011), Marvel Comics Presents (1988 - 1995), New Avengers (2010 - 2012), Secret Avengers (2010 - 2012), Secret Avengers (2013 - 2014)

Comics
Art of Marvel Studios TPB Slipcase (Hardcover), West Coast Avengers (1985) #100, West Coast Avengers (1985) #101, West Coast Avengers Annual (1986) #8, Avengers: The Initiative (2007) #11, Avengers: The Initiative (2007) #12, Avengers: The Initiative (2007) #15, Avengers: The Initiative (2007) #16, Avengers: The Terminatrix Objective (1993) #1, Avengers: The Terminatrix Objective (1993) #2, Avengers: The Terminatrix Objective (1993) #3, Avengers: The Terminatrix Objective (1993) #4, The Crew (2003) #6, Gambit (2012) #13, Incredible Hulks (2010) #607 (MCGUINNESS VARIANT), Invincible Iron Man (2008) #1, Invincible Iron Man (2008) #2, Invincible Iron Man (2008) #25 (TRIMPE VARIANT), Invincible Iron Man (2008) #25 (MOVIE VARIANT), Invincible Iron Man (2008) #25 (2ND PRINTING VARIANT)

L'unica API utilizzata in questa pagina è gateway.marvel.com/v1/public/characters che serve a ricevere le informazioni di un personaggio nel portale Marvel.

Docs

La pagina di Docs (path </docs>) implementa la tecnologia di testing API chiamata Swagger. Essa illustra in dettaglio tutte le funzionalità e i requisiti delle API utilizzate nel progetto.



Scelte implementative

Standardizzazione Alert

Nel progetto per notificare il successo o il fallimento di un'operazione è utilizzato il componente Snackbar dalla libreria MUI (Material UI). Si tratta di un alert che appare in basso in centro alla pagina e restituisce lo stato finale dell'operazione e una breve descrizione. L'utilizzo di questo alert è stato standardizzato inserendo come descrizione il testo ricevuto dalle API nel campo "message" e creando un file dedicato (AlertContext) dove vengono gestite proprietà e le funzioni di apertura e chiusura richiamabili in qualsiasi parte del progetto.

Estrazione di una carta casuale

La procedura di estrazione di una carta, che si svolge nella pagina di Shop, si basa principalmente sul utilizzo della funzione `Math.floor(Math.random() * (totalCards));`. Questa funzione permette pescare un numero casuale intero tra 0 e il numero totale delle carte. Per sapere prima dell'utilizzo il numero totali delle carte ad ogni costruzione della pagina viene utilizzata l'API del portale marvel `"gateway.marvel.com/v1/public/characters"` senza parametri, tra i campi della risposta ci sarà indicato il numero totale dei personaggi. Una volta estratto il numero si cercherà tramite offset del portale Marvel il personaggio corrispondente tramite la stessa API e verranno estratte le informazioni principali.

Salvataggio delle carte

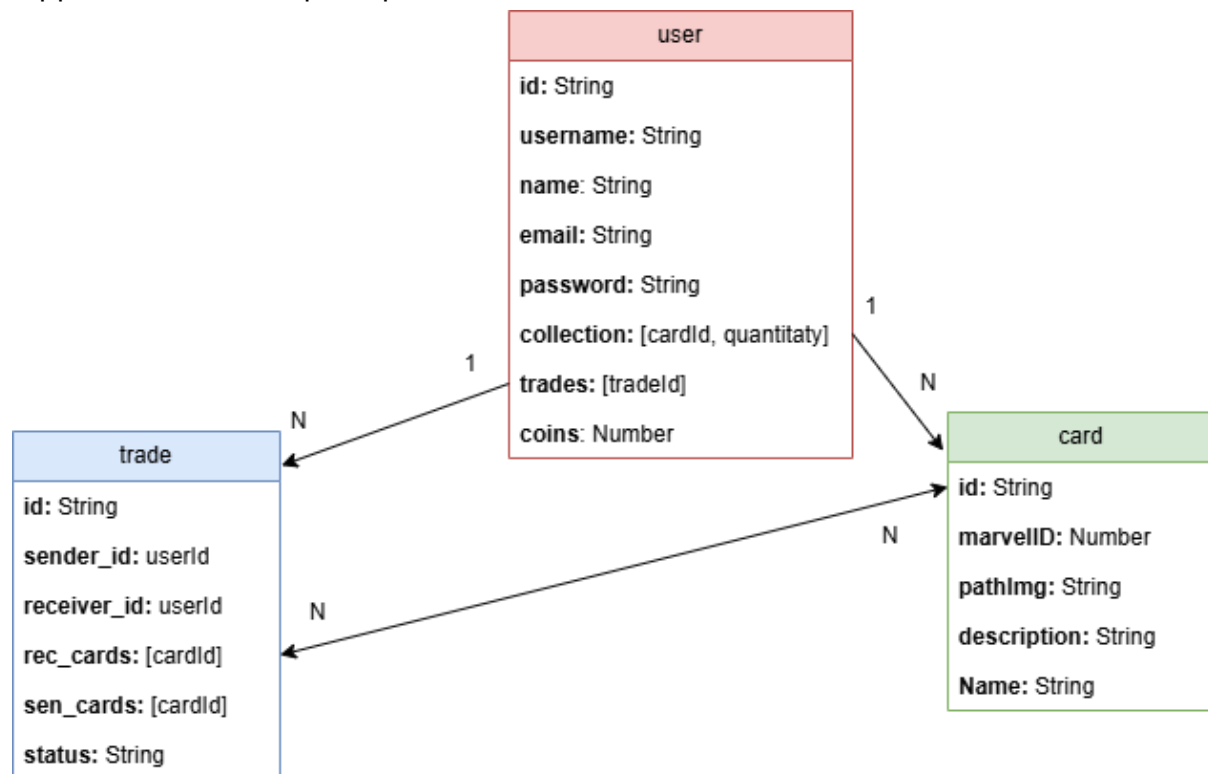
Per le carte è stato pensato di salvare le principali informazioni dei personaggi anche nel database di backend. Questo perché le API messe a disposizione dal Marvel-Portal non permettono la restituzione di più personaggi tramite passaggio di ID, a meno che non siano in ordine, e avendo a disposizione un numero di chiamate limitate fare una chiamata per ogni volta che si visualizza una carta sarebbe troppo.

Il salvataggio avviene durante lo spaccettamento della carta, una volta estratta viene salvata sul database se non già presente.

Questo permette di non richiedere l'utilizzo di API Marvel ma solo i dati del backend del progetto se non per gli spaccettamenti e per le pagine dedicate ai singoli personaggi.

Struttura database

Nella figura seguente viene mostrato lo schema dei vari oggetti nel database, che rappresenta le entità principali e le loro relazioni.



collection è l'unico attributo più strutturato che contiene per ogni elemento l'ID che richiama l'oggetto Card e quantity che indica il numero di carte collezionate.

La password dell'utente è l'unica informazione che viene criptata e successivamente salvata, tramite la funzione **bcrypt.hash(password, salt)**. Il salt è un valore casuale, che viene usato come componente aggiuntiva per rendere il processo di hashing più sicuro.

L'organizzazione delle collections sul database è la seguente:

marvel-album							
LOGICAL DATA SIZE: 85.02KB STORAGE SIZE: 164KB INDEX SIZE: 188KB TOTAL COLLECTIONS: 3							
CREATE COLLECTION							
Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
cards	343	68.04KB	204B	84KB	1	44KB	44KB
trades	16	2.71KB	174B	36KB	1	36KB	36KB
users	8	14.27KB	1.78KB	44KB	3	108KB	36KB

JWT Token

La tecnologia utilizzata per lo spostamento e verifica di dati tra backend e frontend è JWT token.

Un token JWT è composto da tre parti:

1. Header: contiene il tipo di token (JWT) e l'algoritmo di firma utilizzato (es. HMAC SHA256).
2. Payload: include le informazioni come l'ID utente, il nome e altre informazioni rilevanti.
3. Signature: garantisce l'integrità del token, generata combinando header, payload e una chiave segreta. La chiave è salvata e utilizzata solo nel backend.

Nel frontend il token viene rilasciato dall'API di login oppure alla creazione di un nuovo utente e ha una validità di un'ora. Viene salvato poi nel **local storage** del browser che lo rende accessibile da qualsiasi parte nel codice.

La verifica di validità del token è sempre fatta lato back-end, tramite il middleware `authMiddleware` che è sempre utilizzato in ogni API esposta.

Cookie

Il salvataggio del token nel local storage è considerata una pratica poco sicura in quanto sensibile principalmente agli attacchi di tipo Cross-Site Scripting (XSS). Era stato infatti inizialmente pensato l'utilizzo dei **cookies http** per il suo salvataggio. Sono però emerse alcune complicazioni nella messa in produzione del sito, non avendo la possibilità di creare sottodomini di uno specifico url non era possibile rispettare le politiche di domain di un cookie. Le API erano quindi funzionanti ma il cookie non veniva salvato perché considerato non sicuro dal browser. In ambiente locale invece essendo sia frontend che backend su localhost il problema non sussiste.

Il codice del progetto con la modifica strutturale appena descritta è comunque presente su [Github](#) nel branch **main**.

Media Query

Per rendere responsiva a qualsiasi dimensione applicata l'applicazione sono state utilizzate le media query, sono una funzionalità di CSS che permette di applicare stili diversi in base a caratteristiche specifiche del dispositivo o della finestra del browser, come la larghezza dello schermo, l'orientamento, la risoluzione o il tipo di dispositivo.

Sviluppi futuri

- Creazione pacchetti da un admin page
- Pagina che vada a mostrare tutte le carte possibili da trovare
- Reset della password tramite invio mail
- Sistema di refresh del token quando scaduto