

**Corso di Programmazione
Web e
Mobile A.A. 2024-2025 SSRI**

Acquista e Scambia

•Corbetta Simone, 21577A•

Acquista e scambia

1. Introduzione

Progetto "Acquista e Scambia" è una piattaforma web dedicata all'acquisto e allo scambio di figurine riguardo il mondo di Harry Potter, offrendo agli utenti la possibilità di registrarsi e accedere al sito, gestire i propri dati e acquistare crediti, valore inventato per acquistare i pacchetti da 5 figurine. Gli utenti possono acquistare i crediti, acquistare i pacchetti e inserire le carte trovate nel loro album principale o nell'album dei doppioni, visualizzare informazioni dettagliate sulle carte, possono mandare una richiesta di scambio ad altri utenti. Il progetto utilizza: HTML, CSS e JavaScript per l'interfaccia utente, mentre sul lato back-end si utilizza il collegamento con Mongodb per la gestione del database per memorizzare i dati degli utenti e degli album e dello scambio di richiesta/risposta tra utenti, insieme alle API per ottenere informazioni sulle carte dei personaggi di Harry Potter.

1.1. Analisi dei requisiti

1.1.1. Destinatari

Capacità e possibilità tecniche:

I destinatari del progetto devono possedere conoscenze di base sull'utilizzo di piattaforme web e la navigazione su Internet. Non è richiesto un livello avanzato di esperienza tecnica, ma una comprensione generale dell'interazione con siti web. Potrebbero non essere esperti nel campo della gestione di comprare figurine virtuali e come funziona la dinamica dello scambio delle carte tra utenti, ma il sito web è molto semplice e intuitivo. L'accesso a una quantità adeguata di banda Internet consente un'esperienza di navigazione fluida e senza interruzioni.

Motivazione:

Tipo di motivazione: Intrattenimento di passatempo

Livello di motivazione: La motivazione può variare da attiva a passiva a seconda degli utenti, quindi potrebbero interagire con il sito, acquistando crediti per comprare pacchetti di figurine per ampliare la propria collezione di figurine (album principale) e cercare di compiere più scambi possibili per cercare di trovare tutte le carte, mentre altri potrebbero essere più passivi e semplicemente avendo un collezione incompleta o poco ricca di figurine.

Il motivo per cui un utente vuole registrarsi possono essere diversi: per passatempo, per passione del collezionismo di quel genere di carte o addirittura per pura curiosità.

1.1.2. Valore

Il valore dell'applicazione: Servizi e contenuti che danno valore all'applicazione sono: gestione dei propri dati personali, acquisto di pacchetti, visualizzazione degli album, possibilità di scambio con altri utenti, possibilità di vendere per crediti le figurine doppione.

Gli elementi che possono attirare l'interesse degli utenti: un'interfaccia intuitiva e facilità d'uso.

Gli elementi che forniscono valore all'investitore includono: la crescita della base di utenti, le opportunità di monetizzazione con pubblicità o abbonamenti premium e il valore dell'azienda in base al successo.

Le transazioni che generano valore possono essere: abbonamenti premium, l'acquisto dei crediti (magari con aggiunta di bonus o vantaggi per chi è premium), partnership pubblicitarie e acquisizioni.

Un altro valore che può aumentare il successo dell'azienda è quella di ampliare il range del collezionismo con magari altre figurine, tipo carte Pokemon, della Marvel e altre ancora.

1.1.3. Flusso dei dati

Quale qualità, stile e livello di dettagli:

Qualità: I dati degli album e soprattutto delle transazioni tra utenti per gli scambi, sugli utenti devono essere sicuri e aggiornati per fornire un'esperienza utente affidabile. Le informazioni sulle figurine dovrebbero includere dettagli come: il nome del personaggio, l'immagine e altri dati per arricchire l'esperienza dell'utente.

Stile: Gli album sono organizzate in modo intuitivo, consentendo agli utenti di trovare facilmente tutte le carte. Le informazioni sulle figurine sono in un formato leggibile e semplice, con anche le immagini.

Ottenerne i contenuti:

I costi: dipendono dal metodo scelto per ottenere i contenuti. La produzione interna può richiedere risorse umane e finanziarie per la raccolta e l'organizzazione dei dati, nonché per lo sviluppo dei sistemi di gestione dei contenuti. Il reperimento da fonti esterne potrebbe comportare l'accesso a servizi a pagamento per l'utilizzo dei dati.

Archiviare e organizzare i contenuti:

Archiviazione: I contenuti sono archiviati utilizzando un database su MongoDB, per consentire una gestione efficiente dei dati. L'architettura del database è formata da tre collezioni: utenti album e scambio richiesta/risposta.

Organizzazione contenuti: Gli utenti appena acquistano il primo pacchetto e le carte vengono inserite nell'album, vengono creati due album (principale e doppiioni).

Ovviamente l'album principale sarà quello in cui andranno salvati le figurine nuove, invece nei doppioni tutte quelle che ho già trovato e quindi potrò vendere per crediti o scambiare con altri utenti.

1.1.4. Aspetti tecnologici

Archiviazione dei contenuti:

Metodo di archiviazione: I contenuti sono archiviati utilizzando un database su MongoDB, per consentire una gestione efficiente dei dati. L'architettura del database è formata da 3 collezioni: utenti, album e scambio_richieste.

Si potrebbe adottare un sistema di archiviazione basato su cloud per garantire la scalabilità e l'accessibilità dei contenuti.

Tecnologie utilizzate:

HTML, CSS, JavaScript: utilizzate per sviluppare l'interfaccia utente del social network, gestendo la presentazione, l'interazione e la logica del frontend.

Node.js: è stato utilizzato per creare il backend lato server dell'applicazione, gestendo le richieste degli utenti, l'autenticazione, l'accesso al database e altre funzioni.

Express: framework web per Node.js, utilizzato per semplificare la gestione delle route e delle richieste HTTP.

MongoDB: Un database utilizzato per archiviare e gestire i dati degli utenti e degli album e dei messaggi tra utenti nelle collezioni.

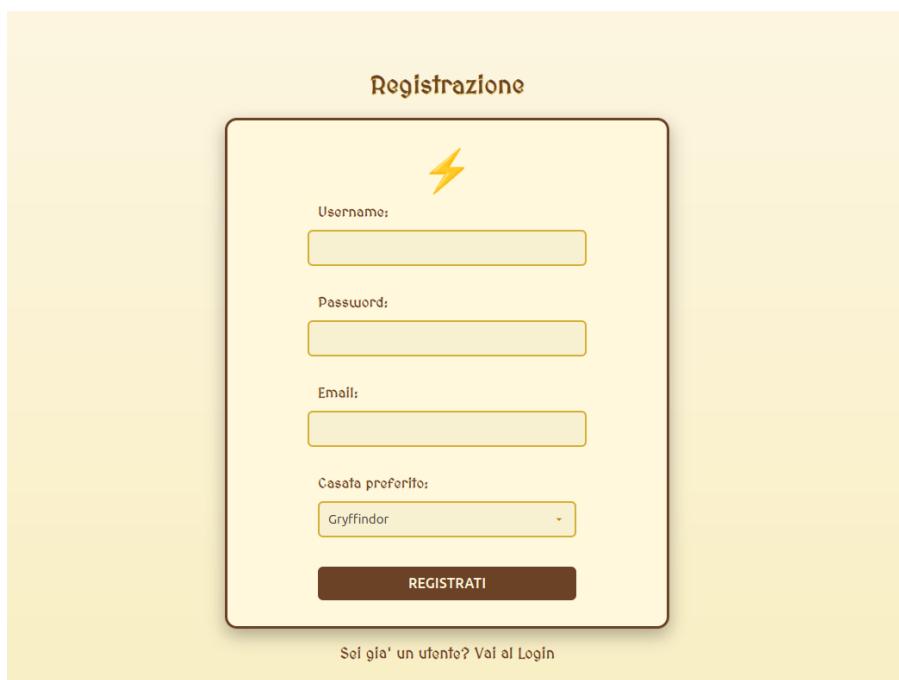
API: per effettuare richieste al link <https://hp-api.onrender.com> e ottenere i dati delle canzoni.

2. Interfacce

Interfaccia di registrazione, accesso e modifica dati utenti:

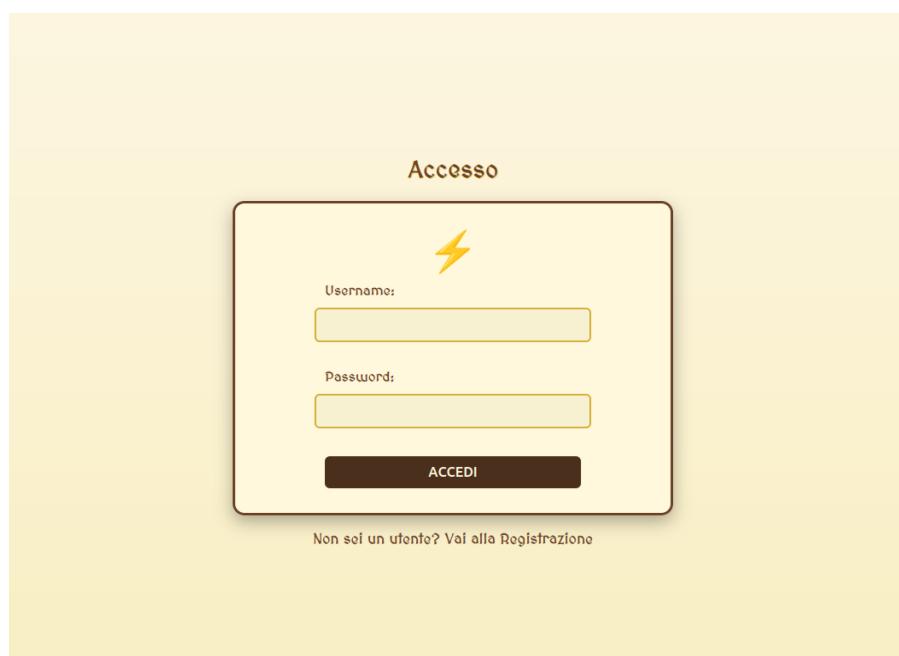
Questa interfaccia consente agli utenti di registrarsi al social network o di accedere se sono già registrati. Essa include campi per l'inserimento di informazioni come nome, email, password ecc...

- Registrazione



The registration interface is titled "Registrazione". It features a yellow lightning bolt icon at the top. Below it are four input fields: "Username:", "Password:", "Email:", and "Casata preferita:". The "Casata preferita:" field contains a dropdown menu with "Gryffindor" selected. At the bottom is a brown "REGISTRATI" button.

- Accesso



The access interface is titled "Accesso". It features a yellow lightning bolt icon at the top. Below it are two input fields: "Username:" and "Password:". At the bottom is a dark brown "ACCEDI" button.

- Modifica dati utenti e' identica a quella della registrazione

Interfaccia della dashboard (home) per l'utente: (ovviamente cambiarla completamente)

Dopo l'accesso, gli utenti verranno indirizzati alla loro home personale, dove si possono svolgere altre funzioni fondamentali del sito web come: il pulsante per l'acquisto dei crediti, la pagina per comprare i pacchetti di figurine, le pagine per gestire la visualizzazione degli album e dei messaggi tra utenti per lo scambio e, ovviamente, le pagine o le funzioni per gestire l'utente (modificarlo o cancellarlo):



Interfaccia di visualizzazione degli album (principale e dei doppioni):



Le interfacce per visualizzare gli album, ovvero le carte, sono molto simili però cambiano delle piccolezza in base a quali funzioni ci sono. Per l'interfaccia dell'album principale si vedono tutte le carte trovate e collezionate nel database e al massimo ci sono tutte le informazioni relative alla carta. Invece nell'interfaccia dell'album dei doppioni e' praticamente uguale, ma in aggiunta contiene due pulsanti in più. Uno per la gestione degli scambi con altri utenti e l'altro invece e' per vendere le carte per dei crediti.

Interfaccia acquisto pacchetti e acquisto crediti:
crediti:



pacchetti:



Queste interfacce sono strutturalmente diverse, ma la loro logica funzionale e' la stessa. Nella prima c'è un pulsante per l'acquisto del pacchetto e mi compaiono 5 figurine da aggiungere nel mio album, che sia principale o dei doppioni.

Invece la seconda interfaccia consiste nell'acquisto dei crediti, usati nella piattaforma per comprare i pacchetti.

Interfaccia scelta della carta dell’altro utente:

Questa interfaccia viene visualizzata quando l’utente preme il bottone “scambio” nella pagina dell’album dei doppioni. Essa permette di scegliere una tra le altre doppioni degli altri utenti da scambiare con quella dell’utente. Scegliendo la carta, fa partire una richiesta di scambio al possessore della figurina.



Interfaccia dei messaggi:

Questa pagina fa visualizzare tutte le richieste di scambio (utente destinatario) e le risposte di conferma (utente mittente) riferite all’utente loggato. I messaggi possono essere di due tipi: scambio o testo.

Scambio vuole dire che un utente a caso ha scelto la figurina dell’utente che vuole scambiare con la sua e il ricevente deve accettare o rifiutare lo scambio (ho messo un aggiunta per fare in modo che la conferma della richiesta sia molto più veloce e diretta, ovvero alla carta proposta ho aggiunto un asterisco se ne è già in possesso).

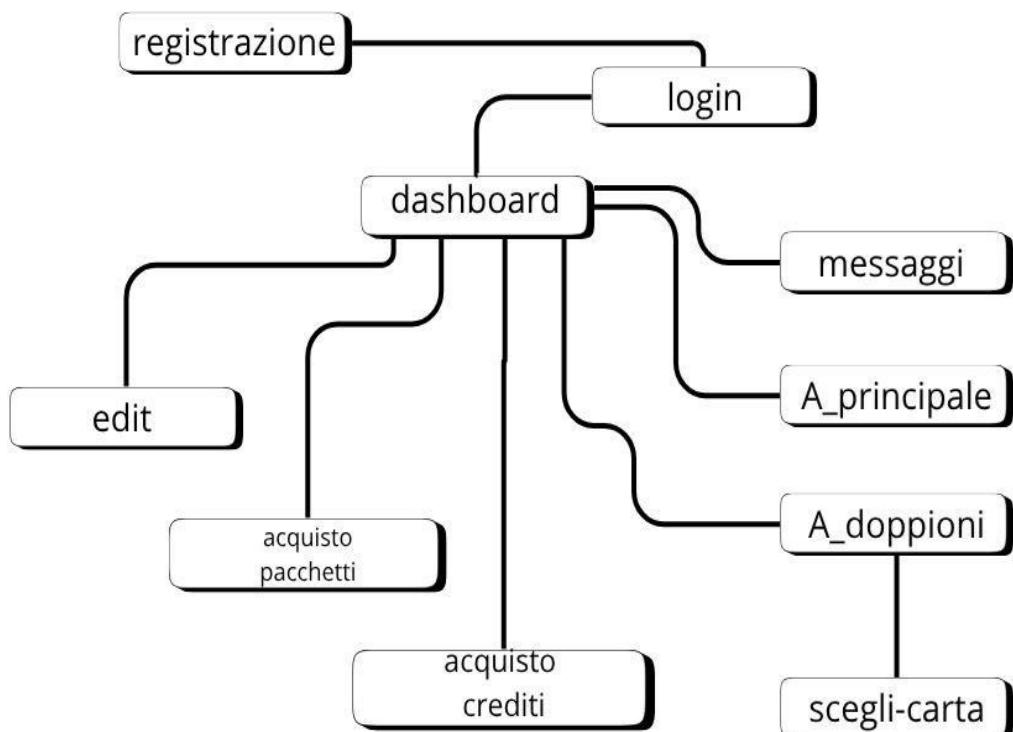
Invece il messaggio di tipo testo e’ la risposta della richiesta, quindi la conferma dello scambio, e quindi si svolge lo scambio, o il rifiuto.

A screenshot of a page titled "Richieste di Scambio". It lists three messages in a scrollable area. 1. Scambio rifiutato: Ciao, ho rifiutato la tua proposta di scambio del 02/06/2025, 00:05:41. 2. Scambio accettato: Ciao, ho accettato la tua proposta di scambio del 02/06/2025, 00:18:02. 3. Scambio rifiutato: Ciao, ho rifiutato la tua proposta di scambio del 02/06/2025, 00:18:50. At the bottom is a "Torna Indietro" button.

3. Architettura

3.1. Ordine gerarchico delle risorse

Architettura del sito



4. Codice

Parti di codice più importanti, NON completi dei file.

4.1. HTML

Registrazione e Accesso:

```
public > register.html > html > body > form > button#submit
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link rel="stylesheet" href=".//assets/css/style.css">
7      <title>Registrazione</title>
8  </head>
9  <body>
10     <h1>Registrazione</h1>
11     <form action="/register" method="POST">
12         <label for="username">Username:</label>
13         <input type="text" id="username" name="username" required>
14         <br>
15         <label for="password">Password:</label>
16         <input type="password" id="password" name="password" required>
17         <br>
18         <label for="email">Email:</label>
19         <input type="text" id="email" name="email">
20         <br>
21         <label for="house_p">Casata preferito:</label>
22         <select id="house_p" name="house_p">
23             <option value="Gryffindor">Gryffindor</option>
24             <option value="Slytherin">Slytherin</option>
25             <option value="Hufflepuff">Hufflepuff</option>
26             <option value="Ravenclaw">Ravenclaw</option>
27         </select>
28         <br>
29         <button id='submit' type="submit">Registrati</button>
30     </form>
31     <a href=".//login.html">Sei già un utente? Vai al Login</a>
32 </body>
33 </html>
34
```

```
public > login.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link rel="stylesheet" href=".//assets/css/style.css">
7      <title>Accesso</title>
8  </head>
9  <body>
10     <h1>Accesso</h1>
11     <form action="/login" method="POST">
12         <label for="username">Username:</label>
13         <input type="text" id="username" name="username" required>
14         <br>
15         <label for="password">Password:</label>
16         <input type="password" id="password" name="password" required>
17         <br>
18         <button id='submit' type="submit">Accedi</button>
19     </form>
20     <a href=".//register.html">Non sei un utente? Vai alla Registrazione</a>
21 </body>
22 </html>
23
```

Modifica dati utente:

```
public > edit.html > html > body > div.edit-form > form#editForm > br
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Modifica i tuoi dati</title>
7      <link rel="stylesheet" href=".//assets/css/style.css">
8  </head>
9  <body>
10     <div class="edit-form">
11         <h1>Modifica i tuoi dati</h1>
12         <form id="editForm">
13             <label for="username">Username:</label>
14             <input type="text" id="username" name="username" required>
15             <br>
16             <label for="password">Nuova Password:</label>
17             <input type="password" id="password" name="password" required>
18             <br>
19             <label for="email">Email:</label>
20             <input type="text" id="email" name="email">
21             <br>
22             <label for="house_p">Casata preferito:</label>
23             <select id="house_p" name="house_p">
24                 <option value="Gryffindor">Gryffindor</option>
25                 <option value="Slytherin">Slytherin</option>
26                 <option value="Hufflepuff">Hufflepuff</option>
27                 <option value="Ravenclaw">Ravenclaw</option>
28             </select>
29             <br>
30             <button id='submit' type="submit">Aggiorna</button>
31         </form>
32         <button class='torna_indietro' onclick="history.back()">Torna indietro</button>
33     </div>
34
```

Dashboard:

```
public > dashboard.html > html > body > div.header-bar > button#logout
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link rel="stylesheet" href=".//assets/css/dashboard.css">
7      <title>Dashboard</title>
8  </head>
9  <body>
10     <!-- Barra di intestazione -->
11     <div class="header-bar">
12         <h1>Dashboard</h1>
13         <button id='logout' onclick='window.location.href = "/login";'>Logout</button>
14         <div class="credit-box">
15             <button onclick="window.location.href='./credit.html'"></button>
16             <p>Crediti: <span id="credit-value">${credits}</span></p>
17         </div>
18     </div>
19
20     <!-- Contenuto principale della dashboard -->
21     <div class="dashboard">
22         <h1>Benvenuto nella tua Dashboard!</h1>
23         <button onclick="window.location.href='./edit'">Modifica Profilo</button>
24         <button onclick="deleteAccount()">Elimina Account</button>
25         <button onclick="window.location.href='./messaggi'">Notifiche</button>
26         <button onclick="window.location.href='./buy_pacchetti'">Acquista pacchetti</button>
27     </div>
28
29     <div class="container text-center mt-5">
30         <div class="d-flex justify-content-center gap-4 mt-4">
31             <!-- Pulsante Album Principale -->
32             <button id="viewMainAlbum" class="btn btn-primary">Visualizza Album Principale</button>
33             <!-- Pulsante Album Duplicati -->
34             <button id="viewDuplicatesAlbum" class="btn btn-secondary">Visualizza Album Duplicati</button>
35         </div>
36     </div>
37
```

Acquista pacchetti:

```
public > ◊ album.html > ⚡ html > ⚡ body > ⚡ button
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Acquisto HP Album</title>
7     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
8     <link rel="stylesheet" href="./assets/css/acquisto.css">
9   </head>
10  <body>

public > ◊ credit.html > ⚡ html > ⚡ body > ⚡ div.dashboard-container > ⚡ div.credit-management > ⚡ div.credit-box > ⚡ button
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <link rel="stylesheet" href="./assets/css/dashboard.css">
7     <title>Gestione Credito</title>
8   </head>
9   <body>
10    <div class="dashboard-container">
11      <h2>Gestione del Credito</h2>
12      <div class="credit-management">
13        <!-- Sezione Acquisto Crediti -->
14        <div class="credit-box">
15          <h3>Acquista Crediti</h3>
16          <button onclick="buyCredits(1)">1 Credito - €1</button>
17          <button onclick="buyCredits(2)">2 Crediti - €2</button>
18          <button onclick="buyCredits(5)">5 Crediti - €5</button>
19        </div>
20      </div>
21      <p id="feedback" style="color: #ffcc00; text-align: center; margin-top: 20px;"></p>
22      <a href="./dashboard.html">Torna indietro</a>
23    </div>
24    <script src="./assets/js/credit.js"></script>
25  </body>
26 </html>
27
```

Acquista crediti:

```
public > ◊ credit.html > ⚡ html > ⚡ body > ⚡ div.dashboard-container > ⚡ div.credit-management > ⚡ div.credit-box > ⚡ button
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <link rel="stylesheet" href="./assets/css/dashboard.css">
7     <title>Gestione Credito</title>
8   </head>
9   <body>
10    <div class="dashboard-container">
11      <h2>Gestione del Credito</h2>
12      <div class="credit-management">
13        <!-- Sezione Acquisto Crediti -->
14        <div class="credit-box">
15          <h3>Acquista Crediti</h3>
16          <button onclick="buyCredits(1)">1 Credito - €1</button>
17          <button onclick="buyCredits(2)">2 Crediti - €2</button>
18          <button onclick="buyCredits(5)">5 Crediti - €5</button>
19        </div>
20      </div>
21      <p id="feedback" style="color: #ffcc00; text-align: center; margin-top: 20px;"></p>
22      <a href="./dashboard.html">Torna indietro</a>
23    </div>
24    <script src="./assets/js/credit.js"></script>
25  </body>
26 </html>
27
```

Visualizzazione album principale e dei doppioni (simile):

```
public > AlbumPrincipale.html > html > head > meta
1  <!DOCTYPE html>
2  <html lang="it">
3  <head>
4  | <meta charset="UTF-8" />
5  | <title>Album Principale</title>
6  | <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
7  | <link rel="stylesheet" href="./assets/css/album_HP.css">
8  </head>
9  <body>
10 | <div class="container py-4">
11 | | <h1 class="mb-4">Il Mio Album di Figurine</h1>
12 | | <button onclick="history.back()">Torna indietro</button>
13 | | <div class="row" id="album"></div>
14 | </div>
15 |
16 | <script src="./assets/js/utils.js"></script>
17 | <script src="./assets/js/app.js"></script>
18 | <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
19 </body>
20 </html>
21
```

Pagina scegli-carta:

```
public > scegli-carta.html > ...
1  <!DOCTYPE html>
2  <html lang="it">
3  <head>
4  | <meta charset="UTF-8" />
5  | <title>Scegli carta da scambiare</title>
6  | <link rel="stylesheet" href="./assets/css/scegli_carta.css">
7  </head>
8  <body>
9  | <h1>Scegli una carta da scambiare</h1>
10 | <div id="offerte-container"></div>
11 | <button onclick="history.back()">Torna indietro</button>
12 </body>
13 <script src="./assets/js/scambio.js" defer></script>
14 </html>
15
```

Pagina dei messaggi tra utenti:

```
public > messaggi.html > html > body > div#richieste-container
1  <!DOCTYPE html>
2  <html lang="it">
3  <head>
4  | <meta charset="UTF-8" />
5  | <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6  | <title>Richieste di Scambio</title>
7  | <link rel="stylesheet" href="./assets/css/harrypotter.css" />
8  </head>
9  <body>
10 | <h1>Richieste di Scambio</h1>
11 |
12 | <div id="richieste-container">
13 | | <!-- Le richieste verranno inserite qui dinamicamente -->
14 | </div>
15 | <button onclick="history.back()">Torna indietro</button>
16 | <script src="./assets/js/messaggi.js"></script>
17 </body>
18 </html>
19
```

4.2. CSS

Ovviamente sono pezzi di codice per vedere come sono strutturati i file css per le diverse pagine del sito.

Registrazione, accesso, modifica dati:

```
public > assets > css > # style.css > #submit
1 @import url('https://fonts.googleapis.com/css2?family=MedievalSharp&display=swap'); /* Importa il font MedievalSharp da Google Fonts */
2
3 body {
4     margin: 0; /* Elimina il margine predefinito del body */
5     padding: 0; /* Elimina il padding predefinito */
6     font-family: 'MedievalSharp', cursive; /* Applica il font MedievalSharp, con fallback corsivo */
7     background: linear-gradient(to bottom, #fdf6e3, #f8eec1); /* Sfondo a gradiente dal beige chiaro a un giallo pallido */
8     color: #3b2f2f; /* Colore del testo marrone */
9     height: 100vh; /* Altezza dell'intera viewport */
10    display: flex; /* Imposta il contenitore come flexbox */
11    justify-content: center; /* Centra orizzontalmente */
12    align-items: center; /* Centra verticalmente */
13    flex-direction: column; /* Dispone i figli in colonna */
14 }
15
16 /* Contenitore del form */
17 form {
18     background: #fffffdc; /* Sfondo color crema (cornsilk) */
19     border: 3px solid #6b4226; /* Bordo marrone scuro spesso 3px */
20     border-radius: 14px; /* Angoli arrotondati */
21     padding: 30px; /* Spaziatura interna di 30px */
22     width: 90%; /* Larghezza responsiva: 90% del contenitore */
23     max-width: 450px; /* Larghezza massima: 450px */
24     box-shadow: 0px 8px 20px rgba(0, 0, 0, 0.3); /* Ombra per profondità */
25     text-align: center; /* Allinea testo al centro */
26     position: relative; /* Posizionamento relativo per pseudo-elementi */
27 }
28
29 /* Icona saetta magica */
30 form::before {
31     content: ">"; /* Inserisce una saetta prima del contenuto del form */
32     display: block; /* Mostra come blocco separato */
33     font-size: 48px; /* Dimensione grande dell'icona */
34     text-align: center; /* Allinea al centro */
35     color: #d4af37; /* Colore oro */
36     margin-bottom: 10px; /* Spazio sotto l'icona */
37 }
38
39 /* Titolo */
40 h1 {
41     color: #6b4226; /* Colore marrone scuro */
42     font-size: 28px; /* Dimensione del font */
43     margin-bottom: 20px; /* Spazio sotto il titolo */
44     text-shadow: 1px 1px 0 #d4af37; /* Ombra dorata leggera per effetto rilievo */
45 }
46
47 /* Etichette */
48 label {
```

```

public > assets > css > # style.css > ↗ #submit
40
47 /* Etichette */
48 label {
49   display: block; /* Visualizza ogni etichetta su una riga separata */
50   font-weight: bold; /* Testo in grassetto */
51   margin-bottom: 6px; /* Spazio sotto l'etichetta */
52   color: #6b4226; /* Colore marrone scuro */
53   text-align: left; /* Allinea il testo a sinistra */
54   margin-left: auto; /* Centra orizzontalmente */
55   margin-right: auto;
56   max-width: 300px; /* Larghezza massima delle etichette */
57 }
58
59 /* Input */
60 input[type="text"],
61 input[type="password"],
62 select {
63   width: 90%; /* Larghezza responsiva */
64   max-width: 300px; /* Larghezza massima */
65   padding: 10px; /* Spaziatura interna */
66   margin: 10px auto 15px auto; /* Margini sopra e sotto */
67   border: 2px solid #d4af37; /* Bordo dorato */
68   border-radius: 6px; /* Angoli arrotondati */
69   background: #f8f1d1; /* Sfondo giallo chiaro */
70   color: #3b2f2f; /* Colore del testo marrone */
71   font-size: 15px; /* Dimensione del testo */
72   display: block; /* Visualizzazione a blocco */
73   outline: none; /* Rimuove bordo blu al focus */
74 }
75
76 /* Effetto focus sugli input */
77 input[type="text"]:focus,
78 input[type="password"]:focus,
79 select:focus {
80   border-color: #b8860b; /* Cambia colore del bordo al focus (dark goldenrod) */
81   background-color: #ffffbe; /* Sfondo ancora più chiaro */
82 }
83
84 /* Select dropdown specifico */
85 select {
86   appearance: none; /* Rimuove lo stile nativo del browser */
87   background-image: url("data:image/svg+xml;utf8,<svg fill='darkgoldenrod' height='24' viewBox='0 0 24 24' width='24' xmlns='http://www.w3.org/2000/svg'><path d='M7 10l5 5 5-5z'/%>"); /* Aggiunge un'icona personalizzata a destra */
88   background-repeat: no-repeat; /* Non ripete l'immagine */
89   background-position: right 10px center; /* Posiziona l'icona a destra e centrata verticalmente */
90   background-size: 16px; /* Dimensione dell'icona */
91   cursor: pointer; /* Cambia il cursore al passaggio */
92 }
93

```

```

public > assets > css > # style.css > ↗ #submit
95 .torna_indietro {
96   font-family: 'MedievalSharp', cursive; /* Font coerente con il resto del layout */
97   background-color: #6b4226; /* Sfondo marrone */
98   color: #fff; /* Testo bianco */
99   border: none; /* Nessun bordo */
100  padding: 8px 14px; /* Spaziatura interna: 8px sopra/sotto, 14px laterali */
101  cursor: pointer; /* Cursore a forma di mano */
102  border-radius: 6px; /* Angoli arrotondati */
103  transition: background-color 0.3s ease; /* Transizione fluida per il colore di sfondo */
104 }
105
106 .torna_indietro:hover {
107   background-color: #4a2e1d; /* Cambia colore al passaggio del mouse */
108 }
109
110 /* Bottone */
111 #submit {
112   width: 90%; /* Larghezza responsiva */
113   max-width: 300px; /* Larghezza massima */
114   padding: 10px; /* Spaziatura interna */
115   background-color: #6b4226; /* Marrone scuro */
116   border: none; /* Nessun bordo */
117   border-radius: 6px; /* Angoli arrotondati */
118   font-size: 16px; /* Dimensione testo */
119   font-weight: bold; /* Testo in grassetto */
120   color: #fff8dc; /* Testo color crema */
121   cursor: pointer; /* Cambia il cursore al passaggio */
122   text-transform: uppercase; /* Tutte le lettere in maiuscolo */
123   transition: background 0.3s ease; /* Transizione sfondo */
124   margin: 0 auto; /* Centra il bottone */
125   display: block; /* Visualizzazione a blocco */
126 }
127
128 #submit:hover {
129   background-color: #4a2e1d; /* Sfondo più scuro al passaggio del mouse */
130 }
131
132 /* Link */
133 a {
134   display: block; /* Link su una riga separata */
135   text-align: center; /* Allineamento al centro */
136   color: #6b4226; /* Marrone scuro */
137   font-size: 17px; /* Dimensione del font */
138   font-weight: bold; /* Testo in grassetto */
139   margin-top: 20px; /* Spazio sopra al link */
140   text-decoration: none; /* Rimuove la sottolineatura */
141   transition: color 0.3s ease; /* Transizione del colore */
142 }

```

Acquisto pacchetti:

```
public > assets > css > # acquisto.css > ↵ h1
1  @import url('https://fonts.googleapis.com/css2?family=MedievalSharp&display=swap'); /* Importa il font MedievalSharp da Google Fonts */
2
3  body {
4    margin: 0; /* Rimuove il margine predefinito del body */
5    padding: 40px; /* Aggiunge 40px di spazio interno su tutti i lati */
6    font-family: 'MedievalSharp', cursive; /* Applica il font MedievalSharp, con fallback a un corsivo */
7    background: linear-gradient(to bottom, #fdf6e3, #f8ecl); /* Sfondo a gradiente dal beige chiaro al giallo pallido */
8    color: #2d1e1e; /* Colore del testo marrone scuro */
9    min-height: 100vh; /* Altezza minima: 100% della viewport */
10   display: flex; /* Imposta il contenitore come flexbox */
11   flex-direction: column; /* Gli elementi sono disposti in colonna */
12   align-items: center; /* Allinea orizzontalmente al centro */
13 }
14
15 /* Titolo */
16 h1 {
17   font-size: 3rem; /* Grandezza del font molto ampia */
18   color: #ffdf00; /* Colore giallo brillante */
19   text-shadow: 2px 2px 4px #000; /* Ombra nera per dare rilievo al testo */
20   margin-bottom: 30px; /* Spazio inferiore di 30px */
21 }
22
23 /* Pulsante acquisto pacchetto */
24 #buy-package {
25   background-color: #6f4e37; /* Marrone scuro */
26   color: #gold; /* Testo dorato */
27   border: 2px solid #gold; /* Bordo dorato spesso 2px */
28   font-weight: bold; /* Testo in grassetto */
29   border-radius: 12px; /* Angoli arrotondati di 12px */
30   padding: 12px 24px; /* Spaziatura interna: 12px sopra/sotto, 24px destra/sinistra */
31   transition: all 0.3s ease; /* Animazione fluida per tutte le proprietà */
32 }
33
34 #buy-package:hover {
35   background-color: #gold; /* Sfondo diventa dorato al passaggio del mouse */
36   color: #2ela09; /* Testo diventa marrone molto scuro */
37   border: 2px solid #2ela09; /* Bordo marrone scuro */
38 }
39
40 /* Container card */
41 #card-container {
42   display: flex; /* Imposta il container come flexbox */
43   flex-wrap: wrap; /* Permette agli elementi di andare a capo */
44   justify-content: center; /* Centra gli elementi orizzontalmente */
45   gap: 1.5rem; /* Spazio di 1.5rem tra gli elementi */
46 }
47
48 /* Card stile Harry Potter */
49 .card {
50   background-color: #rgba(26, 19, 13, 0.95); /* Sfondo marrone molto scuro con opacità */
51   border: 2px solid #gold; /* Bordo dorato */
52   border-radius: 20px; /* Angoli molto arrotondati */
53   width: 280px; /* Larghezza fissa della card */
54   height: 460px; /* Altezza fissa della card */
55   display: flex; /* Imposta come flex container */
56   flex-direction: column; /* Disposizione verticale degli elementi */
57   justify-content: space-between; /* Spazio distribuito tra gli elementi interni */
58   box-shadow: 0 0 25px #rgba(255, 215, 0, 0.4); /* Effetto luce dorata intorno */
59   transition: transform 0.3s ease, box-shadow 0.3s ease; /* Animazione su trasformazione e ombra */
60 }
61
62 .card:hover {
63   transform: scale(1.05); /* Ingrandisce leggermente la card */
64   box-shadow: 0 0 40px #rgba(255, 215, 0, 0.6); /* Aumenta l'intensità dell'ombra dorata */
65 }
66
67 /* Immagine */
68 .card-img-top {
69   height: 250px; /* Altezza fissa dell'immagine */
70   object-fit: cover; /* Copre tutta l'area mantenendo le proporzioni */
71   border-top-left-radius: 18px; /* Arrotonda l'angolo in alto a sinistra */
72   border-top-right-radius: 18px; /* Arrotonda l'angolo in alto a destra */
73 }
74
75 /* Corpo della card */
76 .card-body {
77   flex-grow: 1; /* Il corpo cresce per occupare lo spazio restante */
78   display: flex; /* Flex container */
79   flex-direction: column; /* Elementi disposti in colonna */
80   justify-content: space-between; /* Spazio tra contenuti e pulsante */
81   padding: 1rem; /* Spaziatura interna */
82   text-align: center; /* Testo centrato */
83   color: #fceabb; /* Colore del testo: giallo chiaro/avorio */
84 }
85
86 /* Pulsanti all'interno delle card */
87 .card-body .btn-success {
88   background-color: #2e8b57; /* Verde scuro (SeaGreen) */
89   border: 1px solid #c0b283; /* Bordo beige chiaro */
90   font-weight: bold; /* Testo in grassetto */
91   margin-top: 10px; /* Spazio sopra al pulsante */
92 }
```

```
public > assets > css > # acquisto.css > ↵ h1
46 }
47
48 /* Card stile Harry Potter */
49 .card {
50   background-color: #rgba(26, 19, 13, 0.95); /* Sfondo marrone molto scuro con opacità */
51   border: 2px solid #gold; /* Bordo dorato */
52   border-radius: 20px; /* Angoli molto arrotondati */
53   width: 280px; /* Larghezza fissa della card */
54   height: 460px; /* Altezza fissa della card */
55   display: flex; /* Imposta come flex container */
56   flex-direction: column; /* Disposizione verticale degli elementi */
57   justify-content: space-between; /* Spazio distribuito tra gli elementi interni */
58   box-shadow: 0 0 25px #rgba(255, 215, 0, 0.4); /* Effetto luce dorata intorno */
59   transition: transform 0.3s ease, box-shadow 0.3s ease; /* Animazione su trasformazione e ombra */
60 }
61
62 .card:hover {
63   transform: scale(1.05); /* Ingrandisce leggermente la card */
64   box-shadow: 0 0 40px #rgba(255, 215, 0, 0.6); /* Aumenta l'intensità dell'ombra dorata */
65 }
66
67 /* Immagine */
68 .card-img-top {
69   height: 250px; /* Altezza fissa dell'immagine */
70   object-fit: cover; /* Copre tutta l'area mantenendo le proporzioni */
71   border-top-left-radius: 18px; /* Arrotonda l'angolo in alto a sinistra */
72   border-top-right-radius: 18px; /* Arrotonda l'angolo in alto a destra */
73 }
74
75 /* Corpo della card */
76 .card-body {
77   flex-grow: 1; /* Il corpo cresce per occupare lo spazio restante */
78   display: flex; /* Flex container */
79   flex-direction: column; /* Elementi disposti in colonna */
80   justify-content: space-between; /* Spazio tra contenuti e pulsante */
81   padding: 1rem; /* Spaziatura interna */
82   text-align: center; /* Testo centrato */
83   color: #fceabb; /* Colore del testo: giallo chiaro/avorio */
84 }
85
86 /* Pulsanti all'interno delle card */
87 .card-body .btn-success {
88   background-color: #2e8b57; /* Verde scuro (SeaGreen) */
89   border: 1px solid #c0b283; /* Bordo beige chiaro */
90   font-weight: bold; /* Testo in grassetto */
91   margin-top: 10px; /* Spazio sopra al pulsante */
92 }
```

Visualizzazione dei due album:

```
public > assets > css > #_album_HP.css > ...
1  @import url('https://fonts.googleapis.com/css2?family=MedievalSharp&display=swap'); /* Importa il font MedievalSharp da Google Fonts */
2
3  body {
4    font-family: 'MedievalSharp', cursive; /* Applica il font MedievalSharp con stile corsivo */
5    background: linear-gradient(to bottom, #f8f0d0, #e8d8a8); /* Sfondo sfumato dal beige chiaro a sabbia */
6    color: #3b2ff; /* Colore del testo marrone scuro */
7    padding: 20px; /* Spaziatura interna generale della pagina */
8  }
9
10 h1 {
11   font-size: 32px; /* Dimensione del titolo */
12   text-align: center; /* Allineamento al centro */
13   color: #6b4226; /* Colore marrone ispirato alla pergamena */
14   margin-bottom: 30px; /* Spazio sotto il titolo */
15   text-shadow: 2px 2px #d4af37; /* Ombra dorata sul testo */
16 }
17
18 /* Card delle figurine */
19 .card {
20   background-color: #fff8dc; /* Sfondo crema (cornsilk) */
21   border: 2px solid #88860b; /* Bordo dorato/marrone */
22   border-radius: 12px; /* Angoli arrotondati */
23   box-shadow: 0 6px 15px rgba(0, 0, 0, 0.2); /* Ombra per dare profondità */
24   transition: transform 0.3s ease; /* Transizione fluida su hover */
25   position: relative; /* Necessario per eventuali elementi assoluti */
26 }
27
28 .card:hover {
29   transform: scale(1.03); /* Ingrandisce leggermente la card al passaggio del mouse */
30   box-shadow: 0 8px 20px rgba(0, 0, 0, 0.3); /* Ombra più intensa su hover */
31 }
32
33 .card-img-top {
34   border-top-left-radius: 10px; /* Arrotonda l'angolo in alto a sinistra dell'immagine */
35   border-top-right-radius: 10px; /* Arrotonda l'angolo in alto a destra dell'immagine */
36   max-height: 300px; /* Altezza massima dell'immagine */
37   object-fit: cover; /* L'immagine riempie l'area senza deformarsi */
38 }
39
40 .card-body {
41   text-align: center; /* Centra il contenuto testuale della card */
42   padding: 15px; /* Spaziatura interna */
43 }
44
45 .card-title {
46   font-size: 20px; /* Dimensione del titolo della card */
47   font-weight: bold; /* Grassetto */
48   color: #4b2e2; /* Marrone scuro */
49   margin-bottom: 10px; /* Spazio sotto il titolo */
50 }
```

```
public > assets > css > #_album_HP.css > ...
44
45 .card-title {
46   font-size: 20px; /* Dimensione del titolo della card */
47   font-weight: bold; /* Grassetto */
48   color: #4b2e2; /* Marrone scuro */
49   margin-bottom: 10px; /* Spazio sotto il titolo */
50 }
51
52 /* Casa (house) */
53 .card-text {
54   font-style: italic; /* Testo in corsivo */
55   font-size: 16px; /* Dimensione testo */
56   margin-bottom: 12px; /* Spazio sotto il paragrafo */
57   color: #5b3924; /* Marrone medio */
58 }
59
60 /* Bottoni */
61 .card .btn {
62   font-weight: bold; /* Grassetto */
63   font-size: 14px; /* Dimensione testo */
64   margin-top: 5px; /* Spazio sopra il bottone */
65   background-color: #6b4226; /* Sfondo marrone */
66   color: #fff8dc; /* Testo crema */
67   border: none; /* Nessun bordo */
68   border-radius: 6px; /* Angoli arrotondati */
69   padding: 8px 12px; /* Spaziatura interna */
70   transition: background 0.3s ease; /* Transizione fluida dello sfondo */
71 }
72
73 .card .btn:hover {
74   background-color: #4a2e1d; /* Colore più scuro su hover */
75 }
76
77 .card .scambia-btn {
78   display: block; /* Comportamento a blocco */
79   margin: 10px auto 20px auto; /* Centra il bottone con margini sopra e sotto */
80   background-color: #4b0082; /* Viola Hogwarts */
81   color: #fff8dc; /* Testo crema */
82   border: none; /* Nessun bordo */
83   border-radius: 6px; /* Angoli arrotondati */
84   padding: 8px 16px; /* Spaziatura interna */
85   font-size: 14px; /* Dimensione del testo */
86   font-weight: bold; /* Grassetto */
87   transition: background 0.3s ease; /* Transizione sul cambio sfondo */
88   width: fit-content; /* Larghezza pari al contenuto */
89 }
90
```

```

public > assets > css > # album_HP.css > ...
90
91 .card .scambia-btn:hover {
92 | background-color: □#360061; /* Viola più scuro su hover */
93 }
94
95 /* Modal per i dettagli */
96 .modal-content {
97 | background-color: □#ffffaf; /* Sfondo color crema chiaro */
98 | border: 2px solid □#d4af37; /* Bordo dorato */
99 | border-radius: 10px; /* Angoli arrotondati */
100 | font-family: 'MedievalSharp', cursive; /* Font coerente con il resto del design */
101 }
102
103 .modal-title {
104 | color: □#6b4226; /* Marrone scuro */
105 | font-size: 24px; /* Dimensione titolo del modale */
106 }
107
108 .modal-body p {
109 | font-size: 16px; /* Dimensione testo dei paragrafi */
110 | color: □#3b2fff; /* Testo marrone */
111 }
112
113 .modal-body img {
114 | border-radius: 10px; /* Arrotonda l'immagine */
115 | border: 2px solid □#b8860b; /* Bordo dorato */
116 }
117
118 /* Responsive: riduce margini sui dispositivi piccoli */
119 @media (max-width: 576px) {
120 .card-title {
121 | font-size: 18px; /* Titolo più piccolo su schermi stretti */
122 }
123
124 .card-text {
125 | font-size: 14px; /* Testo descrittivo ridotto */
126 }
127
128 .card .btn {
129 | font-size: 13px; /* Bottoni più piccoli */
130 | padding: 6px 10px; /* Padding ridotto */
131 }
132
133 .modal-body p {
134 | font-size: 14px; /* Testo modale leggermente più piccolo */
135 }
136 }

```

Dashboard e pagina acquista crediti:

```

public > assets > css > # dashboard.css > ↗#logout
1 @import url('https://fonts.googleapis.com/css2?family=MedievalSharp&display=swap'); /* Importa il font MedievalSharp da Google Fonts */
2
3 body {
4   margin: 0; /* Rimuove margini di default */
5   padding: 40px; /* Spaziatura interna su tutti i lati */
6   font-family: 'MedievalSharp', cursive; /* Applica il font MedievalSharp */
7   background: linear-gradient(to bottom, □#fdf6e3, □#f8eec1); /* Sfondo sfumato color pergamina */
8   color: □#2d1e14; /* Testo marrone scuro */
9   min-height: 100vh; /* Altezza minima pari a tutto lo schermo */
10  display: flex; /* Layout flessibile */
11  flex-direction: column; /* Colonne verticali */
12  align-items: center; /* Centra orizzontalmente il contenuto */
13 }
14
15 /* Stile generale dei pulsanti */
16 button.btn {
17   padding: 12px 20px; /* Spaziatura interna */
18   font-size: 16px; /* Dimensione del testo */
19   font-weight: bold; /* Grassetto */
20   border: none; /* Nessun bordo */
21   border-radius: 8px; /* Angoli arrotondati */
22   cursor: pointer; /* Cursore a mano al passaggio */
23   text-transform: uppercase; /* Tutto in maiuscolo */
24   transition: background 0.3s ease, transform 0.2s ease; /* Transizione fluida */
25   box-shadow: 0 4px 8px □rgba(0, 0, 0, 0.4); /* Ombra sotto il bottone */
26 }
27
28 /* Header stile pergamina */
29 .header-bar {
30   width: 100%; /* Larghezza piena */
31   background: □#6b4226; /* Sfondo marrone */
32   color: □#fff8dc; /* Testo color crema */
33   display: flex; /* Layout flessibile */
34   justify-content: space-between; /* Spaziatura tra gli elementi */
35   align-items: center; /* Allineamento verticale centrato */
36   padding: 15px 30px; /* Spaziatura interna */
37   box-shadow: 0 5px 10px □rgba(0,0,0,0.4); /* Ombra sotto */
38   font-size: 18px; /* Dimensione del testo */
39   border-bottom: 4px double □#d4af37; /* Bordo doppio dorato */
40 }
41
42 .header-bar h1 {
43   font-size: 28px; /* Dimensione del titolo */
44   margin: 0; /* Nessun margine */
45   text-shadow: 2px 2px 3px □rgba(0,0,0,0.5); /* Ombra scura */
46   color: □#ffd700; /* Colore oro */
47 }
48

```

```

public > assets > css > # dashboard.css > #logout
48
49 .credit-box {
50   display: flex; /* Disposizione in riga */
51   align-items: center; /* Allineamento verticale centrato */
52   background: #d4af37; /* Sfondo semitrasparente */
53   padding: 6px 14px; /* Spaziatura interna */
54   border-radius: 8px; /* Angoli arrotondati */
55   border: 1px solid #d4af37; /* Bordo dorato */
56 }
57
58 .credit-box p {
59   margin: 0 10px; /* Spaziatura laterale */
60   font-size: 18px; /* Dimensione testo */
61   color: #fff8dc; /* Colore crema */
62 }
63
64 .credit-box button {
65   background: #d4af37; /* Sfondo dorato */
66   border: none; /* Nessun bordo */
67   color: #6b4226; /* Testo marrone */
68   font-size: 18px; /* Dimensione testo */
69   padding: 5px 12px; /* Spaziatura interna */
70   border-radius: 6px; /* Angoli arrotondati */
71   cursor: pointer; /* Cursore a mano */
72   font-weight: bold; /* Grassetto */
73   transition: background 0.3s; /* Transizione sfondo */
74 }
75
76 .credit-box button:hover {
77   background: #e8c86e; /* Dorato più chiaro al passaggio */
78 }
79
80 /* Dashboard contenitore */
81 .dashboard, .dashboard-container {
82   background: #fff8dc; /* Sfondo crema */
83   color: #2d1e1e; /* Testo marrone scuro */
84   padding: 30px; /* Spaziatura interna */
85   border-radius: 16px; /* Angoli arrotondati */
86   max-width: 800px; /* Larghezza massima */
87   width: 100%; /* Larghezza piena */
88   box-shadow: 0px 8px 20px #rgba(0, 0, 0, 0.2); /* Ombra */
89   margin-top: 30px; /* Margine superiore */
90   text-align: center; /* Testo centrato */
91   border: 3px solid #6b4226; /* Bordo marrone */
92 }
93
94 .dashboard h1,

```

```

92 }
93
94 .dashboard h1,
95 .dashboard-container h2 {
96   color: #6b4226; /* Titolo marrone */
97   margin-bottom: 20px; /* Spazio sotto */
98   text-shadow: 1px 1px 0 #d4af37; /* Ombra dorata */
99 }
100
101 .dashboard button,
102 .dashboard-container button {
103   font-family: 'MedievalSharp', cursive; /* Font medievale */
104   background-color: #6b4226; /* Sfondo marrone */
105   color: #fff8dc; /* Testo crema */
106   border: none; /* Nessun bordo */
107   padding: 10px 20px; /* Spaziatura */
108   font-size: 16px; /* Dimensione testo */
109   border-radius: 8px; /* Angoli arrotondati */
110   margin: 10px; /* Margine */
111   cursor: pointer; /* Cursore */
112   transition: background-color 0.3s ease; /* Transizione */
113 }
114
115 .dashboard button:hover,
116 .dashboard-container button:hover {
117   background-color: #4a2eld; /* Colore più scuro su hover */
118 }
119
120 /* Credit management section */
121 .credit-management {
122   display: flex; /* Layout flessibile */
123   justify-content: space-around; /* Spaziatura orizzontale */
124   flex-wrap: wrap; /* Permette di andare a capo */
125   gap: 20px; /* Spaziatura tra gli elementi */
126   margin-top: 20px; /* Margine superiore */
127 }
128
129 .credit-box {
130   background: #f9e9c5; /* Sfondo beige */
131   border: 2px solid #6b4226; /* Bordo marrone */
132   padding: 20px; /* Spaziatura interna */
133   border-radius: 12px; /* Angoli arrotondati */
134   width: 45%; /* Larghezza del contenitore */
135   max-width: 400px; /* Larghezza massima */
136   box-shadow: 3px 3px 12px #rgba(0, 0, 0, 0.2); /* Ombra */
137   text-align: center; /* Testo centrato */
138   color: #2d1e1e; /* Colore testo */
139 }

```

```
public > assets > css > # dashboard.css > #logout
140
141 .credit-box h3 {
142   color: #6b4226; /* Marrone */
143   margin-bottom: 10px; /* Spazio sotto */
144 }
145
146 .credit-box button {
147   background-color: #2e8b57; /* Verde Serpeverde */
148   color: white; /* Testo bianco */
149   font-size: 15px; /* Dimensione testo */
150   font-weight: bold; /* Grassetto */
151   border: none; /* Nessun bordo */
152   padding: 8px 16px; /* Spaziatura interna */
153   margin-top: 10px; /* Margine superiore */
154   border-radius: 6px; /* Angoli arrotondati */
155   cursor: pointer; /* Cursore a mano */
156   transition: background 0.3s; /* Transizione fluida */
157 }
158
159 .credit-box button:hover {
160   background-color: #267b4b; /* Verde più scuro su hover */
161 }
162
163 /* Feedback */
164 #feedback {
165   color: #6b4226; /* Colore marrone */
166   margin-top: 20px; /* Margine superiore */
167   font-size: 16px; /* Dimensione testo */
168   font-style: italic; /* Corsivo */
169 }
170
171 /* Link stile Hogwarts */
172 a {
173   text-decoration: none; /* Nessuna sottolineatura */
174   color: #6b4226; /* Marrone */
175   font-weight: bold; /* Grassetto */
176   transition: color 0.3s; /* Transizione colore */
177 }
178
179 a:hover {
180   color: #d4af37; /* Dorato al passaggio */
181 }
182
183 /* Pulsante principale */
184 #viewMainAlbum.btn-primary {
185   background-color: #7f0909; /* Rosso Grifondoro */
186   color: white; /* Testo crema */
187 }
```

```
public > assets > css > # dashboard.css > #logout
183 /* Pulsante principale */
184 #viewMainAlbum.btn-primary {
185   background-color: #7f0909; /* Rosso Grifondoro */
186   color: white; /* Testo crema */
187 }
188
189 #viewMainAlbum.btn-primary:hover {
190   background-color: #5a0707; /* Rosso più scuro */
191   transform: scale(1.05); /* Ingrandimento */
192 }
193
194 /* Pulsante duplicati */
195 #viewDuplicatesAlbum.btn-secondary {
196   background-color: #0e1a40; /* Blu Corvonero */
197   color: white; /* Testo crema */
198 }
199
200 #viewDuplicatesAlbum.btn-secondary:hover {
201   background-color: #08112c; /* Blu più scuro */
202   transform: scale(1.05); /* Ingrandimento */
203 }
204
205 /* Contenitore dei button con gap */
206 .d-flex.gap-4 {
207   gap: 20px; /* Spaziatura tra gli elementi */
208 }
209
210 /* Contenitore centrato */
211 .container.text-center.mt-5 {
212   margin-top: 80px; /* Margine superiore */
213 }
214
215 #credit-value,
216 p span#credit-value,
217 p:has(#credit-value) {
218   color: black; /* Valore dei crediti in nero */
219 }
220
221 #logout {
222   padding: 10px 20px;
223   background-color: #e74c3c;
224   color: white;
225   border: none;
226   border-radius: 5px;
227   cursor: pointer;
228 }
229
230 /* Logout button hover */
231 #logout:hover {
232   background-color: #d9534f;
233 }
```

Pagina dei messaggi:

```
public > assets > css > # harrypotter.css > span b
1  @import url('https://fonts.googleapis.com/css2?family=MedievalSharp&display=swap'); /* Importa il font MedievalSharp da Google Fonts */
2
3  body {
4    font-family: 'MedievalSharp', cursive; /* Imposta il font MedievalSharp con fallback corsivo */
5    background-color: #fdf6e3; /* Colore di sfondo beige chiaro */
6    color: #2d1e1e; /* Colore del testo marrone scuro */
7    padding: 2rem; /* Padding interno di 2rem */
8    max-width: 800px; /* Larghezza massima del contenuto centrale */
9    margin: auto; /* Centra il contenuto orizzontalmente */
10 }
11
12 h1 {
13   text-align: center; /* Allinea il titolo al centro */
14   color: #6b4226; /* Colore marrone ispirato alla pergamena */
15   border-bottom: 3px double #6b4226; /* Bordo inferiore doppio, marrone */
16   padding-bottom: 10px; /* Spaziatura interna sotto il testo */
17   margin-bottom: 2rem; /* Spazio sotto il titolo */
18 }
19
20 .richiesta {
21   background: #ffff8dc; /* Sfondo color crema (cornsilk) */
22   border-left: 5px solid #6b4226; /* Bordo a sinistra spesso, marrone */
23   padding: 1rem; /* Spaziatura interna di 1rem */
24   margin-bottom: 1.5rem; /* Spazio sotto il blocco */
25   box-shadow: 2px 2px 10px #rgba(107, 66, 38, 0.2); /* Leggera ombra per effetto rilievo */
26   position: relative; /* Necessario per pseudo-elementi come ::after */
27 }
28
29 .richiesta::after {
30   content: ''; /* Pseudo-elemento vuoto per decorazione */
31   display: block; /* Si comporta come blocco */
32   height: 2px; /* Altezza di 2px */
33   background: #b5895e; /* Riga decorativa marrone chiaro */
34   margin-top: 1rem; /* Spazio sopra la riga */
35 }
36
37 button {
38   font-family: 'MedievalSharp', cursive; /* Font coerente con il resto del layout */
39   background-color: #6b4226; /* Sfondo marrone */
40   color: #fff; /* Testo bianco */
41   border: none; /* Nessun bordo */
42   padding: 8px 14px; /* Spaziatura interna: 8px sopra/sotto, 14px laterali */
43   cursor: pointer; /* Cursore a forma di mano */
44   border-radius: 6px; /* Angoli arrotondati */
45   transition: background-color 0.3s ease; /* Transizione fluida per il colore di sfondo */
46 }
47
48 button:hover {
49   background-color: #4a2e1d; /* Cambia colore al passaggio del mouse */
50 }
51
52 span b {
53   font-weight: bold; /* Rende in grassetto il testo all'interno di <span><b> */
54 }
55
56 span[style*="green"] {
57   color: #2e8b57 !important; /* Colore verde intenso (Serpeverde), forza l'override con !important */
58 }
59
60 span[style*="blue"] {
61   color: #1e90ff !important; /* Colore blu vivo (Corvonero), forza l'override con !important */
62 }
63
```

```
public > assets > css > # harrypotter.css > span b
37 button {
38   border: none; /* Nessun bordo */
39   padding: 8px 14px; /* Spaziatura interna: 8px sopra/sotto, 14px laterali */
40   cursor: pointer; /* Cursore a forma di mano */
41   border-radius: 6px; /* Angoli arrotondati */
42   transition: background-color 0.3s ease; /* Transizione fluida per il colore di sfondo */
43 }
44
45 button:hover {
46   background-color: #4a2e1d; /* Cambia colore al passaggio del mouse */
47 }
48
49 span b {
50   font-weight: bold; /* Rende in grassetto il testo all'interno di <span><b> */
51 }
52
53 span[style*="green"] {
54   color: #2e8b57 !important; /* Colore verde intenso (Serpeverde), forza l'override con !important */
55 }
56
57 span[style*="blue"] {
58   color: #1e90ff !important; /* Colore blu vivo (Corvonero), forza l'override con !important */
59 }
```

Pagina scegli-carta:

```
public > assets > css > # scegli_carta.css > ...
1  @import url('https://fonts.googleapis.com/css2?family=MedievalSharp&display=swap'); /* Importa il font MedievalSharp da Google Fonts */
2
3  body {
4      margin: 0; /* Rimuove margini */
5      padding: 40px; /* Spaziatura interna */
6      font-family: 'MedievalSharp', cursive; /* Font principale */
7      background: linear-gradient(to bottom, #fdf6e3, #f8eec1); /* Sfondo sfumato pergamena */
8      color: #2diele; /* Colore del testo */
9      min-height: 100vh; /* Altezza minima: 100% viewport */
10     display: flex; /* Layout flex */
11     flex-direction: column; /* Colonne verticali */
12     align-items: center; /* Allineamento orizzontale centrale */
13  }
14
15  /* Titolo principale */
16  h1 {
17      text-align: center; /* Centrato */
18      font-family: 'Cinzel Decorative', cursive; /* Font decorativo per effetto magico */
19      font-size: 3rem; /* Grandezza titolo */
20      color: #gold; /* Colore oro */
21      text-shadow: 2px 2px 4px #000; /* Ombra nera */
22      margin-bottom: 2.5rem; /* Spazio sotto */
23  }
24
25  /* Contenitore delle carte */
26  #offerte-container {
27      display: flex; /* Layout flessibile */
28      flex-wrap: wrap; /* Permette di andare a capo */
29      justify-content: center; /* Centra orizzontalmente */
30      gap: 2rem; /* Spaziatura tra le card */
31  }
32
33  /* Card con dimensioni fisse e stile coerente */
34  .card {
35      background-color: #rgba(26, 19, 13, 0.95); /* Marrone scuro semitrasparente */
36      border: 2px solid #gold; /* Bordo dorato */
37      border-radius: 20px; /* Angoli arrotondati */
38      width: 280px; /* Larghezza fissa */
39      height: 460px; /* Altezza fissa per uniformità */
40      display: flex; /* Layout interno flessibile */
41      flex-direction: column; /* Disposizione verticale */
42      justify-content: space-between; /* Spazio tra intestazione, contenuto e bottone */
43      box-shadow: 0 0 25px #rgba(255, 215, 0, 0.4); /* Ombra dorata */
44      transition: transform 0.3s ease, box-shadow 0.3s ease; /* Animazione al passaggio */
45  }
46
47  .card:hover {
48      transform: scale(1.05); /* Ingrandimento su hover */
49      box-shadow: 0 0 40px #rgba(255, 215, 0, 0.6); /* Ombra più intensa */
50  }
51
52  /* Immagine in alto nella card */
53  .card-img-top {
54      height: 250px; /* Altezza fissa */
55      object-fit: cover; /* Riempie mantenendo le proporzioni */
56      border-top-left-radius: 18px; /* Arrotondamento angolo sinistro alto */
57      border-top-right-radius: 18px; /* Arrotondamento angolo destro alto */
58      width: 100%; /* Larghezza piena */
59  }
60
61  /* Corpo della card */
62  .card-body {
63      flex-grow: 1; /* Si espande per riempire lo spazio disponibile */
64      display: flex; /* Layout interno flessibile */
65      flex-direction: column; /* Disposizione verticale */
66      justify-content: space-between; /* Spazio tra titolo, testo e bottone */
67      padding: 1rem; /* Spaziatura interna */
68      text-align: center; /* Testo centrato */
69  }
70
71  /* Titolo nella card */
72  .card-title {
73      font-family: 'Cinzel Decorative', cursive; /* Font decorativo */
74      color: #ffd700; /* Colore oro */
75      font-size: 1.4rem; /* Dimensione media */
76      margin-bottom: 0.5rem; /* Spazio sotto */
77  }
78
79  /* Testo descrittivo */
80  .card-text {
81      font-size: 1rem; /* Dimensione normale */
82      color: #ddd; /* Grigio chiaro */
83      font-style: italic; /* Corsivo */
84  }
85
86  /* Bottone nella card ("Conferma scambio") */
87  .card button {
88      display: block; /* Comportamento a blocco */
89      margin: 1rem auto 0.5rem auto; /* Centrato con margine */
90      background: linear-gradient(135deg, #6a0dad, #8a2be2); /* Sfondo viola sfumato */
91      color: #fff; /* Testo bianco */
92  }
```

```
public > assets > css > # scegli_carta.css > ...
34  .card {
35  }
36
37  .card:hover {
38      transform: scale(1.05); /* Ingrandimento su hover */
39      box-shadow: 0 0 40px #rgba(255, 215, 0, 0.6); /* Ombra più intensa */
40  }
41
42  /* Immagine in alto nella card */
43  .card-img-top {
44      height: 250px; /* Altezza fissa */
45      object-fit: cover; /* Riempie mantenendo le proporzioni */
46      border-top-left-radius: 18px; /* Arrotondamento angolo sinistro alto */
47      border-top-right-radius: 18px; /* Arrotondamento angolo destro alto */
48      width: 100%; /* Larghezza piena */
49  }
50
51  /* Corpo della card */
52  .card-body {
53      flex-grow: 1; /* Si espande per riempire lo spazio disponibile */
54      display: flex; /* Layout interno flessibile */
55      flex-direction: column; /* Disposizione verticale */
56      justify-content: space-between; /* Spazio tra titolo, testo e bottone */
57      padding: 1rem; /* Spaziatura interna */
58      text-align: center; /* Testo centrato */
59  }
60
61  /* Titolo nella card */
62  .card-title {
63      font-family: 'Cinzel Decorative', cursive; /* Font decorativo */
64      color: #ffd700; /* Colore oro */
65      font-size: 1.4rem; /* Dimensione media */
66      margin-bottom: 0.5rem; /* Spazio sotto */
67  }
68
69  /* Testo descrittivo */
70  .card-text {
71      font-size: 1rem; /* Dimensione normale */
72      color: #ddd; /* Grigio chiaro */
73      font-style: italic; /* Corsivo */
74  }
75
76  /* Bottone nella card ("Conferma scambio") */
77  .card button {
78      display: block; /* Comportamento a blocco */
79      margin: 1rem auto 0.5rem auto; /* Centrato con margine */
80      background: linear-gradient(135deg, #6a0dad, #8a2be2); /* Sfondo viola sfumato */
81      color: #fff; /* Testo bianco */
82  }
```

```

public > assets > css > # scegli_carta.css > ...
80   .card-text {
82     color: #ddd; /* Grigio chiaro */
83     font-style: italic; /* Corsivo */
84   }
85
86   /* Bottone nella card ("Conferma scambio") */
87   .card button {
88     display: block; /* Comportamento a blocco */
89     margin: 1rem auto 0.5rem auto; /* Centrato con margine */
90     background: linear-gradient(135deg, #6a0dad, #8a2be2); /* Sfondo viola sfumato */
91     color: #fff; /* Testo bianco */
92     border: none; /* Nessun bordo */
93     font-weight: bold; /* Grassetto */
94     padding: 0.6rem 1.2rem; /* Spaziatura */
95     border-radius: 12px; /* Angoli arrotondati */
96     cursor: pointer; /* Cursore a mano */
97     font-family: 'Cinzel Decorative', cursive; /* Font decorativo coerente */
98     box-shadow: 0 0 10px #8a2be2; /* Ombra viola */
99     transition: background 0.3s, transform 0.2s, box-shadow 0.3s; /* Animazioni */
100   }
101
102  .card button:hover {
103    background: linear-gradient(135deg, #8a2be2, #ba55d3); /* Sfumatura più chiara su hover */
104    transform: scale(1.05); /* Ingrandimento */
105    box-shadow: 0 0 15px #ba55d3; /* Ombra più intensa */
106  }
107

```

4.3.

JS

Registrazione e Accesso:

la gestione della registrazione e dell'accesso dell'utente le ho svolte tutte attraverso il server (nella sezione Nodejs)

Modifica dati dell'utente:

```

// Carica i dati dell'utente dal server
document.addEventListener('DOMContentLoaded', async () => {
  const response = await fetch('/getUser');
  if (response.ok) {
    const user = await response.json();
    document.getElementById('username').value = user.username;
    document.getElementById('email').value = user.email;
    document.getElementById('house_p').value = user.house_p;
  } else {
    alert("Errore durante il recupero dei dati dell'utente.");
  }
});

// Aggiorna i dati dell'utente
document.getElementById('editForm').addEventListener('submit', async (event) => {
  event.preventDefault();
  const username = document.getElementById('username').value;
  const password = document.getElementById('password').value;
  const email = document.getElementById('email').value;
  const supereore_p = document.getElementById('house_p').value;

  const response = await fetch('/update', {
    method: 'PUT',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ username, password, email, house_p })
  });

  if (response.ok) {
    alert("Dati aggiornati con successo.");
    window.location.href = '/dashboard.html';
  } else {
    alert("Errore durante l'aggiornamento dei dati.");
  }
});

```

funzione per acquistare i crediti:

```
1 //funzione per acquistare i crediti per acquistare i pacchetti di figurine
2 async function buyCredits(amount) {
3     const username = prompt("Inserisci il tuo username:");
4     if (!username) return alert("Username obbligatorio.");
5
6     try {
7         const response = await fetch('/buy-credits', {
8             method: 'POST',
9             headers: { 'Content-Type': 'application/json' },
10            body: JSON.stringify({ username, credits: amount }),
11        });
12
13        const message = await response.text();
14        document.getElementById('feedback').innerText = message;
15    } catch (error) {
16        console.error(error);
17        alert('Errore durante l\'acquisto dei crediti.');
18    }
19}
20|
```

questa invece e' la funzione per acquistare i pacchetti con i crediti:

```
const cardContainer = document.getElementById('card-container');
/* funzione per l'acquisto del pacchetto in cui nel div cardContainer andra' a caricare
le card (figurine) nel div*/
async function buyPackage() {
    cardContainer.innerHTML = '';

    try {
        //res variabile che contiene il risultato della fetch sotto
        const res = await fetch('/api/buy-package', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ userId })
        });
        //se il risultato portato dal server non da un valore corretto
        //visualizzo errore con alert
        if (!res.ok) {
            const error = await res.json();
            alert(error.message || 'Errore durante l\'acquisto del pacchetto.');
            return;
        }

        //invece se il res e' ok, salvo il contenuto nella variabile { characters }
        //e per ogni personaggio creo la card con la funzione createCharacterCard
        //per poi aggiungerla nel mio div cardContainer
        const { characters } = await res.json();
        characters.forEach(character => {
            const card = createCharacterCard(character, true, false, false);
            cardContainer.appendChild(card);
        });
    } catch (err) {
        console.error('Errore acquisto pacchetto:', err);
    }
}
```

funzione per caricare le carte dell'album principale dal db sul sito:

```
public > assets > js > JS app.js > loadAlbumFromServer
1  let userId = null;
2  //appena entro nelle pagine per visualizzare i due album (principale e dei doppioni)
3  //usando sempre la funzione fetchCurrentUser per controllare se utente e' autenticato
4  //se l'utente e' autorizzato, allora vengono eseguite immediatamente le funzioni per caricare le carte
5  //nella pagina, prese dal db
6  document.addEventListener('DOMContentLoaded', async () => {
7    await fetchCurrentUser();
8    if (userId) {
9      //album Principale
10     loadAlbumFromServer();
11     //album dei doppioni
12     loadDuplicatesFromServer();
13   }
14 });
15 //div in cui verranno caricate le carte
16 const album = document.getElementById('album');
17 const duplicatesAlbum = document.getElementById('duplicates-album');
18
19 //funzione per caricare le carte dell'album principale
20 async function loadAlbumFromServer() {
21   try {
22     const res = await fetch(`/album/${userId}`);
23     const characters = await res.json();
24
25     if (!Array.isArray(characters)) {
26       console.error('Album ricevuto non è un array:', characters);
27       return;
28     }
29
30     characters.forEach(character => {
31       const card = createCharacterCard(character, false, true, false);
32       album.appendChild(card);
33     });
34   } catch (err) {
35     console.error('Errore nel caricamento dell'album:', err);
36   }
37 }
38 }
```

funzione per caricare le figurine dell'album dei doppioni:

```
//funzione per caricare le carte dell'album dei doppioni
//a differenza delle carte dell'album principale che sono le carte che l'utente colleziona
//in aggiunta hanno un pulsante 'Scambio', in cui quando lo si preme viene eseguita la funzione preparaScambio
async function loadDuplicatesFromServer() {
  try {
    const res = await fetch(`/album/${userId}/doppioni`);
    const characters = await res.json();

    if (!Array.isArray(characters)) {
      console.error('Doppioni ricevuti non sono un array:', characters);
      return;
    }

    characters.forEach(character => {
      const card = createCharacterCard(character, false, true, true);

      // Aggiungi pulsante per lo scambio
      const scambiaBtn = document.createElement('button');
      scambiaBtn.textContent = 'Scambia';
      scambiaBtn.className = 'scambia-btn';
      scambiaBtn.addEventListener('click', () => {
        preparaScambio(character);
      });

      card.appendChild(scambiaBtn);
      duplicatesAlbum.appendChild(card);
    });
  } catch (err) {
    console.error('Errore nel caricamento dei doppioni:', err);
  }
}

//questa funzione viene creata per salvare la carta nel localStorage, visto che quando
//decido di scambiare quella carta, successivamente dovrò scegliere una carta doppiona
//di un altro utente per scambiarla con la mia
function preparaScambio(cartaDaScambiare) {
  localStorage.setItem('cartaProposta', JSON.stringify(cartaDaScambiare));
  window.location.href = '/scegli-carta';
```

la funzione per creare la carta ha diversi parametri: il primo (character) contiene dati della figurina, invece dal secondo in poi (quindi showAddButton, showInfoButton e showVendiButton) sono tutti valori booleani.

Li ho messi perche' questa funzione viene usata in diverse pagine, quindi se in una pagina ci devono essere 2 pulsanti che fanno quel compito metterò quei valori a 'true' se voglio che comparino, altrimenti 'false'.

Ho deciso di usarli per il riutilizzo del codice, se no avrei dovuto per ogni pagina che contiene quella funzione rifarla.

Funzione per creare la figurina:

```
//funzione per creare la carta, i parametri sono: character (personaggio);
//showAddButton, ovvero booleano (se true fa visualizzare il pulsante per aggiungere questo personaggio nel mio album principale o doppioni);
//showInfoButton, anch'esso booleano per far visualizzare questo pulsante nel div della card
function createCharacterCard(character, showAddButton = false, showInfoButton = true, showVendiButton = false) {
  const col = document.createElement('div');
  col.className = 'col-md-4 mb-4';

  col.innerHTML =
    `


        <h5 class="card-title">${character.name}</h5>
        <p class="card-text">${character.house} || 'Nessuna casata'</p>
        ${showAddButton ? `<button class="btn btn-success add-to-album">Aggiungi all'album</button>` : ''}
        ${showInfoButton ? `<button class="btn btn-success view-data-figurina">Visualizza altre info</button>` : ''}
        ${showVendiButton ? `<button class="btn btn-success vendi-figurina">Vendi figurina</button>` : ''}
      </div>
    </div>
  `;
}


```

e queste sono le condizione nel caso la variabile fosse stata 'true':

```
//se il booleano showAddButton e' vero, al click del bottone con class=add-to-album, esegui la funzione addToAlbum
if (showAddButton) {
  col.querySelector('.add-to-album').addEventListener('click', () => {
    addToAlbum(character);
  });
}

//se il booleano showInfoButton e' vero, al click del bottone con class=view-data-figurina,
//fai una fetch al server e con i dati della risposta chiama la funzione showModalWithCharacterData
if (showInfoButton) {
  col.querySelector('.view-data-figurina').addEventListener('click', async () => {
    try {
      const response = await fetch(`/character/${character.id}`);
      if (!response.ok) throw new Error('Errore nel recupero dei dati');
      const data = await response.json();
      showModalWithCharacterData(data);
    } catch (error) {
      console.error('Errore durante il fetch del personaggio:', error);
    }
  });
}
```

```

if(showVendiButton){
  col.querySelector('.vendi-figurina').addEventListener('click', async () => {
    try {
      fetch('/album/vendi', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          userId,
          characterId: character.id
        })
      })
      .then(res => res.json())
      .then(data => {
        if (data.error) return alert(data.error);
        alert(data.message);

        // Rimuove la card venduta dal DOM
        const cardElement = document.querySelector(`[data-id="${characterId}"]`);
        if (cardElement) cardElement.remove();
      })
    } catch (error) {
      console.error('Errore durante la vendita per crediti della figurina:', error);
    }
  });
}

```

funzione che viene eseguita dopo aver scelto la carta richiesta di un altro utente:

```

//funzione per confermare che voglio scambiare la mia cartaProposta, che era stata salvata nel localStorage
//con la cartaRichiesta dell'utente con id=userBId
function confermaScambio(userAId, userBId, cartaRichiesta) {
  const cartaProposta = JSON.parse(localStorage.getItem('cartaProposta'));
  if (!cartaProposta) {
    alert('Devi prima selezionare una carta da proporre per lo scambio!');
    return;
  }

  //fetch per inviare questa richiesta di scambio all'utenteB
  fetch('/api/richiesta-scambio', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      userAId,
      userBId,
      card: cartaProposta,
      cardB: cartaRichiesta
    })
  })
  .then(async res => {
    const data = await res.json();
    if (!res.ok) throw new Error(data.error || 'Errore durante l\'invio della richiesta');
    alert(data.message);
    localStorage.removeItem('cartaProposta');
    window.location.href = '/dashboard';
  })
  .catch(err => {
    console.error('Errore nella richiesta di scambio:', err);
    alert('Invio della richiesta fallito: ' + err.message);
    window.location.href = '/dashboard';
  });
}

```

funzione per cariche, dal db, le richieste di scambio e le risposte di ritorno legate a quell'utente:

```
// funzione per caricare le richieste legate all'utente prese dal db
async function caricaRichieste(userId) {
  const container = document.getElementById('richieste-container');
  container.innerHTML = '';

  try {
    const res = await fetch(`/api/richieste/${userId}`);
    const richieste = await res.json();

    if (richieste.length === 0) {
      container.innerHTML = '<p>Nessuna richiesta di scambio al momento.</p>';
      return;
    }

    richieste.forEach(async richiesta => {
      const div = document.createElement('div');
      div.classList.add('richiesta');

      const response = await fetch('/verifica-possesso', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({
          userId: userId,
          cartaOffertaId: richiesta.cartaOfferta.id
        })
      });

      const data = await response.json();

      const nomeCartaOfferta = data.possiedeCarta
        ? `${richiesta.cartaOfferta.name} *`
        : richiesta.cartaOfferta.name;
    });
  }
}
```

```
// Mostra lo stato aggiornato come messaggio
if (richiesta.stato !== 'in attesa') {
  div.innerHTML =
    `<p><strong>Scambio ${richiesta.stato}</strong><br>${richiesta.testo}</p>`;
} else {
  // Visualizzazione normale per richieste in attesa
  div.innerHTML =
    `any
<p><strong>${richiesta.mittente.username}</strong> ti offre
<span style="color: green;"><b>${nomeCartaOfferta}</b></span>
in cambio di
<span style="color: blue;"><b>${richiesta.cartaRichiesta.name}</b></span>
</p>`;
}

const accettaBtn = document.createElement('button');
acceittaBtn.textContent = 'Accetta';
acceittaBtn.style.marginRight = '10px';
acceittaBtn.onclick = () => rispondiARichiesta(richiesta._id, true);

const rifiutaBtn = document.createElement('button');
rifiutaBtn.textContent = 'Rifiuta';
rifiutaBtn.onclick = () => rispondiARichiesta(richiesta._id, false);

div.appendChild(acceittaBtn);
div.appendChild(rifiutaBtn);
}

container.appendChild(div);
};

} catch (err) {
  console.error('Errore nel caricare le richieste:', err);
  container.innerHTML = '<p>Errore nel caricamento delle richieste.</p>';
}
```

invece questa è la funzione di risposta alla richiesta (accetto o rifiuto lo scambio):

```
//funzione per l'accettazione o il rifiuto della richiesta di scambio
//in cui modifco alcuni dati riguardo la richiesta (esempio: da scambio
//diventa di tipo testo, perche' e' solo di conferma o rifiuto, ...)
//e la invio da utente2 a utentel, invece prima era il contrario
function rispondiARichiesta(idRichiesta, accettato) {
  fetch(`/api/richiesta/${idRichiesta}/rispondi`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ accettato })
  })
  .then(res => res.json())
  .then(data => {
    if (accettato) {
      // Dopo la conferma, esegue materialmente lo scambio
      fetch(`/api/scambio/${idRichiesta}`, {
        method: 'POST'
      })
      .then(res => res.json())
      .then(data => {
        alert(`Scambio completato: ${data.message}`);
        window.location.href = '/dashboard';
      })
      .catch(err => {
        console.error('Errore nello scambio:', err);
        alert('Errore durante lo scambio.');
      });
    } else {
      alert(data.message);
      window.location.href = '/album-duplicati';
    }
  })
  .catch(err => {
    console.error('Errore nella risposta alla richiesta:', err);
    alert('Errore nella gestione della richiesta.');
  });
}
```

4.4. API

Per l'utilizzo delle API legate ai personaggi di Harry Potter abbiamo usato link pubblico. L'unico punto in cui ho deciso di creare una funzione che prendesse i dati, che mi servivano, dei personaggi è la funzione seguente:

```
function getFromHarryPotter() {
  const hpApiUrl = 'https://hp-api.onrender.com/api/characters';
  return fetch(hpApiUrl)
    .then(response => response.json())
    .then(data => {
      const selected = [];
      const usedIndices = new Set();
      while (selected.length < 5 && usedIndices.size < data.length) {
        const index = getRandomInt(0, data.length - 1);
        if (!usedIndices.has(index)) {
          usedIndices.add(index);
          selected.push({
            id: index,
            name: data[index].name,
            species: data[index].species,
            house: data[index].house,
            image: data[index].image,
            gender: data[index].gender,
            eyeColour: data[index].eyeColour,
            hairColour: data[index].hairColour,
            actor: data[index].actor
          });
        }
      }
      return selected;
    })
    .catch(error => console.log('Errore nel recupero dati da HP API:', error));
}

function getRandomInt(min, max) {
  min = Math.ceil(min);
  max = Math.floor(max);
  return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

Questa funzione permette attraverso il link delle API di prendere i dati di 5 personaggi randomici, che nel applicazione web saranno le cinque carte trovate nel pacchetto.

Ho deciso di usare le API solo per questa funzione perché è l'unico punto in cui mi servivano i dati dei personaggi per poi inserirli nel mio database. Successivamente, tutte le altre funzioni o servizi del sito web, gestisco i dati attraverso il database.

Node.js

Node.js lo usato per gestire la parte server (back-end) dell'applicativo web attraverso l'aiuto dei pacchetti npm (express, ...). Dove in cui ci sono tutte le funzioni che sono in stretto contatto con comandi relativi a Mongoddb, al controllo degli errori e dei dati e alle funzioni principale che usufruiscono della gestione dei dati attraverso le API, che di solito il client non dovrebbe averne nessun controllo, almeno che non sia la modifica dei suoi dati personali o ad altre funzioni (passando sempre per il server).

Pacchetti npm usati:

```
js server.js > ⚡ getFromHarryPotter > ⚡ then() callback
1  const express = require('express');
2  const mongoose = require('mongoose');
3  const bodyParser = require('body-parser');
4  const bcrypt = require('bcrypt');
5  const path = require('path');
6  const axios = require('axios');
7
8  const app = express();
9  const port = 3000;
10
```

Queste righe invece è come vengono usati le API dei pacchetti sopra:

Le prime 2 righe si riferiscono che i parametri che vengono aggiunti nei percorsi possono essere anche degli oggetti complessi (tipo Array) invece di solo stringhe; e l'ultima riga ci permette di usare i file di codice nella directory ‘public’ insieme alle API del pacchetto Express.

```
6
7  app.use(bodyParser.urlencoded({ extended: true }));
8  app.use(bodyParser.json());
9  app.use(express.static(path.join(__dirname, 'public')));
0
```

Come annunciato precedentemente nella sezione riferita ai file javascript, ho gestito la registrazione e l'accesso interamente lato server.

Registrazione:

```
//rotta per la registrazione
app.post('/register', async (req, res) => {
  const { username, password, email, house_p } = req.body;
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const newUser = new User({ username, password: hashedPassword, email, house_p });
    await newUser.save();
    currentUser = username;
    res.redirect('/dashboard');
  } catch (err) {
    res.status(500).send('Errore durante la registrazione: ' + err.message);
  }
});
```

Accesso:

```
//rotta per il login
app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  try {
    const user = await User.findOne({ username });
    if (user && await bcrypt.compare(password, user.password)) {
      currentUser = username;
      res.redirect('/dashboard');
    } else {
      res.status(401).send('Credenziali non valide.');
    }
  } catch (err) {
    res.status(500).send('Errore durante l\'accesso: ' + err.message);
  }
});
```

adesso vediamo la rotta per acquistare un pacchetto in cui si trova l'utente con quel id; si verifica che sia loggato; si decrementano i crediti di 1 (ogni credito equivale ad un pacchetto); si trova l'album in cui andranno le figurine, se esiste altrimenti si crea; e successivamente si usano i dati dei personaggi trovati attraverso la funzione ‘getFromHarryPotter()’:

```
//rotta per l'acquisto effettivo del pacchetto di figurine
app.post('/api/buy-package', async (req, res) => {
  const { userId } = req.body;
  try {
    const user = await User.findById(userId);
    if (!user) return res.status(404).send('Utente non trovato');

    if (user.credits > 0) {
      user.credits -= 1;
      await user.save();

      // Controlla se l'album esiste
      let album = await Album.findOne({ owner: userId });

      // Se non esiste, crealo
      if (!album) {
        album = new Album({
          owner: userId,
          mainAlbum: [],
          duplicatesAlbum: []
        });
        await album.save();
        console.log('Album creato per il nuovo utente.');
      }

      const characters = await getFromHarryPotter();
      res.status(200).json({ characters, credits: user.credits });
    } else {
      res.status(400).json({ message: 'Crediti insufficienti!' });
    }
  } catch (error) {
    res.status(500).json({ message: 'Errore durante l\'acquisto del pacchetto.' });
  }
});
```

rotte per visualizzare gli album:

```
//rotte per la visualizzazione degli album
app.get('/album/:idUtente', async (req, res) => {
  try {
    const album = await Album.findOne({ owner: req.params.idUtente });
    if (!album) {
      return res.status(404).json({ error: 'Album non trovato' });
    }
    res.json(album.mainAlbum); // Questo ora è array di stringhe
  } catch (err) {
    res.status(500).json({ error: 'Errore del server' });
  }
});

app.get('/album/:idUtente/doppioni', async (req, res) => {
  try {
    const album = await Album.findOne({ owner: req.params.idUtente });
    if (!album) {
      return res.status(404).json({ error: 'Album non trovato' });
    }
    res.json(album.duplicatesAlbum); // Anche questo è array di stringhe
  } catch (err) {
    res.status(500).json({ error: 'Errore del server' });
  }
});
```

la rotta per aggiungere una figurina del pacchetto appena acquistato nel mio database:

```
// Aggiungi una carta all'album
app.post('/api/album/add', async (req, res) => {
  const { userId, character } = req.body;

  try {
    const album = await Album.findOne({ owner: userId });
    if (!album) {
      return res.status(404).json({ error: 'Album non trovato' });
    }

    // Controlla se la carta è già nel mainAlbum
    const inMain = album.mainAlbum.some(c => c.id === character.id);

    if (inMain) {
      album.duplicatesAlbum.push(character);
      await album.save();
      return res.json({ message: 'Aggiunto ai doppioni' });
    }

    album.mainAlbum.push(character);
    await album.save();
    res.json({ message: 'Personaggio aggiunto all\'album principale' });

  } catch (error) {
    console.error('Errore durante l\'aggiunta all\'album:', error);
    res.status(500).json({ error: 'Errore del server' });
  }
});
```

e dopo aver aggiunta la carta viene eliminata dalla pagina perché se resta posso cliccare ancora e si aggiunge di nuovo la carta senza dare errori.

Rotta per eliminare un mio doppione e venderlo per crediti:

```
// Vendì una carta duplicata
app.post('/album/vendi', async (req, res) => {
  const { userId, characterId } = req.body;

  try {
    const user = await User.findById(userId);
    if (!user) return res.status(404).json({ error: 'Utente non trovato' });
    const album = await Album.findOne({ owner: userId });

    // Rimuove la carta dai doppioni
    const initialLength = album.duplicatesAlbum.length;
    album.duplicatesAlbum = album.duplicatesAlbum.filter(card => card.id !== characterId);

    if (album.duplicatesAlbum.length === initialLength)
      return res.status(404).json({ error: 'Carta non trovata nei doppioni' });

    // Aggiunge 0.1 credit
    user.credits = (user.credits || 0) + 0.1;

    await user.save();
    await album.save();

    res.json({ message: 'Carta venduta con successo. +0.1 credit', newCredits: user.credits });
  } catch (err) {
    console.error('Errore nella vendita:', err);
    res.status(500).json({ error: 'Errore del server' });
  }
});
```

dopo aver deciso quale doppione (dei miei) voglio scambiare, passo alla pagina in cui devo scegliere una figurina doppione di un altro utente.

Questa rottta permette di caricare tutte le doppione degli altri utenti tolto l'utente loggato:

```
//rottta per caricare sulla pagina scelta-carta i doppioni degli altri utenti
app.get('/scambi/:idUtente/altre-doppioni', async (req, res) => {
  const { idUtente } = req.params;

  try {
    // Trova tutti gli album tranne quello dell'utente corrente
    const altriAlbum = await Album.find({ owner: { $ne: idUtente } });

    const tutteLeCarte = [];
    for (const album of altriAlbum) {
      album.duplicatesAlbum.forEach(carta => {
        tutteLeCarte.push({
          ...carta.toObject(),
          owner: album.owner.toString() // aggiungi info proprietario
        });
      });
    }

    res.json(tutteLeCarte);
  } catch (err) {
    res.status(500).json({ error: 'Errore durante il recupero dei doppioni' });
  }
});
```

rotta per la gestione dei messaggi per la conferma o il rigetto della richiesta di scambio:

```
//rotta dopo conferma o rifiuto dello scambio dal destinatario della richiesta
app.post('/api/richiesta/:id/rispondi', async (req, res) => {
  const { accettato } = req.body;

  try {
    const richiesta = await Messaggi.findById(req.params.id)
      .populate('mittente')
      .populate('destinatario')
      .populate('cartaOfferta')
      .populate('cartaRichiesta');

    if (!richiesta) {
      return res.status(404).json({ error: 'Richiesta non trovata' });
    }

    const stato = accettato ? 'accettato' : 'rifiutato';
    const testo = `Ciao, ho ${stato} la tua proposta di scambio del ${new Date().toLocaleString('it-IT')}.`;

    // ▲ Salviamo i riferimenti originali prima di sovrascrivere
    const original_cardRichiesta = richiesta.cartaRichiesta;
    const original_cardOfferta = richiesta.cartaOfferta;
    const originaleMittente = richiesta.mittente;
    const originaleDestinatario = richiesta.destinatario;

    // Trasformazione messaggio
    richiesta.tipo = 'testo';
    richiesta.testo = testo;
    richiesta.stato = stato;
    richiesta.cartaOfferta = original_cardOfferta;
    richiesta.cartaRichiesta = original_cardRichiesta;

    // < Inverti mittente/destinatario
    richiesta.mittente = originaleDestinatario._id;
    richiesta.destinatario = originaleMittente._id;

    await richiesta.save();

    res.status(200).json({ message: `Richiesta ${stato}. Messaggio trasformato e inviato al richiedente.` });
  } catch (err) {
    console.error('Errore nella gestione della risposta:', err);
    res.status(500).json({ error: 'Errore durante la risposta alla richiesta.' });
  }
});
```

questa rotta è una parte dello scambio di messaggi tra utenti. Comprende la parte finale in cui l'utente che riceve la richiesta di scambio deve scegliere (accettare o rifiutare).

In questo punto, per semplificare la decisione, ma anche la scelta della carta da scambiare (pagina scelta-carta.html), ho deciso di inserire un asterisco (*) nel caso in cui la carta scelta (o in questo caso da confermare) sia già in possesso dell'utente che sta scegliendo (o confermando):

```
//rotta per verificare che la cartaOfferta dal primo utente (mittente), non sia già
//in possesso dal secondo utente (destinatario)
app.post('/verifica-possezzo', async (req, res) => {
  try {
    const { userId, cartaOffertaId } = req.body;

    const album = await Album.findOne({ owner: userId });

    const possiede = album.mainAlbum.some(carta => carta.id === cartaOffertaId);

    res.json({ possiedeCarta: possiede });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: 'Errore del server' });
  }
});
```

Questa è la rotta per svolgere lo scambio:

```
//rotta per lo scambio
//non ho messo nessun controllo primario tra le carte perche' ho aggiunto
//per ogni carta gia' posseduta un asterisco (*) rendendo per l'utente lo scambio molto meno complesso
app.post('/api/scambio/:id', async (req, res) => {
  try {
    const richiesta = await Messaggi.findById(req.params.id)
      .populate('mittente')
      .populate('destinatario')
      .populate('cartaOfferta')
      .populate('cartaRichiesta');

    if (!richiesta || richiesta.stato !== 'accettato') {
      return res.status(400).json({ error: 'Scambio non valido o non accettato.' });
    }

    const userA = richiesta.destinatario;
    const userB = richiesta.mittente;
    const cardA = richiesta.cartaOfferta;
    const cardB = richiesta.cartaRichiesta;

    const albumA = await Album.findOne({ owner: userA.id });
    const albumB = await Album.findOne({ owner: userB.id });

    if (!albumA || !albumB) {
      return res.status(404).json({ error: 'Uno degli album non esiste' });
    }

    // Rimuovi carte dai doppioni
    albumA.duplicatesAlbum = albumA.duplicatesAlbum.filter(c => c.id !== cardA.id);
    albumB.duplicatesAlbum = albumB.duplicatesAlbum.filter(c => c.id !== cardB.id);

    // Aggiungi carte ai rispettivi mainAlbum
    albumB.mainAlbum.push(cardA);
    albumA.mainAlbum.push(cardB);

    await albumA.save();
    await albumB.save();

    res.status(200).json({ message: 'Scambio eseguito con successo!' });
  } catch (err) {
    console.error('Errore durante lo scambio:', err);
    res.status(500).json({ error: 'Errore del server durante lo scambio' });
  }
});
```

4.6.

MongoDB

Per utilizzare il database prima di tutto bisogna collegarlo al file node.js:

```
// MongoDB connection
mongoose.connect('mongodb+srv://corbetta:1234@pwm.ttn8fle.mongodb.net/PWM')
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.error(err));
```

dopo di che bisogna creare gli schemi delle varie collezioni.

Come elencato nella sezione delle tecnologie utilizzate, il mio progetto si basa su 3 collezioni:

- utenti:

```
//schema Utente
const userSchema = new mongoose.Schema([
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  email: { type: String },
  house_p: {
    type: String,
    enum: ['Gryffindor', 'Slytherin', 'Hufflepuff', 'Ravenclaw'],
    required: true
  },
  credits: { type: Number, default: 0 }
]);
```

- album:

```
//schema Album (contiene sia album principale che album dei doppioni)
const albumSchema = new mongoose.Schema({
  owner: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  mainAlbum: [{ id: Number, name: String, house: String, image: String, species: String, gender: String, eyeColour: String, hairColour: String, actor: String}],
  duplicatesAlbum: [{ id: Number, name: String, house: String, image: String, species: String, gender: String, eyeColour: String, hairColour: String, actor: String}]
});
const Album = mongoose.model('Album', albumSchema, 'albums');
```

- scambio messaggi:

```
const Album = mongoose.model('Album', albumSchema, 'albums');

//schema dei messaggi (richieste -> risposte)
const messaggioSchema = new mongoose.Schema({
  mittente: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  destinatario: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  tipo: { type: String, enum: ['scambio', 'testo'], required: true }, // per messaggi futuri
  testo: { type: String, default: '' },
  cartaOfferta: Object, // struttura della carta offerta
  cartaRichiesta: Object, // struttura della carta richiesta
  stato: { type: String, enum: ['in attesa', 'accettato', 'rifiutato', 'inviauto'], default: 'in attesa' },
  data: { type: Date, default: Date.now }
});
const Messaggi = mongoose.model('Messaggi', messaggioSchema, 'ExchangeRequests');
```

adesso vediamo un po' di esempi pratici su come ho cercato i dati nel database:

- cercare in base a un valore (in questo caso 'username'):

```
if (!currentUser) return res.status(401).send('Non autorizzato');
const user = await User.findOne({ username: currentUser });
if (!user) return res.status(404).send('Utente non trovato');
```

- cancellare un utente che abbia un certo valore:

```
try {
  await User.deleteOne({ username: currentUser });
  currentUser = null;
```

- se non esiste un valore nella collezione, si deve creare:

```
album = new Album({
  owner: userId,
  mainAlbum: [],
  duplicatesAlbum: []
});
await album.save();
```

- cercare dei valori tranne di un certo utente:

```
try {
    // Trova tutti gli album tranne quello dell'utente corrente
    const altriAlbum = await Album.find({ owner: { $ne: idUtente } });
}
```

- cercare delle richieste in base ad uno specifico destinatario con l'aggiunta della funzione populate(...). Questa funzione permette di condividere tutte le richieste sotto forma di Array con l'aggiunta dello username del mittente perché il mittente è solo un riferimento all'oggetto User, ma io voglio il valore anche del suo username:

```
try {
    const richieste = await Messaggi.find({ destinatario: req.params.userId })
        .populate('mittente', 'username');
```

- questa operazione restituisce un valore booleano (true o false) comparando l'id della carta con la carta offerta dall'utente che ha inviato la richiesta di scambio

```
const possiede = album.mainAlbum.some(carta => carta.id === cartaOffertaId);
```

5. Funzionamento:

Potrebbero anche esserci ulteriori modifiche grafiche, ma non funzionali.

Faro' vedere due funzionamenti fondamentali dell'applicazione web:

- 1) lo scambio di due figurine tra 2 utenti in cui ogni immagine verrà spiegata nei dettagli mostrando le interfacce pubblicate anche sopra.
- 2) l'acquisto di un pacchetto, dall'acquisto fino alla visualizzazione delle figurine.

1) SCAMBIO

L'operazione di scambio è semplice per l'utente, ma nel lato back-end è un po' più complessa perché non comprende solo lo scambio effettivo delle figurine tra gli utenti, infatti comprende anche una parte comunicativa tra gli utenti.

Dove sta la semplicità per l'utente?

Adesso lo vediamo passo per passo:

Dopo che l'utente (user1), quello che vuole eseguire lo scambio con un altro utente (user2), si è loggato ed è entrato nella sua pagina personale, user1 deve accedere all'album dei doppioni, visto che sono carte in più rispetto alla mia collezione principale logicamente è sensato che voglia vendere o scambiare solo i doppioni.



Quando a scelto la sua carta doppione che vuole scambiare basta che preme il pulsante ‘Scambio’ sotto la stessa carta e lo porta alla pagina in cui deve scegliere una dei doppiioni degli altri utente registrati su questa applicazione web.



E la scelta della figurina dal voler nella collezione dello user1 è una di quelle semplici per l’utente perché ho deciso di mettere un’aggiunta ad ogni figurina che già possiede lo user1 un simbolo (*), così l’utente può scegliere quella giusta che non ha nella collezione.

La figurina cerchiata è senza asterisco, quindi vuole dire che allo user1 gli manca alla collezione e quindi conferma lo scambio premendo il bottone sotto la carta.

Dopo aver premuto il pulsante viene creata una richiesta da mandare allo user2, ovvero l’utente che possiede la carta richiesta da user1.

```

_id: ObjectId('6846d9e6091a2a06b088835b')
mittente: ObjectId('6838e036153ba026c88def12')
destinatario: ObjectId('6838dea6153ba026c88dd425')
tipo: "scambio"
testo: ""
  > cartaOfferta: Object
  > cartaRichiesta: Object
  stato: "in attesa"
  data: 2025-06-09T12:56:06.796+00:00
  __v: 0

```

Questa richiesta è inizialmente in stato di attesa e resterà in quello stato fino a quando lo user2 non risponde alla richiesta dello user1; CartaOfferta è il doppione offerto da user1 per lo scambio e la cartaRichiesta è quella scelta da user1 nella pagina di scegli-carta.

Adesso si passa alla pagina personale dello user2 in cui deve confermare o rifiutare lo scambio con user1. Dopo essere loggato ed essere entrato nella sua pagina home per rispondere deve premere il pulsante ‘notifiche’:

Richieste di Scambio

Scambio accettato:
Ciao, ho accettato la tua proposta di scambio del 02/06/2025, 00:06:56.

simo ti offre **Fridwulfa** in cambio di **Mr Granger**

Accetta

Rifiuta

la richiesta inviata da user1 compare con un messaggio di predefinito in cui c'è lo username dello user1, la carta offerta (nome colorato in verde) da user1 e la carta (colorata in blu) di user2 che user1 vuole nella sua collezione e sotto i pulsanti per confermare lo scambio o rifiutarlo.

Anche qua ho fatto in modo di semplificare la decisione dell'utente mettendo, in caso la carta offerta fosse già presente nell'album principale dello user2, un asterisco.

In questo caso non è presente nessun asterisco di fianco al nome della figurina offerta, quindi possiamo confermare lo scambio senza problemi.

```
_id: ObjectId('6846d9e6091a2a06b088835b')
mittente: ObjectId('6838dea6153ba026c88dd425')
destinatario: ObjectId('6838e036153ba026c88def12')
tipo: "testo"
testo: "Ciao, ho accettato la tua proposta di scambio del 09/06/2025, 14:57:34..."
cartaOfferta: Object
cartaRichiesta: Object
stato: "accettato"
data: 2025-06-09T12:56:06.796+00:00
__v: 0
```

come si vede, dopo aver confermato lo scambio, la richiesta sul database viene cambiata in modo da modificare lo stato, visto che lo user2 ha preso una decisione (in questo caso affermativa), il tipo della richiesta perchè non ritorna allo user1 una richiesta di tipo scambio, ma solo una risposta testuale e ovviamente gli utenti vengono invertiti visto che la risposta parte da user2 e torna a user1 e non viceversa come prima.

Scambio accettato:
Ciao, ho accettato la tua proposta di scambio del 09/06/2025, 14:57:34.

Torna Indietro

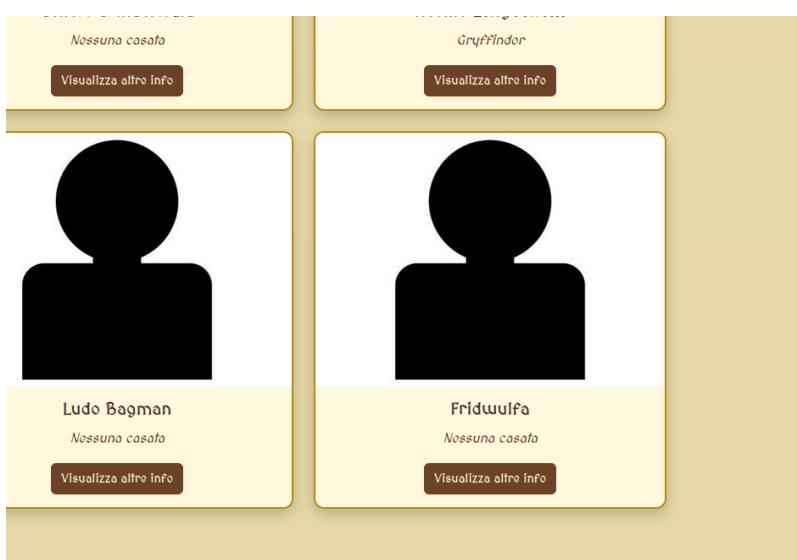
questa sopra è la risposta che viene visualizzata da user1 dopo la conferma o il rifiuto della richiesta iniziale.

Infatti nei rispettivi album principale in fondo ci sono le figurine scambiate correttamente e nell'album dei doppioni sono state cancellate:

- user1



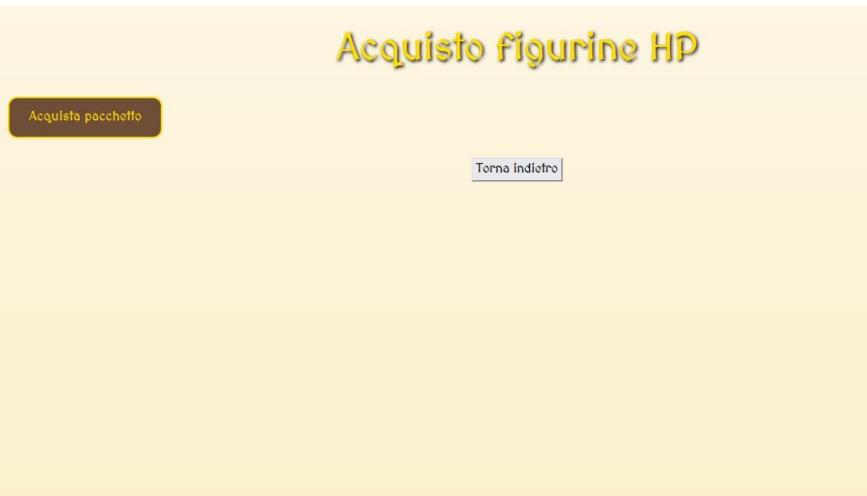
-user2



2) ACQUISTO PACCHETTO

Dopo che l'utente ha fatto l'accesso alla sua pagina principale la prima cosa da pensare prima di comprare dei pacchetti è quella di acquistare i crediti, valore usato per quel sito per l'acquisto dei pacchetti.

Successivamente aver preso abbastanza crediti, l'utente deve andare nella pagina dedicata a questa funzione:



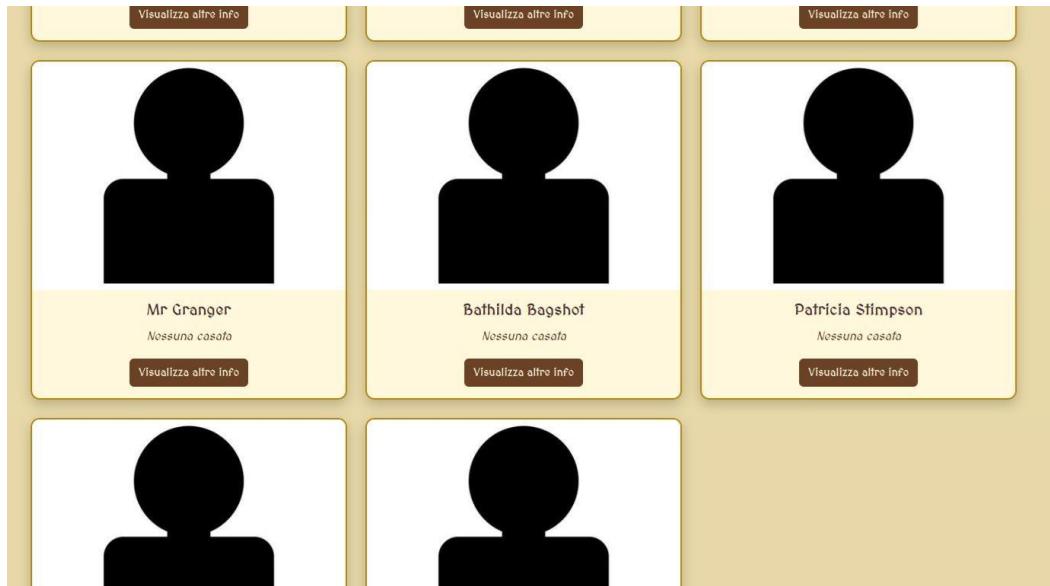
La pagina web inizialmente viene presentato in questo modo molto semplice e intuitivo, dove ci sono due pulsanti uno per tornare indietro e uno per l'acquisto del pacchetto di 5 figurine di Harry Potter.

Appena premuto viene acquistato un pacchetto e compaiono 5 figurine con un bottone 'aggiungi all'album':



Dopo di che si aggiungono tutte le carte al nostro album, se nuova nell'album principale altrimenti nell'album dei doppioni. Una cosa fondamentale è che quando aggiungo la carta, oltre ad aggiungerla nel database dedicato agli album, la figurina viene eliminata appena viene premuto il bottone perché così non do la possibilità all'utente di aggiungere due volte la stessa carta.

E per ultimo se andiamo a vedere nei vari album troviamo tutte le nostre figurine appena trovare nel pacchetto, ovviamente ho evidenziato solo l'album principale perché le prime due carte evidenziate nella foto precedente sono nuove:



6. Conclusione

Il progetto "Acquista e Scambia" è un'applicazione web progettata per offrire agli utenti un'esperienza di gioco sul collezionismo di figurine e l'interazione tra utenti per lo scambio di doppioni. Attraverso l'acquisto di pacchetti di figurine di Harry Potter, la visualizzazione dei propri album e lo scambio interattivo di figurine tra utenti.

La sua interfaccia semplice consente agli utenti di navigare facilmente tra le varie funzionalità, permettendo di vivere un'esperienza unica di gioco sul mondo di tutti i giorni: la virtualizzazione. La funzionalità di acquisto, sia di crediti ma soprattutto di pacchetti di figurine, e lo scambio di figurine attraverso un via vai di richiesta e risposte tra utenti sono un punti fondamentale dell'applicazione, consentendo agli utenti di personalizzare e arricchire la propria esperienza di gioco e di socializzazione con altri utenti.

In conclusione, il progetto "Acquista e Scambia" mira alla collezione e allo scambio delle figurine, offrendo un'esperienza semplice e coinvolgente. Grazie alle sue funzionalità e al design legato principalmente al tema Harry Potter, l'applicazione si presenta come uno strumento per tutti.

In futuro si potrà aggiungere magari la possibilità di aumentare le funzionalità dell'applicazione, magari rendendo lo scambio molto più interessante, per esempio una figurina per due o anche di utilizzando questo metodo dei messaggi tra utenti rendendo lo scambio molto coinvolgente sia per l'utente che propone lo scambio, ma anche l'altro.

Un'altra aggiunta molto interessante per ampliare e rendere migliore l'esperienza di gioco degli utenti potrebbe essere quella di aggiungere dei premi aggiuntivi per ogni obiettivo raggiunto tipo il numero di scambi, il numero di carte collezionate o anche aggiungere una nuova collezione rendendo il gioco continuativo e impegnativo.

7. Nota sitografica

URL:

<https://www.mongodb.com/>

<https://www.npmjs.com/package/express>

https://www.w3schools.com/js/js_api_intro.asp <https://www.html.it/pag/66525/fetch-api/>

Nominativo: Corbetta Simone

Matricola: 21577A

Corso: SSRI