



UNIVERSITÀ DI PARMA

Relazione del progetto di tecnologie internet

Zeudjio Tchifo Simon: 310314

Carattin Daniel: 311340

Indice

1. Introduzione.....	3
1.1 Contestualizzazione del progetto, scopo, obiettivi e breve descrizione del contenuto.....	3
2. Metodologia.....	4
2.1. Descrizione delle tecnologie utilizzate.....	4
2.2. Integrazione per soddisfare le esigenze del progetto.....	5
3. Implementazione delle funzioni.....	6
3.1 Utenti.....	6
3.1.1. HomeScreen.....	6
3.1.2. login.....	7
3.1.3. Registrazione.....	8
3.1.4. UserScreen.....	9
3.1.5. AddtoCart.....	10
3.1.6. CartScreen.....	11
3.1.7. Payment.....	12
3.1.8. GetOrders.....	12
3.1.9. UserProfile.....	13
3.2 Admin.....	14
3.2.1. AdminDashboard.....	14
3.2.1.1 UTENTI.....	15
3.2.1.1.1 GetAllUsers.....	15
3.2.1.1.2 AddUser.....	16
3.2.1.1.3AddAdmin.....	16
3.2.1.1.4 GetUser.....	17
3.2.1.1.5 UpdateUser.....	18
3.2.1.1.6 AdminOrders.....	18
3.2.1.1.7 DeleteUser.....	18
3.2.1.2 PRODOTTI.....	19
3.2.1.2.1 GetAllProducts.....	19
3.2.1.2.2 AddProduct.....	20
3.2.1.2.3 GetProduct.....	20
3.2.1.2.4 UpdateProduct.....	21
3.2.1.2.5 DeleteProduct.....	21
4. Configurazione.....	22
4.1. Database.....	22
4.2. Server.....	25
4.3. Client.....	26

1. INTRODUZIONE

1.1.CONTESTUALIZZAZIONE DEL PROGETTO, SCOPO, OBBIETTIVI E BREVE DESCRIZIONE DEL CONTENUTO

Il presente progetto rappresenta un'implementazione pratica di un'applicazione web per la gestione degli ordini online, più specificamente sulla vendita online di attrezzature per atletica leggera per la specialità dei lanci (lancio del disco, lancio del peso, lancio del giavellotto, lancio del martello e anche il lancio del vortex). Lo scopo principale è fornire un'interfaccia utente intuitiva e una piattaforma affidabile per consentire agli utenti di visualizzare, aggiungere, modificare ed eliminare ordini e prodotti. Gli obiettivi del progetto includono la comprensione delle tecnologie di sviluppo web, come il set MERN (MongoDB, Express.js, React.js, Node.js), l'architettura del sistema e la pratica nell'implementazione di funzionalità di base, come l'autenticazione degli utenti e la gestione del database. La relazione fornirà una descrizione dettagliata del processo di sviluppo, inclusa l'architettura del sistema, la selezione e l'utilizzo delle tecnologie, l'implementazione pratica delle funzionalità, nonché considerazioni e riflessioni sulle sfide incontrate e le future direzioni del progetto.

2. METOLOGIE

2.1. TECNOLOGIE UTILIZZATE E MOTOLOGIE DI SVILUPPO

Nel contesto del nostro progetto, abbiamo adattato una metodologia di sviluppo agile, che si basa su iterazioni rapide e feedback continuo per adattarsi ai cambiamenti dei requisiti e massimizzare il valore del progetto. Abbiamo scelto di utilizzare il set MERN (mongoDB, Express.js, React.js, Node.js) per il nostro stack tecnologico, poiché offre una serie di vantaggi in linea con le esigenze del nostro progetto.

MongoDB: Abbiamo optato per MongoDB come database NoSQL per la sua flessibilità e scalabilità. La struttura dei dati flessibile di MongoDB si adatta bene alle variazioni dei requisiti del progetto e consente una rapida iterazione nello sviluppo dei modelli dati.

Express.js: Abbiamo utilizzato Express.js come framework per lo sviluppo del back-end per la sua semplicità e flessibilità. Express.js semplifica la creazione di API RESTful, gestendo facilmente le richieste HTTP e semplificando il routing e la gestione degli endpoint.

React.js: Per il front-end, abbiamo scelto Rect.js per la sua modularità e facilità di sviluppo delle interfacce utente dinamiche e interattive. Con Rect.js, abbiamo potuto creare componenti riutilizzabili e gestire lo stato dell'applicazione in modo efficiente.

Node.js: Abbiamo utilizzato Node.js come ambiente di runtime per il nostro server back-end. Grazie alla sua architettura asincrona e non bloccante, Node.js è ideale per applicazioni ad alte prestazioni e scalabili.

Ruolo di JavaScript: È importante sottolineare che JavaScript è il linguaggio di programmazione principale utilizzato sia lato client che lato server nel nostro progetto. JavaScript è stato fondamentale per implementare la logica di front-end e back-end, consentendoci di sviluppare un'applicazione web full-stack coesa con coerenza tra front-end e back-end. La sua natura asincrona e non bloccante ha favorito lo sviluppo efficiente e scalabile del nostro progetto, consentendoci di gestire facilmente le operazioni di elaborazione e l'accesso ai dati in modo performante.

2.2 INTEGRAZIONE DELLE TECNOLOGIE

Il set MERN si integra in modo naturale, consentendo lo sviluppo di un'applicazione web full-stack coesa e con coerenza tra front-end e back-end. MongoDB funge da database di supporto per la sua persistenza dei dati, mentre Express.js fornisce un'interfaccia per la logica di business e l'esposizione delle API. React.js gestisce l'interfaccia utente dinamica lato client, interagendo con il back-end tramite chiamate API RESTful. Node.js agisce da middleware tra il front-end e il back-end, gestendo le richieste del client e orchestrando le operazioni di elaborazione e di accesso al database.

Dopo aver impostato il set MERN per lo sviluppo dell'applicazione web full-stack, è comune utilizzare **AXIOS** come libreria per gestire le richieste HTTP tra il client React.js e il server Node.js. Axios semplifica l'invio e la ricezione di dati dal back-end al front-end e viceversa, consentendo una comunicazione efficiente tramite chiamate API RESTful.

3. IMPLEMENTAZIONE DELLE FUNZIONI

3.1 UTENTI

3.1.1 HomeScreen

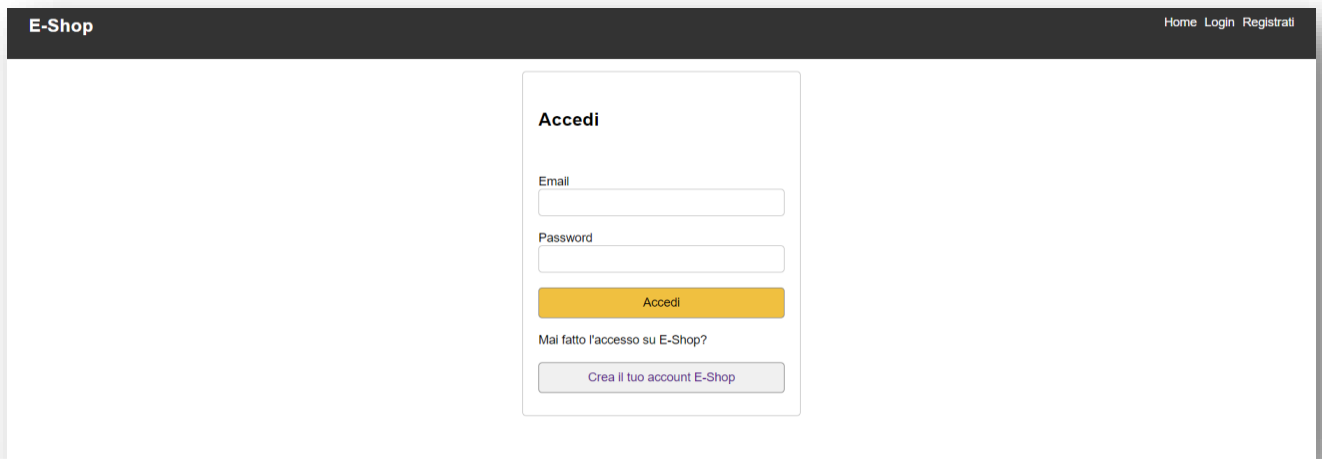
HomeScreen è la schermata principale appena il programma viene avviato, ci viene mostrata una pagina dove un utente può far l'accesso ed è in comune sia per i clienti sia per gli amministratori.

Come si può notare contiene una Navbar dove un utente può cliccare su login o registrazione e un Footer dove abbiamo qualche informazione per quello che riguarda il sito in generale.



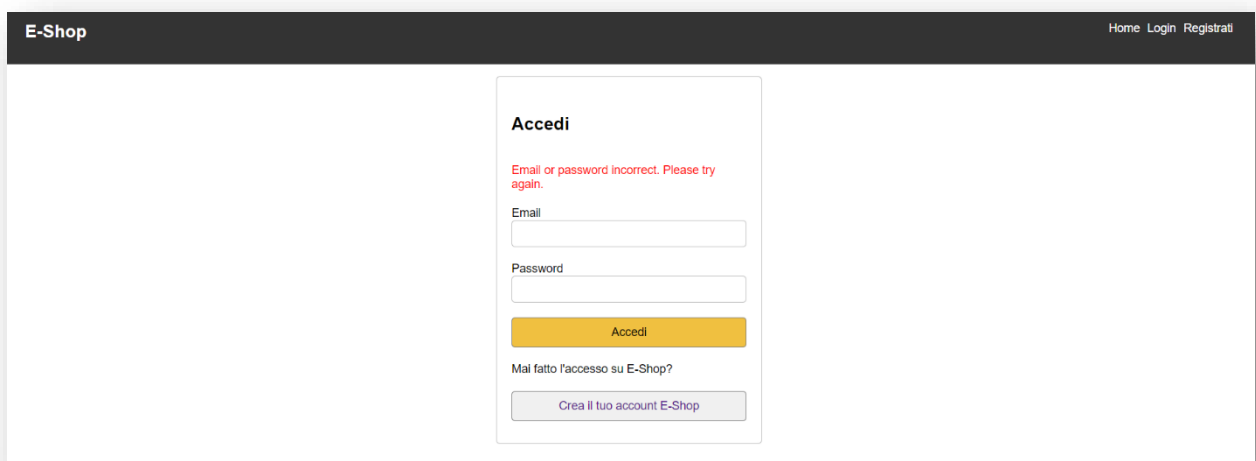
3.1.2 LOGIN

Nel login abbiamo una form che permette sia agli utenti che agli amministratori di accedere. Per distinguere un utente normale da un amministratore abbiamo salvato nel database una variabile `isAdmin` che di default è `false` per un utente che fa la registrazione, e al momento del login in base a sé la variabile è `true` o `false` ti indirizza alla pagina appropriata.



The screenshot shows the E-Shop login page. At the top, there is a dark header with 'E-Shop' on the left and 'Home Login Registrati' on the right. The main content area is white and contains a centered login form titled 'Accedi'. The form has two input fields: 'Email' and 'Password'. Below these fields is a yellow button labeled 'Accedi'. Under the button, there is a link that says 'Mai fatto l'accesso su E-Shop?' followed by a button labeled 'Crea il tuo account E-Shop'.

Una volta su questa pagina se l'utente non ha mai fatto l'accesso può cliccare su crea il tuo account E-Shop e verrà indirizzato sulla pagina di registrazione. Il login è stato implementato anche con la gestione degli errori.



This screenshot shows the E-Shop login page after an incorrect login attempt. The layout is identical to the previous one, but with an error message displayed in red text above the input fields: 'Email or password incorrect. Please try again.' The 'Accedi' button and the 'Crea il tuo account E-Shop' link remain visible.

E-Shop Home Login Registrati

Accedi

Email or password incorrect. Please try again.

Email
simon1@gmail.com

Password

Accedi

Mai fatto l'accesso su E-Shop?

Crea il tuo account E-Shop

3.1.3 REGISTRAZIONE

La Registrazione è riservata solo ai clienti perché come avevamo accennato pocanzi quando un utente viene creato la variabile `isAdmin` è di default `false`. Solo un amministratore già presente nel sistema può assegnare un nuovo amministratore.

E-Shop Home Login Registrati

Registrazione

Nome

Email

Password

Re-enter Password

Registrati

Ti sei già Registrato?

Accedi

E-Shop Home Login Registrati

Registrazione

Tutti i campi sono obbligatori.

Nome
a

Email

Password

Re-enter Password

Registrati

Ti sei già Registrato?

Accedi


In questa pagina di registrazione c'è una gestione di errori sulla compilazione di ogni campo, sulla verifica che non si inserisca un indirizzo mail già esistente, sul fatto che la password non può assomigliare al nome, sulla lunghezza della password e sul fatto che debba aver lettere e numeri e ovviamente la Password deve essere uguale a Re-enter Password.

Una volta che la registrazione è avvenuta con successo ci riporta sulla pagina di login e possiamo fare l'accesso.

3.1.4 UserScreen

Lo UserScreen è la pagina principale che appare quando l'utente fa il login con successo. Abbiamo una Navbar dove l'utente può visualizzare e modificare il proprio profilo, il carrello, gli ordini fatti e il logout e abbiamo anche la Home che praticamente ci porta alla pagina principale dell'utente.


Nello UserScreen abbiamo i prodotti su forma di paginazione e possiamo cliccare su singoli prodotti per la visualizzazione intera dell'immagine, possiamo aggiungere il prodotto al carrello grazie al button "Aggiungi al carrello", c'è anche una barra di ricerca per vari prodotti.

ZeuShop-Ecommerce Home Profile  Ordini Esci

Login successful. Welcome, utente1!

QUESTI SONO I NOSTRI PRODOTTI


Cerca prodotti...



Disco blu

Prezzo: €99


[Aggiungi al carrello](#)



Disco nero bordo argento

Prezzo: €159


[Aggiungi al carrello](#)




Disco nero bordo oro

Prezzo: €200


[Aggiungi al carrello](#)



Giavellotto 600g




Set giavellotti 800g, 700g e 600g



Set giavellotti(bianco rosso) 800g, 700g e 600g

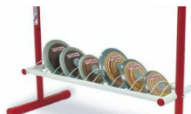
© 2024 Your Website | All rights reserved



Carrello blu per i dischi e altri attrezzi

Prezzo: €100


[Aggiungi al carrello](#)



Scaffale rosso bianco per dischi

Prezzo: €239


[Aggiungi al carrello](#)



Scaffale bianco per disco

Prezzo: €71

[Aggiungi al carrello](#)




Rastrelliera

Rastrelliera circolare per giavellotti

Prezzo: €200

[Aggiungi al carrello](#)




Bandierina

Bandierina per giudice gara

Prezzo: €11

[Aggiungi al carrello](#)



Rastrelliera Blu

Rastrelliera per giavellotti

Prezzo: €110

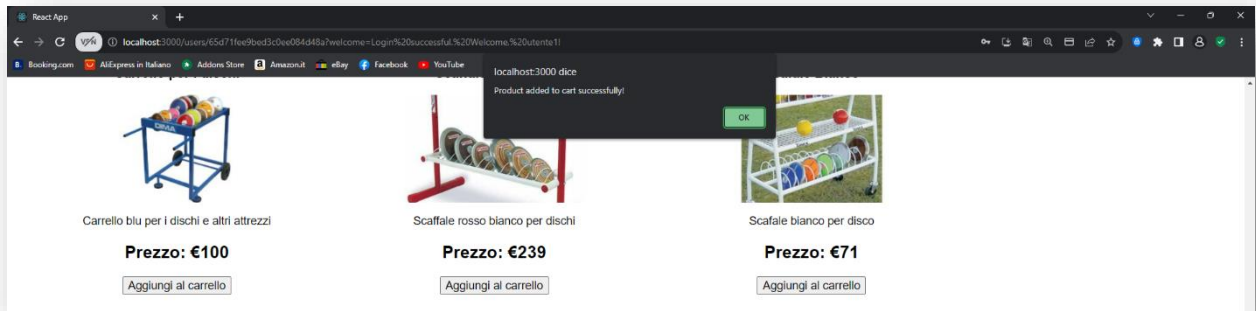
[Aggiungi al carrello](#)

1 2 3 4 5 6

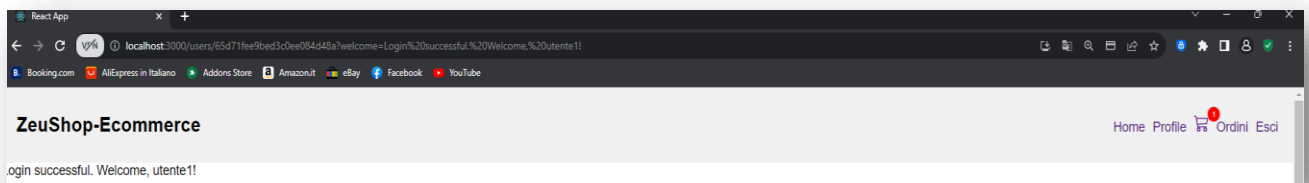
© 2024 Your Website | All rights reserved

3.1.5 AddtoCart

AddtoCart è la funzione che ci permette di aggiungere un prodotto al carrello, quando si aggiunge un prodotto al carrello esce un messaggio di successo da localhost e dopo il clic su ok la pagina si aggiorna.

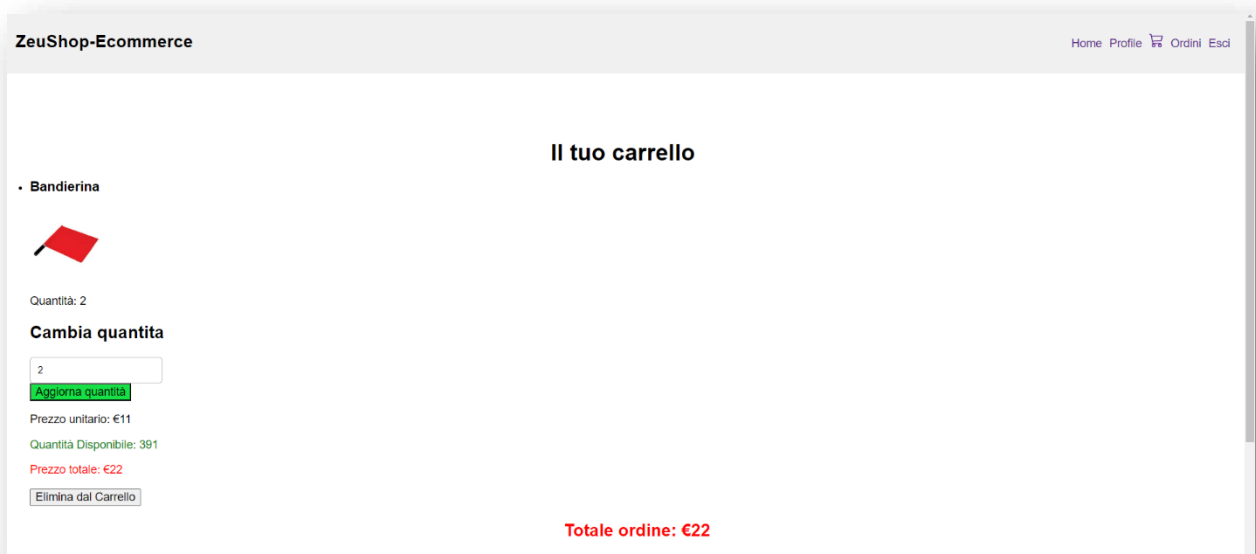


Poi la pagina viene aggiornata di conseguenza anche il carrello e ci viene mostrato la quantità di prodotti che abbiamo nel carrello.



3.1.6 CartScreen

CartScreen è la pagina del carrello dove possiamo vedere tutti i prodotti che abbiamo aggiunto. Nel CartScreen possiamo aggiornare la quantità dei prodotti aggiunti al carrello, li possiamo anche rimuovere dal carrello e vedere la quantità disponibile.



3.1.7 Payment

Payment è la funzione per il processo del pagamento, abbiamo una form nella stessa pagina di CartScreen dove chiediamo l'indirizzo di consegna, la modalità di pagamento con la gestione di errore per ogni campo. Ad esempio, il numero di carta deve essere valido, la data di scadenza valida e anche il CVV.

Dopo aver compilato tutti i campi correttamente si procede all'acquisto e ci appare un messaggio di successo che sparisce dopo qualche secondo.

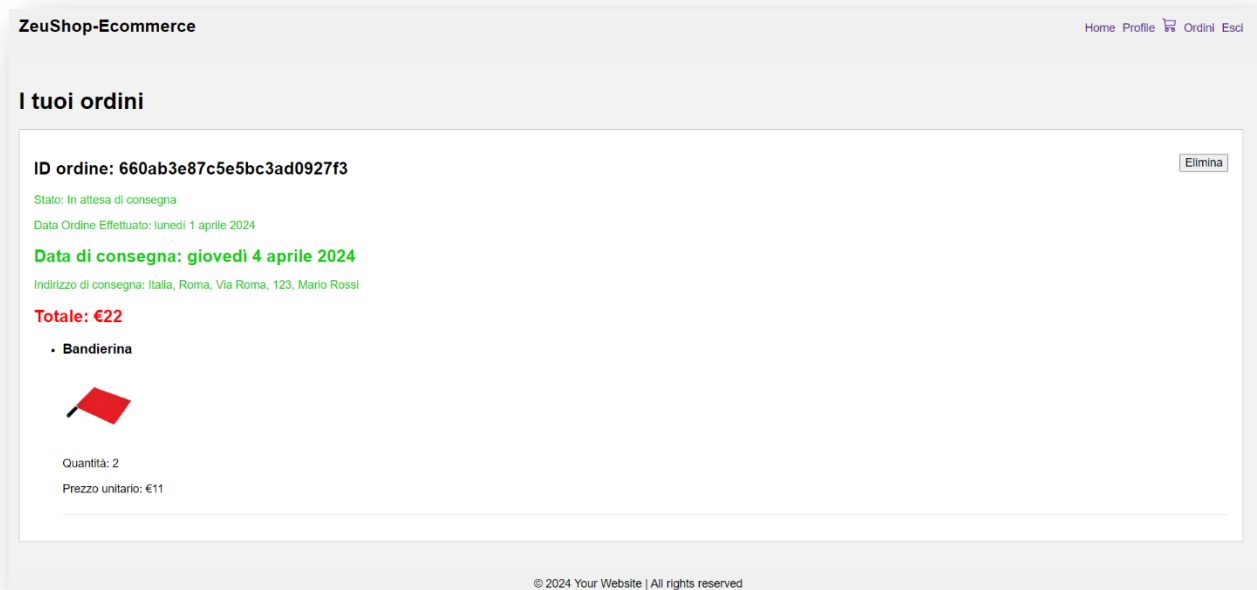
The screenshot displays a payment form with the following elements:

- Totale ordine: €22** (Total order: €22) in red text at the top right.
- Indirizzo di Consegna** (Delivery Address) section:
 - Label: **Indirizzo di Consegna:**
 - Input field containing: `Italia, Roma, Via Roma, 123`
- Metodo di Pagamento** (Payment Method) section:
 - Label: **Numero Carta:**
 - Input field containing: `1234567812345678`
 - Label: **Data di Scadenza:**
 - Input field containing: `11/24`
 - Label: **CVV:**
 - Input field containing: `123`
- ACQUISTA ORA** (BUY NOW) button in yellow.
- Footer: `© 2024 Your Website | All rights reserved`

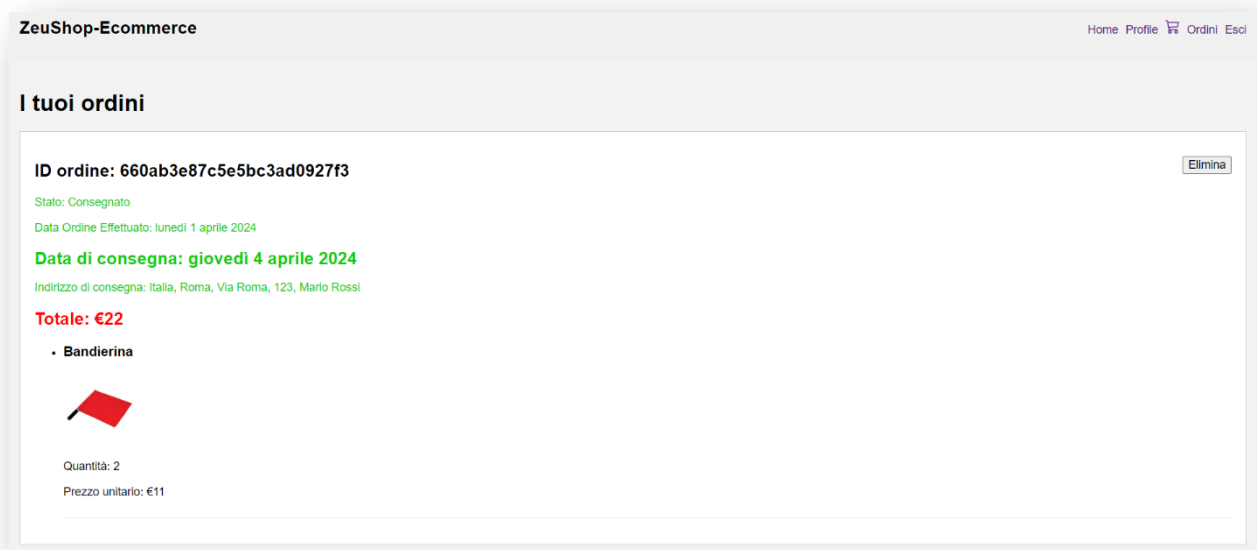
Quando il pagamento avviene con successo, il carrello si svuota e ovviamente la quantità di prodotti si aggiorna nel database.

3.1.8 GetOrders

GetOrders ci permette di vedere i prodotti acquistati, la data e l'ora dell'acquisto ed eventualmente il giorno di consegna che abbiamo impostato a 72 ore dal pagamento però anche in questo caso abbiamo considerato i giorni non lavorativi. Se al calcolo dei tre giorni di consegna se il terzo giorno cade uno dei gironi non lavorativi impostati dal sistema si va avanti di un giorno e così via. Nella pagina degli ordini abbiamo anche una variabile 'Stato' salvato nel database che di default è "In attesa consegna" dopo i tre giorni lo stato cambia a "consegnato".



Come dicevamo pocanzi, in questo caso si può vedere come lo stato è passato da “In attesa di consegna” a “Consegnato”.



Nella pagina di ordini abbiamo anche una funzione per eliminare gli ordini e chiede la conferma di eliminazione dell'ordine.

ID ordine: 660ab3e87c5e5bc3ad0927f3
Elimina

Stato: In attesa di consegna
 Data Ordine Effettuato: lunedì 1 aprile 2024
Data di consegna: giovedì 4 aprile 2024
 Indirizzo di consegna: Italia, Roma, Via Roma, 123, Mario Rossi

Totale: €22

• Bandierina


Quantità: 2
 Prezzo unitario: €11

Sei sicuro di voler eliminare questo ordine?

3.1.9 UserProfile

UserProfile permette a un utente di visualizzare e modificare i suoi dati se necessario.

ZeuShop-Ecommerce
 Home Profile  Ordini Esci

Il Mio Profilo

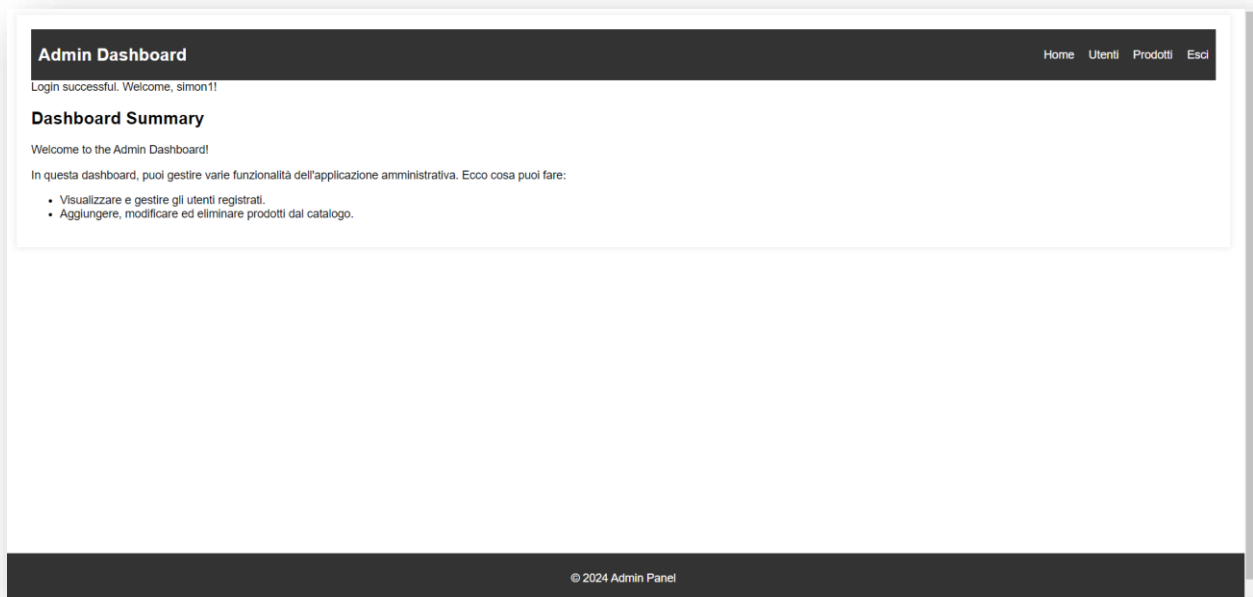
Nome
 Email
 Nuova Password

Come ultima funzione abbiamo il logout che sarebbe il pulsante esci in alto a destra. In questo caso abbiamo usato `LocalStorage`, una property di javascript che ci permette di salvare i dati dell'utente appena fa l'accesso, quando clicca su logout viene tolto dal `LocalStorage` e viene indirizzato alla home principale dove può fare il login di nuovo.

3.2 Admin

3.2.1 AdminDashboard

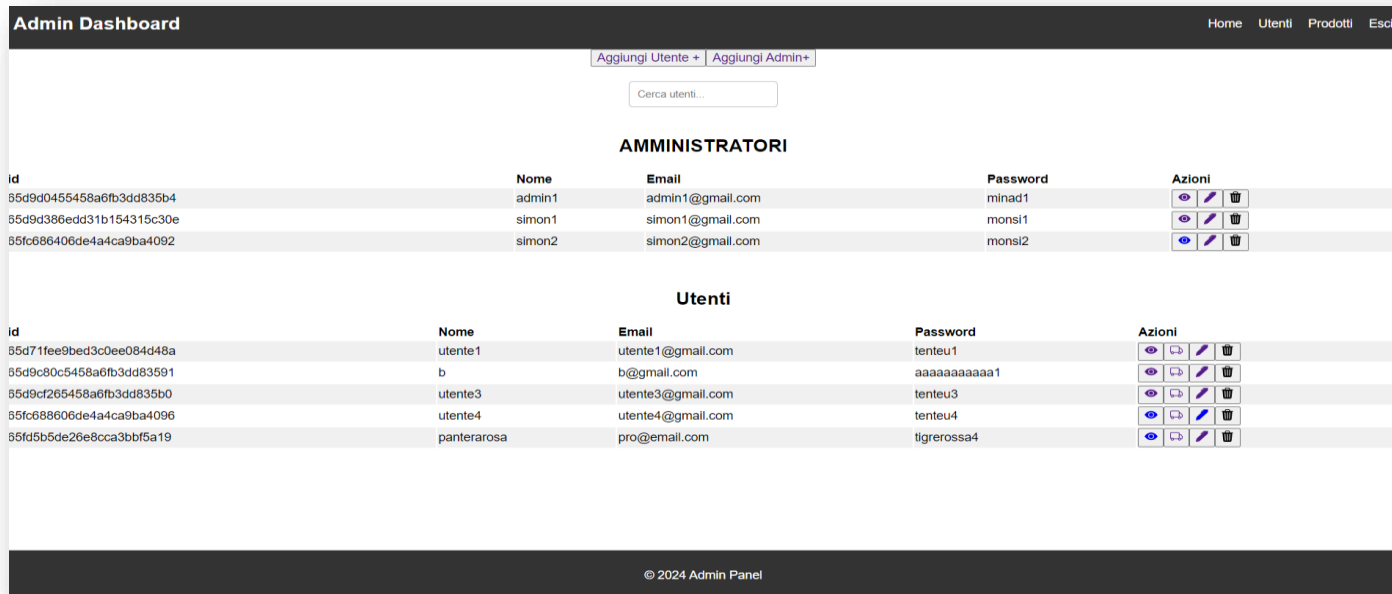
AdminDashboard è la pagina principale dell'amministratore che viene mostrata quando un utente con la variabile `isAdmin` a `true` fa l'accesso. In questa pagina l'admin ha l'opportunità di gestire sia utenti sia prodotti.



3.2.1.1 UTENTI

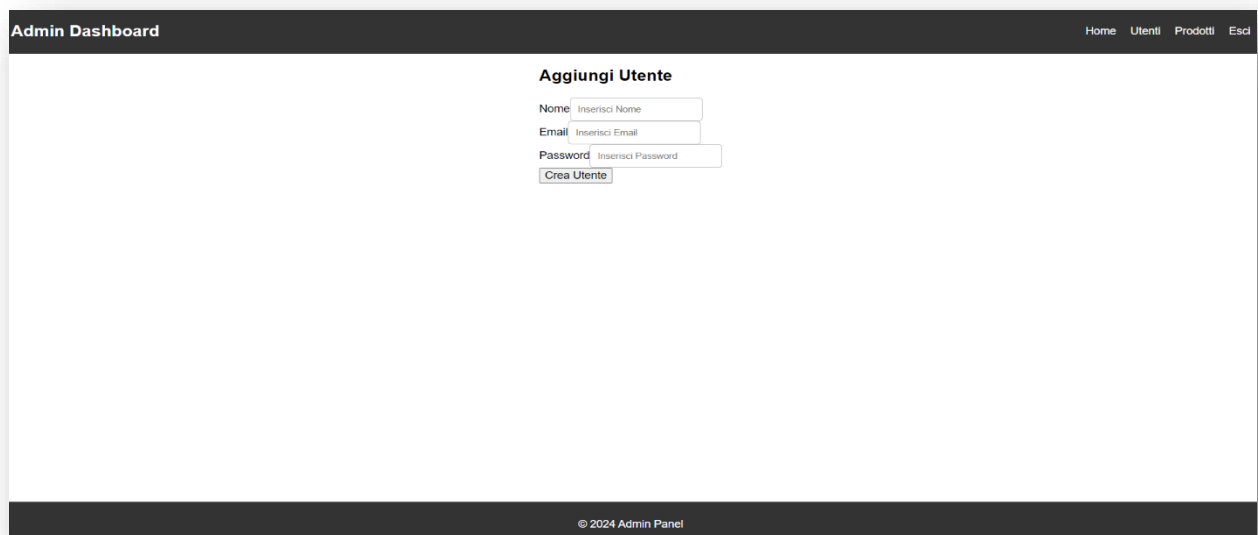
3.2.1.1.1 GetAllUsers

Quando clicchiamo su Utenti ci viene mostrata una pagina di tutti gli utenti compresi anche gli amministratori e in questa pagina si possono compiere diverse operazioni come aggiungere un utente o un admin, vedere le informazioni modificare ed eliminare un utente, si possono anche vedere tutti gli ordini di un utente se quest'ultimo non li ha cancellati. In questa pagina abbiamo anche una barra di ricerca per cercare un utente qualsiasi. Gli utenti vengono mostrati in una tabella dove abbiamo le informazioni essenziali come l'Id, nome, E-mail e password.



3.2.1.1.2 AddUser

AddUser ci permette di aggiungere un utente. Anche in questo caso abbiamo una gestione di errore ben implementata e vari controlli in tutti i campi. Per l'aggiunta di un utente abbiamo la variabile `isAdmin` a `false`.



3.2.1.1.3 AddAdmin

Per l'aggiunta di un nuovo amministratore abbiamo ovviamente la variabile `isAdmin` a `true`.

The screenshot shows the 'Crea Admin' form within the 'Admin Dashboard'. The dashboard has a dark header with 'Admin Dashboard' on the left and navigation links 'Home', 'Utenti', 'Prodotti', and 'Esci' on the right. The form is centered and contains the following fields: 'Nome:' with a text input 'Inserisci Nome', 'Email:' with a text input 'Inserisci email', and 'Password:' with a text input 'Inserisci Password'. Below these fields is a 'Crea Admin' button. The footer of the dashboard is dark and contains the text '© 2024 Admin Panel'.

3.2.1.1.4 GetUser

GetUser è la funzione che ci permette di vedere le informazioni di un utente. Ad esempio, in questo caso abbiamo l'Id, Nome, E-mail e Password dell'utente.

The screenshot shows the 'Info Utente' page within the 'Admin Dashboard'. The dashboard has a dark header with 'Admin Dashboard' on the left and navigation links 'Home', 'Utenti', 'Prodotti', and 'Esci' on the right. The page displays the following user information: 'ID: 65d71fee9bed3c0ee084d48a', 'Nome: utente1', 'Email: utente1@gmail.com', and 'Password: tenteu1'. The footer of the dashboard is dark and contains the text '© 2024 Admin Panel'.

3.2.1.1.5 UpdateUser

Questa è la funzione che ci permette di aggiornare i dati di un utente. Anche in questo caso c'è una gestione accurata degli errori e i controlli necessari.

The screenshot shows the 'Admin Dashboard' interface. At the top, there's a navigation bar with 'Home', 'Utenti', 'Prodotti', and 'Escl'. The main content area is titled 'Aggiorna Utente'. It contains a form with the following fields: 'Nome' (containing 'utente1'), 'Email' (containing 'utente1@gmail.com'), and 'Password' (containing '*****'). Below these fields is a green 'Aggiorna' button. At the bottom of the dashboard, there's a footer that says '© 2024 Admin Panel'.

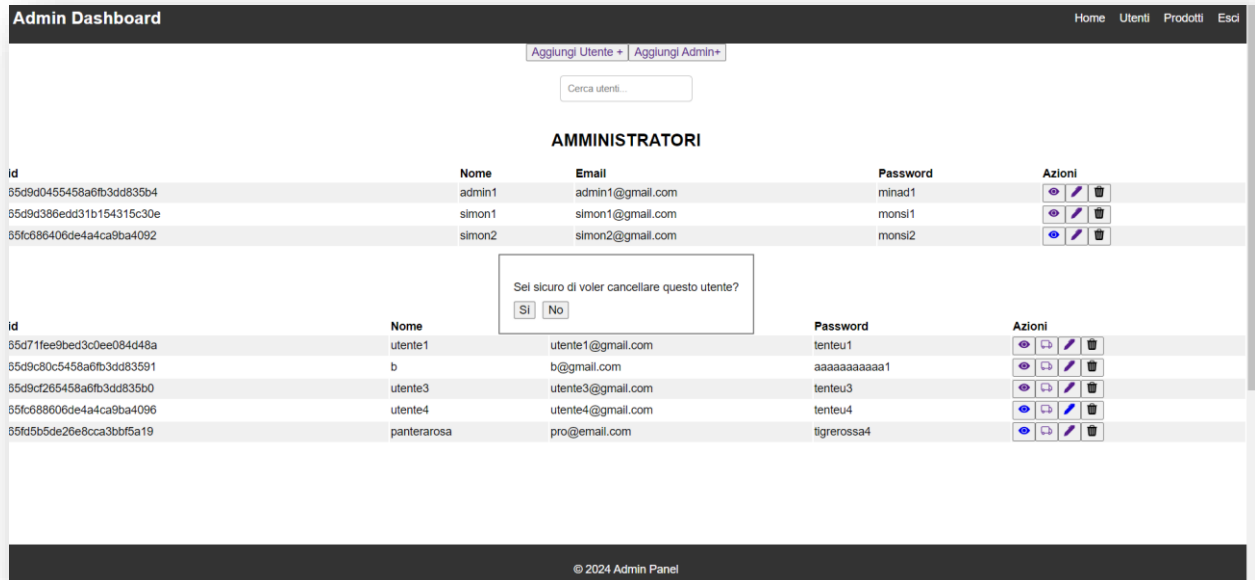
3.2.1.1.6 AdminOrders

AdminOrders permette di vedere gli ordini effettuati da un utente. In questo caso possiamo vedere anche cosa contiene l'ordine con alcune informazioni riguardante i prodotti.

The screenshot shows the 'Admin Dashboard' interface. At the top, there's a navigation bar with 'Home', 'Utenti', 'Prodotti', and 'Escl'. The main content area is titled 'Ordini dell'Utente 65d71fee9bed3c0ee084d48a'. It displays a list of orders. The first order is shown with the following details: 'Order ID: 660ab3e87c5e5bc3ad0927f3', 'Created At: 2024-04-01T13:17:28.845Z', and 'Prezzo Totale: €22'. Under 'Prodotti:', there's a list item 'Nome Prodotto: Bandierina' with a red flag icon. Below the icon, it shows 'Quantità: 2', 'Descrizione Prodotto: Bandierina per giudice gara', and 'Brand Prodotto: DIMA'. At the bottom of the dashboard, there's a footer that says '© 2024 Admin Panel'.

3.2.1.1.7 DeleteUser

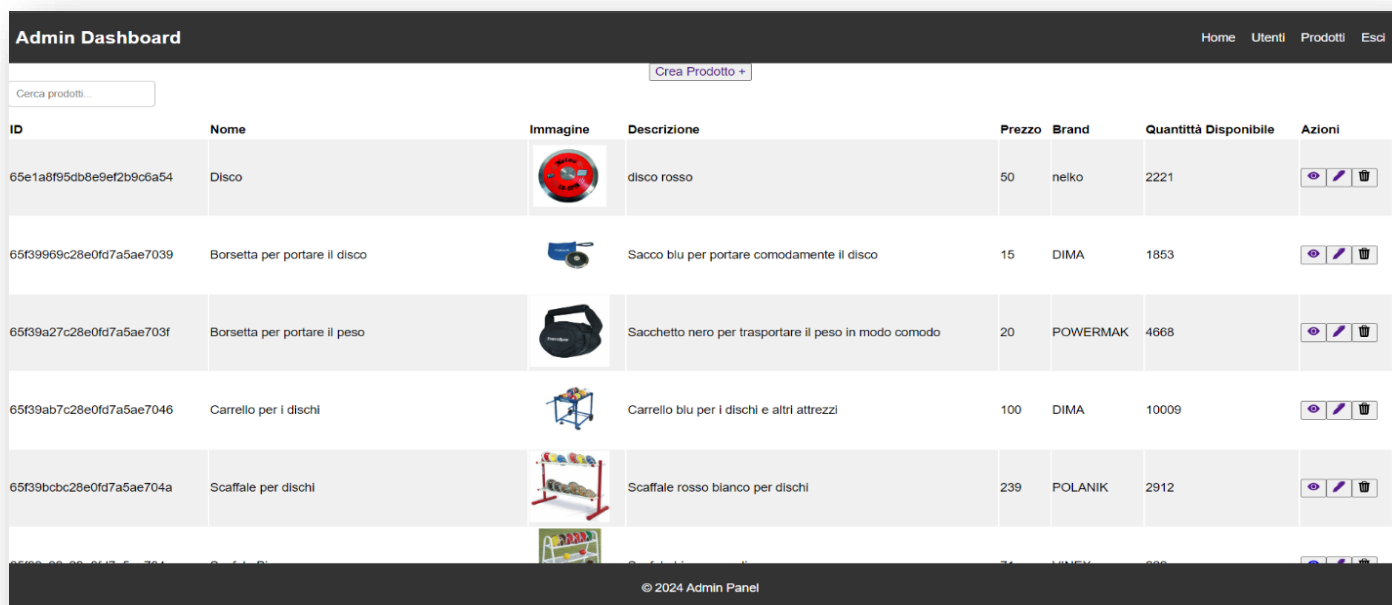
Questa è una funzione per eliminare un utente e quando viene cliccato su elimina esce un messaggio che chiede se effettivamente si vuole eliminare l'utente.



3.2.1.2 PRODOTTI

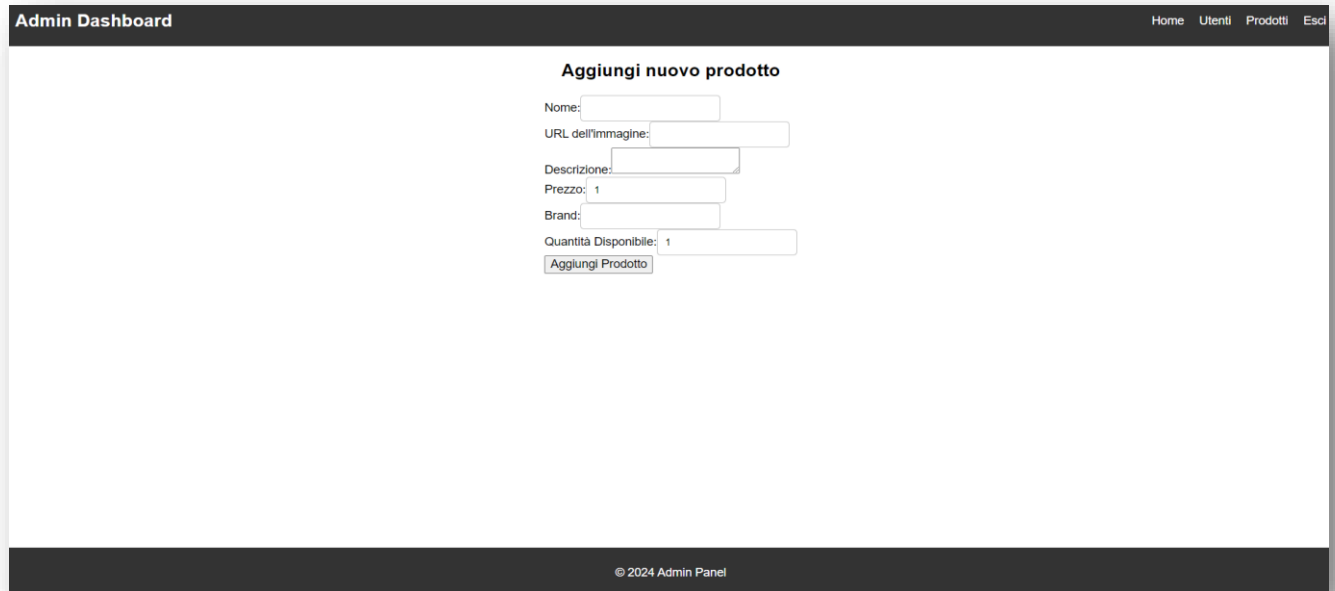
3.2.1.2.1 GetAllProducts

Come per gli utenti, quando clicchiamo su prodotti ci viene mostrata una pagina di tutti i prodotti presenti e possiamo svolgere operazioni come l'aggiunta di un prodotto, visualizzare modificare oppure eliminare un prodotto. Anche in questo caso abbiamo la barra per la ricerca per nome dei prodotti.



3.2.1.2.2 AddProduct

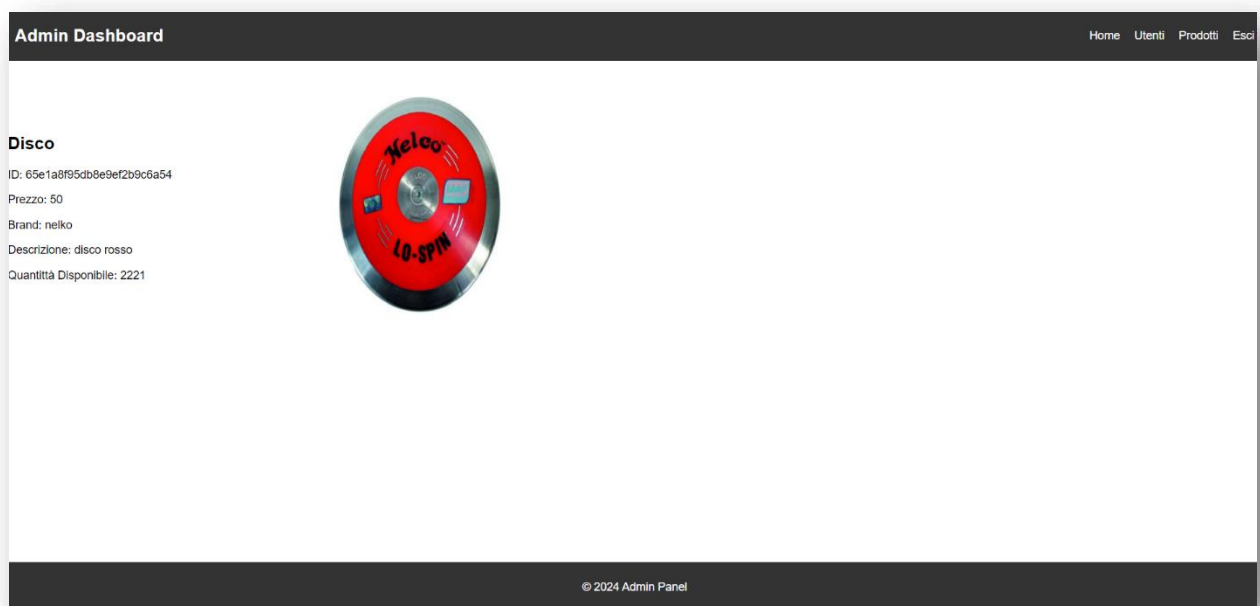
La funzione che ci permette di aggiungere un prodotto. Abbiamo una gestione di errori appropriata, il controllo sulla compilazione di tutti i campi, il controllo sul nome del prodotto, non deve esistere nel database e il controllo anche sull'url dell'immagine



The screenshot shows the 'Admin Dashboard' interface with a dark header containing 'Home', 'Utenti', 'Prodotti', and 'Esci'. The main content area is titled 'Aggiungi nuovo prodotto' and contains a form with the following fields: 'Nome:', 'URL dell'immagine:', 'Descrizione:', 'Prezzo: 1', 'Brand:', and 'Quantità Disponibile: 1'. A button labeled 'Aggiungi Prodotto' is located at the bottom of the form. The footer of the dashboard displays '© 2024 Admin Panel'.

3.2.1.2.3 GetProduct

Possiamo visualizzare le informazioni di un singolo prodotto.



3.2.1.2.4 UpdateProduct

Questa funzione ci permetti di aggiornare un prodotto con la gestione di eventuali errori e i controlli come all'aggiunta di un nuovo prodotto.

The screenshot shows the 'Aggiorna Prodotto' form in the Admin Dashboard. The form contains the following fields:

- Nome: Disco
- URL dell'immagine: /disco1.jpg
- Descrizione: disco rosso
- Prezzo: 50
- Brand: nelko
- Quantità Disponibile: 2221
- Aggiorna button (green)

The dashboard header includes 'Admin Dashboard' and navigation links: Home, Utenti, Prodotti, Esci. The footer shows '© 2024 Admin Panel'.

3.2.1.2.5 DeleteProduct

Questa è una funzione per eliminare un prodotto e quando viene cliccato su elimina esce un messaggio che chiede se effettivamente si vuole eliminare il prodotto.

The screenshot shows the 'Prodotti' table in the Admin Dashboard. The table has the following columns: ID, Nome, Immagine, Descrizione, Prezzo, Brand, Quantità Disponibile, and Azioni. A confirmation dialog is displayed over the table, asking 'Sei sicuro di voler cancellare questo prodotto?' with 'Si' and 'No' buttons.

ID	Nome	Immagine	Descrizione	Prezzo	Brand	Quantità Disponibile	Azioni
35e1a8f95db8e9ef2b9c6a54	Disco		disco rosso	50	nelko	2221	
35f39969c28e0fd7a5ae7039	Borsetta per portare il disco		Sacco blu per portare comodamente il disco	15	DIMA	1853	
35f39a27c28e0fd7a5ae703f	Borsetta per portare il peso		Sacco nero per portare comodamente il peso in modo comodo	20	POWERMAK	4668	
35f39ab7c28e0fd7a5ae7046	Carrello per i dischi		Carrello blu per i dischi e altri attrezzi	100	DIMA	10009	
35f39bcb28e0fd7a5ae704a	Scaffale per dischi		Scaffale rosso bianco per dischi	239	POLANIK	2912	
35f39c28e0fd7a5ae704b	Scaffale per dischi		Scaffale rosso bianco per dischi	239	POLANIK	2912	

The dashboard header includes 'Admin Dashboard' and navigation links: Home, Utenti, Prodotti, Esci. The footer shows '© 2024 Admin Panel'.

4. CONFIGURAZIONE

4.1 Database

Per la persistenza dei dati, l'applicazione fa uso di MongoDB, un database non relazionale orientato ai documenti. MongoDB è stato scelto per la sua scalabilità, flessibilità e facilità d'uso, adatto alle esigenze dinamiche dell'applicazione.

Struttura del Database:

- Il database è organizzato in quattro collezioni principali corrispondenti ai moduli dell'applicazione:

1. Collezione “users” per la gestione degli utenti registrati.

```
server > models > JS userModel.js > ...
1  const mongoose = require('mongoose');
2
3  const userSchema = new mongoose.Schema(
4    {
5      name: { type: String, required: true },
6      email: { type: String, required: true },
7      password: { type: String, required: true },
8      resetToken: { type: String },
9      isAdmin: { type: Boolean, default: false, required: true },
10   },
11   {
12     timestamps: true,
13   }
14 );
15
16 const User = mongoose.model('User', userSchema);
17 module.exports = User;
18
19
```

2. Collezione “products” per la gestione dei prodotti disponibili.

```
server > models > JS productModel.js > ...
1  const mongoose = require('mongoose');
2
3  const productSchema = new mongoose.Schema(
4    {
5      name: { type: String, required: true },
6      image: { type: String, required: true },
7      description: { type: String, required: true },
8      price: { type: Number, required: true },
9      brand: { type: String, required: true },
10     countInStock: { type: Number, required: true, integer: true }
11   },
12   {
13     timestamps: true,
14   }
15 );
16
17
18
19
20 const Product = mongoose.model('Product', productSchema);
21 module.exports = Product;
22
23
24
```

3. Collezione “carts” per la memorizzazione dei carrelli degli utenti.

```

server > models > JS cartModel.js > ...
1  const mongoose = require('mongoose');
2
3  // Definizione dello schema per un elemento nel carrello
4  const cartItemSchema = new mongoose.Schema({
5    // Riferimento all'ID del prodotto associato
6    product: { type: mongoose.Schema.Types.ObjectId, ref: 'Product', required: true },
7    name: { type: String, required: true },
8    image: { type: String, required: true },
9    description: { type: String, required: true },
10   brand: { type: String, required: true },
11   // La quantità di questo prodotto nel carrello
12   quantity: { type: Number, required: true, default: 1 },
13   // Prezzo unitario del prodotto
14   price: { type: Number, required: true }, // Aggiunto il prezzo del prodotto
15
16   countInStock: { type: Number, required: true },
17
18
19
20 });
21
22 // Definizione dello schema per il carrello completo
23 const cartSchema = new mongoose.Schema({
24   // Riferimento all'ID dell'utente proprietario del carrello
25   user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
26   // Array di elementi nel carrello
27   items: [cartItemSchema],
28   // Il prezzo totale del carrello
29   totalPrice: { type: Number, required: true, default: 0 },
30 }, { timestamps: true });
31
32 // Creazione del modello Mongoose per il carrello
33 const Cart = mongoose.model('Cart', cartSchema);
34
35 module.exports = Cart;
36
37
38

```


4. Collezione “orders” per la memorizzazione degli ordini effettuati dagli utenti.

```

server > models > JS orderModel.js > ...
1  const mongoose = require('mongoose');
2
3  const orderSchema = new mongoose.Schema({
4    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
5    items: [{
6      product: { type: mongoose.Schema.Types.ObjectId, ref: 'Product', required: true },
7      name: { type: String, required: true },
8      image: { type: String, required: true },
9      description: { type: String, required: true },
10     brand: { type: String, required: true },
11     quantity: { type: Number, required: true },
12     price: { type: Number, required: true }
13   ]},
14   totalPrice: { type: Number, required: true },
15   deliveryAddress: { type: String, required: true },
16   deliveryDate: { type: Date, required: true },
17   status: { type: String, enum: ['In attesa di consegna', 'Consegnato'], default: 'In attesa di consegna' },
18   createdAt: { type: Date, default: Date.now }
19 });
20
21 const Order = mongoose.model('Order', orderSchema);
22 module.exports = Order;
23
24

```

Tecnologie Utilizzate:

- MongoDB Atlas è stato utilizzato come servizio di database cloud per la gestione dei dati.
- Mongoose, una libreria di modellazione dei dati MongoDB in Node.js, è stata impiegata per definire i modelli dei documenti e facilitare le operazioni di accesso al database.

4.2 Server

Il server è stato configurato utilizzando Node.js insieme al framework Express.js per fornire un'interfaccia RESTful per il client e gestire le richieste http.

Moduli del server:

Nel server abbiamo dei moduli per la gestione di diverse operazioni. Ad esempio abbiamo:

1. Modulo “user”: Gestisce le operazioni relative agli utenti, tra cui la registrazione, l'autenticazione e la gestione del profilo ect...
2. Modulo “product”: Gestisce le operazioni relative ai prodotti, compresa la visualizzazione dei prodotti disponibili e la ricerca ect...
3. Modulo “cart”: Gestisce le operazioni legate ai carrelli degli utenti, inclusa l'aggiunta, la rimozione e il salvataggio dei prodotti nel carrello e anche l'operazione per il pagamento. In questo modulo abbiamo anche raggiunto le operazioni degli ordini come visualizzare ed eliminare gli ordini degli utenti.

Tecnologie Utilizzate:

- Express.js è stato utilizzato per la gestione delle richieste HTTP e la definizione delle rotte dell'applicazione.
- Cors (Cross-Origin Resource Sharing) è stato configurato per consentire le richieste da origini multiple al server.

4.3 Client

Il client dell'applicazione è stato sviluppato utilizzando React.js, una libreria JavaScript per la creazione di interfacce utente dinamiche. Axios viene impiegato per effettuare richieste HTTP al server e gestire la comunicazione tra il client e il back-end, garantendo un flusso efficiente di dati. Inoltre, React Router è utilizzato per gestire la navigazione e definire rotte all'interno dell'applicazione client, assicurando una navigazione fluida tra le diverse sezioni dell'app.

L'applicazione è progettata per offrire agli utenti un'esperienza completa di shopping online. Include funzionalità di registrazione e login per gli utenti, nonché la gestione dei profili utente. La piattaforma consente agli utenti di visualizzare i prodotti disponibili, ottenere dettagli su di essi e aggiungerli al carrello per l'acquisto. Inoltre, offre un'interfaccia per la gestione e la visualizzazione degli ordini effettuati dagli utenti.

In conclusione, l'applicazione si concentra su un'esperienza utente fluida e intuitiva per esplorare, selezionare e acquistare prodotti online.