



A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Instrumental

e-commerce strumenti musicali e materiali audio

Simone Seria 344091

Fabbriciani Giulia Diletta 347101

Obiettivi del progetto

Il progetto mira a creare un servizio **e-commerce** dedicato alla vendita di **strumenti musicali, accessori e attrezzature audio**.

I **fruitori** dovranno poter navigare tra i prodotti del catalogo, effettuare ricerche mirate, visualizzare il carrello e infine completare l'acquisto.

Gli **erogatori** avranno la possibilità di accedere ad un pannello di controllo apposito per la gestione e la visualizzazione degli utenti, dei prodotti e degli ordini.



Beyerdynamic DT 770 Pro

119.99 €

Add to cart



Hofner HCT 500/1 BK Violin Bass

679.99 €

Add to cart



Donner DDP-95 Pianoforte digitale

499.99 €

Add to cart



Soundcore by Anker Q20i Cuffie Bluetooth

45.99 €

Add to cart



Ibanez GSRM20B-WNF basso elettrico

209.00 €

Add to cart



Yamaha C40II Chitarra Classica

122.00 €

Add to cart



TIGER JDS14-RD - Batteria

296.99 €

Add to cart



VOX Amplug 2 Ap2-BS, Bass

38.00 €

Add to cart

Architettura del sistema

```
form onSubmit={handleSubmit(onSubmit)}>
  <div className={styles.divModal}>
    <label htmlFor="name">Product Name</label>
    <input
      id="name"
      {...register("name", { required: "Product name is required" })}
    />
    {errors.name && <p className={styles.error}>{errors.name.message}</p>}
  </div>
```

```
<Route path="/" element={<HomePage />} />
<Route path="/services" element={<Services />} />
<Route path="/about" element={<About />} />
<Route path="/cart" element={<Cart />} />
<Route path="/register" element={<Register />} />
<Route path="/login" element={<Login />} />
<Route path="/personalarea" element={<PersonalArea />} />
<Route path="/admin" element={<AdminDashboard />} />
<Route path="/admin/products" element={<AdminProducts />} />
<Route path="/admin/customers" element={<AdminCostumers />} />
```

```
import styles from "../AdminCostumers.module.css";
```

```
app.use(cors());
app.use(express.json());
const port = process.env.PORT;
const connectionString = process.env.DATABASE_URL;
const sql = postgres(connectionString);

const multer = require('multer'); // Multer per management delle immagini
const fs = require('fs');
const path = require('path');
```

Frontend:

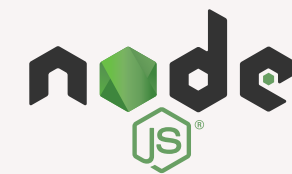
Backend:

Database:

sviluppato con React, utilizzando
librerie di supporto:



per lo stile:



API RESTful

Esempio:

```
export async function getProducts(product='') {
  const url = `${BASE_URL}/products/${product}`;
  const options = {
    method: 'GET',
    headers: { 'Content-Type': 'application/json' }
  };
  return fetch(url, options).then(res => res.json());
}
```

Questa funzione effettua una richiesta HTTP GET a un endpoint (definito come BASE_URL/products/\${product}) per ottenere informazioni sui prodotti.

```
const ProductsData = await getProducts(); //ottiene i prodotti
setProducts(ProductsData); //salva i prodotti nello stato setProducts

} catch (error) {
  console.error("Error while trying to pull data", error);
} finally {
  setLoading(false);
}
```

```
app.get('/api/products', async (req, res) => {
  try {
    const result = await sql`
      SELECT id, name, price, created_at, image_src
      FROM products`;

    if(result.length === 0){
      res.status(404).json({error: 'No products found'});
    }
  } catch (error) {
    console.error(error);
  }
});
```

risponde alle richieste **GET** inviate all'endpoint **/api/products** per ottenere una lista di tutti i prodotti presenti nel database.

Le **API RESTful** (representational state transfer) permettono una comunicazione efficace tra **frontend e backend**, garantendo modularità e riusabilità. Sono progettate per gestire in modo efficiente la richiesta e lo **scambio di dati**, indispensabili per applicazioni come l'e-commerce

Nel nostro e-commerce, le API RESTful consentono agli **utenti** di interagire facilmente con il sistema per cercare prodotti, gestire il carrello e completare acquisti, mentre gli **amministratori** possono gestire il catalogo prodotti, monitorare gli ordini e aggiornare le informazioni.

Esempio:

l'applicazione fa una richiesta GET a /api/products tramite una funzione asincrona (getProducts), **questa funzione richiede i prodotti dal server e li passa al frontend**, dove possono essere mostrati all'utente o utilizzati per altre operazioni.

L'interazione tra questo endpoint e il frontend rende possibile visualizzare l'intero catalogo dei prodotti e aggiornare dinamicamente l'interfaccia ogni volta che nuovi prodotti vengono aggiunti o aggiornati nel database.

Metodi HTTP utilizzati:

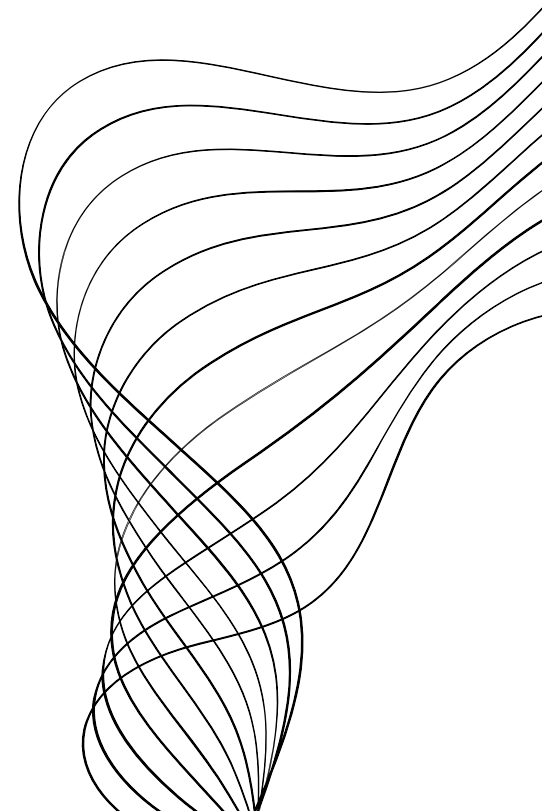
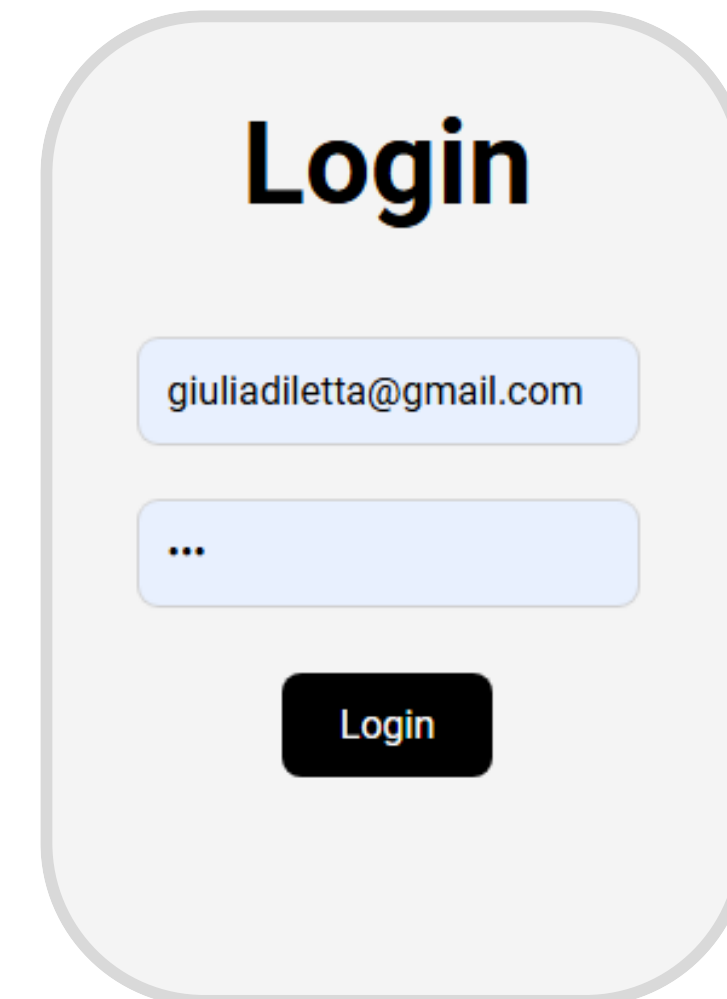
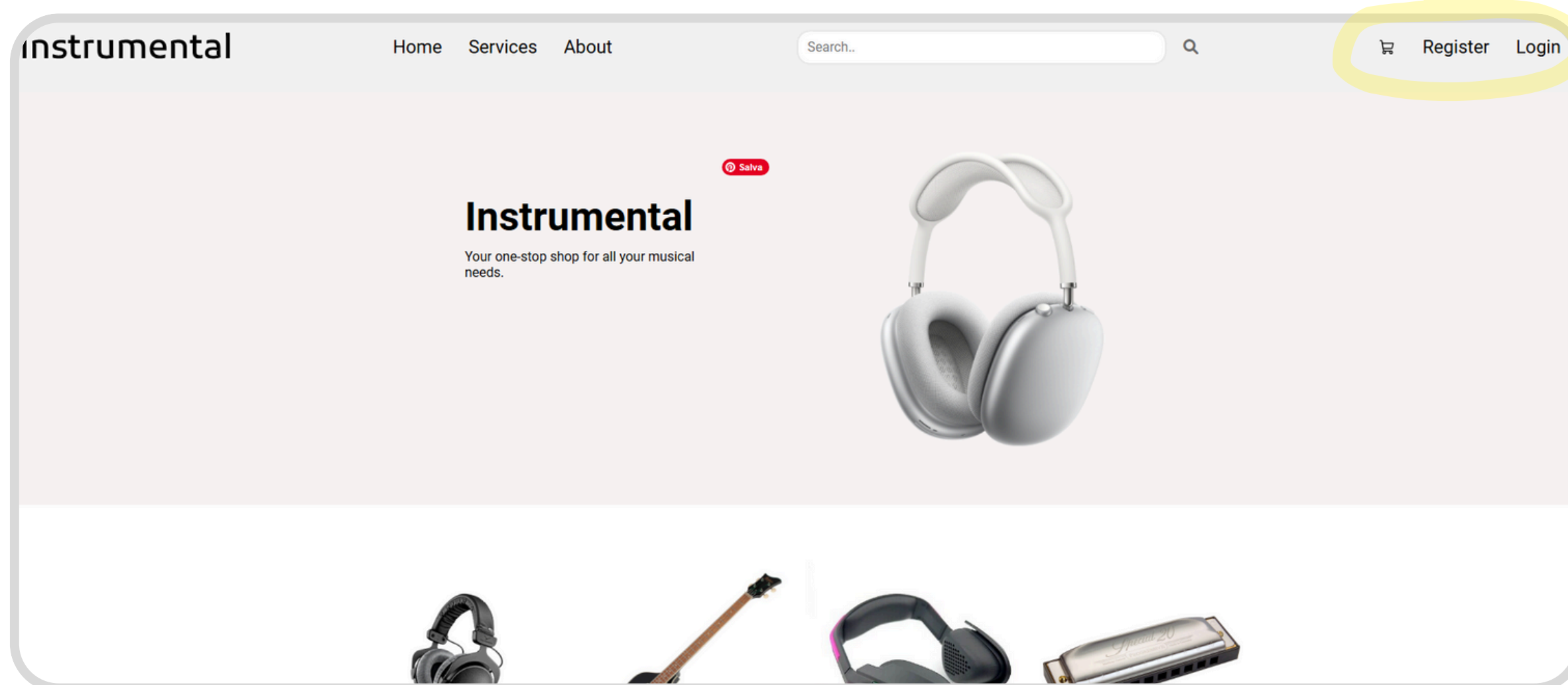
- GET
- POST
- PUT
- DELETE

I dati vengono scambiati in formato **Json** --> permette di trasmettere solo i dati necessari in modo conciso. Questo rende il trasferimento dei dati tra il server e il client rapido ed efficiente, migliorando la velocità complessiva delle applicazioni.

```
{  
  "name": "Chitarra Acustica",  
  "price": 199.99,  
  "image": "chitarra.jpg"  
}
```

Funzionalità

Gli utenti dopo aver effettuato l'accesso (se già registrati) vengono indirizzati alla **homepage**



Instrumental

Your one-stop shop for all your musical needs.

[Salva](#)

Shopping cart



Hofner HCT 500/1 BK Violin
Bass

679.99 €

[Remove](#)

Beyerdynamic DT 770 Pro

119.99 €

[Remove](#)

Total:

Gli utenti possono esplorare le diverse sezioni della piattaforma, come "About" e "Services", per informarsi sulla piattaforma e sui servizi offerti. Possono utilizzare la barra di ricerca per trovare rapidamente i prodotti di loro interesse e aggiungerli al carrello, sempre accessibile per visualizzare i prodotti scelti. Dal carrello, l'utente può procedere al checkout e completare l'acquisto.

L’amministratore invece avrà accesso ad una **dashboard** che funge da centro di controllo per monitorare l’attività del sito:

🏠 Home

📊 Dashboard

🛒 Products

👤 Customers

Dashboard

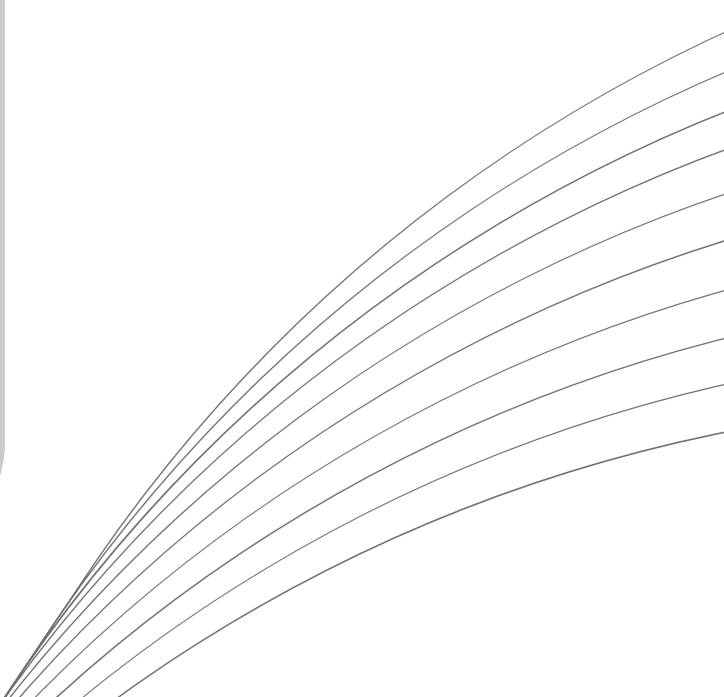
8819.77€
📄 Revenue

12
🛒 Orders

5
👤 Customers

Order Id	User Id	Created At	Product	Total Price
1	1	2024-07-16T23:32:47.081Z	casio pianoforte, Pianoforte Casio	499.99
3	1	2024-07-16T23:33:24.400Z	casio pianoforte, Pianoforte Casio	499.99
4	1	2024-07-16T23:57:13.295Z	, Casio PX-S1000 Privia Black, Beyerdynamic DT 770 Pro, Hofner HCT 500/1 BK Violin Bass	1299.97
8	1	2024-07-17T16:36:56.283Z	Casio PX-S1000 Privia Black, Casio PX-S1000 Privia Black, Casio PX-S1000 Privia Black, Casio PX-S1000 Privia Black, Casio PX-S1000 Privia Black	2499.95
9	1	2024-07-17T16:44:38.457Z	Casio PX-S1000 Privia Black, Beyerdynamic DT 770 Pro, Hofner HCT 500/1 BK Violin Bass	1299.97

pagina /admin



Customers

pagina /customers

First Name	Last Name	Email	Id
Simone	Seria	simoneseria@live.com	1
ff	ss	super.utente@gmail.com	160
Fabio	Seria	pop	161
ff	ff	opo	162
Giulia Diletta	Fabbriciani	giuliadiletta@gmail.com	163



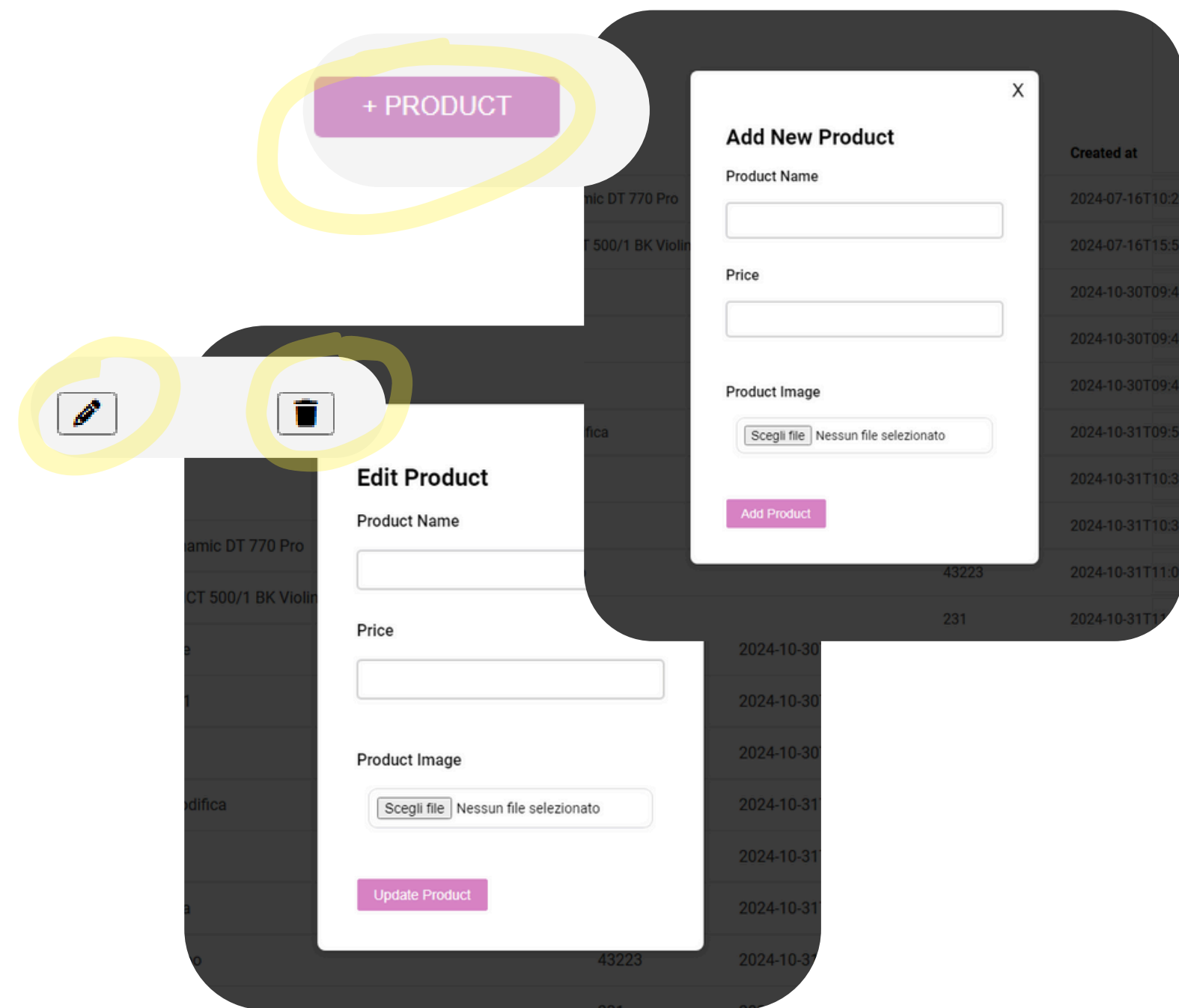
Products

pagina /products



Products Id	Name	Price	Created at		
2	Beyerdynamic DT 770 Pro	119.99	2024-07-16T10:24:40.635Z		
3	Hofner HCT 500/1 BK Violin Bass	679.99	2024-07-16T15:51:34.320Z		
20	dva cuffie	300	2024-10-30T09:40:03.623Z		
21	prodotto1	2313	2024-10-30T09:40:17.194Z		
22	cuffie	231	2024-10-30T09:43:39.398Z		
23	cuffie modifica	30	2024-10-31T09:56:20.719Z		





```
/* Modale con form per aggiungere prodotto */
<Modal show={showModal} onClose={handleCloseModal}>
  <h2 className={styles.h2Modal}>Add New Product</h2>

  <form onSubmit={handleSubmit(onSubmit)}>
    <div className={styles.divModal}>
      <label htmlFor="name">Product Name</label>
      <input
        id="name"
        {...register("name", { required: "Product name is required" })}>
      />
      {errors.name && <p className={styles.error}>{errors.name.message}</p>}
    </div>

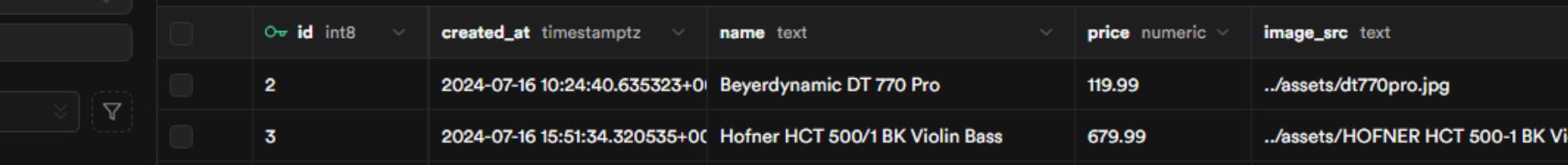
    <div className={styles.divModal}>
      <label htmlFor="price">Price</label>
      <input
        id="price"
        type="number"
        step="0.01"
        {...register("price", { required: "Price is required", min: 0 })}>
      />
      {errors.price && <p className={styles.error}>{errors.price.message}</p>}
    </div>

    <div className={styles.divUpload}>
      <label htmlFor="image">Product Image</label>
      <input
        id="image"
```

```
const handleDelete = async (productId) => {
  try {
    // funzione per eliminare il prodotto
    await deleteProduct(productId);
    // tramite filter creiamo un nuovo array che contiene tutti i prodotti tranne quello che vogliamo eliminare
    setProducts(products.filter(product => product.id !== productId));
  } catch (error) {
    console.error("Error deleting product:", error);
  }
};
```



Il database è stato progettato per supportare le funzionalità principali dell'e-commerce, come la **gestione di utenti, prodotti, ordini e immagini**.



The screenshot shows a PostgreSQL database interface. On the left, there's a sidebar with a search bar and a list of tables: 'orders', 'products', and 'users'. The 'products' table is selected. The main area displays the 'products' table with the following columns: 'id' (integer), 'created_at' (timestamp), 'name' (text), 'price' (numeric), and 'image_src' (text). The table contains 10 rows of data.

id	created_at	name	price	image_src
2	2024-07-16 10:24:40.635323+00	Beyerdynamic DT 770 Pro	119.99	../assets/dt770pro.jpg
3	2024-07-16 15:51:34.320535+00	Hofner HCT 500/1 BK Violin Bass	679.99	../assets/HOFNER HCT 500-1 BK Violin Ba
20	2024-10-30 09:40:03.623986+00	dva cuffie	300	../assets/s-l400.jpg
21	2024-10-30 09:40:17.194211+00	prodotto1	2313	../assets/Hohner-Special-20-Armoniche-M
22	2024-10-30 09:43:39.398516+00	cuffie	231	../assets/s-l400.jpg
23	2024-10-31 09:56:20.719462+00	cuffie modifica	30	../assets/be407fad48c9ccafcd87007eb66
29	2024-10-31 10:36:01.673014+00	rete	654	../assets/s-l400.jpg
30	2024-10-31 10:36:23.385031+00	prova dva	3214	../assets/s-l400.jpg
31	2024-10-31 11:09:42.191904+00	capitanooo	43223	../assets/imagemagic.jfif

(Le tabelle sono collegate tra loro per mantenere l'integrità dei dati e garantire l'efficienza delle query)

```
const cors = require('cors');
const postgres = require('postgres');
const express = require('express');
const app = express();
app.use(cors());
app.use(express.json());
const port = process.env.PORT;
const connectionString = process.env.DATABASE_URL;
const sql = postgres(connectionString);
```

impostiamo la libreria postgres e configuriamo la connessione tramite un URL di connessione memorizzato nelle variabili d'ambiente.