Relazione Project Work Ardini Andrea

Progetto realizzato: Applicazione Web per la Gestione di Clienti e Ordini

Il progetto richiede di creare un'applicazione web semplice e moderna per la gestione dei dati di due entità correlate tra loro attraverso una relazione uno a molti. Sono state fornite delle coppie di entità tra cui scegliere e Il progetto realizzato consiste in una Single Page Application (SPA) che simuli in parte un CRM per la gestione dei clienti e dei relativi ordini.

L'obiettivo principale dell'applicazione è fornire uno strumento intuitivo che permetta di visualizzare rapidamente le informazioni sui clienti e di consultare lo storico degli ordini associati a ciascuno di essi.

Le funzionalità richieste sono state identificate come segue:

- Visualizzazione centralizzata dei clienti: L'utente deve poter accedere a una pagina principale che mostri una lista completa di tutti i clienti registrati. Per ogni cliente, dovranno essere immediatamente visibili le informazioni essenziali (nome e immagine) per una rapida identificazione.

- Accesso al Dettaglio del Cliente: Dalla lista principale, l'utente deve poter selezionare un singolo cliente per accedere a una pagina di dettaglio dedicata.

- Consultazione dello Storico Ordini: La pagina di dettaglio del cliente non deve solo mostrare i dati anagrafici (come indirizzo ed email), ma anche elencare in modo chiaro e ordinato tutti gli ordini che quel cliente ha effettuato, con le relative informazioni (descrizione, data, totale).

L'applicazione è composta da due parti principali:

Backend API RESTful: Realizzato con il framework Laravel, si occupa della logica di business, della gestione dei dati e dell'esposizione delle API necessarie al frontend.

Frontend: Realizzato con il framework Angular, fornisce l'interfaccia utente per visualizzare la lista dei clienti, navigare nel dettaglio di un singolo cliente e visualizzare la lista degli ordini a lui associati.

La comunicazione tra frontend e backend avviene tramite chiamate HTTP a endpoint JSON.

Struttura del Layout e Design dell'Interfaccia:

Per soddisfare le esigenze identificate, è stato progettato un layout pulito e minimale, strutturato su due viste principali.

Pagina 1: Lista Clienti

Questa è la schermata di atterraggio dell'applicazione.

Layout: La pagina presenta un titolo centrale e, subito sotto, una lista verticale di "card", una per ogni cliente.

Card Cliente: Ogni card è pensata per essere visivamente gradevole e informativa. Contiene:

Un'immagine del cliente (avatar) sulla sinistra per un riconoscimento immediato.

Le informazioni principali sulla destra: il nome in evidenza e, sotto, i contatti come email e indirizzo.

Interazione: L'intera card è cliccabile. Un clic porta l'utente alla pagina di dettaglio del cliente selezionato, garantendo un'interazione semplice e diretta.

Pagina 2: Dettaglio Cliente

Questa pagina è dedicata alla visione completa delle informazioni di un singolo cliente.

Layout: La pagina è divisa in due sezioni principali.

Sezione Anagrafica: In alto, una card prominente mostra i dati principali del cliente (immagine, nome, email, indirizzo), in modo simile alla lista ma con più spazio e risalto.

Sezione Storico Ordini: Sotto i dati anagrafici, una seconda sezione è dedicata agli ordini. Gli ordini sono presentati come una lista, dove ogni elemento mostra chiaramente il numero d'ordine, la descrizione, la data e l'importo totale.

Navigazione: Un link o un pulsante "Indietro" permette all'utente di tornare facilmente alla lista principale.

Sviluppo:

Lo sviluppo è stato condotto utilizzando un ambiente di sviluppo containerizzato basato su Docker e Docker Compose per garantire coerenza, isolamento e facilità di setup.

L'ambiente è orchestrato da un file docker-compose.yml che definisce quattro servizi principali:
1. backend: Container con PHP 8.2-FPM per l'esecuzione di Laravel.
2. frontend: Container con Node.js 18 e Angular CLI per lo sviluppo e la compilazione del frontend.
3. webserver: Container Nginx che agisce da web server per il backend Laravel.
4. db: Container MySQL 8.0 per il database relazionale.

Fasi di Sviluppo:
1. Setup dell'Ambiente Docker: Creazione dei Dockerfile e del file docker-compose.yml per automatizzare l'ambiente.

2. Sviluppo Backend (Laravel):
   - Installazione e configurazione di un nuovo progetto Laravel.
   - Definizione dei modelli Customer e Order e delle relative migrazioni per creare la struttura del database.
   - Implementazione delle relazioni "uno-a-molti" tra Customer e Order.
   - Creazione di Seeder per popolare il database con dati di prova.
   - Sviluppo di un CustomerController per gestire le richieste API.
   - Definizione delle rotte API tramite Route::apiResource per esporre gli endpoint RESTful.

3. Sviluppo Frontend (Angular):
   - Creazione di un nuovo progetto Angular basato su Moduli.
   - Implementazione di un CustomerService per incapsulare la logica di comunicazione con il backend tramite HttpClient.
   - Sviluppo dei componenti: CustomerListComponent per la visualizzazione della lista, CustomerDetailComponent per il dettaglio e OrderListComponent per la visualizzazione della lista degli ordini.
   - Configurazione del AppRoutingModule per gestire la navigazione tra le viste.

4. Version Control: L'intero progetto è stato tracciato con Git e ospitato su un repository GitHub, https://github.com/Aranweb/pw2-andrea-ardini.

Istruzioni per l'Esecuzione

Sono fornite due modalità per avviare l'applicazione.

Metodo A: Esecuzione con Docker (Consigliato)

Questo metodo è il più semplice e affidabile, poiché ricrea l'ambiente esatto in cui l'applicazione è stata sviluppata.

Prerequisiti: Docker e Docker Compose installati.

1. Decomprimere il file `.zip` della versione Docker.
2. Aprire un terminale nella cartella radice del progetto.
3. Avviare i container in background:

   ```
   docker-compose up -d --build
   ```

4. Installare le dipendenze del backend:

   ```
   docker-compose exec backend composer install
   ```

5. Generare la chiave applicativa di Laravel:

   ```
   docker-compose exec backend php artisan key:generate
   ```

6. Creare le tabelle del database e popolarle:

    docker-compose exec backend php artisan migrate --seed

7. Installare le dipendenze del frontend:

    docker-compose exec frontend npm install

8. L'applicazione è pronta. Accedere ai seguenti indirizzi dal browser:
    Frontend Angular: http://localhost:4200
    Backend API Laravel: http://localhost:8080

Metodo B: Esecuzione Manuale (Senza Docker)

Questo metodo richiede la configurazione manuale di un ambiente di sviluppo locale.

Prerequisiti:
- Un server web locale con PHP e MySQL (es. XAMPP, WAMP, MAMP).
- Composer installato globalmente.
- Node.js e npm installati globalmente.

1. Setup del Backend:
    - Decomprimere il file .zip della versione "No Docker".
    - Creare un nuovo database vuoto nel proprio server MySQL (es. pw2_db).
    - Importare nel database appena creato il file dump.sql.
    - Navigare nella cartella backend/.
    - Copiare il file .env.example in un nuovo file chiamato .env.
    - Modificare il file .env con le credenziali del proprio database locale.
    - Installare le dipendenze:

        composer install

    - Generare la chiave applicativa:

        php artisan key:generate

    - Avviare il server di sviluppo di Laravel:

        php artisan serve

    - Backend API Laravel disponibile su http://localhost:8000.

2. Setup del Frontend:
    - Aprire un nuovo terminale e navigare nella cartella frontend/.
    - Installare le dipendenze:

        npm install

- Avviare il server di sviluppo di Angular:

  ng serve

- Frontend Angular disponibile su http://localhost:4200

Nota: Potrebbe essere necessario modificare l'URL dell'API nel file frontend/src/app/customer.service.ts per puntare all'indirizzo del server Laravel (es. http://localhost:8000/api).