



SAPIENZA
UNIVERSITÀ DI ROMA

MyShoppingList: Progettazione e sviluppo di una piattaforma per la gestione ottimizzata e personalizzata della spesa quotidiana

Facoltà di Ingegneria dell'informazione, informatica e statistica
Corso di Laurea in Ingegneria Informatica e Automatica

Marco Pedicillo
Matricola 1983285

Relatore
Leonardo Querzoni

Anno Accademico 2023/2024

Indice

1	Introduzione	1
1.1	Front-end	1
1.2	Back-end	2
1.3	Utenti	3
1.3.1	Utente non registrato	3
1.3.2	Utente registrato	3
1.3.3	Admin	3
2	Analisi concettuale del sistema	4
2.1	Schema Entità-Relazione 1	4
2.2	Schema Entità-Relazione 2	6
2.3	Le business rules	7
3	Tecnologie e metodologie utilizzate	8
3.1	HTML e CSS	8
3.2	JavaScript	8
3.3	PHP	9
3.4	Autenticazione con servizi esterni	9
3.5	API	10
4	Home Page	11
4.1	Navbar	12
4.2	Recensioni	13
4.2.1	Lato server	13
4.3	Footer	15
4.4	Responsiveness	16
5	User Page	17
5.1	Registrazione	17
5.1.1	Lato server	18
5.2	Login	20
5.2.1	Lato server	21
5.3	Aggiornamento profilo	22
5.4	Login di terze parti	23

6 Pagina dei prodotti	25
6.1 Localizzazione	26
6.2 Barra di ricerca	26
6.2.1 Lato server	26
6.3 Filtri delle categorie	27
6.3.1 Lato server	28
6.4 Controllo della salubrità	29
6.4.1 Lato server	29
6.5 Ricette suggerite	30
6.5.1 Lato server	31
6.6 Carrello	32
6.7 Responsiveness	33
7 Checkout	34
7.1 Utente non registrato	34
7.2 Utente registrato	35
7.3 Lato server	37
7.3.1 Supermercato più vicino	37
7.3.2 Supermercato più economico	38
7.3.3 Supermercato consigliato	39
8 Pagina preferiti	40
8.1 Lato server	41
9 Admin	42
9.1 Supermercati	43
9.1.1 Lato server	44
9.2 Prodotti	45
9.2.1 Lato server	46
9.3 Utenti	47
9.3.1 Lato server	47
9.4 Recensioni	48
9.5 Footer	48
10 Test	49
10.1 Tecnologie usate	49
10.2 Test unitari	50
10.3 Test funzionali	51
10.4 Test di accettazione	52
11 Conclusioni	53
Ringraziamenti	54

Elenco delle figure

1.1	Tabelle Database	2
2.1	Schema Entità-Relazione 1	4
2.2	Schema Entità-Relazione 2	6
4.1	Home Page	11
4.2	Popup Login	12
4.3	Recensioni	13
4.4	Footer	15
4.5	Dimensioni Mobile	16
4.6	Dimensioni Tablet	16
5.1	Pagina di Registrazione	17
5.2	Navbar per utente autenticato	18
5.3	Pagina Utente	20
6.1	Pagina Prodotti	25
6.2	Barra di Ricerca	26
6.3	Filtri Categoria	27
6.4	Prodotto non Healthy	29
6.5	Pop-up Avviso Salubrità	29
6.6	Ricette	30
6.7	Carrello	32
6.8	Pop-up Avviso Budget	32
6.9	Versione Mobile	33
6.10	Versione Tablet	33
7.1	Checkout utente non registrato	34
7.2	Checkout utente registrato	35
7.3	PDF Checkout	36
8.1	Pagina carrelli preferiti	40
9.1	Home Page Admin	42
9.2	Pagina supermercati Admin	43
9.3	Pagina prodotti Admin	45
9.4	Pagina utenti Admin	47
9.5	Footer Admin	48

Elenco dei CodeSnap

3.1	Script creazione di una mappa	10
4.1	Aggiunta Recensione	14
4.2	Eliminazione Recensione	14
4.3	Script Back to Top	15
5.1	Nuova Registrazione	19
5.2	Autenticazione	21
5.3	Aggiornamento profilo	22
5.4	Login GitHub	24
6.1	Script Barra di Ricerca	27
6.2	Verifica Filtri Categoria	28
6.3	Check Health	30
6.4	Esempio Ricette	31
7.1	Individuazione supermercato più vicino	37
7.2	Individuazione supermercato più economico	38
7.3	Individuazione supermercato consigliato	39
8.1	GET e DELETE carrelli preferiti	41
9.1	Aggiunta e Eliminazione supermercato	44
9.2	Aggiunta e Eliminazione prodotto	46
9.3	Eliminazione utente	47
9.4	Script modifica contatto telefonico	48
10.1	Test unitari	50
10.2	Test funzionali	51
10.3	Test di accettazione	52

Elenco delle tabelle

2.1	Entità 1	5
2.2	Relazioni 1	5
2.3	Entità 2	6
2.4	Relazioni 2	7
2.5	Vincoli d'integrità	7

Capitolo 1

Introduzione



Il progetto MyShoppingList, realizzato insieme ai colleghi Mattia Raffaele Ricciardelli, Federico Corsale e Giulia Battioni, è stato ideato per rendere la spesa più semplice, conveniente e pratica. Il sito è pensato per chi desidera trovare il supermercato ideale, che combini prezzi convenienti e vicinanza, senza complicazioni.

Questa piattaforma intuitiva permette agli utenti di creare liste della spesa in pochi passaggi, sfruttando un'interfaccia semplice e immediata. Grazie a un algoritmo avanzato, MyShoppingList consiglia i supermercati migliori in base ai criteri di risparmio e praticità. Per ciascun articolo, il sito fornisce dettagli chiari e completi su prezzi e informazioni specifiche.

Inoltre, gli utenti registrati possono godere di una grande personalizzazione in base ai propri bisogni specifici. MyShoppingList vuole quindi essere non solo una guida per risparmiare, ma anche uno strumento per rendere più organizzata e pratica l'esperienza di acquisto quotidiana.

Per consultare la documentazione del progetto si scannerizzi il QR Code.

1.1 Front-end

MyShoppingList è stato ideato per offrire un'esperienza di utilizzo intuitiva e accessibile, così che ogni utente possa navigare e interagire senza difficoltà. Il front-end è stato progettato con grande attenzione alla facilità di navigazione, ponendo al centro un'interfaccia grafica semplice, ordinata e funzionale.

Per rendere l'interazione fluida e piacevole, sono stati implementati elementi interattivi come moduli di ricerca intuitivi per individuare rapidamente i prodotti e filtri dinamici per personalizzare le ricerche in modo mirato. Inoltre, la piattaforma è completamente responsive, adattandosi automaticamente a qualsiasi dispositivo

– desktop, smartphone o tablet – per garantire un’esperienza d’uso costante e ottimizzata. Questo è stato possibile grazie all’uso di CSS e delle *media query*, che permettono di adattare lo stile grafico in base al tipo di dispositivo.

Tutte queste scelte mirano a offrire un’esperienza digitale accessibile, che possa soddisfare le esigenze di un’utenza diversificata, indipendentemente dalle competenze tecniche o dalla familiarità con le piattaforme web. L’obiettivo è creare un ambiente inclusivo e facile da utilizzare, senza ostacoli.

1.2 Back-end

Il back-end di MyShoppingList è sviluppato interamente in PHP, scelto per la sua versatilità e per la gestione efficiente delle operazioni di database. Per facilitare il processo di sviluppo e testing, è stato utilizzato XAMPP, una suite di sviluppo che include Apache, MySQL e PHP, consentendo di replicare rapidamente un ambiente di server locale simile a quello di produzione.

Apache, uno dei server web più comuni, gestisce le richieste HTTP dei browser e distribuisce i contenuti agli utenti, simulando in XAMPP un contesto di web server reale, utile per testare l’applicazione in condizioni realistiche. *MySQL* memorizza e recupera i dati cruciali per l’applicazione, come informazioni su utenti, prodotti e liste della spesa. In questo database, i dati sono organizzati in tabelle strutturate in righe e colonne, dove ogni tabella rappresenta un’entità specifica e ogni colonna descrive un attributo dell’entità.

Tabella	Azione	Righe	Tipo	Codifica caratteri	Dimensione	Overhead
cart	★ Mostra Struttura Cerca Inserisci Svuota Elimina	0	InnoDB	utf8mb4_general_ci	16.0 Kib	-
categories	★ Mostra Struttura Cerca Inserisci Svuota Elimina	11	InnoDB	utf8mb4_general_ci	16.0 Kib	-
preferences	★ Mostra Struttura Cerca Inserisci Svuota Elimina	0	InnoDB	utf8mb4_general_ci	16.0 Kib	-
products	★ Mostra Struttura Cerca Inserisci Svuota Elimina	199	InnoDB	utf8mb4_general_ci	128.0 Kib	-
products_prices	★ Mostra Struttura Cerca Inserisci Svuota Elimina	199	InnoDB	utf8mb4_general_ci	16.0 Kib	-
reviews	★ Mostra Struttura Cerca Inserisci Svuota Elimina	0	InnoDB	utf8mb4_general_ci	16.0 Kib	-
session_location	★ Mostra Struttura Cerca Inserisci Svuota Elimina	2	InnoDB	utf8mb4_general_ci	16.0 Kib	-
supermarkets	★ Mostra Struttura Cerca Inserisci Svuota Elimina	71	InnoDB	utf8mb4_general_ci	16.0 Kib	-
supermarkets_products	★ Mostra Struttura Cerca Inserisci Svuota Elimina	14,129	InnoDB	utf8mb4_general_ci	1.5 MiB	-
utenti	★ Mostra Struttura Cerca Inserisci Svuota Elimina	0	InnoDB	utf8mb4_general_ci	32.0 Kib	-

Figura 1.1 Tabelle Database

PHP, linguaggio di scripting lato server di cui parleremo più avanti, gestisce la logica dell’applicazione, collaborando con Apache per rispondere alle richieste degli utenti e con MySQL per il recupero e l’aggiornamento dei dati. Grazie a PHP, le pagine web sono dinamiche, e rispondono in tempo reale alle azioni degli utenti.

L’architettura di MyShoppingList segue il modello *client-server*: il browser (client) invia richieste al server, che risponde utilizzando PHP e interagisce con MySQL per gestire i dati. Le operazioni di base, come aggiungere, leggere, modificare o cancellare informazioni, vengono eseguite tramite query SQL. Questo sistema garantisce che tutte le azioni siano gestite in modo sicuro e affidabile sul server, offrendo un’interazione rapida e coerente per l’utente.

1.3 Utenti

Sono presenti tre tipologie di Utenti all'interno della piattaforma: utente non registrato, utente registrato (*sportivo* o *normale*), e utente Admin.

1.3.1 Utente non registrato

Gli utenti non registrati hanno accesso a tutte le funzionalità principali del sito, come la possibilità di esplorare i prodotti, aggiungerli al carrello e visualizzare il supermercato consigliato.

1.3.2 Utente registrato

L'accesso al sito porta con sé una serie di benefici che arricchiscono l'esperienza utente, oltre a tutte le funzionalità di base. Una volta registrati, gli utenti possono arricchire il proprio profilo indicando preferenze legate al loro stile di vita, come ad esempio se hanno un'attività fisica intensa o una routine più sedentaria. Questo consente al sistema di proporre suggerimenti di acquisto mirati, in linea con i bisogni e le abitudini personali.

In aggiunta, è possibile impostare un limite di spesa. Ciò consente agli utenti di pianificare i propri acquisti in modo più consapevole. Il sito tiene traccia del carrello e avvisa automaticamente se si supera il budget prefissato.

Un altro vantaggio della registrazione è la possibilità di ottenere una panoramica completa dei supermercati: non solo il più conveniente, ma anche quello più vicino e quello più economico. Questo rende più rapide e informate le decisioni d'acquisto. Inoltre, gli utenti registrati possono scaricare facilmente la lista della spesa in formato PDF, semplificando ulteriormente la gestione delle proprie necessità.

Gli utenti registrati possono anche scrivere, modificare ed eliminare la propria recensione riguardante la piattaforma all'interno della home page e salvare le proprie liste della spesa preferite per un rapido riutilizzo, rendendo più agevoli le spese abituali o ripetitive.

1.3.3 Admin

L'utente con ruolo di amministratore ha accesso a funzionalità avanzate, ma non può utilizzare le opzioni base della piattaforma. In qualità di amministratore, può gestire i supermercati, con la possibilità di aggiungere, modificare o rimuovere quelli presenti nel sistema. Inoltre, ha il controllo completo sui prodotti, potendo modificarli, aggiungerli o eliminarli a seconda delle necessità.

L'amministratore può anche visualizzare e rimuovere gli utenti registrati, gestendo così l'elenco degli iscritti. In aggiunta, ha la facoltà di cancellare le recensioni lasciate dagli utenti, assicurando che il contenuto rimanga sempre appropriato. Infine, può modificare i link che riportano alle pagine del sito presenti nel footer e aggiornare le informazioni di contatto, mantenendo il sito sempre allineato con le necessità aziendali.

Capitolo 2

Analisi concettuale del sistema

Per l'analisi concettuale del sistema, utilizziamo il diagramma Entità-Relazione, uno strumento grafico fondamentale per progettare la base di dati. Questo schema permette di rappresentare visivamente le entità principali, i loro attributi e le relazioni tra loro. L'analisi concettuale costituisce il primo passo per creare un modello dati solido, identificando le componenti essenziali e ottimizzando l'architettura della piattaforma. Il diagramma ER aiuta a garantire una gestione efficiente dei dati, riducendo le ridondanze e migliorando la coerenza delle informazioni nel sistema.

2.1 Schema Entità-Relazione 1

Nella *Figura 2.1* è rappresentato il diagramma ER relativo ai prodotti, alla loro categoria, ai supermercati in cui sono presenti e al carrello che viene riempito di questi prodotti. La *Tabella 2.1* contiene l'elenco delle entità con la descrizione per ciascuna di esse; contiene, inoltre, l'elenco degli attributi e gli identificatori. La *Tabella 2.2* contiene l' elenco delle relazioni con una breve descrizione di ciascuna. Inoltre, vengono riportate le componenti delle relazioni, o i loro attributi, e i loro identificatori.

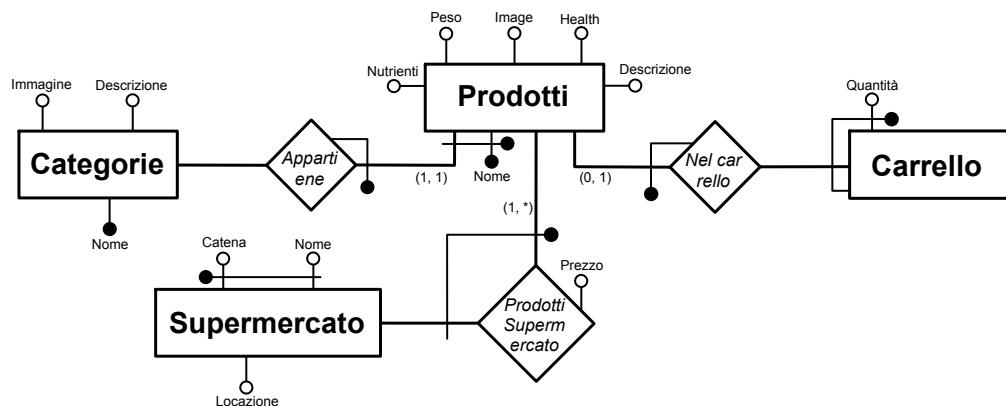


Figura 2.1 Schema Entità-Relazione 1

Entità	Descrizione	Attributi	Identificatori
Prodotti	Rappresenta i prodotti disponibili per l'acquisto	Peso, Immagine, Health, Descrizione, Nome, Nutrienti	{Nome}
Categorie	Rappresenta le diverse categorie di prodotti	Immagine, Descrizione, Nome	{Nome}
Carrello	Contiene i prodotti selezionati dagli utenti per l'acquisto	Quantità	{Quantità}
Supermercato	Rappresenta i supermercati dove sono disponibili i prodotti	Nome, Catena, Locazione	{Nome, Catena}

Tabella 2.1 Entità 1

Relazione	Descrizione	Componenti	Attributi	Identificatori
Appartiene	Lega i prodotti alla propria categoria di appartenenza	Prodotti e Categorie		Prodotti
Nel carrello	Rappresenta i prodotti presenti nel carrello	Prodotti e Carrello		Prodotti
Prodotti Supermercato	Rappresenta i prodotti disponibili in un supermercato	Prodotti e Supermercato	Prezzo	Prodotti e Supermercato

Tabella 2.2 Relazioni 1

2.2 Schema Entità-Relazione 2

Nella Figura 2.2 è rappresentato il diagramma ER relativo agli utenti, alle loro recensioni e ai loro carrelli preferiti. La Tabella 2.3 contiene l'elenco delle entità con la descrizione per ciascuna di esse; contiene, inoltre, l'elenco degli attributi e gli identificatori. La Tabella 2.4 contiene l'elenco delle relazioni con una breve descrizione di ciascuna. Inoltre, vengono riportate le componenti delle relazioni, o i loro attributi, e i loro identificatori.

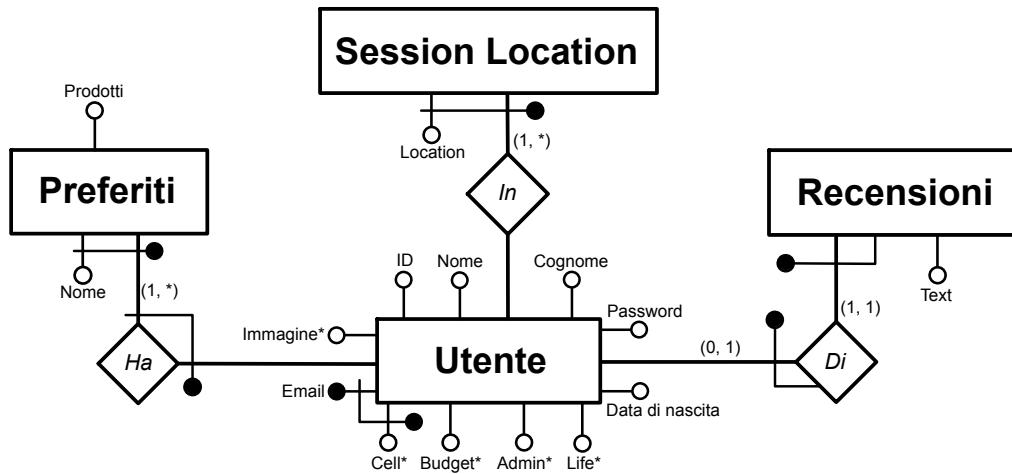


Figura 2.2 Schema Entità-Relazione 2

Entità	Descrizione	Attributi	Identificatori
Utente	Rappresenta l'utente registrato che utilizza la piattaforma	ID, Nome, Cognome, Data di nascita, Email, Password, Cell*, Immagine*, Budget*, Stile di vita*, Admin*	{Email, Cell*}
Preferiti	Rappresenta i carrelli salvati sul profilo dall'utente	Nome, Prodotti	
Recensioni	Rappresenta le recensioni rilasciate dagli utenti sulla piattaforma	Text	
Session Location	Indica la locazione dell'utente	Location	

Tabella 2.3 Entità 2

Relazione	Descrizione	Componenti	Attributi	Identificatori
Ha	Lega l'utente ai suoi carrelli salvati sul profilo	Utente e Preferiti		Utente e Preferiti
Di	Lega l'utente alle recensioni che ha scritto	Utente e Recensioni		Utente
In	Lega l'utente alla sua location	Utente e Session Location		

Tabella 2.4 Relazioni 2

2.3 Le business rules

Il modello concettuale ER ha una capacità espressiva limitata, il che implica che alcuni vincoli della realtà rappresentata non possono essere descritti unicamente tramite questo modello. Per completare il diagramma ER, è quindi necessario integrarlo con un insieme di regole aziendali, dette *business rules*.

Vincoli d'integrità non esprimibili nel diagramma ER	
1	Quando si accede ai <i>Prodotti</i> si vuole sempre sapere a quale <i>Categoria</i> appartengono.
2	Quando si accede al <i>Carrello</i> si vuole sempre sapere quali <i>Prodotti</i> sono già all'interno.
3	Quando si accede al <i>Carrello</i> si vuole solo sapere <i>Nome</i> , <i>Immagine</i> e a quale <i>Categoria</i> appartiene quel prodotto.
4	Nella relazione <i>Ha</i> si vuole sapere solo <i>l'email dell'Utente</i> .
5	Nella relazione <i>Di</i> si vuole sapere solo <i>l'email dell'Utente</i> .
6	Quando si accede ai <i>Preferiti</i> si vuole sempre sapere quale <i>Utente</i> ha salvato quel <i>Carrello</i> .
7	Quando si accede alle <i>Recensioni</i> si vuole sempre sapere a quale <i>Utente</i> appartengono.
8	Quando si accede a <i>Session Location</i> si vuole sapere <i>l'ID dell'Utente</i> che partecipa alla relazione <i>In</i> .

Tabella 2.5 Vincoli d'integrità

Capitolo 3

Tecnologie e metodologie utilizzate

MyShoppingList è stato sviluppato utilizzando le principali tecnologie per il web: HTML, CSS, JavaScript e PHP. Questi strumenti sono stati combinati per offrire un'esperienza utente fluida e interattiva, curando ogni aspetto, dalla struttura alla presentazione, fino alla funzionalità lato server. Inoltre, il progetto integra un'API per le mappe, migliorando la navigazione e la localizzazione dei supermercati, e offre autenticazione tramite servizi esterni, rendendo il processo di login sicuro e pratico.

3.1 HTML e CSS

HTML (HyperText Markup Language) e CSS (Cascading Style Sheets) sono due tecnologie fondamentali che collaborano per creare pagine web ben strutturate e graficamente curate.

HTML è il linguaggio di markup che stabilisce la struttura di base di una pagina web. Permette di organizzare il contenuto, come testo, immagini, video e altri elementi multimediali, in un formato che può essere interpretato dai browser. Attraverso tag come intestazioni, paragrafi, liste e link, HTML definisce un ordine chiaro e facilmente navigabile per le informazioni.

CSS, invece, si occupa della parte estetica del sito. Mentre HTML organizza il contenuto, CSS ne definisce l'aspetto, intervenendo su colori, font, spaziature, layout e adattabilità su dispositivi differenti. Separando la parte strutturale da quella visiva, CSS rende più semplice l'aggiornamento e la gestione dello stile del sito. È possibile applicare CSS direttamente nel documento HTML o tramite fogli di stile esterni, favorendo la riusabilità del codice e garantendo un design uniforme su tutto il sito.

3.2 JavaScript

JavaScript è essenziale per rendere i siti web interattivi e dinamici. Questo linguaggio di programmazione, eseguito direttamente nel browser dell'utente, permette di modificare gli elementi HTML e CSS in tempo reale. Grazie a JavaScript, un sito può reagire prontamente alle azioni dell'utente, come click, scroll o l'invio di moduli.

Questo migliora l'esperienza dell'utente, rendendo l'interfaccia più fluida e facile da navigare, consentendo anche l'integrazione di funzionalità avanzate.

Un aspetto importante di JavaScript è la sua capacità di supportare AJAX (Asynchronous JavaScript and XML), un insieme di tecniche che permette di inviare richieste al server e ricevere dati senza dover ricaricare l'intera pagina. In pratica, quando l'utente compie un'azione, come inviare un modulo o selezionare un'opzione, JavaScript invia una richiesta al server e aggiorna solo le parti della pagina necessarie. Questo processo avviene in background, garantendo aggiornamenti rapidi del contenuto senza interruzioni. L'uso di AJAX aumenta notevolmente la velocità del sito, riducendo i tempi di attesa e migliorando l'esperienza di navigazione.

3.3 PHP

PHP (PHP: Hypertext Preprocessor) è un linguaggio di scripting utilizzato lato server, fondamentale nello sviluppo di applicazioni web. A differenza di HTML, CSS e JavaScript, che operano nel browser dell'utente, PHP lavora sul server, gestendo la logica dell'applicazione e interagendo con i database. Esso si occupa di elaborare le richieste inviate dagli utenti e generare dinamicamente il contenuto delle pagine, assicurando che le informazioni siano personalizzate e sempre aggiornate in base ai dati conservati nel database.

Un punto di forza di PHP è la sua stretta integrazione con database relazionali come MySQL, che consente la creazione di applicazioni web scalabili e performanti, in grado di gestire volumi di dati elevati. Inoltre, PHP si combina perfettamente con AJAX per migliorare l'interazione dell'utente. In un'applicazione che manipola dati complessi, AJAX consente di inviare richieste al server per aggiornare solo parti specifiche della pagina, evitando il ricaricamento completo. PHP si occupa di elaborare queste richieste, accedere ai dati nel database e restituire una risposta al client, creando un'esperienza più fluida e reattiva.

Nel progetto MyShoppingList, l'uso combinato di PHP e AJAX è stato cruciale. Per esempio, durante la navigazione tra i prodotti, AJAX permette di aggiornare i risultati in tempo reale senza ricaricare l'intera pagina, migliorando la velocità e l'efficacia della ricerca. Per quanto riguarda il login, PHP gestisce la verifica delle credenziali mentre AJAX consente di validare i dati inseriti in tempo reale, ottimizzando sia l'efficienza che la sicurezza del processo di autenticazione.

3.4 Autenticazione con servizi esterni

L'autenticazione tramite servizi di terze parti è un sistema che consente agli utenti di accedere a MyShoppingList utilizzando il proprio account di una piattaforma esterna, evitando la necessità di creare nuove credenziali specifiche per il sito. Questo processo si basa sul protocollo *OAuth 2.0*, che permette di gestire in modo sicuro l'autorizzazione dell'utente senza condividere direttamente le credenziali di login. Nel nostro caso abbiamo deciso di optare per un autenticazione tramite l'account della piattaforma GitHub.

Quando un utente desidera effettuare l'accesso al sito tramite GitHub, viene reindirizzato alla pagina di autorizzazione di GitHub, dove gli viene chiesto di concedere il permesso all'applicazione di accedere ad alcune informazioni del suo profilo. Una volta che l'utente autorizza l'accesso, GitHub fornisce un codice di autorizzazione, che viene poi utilizzato dal server per ottenere un token di accesso. Questo token permette di interagire con l'API di GitHub e recuperare i dati necessari.

L'integrazione tra PHP e GitHub consente di gestire questa comunicazione: PHP invia la richiesta di autorizzazione, ottiene il token di accesso e utilizza questo token per recuperare informazioni sull'utente, che vengono poi memorizzate nella sessione per utilizzarle successivamente. Il sistema è reso ancora più fluido grazie all'utilizzo di AJAX, che consente di eseguire queste operazioni in background senza interrompere l'interazione dell'utente con il sito.

Questa modalità di login semplifica l'accesso a MyShoppingList e migliora l'esperienza dell'utente, poiché consente di utilizzare un account già esistente senza dover ricordare nuove credenziali.

3.5 API

Per la gestione delle mappe che mostrano la posizione dei supermercati nel processo di checkout, *Capitolo 7*, è stata utilizzata la libreria open-source Leaflet. Leaflet è una libreria JavaScript leggera e altamente versatile, progettata per facilitare la creazione di mappe interattive, permettendo la visualizzazione e la navigazione in modo semplice e intuitivo. Grazie alla sua struttura efficiente, consente di aggiungere funzionalità avanzate come marker, popup e livelli, senza compromettere le performance del sito. Per importarla è sufficiente includere all'interno del codice uno script.

Per implementare una mappa, è necessario creare un'istanza della mappa in JavaScript e configurarla con le coordinate geografiche del punto di interesse. Nel caso specifico del nostro sito, dopo aver ricevuto dal server le coordinate del supermercato più conveniente tramite l'algoritmo di selezione, viene creata una mappa centrata sulla posizione esatta del supermercato. Un marker viene poi posizionato sulla mappa per indicare visibilmente la sua collocazione (*Codice 3.1*). Questo approccio consente di visualizzare in tempo reale la posizione dei supermercati.

```
Latitudine = parseFloat(response.recommendedSupermarket.latitude);
Longitudine = parseFloat(response.recommendedSupermarket.longitude);
var map = L.map('map3').setView([Latitudine, Longitudine], 15);
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 19, attribution: '&copy;
    <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>');
L.marker([Latitudine, Longitudine], { icon: greenIcon }).addTo(map);
```

Codice 3.1 Script creazione di una mappa

Capitolo 4

Home Page

Entrando nel sito, l'utente viene automaticamente indirizzato alla pagina iniziale, la Home Page, dove viene offerta una panoramica generale sul funzionamento e sugli scopi della piattaforma. In particolare, vengono presentati in maniera chiara e sintetica i punti cardine del sito, ovvero:

- Convenienza
- Vicinanza
- Salubrità

È possibile, scorrendo in basso, visualizzare una dettagliata rassegna sia delle catene dei supermercati selezionati che dei principali marchi di prodotti trattati dal sito, maggiormente diffusi nell'uso quotidiano.

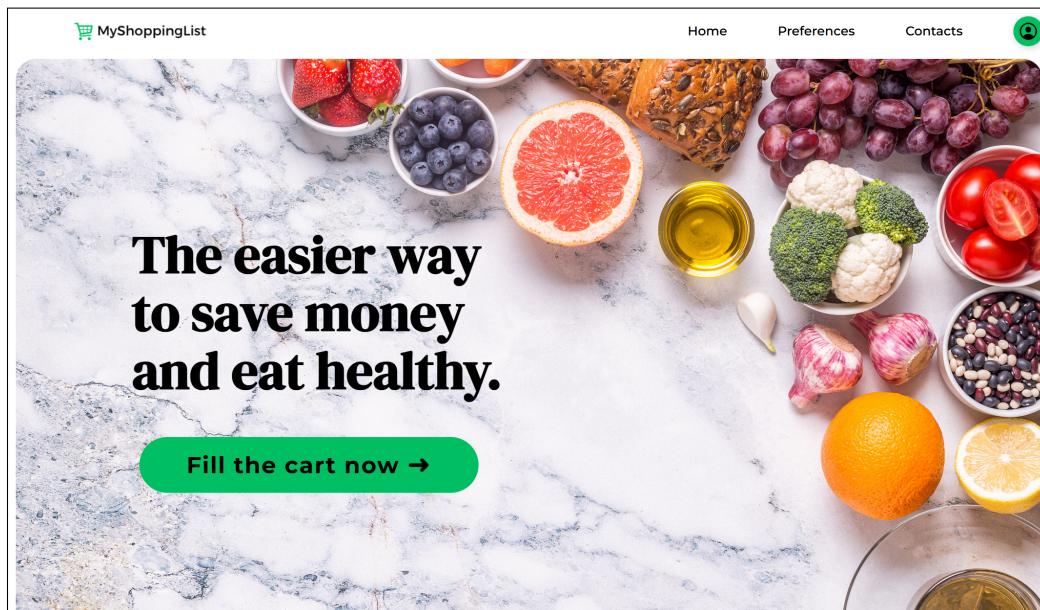


Figura 4.1 Home Page

Un elemento centrale nella pagina è il pulsante *Fill the cart now*, che cattura subito l'attenzione dell'utente. Attraverso un link ipertestuale, il pulsante indirizza l'utente alla sezione principale del sito, dove potrà scegliere e aggiungere i prodotti al proprio carrello.

4.1 Navbar

In cima alla pagina è posizionata una barra di navigazione, visibile e accessibile da ogni sezione del sito. Questo menù fornisce un modo semplice e diretto per esplorare le principali funzionalità e risorse del sito. Tramite la navbar, gli utenti possono facilmente accedere al footer, registrarsi o entrare nel proprio account, consultare la propria lista dei carrelli salvati, oppure tornare alla Home Page.

Cliccando sul pulsante in alto a destra, l'utente viene indirizzato alla sezione di login o registrazione. Al momento del click, uno script JavaScript viene eseguito per mostrare il pop-up di login (*Figura 4.2*). Lo script agisce modificando l'attributo CSS *display* del pop-up, inizialmente nascosto (*display: none*), rendendolo visibile all'utente. Il pop-up funge da accesso all'area personale dell'utente, permettendo di scegliere tra il login con credenziali esistenti, l'accesso tramite GitHub come descritto precedentemente, oppure, per i nuovi utenti, la registrazione per creare un nuovo account.

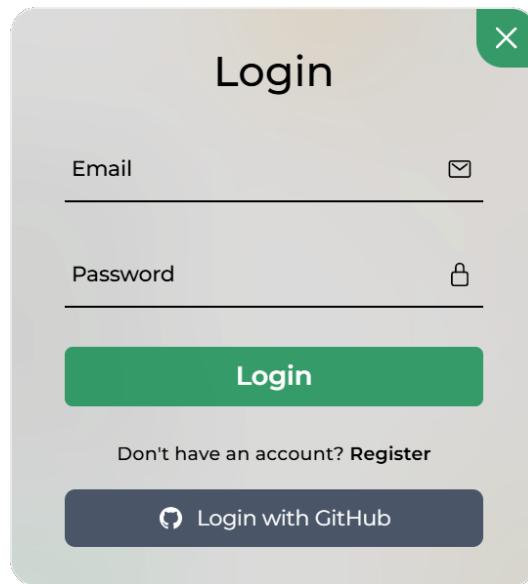


Figura 4.2 Popup Login

4.2 Recensioni

Come accennato prima, solo gli utenti registrati hanno la possibilità di lasciare una recensione sul sito, con l'opzione di modificarla o rimuoverla in seguito. Tuttavia, queste recensioni rimangono visibili sulla Home Page per tutti gli utenti, indipendentemente dal fatto che siano registrati o meno.

The screenshot shows a user interface for leaving a review. At the top, there's a button labeled 'Leave a review'. Below it is a text input field with placeholder text 'Write your review here...'. Underneath the input field are three buttons: 'Send' (green), 'Delete' (red), and 'Edit' (green). Below this section, there's a heading 'Reviews:' followed by a single review entry: 'marco.pedicillo@outlook.it: Piattaforma molto utile e fatta bene!'

Figura 4.3 Recensioni

4.2.1 Lato server

Grazie a richieste AJAX inviate al server, citate prima, PHP può elaborare e gestire l'inserimento, la modifica o l'eliminazione di recensioni visibili nella Home Page:

- Se la richiesta è di tipo **POST** e include il parametro *add*, il codice inserisce una nuova recensione. Si recuperano l'email e il testo della recensione dai dati inviati, si concatenano e il risultato viene memorizzato nel database usando una query SQL. Se l'inserimento va a buon fine, viene mostrato un messaggio di successo (*Codice 4.1*).
- In caso di richiesta **POST** con il parametro *mod*, il codice aggiorna il testo della recensione per un'email specifica, sempre tramite una query SQL. La nuova recensione sostituisce quella precedente per lo stesso utente nel database. La figura del codice viene omessa poichè molto simile a quella dell'aggiunta.
- Se la richiesta è di tipo **DELETE**, il codice elimina la recensione associata a un'email specifica. Si recupera l'email dai dati inviati, la quale viene usata per identificare la recensione da rimuovere nel database. Se l'operazione ha successo, viene restituito un messaggio di conferma (*Codice 4.2*).

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {  
    parse_str(file_get_contents("php://input"), $data);  
  
    $email = isset($_POST['email']) ? $_POST['email'] : 'No session ID set';  
    $text = $_POST['text'];  
  
    $fullText = $email . ":" . $text;  
  
    if(isset($data['add']) && $data['add']){
        $stmt = $conn→prepare("INSERT INTO reviews (email, text) VALUES (?, ?)");
        $stmt→bind_param("ss", $email, $fullText);  
  
        if ($stmt→execute()) {
            echo "Recensione aggiunta con successo!";
        } else {
            echo "Errore: " . $stmt→error;
        }
    }
}
```

Codice 4.1 Aggiunta Recensione

```
if ($_SERVER["REQUEST_METHOD"] == "DELETE") {  
    parse_str(file_get_contents("php://input"), $data);  
    $email = $data['email'];  
  
    $stmt = $conn→prepare("DELETE FROM reviews WHERE email = ?");  
    $stmt→bind_param("s", $email);  
  
    if ($stmt→execute()) {
        echo "Recensione eliminata con successo!";
    } else {
        echo "Errore: " . $stmt→error;
    }
}
```

Codice 4.2 Eliminazione Recensione

4.3 Footer

Nel footer gli utenti possono trovare le informazioni di contatto, che includono i link ai profili social ufficiali di MyShoppingList, l'indirizzo e-mail per assistenza e il numero verde per il supporto clienti. Come vedremo nel *Capitolo 9* tutte queste informazioni possono essere modificate da un utente *Admin*.

Quando l'utente arriva alla fine della Home Page, nell'angolo in basso a destra comparirà un pulsante *Back to Top*. Cliccando su di esso, l'utente viene riportato automaticamente in cima alla pagina (*Codice 4.3*).

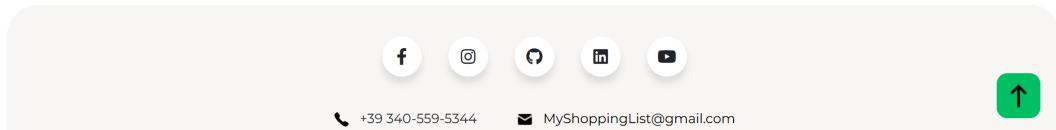


Figura 4.4 Footer

```
window.onscroll = function() {scrollFunction()};

function scrollFunction() {
    var backToTopBtn = document.getElementById("back-to-top-btn");
    if (document.body.scrollTop > 2000 || document.documentElement.scrollTop > 2000) {
        backToTopBtn.classList.add("show");
    } else {
        backToTopBtn.classList.remove("show");
    }
}

document.getElementById("back-to-top-btn").addEventListener("click", function(event) {
    event.preventDefault();
    document.body.scrollTop = 0;
    document.documentElement.scrollTop = 0;
});
```

Codice 4.3 Script Back to Top

4.4 Responsiveness

Come già menzionato, tutte le pagine del sito sono state progettate per adattarsi perfettamente a schermi di varie dimensioni grazie all'uso delle *media query*. Nella home page, in particolare, il ridimensionamento della finestra porta a una riorganizzazione della barra di navigazione: il link *Contacts* si trasforma in un'icona a forma di telefono, mentre *Home* viene sostituito dal logo di MyShoppingList, che diventa cliccabile. Anche gli altri elementi della pagina vengono riorganizzati per garantire un'esperienza di navigazione agevole e ben strutturata su dispositivi con schermi più piccoli.

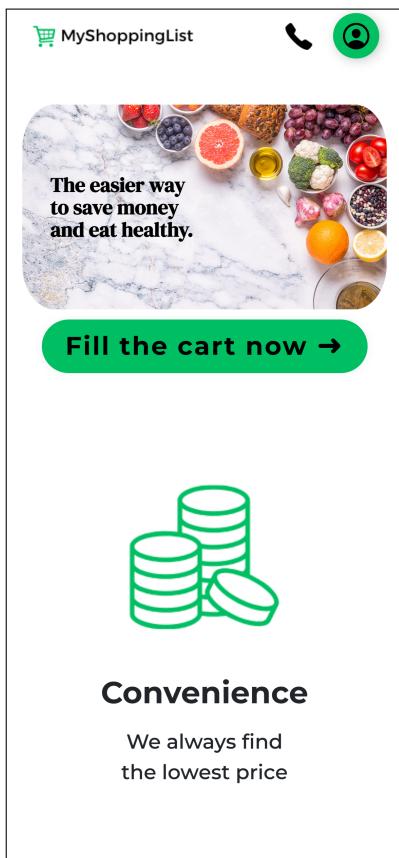


Figura 4.5 Dimensioni Mobile

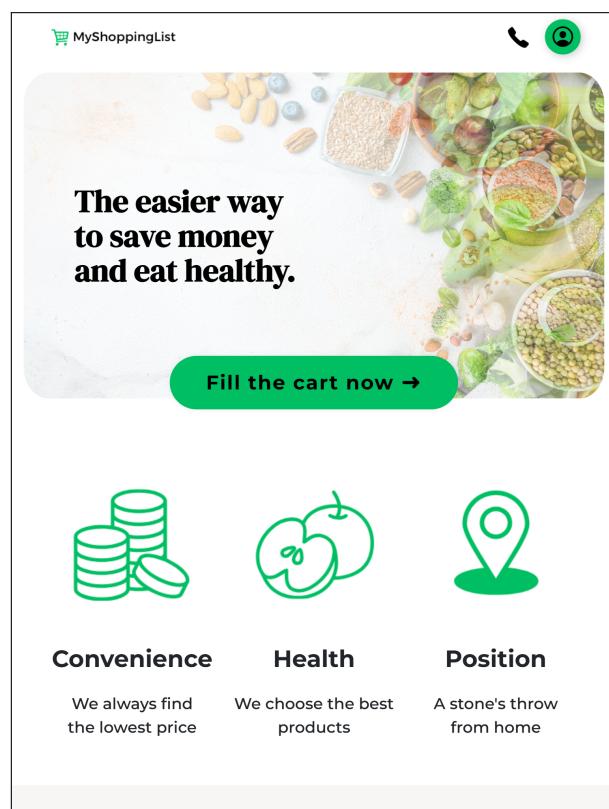


Figura 4.6 Dimensioni Tablet

Capitolo 5

User Page

Un utente può accedere al proprio account tramite il popup di login o, se è al primo accesso, può registrarsi alla piattaforma attraverso la pagina apposita oppure può autenticarsi direttamente con GitHub.

5.1 Registrazione

Per registrarsi come nuovo utente è sufficiente cliccare sul pulsante *Register* posizionato nella parte inferiore del popup di login. Questo collegamento porterà l'utente a una nuova sezione della piattaforma, dove sarà possibile completare un form con i propri dati personali e creare un account (*Figura 5.1*).

Figura 5.1 Pagina di Registrazione

Dopo aver inserito i propri dati, selezionato lo stile di vita preferito e, facoltativamente, specificato il *Max Amount*, l'utente può completare la registrazione cliccando sul pulsante *Register Now!*. In questo momento, il browser invia una richiesta AJAX al server con le informazioni fornite. Il server elabora la richiesta e salva i dati nel database relativo agli utenti.

Se la registrazione è avvenuta correttamente, l'utente viene automaticamente autenticato e reindirizzato alla Home Page. Ora, nella barra di navigazione, il pulsante di login è sostituito da un'icona che consente l'accesso al profilo, rappresentata dall'iniziale del nome dell'utente (*Figura 5.2*).



Figura 5.2 Navbar per utente autenticato

5.1.1 Lato server

Quando l'utente compila il modulo di registrazione e clicca su *Register Now!* tutte le informazioni fornite vengono raccolte e inserite in un oggetto `FormData`. Questa interfaccia JavaScript è progettata per semplificare l'invio di dati tramite richieste HTTP, permettendo di includere sia file che informazioni testuali, organizzati in modo che il server possa gestirli facilmente.

Una volta creato l'oggetto `FormData`, viene inviato al server tramite una richiesta AJAX. Il server riceve i dati e, in base al tipo di richiesta (*POST* o *GET*) e al contenuto, esegue le operazioni necessarie per completare la registrazione dell'utente. I passaggi in ordine sono (*Codice 5.1*):

1. I dati vengono estratti dai campi `$_POST` e assegnati a variabili corrispondenti, come `$name`, `$surname`, `$birthdate`, ecc.
2. Il percorso della cartella in cui verranno salvati gli avatar è specificato in `$target_dir`. `$target_file` combina la cartella di destinazione con il nome del file dell'immagine per ottenere il percorso completo dell'immagine.
3. Se c'è un errore nel caricamento dell'immagine (`UPLOAD_ERR_OK` verifica che non ci siano errori), viene stampato un messaggio di errore.
4. Se il caricamento dell'immagine ha generato un errore, il codice esegue comunque l'inserimento nel database, ma senza il campo dell'immagine.
5. Entrambe le query `INSERT` verificano l'esito dell'inserimento: se ha successo, viene stampato un messaggio di conferma, altrimenti viene mostrato un errore con i dettagli.

Inoltre, con l'implementazione di un vincolo di unicità sulla colonna *e-mail* nel database, non è consentito creare più account utilizzando lo stesso indirizzo e-mail.

```
if($_SERVER['REQUEST_METHOD'] == 'POST'){

    $name = $_POST['name'];
    $surname = $_POST['surname'];
    $birthdate = $_POST['birthdate'];
    $phone = $_POST['phone'];
    $email = $_POST['email'];
    $lifestyle = $_POST['stile_di_vita'];
    $maxamount = $_POST['importo'];
    $password = $_POST['password'];
    $ID=$_POST['id'];
    $admin=$_POST['admin'];

    $target_dir = " ../../uploads/";
    $target_file = $target_dir . basename($_FILES["avatar-upload"]["name"]);

    if ($_FILES["avatar-upload"]["error"] != UPLOAD_ERR_OK) {
        echo "Si è verificato un errore durante il caricamento del file: "
        . $_FILES["avatar-upload"]["error"];

        $sql = "INSERT INTO utenti (id, email, pass, nome, cognome, born, cell, life,
                                budget, adm)
VALUES ('$ID', '$email', '$password', '$name', '$surname', '$birthdate', '$phone',
        '$lifestyle', '$maxamount', '$admin')";

        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
        } else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }
    } else {
        if (move_uploaded_file($_FILES["avatar-upload"]["tmp_name"], $target_file)) {
            echo "Il file ". htmlspecialchars( basename( $_FILES["avatar-upload"]["name"])) .
                " è stato caricato.";
        } else {
            echo "Si è verificato un errore durante il caricamento del file.";
        }

        $sql = "INSERT INTO utenti (id, immagine, email, pass, nome, cognome, born, cell,
                                life, budget, adm)
VALUES ('$ID', '$target_file', '$email', '$password', '$name', '$surname', '$birthdate',
        '$phone', '$lifestyle', '$maxamount', '$admin')";

        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
        } else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }
    }
}
```

Codice 5.1 Nuova Registrazione

5.2 Login

Per accedere al proprio account, l'utente registrato deve inserire l'indirizzo e-mail e la password, scelti durante la registrazione, nei campi del modulo di accesso. Una volta completati i campi, è sufficiente premere il pulsante *Login* per inviare i dati al server. Se le credenziali sono corrette, il sistema autentica l'utente e lo reindirizza automaticamente alla propria pagina personale (*Figura 5.3*). In caso di errore nelle credenziali, verrà mostrato un avviso sotto forma di pop-up.

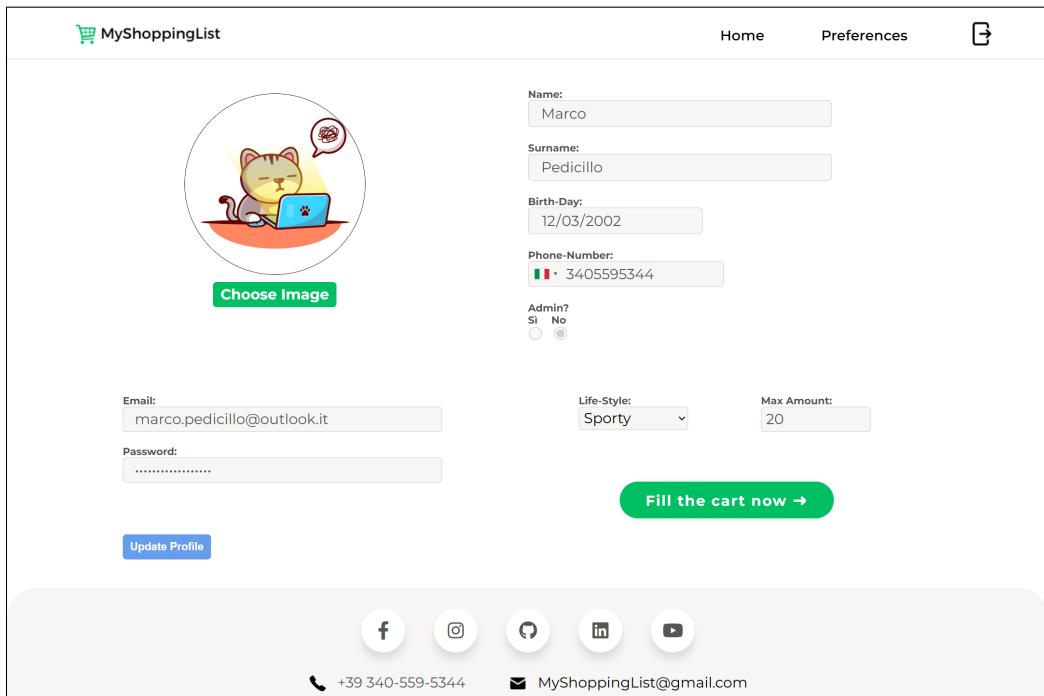


Figura 5.3 Pagina Utente

Nel profilo, l'utente può aggiornare le proprie informazioni personali, effettuare il logout, visualizzare i carrelli salvati e accedere direttamente alla pagina dei prodotti, dove è possibile selezionare e aggiungere articoli al carrello per finalizzare la spesa.

Tutte queste funzioni sono implementate sempre tramite richieste AJAX che vengono elaborate e gestite tramite PHP.

5.2.1 Lato server

Quando l'utente inserisce l'email e la password nei campi appositi, i dati vengono inviati al server tramite una richiesta AJAX gestita da JavaScript. Il server riceve la richiesta e prima di tutto determina se si tratta di una richiesta di tipo **GET** o **POST**. In base a questa distinzione e ai dati inviati, viene utilizzato il metodo `isset` di PHP, che verifica se una variabile è definita e non è nulla. Questo passo è fondamentale per controllare che i dati richiesti siano presenti e validi prima di procedere con ulteriori operazioni.

Una volta identificato il tipo di richiesta e validati i dati, il server esegue una query nel database per cercare un utente che corrisponda alle credenziali fornite. Se la query restituisce un risultato positivo, cioè se esiste un utente con quelle credenziali, il server risponde con un messaggio di *success*, permettendo all'utente di accedere al proprio account. I dati dell'utente vengono quindi prelevati dal database e utilizzati per riempire i campi della pagina. In caso contrario, se le credenziali non corrispondono a nessun utente, il server restituisce un messaggio di *failure* e impedisce l'accesso, mostrando un pop-up di errore nel browser del client (*Codice 5.2*).

Va evidenziato che la tabella utenti del database è strutturata con una chiave univoca per l'indirizzo e-mail, assicurando così che ogni credenziale sia unica e minimizzando il rischio di errori durante il processo di accesso.

```
if($_SERVER['REQUEST_METHOD'] == 'POST'){
    if(isset($_POST['email']) && isset($_POST['pass'])){
        $email = $_POST['email'];
        $pass = $_POST['pass'];

        $sql = "SELECT * FROM utenti WHERE email = '$email' AND pass = '$pass'";
        $result = $conn→query($sql);

        if ($result→num_rows > 0) {
            echo "success";
        } else {
            echo "failure";
        }
    }

    else {
        echo "I dati inviati non sono validi.";
    }
}
```

Codice 5.2 Autenticazione

5.3 Aggiornamento profilo

L'utente autenticato può aggiornare le proprie informazioni personali nella sezione del profilo. Per farlo, basta modificare i campi desiderati e cliccare sul pulsante *Update Profile* situato in basso a sinistra. È importante notare che alcuni campi, come nome, cognome, data di nascita, email e la checkbox per i diritti da amministratore, non sono modificabili, in quanto considerati dati fondamentali dell'utente. Tuttavia, l'utente ha la possibilità di aggiornare la foto del profilo, il numero di cellulare, la password, lo stile di vita e il budget di spesa.

La funzionalità di aggiornamento del profilo è stata implementata seguendo una logica simile a quella utilizzata per la creazione di un nuovo account. In pratica, il browser client raccoglie i dati aggiornati in un oggetto FormData e li invia al server tramite una richiesta HTTP. Una volta ricevuta la richiesta, il server esamina i dati per verificare se i dati sono stati realmente modificati. Successivamente, vengono aggiornate le informazioni nel database, individuando l'utente tramite l'indirizzo e-mail (*Codice 5.3*).

```
if ($image_changed) {
    $sql = "UPDATE utenti SET
        immagine='$target_file',
        cell='$phone',
        pass='$password',
        life='$lifestyle',
        budget='$maxamount'
        WHERE email='$email'";
} else {
    $sql = "UPDATE utenti SET
        cell='$phone',
        pass='$password',
        life='$lifestyle',
        budget='$maxamount'
        WHERE email='$email'";
}

if ($conn→query($sql) === TRUE) {
    echo "Profilo aggiornato con successo";
} else {
    echo "Errore durante l'aggiornamento del profilo: " . $conn→error;
}
```

Codice 5.3 Aggiornamento profilo

5.4 Login di terze parti

Come descritto nel *Capitolo 3.4*, abbiamo implementato la possibilità di autenticarsi tramite la piattaforma GitHub senza dover creare un account con delle nuove credenziali. L'utente autorizza l'applicazione ad accedere ai propri dati, che vengono quindi utilizzati per personalizzare l'interazione con la piattaforma. Il procedimento è il seguente (*Codice 5.4*):

1. L'utente clicca su un link che lo indirizza alla pagina di autorizzazione di GitHub, dove dovrà dare il permesso per accedere al suo account. Il client, MyShoppingList, invia una richiesta di autorizzazione a GitHub, includendo il proprio *client_id*, la *redirect_uri* e lo *scope* (l'accesso ai dati dell'utente).
2. Se l'utente è autenticato su GitHub, vedrà una schermata in cui dovrà autorizzare l'accesso alla sua applicazione, concedendo permessi per accedere alle informazioni del suo profilo (come nome e email). Una volta autorizzato, GitHub invia un codice di autorizzazione alla *redirect_uri* di MyShoppingList.
3. Il server (PHP) riceve il codice di autorizzazione e lo scambia con un token di accesso. Questo passaggio avviene tramite una richiesta **POST** al server di GitHub, utilizzando il codice ricevuto. GitHub restituisce un *access_token*, che è utilizzato per autenticare le richieste successive al loro API.
4. Una volta ottenuto il token, il server lo usa per fare una richiesta API a GitHub e ottenere informazioni sull'utente, come il suo login, nome ed email. Questi dati vengono poi salvati nella sessione del server, permettendo alla piattaforma di personalizzare l'esperienza dell'utente.
5. Dopo aver completato l'autenticazione e recuperato i dati, l'utente viene reindirizzato alla pagina principale di MyShoppingList, dove può accedere alle funzionalità personalizzate in base alle informazioni ottenute da GitHub.

Questa integrazione non solo semplifica l'accesso per l'utente, ma riduce il rischio di errori e migliora la sicurezza, eliminando la necessità di creare e ricordare nuove credenziali. L'autenticazione tramite GitHub offre, inoltre, una gestione veloce e affidabile dei dati di accesso, garantendo un'esperienza utente senza interruzioni.

```
if (isset($_GET['code'])) {
    $code = $_GET['code'];
    $_SESSION['logged_github'] = false;
    $_SESSION['name'] = '';
    $_SESSION['email_git'] = '';

    $tokenUrl = 'https://github.com/login/oauth/access_token';
    $postData = [
        'client_id' => $clientId,
        'client_secret' => $clientSecret,
        'code' => $code,
        'redirect_uri' => $redirectUri,
    ];

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $tokenUrl);
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($postData));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array('Accept: application/json'));
    $response = curl_exec($ch);
    curl_close($ch);
    $data = json_decode($response, true);
    $accessToken = $data['access_token'];
    $userUrl = 'https://api.github.com/user';
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $userUrl);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'User-Agent: Demo',
        "Authorization: token $accessToken"
    ));
    $userData = curl_exec($ch);
    curl_close($ch);
    $user = json_decode($userData, true);
    $_SESSION['user'] = $user;
    $_SESSION['name']=$user['name'];

    $emailUrl = 'https://api.github.com/user/emails';
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $emailUrl);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'User-Agent: Demo',
        "Authorization: token $accessToken"
    ));
    $userEmail = curl_exec($ch);
    curl_close($ch);
    $email=json_decode($userEmail, true);
    $_SESSION['email']=$email[0]['email'];
    if(isset($_SESSION['email']) && !empty($_SESSION['email'])){
        echo "email settata";
        $_SESSION['logged_github'] = true;
    }

    header('Location: http://localhost/msl/MyShoppingList/src/User_Page/user.html');
    exit;
}

} else {
    echo 'Errore: codice di autorizzazione mancante.';
}
```

Codice 5.4 Login GitHub

Capitolo 6

Pagina dei prodotti

La sezione prodotti di MyShoppingList è il cuore dell'esperienza d'acquisto per l'utente (*Figura 6.1*). Qui, l'utente può esplorare la vasta gamma di articoli, selezionare quelli di interesse e aggiungerli facilmente al carrello virtuale. Inoltre, la pagina offre una raccolta di ricette tradizionali e semplici che aiutano nella scelta dei pasti: cliccando su una ricetta, gli ingredienti necessari vengono aggiunti automaticamente al carrello, semplificando il processo di acquisto. Una volta completata la selezione, l'utente può proseguire alla fase di checkout.

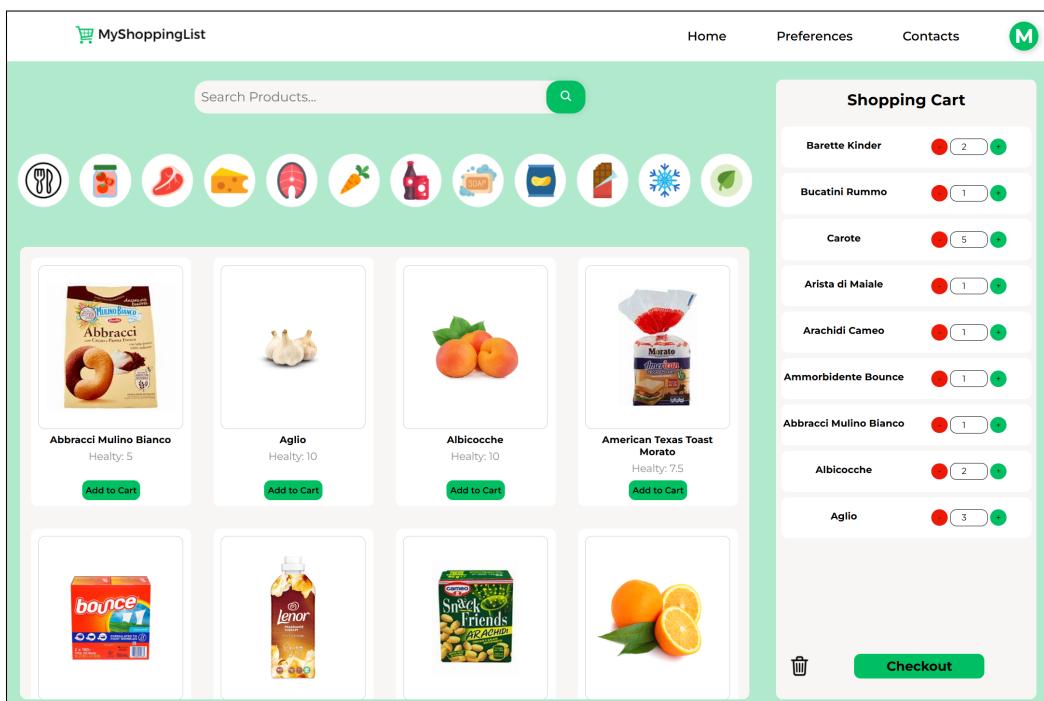


Figura 6.1 Pagina Prodotti

La pagina prodotti è progettata per mostrare in modo dinamico i prodotti disponibili e la maggior parte delle sue funzionalità è gestita dal server. Questo funge da intermediario tra client e database, gestendo richieste come il filtraggio per categoria, la ricerca di articoli specifici e l'aggiunta delle ricette al carrello. Anche le operazioni sul carrello, come l'aggiornamento delle quantità o lo svuotamento con un solo click, sono affidate al server, che risponde alle azioni dell'utente eseguendo le necessarie query sul database.

6.1 Localizzazione

Al momento dell'accesso alla pagina, il browser richiede l'autorizzazione per ottenere la posizione dell'utente. Questo permesso è fondamentale: senza di esso, durante il checkout non sarà possibile visualizzare il supermercato consigliato e, per gli utenti registrati, neanche individuare quello più vicino o più economico. Se l'utente accetta di condividere la posizione, le coordinate verranno memorizzate nel database e utilizzate per calcolare la distanza dai vari supermercati.

6.2 Barra di ricerca

La barra di ricerca è stata ideata per agevolare l'utente nel trovare rapidamente i prodotti di interesse. È costituita da un campo input, dove l'utente può inserire parole chiave, e da un pulsante per avviare la ricerca (*Figura 6.2*). Inserendo un termine e cliccando su *Cerca*, viene inviata una richiesta AJAX al server. Quest'ultimo elabora la richiesta, effettua una ricerca dei prodotti corrispondenti alla parola chiave, e restituisce i risultati. Se trovati, i prodotti vengono visualizzati immediatamente nella sezione dedicata, aggiornata dinamicamente.



Figura 6.2 Barra di Ricerca

6.2.1 Lato server

La Barra di Ricerca utilizza l'operatore SQL `LIKE` per individuare nel database i prodotti che contengono la parola chiave inserita dall'utente. Quando l'utente digita il nome di un prodotto e preme invio o il pulsante di ricerca, il server riceve una richiesta HTTP con il termine di ricerca. Se il campo di ricerca è vuoto, il server restituisce tutti i prodotti disponibili nel database; se invece è presente una parola chiave, viene eseguita una query per trovare tutti i prodotti il cui nome contiene il termine specificato (*Codice 6.1*).

```
if (isset($_GET['Nome'])) {
    $Nome = $_GET['Nome'];
    if ($Nome==""){
        $sql3 = "SELECT * FROM `products`";
    }else{
        $sql3 = "SELECT * FROM `products` WHERE Nome LIKE '%$Nome%'";
    }
    $risultato = $conn→query($sql3);
    $productsC = array();
    if ($risultato→num_rows > 0) {
        while ($row = $risultato→fetch_assoc()) {
            $productsC[] = $row;
        }
    }
    echo json_encode($productsC);
}
```

Codice 6.1 Script Barra di Ricerca

6.3 Filtri delle categorie

I filtri di categoria sono stati progettati per rendere la ricerca di prodotti più intuitiva e veloce per l'utente, analogamente alla barra di ricerca. Ogni filtro, rappresentato da un pulsante HTML con un ID specifico per la categoria (ad esempio: Carne, Pesce, Surgelati), ha un *event listener* per rilevare i click. Quando un filtro viene selezionato, l'ID del pulsante identifica la categoria e, tramite JavaScript, viene inviata una richiesta AJAX al server con il parametro della categoria. Se la richiesta ha esito positivo, la lista dei prodotti viene aggiornata dinamicamente per mostrare solo quelli della categoria scelta; in caso di errore, un messaggio viene riportato nella console di debug (*Figura 6.3*).



Figura 6.3 Filtri Categoria

6.3.1 Lato server

Quando l'utente seleziona un filtro nella pagina dei prodotti, il browser invia al server una richiesta con il nome della categoria selezionata. Il server verifica quindi se questa corrisponde a una categoria presente nel database o a un filtro speciale, come *Healthy* o *Tutti i Prodotti*. Se la categoria è valida, il server esegue una query specifica per ottenere i prodotti associati. In caso di filtro speciale, come *Healthy*, viene eseguita una query che restituisce solo i prodotti con un punteggio di salubrità pari a 8 o superiore, garantendo la visualizzazione di articoli salutari. Per il filtro *Tutti i Prodotti*, invece, viene visualizzato l'intero catalogo (*Codice 6.2*).

```
function getProductsByCategory($conn, $categoria) {
    if ($categoria == "All") {
        $sql2 = "SELECT * FROM `products`";
    } else if ($categoria == "Healty") {
        $sql2 = "SELECT * FROM `products` WHERE health≥8";
    } else {
        $sql2 = "SELECT * FROM `products` WHERE category = '$categoria'";
    }

    $risultato = $conn→query($sql2);
    $productsC = array();
    if ($risultato→num_rows > 0) {
        while ($row = $risultato→fetch_assoc()) {
            $productsC[] = $row;
        }
    }
    return json_encode($productsC);
}

if (isset($_GET['Categoria'])) {
    $categoria = $_GET['Categoria'];
    $productsC = User::getProductsByCategory($conn, $categoria);
    echo $productsC;
}
```

Codice 6.2 Verifica Filtri Categoria

6.4 Controllo della salubrità

Ogni alimento nella piattaforma ha un punteggio che ne indica la salubrità, e viene considerato salutare se il suo punteggio *Healthy* è pari o superiore a 6 (*Figura 6.4*). Quando un utente loggato con stile di vita *sportivo* prova ad aggiungere un prodotto al proprio carrello, il sistema controlla che l'alimento sia compatibile con lo stile di vita scelto. Se il prodotto non risulta salutare, compare un pop-up di avviso (*Figura 6.5*). A quel punto, l'utente può decidere se aggiungere comunque il prodotto alla lista della spesa. Se l'utente cerca di aggiungere al carrello un prodotto non conforme per l'ennesima volta, questo verrà aggiunto solo se già presente nel carrello. Quindi è possibile aumentare e diminuire la quantità del prodotto non salutare senza che compaia di continuo l'avviso. Da notare che il controllo sugli alimenti avviene solo per utenti registrati sportivi.



Figura 6.4 Prodotto non Healthy

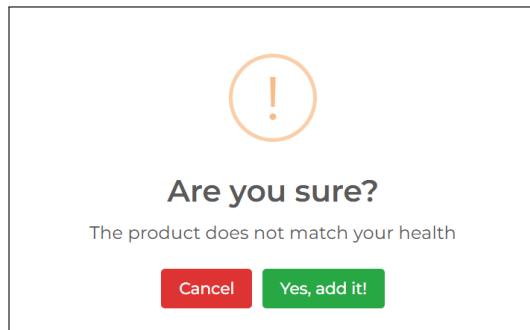


Figura 6.5 Pop-up Avviso Salubrità

6.4.1 Lato server

Ogni volta che un utente registrato con lo stile di vita *sportivo* tenta di aggiungere un prodotto al carrello, il server esegue una verifica sia sul suo stile di vita che sulla salubrità del prodotto per determinare se può essere incluso nel carrello. Quando l'utente clicca sul pulsante *Add to Cart* per un prodotto, il server riceve una richiesta AJAX contenente l'e-mail dell'utente, il nome del prodotto e il relativo punteggio di salubrità. Prima di tutto, viene controllato se il prodotto è già presente nel carrello; se sì, non sono necessari ulteriori controlli, poiché si presume che siano già stati effettuati. Se il prodotto non è presente nel carrello e il punteggio di salubrità è inferiore a 6, il server verifica lo stile di vita dell'utente tramite una query al database con l'indirizzo e-mail. Se l'utente ha uno stile di vita sportivo, viene inviato un messaggio al browser per informare l'utente che il prodotto non è adatto al suo stile di vita, mostrando un avviso tramite pop-up (*Codice 6.3*).

```

if (isset($_GET['action']) && $_GET['action'] == 'check_health') {
    if (isset($_GET['email']) && isset($_GET['health'])) {

        $email = $_GET['email'];
        $health = $_GET['health'];
        $name = $_GET['name'];
        $check_prodotto = "SELECT * FROM `Cart` WHERE Nome = '$name'";
        $result = $conn->query($check_prodotto);

        if ($result->num_rows > 0) { echo json_encode(false); exit; }
        if ($health < 6 && $health != null) {
            $check_health = "SELECT * FROM `utenti` WHERE email = '$email' AND life = 'Sportivo'";
            $result = $conn->query($check_health);
            if ($result && $result->num_rows > 0) { echo json_encode(true); exit; }
        }
        echo json_encode(false); exit;
    }
}

```

Codice 6.3 Check Health

6.5 Ricette suggerite

Ogni ricetta disponibile viene mostrata in un box dedicato, contenente un’immagine del piatto, una breve descrizione e un pulsante *Try it* che consente di aggiungere automaticamente tutti gli ingredienti al carrello (*Figura 6.6*). Come nel caso dei filtri, ogni pulsante nei vari box è associato a un ID specifico, corrispondente alla ricetta rappresentata, permettendo al sistema di gestire e riconoscere correttamente la selezione effettuata.

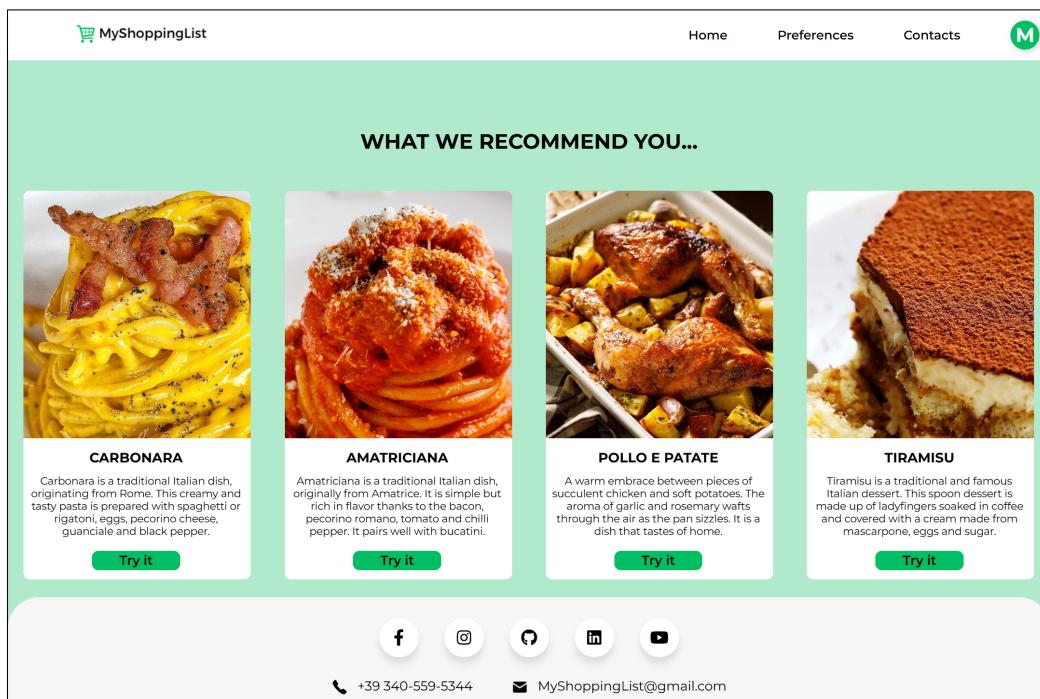


Figura 6.6 Ricette

6.5.1 Lato server

Ogni ricetta proposta dispone di un pulsante che, al momento del click, invia al server una richiesta AJAX contenente il nome della ricetta selezionata. Il server riceve la richiesta, estrae il nome della ricetta e individua il codice da eseguire. Una volta identificato il codice appropriato, il server avvia l'inserimento degli ingredienti necessari nel carrello tramite query SQL. Prima di procedere con l'inserimento, verifica se il prodotto è già presente nel carrello: se lo è, ne aggiorna la quantità invece di inserirlo nuovamente.

Dopo aver completato l'operazione, il server genera un array contenente il nome, la quantità, la categoria e l'immagine di ogni ingrediente aggiunto o aggiornato, restituendolo al browser del client. In questo modo, il browser può aggiornare dinamicamente il carrello, riflettendo le modifiche apportate (*Codice 6.4*).

```

if (isset($_GET['Ricetta'])) {
    $Ricetta = $_GET['Ricetta'];
    if ($Ricetta == "Carbonara") {

        $existingProduct = "SELECT * FROM `Cart` WHERE Nome = 'Uova'";
        $res = $conn->query($existingProduct);
        if ($res->num_rows > 0) {
            $update_product = "UPDATE `Cart` SET Quantità = Quantità + 1 WHERE Nome = 'Uova'";
            $res = $conn->query($update_product);
        } else {
            $insert_product = "INSERT INTO `Cart` (Nome, Quantità, Categoria, Immagine)
                            VALUES ('Uova', 1, 'Dispensa', 'products/uova.jpg')";
            $res = $conn->query($insert_product);
        }

        $existingProduct = "SELECT * FROM `Cart` WHERE Nome = 'Guanciale Beretta'";
        $res = $conn->query($existingProduct);
        if ($res->num_rows > 0) {
            $update_product = "UPDATE `Cart` SET Quantità = Quantità + 1 WHERE Nome = 'Guanciale Beretta'";
            $res = $conn->query($update_product);
        } else {
            $insert_product = "INSERT INTO `Cart` (Nome, Quantità, Categoria, Immagine)
                            VALUES ('Guanciale Beretta', 1, 'Salumi e Formaggi',
                                    'products/Guanciale_Beretta.jpg')";
            $res = $conn->query($insert_product);
        }

        ...
    }

    $prodotti = [
        [
            "Nome" => "Uova",
            "Quantità" => 1,
            "Categoria" => "Dispensa",
            "Immagine" => "products/uova.jpg"
        ],
        [
            "Nome" => "Guanciale Beretta",
            "Quantità" => 1,
            "Categoria" => "Salumi e Formaggi",
            "Immagine" => "products/Guanciale_Beretta.jpg"
        ]
        ...
    ];
    $json_prodotti = json_encode($prodotti); echo $json_prodotti;
}
}

```

Codice 6.4 Esempio Ricette

6.6 Carrello

Nella sezione del carrello (*Figura 6.7*), l’utente ha la possibilità di visualizzare in modo chiaro e ben organizzato tutti i prodotti che ha aggiunto alla lista della spesa. Questa sezione è strutturata all’interno di un box appositamente progettato, il quale è reso scorrevole grazie all’utilizzo della proprietà CSS *overflow-y: scroll*. Ciò permette all’utente di navigare facilmente tra gli articoli, senza che il numero di prodotti presenti influisca sulla visualizzazione.

Per ogni prodotto, l’utente ha la possibilità di modificare la quantità desiderata. Sono presenti dei bottoni che permettono di aumentare o diminuire il numero di unità per ciascun articolo. Al click di ciascun bottone, viene inviata una richiesta al server tramite AJAX, che aggiorna in tempo reale il database con la nuova quantità selezionata.

Inoltre, nella parte inferiore sinistra del box, è presente un’icona del cestino che consente all’utente di svuotare completamente il carrello. Anche questa funzionalità è gestita tramite una richiesta AJAX, garantendo un’operazione veloce e senza interruzioni.

Infine, il pulsante *Checkout* permette di procedere alla visualizzazione del supermercato consigliato. Se l’utente è loggato e ha impostato un budget massimo, al click verrà confrontato il totale della spesa con il budget. Se il totale supera il limite, un pop-up di avvertimento appare (*Figura 6.8*). Se il carrello è vuoto, un pop-up di errore segnalerà l’assenza di articoli.



Figura 6.7 Carrello

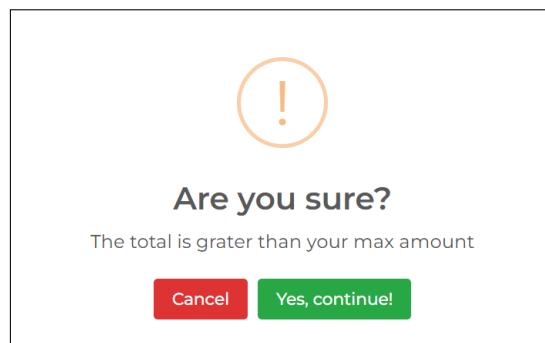


Figura 6.8 Pop-up Avviso Budget

6.7 Responsiveness

Quando la pagina viene ridimensionata, il box dei filtri diventa scorrevole, permettendo all'utente di esplorare tutte le opzioni anche con uno spazio visivo limitato. La barra di navigazione si adatta alla nuova pagina, come già implementato nella Home Page, per garantire un layout ottimale. Il carrello, invece, passa a una modalità a comparsa: non appare sempre in primo piano, ma si apre solo al click sull'icona del carrello accanto alla barra di ricerca, liberando così spazio prezioso su dispositivi mobili (*Figura 6.9*) e tablet (*Figura 6.10*). Anche tutti gli altri elementi della pagina sono stati ottimizzati in modo da rispondere dinamicamente alle diverse dimensioni dello schermo.

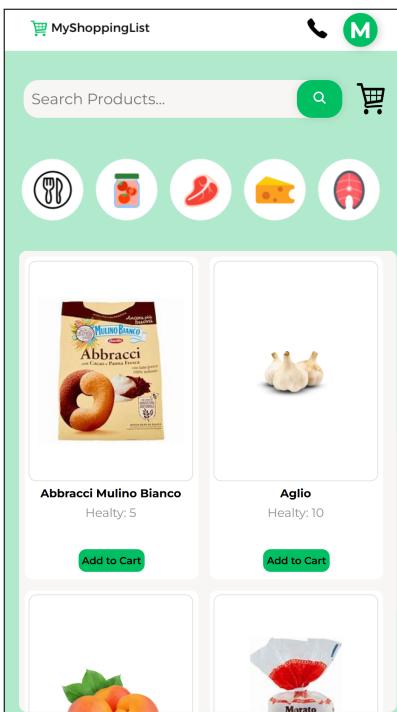


Figura 6.9 Versione Mobile

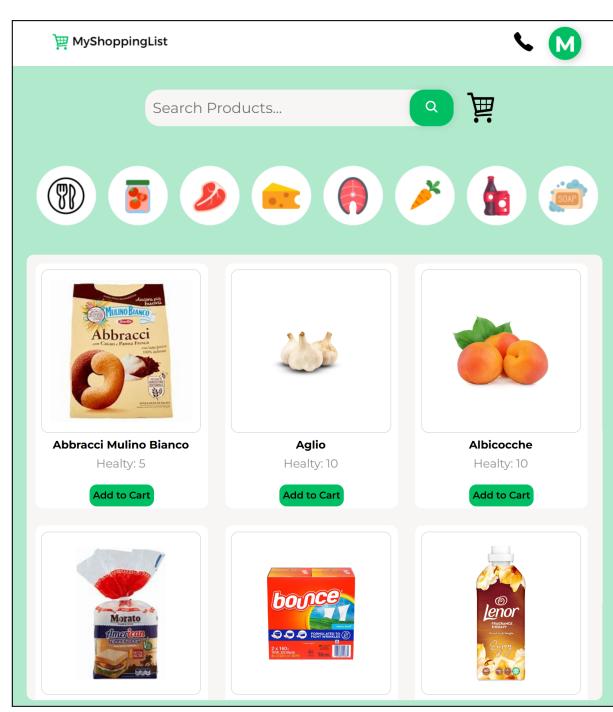


Figura 6.10 Versione Tablet

Capitolo 7

Checkout

Una volta completata la lista della spesa, l'utente può procedere con il checkout per scoprire quale supermercato offre le condizioni migliori. Selezionando il pulsante di checkout e dopo aver superato i controlli descritti in precedenza, si apre un pop-up che fornisce un riepilogo degli articoli selezionati, evidenziando il supermercato più vicino, quello più economico e quello consigliato con il costo totale della spesa associato. Questo risultato è ottenuto tramite l'accesso alla posizione dell'utente e l'uso di un algoritmo avanzato che analizza tutti i supermercati per individuare l'opzione ottimale sia in termini di distanza che di convenienza. Il layout del checkout varia tra utenti non registrati e utenti registrati. Per la visualizzazione delle mappe facciamo riferimento alla precedente spiegazione nel *Capitolo 3.5*.

7.1 Utente non registrato

Shopping Cart

name	quantity	price
Cacao Perugina	3	2.25
CremaNovi alla nocciola	1	4.25
Ciliegie	4	4.52
Arrosticini	2	8.18
Cotolette di Pollo Amadori	2	5.48
Arachidi Cameo	4	2.29
Birra Heineken	1	1.92
Coca Cola	1	1.96
Barette Kinder	1	3.22
Cornetti Tre Marie	1	3.05

SAVE YOUR CART **TOT: 75.71**

Want to know the closest or cheapest supermarkets?

Register Now!

Suggested Supermarket:

 Todis
supermarket.com

Placed: Casilina
Distance: 481.85 km
Total cost: 75.71 €



Figura 7.1 Checkout utente non registrato

Quando un utente non registrato procede al checkout, il pop-up mostrerà solo il supermercato consigliato. Le informazioni relative al supermercato più vicino e al più economico appariranno parzialmente visibili con un effetto di sfocatura (*Figura 7.1*). Anche la funzione per salvare il carrello tra i preferiti sarà disabilitata. Inoltre, cliccando su *Save your cart* per scaricare il PDF del carrello, verrà visualizzato un pop-up che invita l'utente a effettuare il login o a registrarsi, poiché questa funzionalità è riservata agli utenti registrati.

Per ottenere l'effetto di sfocatura, si è fatto uso di CSS insieme a Javascript. È stata creata una classe CSS chiamata *blurred*, che applica il filtro *filter: blur* per rendere sfocato l'elemento a cui è assegnata. Tramite Javascript, questa classe viene poi aggiunta dinamicamente ai box con le informazioni del supermercato più vicino e di quello più economico quando un utente non registrato effettua il checkout, oscurando così i dettagli e impedendo qualsiasi interazione.

7.2 Utente registrato

Shopping Cart		
name	quantity	price
Cacao Perugina	3	2.25
CremaNovi alla nocciola	1	4.25
Ciliegie	4	4.52
Arrosticini	2	8.18
Cotolette di Pollo Amadori	2	5.48
Arachidi Cameo	4	2.29
Birra Heineken	1	1.92
Coca Cola	1	1.96
Barette Kinder	1	3.22
Cornetti Tre Marie	1	3.05
		TOT: 75.71
SAVE YOUR CART ★		

Cheapest Supermarket :



Placed: San Lorenzo
Distance: 490.46 km
Total cost: 74.15 €



Closest Supermarket:



Placed: Tor Vergata
Distance: 479.91 km
Total cost: 76.33 €



Suggested Supermarket:



Placed: Casilina
Distance: 481.85 km
Total cost: 75.71 €



Figura 7.2 Checkout utente registrato

Un utente registrato avrà accesso a tutte le funzionalità del checkout. Potrà visualizzare non solo il supermercato consigliato, ma anche il supermercato più vicino e quello più economico, ciascuno corredata dalla mappa Leaflet. Inoltre, potrà salvare i carrelli tra i preferiti cliccando sul pulsante a forma di stella, rendendoli rapidamente accessibili per futuri acquisti (*Figura 7.2*). Se desidera scaricare la lista della spesa, potrà utilizzare il pulsante *Save your cart* per ottenere un file PDF personalizzato che include i prodotti selezionati con i relativi prezzi presso il supermercato consigliato. All'interno del PDF sarà presente anche un link diretto per visualizzare la posizione del supermercato su Google Maps (*Figura 7.3*).

MyShoppingList		
Email: marco.pedicillo@outlook.it		
Supermarket: Todis Casilina		
View on Google Maps		
Distance: 481.85 km		
Product Prices (Recommended Supermarket)		
Name	Quantity	Price (€)
Cacao Perugina	3	2.25
CremaNovi alla nocciola	1	4.25
Ciliegie	4	4.52
Arrosticini	2	8.18
Cotolette di Pollo Amadori	2	5.48
Arachidi Cameo	4	2.29
Birra Heineken	1	1.92
Coca Cola	1	1.96
Barette Kinder	1	3.22
Cornetti Tre Marie	1	3.05
Total Price:		75.71 €

Figura 7.3 PDF Checkout

Questa funzionalità è stata realizzata grazie alla libreria Javascript *jsPDF*. Quest'ultima è una libreria open-source che consente di creare file PDF personalizzati in modo dinamico direttamente nel browser, senza bisogno di elaborazioni lato server. Con *jsPDF* e Javascript, è possibile formattare e aggiungere vari tipi di contenuti, come testo, immagini e link ipertestuali, offrendo numerose opzioni per personalizzare il documento.

7.3 Lato server

L'algoritmo che gestisce la visualizzazione del checkout rappresenta la funzionalità principale a carico del server e costituisce il nucleo della piattaforma. Grazie a questo algoritmo, MyShoppinList permette all'utente di visualizzare il supermercato più vicino, il più economico e quello consigliato, combinando i criteri di distanza e risparmio economico in una logica di elaborazione integrata.

Al termine del processo, i dati relativi ai supermercati consigliato, più vicino e più economico vengono raccolti in un array associativo e inviati al browser del client. Quest'ultimo utilizza tali informazioni per completare il checkout e per generare le mappe che mostrano la posizione dei supermercati.

7.3.1 Supermercato più vicino

Quando il server riceve la richiesta di checkout da parte dell'utente, inizia recuperando la sua posizione dalla tabella *session_location* del database. Questa posizione viene poi utilizzata per determinare il supermercato più vicino. Per fare ciò, il server esegue una query sulla tabella *supermarkets*, calcolando la distanza tra le coordinate dell'utente e quelle dei vari supermercati. I risultati vengono ordinati in base alla distanza e viene selezionato il supermercato più vicino. Una volta individuato il supermercato, il server esegue una seconda query per calcolare il totale della spesa dell'utente (*Codice 7.1*).

```
$nearest_supermarket = "SELECT chain, name, ST_X(location) AS latitude, ST_Y(location) AS longitude,
                           ST_Distance_Sphere(location, POINT($user_latitude, $user_longitude))
                           AS distance FROM `supermarkets` ORDER BY distance LIMIT 1";
$result = $conn->query($nearest_supermarket);

$row = $result->fetch_assoc();
$nearest_supermarket_chain = $row['chain'];
$nearest_supermarket_name = $row['name'];
$nearest_supermarket_latitude = $row['latitude'];
$nearest_supermarket_longitude = $row['longitude'];
$nearest_supermarket_distance = $row['distance'];
$nearest_supermarket_distance = round($nearest_supermarket_distance / 1000, 2);
$nearest_supermarket_distance = strval($nearest_supermarket_distance) . " km";

$nearest_supermarket_totalPrice = "SELECT SUM(price * Quantità) AS totalPrice FROM Cart,
                                         supermarkets_products WHERE Cart.Nome = supermarkets_products.product_name
                                         AND supermarkets_products.supermarket_chain = '$nearest_supermarket_chain'
                                         AND supermarkets_products.supermarket_name = '$nearest_supermarket_name'";
$result = $conn->query($nearest_supermarket_totalPrice);
$row = $result->fetch_assoc();
$nearest_supermarket_totalPrice = $row['totalPrice'];
$nearest_supermarket_totalPrice = round($nearest_supermarket_totalPrice, 2);

$nearest_supermarket = [
    "chain" => $nearest_supermarket_chain,
    "name" => $nearest_supermarket_name,
    "latitude" => $nearest_supermarket_latitude,
    "longitude" => $nearest_supermarket_longitude,
    "distance" => $nearest_supermarket_distance,
    "totalPrice" => $nearest_supermarket_totalPrice
];

```

Codice 7.1 Individuazione supermercato più vicino

7.3.2 Supermercato più economico

Per determinare il supermercato più economico, a differenza di quello più vicino, il server inizia calcolando il totale della spesa per ciascun supermercato. Questo avviene tramite una query SQL che somma il costo dei prodotti presenti nel carrello dell'utente per ogni supermercato. La query esegue un join tra la tabella del carrello e quella dei prodotti dei supermercati, abbinando i prodotti nel carrello ai relativi prezzi nei diversi supermercati. Una volta ottenuto il totale per ogni supermercato, il sistema seleziona automaticamente quello più conveniente. Successivamente, una seconda query recupera le coordinate del supermercato scelto e calcola la distanza dalla posizione dell'utente in chilometri utilizzando la funzione SQL `ST_Distance_Sphere` (*Codice 7.2*).

```
$totalPrice = "SELECT SUM(price * Quantità) AS totalPrice, supermarket_chain, supermarket_name
    FROM Cart, supermarkets_products
    WHERE Cart.Nome = supermarkets_products.product_name
    GROUP BY supermarket_chain, supermarket_name";
$result = $conn->query($totalPrice);

$cheapestSupermarket = "SELECT supermarket_chain, supermarket_name, MIN(totalPrice)
    AS totalPrice FROM ($totalPrice) AS totalPrices
    GROUP BY supermarket_chain, supermarket_name
    ORDER BY totalPrice LIMIT 1";
$result = $conn->query($cheapestSupermarket);
$row = $result->fetch_assoc();

$cheapest_supermarket_chain = $row['supermarket_chain'];
$cheapest_supermarket_name = $row['supermarket_name'];
$cheapest_supermarket_totalPrice = $row['totalPrice'];
$cheapest_supermarket_totalPrice = round($cheapest_supermarket_totalPrice, 2);

$cheapest_supermarket_location = "SELECT ST_X(location) AS latitude, ST_Y(location) AS longitude
    FROM supermarkets WHERE chain = '$cheapest_supermarket_chain'
    AND name = '$cheapest_supermarket_name'";
$result = $conn->query($cheapest_supermarket_location);
$row = $result->fetch_assoc();
$cheapest_supermarket_latitude = $row['latitude'];
$cheapest_supermarket_longitude = $row['longitude'];

$cheapest_supermarket_distance = "SELECT ST_Distance_Sphere(location,
    POINT($user_latitude, $user_longitude)) AS distance
    FROM supermarkets WHERE chain = '$cheapest_supermarket_chain'
    AND name = '$cheapest_supermarket_name'";
$result = $conn->query($cheapest_supermarket_distance);
$row = $result->fetch_assoc();
$cheapest_supermarket_distance = $row['distance'];
$cheapest_supermarket_distance = round($cheapest_supermarket_distance / 1000, 2);
$cheapest_supermarket_distance = strval($cheapest_supermarket_distance) . " km";

$cheapest_supermarket = [
    "chain" => $cheapest_supermarket_chain,
    "name" => $cheapest_supermarket_name,
    "latitude" => $cheapest_supermarket_latitude,
    "longitude" => $cheapest_supermarket_longitude,
    "distance" => $cheapest_supermarket_distance,
    "totalPrice" => $cheapest_supermarket_totalPrice
];

```

Codice 7.2 Individuazione supermercato più economico

7.3.3 Supermercato consigliato

Infine, il server seleziona il supermercato consigliato, che rappresenta il miglior compromesso tra convenienza economica e vicinanza alla posizione dell'utente. Questo processo avviene in due fasi tramite query SQL: inizialmente, i supermercati vengono classificati in base alla distanza, e successivamente in base al prezzo, con l'assegnazione di un punteggio per ogni supermercato in entrambe le classifiche. Viene poi calcolata la somma dei punteggi relativi alla vicinanza e alla convenienza economica per ciascun supermercato. Il supermercato con il punteggio complessivo più basso è quello scelto come consigliato. Una volta selezionato, il server ricalcola la distanza dall'utente e il totale della spesa per quel supermercato (*Codice 7.3*).

```
$nearestSupermarketRanking = "SELECT chain, supermarkets.name, ST_Distance_Sphere(location,
                                         POINT($user_latitude, $user_longitude))
                                         AS distance FROM `supermarkets` ORDER BY distance";
$result = $conn->query($nearestSupermarketRanking);
$nearestSupermarketRankingArray = array();
$rank = 1;
while ($row = $result->fetch_assoc()) {
    $nearestSupermarketRankingArray[$row['chain']][$row['name']] = $rank;
    $rank++;
}

$cheapestSupermarketRanking = "SELECT supermarket_chain, supermarket_name, SUM(price * Quantità)
                                         AS totalPrice FROM Cart, supermarkets_products
                                         WHERE Cart.Nome = supermarkets_products.product_name
                                         GROUP BY supermarket_chain, supermarket_name ORDER BY totalPrice";
$result = $conn->query($cheapestSupermarketRanking);
$cheapestSupermarketRankingArray = array();
$rank = 1;
while ($row = $result->fetch_assoc()) {
    $cheapestSupermarketRankingArray[$row['supermarket_chain']][$row['supermarket_name']] = $rank;
    $rank++;
}

$recommendedSupermarketRankingArray = array();
foreach ($nearestSupermarketRankingArray as $chain => $supermarkets) {
    foreach ($supermarkets as $name => $nearestRank) {
        if (isset($cheapestSupermarketRankingArray[$chain][$name])) {
            $recommendedRank = $nearestRank + $cheapestSupermarketRankingArray[$chain][$name];
            $recommendedSupermarketRankingArray[$chain][$name] = $recommendedRank;
        }
    }
}

$recommended_supermarket_chain = null;
$recommended_supermarket_name = null;
$recommended_supermarket_rank = PHP_INT_MAX;
foreach ($recommendedSupermarketRankingArray as $chain => $supermarkets) {
    foreach ($supermarkets as $name => $rank) {
        if ($rank < $recommended_supermarket_rank) {
            $recommended_supermarket_chain = $chain;
            $recommended_supermarket_name = $name;
            $recommended_supermarket_rank = $rank;
        }
    }
}
```

Codice 7.3 Individuazione supermercato consigliato

Capitolo 8

Pagina preferiti

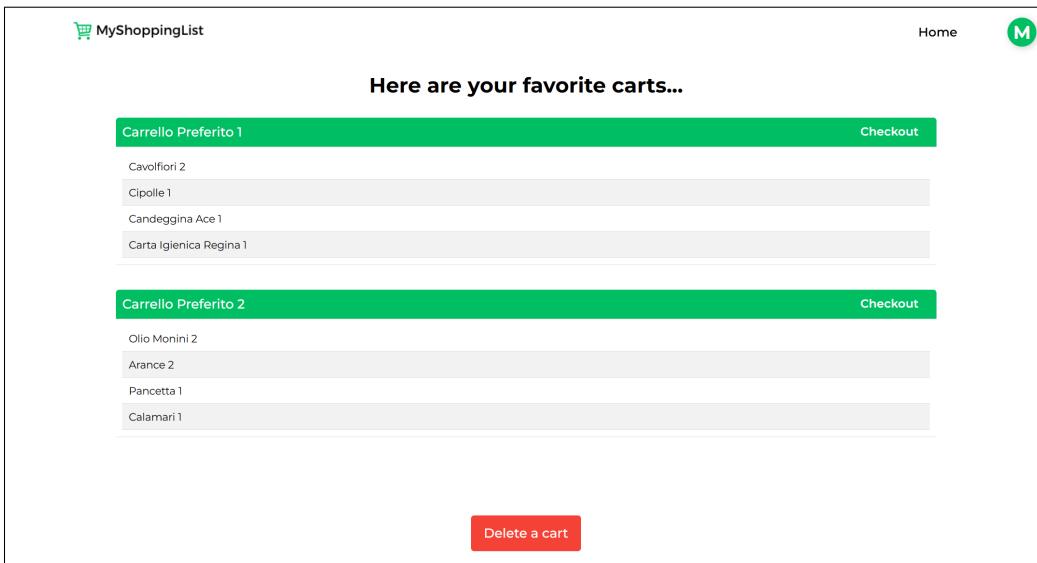


Figura 8.1 Pagina carrelli preferiti

Come descritto nel capitolo precedente, nel processo di checkout, gli utenti registrati hanno la possibilità di aggiungere un carrello ai preferiti tramite un pulsante a forma di stella. Basta cliccare su questo pulsante e inserire il nome desiderato per salvare il carrello. La pagina dei carrelli preferiti è una funzionalità esclusiva che consente agli utenti di conservare i carrelli contenenti i prodotti da acquistare in futuro, evitando di doverli selezionare nuovamente. Una volta salvato, il carrello può essere facilmente recuperato in qualsiasi momento attraverso la sezione apposita sulla barra di navigazione. Tramite il pulsante *Checkout* in alto a destra di ogni carrello preferito, l'utente può accedere direttamente al checkout per acquistare i prodotti salvati. Questa funzionalità è particolarmente utile per acquisti periodici, in quanto permette di ripetere rapidamente gli ordini precedenti senza dover rifare l'intero processo di selezione dei prodotti (*Figura 8.1*).

Per rimuovere un carrello preferito, è sufficiente cliccare sul pulsante rosso *Delete a cart* e inserire nel campo testuale il nome del carrello che si desidera eliminare.

8.1 Lato server

La gestione della pagina dei carrelli preferiti avviene come sempre tramite richieste AJAX, con PHP, sul lato server, che si occupa di elaborare e gestire queste richieste.

Nel *Codice 8.1* sono presenti due blocchi di codice PHP che gestiscono rispettivamente una richiesta **GET**, per ottenere i carrelli preferiti di un utente, e una richiesta **DELETE**, per rimuovere un carrello preferito:

- Quando il server riceve una richiesta **GET**, verifica se l'email dell'utente è stata fornita tramite il parametro *email*. Il server esegue una query SQL per selezionare tutte le righe della tabella *preferences* che corrispondono a quell'utente. La query restituisce i dati relativi ai carrelli. Questi dati vengono poi organizzati in un array associativo e inviati al client sotto forma di risposta JSON, permettendo così di visualizzare i carrelli preferiti dell'utente.
- Quando viene inviata una richiesta **DELETE**, il codice estrae i dati del carrello da eliminare tramite il corpo della richiesta. Il server quindi esegue una query SQL per eliminare il record corrispondente nella tabella *preferences*, utilizzando i parametri *email* e *name_cart*.

```

if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    if(isset($_GET['email']) && !isset($_GET['cart_name'])) {
        $email = $_GET['email'];
        $sql = "SELECT * FROM preferences WHERE user_id = '$email';";
        $result = $conn->query($sql);
        $preferences = array();
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $preferences[] = array(
                    'email' => $row['user_id'],
                    'name_cart' => $row['name_cart'],
                    'products' => $row['products']
                );
            }
        }
        header('Content-Type: application/json'); echo json_encode($preferences);
    }
}

if($_SERVER['REQUEST_METHOD']=='DELETE'){
    parse_str(file_get_contents("php://input"), $data);
    $name = $data['name'];
    $email= $data['email'];
    $stmt = $conn->prepare("DELETE FROM preferences WHERE user_id = ?
                            AND name_cart = ?");
    $stmt->bind_param("ss", $email, $name);
    if ($stmt->execute()) { echo "Prodotto eliminato con successo!";
    } else { echo "Errore: " . $stmt->error; }
}

```

Codice 8.1 GET e DELETE carrelli preferiti

Capitolo 9

Admin

L'amministratore dispone di funzionalità avanzate non accessibili agli utenti comuni, ma non ha accesso alle funzioni principali dedicate alla gestione dei carrelli della spesa, poiché il suo ruolo è focalizzato sull'amministrazione generale della piattaforma. Le sue responsabilità includono la gestione dei supermercati, dei prodotti, degli utenti registrati e delle recensioni di questi ultimi. Inoltre, l'Admin può aggiornare i link ai social network e i contatti presenti nel footer, assicurando che queste informazioni siano sempre accurate e aggiornate per le necessità del pubblico.

Per creare un account Admin, è sufficiente selezionare l'opzione *Admin?*: *Sì* nella pagina di registrazione. Dopo essersi registrato, l'utente viene reindirizzato alla nuova Home Page. Qui, invece del pulsante *Fill the cart now*, appaiono tre nuovi pulsanti che conducono ciascuno a una pagina dedicata (*Figura 9.1*).

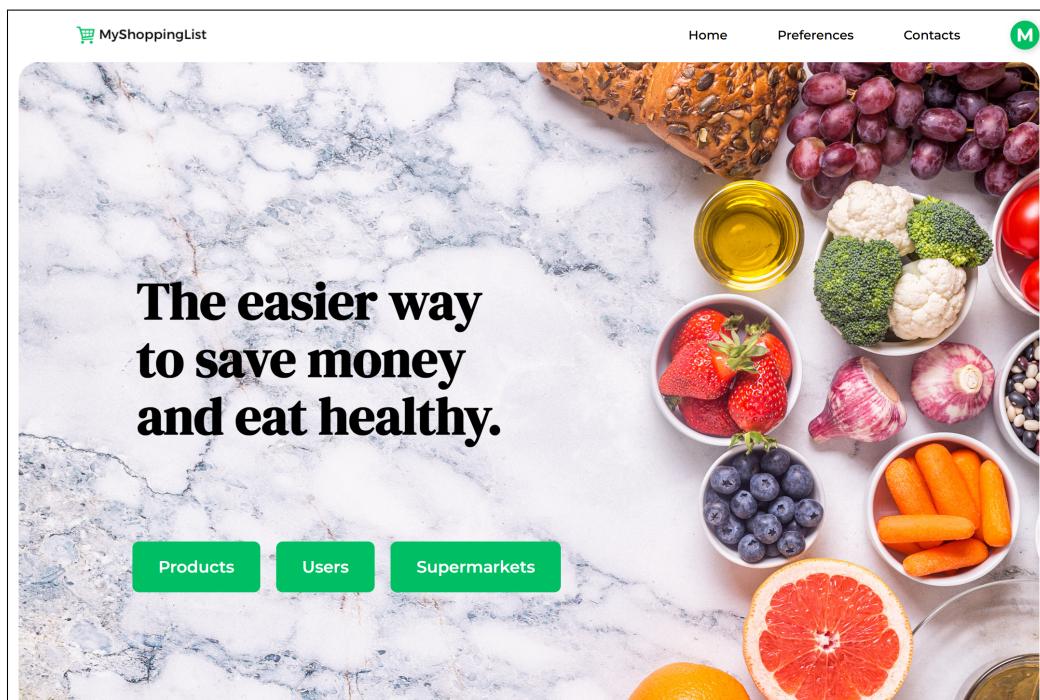


Figura 9.1 Home Page Admin

9.1 Supermercati

La pagina dei supermercati riservata all'Admin presenta un elenco scrollabile che mostra tutte le informazioni sui supermercati registrati, tra cui la catena, il nome e la posizione. Al termine della tabella, sono presenti tre pulsanti distinti per aggiungere, modificare o eliminare un supermercato (*Figura 9.2*):

- Per aggiungere un nuovo supermercato, si apre una finestra pop-up in cui inserire la catena, il nome, la latitudine e la longitudine del punto vendita.
- Per modificare i dati di un supermercato esistente, si apre una finestra simile in cui è possibile aggiornare catena, nome, latitudine e longitudine.
- Per rimuovere un supermercato, basta inserire la catena e il nome del supermercato da eliminare nella finestra pop-up che viene visualizzata.

Chain	Name	Location
Carrefour	Laurentina	41.855445 12.483314
Carrefour	Nomentana	41.919477 12.532757
Carrefour	Parioli	41.920283 12.492215
Carrefour	Ponte Milvio	41.927403 12.469468
Carrefour	Porta Maggiore	41.86628 12.527536
Carrefour	Selva Candida	41.950928 12.366331
Carrefour	Tiburtina	41.913002 12.514929
Carrefour	Tuscolana	41.869871 12.529141
Coop	Aventino	41.880237 12.484294
Coop	Eur	41.830397 12.470904
Coop	Garbatella	41.864568 12.479283
Coop	Grottarossa	41.981713 12.532292
Coop	Magliana	41.830877 12.429167
Coop	Monte Mario	41.935759 12.447064
Coop	Monteverde	41.885456 12.442292

Buttons at the bottom: Register a New Supermarket (green), Edit a Registered Supermarket (white), Delete a Registered Supermarket (red).

Figura 9.2 Pagina supermercati Admin

La possibilità di gestire i supermercati consente all'Admin di mantenere aggiornata la piattaforma, assicurando che gli utenti abbiano sempre accesso alle informazioni più accurate e rilevanti sui punti vendita disponibili.

9.1.1 Lato server

Nel *Codice 9.1* sono presenti due blocchi di codice PHP che gestiscono rispettivamente una richiesta **POST**, per l'aggiunta di un nuovo supermercato, e una richiesta **DELETE**, per la rimozione di uno esistente:

- Quando il server riceve la richiesta, estrae i dati dal corpo della richiesta JSON. Se l'attributo *add* è presente, vengono estratti i valori relativi alla catena, al nome, e alla posizione del supermercato. Se uno di questi campi risulta vuoto, viene restituito un messaggio di errore e la connessione al database viene chiusa. Altrimenti, il codice prepara un'istruzione SQL per inserire il supermercato nella tabella *supermarkets*, usando `ST_GeomFromText` per trasformare la posizione in un formato spaziale compatibile con il database. Il codice per la modifica è simile a questo, ma utilizza un'istruzione SQL `UPDATE`.
- Quando il server riceve la richiesta, estrae i valori di *chain* e *name* dal corpo della richiesta e prepara un'istruzione SQL per eliminare il supermercato corrispondente dalla tabella *supermarkets*. L'istruzione **DELETE** utilizza una dichiarazione parametrizzata per prevenire attacchi di SQL injection.

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $data = json_decode(file_get_contents('php://input'), true);
    if (isset($data['add']) && $data['add']) {
        $chain = $data['chain'];
        $name = $data['name'];
        $latitude = $data['latitude'];
        $longitude = $data['longitude'];
        if (empty($chain) || empty($name) || empty($latitude) || empty($longitude)) {
            echo "Errore: Tutti i campi sono obbligatori.";
            $conn->close();
            exit();
        }
        $location = "POINT($latitude $longitude)";
        $stmt = $conn->prepare("INSERT INTO supermarkets (chain, name, location)
                               VALUES (?, ?, ST_GeomFromText(?))");
        $stmt->bind_param("sss", $chain, $name, $location);
        if ($stmt->execute()) { echo "Supermercato aggiunto con successo!"; }
        else { echo "Errore: " . $stmt->error; }
    }
}

if ($_SERVER['REQUEST_METHOD'] == 'DELETE') {
    parse_str(file_get_contents("php://input"), $data);
    $chain = $data['chain'];
    $name = $data['name'];
    $stmt = $conn->prepare("DELETE FROM supermarkets WHERE chain = ? AND name = ?");
    $stmt->bind_param("ss", $chain, $name);
    if ($stmt->execute()) { echo "Recensione eliminata con successo!"; }
    else { echo "Errore: " . $stmt->error; }
}

```

Codice 9.1 Aggiunta e Eliminazione supermercato

9.2 Prodotti

La pagina dedicata alla gestione dei prodotti da parte dell'Admin mostra un elenco scorrevole con tutte le informazioni relative ai prodotti disponibili, comprese l'immagine, il nome, la categoria, il peso, una descrizione, i nutrienti e il valore Healthy. Alla fine della tabella, sono presenti tre pulsanti distinti che permettono di aggiungere, modificare o eliminare un prodotto (*Figura 9.3*):

- Per aggiungere un nuovo prodotto, si apre una finestra pop-up che consente di inserire tutti i dati precedentemente menzionati.
- Per modificare i dettagli di un prodotto esistente, si apre una finestra simile che permette di aggiornare tutte le informazioni.
- Per rimuovere un prodotto, è sufficiente digitare il nome del prodotto da eliminare nella finestra pop-up che appare.

MyShoppingList							Home	M
Image	Name	Category	Weight	Description	Nutrients	Health		
	Bistecca di Vitello	Carne	250	Bistecca di vitello succosa e pregiata, ideale per grigliate.	nutrienti da specificare	8		
	Broccoli	Vegetali	1000	Broccoli verdi e ricchi di antiossidanti, ottimi al vapore o in padella.	nutrienti da specificare	10		
	Brownie Milka	Dolci	1500	Tavoletta di cioccolato Milka, irresistibile e cremosa.	nutrienti da specificare	5		
	Bucatini Rummo	Dispensa	500	Mais Valfrutta, ideale per insalate e contorni.	nutrienti da specificare	7		
	Cacao Perugina	Dispensa	200	Marmellata di albicocca Zueg, perfetta per la colazione.	nutrienti da specificare	6.5		

[Register a New Product](#)
[Modify a Registered Product](#)
[Delete a Registered Product](#)

Figura 9.3 Pagina prodotti Admin

La gestione dei prodotti permette all'Admin di garantire un catalogo sempre aggiornato, offrendo agli utenti un'esperienza ottimale con informazioni precise e tempestive sui prodotti disponibili.

9.2.1 Lato server

Nel Codice 9.2 sono presenti due blocchi di codice PHP che gestiscono rispettivamente una richiesta **POST**, per l'aggiunta di un nuovo prodotto, e una richiesta **DELETE**, per la rimozione di uno esistente:

- Quando il server riceve la richiesta, estrae i dati dal corpo della richiesta JSON, quindi controlla la presenza del parametro *add* per confermare l'intenzione di aggiungere un prodotto. Successivamente, raccoglie i dati principali come nome, peso, categoria, descrizione, nutrienti, valore Healthy e immagine. Alcuni campi, come peso e descrizione, vengono impostati come NULL se lasciati vuoti. Infine, lo script prepara ed esegue una query SQL per inserire un nuovo record nella tabella *products*, assicurandosi che i dati siano ben strutturati e protetti da eventuali vulnerabilità tramite il metodo **prepare** di MySQL. Il codice per la modifica è simile a questo, ma utilizza un'istruzione SQL UPDATE.
- Il server decodifica il nome del prodotto da eliminare e viene utilizzata una query SQL predefinita per evitare SQL injection, grazie alla funzione **prepare**. Se la query va a buon fine, il sistema conferma l'eliminazione del prodotto, mentre eventuali errori vengono segnalati.

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $data = json_decode(file_get_contents('php://input'), true);
    if (isset($data['add']) && $data['add']) {
        $nome = $data['nome'];
        $peso = $data['peso'];
        $categoria = $data['categoria'];
        $descrizione = $data['descrizione'];
        $nutrients = $data['nutrients'];
        $health = $data['health'];
        $image = $data['image'];
        if (empty($nome)) {
            echo "Errore: Il campo nome è obbligatorio.";
            $conn→close();
            exit();
        }
        if (empty($peso)) { $peso=NULL; }
        if (empty($health)) { $health=NULL; }

        ...

        $stmt = $conn→prepare("INSERT INTO products (Nome, peso, image, category,
                                         description, nutrients, health) VALUES (?, ?, ?, ?, ?, ?, ?)");
        $stmt→bind_param("sssssss", $nome, $peso, $image, $categoria, $descrizione, $nutrients, $health);

        if ($stmt→execute()) { echo "Prodotto aggiunto con successo!"; }
        else { echo "Errore: " . $stmt→error; }
    }
}

if ($_SERVER['REQUEST_METHOD'] == 'DELETE') {
    parse_str(file_get_contents("php://input"), $data);
    $name = $data['name'];
    $stmt = $conn→prepare("DELETE FROM products WHERE Nome = ?");
    $stmt→bind_param("s", $name);

    if ($stmt→execute()) { echo "Prodotto eliminato con successo!"; }
    else { echo "Errore: " . $stmt→error; }
}

```

Codice 9.2 Aggiunta e Eliminazione prodotto

9.3 Utenti

La sezione di gestione utenti per l'Admin presenta una lista completa di tutti gli utenti registrati, esclusi gli Admin, includendo informazioni come email, numero di cellulare, stile di vita e budget impostato. Al termine della tabella, è disponibile un unico pulsante per l'eliminazione di un utente (*Figura 9.4*):

- Per rimuovere un utente è sufficiente inserire l'email dell'utente da eliminare nella finestra di pop-up che appare.

Email	Cell	Life	Budget
marco.pedicillo@outlook.it	3405595344	Sportivo	100
battioni.19866870@studenti.uniroma1.it	3317355120	Normale	25
ricciardelli.1983489@studenti.uniroma1.it	3391139821	Sportivo	30
corsale.1985903@studenti.uniroma1.it	3249099094	Sportivo	50

[Delete a Registered User](#)

Figura 9.4 Pagina utenti Admin

La gestione degli utenti permette all'Admin di mantenere un controllo puntuale e sicuro della comunità, garantendo agli iscritti un'esperienza sempre in linea con gli standard della piattaforma.

9.3.1 Lato server

Nel *Codice 9.3* è presente un blocco di codice PHP che gestisce una richiesta **DELETE**, per la rimozione di un utente non Admin esistente:

- Lo script estrae l'email dell'utente da eliminare dal corpo della richiesta e verifica che l'utente non sia un Admin, controllando se il campo *adm* è uguale a 0 o nullo. Successivamente, prepara ed esegue una query SQL per eliminare l'utente dalla tabella *utenti* in base all'indirizzo email specificato, utilizzando parametri per prevenire vulnerabilità di tipo SQL injection.

```

if ($_SERVER['REQUEST_METHOD'] == 'DELETE') {
    parse_str(file_get_contents("php://input"), $data);
    $email = $data['email'];

    $stmt = $conn->prepare("DELETE FROM utenti WHERE email = ? AND (adm LIKE 0 OR adm IS NULL)");
    $stmt->bind_param("s", $email);

    if ($stmt->execute()) { echo "Recensione eliminata con successo!"; }
    else { echo "Errore: " . $stmt->error; }
}

```

Codice 9.3 Eliminazione utente

9.4 Recensioni

Come già illustrato, un amministratore non può lasciare o modificare recensioni, ma ha il permesso di rimuovere una recensione specifica di un utente qualora la ritenga inappropriata. Per farlo, è sufficiente che inserisca l'email dell'utente di cui desidera eliminare la recensione nel campo di testo dopo aver premuto il pulsante *Delete*.

Facciamo riferimento al *Codice 4.2*; l'unica variazione riguarda la richiesta AJAX eseguita nel codice JavaScript. Per un utente che desidera cancellare una propria recensione, il controllo avviene tramite l'email registrata durante il login. Nel caso di un amministratore, invece, il controllo si basa sull'email inserita nel campo di testo e non su quella associata al suo account.

9.5 Footer

Un Admin ha la possibilità di aggiornare gli URL dei social network e i contatti come numero di telefono ed email, garantendo così che la piattaforma abbia sempre le informazioni più recenti e utili per gli utenti. Cliccando sul pulsante situato in alto a destra di ciascun social o accanto ai contatti, si aprirà automaticamente un box dove è possibile inserire il nuovo URL o contatto (*Figura 9.5*).

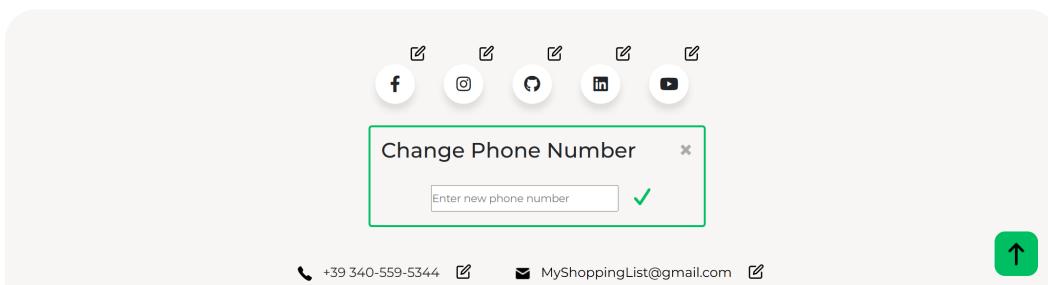


Figura 9.5 Footer Admin

Tutto questo processo viene gestito attraverso funzioni JavaScript, che consentono di aggiornare dinamicamente il contenuto HTML della pagina o di modificare i collegamenti ai profili social (*Codice 9.4*).

```
function openModalTel() {
    ...
    var saveBtn = document.getElementById('save-phone-number');
    saveBtn.onclick = function() {
        var newPhoneNumber = document.getElementById('new-phone-number').value;
        if (newPhoneNumber) {
            document.getElementById('tel').innerHTML = '' +
                newPhoneNumber;
            localStorage.setItem('phoneNumber', newPhoneNumber);
        }
    }
}
```

Codice 9.4 Script modifica contatto telefonico

Capitolo 10

Test

Il testing del software consiste in un'attività mirata a rilevare difetti legati alla correttezza, alla completezza e alla stabilità di un programma in fase di sviluppo. Questo processo si propone di garantire la qualità del software, identificando anomalie che si manifestano quando particolari dati di input, in combinazione con specifiche condizioni operative, provocano errori o comportamenti inattesi.

10.1 Tecnologie usate

Poiché abbiamo sviluppato l'intera piattaforma con JavaScript e PHP, ci siamo serviti di due strumenti per lo sviluppo dei test, *PHPUnit* e *Codeception*:

- PHPUnit è una libreria di testing unitario per PHP, progettata per verificare il comportamento di singole unità di codice, come classi o metodi. Consente agli sviluppatori di scrivere test automatizzati per confermare che le unità del software funzionino come previsto in isolamento, facilitando il rilevamento di bug e migliorando la qualità del codice.
- Codeception è un framework di testing più completo che supporta diverse tipologie di test, tra cui test unitari, funzionali e accettazione. Grazie alla sua flessibilità, permette di simulare scenari complessi, come interazioni tra utenti e applicazioni, verificando il comportamento del sistema nel suo complesso.

10.2 Test unitari

I test unitari si concentrano sulla verifica delle singole unità di codice, come metodi o classi, in isolamento dagli altri componenti del sistema. L'obiettivo principale di questi test è assicurare che ogni unità di codice funzioni esattamente come previsto, senza considerare le interazioni con altri componenti o il sistema complessivo.

Nel nostro caso questi test sono stati utilizzati per validare il comportamento delle classi principali, come *Product*, *User*, *Supermarket* e *Admin*, attraverso l'analisi delle loro proprietà e metodi.

Prendiamo in considerazione le classi relative ai prodotti e agli utenti (*Codice 10.1*). I test si sono concentrati sul metodo GET, verificando che i dati di entrambi le classi fossero gestiti correttamente. Ad esempio, è stato testato che il nome, il peso, la categoria, la descrizione e altri dettagli del prodotto fossero restituiti in modo corretto, come anche l'ID, l'email, il nome, il cognome, il telefono e lo stile di vita di un utente. I dati utilizzati per i test sono stati creati appositamente per simulare scenari realistici.

```
public function testProductGetters() {  
  
    $this→assertEquals("Aglio", $this→product→getName());  
    $this→assertEquals(1000, $this→product→getPeso());  
    $this→assertEquals("products/aglio.jpg", $this→product→getImage());  
    $this→assertEquals("Vegetali", $this→product→getCategory());  
    $this→assertEquals("Aglio utilizzato in molte preparazioni", $this→product→getDescription());  
    $this→assertEquals("nutrienti da specificare", $this→product→getNutrients());  
    $this→assertEquals(10, $this→product→getHealth());  
  
}  
  
public function testUserGetters() {  
  
    $this→assertEquals(1, $this→user→getId());  
    $this→assertEquals("user@example.com", $this→user→getEmail());  
    $this→assertEquals("password123", $this→user→getPassword());  
    $this→assertEquals("John", $this→user→getFirstName());  
    $this→assertEquals("Doe", $this→user→getLastName());  
    $this→assertEquals("1990-01-01", $this→user→getBirthDate());  
    $this→assertEquals("1234567890", $this→user→getPhoneNumber());  
    $this→assertEquals("Normale", $this→user→getLifestyle());  
    $this→assertEquals(50, $this→user→getBudget());  
  
}
```

Codice 10.1 Test unitari

10.3 Test funzionali

I test funzionali sono una tipologia di verifica del software che si concentra sul comportamento complessivo del sistema, assicurandosi che le funzionalità implementate rispettino i requisiti e forniscano risultati corretti. A differenza dei test unitari, che si occupano di singole unità di codice, i test funzionali valutano il sistema come un insieme, simulando scenari reali d'uso e interazioni tipiche da parte degli utenti.

Nel nostro caso, i test funzionali si sono focalizzati su alcune delle user stories essenziali, come la gestione dei carrelli e la ricerca dei prodotti attraverso i filtri delle categorie (*Codice 10.2*).

Tra i test eseguiti, è stata verificata la corretta aggiunta di un prodotto al carrello, assicurandosi che l'inserimento sia avvenuto con successo e che i dati siano stati salvati correttamente nel database. Inoltre, sono stati testati i metodi di ricerca dei prodotti per categoria, in particolare : la selezione di tutte le categorie e la categoria specifica *Healthy*. Per quest'ultima, si è verificato che i prodotti restituiti avessero un valore di salubrità non minore di 8.

```
public function testUserAddToCart() {
    $result = $this->user->addProductToCart($this->conn, $this->product);
    $this->assertStringContainsString("success", $result);

    $select_cart = "SELECT * FROM `Cart` WHERE Nome = 'Aglio'";
    $res = $this->conn->query($select_cart);
    $this->assertGreaterThan(0, $res->num_rows);
}

public function testGetProductsByCategoryAll() {
    $result = User::getProductsByCategory($this->conn, "All");
    $products = json_decode($result, true);

    $this->assertIsArray($products);
    $this->assertGreaterThanOrEqual(0, count($products));
}

public function testGetProductsByCategoryHealthy() {
    $result = User::getProductsByCategory($this->conn, "Healthy");
    $products = json_decode($result, true);

    $this->assertIsArray($products);
    foreach ($products as $product) {
        $this->assertGreaterThanOrEqual(8, $product['health']);
    }
}
```

Codice 10.2 Test funzionali

10.4 Test di accettazione

I test di accettazione, implementati utilizzando CodeCeption, consistono in una simulazione attraverso il browser di user stories complesse. Attraverso questi test possiamo vedere in diretta come risponde la piattaforma ai vari input da parte di un utente. I nostri test sono stati eseguiti su due user stories abbastanza grandi:

- **Gestione dei supermercati:** Dopo aver effettuato l'accesso con un account amministratore dalla Home Page, abbiamo simulato l'aggiunta, la modifica e l'eliminazione di un supermercato dalla relativa pagina (*Codice 10.3*). Questo test ha verificato come un utente Admin possa interagire con il sistema per gestire correttamente le informazioni sui supermercati.
- **Checkout:** Dopo aver effettuato l'accesso con un account utente registrato non Admin, è stato simulato l'intero processo, dalla selezione dei prodotti al checkout, includendo la modifica delle quantità e il salvataggio del carrello in formato PDF. È stata inoltre simulata la geolocalizzazione per ricreare uno scenario di utilizzo reale.

```
public function AggiuntaSupermercato (AcceptanceTester $I) {  
    $I→maximizeWindow(); $I→amOnPage('/home.html'); $I→wait(3);  
    $I→see('How do we help you?'); $I→click('Login'); $I→wait(2);  
    $I→fillField('campo1', 'pedicillo.1983285@studenti.uniroma1.it');  
    $I→fillField('campo2', 'password'); $I→click('accesso'); $I→wait(3);  
  
    $I→click('Home'); $I→wait(2);  
    $I→click('Supermarkets'); $I→wait(2);  
  
    $I→click('Register a New Supermarket');  
    $I→fillField('#Aggiungi_chainInput', 'Conad');  
    $I→fillField('#Aggiungi_nameInput', 'Tiburtina');  
    $I→fillField('#Aggiungi_latitudineInput', '45.123456');  
    $I→fillField('#Aggiungi_longitudineInput', '9.123456');  
    $I→click('#submitAggiungiSupermarkets'); $I→wait(2);  
  
    $I→click('Edit a Registered Supermarket');  
    $I→fillField('#Modifica_chainInput', 'Conad');  
    $I→fillField('#Modifica_nameInput', 'Tiburtina');  
    $I→fillField('#Modifica_latitudineInput', '40');  
    $I→fillField('#Modifica_longitudineInput', '40');  
    $I→click('#submitModificaSupermarkets'); $I→wait(2);  
  
    $I→click('Delete a Registered Supermarket');  
    $I→fillField('#Delete_chainInput', 'Conad');  
    $I→fillField('#Delete_nameInput', 'Tiburtina');  
    $I→click('#submitDeleteSupermarkets');  
}
```

Codice 10.3 Test di accettazione

Capitolo 11

Conclusioni

Lo sviluppo della piattaforma MyShoppingList è stato un percorso stimolante che ha permesso di mettere in pratica molte delle competenze teoriche acquisite durante il percorso di studi. Dalla progettazione di una base di dati alla realizzazione di un'interfaccia grafica intuitiva, ogni fase del lavoro ha rappresentato un'opportunità per consolidare le conoscenze in ambito informatico e progettuale. La piattaforma ha evidenziato l'importanza di un approccio metodico e iterativo nello sviluppo software, permettendo di affrontare con successo requisiti complessi come l'ottimizzazione del carrello della spesa basata su criteri di prezzo e vicinanza.

Uno degli aspetti più significativi è stata la possibilità di creare un'esperienza utente che combinasse funzionalità avanzate con un design semplice e intuitivo. Il progetto ha trasformato un'idea iniziale in una soluzione concreta per gli utenti, valorizzando l'impiego di tecnologie come PHP, JavaScript e API per garantire una piattaforma interattiva ed efficiente.

Lo sviluppo della piattaforma è stata anche un'importante esperienza di collaborazione tra colleghi. Lavorare in squadra ha permesso di condividere competenze, confrontare idee e trovare soluzioni a qualunque problema. Questa esperienza ha sottolineato l'importanza del lavoro di squadra, non solo per raggiungere gli obiettivi tecnici, ma anche per favorire la crescita personale e professionale di ciascun membro del team.

Nonostante il risultato raggiunto, MyShoppingList rappresenta un punto di partenza con molte possibilità di evoluzione. In futuro, il progetto potrebbe includere dati reali collegati ai database dei supermercati, l'integrazione di intelligenza artificiale per suggerimenti personalizzati e lo sviluppo di un'app mobile per ampliare l'accessibilità del servizio. Inoltre, potrebbe essere interessante introdurre una funzionalità di notifica che avvisi l'utente in tempo reale di offerte speciali o variazioni di prezzo nei supermercati vicini.

Il progetto MyShoppingList si propone, quindi, di rivoluzionare il modo di fare la spesa, rendendolo più organizzato e intelligente. Con ulteriori miglioramenti, la piattaforma potrebbe diventare uno strumento indispensabile per chi cerca di ottimizzare il proprio tempo e risparmiare.