

Elaborato Novel Alessio

Abstract in lingua inglese	2
Descrizione dell'applicazione	3
Definizione delle funzioni offerte dal sistema	4
Use case diagram:	4
Funzioni utente:	4
Funzioni utente backoffice	5
Funzioni supermercato:	6
Funzioni admin/dipendente	6
Funzioni admin:	6
Architettura del sistema e topologia di rete della soluzione proposta	8
Topologia logica e architettura della rete:	8
Architettura di sistema:	9
Descrizione di rete:	9
Tecnologie e protocolli di rete utilizzati	10
Protocolli	10
Tecnologie e linguaggi	10
Progettazione dell'applicazione	11
Descrizione progettazione	11
Descrizione dati:	11
Funzioni utente (sito web):	13
Funzioni backoffice (sito web):	18
Funzioni utente (applicazione smartphone):	22
Web services:	27
Sviluppo Prototipo	36

1 Abstract in lingua inglese

My idea came up by thinking, "What can be useful to the majority of people?". The answer is the most obvious of all, and my idea is based on exactly that...money. Everyone wants to spend as little money as possible, so I wanted to offer a service that would allow people to save both time and money when they go to the supermarket.

To do this I made a website and an application, on this website users have to create an account, and after they can put in their shopping list the specific product they want to buy, this is because each product has a unique barcode.

So after adding all the products to the cart, the system shows to the user which supermarkets have the best price, the best rating and are closest.

In this way, a person who has to do the supermarket shopping, discovers immediately if a supermarket does not have all the products he needs, and then can understand, before leaving home, which supermarket is more suitable for his needs.

The project, however, to work requires that all supermarkets register and enter what products they sell and at what price.

2 Descrizione dell'applicazione

Il progetto è stato pensato per aiutare le persone che devono fare la spesa al supermercato, infatti il programma dovrà, in base ai prodotti richiesti dall'utente, mostrare all'utente i supermercati più economici, più vicini e meglio recensiti.

Per fare ciò l'applicazione viene divisa in tre utilizzatori principali:

- **Il supermercato**, il quale dovrà registrarsi con i dati del supermercato, e dopo potrà gestire tutti i prodotti che fornisce, potrà aggiungere, modificare o eliminare i prodotti.
I prodotti dovranno essere identificati tramite l'identificativo situato sotto al codice a barre, così da rendere univoco e non ridondante ogni prodotto.
- **L'utente**, il quale, dopo essersi registrato e fatto il login, potrà creare la sua lista della spesa, specificando esattamente i prodotti che intende comprare.
- **Il dipendente**, il quale eseguirà le principali funzioni di amministrazione.

Il ruolo dell'applicazione sarà quindi quello di confrontare i prodotti aggiunti nella lista della spesa dell'utente con i prodotti registrati nei supermercati, per poter verificare in quali supermercati siano presenti tutti i prodotti richiesti, e poi mostrare la lista dei supermercati più vicini, più convenienti e più rilevanti che vendono i prodotti richiesti.

L'utente quindi potrà registrarsi e loggarsi, aggiungere dei prodotti nel suo carrello, confrontare i vari supermercati (per distanza, per prezzo e per valutazione) e valutare ogni supermercato.

Il supermercato potrà fare una richiesta per entrare nel servizio, e dopo dovrà indicare tutti i prodotti che vende e a quale prezzo, così da poter rendere possibile poi il confronto per gli utenti.

Il dipendente potrà accettare le richieste dei supermercati.

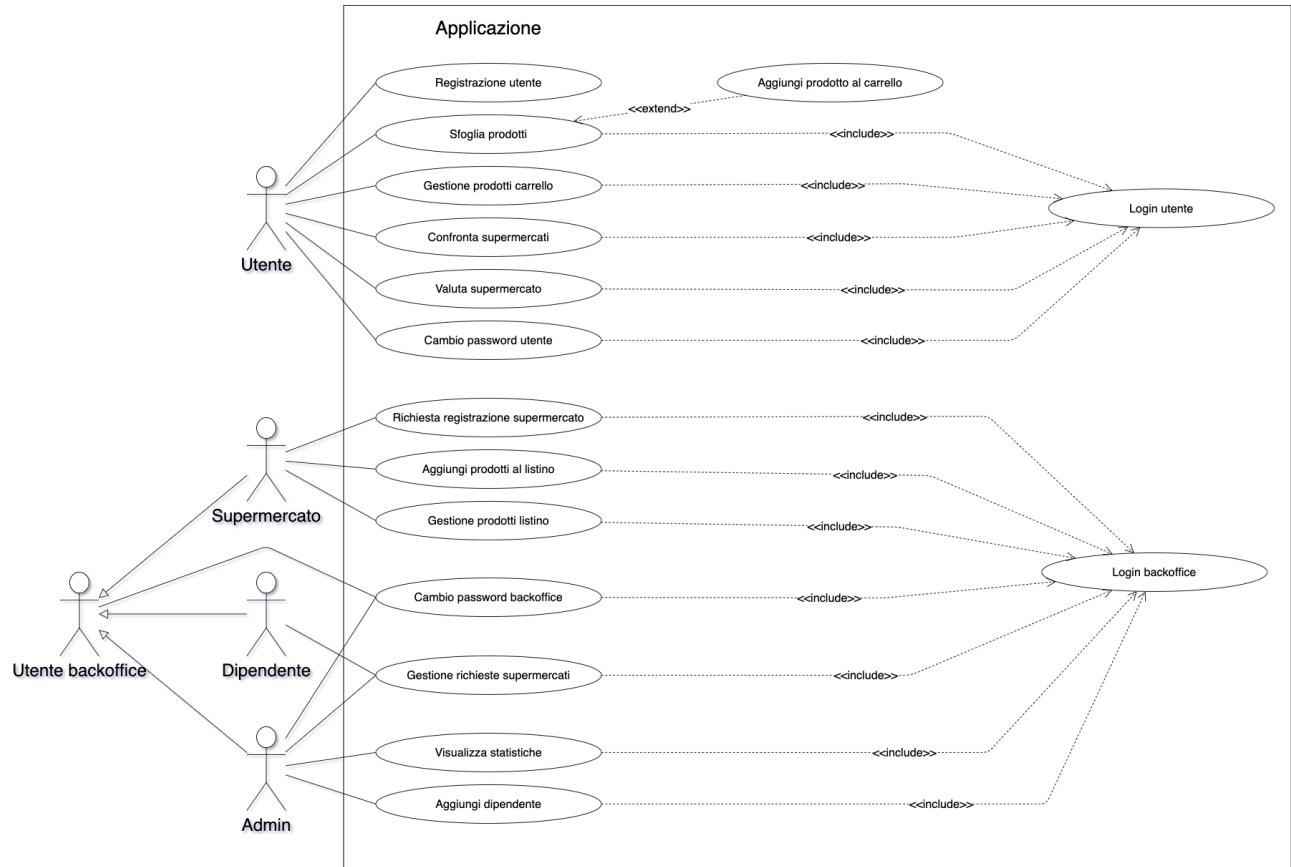
L'admin (che sarà il dipendente principale) potrà, oltre alle funzioni del dipendente, aggiungere un nuovo dipendente e visualizzare le statistiche dei prodotti (così da poterle vendere a chi interessato).

Si pensa di avere un'applicazione smartphone, la quale svolgerà solo le funzioni dell'utente, e il sito web, in cui ci saranno tutte le funzioni.

Il sito web sarà diviso in frontend (per gli utenti) e backoffice (per dipendenti e supermercati).

3 Definizione delle funzioni offerte dal sistema

Use case diagram:



Le funzioni principali offerte dal sistema sono le seguenti:

Funzioni utente:

Registrazione utente:

Il sistema chiede all'utente di inserire i suoi dati (nome, cognome, email e password), nel momento in cui l'utente clicca “registri” verrà mostrato in output un messaggio di errore se l'email è già stata utilizzata, altrimenti verrà mostrato un messaggio di conferma della registrazione.

Login utente:

Il sistema chiede all'utente di inserire i suoi dati: email e password. Sarà possibile cliccare su un tasto che renderà la password visibile. Se le credenziali sono corrette l'utente accede alla home page, altrimenti viene mostrato un messaggio di credenziali errate.

Aggiungi prodotto al carrello:

Il sistema permette all'utente di inserire un prodotto nel suo carrello, scegliendone la quantità.

Sfoglia prodotti:

Il sistema permette all'utente di digitare il nome di un prodotto nella barra di ricerca, al click del tasto invio i prodotti vengono cercati e viene mostrata una lista di prodotti filtrata per quel nome oppure il sistema mostra all'utente i prodotti più cliccati delle ultime due settimane.

In output per ogni prodotto verrà visualizzato nome, immagine e numero di rilevanza (numero di click su quei prodotti nelle ultime due settimane).

Gestione prodotti carrello:

Il sistema mostra all'utente tutti i prodotti presenti nel suo carrello, consentendo di svuotarlo (nel caso in cui voglia fare un nuovo carrello).

In output per ogni prodotto verrà visualizzato nome, immagine e quantità.

Poi per ogni singolo prodotto presente nel carrello, il sistema consentirà all'utente di eliminarlo o di modificarne la quantità.

Confronta supermercati:

Il sistema consente, successivamente all'aver aggiunto i prodotti richiesti nel suo carrello, di mostrare i supermercati che vendono tutti i prodotti richiesti, mostrando in output la distanza, il prezzo e valutazione.

L'utente potrà decidere secondo quale criterio, tra i tre elencati prima, ordinare i supermercati.

Per ogni supermercato verrà inoltre mostrato in output anche il nome e l'indirizzo.

Valuta supermercato:

Il sistema consente all'utente di dare una valutazione al supermercato da 1 a 10 (potrà dare un voto basso nel caso in cui il supermercato abbia i prezzi discordanti tra realtà e applicazione).

Cambio password utente:

Il sistema consente all'utente di inserire la vecchia password, e di inserire due volte la nuova password, se tutti i dati corrispondono la password dell'account verrà cambiata.

Funzioni utente backoffice

Cambia password backoffice:

Il sistema consente all'utente del backoffice di inserire la vecchia password, e di inserire due volte la nuova password, se tutti i dati corrispondono la password dell'account verrà cambiata.

Login backoffice:

Il sistema chiede all'utente di inserire i suoi dati: email e password.
Sarà possibile cliccare su un tasto che renderà la password visibile.
Se le credenziali sono corrette l'utente accede alla sua home page (che cambia in base a se l'utente è un supermercato, un dipendente o l'admin), altrimenti viene mostrato un messaggio di credenziali errate.

Funzioni supermercato:

Richiesta registrazione supermercato:

Il sistema consente ai supermercati che vogliono partecipare al servizio di eseguire una richiesta di registrazione, fornendo i propri dati.
Una volta che la richiesta viene inviata, il supermercato deve attendere che la richiesta venga accettata per poter iniziare a registrare i propri prodotti.

Aggiungi prodotto al listino:

Il sistema consente al supermercato di inserire il codice del prodotto (quello sotto al codice a barre su ogni prodotto) e il prezzo al quale quel supermercato lo vende. Se tutto va a buon fine il prodotto verrà aggiunto al listino.

Gestione prodotti del listino:

Il sistema mostra in output tutti i prodotti del supermercato.
Per ogni prodotto verrà visualizzato il nome, l'immagine e il prezzo.
Il sistema permette al supermercato, per ogni singolo prodotto presente nel listino, di eliminarlo o di modificarne il prezzo.

Funzioni admin/dipendente

Gestione richieste supermercato:

Il sistema permette al dipendente di visualizzare tutte le richieste di registrazione dei vari supermercati, sarà possibile filtrare le richieste per provincia (così ogni dipendente può visualizzare i supermercati su cui è incaricato di fare i controlli).
Il sistema permette al dipendente, per ogni singolo supermercato che ha eseguito una richiesta, di accettarlo nel servizio oppure rifiutarlo (se dai controlli risulta insistente per esempio).

Funzioni admin:

Visualizza statistiche:

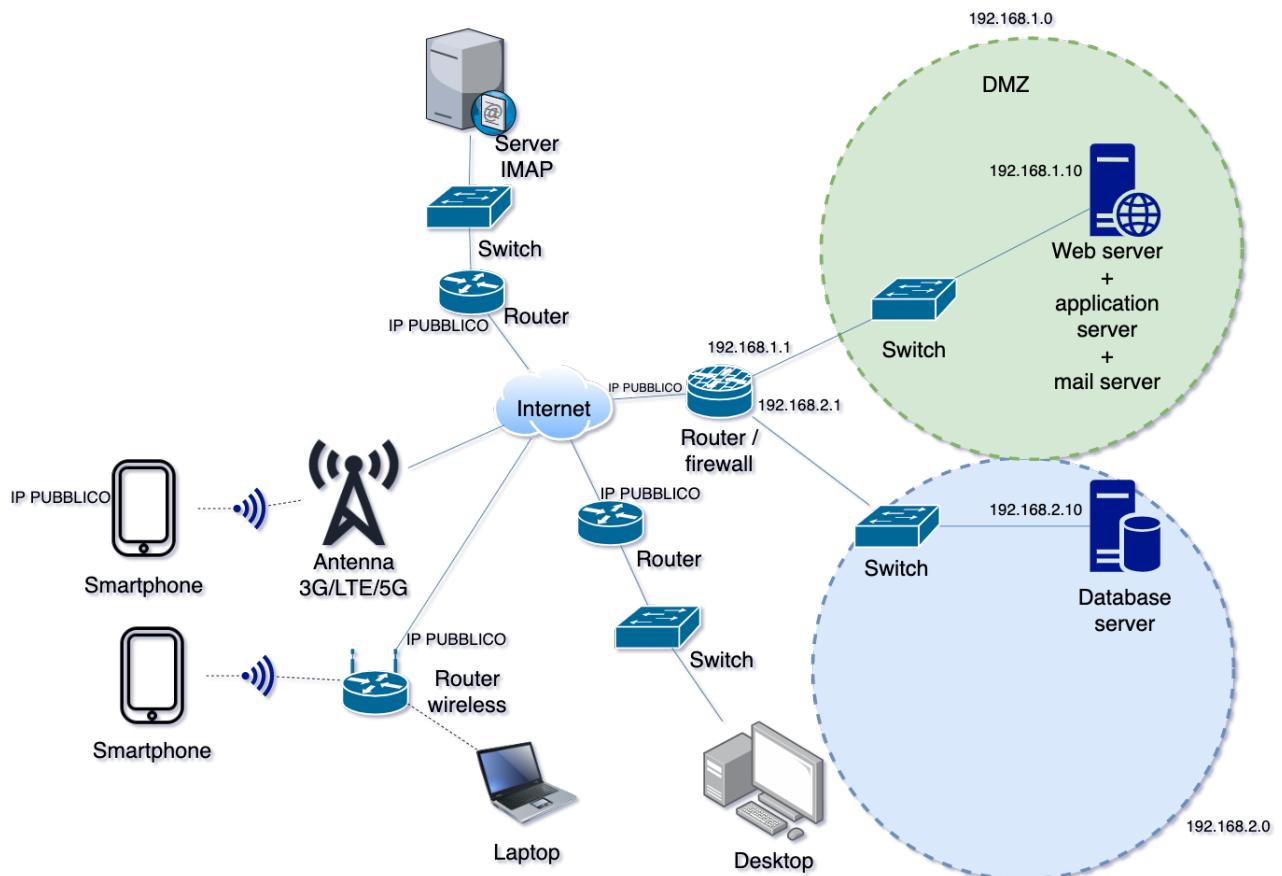
Il sistema permette all'admin di selezionare una data di inizio e una data di fine.
Quando effettuerà la ricerca verrà mostrato in output tutti i prodotti cliccati in quell'arco di tempo con il loro numero di rilevanza (numero di click).

Aggiungi dipendente:

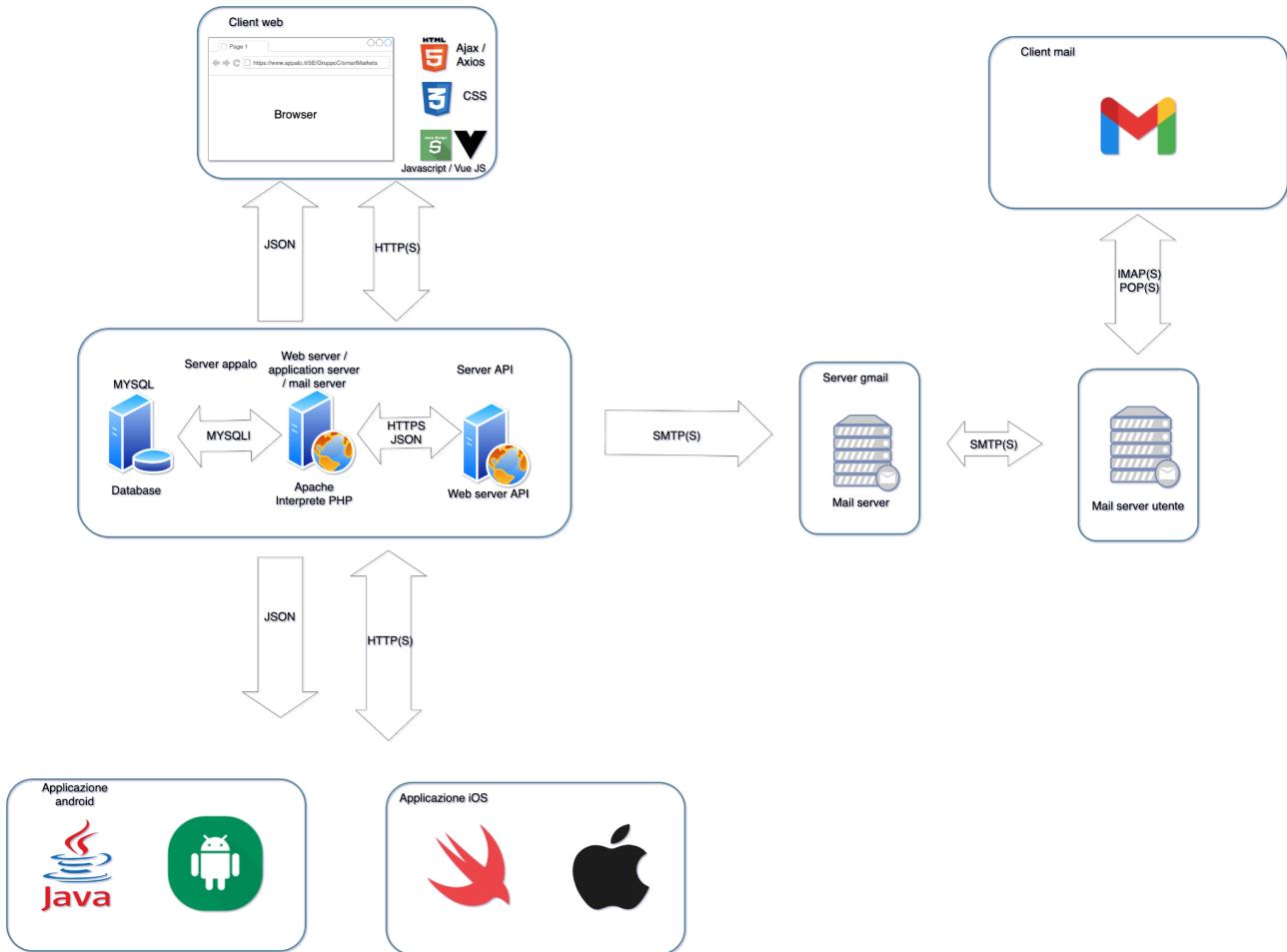
Il sistema permette all'admin di inserire i dati appena assume un nuovo dipendente, per poi inviarglieli tramite email.

4 Architettura del sistema e topologia di rete della soluzione proposta

Topologia logica e architettura della rete:



Architettura di sistema:



Descrizione di rete:

Il sistema è progettato avendo a disposizione un server web, il quale eroga le pagine web, con all'interno installato apache, insieme ad un application server, l'interprete PHP.

Il server web dovrà collegarsi al database server per accedere ai vari dati salvati tramite MYSQLI.

Il sito web chiederà al server web le pagine html con il protocollo HTTP(S).

Il sito web e le applicazioni smartphone chiameranno i web service, presenti nel server web, per ottenere e inviare dati in JSON.

Ci sarà anche un server web esterno, il quale alle chiamate API risponderà in JSON con i dati richiesti.

Visto che è pensato l'invio delle mail in determinate funzioni, è prevista la presenza di un server mail, il quale comunicherà con un server SMTP esterno.

5 Tecnologie e protocolli di rete utilizzati

Protocolli

- *Livello applicazione:*

- HTTP(S); utilizzato per la condivisione delle pagine web per il sito e per le chiamate ai web services, con l'implementazione dei protocolli TLS/SSL
- SMTP; utilizzato per l'invio delle mail come notifiche

- *Livello trasporto:*

- TCP; per instaurare la connessione tra client e server per l'erogazione di pagine web e per i web services

- *Livello di rete:*

- IPv4; per indirizzamento e instradamento dei pacchetti

- *Livello fisico:*

- Ethernet; per il collegamento dei server al router (e per l'eventualità in cui venga utilizzato il sito tramite una connessione cablata)
- Wi-Fi; per connettersi all'applicazione e al sito in modalità wireless
- 3G,LTE,5G; per connettersi all'applicazione con le reti telefoniche

Tecnologie e linguaggi

- HTML; per la scrittura e il contenuto delle pagine del sito web
- CSS; per la parte grafica del sito web
- PHP; per utilizzare le sessioni dell'utente e per la scrittura dei web service, e per invio di mail e funzione di hash per le password
 - MYSQLI; per il dialogo col database
- JavaScript; per modificare la pagina una volta caricata e per renderla dinamica
 - Vue JS; per facilitare certi comandi di javascript
 - Axios; per fare chiamate asincrone alle api
 - Ajax; per fare chiamate asincrone ai web services
 - jQuery; è una libreria javascript che semplifica l'utilizzo di la gestione degli eventi
- JSON; come formato per inviare e leggere i dati
- Java; per lo sviluppo dell'applicazione Android
- XML; per la parte grafica dell'applicazione Android
- SwiftUI; per lo sviluppo dell'applicazione iOS
- Apache+interprete PHP; apache per erogazione di pagine statiche, interprete PHP si occupa di elaborare il codice presente nelle pagine dinamiche
- MySQL; il software DBMS che è installato sul database server

6 Progettazione dell'applicazione

Descrizione progettazione

L'applicazione deve essere scritta per due sistemi diversi: un sito web e un applicazione smartphone.

Questo fa sì che venga scelto l'uso dei web services, i quali, chiamati con i giusti parametri, leggeranno o modificheranno i dati dal database e li ritorneranno i dati in formato JSON.

La parte di backoffice, pur essendo pensata per girare su un solo sistema (il sito web), verrà sviluppata utilizzando comunque i web services, per poter permettere un futuro aggiornamento facilmente implementabile, avendo i web services già pronti.

Descrizione dati:

Descrizione delle entità:

Utente: è la persona che vuole utilizzare il servizio, potrà aggiungere prodotti alla suo carrello e potrà valutare i supermercati.

Del utente si vuole sapere nome, cognome, email e password.

Supermercato: è il supermercato che si registra al progetto e vende determinati prodotti. Del supermercato si vuole sapere nome, indirizzo, latitudine, longitudine e provincia, in più anche email e password come credenziali per accedere alla sua area.

Dipendente: è la persona che si occupa di gestire le richieste di registrazione dei supermercati.

Del dipendente si vuole sapere nome, cognome, codice fiscale, email e password.

Admin: è la persona che si occupa di gestire le richieste di registrazione dei supermercati, di aggiungere dipendenti e di visualizzare le statistiche dei prodotti.

Del dipendente si vuole sapere nome, cognome, codice fiscale, email e password.

Prodotto della lista: è il prodotto che l'utente aggiunge alla sua lista della spesa.

Del prodotto si vuole sapere il codice e la quantità.

Prodotto rilevante: è il prodotto che viene cliccato da un utente.

Del prodotto si vuole sapere la data e il codice.

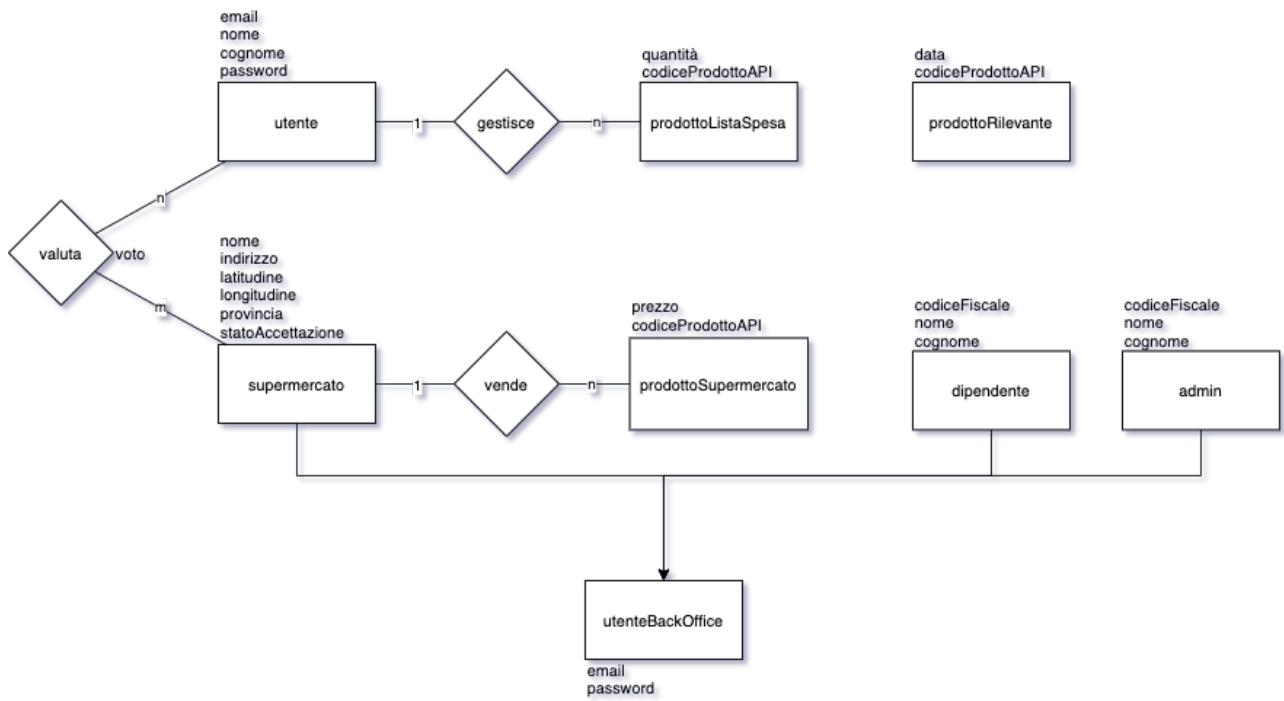
Prodotto supermercato: è il prodotto che il supermercato aggiunge al suo listino.

Del prodotto si vuole sapere il codice e il prezzo.

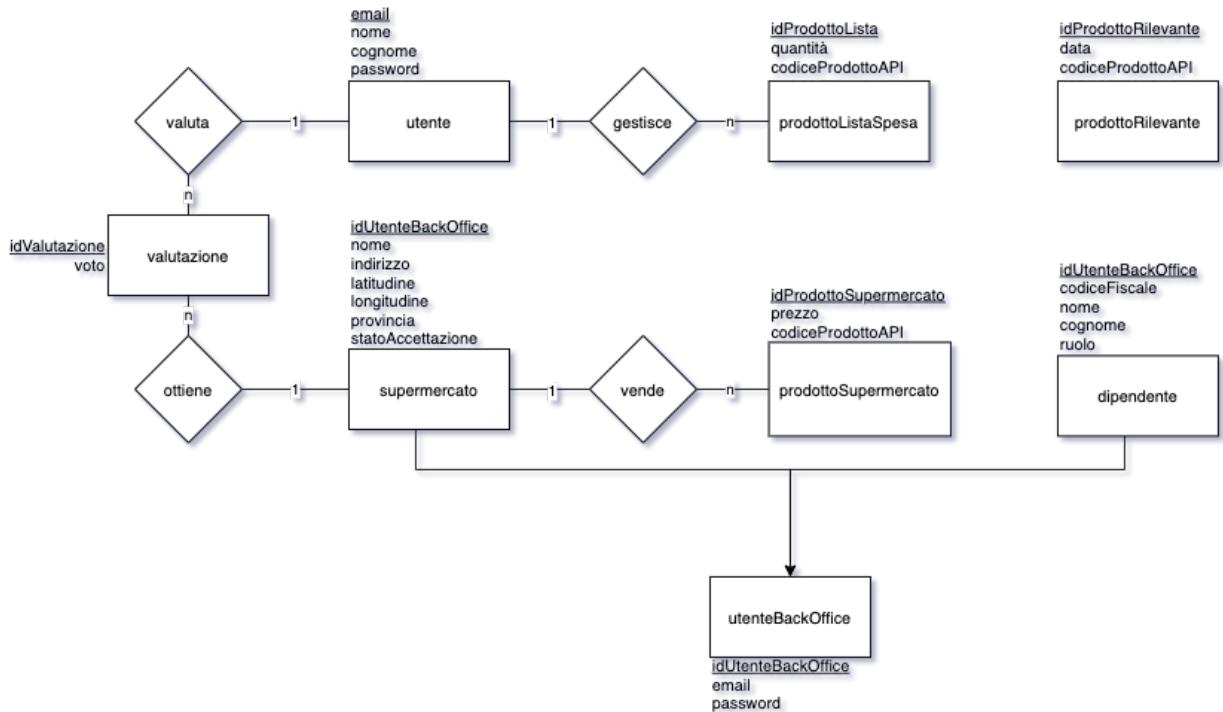
Valutazione: è il voto che l'utente dà a un supermercato.

Del valutazione si vuole sapere il voto.

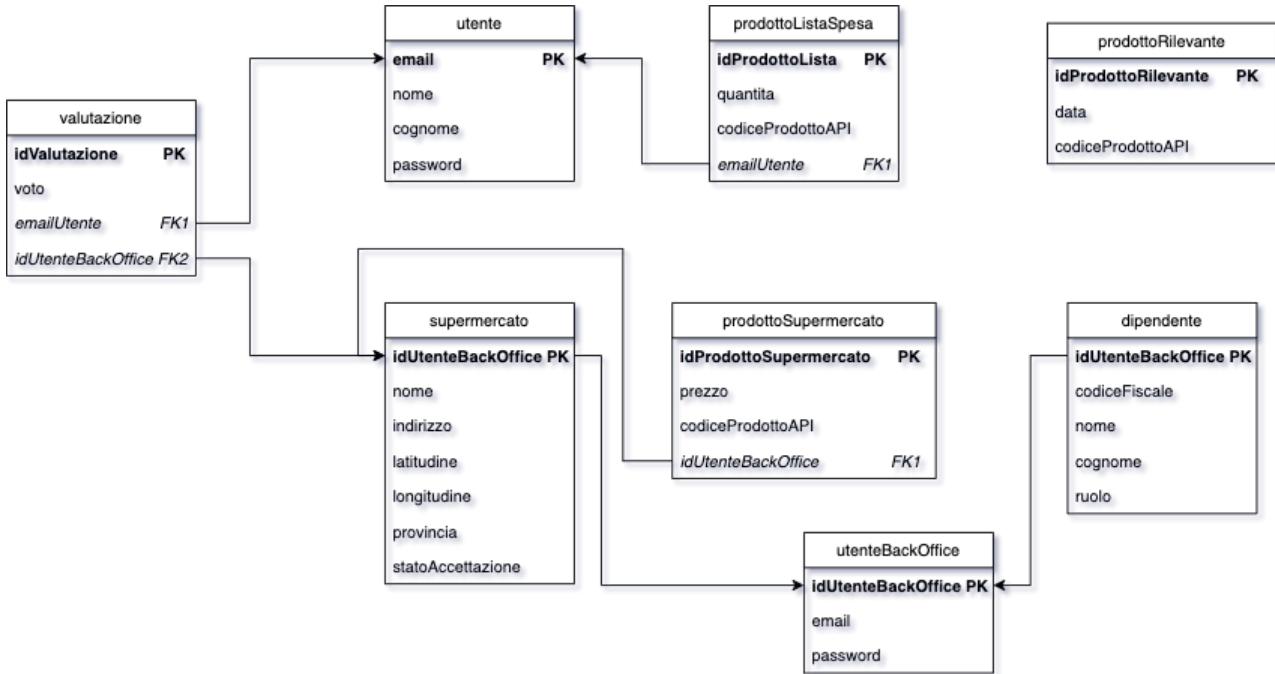
Schema concettuale:



Schema concettuale-ristrutturato:



Schema logico-relazionale:



Funzioni utente (sito web):

- *Login*

Questa pagina, la quale sarà necessaria per accedere a tutte le altre funzionalità (esclusa [Registrazione](#)), dovrà presentare due input di testo, uno per la email e uno per la password, un bottone per la visualizzazione della password, il quale al click mostrerà/censurerà la password, e un tasto “Accedi”, ci sarà anche un tasto che reinderizzerà alla [Registrazione](#).

Al click del tasto “Accedi” verrà controllato che nei due input ci sia presente del testo (in caso contrario verrà mostrato un errore), e verrà effettuata una chiamata asincrona al web service [checkLogin_utente.php](#):

- Dati da inviare: *email, password*
- Dati da ricevere: *risultato, email*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (credenziali corrette); l'email viene salvata nella sessione e l'utente viene reindirizzato alla home page
- *Risultato* = 1 (credenziali errate); viene mostrato in output il messaggio di errore “Credenziali errate”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

- *Registrazione*

Questa pagina dovrà presentare quattro input di testo, uno per la email, uno per la password, uno per il nome e l'ultimo per il cognome, e un tasto “Registrati”, ci sarà anche un tasto che reinderizzerà al [Login](#).

Al click del tasto “Registrati” verrà controllato che nei quattro input ci sia presente del testo (in caso contrario verrà mostrato un errore), e verrà effettuata una chiamata asincrona al web service [checkRegistrazione_utente.php](#):

- Dati da inviare: *email, password, nome, cognome*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (registrazione effettuata); viene mostrato in output il messaggio “Registrazione effettuata”
- *Risultato* = 1 (email già utilizzata); viene mostrato in output il messaggio di errore “Email già utilizzata”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Home*

Questa pagina dovrà presentare un messaggio di benvenuto, con una spiegazione generale di cosa può fare il sito. In questa pagina sarà presente il menù, il quale avrà i buttoni per andare in **Home**, **Ricerca prodotto**, **Visualizza carrello**, **Valuta supermercato**. Sarà anche possibile andare sulla pagina **Cambia password**.

● *Cambia password*

Questa pagina dovrà presentare tre input di testo, uno per la vecchia password, uno per la nuova password e uno per la conferma della nuova password, e un tasto “Cambia password”.

Al click del tasto “Cambia password” verrà controllato che nei tre input ci sia presente del testo e che le due nuove password corrispondano (in caso contrario verrà mostrato un errore), e verrà effettuata una chiamata asincrona al web service **changePassword_utente.php**:

- Dati da inviare: *vecchiaPassword, nuovaPassword, emailUtente*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (password cambiata); viene mostrato in output il messaggio “Password cambiata”
- *Risultato* = 1 (password sbagliata); viene mostrato in output il messaggio di errore “Vecchia password non corrisponde”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Ricerca prodotto*

Questa pagina presenterà una barra di ricerca e un bottone “Mostra prodotti rilevanti”, il quale al click chiamerà **Mostra prodotti rilevanti**.

Appena l’utente inserirà del testo nella barra di ricerca e schiacerà il tasto invio verrà effettuata una chiamata asincrona al servizio API [“https://api.upcitemdb.com/prod/trial/search”](https://api.upcitemdb.com/prod/trial/search) passando come parametro il *testo* presente nella barra di ricerca (guardare documentazione sul sito):

- Dati da inviare: *s* (stringa di ricerca)
- Dati da ricevere: *response, data*

Quindi, in assenza di errori, verranno mostrati i prodotti (contenuti in `data.items[]`). Altrimenti mostrato un messaggio di errore (`response.status`)
Per ogni prodotto sarà mostrato il nome e l'immagine e si potrà sceglierne la quantità.
A questo punto sarà possibile per ogni prodotto aggiungerlo al carrello (vedi [Aggiungi prodotto al carrello](#)).

● *Mostra prodotti rilevanti*

Questa pagina, al caricamento, effettuerà una chiamata asincrona al web service [getProdottiRilevanti_utente.php](#):

- Dati da inviare:
- Dati da ricevere: `risultato, prodotti[codiceProdotto, rilevanza, nome, immagine]`

Le azioni successive dipenderanno dal valore di `risultato`:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output la lista dei prodotti rilevanti
- Risultato = 1 (nessun risultato); viene mostrato in output “nessun prodotto rilevante al momento”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

Per ogni prodotto sarà mostrato il nome, il numero di rilevanza e l'immagine e si potrà sceglierne la quantità.

A questo punto sarà possibile per ogni prodotto aggiungerlo al carrello (vedi [Aggiungi prodotto al carrello](#)).

● *Aggiungi prodotto al carrello*

Questa funzione, chiamata al click del tasto “+” situato su ogni prodotto nelle pagine [Mostra prodotti rilevanti](#) e [Ricerca prodotto](#), controllerà che il valore di input di quantità sia valido e effettuerà una chiamata asincrona al web service [addProdotto_utente.php](#):

- Dati da inviare: `codiceProdotto, quantità, email`
- Dati da ricevere: `risultato`

Le azioni successive dipenderanno dal valore di `risultato`:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Visualizza carrello*

Questa pagina effettuerà al caricamento una chiamata asincrona al web service [getCarrello_utente.php](#):

- Dati da inviare: `emailUtente`

- Dati da ricevere: *risultato, prodotti[idProdottoLista, quantità, nome, immagine]*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output la lista dei prodotti nella lista
- Risultato = 1 (carrello vuoto); viene mostrato in output il messaggio “nessun prodotto nel carrello”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

A questo punto sarà possibile per ogni prodotto modificarne la quantità (vedi [Modifica quantità prodotto carrello](#)) o eliminarlo dal carrello (vedi [Elimina prodotto carrello](#)).

Ci sarà anche un tasto “svuota carrello” il quale chiamerà il web service [deleteCarrello.php](#):

- Dati da inviare: *emailUtente*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

• *Modifica quantità prodotto carrello*

Questa funzione, chiamata al click del tasto “modifica” situato su ogni prodotto nella pagina [Visualizza carrello](#), controllerà che il valore di input di quantità sia valido e effettuerà una chiamata asincrona al web service [changeProdottoCarrello_utente.php](#):

- Dati da inviare: *idProdottoLista, quantita*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

• *Elimina prodotto carrello*

Questa funzione, chiamata al click del tasto “elimina” situato su ogni prodotto nella pagina [Visualizza carrello](#), effettuerà una chiamata asincrona al web service [deleteProdottoCarrello_utente.php](#):

- Dati da inviare: *idProdottoLista*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Elimina tutto il carrello*

Questa funzione, chiamata al click del tasto “svuota carrello” situato nella pagina [Visualizza carrello](#), effettuerà una chiamata asincrona al web service [deleteCarrello.php](#):

- Dati da inviare: *emailUtente*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Mostra supermercati*

Questa pagina effettuerà al caricamento una chiamata asincrona al web service [getSupermercati_utente.php](#):

- Dati da inviare: *emailUtente, posizione, tipo(=prezzo)*
- Dati da ricevere: *risultato, supermercati[nome, indirizzo, prezzo, distanza, valutazione]*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output la lista dei supermercati
- Risultato = 1 (nessun supermercato trovato); viene mostrato in output il messaggio “nessun supermercato trovato”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

Sarà possibile ora cliccare sui bottoni “prezzo” “distanza” “valutazione”, i quali richiameranno lo stesso web service con gli stessi dati, ad eccezione di tipo, che cambierà in base all’ordinamento richiesto.

● *Ricerca supermercato*

Questa pagina, necessaria per accedere alla funzione [Valuta supermercato](#), dovrà presentare due input di testo, uno per il nome del supermercato e uno per la provincia, e un tasto “Cerca”.

Al click del tasto “Cerca” verrà controllato che nei due input ci sia presente del testo (in caso contrario verrà mostrato un errore), e effettuerà una chiamata asincrona al web service [searchSupermercati_utente.php](#):

- Dati da inviare: *emailUtente, nome, provincia*
- Dati da ricevere: *risultato, supermercati[idUtenteBackoffice, valutazione, nome, indirizzo]*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (operazione eseguita correttamente); viene mostrata la lista dei supermercati
- *Risultato* = 1 (nessun supermercato trovato); viene mostrato in output il messaggio di errore “nessun supermercato trovato”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

Per ogni supermercato sarà possibile inserire la valutazione (vedi [Valuta supermercato](#))

• [Valuta supermercato](#)

Questa funzione, dovrà presentare un input di testo per la valutazione e un tasto “valuta” per ogni supermercato della funzione [Ricerca supermercato](#).

Al click del tasto “Valuta” verrà controllato che nell’ input ci sia presente del testo (in caso contrario verrà mostrato un errore), e effettuerà una chiamata asincrona al web service [addValutazione_utente.php](#):

- Dati da inviare: *email*, *voto*, *idUtenteBackOffice*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (operazione eseguita correttamente); viene mostrato un messaggio di conferma
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

• [Logout](#)

Viene cancellata la sessione e l’utente viene rimandato al login

[Funzioni backoffice \(sito web\):](#)

Tutte le funzioni del dipendente possono essere svolte anche dall’admin.

• [Login \(backoffice\)](#)

Questa pagina, necessaria per accedere a tutte le altre funzionalità, dovrà presentare due input di testo, uno per la email e uno per la password, un bottone per la visualizzazione della password, il quale al click mostrerà/censurerà la password, e un tasto “Accedi”, ci sarà anche un tasto che reinderizzerà alla [Richiesta registrazione \(supermercato\)](#).

Al click del tasto “Accedi” verrà controllato che nei due input ci sia presente del testo (in caso contrario verrà mostrato un errore), e effettuerà una chiamata asincrona al web service [checkLogin_backoffice.php](#):

- Dati da inviare: *email*, *password*
- Dati da ricevere: *risultato*, *idUtenteBackOffice*, *ruolo*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (credenziali corrette); l'idUtenteBackOffice viene salvata nella sessione e l'utente viene reindirizzato alla home page in base al ruolo
- *Risultato* = 1 (credenziali errate); viene mostrato in output il messaggio di errore “Credenziali errate”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

• Cambia password (backoffice)

Questa pagina dovrà presentare tre input di testo, uno per la vecchia password, uno per la nuova password e uno per la conferma della nuova password, e un tasto “Cambia password”.

Al click del tasto “Cambia password” verrà controllato che nei tre input ci sia presente del testo e che le due nuove password corrispondano (in caso contrario verrà mostrato un errore), e effettuerà una chiamata asincrona al web service `changePassword_utente.php`:

- Dati da inviare: `vecchiaPassword`, `nuovaPassword`, `idUtenteBackoffice`
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (password cambiata); viene mostrato in output il messaggio “Password cambiata”
- *Risultato* = 1 (vecchia password sbagliata); viene mostrato in output il messaggio di errore “Vecchia password non corrisponde”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

• Richiesta registrazione (supermercato)

Questa pagina dovrà presentare sette input di testo, nome, indirizzo, latitudine, longitudine, provincia, email e password, e un tasto “Registrati”, ci sarà anche un tasto che reinderizzerà alla [Login \(backoffice\)](#).

Al click del tasto “Registrati” verrà controllato che nei sette input ci sia presente del testo (in caso contrario verrà mostrato un errore), e effettuerà una chiamata asincrona al web service `checkRegistrazione_supermercato.php`:

- Dati da inviare: `email`, `password`, `nome`, `indirizzo`, `latitudine`, `longitudine`, `provincia`
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (registrazione effettuata); viene mostrato in output il messaggio “Richiesta registrazione effettuata”
- *Risultato* = 1 (email già utilizzata); viene mostrato in output il messaggio di errore “Email già utilizzata”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

• Registra prodotto (supermercato)

Questa pagina, avrà due input di testo, uno per il codice del prodotto (quello sotto il codice a barre) e uno per il prezzo del prodotto, e avrà un bottone “aggiungi”.

Al click del bottone verrà controllato che il valore di input del prezzo e del codice siano valido e effettuerà una chiamata asincrona al web service [addProdotto_supermercato.php](#):

- Dati da inviare: *codiceProdotto, prezzo, idUtenteBackOffice*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

• [Visualizza prodotti \(supermercato\)](#)

Questa pagina al caricamento effettuerà una chiamata asincrona al web service [getProdotti_supermercato.php](#):

- Dati da inviare: *idUtenteBackoffice*
- Dati da ricevere: *risultato, prodotti[idProdottoSupermercato, codiceProdotto, prezzo, nome, immagine]*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output la lista dei prodotti del supermercato
- Risultato = 1 (nessun prodotto); viene mostrato in output il messaggio “nessun prodotto presente”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

A questo punto sarà possibile per ogni prodotto modificarne la quantità (vedi [Modifica prezzo prodotto \(supermercato\)](#)) o eliminarlo dal carrello (vedi [Elimina prodotto \(supermercato\)](#)).

• [Modifica prezzo prodotto \(supermercato\)](#)

Questa funzione, chiamata al click del tasto “modifica” situato su ogni prodotto, controllerà che il valore di input del prezzo sia valido e effettuerà una chiamata asincrona al web service [changeProdotto_supermercato.php](#):

- Dati da inviare: *idProdottoSupermercato, prezzo*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Elimina prodotto (supermercato)*

Questa funzione, chiamata al click del tasto “elimina” situato su ogni prodotto e effettuerà una chiamata asincrona al web service `deleteProdotto_supermercato.php`:

- Dati da inviare: *idProdottoSupermercato*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Visualizza richieste supermercati (dipendente)*

Questa pagina mostrerà al dipendente un menù a tendina dove potrà scegliere le provincie.

Una volta scelta la provincia verrà effettuata una chiamata asincrona al web service `getRichiesteSupermercati_dipendente.php`:

- Dati da inviare: *provincia*
- Dati da ricevere: *risultato, supermercati[idUtenteBackOffice, nome, indirizzo, latitudine, longitudine]*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output la lista dei supermercati
- Risultato = 1 (nessun risultato); viene mostrato in output un messaggio “nessuna richiesta in questa provincia”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

A questo punto sarà possibile per ogni supermercato accettarlo o rifiutarlo (vedi [Accetta/rifiuta supermercato \(dipendente\)](#)).

● *Accetta/rifiuta supermercato (dipendente)*

Questa funzione, chiamata al click dei tasti “accetta” e “rifiuta” di [Visualizza richieste supermercato \(dipendente\)](#).

Una volta scelta la provincia verrà effettuata una chiamata asincrona al web service `setRichiesteSupermercato_dipendente.php`:

- Dati da inviare: *idUtenteBackOffice, statoAccettazione*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● Visualizza statistiche (admin)

Questa pagina chiederà di inserire all'admin due date, una di inizio e una di fine. Una volta cliccato il tasto “cerca” verrà effettuata una chiamata asincrona al web service `getStatistiche_admin.php`:

- ❑ Dati da inviare: `dataInizio, dataFine`
- ❑ Dati da ricevere: `risultato, prodotti[codiceProdotto, rilevanza, nome, immagine]`

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output la lista dei prodotti
- Risultato = 1 (nessun risultato); viene mostrato in output un messaggio “nessun risultato”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● Registra dipendente (admin)

Questa pagina dovrà presentare cinque input di testo, uno per il codice fiscale, uno per la email, uno per la password, uno per il nome e l'ultimo per il cognome, e un tasto “Registra”.

Al click del tasto “Registra” verrà controllato che nei cinque input ci sia presente del testo (in caso contrario verrà mostrato un errore), e effettuerà una chiamata asincrona al web service `newDipendente_admin.php`:

- ❑ Dati da inviare: `cf, email, password, nome, cognome`
- ❑ Dati da ricevere: `risultato`

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (registrazione effettuata); viene mostrato in output il messaggio “Registrazione effettuata”
- *Risultato* = 1 (email già utilizzata); viene mostrato in output il messaggio di errore “Email già utilizzata”
- *Risultato* = 2 (cf già presente); viene mostrato in output il messaggio di errore “Dipendente già registrato, cf già presente”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

Funzioni utente (applicazione smartphone):

● Login

Questa pagina, la quale sarà necessaria per accedere a tutte le altre funzionalità (esclusa **Registrazione**), dovrà presentare due input di testo, uno per la email e uno per la password, un bottone per la visualizzazione della password, il quale al click mostrerà/censurerà la password, e un tasto “Accedi”, ci sarà anche un tasto che reinderizzerà alla **Registrazione**.

Al click del tasto “Accedi” verrà controllato che nei due input ci sia presente del testo (in caso contrario verrà mostrato un errore), e verrà effettuata una chiamata asincrona al web service `checkLogin_utente.php`:

- Dati da inviare: *email, password*
- Dati da ricevere: *risultato, email*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (credenziali corrette); l'email viene salvata nella sessione e l'utente viene reindirizzato alla home page
- *Risultato* = 1 (credenziali errate); viene mostrato in output il messaggio di errore "Credenziali errate"
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Registrazione*

Questa pagina dovrà presentare quattro input di testo, uno per la email, uno per la password, uno per il nome e l'ultimo per il cognome, e un tasto "Registrati", ci sarà anche un tasto che reinderizzerà al [Login](#).

Al click del tasto "Registrati" verrà controllato che nei quattro input ci sia presente del testo (in caso contrario verrà mostrato un errore), e verrà effettuata una chiamata asincrona al web service [checkRegistrazione_utente.php](#):

- Dati da inviare: *email, password, nome, cognome*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (registrazione effettuata); viene mostrato in output il messaggio "Registrazione effettuata"
- *Risultato* = 1 (email già utilizzata); viene mostrato in output il messaggio di errore "Email già utilizzata"
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Home*

Questa pagina dovrà presentare un messaggio di benvenuto, con una spiegazione generale di cosa può fare il sito. In questa pagina sarà presente il menù, il quale avrà i buttoni per andare in [Home](#), [Ricerca prodotto](#), [Visualizza carrello](#), [Valuta supermercato](#). Sarà anche possibile andare sulla pagina [Cambia password](#).

● *Cambia password*

Questa pagina dovrà presentare tre input di testo, uno per la vecchia password, uno per la nuova password e uno per la conferma della nuova password, e un tasto "Cambia password".

Al click del tasto "Cambia password" verrà controllato che nei tre input ci sia presente del testo e che le due nuove password corrispondano (in caso contrario verrà mostrato un errore), e verrà effettuata una chiamata asincrona al web service [changePassword_utente.php](#):

- Dati da inviare: *vecchiaPassword, nuovaPassword, emailUtente*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (password cambiata); viene mostrato in output il messaggio “Password cambiata”
- *Risultato* = 1 (password sbagliata); viene mostrato in output il messaggio di errore “Vecchia password non corrisponde”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Ricerca prodotto*

Questa pagina presenterà una barra di ricerca e un bottone “Mostra prodotti rilevanti”, il quale al click chiamerà **Mostra prodotti rilevanti**.

Appena l’utente inserirà del testo nella barra di ricerca e schiacerà il tasto invio verrà effettuata una chiamata asincrona al servizio API [“https://api.upcitemdb.com/prod/trial/search”](https://api.upcitemdb.com/prod/trial/search) passando come parametro il *testo* presente nella barra di ricerca (guardare documentazione sul sito):

- Dati da inviare: *s* (stringa di ricerca)
- Dati da ricevere: *response, data*

Quindi, in assenza di errori, verranno mostrati i prodotti (contenuti in *data.items[]*).

Altrimenti mostrato un messaggio di errore (*response.status*)

Per ogni prodotto sarà mostrato il nome e l’immagine e si potrà sceglierne la quantità.

A questo punto sarà possibile per ogni prodotto aggiungerlo al carrello (vedi **Aggiungi prodotto al carrello**).

● *Mostra prodotti rilevanti*

Questa pagina, al caricamento, effettuerà una chiamata asincrona al web service **getProdottiRilevanti_utente.php**:

- Dati da inviare:
- Dati da ricevere: *risultato, prodotti[codiceProdotto, rilevanza, nome, immagine]*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output la lista dei prodotti rilevanti
- Risultato = 1 (nessun risultato); viene mostrato in ouput “nessun prodotto rilevante al momento”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

Per ogni prodotto sarà mostrato il nome, il numero di rilevanza e l’immagine e si potrà sceglierne la quantità.

A questo punto sarà possibile per ogni prodotto aggiungerlo al carrello (vedi **Aggiungi prodotto al carrello**).

● Aggiungi prodotto al carrello

Questa funzione, chiamata al click del tasto “+” situato su ogni prodotto nelle pagine [Mostra prodotti rilevanti](#) e [Ricerca prodotto](#), controllerà che il valore di input di quantità sia valido e effettuerà una chiamata asincrona al web service [addProdotto_utente.php](#):

- Dati da inviare: *codiceProdotto, quantità, email*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● Visualizza carrello

Questa pagina effettuerà al caricamento una chiamata asincrona al web service [getCarrello_utente.php](#):

- Dati da inviare: *emailUtente*
- Dati da ricevere: *risultato, prodotti[idProdottoLista, quantità, nome, immagine]*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output la lista dei prodotti nella lista
- Risultato = 1 (carrello vuoto); viene mostrato in output il messaggio “nessun prodotto nel carrello”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

A questo punto sarà possibile per ogni prodotto modificarne la quantità (vedi [Modifica quantità prodotto carrello](#)) o eliminarlo dal carrello (vedi [Elimina prodotto carrello](#)).

Ci sarà anche un tasto “svuota carrello” il quale chiamerà il web service [deleteCarrello.php](#):

- Dati da inviare: *emailUtente*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● Modifica quantità prodotto carrello

Questa funzione, chiamata al click del tasto “modifica” situato su ogni prodotto nella pagina [Visualizza carrello](#), controllerà che il valore di input di quantità sia valido e

effettuerà una chiamata asincrona al web service `changeProdottoCarrello_utente.php`:

- Dati da inviare: `idProdottoLista, quantita`
- Dati da ricevere: `risultato`

Le azioni successive dipenderanno dal valore di `risultato`:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Elimina prodotto carrello*

Questa funzione, chiamata al click del tasto “elimina” situato su ogni prodotto nella pagina `Visualizza carrello`, effettuerà una chiamata asincrona al web service `deleteProdottoCarrello_utente.php`:

- Dati da inviare: `idProdottoLista`
- Dati da ricevere: `risultato`

Le azioni successive dipenderanno dal valore di `risultato`:

- Risultato = 0 (operazione eseguite correttamente); viene mostrato in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Elimina tutto il carrello*

Questa funzione, chiamata al click del tasto “svuota carrello” situato nella pagina `Visualizza carrello`, effettuerà una chiamata asincrona al web service `deleteCarrello.php`:

- Dati da inviare: `emailUtente`
- Dati da ricevere: `risultato`

Le azioni successive dipenderanno dal valore di `risultato`:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output un messaggio di conferma
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

● *Mostra supermercati*

Questa pagina effettuerà al caricamento una chiamata asincrona al web service `getSupermercati_utente.php`:

- Dati da inviare: `emailUtente, posizione, tipo(=prezzo)`
- Dati da ricevere: `risultato, supermercati[nome, indirizzo, prezzo, distanza, valutazione]`

Le azioni successive dipenderanno dal valore di `risultato`:

- Risultato = 0 (operazione eseguite correttamente); viene mostrata in output la lista dei supermercati
- Risultato = 1 (nessun supermercato trovato); viene mostrato in output il messaggio “nessun supermercato trovato”
- Risultato = -1 (errore); viene mostrato in output un messaggio di errore generale

Sarà possibile ora cliccare sui bottoni “prezzo” “distanza” “valutazione”, i quali richiameranno lo stesso web service con gli stessi dati, ad eccezione di tipo, che cambierà in base all’ordinamento richiesto.

● *Ricerca supermercato*

Questa pagina, necessaria per accedere alla funzione **Valuta supermercato**, dovrà presentare due input di testo, uno per il nome del supermercato e uno per la provincia, e un tasto “Cerca”.

Al click del tasto “Cerca” verrà controllato che nei due input ci sia presente del testo (in caso contrario verrà mostrato un errore), e effettuerà una chiamata asincrona al web service **searchSupermercati_utente.php**:

- Dati da inviare: *emailUtente, nome, provincia*
- Dati da ricevere: *risultato, supermercati[idUtenteBackoffice, valutazione, nome, indirizzo]*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (operazione eseguita correttamente); viene mostrata la lista dei supermercati
- *Risultato* = 1 (nessun supermercato trovato); viene mostrato in output il messaggio di errore “nessun supermercato trovato”
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

Per ogni supermercato sarà possibile inserire la valutazione (vedi **Valuta supermercato**)

● *Valuta supermercato*

Questa funzione, dovrà presentare un input di testo per la valutazione e un tasto “valuta” per ogni supermercato della funzione **Ricerca supermercato**.

Al click del tasto “Valuta” verrà controllato che nell’ input ci sia presente del testo (in caso contrario verrà mostrato un errore), e effettuerà una chiamata asincrona al web service **addValutazione_utente.php**:

- Dati da inviare: *email, voto, idUtenteBackOffice*
- Dati da ricevere: *risultato*

Le azioni successive dipenderanno dal valore di *risultato*:

- *Risultato* = 0 (operazione eseguita correttamente); viene mostrato un messaggio di conferma
- *Risultato* = -1 (errore); viene mostrato in output un messaggio di errore generale

- *Logout*

Viene cancellata la sessione e l'utente viene rimandato al login

Web services:

Tutti i web services avranno il collegamento al database e dovranno ritornare i dati in formato JSON

- *checkLogin_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *email, password*

La funzione, quindi, eseguirà una select dal database nella tabella utente, impostando come clausola WHERE la *email*.

A questo punto tramite la funzione di verifica della password criptata, verrà controllata la corrispondenza tra la *password* e la password sul database.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; credenziali corrette
- *Risultato* = 1; mail non presente / password sbagliata
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, email*

- *checkRegistrazione_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *email, password, nome, cognome*

La funzione, quindi, eseguirà una select dal database nella tabella utente, impostando come clausola WHERE la *email*.

Se non ci sono risultati, verrà eseguito un insert nella tabella con tutti i dati dell'utente.

Verrà anche inviata una mail all'utente di benvenuto e conferma registrazione.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; registrazione effettuata
- *Risultato* = 1; email già utilizzata (nel caso la select trovi qualcosa)
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

- *changePassword_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *vecchiaPassword, nuovaPassword, emailUtente*

La funzione, quindi, eseguirà una select dal database nella tabella utente, impostando come clausola WHERE la *email*.

A questo punto tramite la funzione di verifica della password criptata, verrà controllata la corrispondenza tra la *vecchiaPassword* e la password sul database. Quindi verrà eseguito un update con la *nuovaPassword*.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; password cambiata
- *Risultato* = 1; password sbagliata
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *getProdottiRilevanti_utente.php*

Questa funzione eseguirà una select dal database nella tabella prodottoRilevante impostando come clausola WHERE una data compresa tra la data momentanea (ricavato tramite una funzione) e due settimane prima della data momentanea, facendo un COUNT e raggruppando tutto per il *codiceProdotto*.

A questo punto la funzione, per ottenere per ogni prodotto il nome e l'immagine, chiamerà il servizio API “<https://api.upcitemdb.com/prod/trial/lookup>” passando come parametro il *codiceProdotto* (guardare documentazione sul sito).

A questo punto potrà creare il vettore prodotti con nome, immagine, rilevanza e codice.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; operazione eseguita correttamente
- *Risultato* = 1; nessun risultato
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, prodotti[codiceProdotto, rilevanza, nome, immagine]*

● *addProdotto_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *codiceProdotto, quantità, email*

La funzione, quindi, eseguirà una select dal database nella tabella prodottoListaSpesa, impostando come clausola WHERE la *email* e *codiceProdotto*.

A questo punto, se il prodotto è già presente nel carrello (quindi la query torna un record), verranno sommate la quantità presente nel database e quella ricevuta, e verrà eseguito un update solo per la quantità.

Altrimenti verrà eseguito un insert con i dati ricevuti.

Verrà eseguito a prescindere anche un insert nella tabella prodottoRilevante, nella quale verrà inserito solo il *codiceProdotto* e la data.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; prodotto aggiunto correttamente
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

- *getCarrello_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *email*

Questa funzione eseguirà una select dal database nella tabella prodottoListaSpesa impostando come clausola WHERE la *email*.

A questo punto la funzione, per ottenere per ogni prodotto il nome e l'immagine, chiamerà il servizio API “<https://api.upcitemdb.com/prod/trial/lookup>” passando come parametro il *codiceProdotto* (guardare documentazione sul sito).

A questo punto potrà creare il vettore prodotti con nome, immagine, rilevanza e codice.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; operazione eseguita correttamente
- *Risultato* = 1; carrello vuoto
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, prodotti[idProdottoLista, quantità, nome, immagine]*

- *deleteCarrello_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *email*

Questa funzione eseguirà una delete dal database nella tabella prodottoListaSpesa impostando come clausola WHERE la *email*, cancellando tutti i prodotti nella lista di quell'utente.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; operazione eseguita correttamente
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

- *changeProdottoCarrello_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *idProdottoLista, quantita*

La funzione, quindi, eseguirà un update nel database nella tabella prodottoListaSpesa, impostando come clausola WHERE *idProdottoLista* e modificando *quantità*.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; quantità modificata correttamente

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *deleteProdottoCarrello_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *idProdottoLista*

La funzione, quindi, eseguirà una delete nel database nella tabella prodottoListaSpesa, impostando come clausola WHERE *idProdottoLista*.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; quantità modificata correttamente

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *getSupermercati_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *emailUtente, posizione, tipo*

Questa funzione eseguirà una select dal database nella tabella prodottoListaSpesa impostando come clausola WHERE *emailUtente*.

La query dovrà ottenere la lista di tutti i supermercati, che vendono tutti i prodotti richiesti dall'utente, con il rispettivo prezzo totale, la distanza e la valutazione (tramite la tabella valutazione).

Quindi verrà creato un array di *supermercati[nome, indirizzo, prezzo, distanza, valutazione]* ordinato in base a *tipo*.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; operazione eseguita correttamente

→ *Risultato* = 1; nessun supermercato trovato

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, supermercati[nome, indirizzo, prezzo, distanza, valutazione]*

● *searchSupermercati_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *emailUtente, nome, provincia*

Questa funzione eseguirà una select dal database nella tabella supermercato con la parola chiave *nome* e *provincia*.

Così verrà controllato per ogni supermercato trovato che nella tabella valutazione non ci sia già una valutazione dell'utente per quel supermercato.

Quindi verrà creato un array di *supermercati[idUtenteBackoffice, valutazione, nome, indirizzo]*.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; operazione eseguita correttamente
- *Risultato* = 1; nessun supermercato trovato
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, supermercati[idUtenteBackoffice, valutazione, nome, indirizzo]*

● *addValutazione_utente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *emailUtente, voto, idUtenteBackOffice*

Questa funzione eseguirà una select dal database impostando come clausola WHERE *email* e *idUtenteBackOffice* per vedere se l'utente ha già dato in passato una valutazione a quel supermercato, in questo caso verrà fatto un update con la nuova valutazione.

Se l'utente non ha mai valutato il supermercato verrà fatta una insert della valutazione.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; operazione eseguita correttamente
- *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *checkLogin_backoffice.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *email, password*

La funzione, quindi, eseguirà una select dal database nella tabella utenteBackOffice, impostando come parametri la *email*.

A questo punto tramite la funzione di verifica della password criptata, verrà controllata la corrispondenza tra la *password* e la password sul database.

Quindi controllerà se l'utente è un supermercato oppure un dipendente (o admin).

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; credenziali corrette

→ *Risultato* = 1; mail non presente / password sbagliata

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, idUtenteBackOffice, ruolo*

● *changePassword_backoffice.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *vecchiaPassword, nuovaPassword, idUtenteBackOffice*

La funzione, quindi, eseguirà una select dal database nella tabella utenteBackOffice, impostando come parametri la *email*.

A questo punto tramite la funzione di verifica della password criptata, verrà controllata la corrispondenza tra la *vecchiaPassword* e la password sul database. Quindi verrà eseguito un update con la *nuovaPassword*.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; password cambiata
→ *Risultato* = 1; password sbagliata
→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *checkRegistrazione_supermercato.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *email, password, nome, indirizzo, latitudine, longitudine, provincia*

La funzione, quindi, eseguirà una select dal database nella tabella utenteBackOffice, impostando come parametri la *email*.

Se non ci sono risultati, verrà eseguito un insert nella tabella con email e password dell'utente.

Subito dopo verrà fatta una select con l'email per prendere l'id incrementale assegnato e verrà fatta una insert nella tabella supermercato con tutto il resto dei dati.

Verrà anche inviata una mail all'utente di benvenuto e conferma richiesta registrazione.

Di conseguenza *risultato* potrà avere i seguenti valori:

- *Risultato* = 0; richiesta registrazione effettuata
→ *Risultato* = 1; email già utilizzata
→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *addProdotto_supermercato.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *codiceProdotto, prezzo, idUtenteBackOffice*

La funzione, quindi, eseguirà una select dal database nella tabella prodottoSupermercato, impostando come parametri la *idUtenteBackOffice* e *codiceProdotto*.

A questo punto, se il prodotto è già presente nel database, verrà sostituito il prezzo con un update.

Altrimenti verrà eseguito un insert con i dati ricevuti.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; prodotto aggiunto correttamente

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *getProdotti_supermercato.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *idUtenteBackOffice*

Questa funzione eseguirà una select dal database nella tabella prodottoListaSupermercato impostando come parametri *idUtenteBackOffice*.

A questo punto la funzione, per ottenere per ogni prodotto il nome e l'immagine, chiamerà il servizio API “<https://api.upcitemdb.com/prod/trial/lookup>” passando come parametro il *codiceProdotto* (guardare documentazione sul sito).

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; operazione eseguita correttamente

→ *Risultato* = 1; nessun prodotto presente nel supermercato

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, prodotti[idProdottoSupermercato, codiceProdotto, prezzo, nome, immagine]*

● *changeProdotto_supermercato.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *idProdottoSupermercato, prezzo*

La funzione, quindi, eseguirà un update nel database nella tabella prodottoSupermercato, impostando come parametri *idProdottoSupermercato* e modificando *prezzo*.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; prezzo modificato correttamente

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *deleteProdotto_supermercato.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *idProdottoSupermercato*

La funzione, quindi, eseguirà una delete nel database nella tabella prodottoSupermercato, impostando come parametri *idProdottoSupermercato*.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; prodotto eliminato correttamente

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

● *getRichiesteSupermercati_dipendente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *provincia*

La funzione, quindi, eseguirà una select nel database nella tabella supermercato, impostando come parametri *provincia* e *statoAccettazione*.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; operazione eseguita correttamente

→ *Risultato* = 1; nessuna risultato

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, supermercati[idUtenteBackOffice, nome, indirizzo, latitudine, longitudine]*

● *setRichiestaSupermercato_dipendente.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *idUtenteBackOffice, statoAccettazione*

La funzione, quindi, eseguirà un update nel database nella tabella supermercato, impostando come parametri *idUtenteBackOffice* e *statoAccettazione*.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; operazione eseguita correttamente

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato*

- *getStatistiche_admin.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *dataInizio, dataFine*

Questa funzione eseguirà una select dal database nella tabella prodottoRilevante impostando come parametri le due date ricevute facendo un COUNT e raggruppando tutto per il *codiceProdotto*.

A questo punto la funzione, per ottenere per ogni prodotto il nome e l'immagine, chiamerà il servizio API “<https://api.upcitemdb.com/prod/trial/lookup>” passando come parametro il *codiceProdotto* (guardare documentazione sul sito).

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; operazione eseguita correttamente

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

- Dati da inviare: *risultato, prodotti[codiceProdotto, rilevanza, nome, immagine]*

- *newDipendente_admin.php*

Questa funzione deve ricevere dei parametri in input che sono:

- Dati da ricevere: *cf, email, password, nome, cognome*

La funzione, quindi, eseguirà una select dal database nella tabella utenteBackoffice, impostando come parametri la *email*.

Successivamente eseguirà una select dal database nella tabella dipendente, impostando come parametri il *cf*.

Se non ci sono risultati, verrà eseguito un insert nelle tabelle con tutti i dati dell'utente.

Verrà anche inviata una mail all'dipendente in cui viene confermata registrazione e gli viene consigliato di cambiare la password.

Di conseguenza *risultato* potrà avere i seguenti valori:

→ *Risultato* = 0; registrazione effettuata

→ *Risultato* = 1; email già utilizzata

→ *Risultato* = 1; cf già presente

→ *Risultato* = -1; errore

Quindi verranno tornati come risposta i seguenti dati:

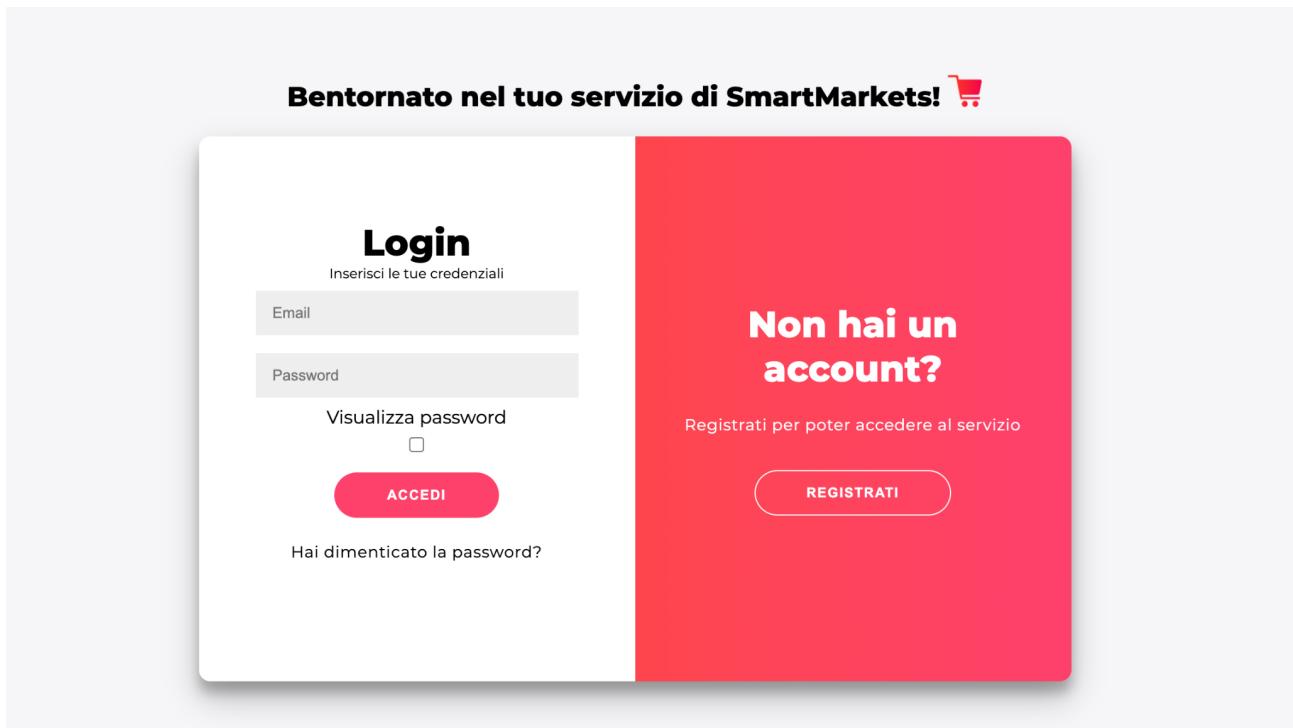
- Dati da inviare: *risultato*

7 Sviluppo Prototipo

Connessione DB:

```
login.php           script.js          prodottiLista.php      supermercati.php      db_connection.php
1 <?php
2 ob_start();
3
4 session_start();
5
6 $host = "31.11.39.26";
7 $user = "Sql1506670";
8 $password = "0831qrr4f2";
9 $db = "Sql1506670_3";
10 $conn = mysqli_connect($host, $user, $password, $db);
11
12 if ($conn->connect_errno) {
13     printf("Connect failed: %s\n", $conn->connect_error);
14 }
15
```

Login e Registrazione:



```

login.php
1 <?php
2 session_start();
3 if (isset($_SESSION['email'])) {
4     header("location:index.php");
5 }
6 ?>
7 <!doctype html>
8
9 <html lang="it">
10 <head>
11     <meta charset="utf-8">
12     <link rel="icon" type="image/png" href="images/icon.png" />
13
14     <title>Login</title>
15     <meta name="description" content="The HTML5 Herald">
16     <meta name="author" content="SitePoint">
17     <link rel="stylesheet" href="css/style.css">
18
19 </head>
20
21 <body>
22
23     <h2>Bentornato nel tuo servizio di SmartMarkets!  </h2>
24     <div class="container" id="container">
25         <div class="form-container sign-up-container">
26             <form id="registrazione" method="post">
27                 <h1>Registrati</h1>
28                 <span>Inserisci i tuoi dati</span>
29                 <input id="nome" type="text" placeholder="Nome" />
30                 <input id="cognome" type="text" placeholder="Cognome" />
31                 <input id="emailReg" type="email" placeholder="Email" />
32                 <input id="passwordReg" type="password" placeholder="Password" />
33                 <div style="color:#FF8C00; margin-bottom:10px" id="registrazione-check" class="output-check"></div>
34                 <button type="submit">Registrati</button>
35             </form>
36
37             login.php
38                 <button type="submit">Registrati</button>
39             </form>
40         </div>
41         <div class="form-container sign-in-container">
42             <form id="login" method="post" data-bitwarden-watching="1">
43                 <h1>Login</h1>
44                 <span>Inserisci le tue credenziali</span>
45                 <input id="email" type="email" placeholder="Email" />
46                 <input id="password" type="password" placeholder="Password" />
47                 <div> Visualizza password<input style="display:inline" type="checkbox" id="check"></div>
48                 <div style="color:#FF8C00; margin-bottom:10px" id="login-check" class="output-check"></div>
49                 <button type="submit">Accedi</button>
50                 <p id="easterEgg" onclick="easterEgg()">Hai dimenticato la password?</p>
51             </form>
52         </div>
53         <div class="overlay-container">
54             <div class="overlay">
55                 <div class="overlay-panel overlay-left">
56                     <h1>Bentornato!</h1>
57                     <p>Accedi con le tue credenziali</p>
58                     <button class="ghost" id="signIn">Login</button>
59                 </div>
60                 <div class="overlay-panel overlay-right">
61                     <h1>Non hai un account?</h1>
62                     <p>Registrati per poter accedere al servizio</p>
63                     <button class="ghost" id="signUp">Registrati</button>
64                 </div>
65             </div>
66         </div>
67
68         <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
69         <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
70         <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js" type="text/javascript"></script>
71         <script>
72             $(document).ready(function() {

```

```
login.php
48 </div>
49 <div class="overlay-container">
50   <div class="overlay">
51     <div class="overlay-panel overlay-left">
52       <h1>Bentornato!</h1>
53       <p>Accedi con le tue credenziali</p>
54       <button class="ghost" id="signIn">Login</button>
55     </div>
56     <div class="overlay-panel overlay-right">
57       <h1>Non hai un account?</h1>
58       <p>Registrati per poter accedere al servizio</p>
59       <button class="ghost" id="signUp">Registrati</button>
60     </div>
61   </div>
62 </div>
63 </div>
64 <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
65 <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
66 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js" type="text/javascript"></script>
67 <script>
68   $(document).ready(function() {
69     $('#check').click(function() {
70       if (document.getElementById('check').checked) {
71         $('#password').get(0).type = 'text';
72       } else {
73         $('#password').get(0).type = 'password';
74       }
75     });
76   });
77 </script>
78 <script src="script/script.js"></script>
79 </body>
80 </html>
81
```

```

login.php | checkLogin.php
1 <?php
2 session_start();
3 require 'db_connection.php';
4 $error = 0;
5
6 if (isset($_REQUEST['email']) && isset($_REQUEST['password'])) {
7     $email = $_REQUEST['email'];
8     $password = $_REQUEST['password'];
9
10    $query = "SELECT email, nome, cognome, password FROM novelUtente WHERE email = ?";
11
12    $stmt = $conn->prepare($query);
13    $stmt->bind_param('s', $email); // 's' specifies the variable type => 'string'
14    $stmt->execute();
15    $stmt->bind_result($email, $nome, $cognome, $passwordInDB);
16    $stmt->store_result();
17    $num = $stmt->num_rows;
18
19    if ($num == 1) {
20        while ($stmt->fetch()) {
21            if (password_verify($password, $passwordInDB)) {           //verifica se la password cryptata nel database associata all'email
22                $updates = array(
23                    "nome"  => $nome,
24                    "cognome" => $cognome,
25                    "email"  => $email,
26                    "risultato" => (0)
27                );
28                $_SESSION['email']=$email;
29                $_SESSION['nome']=$nome;
30                $_SESSION['cognome']=$cognome;
31            }
32            else {
33                $error = -1; //password errata
34            }
35        }
36    }
37    $num = $stmt->num_rows;
38
39    if ($num == 1) {
40        while ($stmt->fetch()) {
41            if (password_verify($password, $passwordInDB)) {           //verifica se la password cryptata nel database associata all'email
42                $updates = array(
43                    "nome"  => $nome,
44                    "cognome" => $cognome,
45                    "email"  => $email,
46                    "risultato" => (0)
47                );
48                $_SESSION['email']=$email;
49                $_SESSION['nome']=$nome;
50                $_SESSION['cognome']=$cognome;
51            }
52            else {
53                $error = -1; //password errata
54            }
55        }
56    }
57    else {
58        $error = -2; //mail non presente
59    }
60    else {
61        $error = -3; //credenziali non inserite
62    }
63    if ($error != 0) {
64        $updates = array(
65            "risultato" => ($error)
66        );
67    }
68    $arr[0] = array_map('utf8_encode', $updates);
69
70    echo json_encode($arr[0]);
71
72}

```

```
login.php           checkRegistrazione.php
1 <?php
2 require('db_connection.php');
3 use PHPMailer\PHPMailer\PHPMailer;
4 $error = 0;
5
6 if (isset($_REQUEST['email'])) {
7     $nome = $_REQUEST['nome'];
8     $cognome = $_REQUEST['cognome'];
9     $email = $_REQUEST['email'];
10    $password = $_REQUEST['password'];
11
12    $stmt = $conn->prepare('SELECT * FROM novelUtente WHERE email = ?');
13    $stmt->bind_param('s', $email); // 's' specifies the variable type => 'string'
14
15    $stmt->execute();
16    $stmt->store_result();
17    $num = $stmt->num_rows;
18
19    if ($num > 0) {
20        $error = -1; //mail o cf già presente
21    }
22    if ($error == 0) {
23        $criptedPW = password_hash($password, PASSWORD_DEFAULT); //criptazione password hash
24        $qu = "INSERT into novelUtente VALUES(?, ?, ?, ?)";
25        $stmt = $conn->prepare($qu);
26
27        $stmt->bind_param("ssss", $email, $nome, $cognome, $criptedPW);
28        $stmt->execute();
29
30        $updates = array(
31            "risultato" => (0)
32        );
33
34        require_once 'PHPMailer/PHPMailer.php';
35        require_once 'PHPMailer/SMTP.php'.
```

```

login.php           checkRegistrazione.php
33
34     require_once 'PHPMailer/PHPMailer.php';
35     require_once 'PHPMailer/SMTP.php';
36     require_once 'PHPMailer/Exception.php';
37     $mail = new PHPMailer(); //link al mailer che permette di usare gmail
38     $mail->isSMTP();
39     $mail->Host = "smtp.gmail.com";
40     $mail->SMTPAuth = true;
41     $mail->Username = "smart.markets.volta@gmail.com";
42     $mail->Password = "AleNovel02";
43     $mail->Port = 465;
44     $mail->SMTPSecure = "ssl";
45     $mail->isHTML(true);
46     $mail->setFrom("smart.markets.volta@gmail.com");
47     $mail->addAddress($email);
48
49     $mail->Subject = 'Registrazione effettuata!';
50     $message = '<html><body>';
51     $message .= '<h1 style="color:#866ec6">Benvenuto</h1><br>
52         Salve,<b>' . $nome . ' ' . $cognome . '</b> il suo <b>account</b> è stato registrato correttamente
53         <br><br><i>Comincia ad usare il nostro servizio: https://www.appalo.it/5E/GruppoC/smartMarkets/</i>';
54     $message .= '</body></html>';
55     $mail->Body = $message;
56
57     if($mail->send){//invio
58         $status = "success";
59     }else{
50         $status = "failed";
51     }
52
53     $arr[0] = array_map('utf8_encode', $updates);
54     echo json_encode($arr[0]);
55 }
56 } else {
57
58     $mail->Port = 465;
59     $mail->SMTPSecure = "ssl";
60     $mail->isHTML(true);
61     $mail->setFrom("smart.markets.volta@gmail.com");
62     $mail->addAddress($email);
63
64     $mail->Subject = 'Registrazione effettuata!';
65     $message = '<html><body>';
66     $message .= '<h1 style="color:#866ec6">Benvenuto</h1><br>
67         Salve,<b>' . $nome . ' ' . $cognome . '</b> il suo <b>account</b> è stato registrato correttamente
68         <br><br><i>Comincia ad usare il nostro servizio: https://www.appalo.it/5E/GruppoC/smartMarkets/</i>';
69     $message .= '</body></html>';
70     $mail->Body = $message;
71
72     if($mail->send){//invio
73         $status = "success";
74     }else{
75         $status = "failed";
76     }
77
78     $arr[0] = array_map('utf8_encode', $updates);
79     echo json_encode($arr[0]);
80 }
81 } else {
82     $error = -3; //mail non inserita
83 }
84
85 if ($error != 0) {
86     $updates = array(
87         "risultato" => ($error)
88     );
89     $arr[0] = array_map('utf8_encode', $updates);
90     echo json_encode($arr[0]);
91 }
92
93 }
```

login.php	script.js	search.js
1 const signUpButton = document.getElementById('signUp'); 2 const signInButton = document.getElementById('signIn'); 3 const container = document.getElementById('container'); 4 5 signUpButton.addEventListener('click', () => { 6 container.classList.add("right-panel-active"); 7 }); 8 9 signInButton.addEventListener('click', () => { 10 container.classList.remove("right-panel-active"); 11}); 12 13 14 \$(document).ready(function() { 15 \$('#login').on('submit', function(e) { 16 \$("#login-check").html(""); 17 e.preventDefault(); 18 var email = document.getElementById("email").value; 19 var password = document.getElementById("password").value; 20 \$.ajax({ 21 url: 'webServices/checkLogin.php', 22 type: "POST", 23 data: { 24 email: email, 25 password: password 26 }, 27 success: function(data) { 28 const risultato = JSON.parse(data); 29 if (risultato.risultato != 0) { 30 \$("#login-check").html("Credenziali errate"); 31 } else { 32 location.reload(); 33 } 34 }, 35 error: function(jXHR, textStatus, errorThrown) { 36 \$("#login-check").html(errorThrown); 37 } 38 }); 39 }); 40 41 \$('#registrazione').on('submit', function(e2) { 42 e2.preventDefault(); 43 \$("#registrazione-check").html(""); 44 var email = document.getElementById("emailReg").value; 45 var password = document.getElementById("passwordReg").value; 46 var nome = document.getElementById("nome").value; 47 var cognome = document.getElementById("cognome").value; 48 \$.ajax({ 49 url: 'webServices/checkRegistrazione.php', 50 type: "POST", 51 data: { 52 email: email, 53 password: password, 54 nome: nome, 55 cognome: cognome 56 }, 57 success: function(data) { 58 const risultato = JSON.parse(data); 59 if (risultato.risultato != 0) { 60 \$("#registrazione-check").html("Errore"); 61 } else { 62 \$("#registrazione-check").html("Registrazione effettuata"); 63 } 64 }, 65 error: function(jXHR, textStatus, errorThrown) { 66 alert(errorThrown + "banana 2"); 67 } 68 }); 69 });		

```

login.php | script.js | search.js
-----|-----|-----
42 e2.preventDefault();
43 $("#registrazione-check").html("");
44 var email = document.getElementById("emailReg").value;
45 var password = document.getElementById("passwordReg").value;
46 var nome = document.getElementById("nome").value;
47 var cognome = document.getElementById("cognome").value;
48 $.ajax({
49   url: 'webServices/checkRegistrazione.php',
50   type: "POST",
51   data: {
52     email: email,
53     password: password,
54     nome: nome,
55     cognome: cognome
56   },
57   success: function(data) {
58     const risultato = JSON.parse(data);
59     if (risultato.risultato != 0) {
60       $("#registrazione-check").html("Errore");
61     } else {
62       $("#registrazione-check").html("Registrazione effettuata");
63     }
64   },
65   error: function(jXHR, textStatus, errorThrown) {
66     alert(errorThrown + "banana 2");
67   }
68 });
69 });
70 });
71
72 function easterEgg(){
73   $("#easterEgg").html("Potevi salvartela da qualche parte");
74 }
75

```

Prodotti rilevanti e ricerca prodotto:

The screenshot shows a mobile application interface for grocery delivery. On the left is a vertical navigation bar with five icons: Home (house), Search (magnifying glass), Cart (shopping cart), Favorites (star), and Share (arrow). The main content area displays three recommended products:

- Nutella**: A jar of Nutella spread. Below it, the text "Nutella" is followed by a thumbs-down icon with the number 19, and a quantity selector with values 1 and +.
- (12 Pack) Nutella & Go! Hazelnut Spread with Breadsticks, 1.9 oz each**: A container of Nutella & Go! spread. Below it, the text "(12 Pack) Nutella & Go! Hazelnut Spread with Breadsticks, 1.9 oz each" is followed by a thumbs-down icon with the number 2, and a quantity selector with values 1 and +.
- Gerber Yogurt Blends Snack Banana Yogurt 4-3.5 oz. Cups**: A box of Gerber yogurt blends. Below it, the text "Gerber Yogurt Blends Snack Banana Yogurt 4-3.5 oz. Cups" is followed by a thumbs-down icon with the number 2, and a quantity selector with values 1 and +.

A small button labeled "Mostra più rilevanti" is located at the top right of the product grid.

Home
Search
Cart
Favorites
Logout

Ricerca

Aggiungi i prodotti alla tua lista della spesa

coca cola soda
Mostra più rilevanti



Authentic Coca-Cola Coke Tall Glass Orlando



Authentic Coca-Cola Coke Chalk



Authentic Coca-Cola Coke Chalk Spatula

```

login.php
1 <?php
2
3 require_once('db_connection.php');
4
5 $date = date("Y-m-d");
6
7 $email = $_REQUEST['email'];
8 $codice = $_REQUEST['codice'];
9 $quantita = $_REQUEST['quantita'];
10 $immagine = $_REQUEST['immagine'];
11 $nome = $_REQUEST['nome'];
12
13 $stmt = $conn->prepare('SELECT * FROM novelProdotto WHERE codice = ?');
14 $stmt->bind_param('s', $codice); // 's' specifies the variable type => 'string'
15
16 $stmt->execute();
17 $stmt->store_result();
18 $num = $stmt->num_rows;
19
20 if ($num > 0) {
21     echo "Prodotto già presente";
22 }else{
23     echo "Prodotto non presente";
24     $qu = "INSERT into novelProdotto (codice, nome, immagine) VALUES(?, ?, ?)";
25     $stmt = $conn->prepare($qu);
26     $stmt->bind_param("sss", $codice, $nome, $immagine);
27     $stmt->execute();
28 }
29
30 $stmt = $conn->prepare('SELECT quantita FROM novelProdottoListaSpesa WHERE codiceProdotto = ? AND emailUtente = ?');
31 $stmt->bind_param('ss', $codice, $email); // 's' specifies the variable type => 'string'
32
33 $stmt->execute();
34 $stmt->bind_result($quantitaDB);
35 <?>$stmt->store_result();

```

```
23 echo "Prodotto non presente";
24 $qu = "INSERT into novelProdotto (codice, nome, immagine) VALUES(?, ?, ?)";
25 $stmt = $conn->prepare($qu);
26 $stmt->bind_param("sss", $codice, $nome, $immagine);
27 $stmt->execute();
28 }
29
30 $stmt = $conn->prepare('SELECT quantita FROM novelProdottoListaSpesa WHERE codiceProdotto = ? AND emailUtente = ?');
31 $stmt->bind_param('ss', $codice, $email); // 's' specifies the variable type => 'string'
32
33 $stmt->execute();
34 $stmt->bind_result($quantitaDB);
35 $stmt->store_result();
36 $num = $stmt->num_rows;
37
38 if ($num > 0) {
39     while ($stmt->fetch()) {
40         $quantita = $quantitaDB + $quantita;
41         $qu = "UPDATE novelProdottoListaSpesa SET quantita = ? WHERE emailUtente = ? AND codiceProdotto = ?";
42         $stmt = $conn->prepare($qu);
43         $stmt->bind_param("sss", $quantita, $email, $codice);
44         $stmt->execute();
45     }
46 }else{
47     $qu = "INSERT into novelProdottoListaSpesa (quantita, emailUtente, codiceProdotto) VALUES(?, ?, ?)";
48     $stmt = $conn->prepare($qu);
49     $stmt->bind_param("sss", $quantita, $email, $codice);
50     $stmt->execute();
51 }
52 $qu = "INSERT into novelProdottoRilevante (data, codiceProdotto) VALUES(?, ?)";
53 $stmt = $conn->prepare($qu);
54 $stmt->bind_param("ss", $date, $codice);
55 $stmt->execute();
56 }
```

```

login.php          ricerca.php
1  <?php header('Access-Control-Allow-Origin: *');
2  session_start();
3  ?>
4  <link rel="stylesheet" href="css/search.css">
5  <div id="cerca">
6      <div class="overlay" id="pop">
7          <div class="popup">
8              <h2>Prodotto aggiunto</h2><br>
9              <p>Hai aggiunto al tuo carrello <strong id="nome">nome</strong> in una quantità di <strong id="quantita">nome</strong></p>
10             <span id="close">X</span>
11         </div>
12     </div>
13 </div>
14 <h1 class="main-title">Ricerca</h1>
15 <p class="subtitle">Aggiungi i prodotti alla tua lista della spesa</p>
16 <div id="vue-container">
17     <div class="products-container">
18         <!-- IL tuo codice Qui -->
19         <div class="input-container">
20             <input v-model="ricerca" placeholder="Cerca prodotto" @keyup.enter="esegui_ricerca" />
21         </div>
22         <button class="button-title" v-on:click="rilevanti">Mostra più rilevanti</button>
23         <h1 class="errore">{{ errore }}</h1>
24         <div class="products">
25             <div class="product" v-for="(prodotto,index) in prodotti">
26                 
27
28                 <div class="product-texts">
29                     <h3 class="product-name">{{ prodotto.title }}</h3>
30                     <i class="fa fa-fire" style="color:white;font-size: 25px"> {{ prodotto.quantita }}</i><br>
31                     <input class="quantita" :id="'quantita-' + index" type="number" min="1" :value="1">
32                     <button :onclick="addProdotto(' + index + ')" class="button-title product-button">
33                         <i class="fa fa-plus"></i>
34                     </button>
35                     <input :id="'codice-' + index" type="hidden" :value="prodotto.ean">
36
37             </div>
38         </div>
39     </div>
40
41     </div>
42
43 </div>
44 </div>
45 <script src="script/search.js"></script>
46 <script>
47     document.getElementById("close").onclick = function(e){
48         document.getElementById('pop').style.display="none";
49     }
50
51 // chiudi il popup quando clicchi sullo sfondo nero
52     document.getElementById("pop").onclick = function(e){
53         document.getElementById('pop').style.display="none";
54     }
55     document.getElementById("pop").onclick = function(e){
56         document.getElementById('pop').style.display="none";
57     }
58     function addProdotto(i){
59
60         var codice = document.getElementById("codice-" + i).value;
61         var quantita = document.getElementById("quantita-" + i).value;
62         var email = document.getElementById("email-" + i).value;
63         var immagine = document.getElementById("immagine-" + i).value;
64         var nome = document.getElementById("nome-" + i).value;
65         if(quantita<1){
66             alert("Valore quantità non valido");
67         }else{
68             // invia i dati
69         }
70     }

```

```
login.php | ricerca.php
58 function addProdotto(i){
59
60     var codice = document.getElementById("codice-" + i).value;
61     var quantita = document.getElementById("quantita-" + i).value;
62     var email = document.getElementById("email-" + i).value;
63     var immagine = document.getElementById("immagine-" + i).value;
64     var nome = document.getElementById("nome-" + i).value;
65
66     if(quantita<1){
67         alert("Valore quantità non valido");
68     }else{
69         $.ajax({
70             url: 'webServices/addProdottoLista.php',
71             type: "POST",
72             data: {
73                 email: email,
74                 codice: codice,
75                 immagine: immagine,
76                 quantita: quantita,
77                 nome: nome
78             },
79             success: function(data) {
80                 var element = document.getElementById("tutto");
81                 $("#nome").html(nome);
82                 $("#quantita").html(quantita);
83                 document.getElementById('pop').style.display="block";
84             },
85             error: function(jXHR, textStatus, errorThrown) {
86                 alert(errorThrown + "banana 2");
87             }
88         });
89     }
90 
```

```
login.php | script.js | prodottiRilevanti.php
1 <?php
2 ob_start(); ?>
3 <?php
4
5 require_once('db_connection.php');
6
7 $date = date("Y-m-d", strtotime("-14 days"));
8
9
10 class Library {
11
12 }
13
14
15 class Book {
16
17 }
18 $query = "SELECT codiceProdotto, COUNT(*) as quantita, nome, immagine FROM `novelProdottoRilevante`, `novelProdotto` WHERE data >
19 $rec = mysqli_query($conn, $query) or die($query . "<br>" . mysqli_error($conn));
20 $num = mysqli_num_rows($rec);
21 if ($num == 0) {
22     $json = array(
23         "risultato"    => '-1',
24         "prodotti"    => '0'
25     );
26     $con[0] = $updates;
27 } else {
28     $i = 0;
29
30     while ($array = mysqli_fetch_array($rec)) {
31         $books[$i] = new Book();
32
33         $books[$i]->ean = $array['codiceProdotto'];
34
35 }
```

```
login.php | script.js | prodottiRilevanti.php
21 if ($num == 0) {
22     $json = array(
23         "risultato"    => '-1',
24         "prodotti"    => '0'
25     );
26     $con[0] = $updates;
27 } else {
28     $i = 0;
29
30     while ($array = mysqli_fetch_array($rec)) {
31         $books[$i] = new Book();
32
33         $books[$i]->ean = $array['codiceProdotto'];
34
35         $books[$i]->quantita = $array['quantita'];
36
37         $books[$i]->title = $array['nome'];
38
39         $books[$i]->images[] = $array['immagine'];
40
41         $i++;
42     }
43     $library = new Library();
44
45     $library->prodotti = $books;
46 }
47 echo json_encode($library);
48 ?>
49
50 <?php
51
52 // MyClass.php
53
```

```

login.php | script.js | search.js

```

```

1 var app = new Vue({
2   el: '#vue-container',
3   data: {
4     ricerca: '',
5     errore: '',
6     prodotti: []
7   },
8   created: function () {
9     axios.get('https://www.appalo.it/5E/GruppoC/smartMarkets/webServices/prodottiRilevanti.php')
10    .then(function(risultato) {
11      app.prodotti = risultato.data.prodotti;
12      console.log(risultato.data.prodotti);
13    });
14  },
15  methods: {
16    esegui_ricerca() {
17      app.prodotti = [];
18      app.errore = '';
19      axios.get('https://api.upcitemdb.com/prod/trial/search', {
20        params: {
21          s: this.ricerca,
22          match_mode: 1,
23          offset: 15,
24          type: 'product'
25        }
26      }).then(function(risultato) {
27        console.log(risultato);
28        app.prodotti = risultato.data.items;
29      }).catch(function(error){
30        console.log(error.response);
31        if (error.response.status == 429){
32          app.errore = 'Richieste esaurite';
33        }else if (error.response.status == 404){
34          app.errore = 'Prodotto non trovato';
35        }else if (error.response.status == 400){
36          app.errore = 'Ricerca non valida';
37        }else if (error.response.status == 'default'){
38          alert(error.response);
39        }
40      });
41    },
42    rilevanti() {
43      app.prodotti = [];
44      app.errore = '';
45      axios.get('https://www.appalo.it/5E/GruppoC/smartMarkets/webServices/prodottiRilevanti.php')
46      .then(function(risultato) {
47        app.prodotti = risultato.data.prodotti;
48        console.log(risultato.data.prodotti);
49      });
50    }
51  }
52 });

```

Carrello e confronto supermercati:

Lista della spesa

Gestisci i prodotti della tua lista della spesa

[Visualizza Prodotti](#) [Visualizza supermercati](#)



Nutella

32 ✓

[Elimina prodotto](#)



Coca Cola

2 ✓

[Elimina prodotto](#)



(12 Pack) Nutella & Go! Hazelnut Spread with Breadsticks, 1.9 oz each

Lista della spesa

Gestisci i prodotti della tua lista della spesa

[Visualizza Prodotti](#) [Visualizza supermercati](#)

Eurospesa Supermercati
Via della Raffineria, 6, 34138 Trieste TS
17.3 €
5.58 Km
Valutazione: 7.5/10

Supermercato Despar Donatello
Via del Donatello, 14, 34128 Trieste TS
19 €
6.99 Km
Valutazione: 10.0/10

```

login.php | script.js | lista.php
1 <?php header('Access-Control-Allow-Origin: *');
2 session_start();
3 ?>
4 <link rel="stylesheet" href="css/search.css">
5 <div id="lista">
6   <h1 class="main-title">Lista della spesa</h1>
7   <p class="subtitle">Gestisci i prodotti della tua lista della spesa</p>
8   <button onclick="visualizzaProdotti()" class="button-title">Visualizza Prodotti</button>
9   <button onclick="visualizzaSupermercati()" class="button-title">Visualizza supermercati</button>
10  <div id="corpo">
11    </div>
12  </div>
13  <script>
14  $(document).ready(function() {
15    visualizzaProdotti();
16  });
17
18  function visualizzaProdotti(){
19    $.ajax({
20      url: 'prodottiLista.php',
21      success: function(data) {
22        $("#corpo").html(data);
23      },
24      error: function(jXHR, textStatus, errorThrown) {
25        alert(errorThrown + "Errore nella chiamata della pagina");
26      }
27    });
28  });
29
30 }
31
32 function visualizzaSupermercati(){
33   $.ajax({
34     url: 'supermercati.php',
35     success: function(data) {
36       $("#corpo").html(data);
37     },
38     error: function(jXHR, textStatus, errorThrown) {
39       alert(errorThrown + "Errore nella chiamata della pagina");
40     }
41   });
42 }
43 </script>

```

```

login.php | script.js | lista.php
11  <div id="corpo">
12
13  </div>
14  <script>
15  $(document).ready(function() {
16    visualizzaProdotti();
17  });
18
19  function visualizzaProdotti(){
20    $.ajax({
21      url: 'prodottiLista.php',
22      success: function(data) {
23        $("#corpo").html(data);
24      },
25      error: function(jXHR, textStatus, errorThrown) {
26        alert(errorThrown + "Errore nella chiamata della pagina");
27      }
28    });
29  };
30
31  function visualizzaSupermercati(){
32    $.ajax({
33      url: 'supermercati.php',
34      success: function(data) {
35        $("#corpo").html(data);
36      },
37      error: function(jXHR, textStatus, errorThrown) {
38        alert(errorThrown + "Errore nella chiamata della pagina");
39      }
40    });
41  };
42 }
43 </script>

```

login.php	script.js	prodottiLista.php
<pre> 1 <?php 2 session_start(); 3 ?> 4 <div id="products" class="products"> 5 <div class="product" v-for="(prodotto,index) in prodotti"> 6 7 <div class="product-texts"> 8 <h3 class="product-name">{{ prodotto.title }}</h3> 9 <input class="quantita" :id="'quantita-' + index" type="number" min="1" :value="prodotto.quantita"> 10 <button :onclick="'updateProdotto(' + index + ')''" class="button-title product-button"> 11 <i class="fa fa-check"></i> 12 </button>
 13 <button :onclick="'deleteProdotto(' + index + ')''" class="button-title product-button"> 14 Elimina prodotto 15 </button> 16 <input :id="'codice-' + index" type="hidden" :value="prodotto.ean"> 17 <input :id="'email-' + index" type="hidden" value="<?php echo \$_SESSION['email'] ?>"> 18 </div> 19 </div> 20 </div> 21 <script> 22 var app = new Vue({ 23 el: '#products', 24 data: { 25 errore: '', 26 prodotti: [], 27 email: '<?php echo \$_SESSION['email'] ?>' 28 }, 29 created: function () { 30 console.log(this.email); 31 axios.get('https://www.appalo.it/5E/GruppoC/smartMarkets/webServices/prodottiLista.php', { 32 params: { 33 email: this.email 34 } 35 }) 36 }, 37 methods: { 38 caricaProdotti() { 39 app.errore = ''; 40 axios.get('https://www.appalo.it/5E/GruppoC/smartMarkets/webServices/prodottiLista.php', { 41 params: { 42 email: this.email 43 } 44 }) 45 }, 46 updateProdotto(i){ 47 var codice = document.getElementById("codice-" + i).value; 48 var quantita = document.getElementById("quantita-" + i).value; 49 var email = document.getElementById("email-" + i).value; 50 if(quantita<1){ 51 alert("Quantità minima 1"); 52 } else{ 53 axios.put('https://www.appalo.it/5E/GruppoC/smartMarkets/webServices/prodottiLista.php', { 54 params: { 55 codice: codice, 56 quantita: quantita, 57 email: email 58 } 59 }) 60 } 61 } 62 } 63 }); 64 </pre>		

login.php	script.js	prodottiLista.php	
-----------	-----------	-------------------	--

```

55     console.log(resultato.data.prodotti);
56   });
57 }
58 });
59 });
60 function updateProdotto(i){
61
62   var codice = document.getElementById("codice-" + i).value;
63   var quantita = document.getElementById("quantita-" + i).value;
64   var email = document.getElementById("email-" + i).value;
65   if(quantita<1){
66     alert("Valore quantità non valido");
67   }else{
68     $.ajax({
69       url: 'webServices/updateProdottoLista.php',
70       type: "POST",
71       data: {
72         email: email,
73         codice: codice,
74         quantita: quantita
75       },
76       success: function(data) {
77         $("#nome").html(name);
78         $("#quantita").html(quantita);
79         document.getElementById('pop').style.display="block";
80       },
81       error: function(jXHR, textStatus, errorThrown) {
82         alert(errorThrown + "banana 2");
83       }
84     });
85   }
86 }
87 </script>
88

```

login.php	script.js	prodottiLista.php	supermercati.php
-----------	-----------	-------------------	------------------

```

1 <?php
2 session_start();
3 ?>
4 <div id="supermarkets" class="supermarkets">
5   <p id="errore"> </p>
6   <div class="supermarket" v-for="(supermercato,index) in supermercati">
7     <div class="supermarket-texts">
8       <h3 class="supermarket-name"> {{ supermercato.nome }} </h3>
9       <p class="supermarket-text"> {{ supermercato.indirizzo }} </p>
10      <p class="supermarket-text"> {{ supermercato.prezzo }} € </p>
11      <p class="supermarket-text"> {{ supermercato.distanza }} Km </p>
12      <p class="supermarket-text"> Valutazione: {{ supermercato.valutazione }}>/10 </p>
13    </div>
14  </div>
15 </div>
16 <script>
17 var app = new Vue({
18   el: '#supermarkets',
19   data: {
20     errore: '',
21     supermercati: [],
22     email: '<?php echo $_SESSION['email']?>'
23   },
24   methods: {
25     caricaSupermercati(lat, lon) {
26       console.log(this.email);
27       axios.get('https://www.appalo.it/5E/GruppoC/smartMarkets/webServices/getSupermercati.php', {
28         params: {
29           email: this.email,
30           latitudine: lat,
31           longitudine: lon
32         }
33       }
34     }.then(function(resultato) {
35       if(resultato.data.resultato !=0){

```

login.php	script.js	prodottiLista.php	supermercati.php
-----------	-----------	-------------------	------------------

```

24 methods: {
25   caricaSupermercati(lat, lon) {
26     console.log(this.email);
27     axios.get('https://www.appalo.it/5E/GruppoC/smartMarkets/webServices/getSupermercati.php', {
28       params: {
29         email: this.email,
30         latitudine: lat,
31         longitudine: lon
32       }
33     })
34   ).then(function(risultato) {
35     if(risultato.data.risultato !=0){
36       alert("Nessun oggetto nella lista");
37     }else{
38       app.supermercati = risultato.data.supermercati;
39       console.log(risultato.data.supermercati);
40     }
41   });
42 }
43 }
44 });
45
46 $(document).ready(function() {
47   var x = document.getElementById("errore");
48
49   if (navigator.geolocation) {
50     navigator.geolocation.getCurrentPosition(showPosition);
51   } else {
52     x.innerHTML = "Geolocation is not supported by this browser.";
53   }
54
55   function showPosition(position) {
56     console.log(position.coords.latitude);
57     console.log(position.coords.longitude);
58     app.caricaSupermercati(position.coords.latitude, position.coords.longitude);
59   }
60
61 });
62 </script>
63

```

login.php	script.js	prodottiLista.php	supermercati.php
-----------	-----------	-------------------	------------------

```

30       latitudine: lat,
31       longitudine: lon
32     }
33   }
34   ).then(function(risultato) {
35     if(risultato.data.risultato !=0){
36       alert("Nessun oggetto nella lista");
37     }else{
38       app.supermercati = risultato.data.supermercati;
39       console.log(risultato.data.supermercati);
40     }
41   });
42 }
43 }
44 });
45
46 $(document).ready(function() {
47   var x = document.getElementById("errore");
48
49   if (navigator.geolocation) {
50     navigator.geolocation.getCurrentPosition(showPosition);
51   } else {
52     x.innerHTML = "Geolocation is not supported by this browser.";
53   }
54
55   function showPosition(position) {
56     console.log(position.coords.latitude);
57     console.log(position.coords.longitude);
58     app.caricaSupermercati(position.coords.latitude, position.coords.longitude);
59   }
60
61 });
62 </script>
63

```

login.php	script.js	prodottiLista.php — smartMarkets	supermercati.php	prodottiLista.php — smartMarkets/...
-----------	-----------	----------------------------------	------------------	--------------------------------------

```

1 <?php
2
3 require_once('db_connection.php');
4
5 if (isset($_REQUEST['email'])) {
6     $email = $_REQUEST['email'];
7
8
9     class Library {
10
11         }
12
13
14     class Book {
15
16         }
17     $library = new Library();
18     $query = "SELECT codiceProdotto, quantita, nome, immagine FROM `novelProdottoListaSpesa`, `novelProdotto` WHERE emailUtente = '$email'";
19     $rec = mysqli_query($conn, $query) or die($query . "<br>" . mysqli_error($conn));
20     $num = mysqli_num_rows($rec);
21     if ($num == 0) {
22         $library->prodotti = [];
23         $library->risultato = -1;
24     } else {
25         $i = 0;
26
27         while ($array = mysqli_fetch_array($rec)) {
28             $books[$i] = new Book();
29
30             $books[$i]->ean = $array['codiceProdotto'];
31
32             $books[$i]->quantita = $array['quantita'];
33
34             $books[$i]->title = $array['nome'];
35
36         }
37
38         $i++;
39     }
40
41     $library->prodotti = $books;
42     $library->risultato = 0;
43 }
44 echo json_encode($library);
45
46
47
48
49
50 // MyClass.php
51

```

login.php	script.js	prodottiLista.php — smartMarkets	supermercati.php	prodottiLista.php — smartMarkets/...
-----------	-----------	----------------------------------	------------------	--------------------------------------

```

18 $query = "SELECT codiceProdotto, quantita, nome, immagine FROM `novelProdottoListaSpesa`, `novelProdotto` WHERE emailUtente = '$email'";
19 $rec = mysqli_query($conn, $query) or die($query . "<br>" . mysqli_error($conn));
20 $num = mysqli_num_rows($rec);
21 if ($num == 0) {
22     $library->prodotti = [];
23     $library->risultato = -1;
24 } else {
25     $i = 0;
26
27     while ($array = mysqli_fetch_array($rec)) {
28         $books[$i] = new Book();
29
30         $books[$i]->ean = $array['codiceProdotto'];
31
32         $books[$i]->quantita = $array['quantita'];
33
34         $books[$i]->title = $array['nome'];
35
36         $books[$i]->images[] = $array['immagine'];
37
38         $i++;
39     }
40
41     $library->prodotti = $books;
42     $library->risultato = 0;
43 }
44 echo json_encode($library);
45
46
47
48
49
50 // MyClass.php
51

```

login.php	script.js	prodottiLista.php	supermercati.php	getSupermercati.php
<pre> 1 <?php 2 3 require_once('db_connection.php'); 4 5 if (isset(\$_REQUEST['email'])){ 6 \$email = \$_REQUEST['email']; 7 8 \$query = "SELECT codiceProdotto, quantita FROM `novelProdottoListaSpesa` WHERE emailUtente = '".\$email."' ORDER BY codiceProdotto"; 9 \$rec = mysqli_query(\$conn, \$query) or die(\$query . "
" . mysqli_error(\$conn)); 10 \$num = mysqli_num_rows(\$rec); 11 if (\$num == 0) { 12 \$library->supermercati = []; 13 \$library->risultato = -1; 14 } else { 15 \$stringaProdotti = ""; 16 \$iQuantita=0; 17 while (\$array = mysqli_fetch_array(\$rec)) { 18 \$quantita[\$array['codiceProdotto']] = \$array['quantita']; 19 if(\$iQuantita==0){ 20 \$stringaProdotti = \$stringaProdotti.\$array['codiceProdotto']; 21 }else{ 22 \$stringaProdotti = \$stringaProdotti.", ".\$array['codiceProdotto']; 23 } 24 \$iQuantita++; 25 } 26 } 27 28 class Json { 29 30 } 31 32 33 class Supermercato { 34 35 36 37 38 39 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 179 180 181 182 183 184 185 186 187 188 189 189 190 191 192 193 194 195 196 197 197 198 199 199 200 201 202 203 204 205 206 207 207 208 209 209 210 211 212 213 214 215 216 216 217 218 218 219 219 220 221 222 223 224 225 226 226 227 227 228 228 229 229 230 231 232 233 234 235 235 236 236 237 237 238 238 239 239 240 241 242 243 244 245 245 246 246 247 247 248 248 249 249 250 251 252 253 254 255 256 257 258 259 259 260 260 261 261 262 262 263 263 264 264 265 265 266 266 267 267 268 268 269 269 270 270 271 271 272 272 273 273 274 274 275 275 276 276 277 277 278 278 279 279 280 280 281 281 282 282 283 283 284 284 285 285 286 286 287 287 288 288 289 289 290 290 291 291 292 292 293 293 294 294 295 295 296 296 297 297 298 298 299 299 300 300 301 301 302 302 303 303 304 304 305 305 306 306 307 307 308 308 309 309 310 310 311 311 312 312 313 313 314 314 315 315 316 316 317 317 318 318 319 319 320 320 321 321 322 322 323 323 324 324 325 325 326 326 327 327 328 328 329 329 330 330 331 331 332 332 333 333 334 334 335 335 336 336 337 337 338 338 339 339 340 340 341 341 342 342 343 343 344 344 345 345 346 346 347 347 348 348 349 349 350 350 351 351 352 352 353 353 354 354 355 355 356 356 357 357 358 358 359 359 360 360 361 361 362 362 363 363 364 364 365 365 366 366 367 367 368 368 369 369 370 370 371 371 372 372 373 373 374 374 375 375 376 376 377 377 378 378 379 379 380 380 381 381 382 382 383 383 384 384 385 385 386 386 387 387 388 388 389 389 390 390 391 391 392 392 393 393 394 394 395 395 396 396 397 397 398 398 399 399 400 400 401 401 402 402 403 403 404 404 405 405 406 406 407 407 408 408 409 409 410 410 411 411 412 412 413 413 414 414 415 415 416 416 417 417 418 418 419 419 420 420 421 421 422 422 423 423 424 424 425 425 426 426 427 427 428 428 429 429 430 430 431 431 432 432 433 433 434 434 435 435 436 436 437 437 438 438 439 439 440 440 441 441 442 442 443 443 444 444 445 445 446 446 447 447 448 448 449 449 450 450 451 451 452 452 453 453 454 454 455 455 456 456 457 457 458 458 459 459 460 460 461 461 462 462 463 463 464 464 465 465 466 466 467 467 468 468 469 469 470 470 471 471 472 472 473 473 474 474 475 475 476 476 477 477 478 478 479 479 480 480 481 481 482 482 483 483 484 484 485 485 486 486 487 487 488 488 489 489 490 490 491 491 492 492 493 493 494 494 495 495 496 496 497 497 498 498 499 499 500 500 501 501 502 502 503 503 504 504 505 505 506 506 507 507 508 508 509 509 510 510 511 511 512 512 513 513 514 514 515 515 516 516 517 517 518 518 519 519 520 520 521 521 522 522 523 523 524 524 525 525 526 526 527 527 528 528 529 529 530 530 531 531 532 532 533 533 534 534 535 535 536 536 537 537 538 538 539 539 540 540 541 541 542 542 543 543 544 544 545 545 546 546 547 547 548 548 549 549 550 550 551 551 552 552 553 553 554 554 555 555 556 556 557 557 558 558 559 559 560 560 561 561 562 562 563 563 564 564 565 565 566 566 567 567 568 568 569 569 570 570 571 571 572 572 573 573 574 574 575 575 576 576 577 577 578 578 579 579 580 580 581 581 582 582 583 583 584 584 585 585 586 586 587 587 588 588 589 589 590 590 591 591 592 592 593 593 594 594 595 595 596 596 597 597 598 598 599 599 600 600 601 601 602 602 603 603 604 604 605 605 606 606 607 607 608 608 609 609 610 610 611 611 612 612 613 613 614 614 615 615 616 616 617 617 618 618 619 619 620 620 621 621 622 622 623 623 624 624 625 625 626 626 627 627 628 628 629 629 630 630 631 631 632 632 633 633 634 634 635 635 636 636 637 637 638 638 639 639 640 640 641 641 642 642 643 643 644 644 645 645 646 646 647 647 648 648 649 649 650 650 651 651 652 652 653 653 654 654 655 655 656 656 657 657 658 658 659 659 660 660 661 661 662 662 663 663 664 664 665 665 666 666 667 667 668 668 669 669 670 670 671 671 672 672 673 673 674 674 675 675 676 676 677 677 678 678 679 679 680 680 681 681 682 682 683 683 684 684 685 685 686 686 687 687 688 688 689 689 690 690 691 691 692 692 693 693 694 694 695 695 696 696 697 697 698 698 699 699 700 700 701 701 702 702 703 703 704 704 705 705 706 706 707 707 708 708 709 709 710 710 711 711 712 712 713 713 714 714 715 715 716 716 717 717 718 718 719 719 720 720 721 721 722 722 723 723 724 724 725 725 726 726 727 727 728 728 729 729 730 730 731 731 732 732 733 733 734 734 735 735 736 736 737 737 738 738 739 739 740 740 741 741 742 742 743 743 744 744 745 745 746 746 747 747 748 748 749 749 750 750 751 751 752 752 753 753 754 754 755 755 756 756 757 757 758 758 759 759 760 760 761 761 762 762 763 763 764 764 765 765 766 766 767 767 768 768 769 769 770 770 771 771 772 772 773 773 774 774 775 775 776 776 777 777 778 778 779 779 780 780 781 781 782 782 783 783 784 784 785 785 786 786 787 787 788 788 789 789 790 790 791 791 792 792 793 793 794 794 795 795 796 796 797 797 798 798 799 799 800 800 801 801 802 802 803 803 804 804 805 805 806 806 807 807 808 808 809 809 810 810 811 811 812 812 813 813 814 814 815 815 816 816 817 817 818 818 819 819 820 820 821 821 822 822 823 823 824 824 825 825 826 826 827 827 828 828 829 829 830 830 831 831 832 832 833 833 834 834 835 835 836 836 837 837 838 838 839 839 840 840 841 841 842 842 843 843 844 844 845 845 846 846 847 847 848 848 849 849 850 850 851 851 852 852 853 853 854 854 855 855 856 856 857 857 858 858 859 859 860 860 861 861 862 862 863 863 864 864 865 865 866 866 867 867 868 868 869 869 870 870 871 871 872 872 873 873 874 874 875 875 876 876 877 877 878 878 879 879 880 880 881 881 882 882 883 883 884 884 885 885 886 886 887 887 888 888 889 889 890 890 891 891 892 892 893 893 894 894 895 895 896 896 897 897 898 898 899 899 900 900 901 901 902 902 903 903 904 904 905 905 906 906 907 907 908 908 909 909 910 910 911 911 912 912 913 913 914 914 915 915 916 916 917 917 918 918 919 919 920 920 921 921 922 922 923 923 924 924 925 925 926 926 927 927 928 928 929 929 930 930 931 931 932 932 933 933 934 934 935 935 936 936 937 937 938 938 939 939 940 940 941 941 942 942 943 943 944 944 945 945 946 946 947 947 948 948 949 949 950 950 951 951 952 952 953 953 954 954 955 955 956 956 957 957 958 958 959 959 960 960 961 961 962 962 963 963 964 964 965 965 966 966 967 967 968 968 969 969 970 970 971 971 972 972 973 973 974 974 975 975 976 976 977 977 978 978 979 979 980 980 981 981 982 982 983 983 984 984 985 985 986 986 987 987 988 988 989 989 990 990 991 991 992 992 993 993 994 994 995 995 996 996 997 997 998 998 999 999 1000 1000 1001 1001 1002 1002 1003 1003 1004 1004 1005 1005 1006 1006 1007 1007 1008 1008 1009 1009 1010 1010 1011 1011 1012 1012 1013 1013 1014 1014 1015 1015 1016 1016 1017 1017 1018 1018 1019 1019 1020 1020 1021 1021 1022 1022 1023 1023 1024 1024 1025 1025 1026 1026 1027 1027 1028 1028 1029 1029 1030 1030 1031 1031 1032 1032 1033 1033 1034 1034 1035 1035 1036 1036 1037 1037 1038 1038 1039 1039 1040 1040 1041 1041 1042 1042 1043 1043 1044 1044 1045 1045 1046 1046 1047 1047 1048 1048 1049 1049 1050 1050 1051 1051 1052 1052 1053 1053 1054 1054 1055 1055 1056 1056 1057 1057 1058 1058 1059 1059 1060 1060 1061 1061 1062 1062 1063 1063 1064 1064 1065 1065 1066 1066 1067 1067 1068 1068 1069 1069 1070 1070 1071 1071 1072 1072 1073 1073 1074 1074 1075 1075 1076 1076 1077 1077 1078 1078 1079 1079 1080 1080 1081 1081 1082 1082 1083 1083 1084 1084 1085 1085 1086 1086 1087 1087 1088 1088 1089 1089 1090 1090 1091 1091 1092 1092 1093 1093 1094 1094 1095 1095 1096 1096 1097 1097 1098 1098 1099 1099 1100 1100 1101 1101 1102 1102 1103 1103 1104 1104 1105 1105 1106 1106 1107 1107 1108 1108 1109 1109 1110 1110 1111 1111 1112 1112 1113 1113 1114 1114 1115 1115 1116 1116 1117 1117 1118 1118 1119 1119 1120 1120 1121 1121 1122 1122 1123 1123 1124 1124 1125 1125 1126 1126 1127 1127 1128 1128 1129 1129 1130 1130 1131 1131 1132 1132 1133 1133 1134 1134 1135 1135 1136 1136 1137 1137 1138 1138 1139 1139 1140 1140 1141 1141 1142 1142 1143 1143 1144 1144 1145 1145 1146 1146 1147 1147 1148 1148 1149 1149 1150 1150 1151 1151 1152 1152 1153 1153 1154 1154 1155 1155 1156 1156 1157 1157 1158 1158 1159 1159 1160 1160 1161 1161 1162 1162 1163 1163 1164 1164 1165 1165 1166 1166 1167 1167 1168 1168 1169 1169 1170 1170 1171 1171 1172 1172 1173 1173 1174 1174 1175 1175 1176 1176 1177 1177 1178 1178 1179 1179 1180 1180 1181 1181 1182 1182 1183 1183 1184 1184 1185 1185 1186 1186 1187 1187 1188 1188 1189 1189 1190 1190 1191 1191 1192 1192 1193 1193 1194 1194 1195 1195 1196 1196 1197 1197 1198 1198 1199 1199 1200 1200 1201 1201 1202 1202 1203 1203 1204 1204 1205 1205 1206 1206 1207 1207 1208 1208 1209 1209 1210 1210 1211 1211 1212 1212 1213 1213 1214 1214 1215 1215 1216 1216 1217 1217 1218 1218 1219 1219 1220 1220 1221 1221 1222 1222 1223 1223 1224 1224 1225 1225 1226 1226 1227 1227 1228 1228 1229 1229 1230 1230 1231 1231 1232 1232 1233 1233 1234 1234 1235 1235 1236 1236 1237 1237 1238 1238 1239 1239 1240 1240 1241 1241 1242 1242 1243 1243 1244 1244 1245 1245 1246 1246 1247 1247 1248 1248 1249 1249 1250 1250 1251 1251 1252 1252 1253 1253 1254 1254 1255 1255 1256 1256 1257 1257 1258 1258 1259 1259 1260 1260 1261 1261 1262 1262 1263 1263 1264 1264 1265 1265 1266 1266 1267 1267 1268 1268 1269 1269 1270 1270 1271 1271 1272 1272 1273 1273 1274 1274 1275 1275 1276 1276 1277 1277 1278 1278 1279 1279 1280 1280 1281 1281 1282 1282 1283 1283 1284 1284 1285 1285 1286 1286 1287 1287 1288 1288 1289 1289 1290 1290 1291 1291 1292 1292 1293 1293 1294 1294 1295 1295 1296 1296 1297 1297 1298 1298 1299 1299 1300 1300 1301 1301 1302 1302 1303 1303 1304 1304 1305 1305 1306 1306 1307 1307 1308 1308 1309 1309 1310 1310 1311 1311 1312 1312 1313 1313 1314 1314 1315 1315 1316 1316 1317 1317 1318 1318 1319 1319 1320 1320 1321 1321 1322 1322 1323 1323 1324 1324 1325 1325 1326 1326 1327 1327 1328 1328 1329 1329 1330 1330 1331 1331 1332 1332 1333 1333 1334 1334 1335 1335 1336 1336 1337 1337 1338 1338 1339 1339 1340 1340 1341 1341 1342 1342 1343 1343 1344 1344 1345 1345 1346 1346 1347 1347 1348 1348 1349 1349 1350 1350 1351 1351 1352 1352 1353 1353 1354 1354 1355 1355 1356 1356 1357 1357 1358 1358 1359 1359 1360 1360 1361 1361 1362 1362 1363 1363 1364 1364 1365 1365 1366 1366 1367 1367 1368 1368 1369 1369 1370 1370 1371 1371 1372 1372 1373 1373 1374 1374 1375 1375 1376 1376 1377 1377 1378 1378 1379 1379 1380 1380 1381 1381 1382 1382 1383 1383 1384 1384 1385 1385 1386 1386 1387 1387 1388 1388</pre>				

login.php	script.js	prodottiLista.php	supermercati.php	getSupermercati.php
-----------	-----------	-------------------	------------------	---------------------

```

56     $prezzo = 0;
57
58     $query = "SELECT * FROM `novelProdottoSupermercato`, `novelSupermercato` WHERE novelProdottoSupermercato.codiceProdotto IN
59     $rec2 = mysqli_query($conn, $query) or die($query . "<br>" . mysqli_error($conn));
60     $num2 = mysqli_num_rows($rec2);
61     while ($array2 = mysqli_fetch_array($rec2)) {
62         $prezzo = $prezzo + ($array2['prezzo'] * $quantita[$array2['codiceProdotto']]);
63     }
64
65     $supermercato[$i]->prezzo = $prezzo;
66
67     $query = "SELECT AVG(voto) as valutazione FROM `novelValutazione` WHERE novelValutazione.idUtenteBackOffice = '". $array['
68     $rec2 = mysqli_query($conn, $query) or die($query . "<br>" . mysqli_error($conn));
69     $num2 = mysqli_num_rows($rec2);
70     while ($array2 = mysqli_fetch_array($rec2)) {
71         $valutazione = $array2['valutazione'];
72     }
73
74     $supermercato[$i]->valutazione = number_format($valutazione, 1);
75
76     $earthRadius = 6371;
77     $latFrom = deg2rad($_REQUEST['latitudine']);
78     $lonFrom = deg2rad($_REQUEST['longitudine']);
79     $latTo = deg2rad($array['latitudine']);
80     $lonTo = deg2rad($array['longitudine']);
81
82     $latDelta = $latTo - $latFrom;
83     $lonDelta = $lonTo - $lonFrom;
84
85     $angle = 2 * asin(sqrt(pow(sin($latDelta / 2), 2) + cos($latFrom) * cos($latTo) * pow(sin($lonDelta / 2), 2)));
86     $distance = $angle * $earthRadius;
87     $supermercato[$i]->distanza = number_format($distance, 2);
88     $i++;
89 }
$cjson->supermercati = $supermercato;

```

login.php	script.js	prodottiLista.php	supermercati.php	getSupermercati.php
-----------	-----------	-------------------	------------------	---------------------

```

67     $query = "SELECT AVG(voto) as valutazione FROM `novelValutazione` WHERE novelValutazione.idUtenteBackOffice = '". $array['
68     $rec2 = mysqli_query($conn, $query) or die($query . "<br>" . mysqli_error($conn));
69     $num2 = mysqli_num_rows($rec2);
70     while ($array2 = mysqli_fetch_array($rec2)) {
71         $valutazione = $array2['valutazione'];
72     }
73
74     $supermercato[$i]->valutazione = number_format($valutazione, 1);
75
76     $earthRadius = 6371;
77     $latFrom = deg2rad($_REQUEST['latitudine']);
78     $lonFrom = deg2rad($_REQUEST['longitudine']);
79     $latTo = deg2rad($array['latitudine']);
80     $lonTo = deg2rad($array['longitudine']);
81
82     $latDelta = $latTo - $latFrom;
83     $lonDelta = $lonTo - $lonFrom;
84
85     $angle = 2 * asin(sqrt(pow(sin($latDelta / 2), 2) + cos($latFrom) * cos($latTo) * pow(sin($lonDelta / 2), 2)));
86     $distance = $angle * $earthRadius;
87     $supermercato[$i]->distanza = number_format($distance, 2);
88     $i++;
89 }
$json->supermercati = $supermercato;
$json->risultato = 0;
90
91 }
echo json_encode($json);
92
93 }
94
95 }
96
97
98
99 // MyClass.php
100

```