What's In My Fridge

Dipartimento di Matematica, Informatica,
Scienze Fisiche e Scienze della Terra
Corso di Laurea Triennale in informatica
Progetto di programmazione Web e Mobile
A.A. 2021/2022

Docente: Andrea Nucita
Studenti: Erika Casablanca, Carmelo Trifirò

Indice

1 What's In My Fridge

1.1 Introduzione

What's In My Fridge (WIMF) è una web app che ha come obiettivo quello di aiutare gli utenti a gestire il proprio frigorifero, in modo da limitare gli sprechi di cibo, dovuti a possibili dimenticanze.

1.2 Funzionalità

WIMF permette ad ogni utente di registrarsi e di gestire il proprio frigorifero, o uno condiviso con familiari o coinquilini, il tutto con un'interfaccia semplice e intuitiva. Infatti subito dopo la registrazione o il login l'utente può creare un nuovo frigo o collegarsi a uno già esistente tramite un ID. Una volta fatto questo l'utente può accedere all'area My Fridge per iniziare a inserire i suoi alimenti all'interno del frigo tramite un semplice form dov'è possibile inserire il nome dell'alimento, la quantità o il peso e la data di scadenza. Fatto questo la web app si occuperà di inserire gli alimenti nel frigo ordinandoli per data di scadenza e offrendo un ulteriore indicatore dato dallo status.

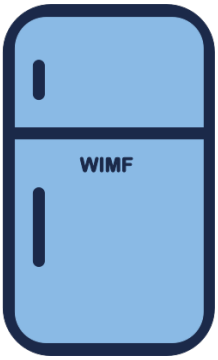

Figure 1: *Logo di What's In My Fridge*

2 Implementazione

Per realizzare What's In My Fridge abbiamo utilizzato vari linguaggi:

- **PHP**, utilizzato principalmente per interagire con il database tramite query in SQL;
- **HTML**, utilizzato per creare la navbar e il footer e in generale ogni pagina del sito;

- **JavaScript**, utilizzato per rendere le pagine dinamiche, sfruttato principalmente nell'area My Fridge;
- **CSS**, utilizzato per modificare il modo in cui viene visualizzata la web app;

Come servizio web abbiamo implementato un servizio RESTFUL.

2.1 Il database

Prima di parlare nello specifico del codice PHP, intruduciamo il database e la sua struttura. Il database di What's In My Fridge è formato da 4 tabelle:

- **users**: id (chiave primaria auto incrementata), email, password e `id_fridge` (chiave esterna);
- **food**: id (chiave primaria auto incrementata), `nome_cibo`;
- **fridge**: id (chiave primaria auto incrementata);
- **contain**: `id_frigo` (chiave esterna), `id_cibo` (chiave esterna), quantita, grammi, `data_scadenza`, `id_riga` (chiave primaria auto incrementata);

Parte del codice PHP ci permette di effettuare delle query in SQL su queste tabelle, in modo da effettuare le operazioni "CRUD", nel nostro caso sono:

- **Creazione del frigo**
- **Lettura dei dati riguardanti il cibo presente nel frigo**
- **Aggiornamento delle quantità o peso di cibo presenti nel frigo**
- **Cancellazione degli alimenti nel frigo**

2.2 `db_config.php`

Questo file viene utilizzato come supporto per `db_coding.php`, questo infatti contiene tutte le variabili utili al collegamento al database che vengono utilizzate in ogni query.

```
<?php
    $dbhost="localhost";
    $dbname="what_s_in_my_fridge";
    $dbuser="admin";
    $dbpassword="";
?>
```

2.3 db_coding.php

Questo file si occupa di effettuare tutte le possibili query utili alla web app, quindi registrazione e login dell'utente, creazione di un nuovo frigo, inserimento di un nuovo alimento, ecc...

Tutto si basa sui **PDO** (PHP Data Objects), preferiti a MySQLi in quanto possono essere utilizzati con 12 diversi sistemi di database, mentre MySQLi funziona solo con i database MySQL.

```
try{
        $conn = new
            PDO("mysql:host=".$GLOBALS['dbhost'].";dbname=".$GLOBALS['dbname'],
            $GLOBALS['dbuser'],$GLOBALS['dbpassword']);
        $conn->setAttribute(PDO::ATTR_ERRMODE,
            PDO::ERRMODE_EXCEPTION);
        $stmt = $conn->prepare("SELECT * FROM fridge WHERE id = ?");
        $stmt->execute([$id]);
        $res = $stmt->fetch();
}catch(PDOException $e){
        return array("KO", "Errore durante la creazione del frigo");
    }
```

Come si evince da questo snippet, viene prima effettuata la connessione al database e poi viene eseguita la query con l'eventuale utilizzo di variabili, poi tramite il metodo fetch() estrapoliamo il risultato della query; usiamo try e catch per catturare eventuali errori durante la connessione.

```
function getContainedFood($id_fridge) //funzione che restituisce
    tutte le informazioni del cibo contenuto in un frigo sfruttando
{
    try{
        $conn = new
            PDO("mysql:host=".$GLOBALS['dbhost'].";dbname=".$GLOBALS['dbname'],
            $GLOBALS['dbuser'],$GLOBALS['dbpassword']);
        $conn->setAttribute(PDO::ATTR_ERRMODE,
            PDO::ERRMODE_EXCEPTION);
        $stmt = $conn->prepare("SELECT nome_cibo, quantita, grammi,
            data_scadenza,id_riga FROM food INNER JOIN contain ON
            food.id = contain.id_cibo WHERE id_frigo = ? ORDER BY
            data_scadenza"); //inner join per ottenere tutti i dati
            relativi al cibo contenuto nel frigo
        $stmt->execute([$id_fridge]);
```

```
        $res = $stmt->fetchAll();              //estrae il
            risultato della query

        if($res !=null)
            return $res;
        else
            return NULL;
    }catch(PDOException $e){
        return NULL;
    }
}
```

2.4 navbar.html

Questo file viene utilizzato per creare una navbar in tutta la web app. Essa é implementata come una lista puntata ma che, tramite il CSS, appare con tutte le caratteristiche di una classica navbar. Oltre alla navbar é presente anche un footer con i nostri contatti. Questo file importa un file CSS esterno, mystyle.css, file che gestisce anche il CSS del body di tutte le pagine del sito.

2.5 homepage.php

Questa pagina é una pagina di riepilogo e presentazione con la possibilitá di visitare le altre pagine tramite la navbar presentata prima. Il CSS di questa pagina é un CSS interno, che gestisce l'allineamento dei titoli, del testo e del logo della Web App.

2.6 register.php

In questa pagina é presente il form di registrazione dell'utente, formato da tre campi, ossia **email**, **password** e **conferma password**.

Nel momento in cui viene premuto il pulsante di registrazione viene controllato che sia stata inserita una mail valida, che sia stato inserito qualcosa nel campo password e che il campo password e conferma password siano effettivamente uguali, altrimenti riporta vari messaggi di errore.

Se é stato inserito tutto correttamente viene effettuata la registrazione tramite l'inserimento nella tabella **users** di un nuovo utente con la mail e la password, che viene criptata proprio in fase di registrazione prima di essere inserita nel database tramite la funzione:

```
password_hash($_POST['password'], PASSWORD_DEFAULT);
```

Questo metodo peró funziona solo per il criptaggio, infatti non é possibile estrapolare poi la stringa corrispondente alla password ma é comunque possibile controllare in login se la password inserita é quella salvata nel database. Il CSS del form di registrazione, come anche quello del login, é scritto all'interno del file **formtype.css**, utilizzato appositamente per questo genere di form.

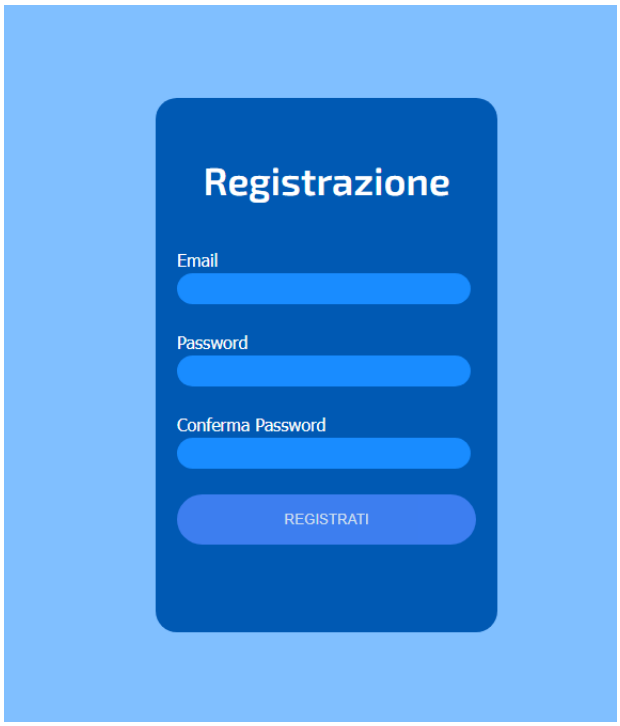

Figure 2: *Form di registrazione*

2.7 login.php

Questa pagina gestisce invece il login dell'utente: quando questo prova ad accedere inserendo la sua mail e password, vengono effettuate delle query che controllano se esiste un utente registrato nella tabella **users** con quella mail e quella password, questa viene controllata con la funzione *password_verify()* che prende come input la password inserita in login e la password criptata da controllare; se la password inserita ha un riscontro con quella criptata ritorna true, false altrimenti.

Fatto questo l'utente viene poi reindirizzato alla pagina **myFridge.php**. Come scritto prima, il CSS esterno di questa pagina all'interno del file **form-type.css**.

2.8 index.php

In questa pagina abbiamo semplicemente le impostazioni dell'utente: in questo caso si dá all'utente la possibilitá di modificare la propria password (facendo un update della tabella users), effettuare il logout e di visualizzare e modificare il suo codice frigo (con un altro update sulla medesima tabella), cosí da poterlo condividere con i suoi familiari o coinquilini che a loro volta possono modificare il proprio codice frigo per renderlo corrispondente a quello giá creato dal primo utente.

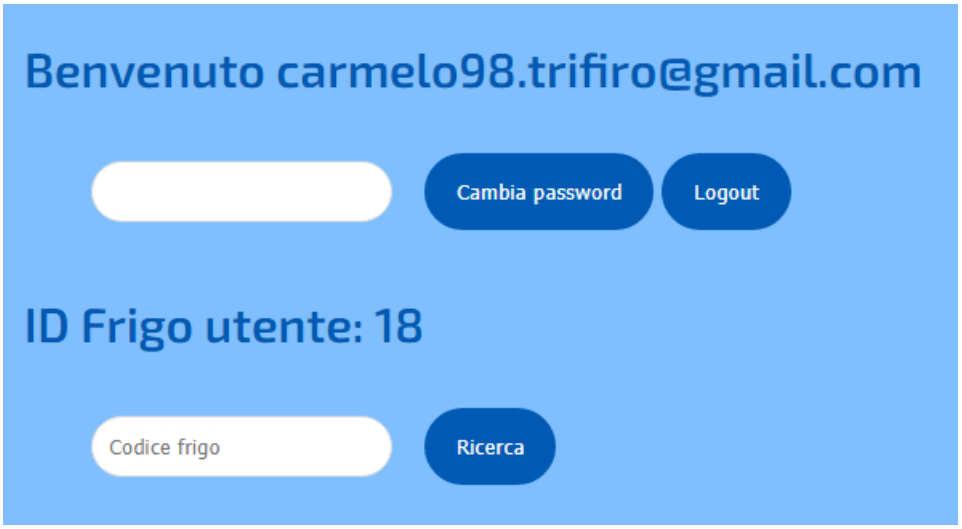

Figure 3: *Pagina impostazioni account*

2.9 myFridge.php

Questa pagina racchiude l'essenza della Web App: alla creazione di un nuovo account verrá chiesto all'utente di creare un nuovo frigo (inserendo un nuovo frigo nella tabella fridge) o di inserire un codice di un frigo gi esistente (aggiornando la tabella users), una volta fatto sará possibile visualizzare il proprio frigo e inserire al suo interno i vari alimenti.
Per ogni alimento inserito vanno specificati il nome (se non presente nel menú a tendina é possibile inserirlo manualmente nell'area di testo dedicata), la quantitá o il peso (selezionabili tramite un radio button) e la data di scadenza. Una volta inserito l'alimento verrá inserita nella tabella contain una

nuova riga contenente l'id del frigo dell'utente, l'id dell'utente, la quantitá o il peso dell'alimento e la data di scadenza; poi per identificare l'alimento sfrutteremo il campo `id_row` per indicare l'indice della riga della tabella corrispondente.

Fatto questo, verrá mostrato l'alimento nell'Area My Fridge con tutte le sue informazioni, e in aggiunta verrá mostrato lo status dell'alimento e il pulsante di cancellazione.

Figure 4: *Screenshot di un'Area My Fridge*

Lo status non é altro che un emoji di una sferetta rossa, gialla o verde, che viene mostrata accanto all'alimento in base alla sua data di scadenza:

- Rosso per alimenti scaduti o che scadono in data odierna;
- Giallo per alimenti che scadono entro una settimana dalla data odierna;
- Verde per alimenti che scadono tra pi di una settimana rispetto alla data odierna;

```
function emoji_status(expiration_date) {
    const current_date = new Date();
                                                //funzione che
        controlla la data di scadenza dell'alimento e
        restituisce uno status con colori differenti
    const exp_date = new Date(expiration_date);
```

```
        let difference_in_time = exp_date.getTime() -
            current_date.getTime();   //variabile contenente la
            differenza in millisecondi tra la data di scadenza e la
            data odierna
        let difference_in_days = difference_in_time / (1000 * 3600
            * 24);       //calcola invece la differenza in giorni

        if (difference_in_days <0 )
                                                          //se la
            differenza  minore di 0 allora l'alimento  scaduto o
            scade oggi
            return "&#128308;";

                //ritorna il codice corrispondente all'emoji del
                cerchio rosso
        else if(7>difference_in_days>0)
                                                       //se l'alimento
            scade tra una settimana ritorna un cerchio giallo
            return "&#128993;";
        else if(difference_in_days>7)
                                                         //se scade tra
            pi di una settimana cerchio verde
            return "&#128994;";
    }
```

Il tasto cancella elimina il relativo elemento dal frigo. É anche possibile aumentare o diminuire le quantitá o grammi di un alimento giá presente nel frigo tramite i tasti **Inserisci alimento** e **Modifica quantitá**. Bisogna peró sottolineare che, affinché la modifica avvenga con successo, bisogna inserire lo stesso alimento con la stessa data di scadenza, cosí facendo e cliccando su **Inserisci alimento** verrá aumentata la quantitá o il peso dell'alimento, viceversa verrá decrementata.

Per stampare sulla pagina tutti gli alimenti con i loro dati, lo status e il tasto di cancellazione corrispondente, abbiamo usato il seguente codice in JavaScript:

```
<script>

    //script utilizzato per la stampa dinamica di tutti gli
    alimenti salvati nel frigo
        <?php
            $aliments =
```

```
        json_encode(getContainedFood($_SESSION["fridge"]));
        //la funzione restituisce l'array contenente tutti i
        cibi ma  codificato in PHP, usiamo json_encode per
        fare la conversione
    echo "const food_fridge = ". $aliments . ";\n";
                            //usiamo echo per dichiarare
        l'array in javascript

    $id_food =
        json_encode(getContainedFoodId($_SESSION["fridge"]));
    echo "const jid_food =". $id_food . ";\n";
?>

var food_table = document.getElementById('frigo');

for(let i = 0; i<food_fridge.length; i++)
{
    const tr = food_table.insertRow();
    for(let j=0;j<4;j++)
    {
        let td = tr.insertCell();
        if(j==2 && food_fridge[i][j])
            td.innerHTML = food_fridge[i][j]+"g";
                                    //se l'alimento
                salvato in grammi allora stampa la g alla fine
        else
            td.innerHTML = food_fridge[i][j];
    }

    let status = tr.insertCell();
                                              //dopo aver
        stampato tutte le informazioni allora aggiunge lo
        status alla riga corrispondente al cibo
    let emoji = emoji_status(food_fridge[i][3]);
    status.innerHTML = emoji;

    var button = document.createElement('button');

    button.className = "table-button";
    button.name = "cancel_food";
        //Crea il bottone di cancellazione dell'alimento
    button.type = "submit";
    button.value = food_fridge[i][4];
```

```
            //il valore  corrispondente all'id della riga della
            tabella contain per il bottone
        button.innerHTML = "Cancella";
        tr.appendChild(button);
            //crea automaticamente le righe contenenti i vari
            alimenti

    }
</script>
```

Il CSS relativo a questa pagina é scritto nel file **fridge.css**: questo é stato utilizzato per la visualizzazione sia del frigorifero che per tutti i "form-inline", ossia per tutte le sezioni dove vengono inseriti dei dati (eccezion fatta per registrazione e login).

3 Possibili migliorie

Tra le possibili funzionalitá sarebbe possibile inserire:

- Un registro delle attivitá con le modifiche effettuate sul frigo con data e ora delle modifiche;
- Implementazione di un sistema gerarchico per la gestione del frigo: il creatore di un nuovo frigo verrá identificato come master, gli altri utenti che vorranno condividere il frigo con lui manderanno una richiesta per avere un permesso. Pensato per evitare che gli utenti accedano ai frighi di altri utenti e cancellino alimenti.
- Un tasto oltre a quello della cancellazione dell'alimento che rimandi a una ricerca su Google con eventuali ricette con l'alimento corrispondente;
- La possibilitá di ricevere notifiche quando un alimento é prossimo alla scadenza;