

Elaborato Informatica e Telecomunicazioni

I.I.S Curie Sraffa

Emre Tugrul

Elaborato per Esame di Stato
A.S 2020/2021



I.I.S Curie Sraffa
Indirizzo informatico
Milano, Italia

Indice

1	Analisi Problema	2
2	Linguaggi di programmazione e Software utilizzati	2
3	Struttura del progetto	3
4	Struttura Piattaforma	4
4.1	Database	4
4.2	Login-Logout	5
4.3	Interfaccia Personale	9
4.3.1	Sezione della sala	9
4.3.2	Sezione della cucina	11
5	Interfaccia Admin	19
5.1	Database	19
5.2	Struttura progetto	19
5.3	Login-Logout	20
5.4	Struttura piattaforma	20
5.5	Interfaccia Home	21
5.6	Interfaccia Menu	22
6	Conclusione	24

1 Analisi Problema

Il problema si presenta in modo abbastanza generico, viene richiesta la gestione delle ordinazioni di un ristorante tramite una piattaforma online. L'accesso alla piattaforma sarà permesso solamente ai camerieri e agli chef autorizzati dal ristorante. Serviranno quindi due interfacce che dovranno comunicare tra di loro, cioè la cucina e la sala. Dovranno inviare e ricevere dati, sia tra di loro che ad un database che dovrà essere messo a disposizione ad entrambi.

Una cosa di cui ho tenuto conto è che la piattaforma non dovrà essere esposta alla clientela, di conseguenza ho cercato di renderlo il più veloce, semplice, user-friendly possibile, evitando decorazioni e abbellimenti che avrebbero rallentato il sistema.

Per gestire le ordinazioni ho pensato ai passaggi principali che si seguono durante un'ordinazione, cioè la visualizzazione dei piatti, invio dell'ordine, elaborazione della richiesta da parte della cucina, notifica e servizio del piatto preparato.

Verrà inoltre implementata una sezione di login in cui il personale potrà accedere tramite credenziali fornite dall'amministratore o dal gestore del ristorante. In base alle credenziali che verranno inserite verrà aperta dinamicamente la pagina dedicata, in base al ruolo del personale; quindi cameriere oppure cuoco.

Infine servirà un'interfaccia ADMIN da cui saranno possibili operazioni per gestire modifiche alla piattaforma (come ad esempio la modifica del menù)

2 Linguaggi di programmazione e Software utilizzati

- **Database**

Per il **Database** ho utilizzato DevServer17 che allo stesso tempo funge da HTTP server e DATABASE server.

HTTP Server è un servizio applicativo che è in grado di gestire le richieste di pagine WEB da parte di un client. Quindi è ciò che permetterà il collegamento al localhost per effettuare i vari test.

Mentre **DATABASE Server** è un servizio applicativo che permette ad altri programmi (generalmente client) di interagire con il database che viene gestito dal server.

- **Interfaccia FRONT-END**

Per **interfaccia FRONT-END** si intende la parte di programma che si andrà ad interfacciare con l'utenza, che in questo caso sarebbero i camerieri e i cuochi. Si occupa prevalentemente di prelevare dati in ingresso da parte della clientela.

- **Interfaccia BACK-END**

L'**interfaccia BACK-END** non è la parte con cui l'utente interagisce direttamente, ma serve per il corretto funzionamento dell'applicazione. Elabora i dati ingresso forniti attraverso l'interfaccia FRONT-END.

Per avere un'interfaccia accattivante ma allo stesso tempo semplice ho importato un template predefinito tramite **Bootstrap**, ovvero un insieme di modelli HTML e CSS, messi a disposizione principalmente ai programmatori per avere un'interfaccia grafica.

3 Struttura del progetto

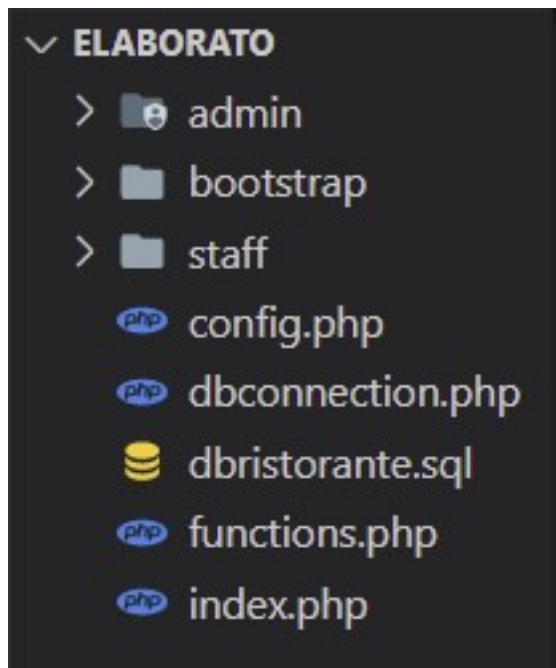


Figura 1: Struttura generale.

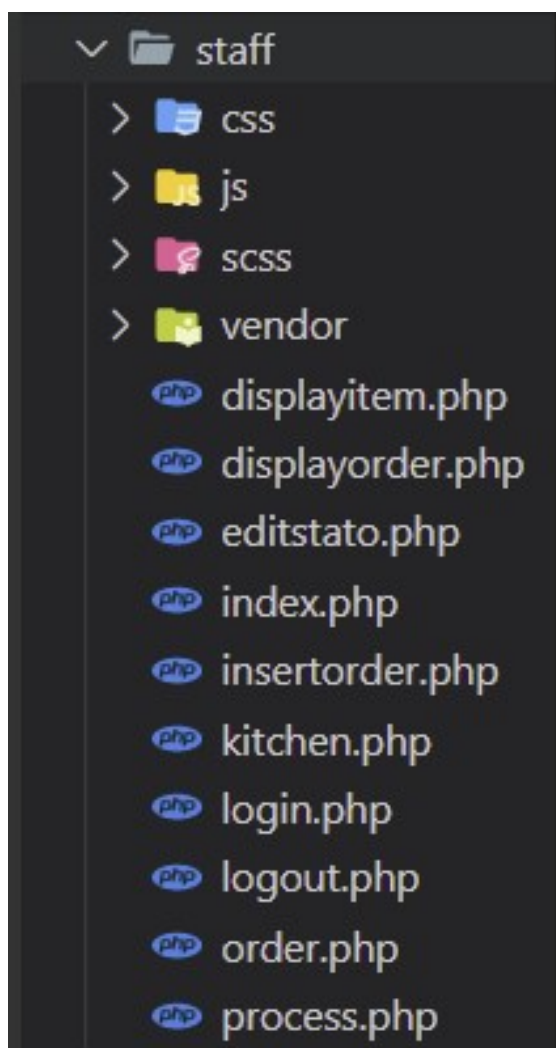


Figura 2: Struttura dello Staff.

Admin: Gestione interfaccia Admin.

Bootstrap: Strumenti parte grafica sito.

Staff: Gestione interfaccia personale.

Config: Configurazione del Database.

DBconnection: Connessione al DB.

DBristorante: Importazione Database.

Functions: Funzioni utili in più file.

Index: Pagina principale del sito.

Staff: Gestione interfaccia Admin.

CSS: Strumenti parte grafica sito.

JS: Gestione interfaccia personale.

SCSS: Configurazione del Database.

VENDOR: Connessione al DB.

DisplayItem: Importazione Database.

DisplayOrder: Funzioni utili in più file.

EditStato: Modifica stato piatti.

Index: Pagina principale del sito.

InsertOrder: Inserimento ordini.

Kitchen: Gestione cucina.

Login: Pagina di Login.

Logout: Gestione Logout.

Order: Pagina per ordinazioni.

Process: Gestione Login.

4 Struttura Piattaforma

4.1 Database

Per poter gestire tutti questi dati bisogna creare un supporto che possa immagazzinarli e gestirli. Prima di creare l'effettivo database ho creato il **modello E-R** e il relativo modello logico. Il modello E-R viene spesso utilizzato nella prima fase della progettazione di una base di dati, nella quale è necessario tradurre le informazioni risultanti dall'analisi in uno schema concettuale.

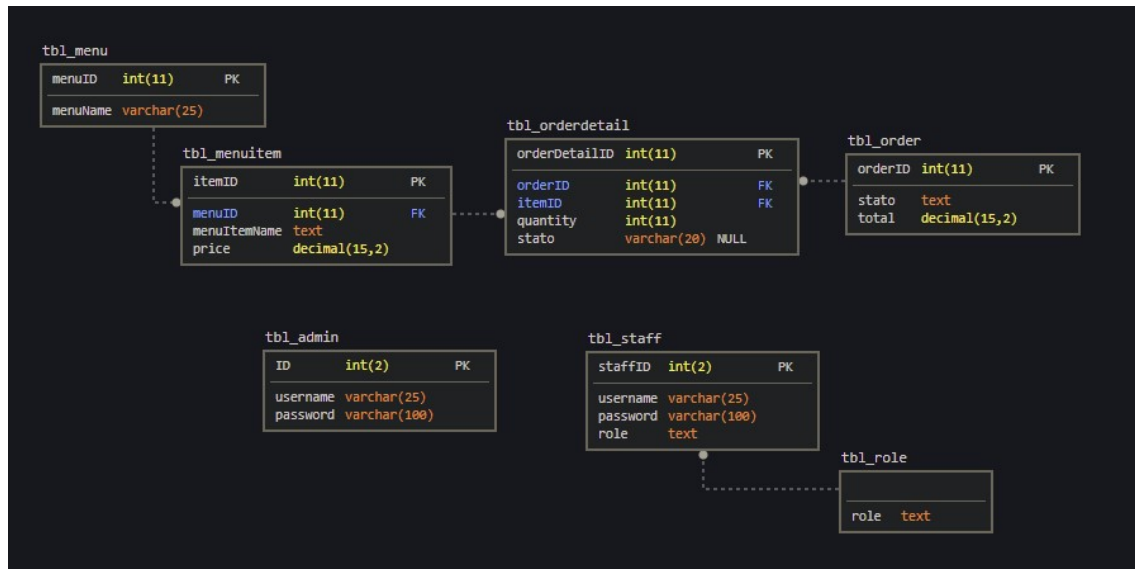


Figura 3: Modello Entità-Relazione.

La tabella **Admin** contiene un campo ID univoco, che tiene conto di ogni persona autorizzata ad entrare nel reparto Admin.

La tabella **Staff** contiene campo ID che ha la stessa funzione della precedente e inoltre ha una chiave secondaria, connessa alla tabella **Role** che contiene di default cameriere e cuoco come ruoli possibili.

Nella tabella **OrderDetail** vengono inseriti tutti gli ordini di ogni piatto con quantità e stato del piatto.

Nella tabella **Order** vengono inseriti l'effettivo ordine che poi conterrà un insieme di ordini di piatti. Contiene un ID, lo stato attuale ed il costo totale dell'ordine.

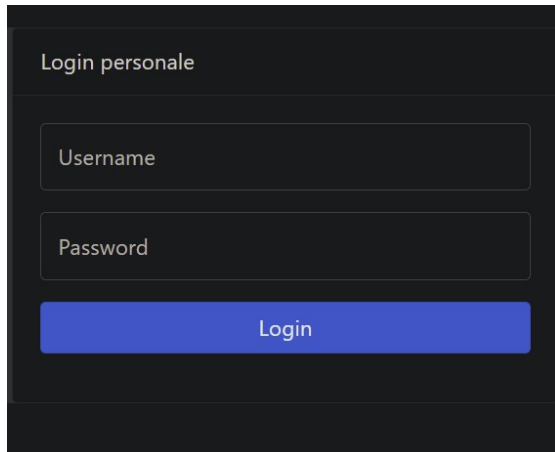
La tabella **MenuItem** contiene tutti i piatti presenti nel Menù con IDPiatto e prezzo.

La tabella **Menù** contiene il tipo di piatto con relativo ID. Nel nostro caso, un esempio può essere l'OrderID che è contenuto nella tabella **Order** e **OrderDetail**.

Nella tabella **Order** l'OrderID funge da chiave primaria, quindi identifica univocamente un record; mentre nella tabella **OrderDetail** funge da chiave secondaria, cioè collega le due tabelle e al variare del valore di OrderId nella tabella **Order**, varierà anche OrderID della tabella **OrderDetail**. Le operazioni principali saranno aggiunta, modifica, eliminazione e aggiornamento di dati. Quindi il Database verrà interrogato tramite le Query che invieremo tramite **PHP**, per ottenere le operazioni desiderate. Tutte queste tabelle sono collegate con degli INNER JOIN che sono delle operazioni che consentono di creare una nuova tabella, fusione di due o più tabelle che avranno un campo in comune.

4.2 Login-Logout

Per tenere conto di tutto il personale e per ragioni di sicurezza, servirà una sezione di Login in cui prima di poter accedere al sito del ristorante, bisognerà effettuare l'accesso. Verranno utilizzati **Username** e **Password** che saranno presenti nel Database e saranno controllati tramite PHP. Quindi verrà creata una Form tramite HTML, che conterrà Username e Password, ed un bottone che servirà per inviare i dati ed effettuare i vari controlli necessari all'accesso all'interfaccia desiderata.



The screenshot shows a login form titled "Login personale" on a dark background. It contains two input fields: "Username" and "Password". Below these fields is a blue button labeled "Login".

Nella sezione di Login ci sono due campi di Input (Username e Password) ed un bottone per inviare le credenziali.

Figura 4: Interfaccia di Login.

```
<form id="loginform">
  <div class="form-group">
    <div class="form-label-group">
      <input type="text" id="inputUsername" name="username" class="form-control" placeholder="Username" required="required" />
      <label for="inputUsername">Username</label>
    </div>
  </div>
  <div class="form-group">
    <div class="form-label-group">
      <input type="password" id="inputPassword" name="password" class="form-control" placeholder="Password" required="required" />
      <label for="inputPassword">Password</label>
    </div>
  </div>
  <div class="form-group">
    <div id="warningbox">
    </div>
    <input type="submit" class="btn btn-primary btn-block" form="loginform" name="btnlogin" value="Login" />
  </div>
</form>
```

Figura 5: Form per Login (HTML).

Questi due campi dovranno essere inseriti obbligatoriamente e si potranno evitare controlli con JavaScript tramite il parametro "required". Sia Username che Password dovranno quindi essere valorizzati. Infine ci sarà un bottone per l'invio dei dati. Ora che la parte grafica è pronta, si potrà pensare a come gestire l'invio delle credenziali.

```

<script type="text/javascript">
    $('#loginform').submit(function() {
        $.ajax({
            type: "POST",
            url: 'process.php',
            data: {
                username: $("#inputUsername").val(),
                password: $("#inputPassword").val()
            },
            success: function(data)
            {
                if (data === 'correct') {
                    window.location.replace('index.php');
                }
                else {
                    $("#warningbox").html("<div class='alert alert-danger' role='alert'>" + data + "</div>");
                }
            }
        });
        return false;
    });
</script>

```

Figura 6: AJAX Per Login.

L'invio delle credenziali avverrà tramite **AJAX**, una tecnica di jQuery(libreria Javascript) che permette lo scambio di dati tra server e pagina WEB. Tutto ciò avverrà mantenendo la stessa pagina WEB, quindi non servirà ricaricare la pagina o passare ad un'altra. Verrà utilizzato il metodo **POST** per l'invio dei dati, permettendo che i dati vengano incapsulati nel pacchetto che viene inviato al server. Con il metodo GET invece, i dati inviati vengono passati tramite URL, rendendoli visibili e facilmente intercettabili. Il campo **DATA** conterrà i dati da inviare e nell'URL metteremo l'indirizzo della pagina PHP che gestirà il controllo delle credenziali. Se i dati saranno corretti, l'utente verrà indirizzato alla pagina Home, altrimenti verrà avvertito che Username o Password sono errati o inesistenti.

```

process.php > ...
<?php
include("../functions.php");

//se username e password sono valorizzati
if (isset($_POST['username']) && isset($_POST['password'])) {
    //ricavo con escape_string per evitare sql injection
    $username = $sqlconnection->real_escape_string($_POST['username']);
    $password = $sqlconnection->real_escape_string($_POST['password']);
    //controllo le credenziali
    $sql = "SELECT * FROM tbl_staff WHERE username = '$username' AND password = '$password'";

    if ($result = $sqlconnection->query($sql)) {
        if ($row = $result->fetch_array(MYSQLI_ASSOC)) {
            $uid = $row['staffID'];
            $username = $row['username'];
            $role = $row['role'];

            $_SESSION['uid'] = $uid;
            $_SESSION['username'] = $username;
            $_SESSION['user_level'] = "staff";
            $_SESSION['user_role'] = $role;

            echo "correct";
        } else {
            echo "Username o password errati";
        }
    }
}
}

```

Figura 7: Ricavo i dati e li controllo nel DB.

Nella pagina **Process.php** verranno ricevuti gli Input dal metodo Ajax, vengono processati e si fornisce "correct" se le credenziali sono corrette. Per controllare le credenziali si farà uso di una Query che controlla se nel Database ci sono risultati con Username e Password ricevute. In caso positivo verranno estratti anche altri dati che serviranno successivamente e verranno assegnati alle variabili di sessione. Le variabili di sessione servono ad immagazzinare dati e metterli a disposizione di più pagine. In questo caso ci servirà il ruolo della persona che ha effettuato il Login, per indirizzarlo alla pagina della cucina o della sala.

```
<?php
if ($_SESSION['user_role'] == "waiter") {
    echo '
        <li class="nav-item">
            <a class="nav-link" href="order.php">
                <i class="fas fa-fw fa-book"></i>
                <span>Ordini</span></a>
            </li>';
}

if ($_SESSION['user_role'] == "chef") {
    echo '
        <li class="nav-item">
            <a class="nav-link" href="kitchen.php">
                <i class="fas fa-fw fa-utensils"></i>
                <span>Cucina</span></a>
            </li>';
}
?>
```

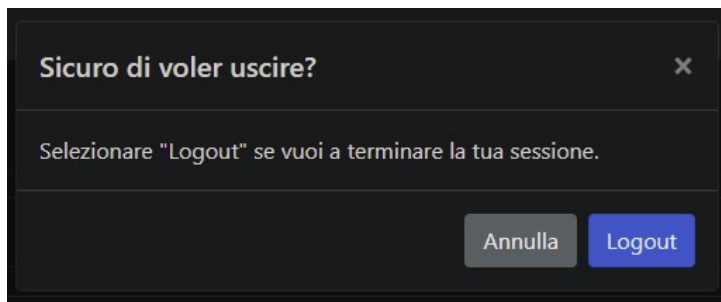
Nella pagina principale, infatti, dovremo differenziare tra cuochi e camerieri. Se il ruolo ricavato dalla sessione ruolo sarà "waiter" allora ci sarà un collegamento per la pagina delle ordinazioni. Se, invece, sarà "chef", allora il collegamento sarà relativo alla cucina. Questo sarà molto utile per evitare di creare una copia di tutto il progetto per ogni ruolo e rallentare inutilmente il sito.

Figura 8: Seleziona ruolo.

Ogni pagina che dovrà utilizzare le variabili di sessione dovrà richiamare la funzione **session_start()** per potersi sincronizzare su una sessione già esistente oppure crearne una nuova. Per evitare di scrivere la funzione, questa verrà inserita nel file delle funzioni che verrà richiamata in ogni pagina. Ad inizio pagina avremo quindi **include(functions.php)** che importerà le funzioni che ci serviranno in più pagine, attraverso la quali verrà avviata la sessione.

```
functions.php > ...
1  <?php
2  //prima controlla connessione al database
3  require("dbconnection.php");
4  session_start();
5
6  //contare righe query
7  > function getNumRowsQuery($query) { ...
13 }
14 //per stampare roba nella console
15 > function prova_log($message) { ...
19 }
20
```

Figura 9: Pagina in cui ci saranno funzioni utili.



Una volta che si avrà l'accesso alla pagina Home, verrà creata una sessione che rimarrà attiva fino a quando non si farà il Logout. Questa potrà essere importata nelle varie pagine PHP quando se ne avrà la necessità.

Figura 10: Conferma Logout.

Il Logout sarà disponibile nella pagina principale. Dopo aver cliccato sul bottone apposito, comparirà una finestra per avvertire l'utente che si sta effettivamente uscendo dal proprio account. Per evitare di rifare il Login cliccando erroneamente.

```
<div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModallabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModallabel">Sicuro di voler uscire?</h5>
        <button class="close" type="button" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">×</span>
        </button>
      </div>
      <div class="modal-body">Selezionare "Logout" se vuoi a terminare la tua sessione.</div>
      <div class="modal-footer">
        <button class="btn btn-secondary" type="button" data-dismiss="modal">Annulla</button>
        <a class="btn btn-primary" href="logout.php">Logout</a>
      </div>
    </div>
  </div>
</div>
```

Figura 11: Logout (HTML).

```
logout.php
<?php
include("../functions.php");
if((isset($_SESSION['uid']) && isset($_SESSION['username']) && isset($_SESSION['user_level'])) ) {
    if($_SESSION['user_level'] == "staff") {
        session_destroy();
        header("Location: login.php");
    }else
        header("Location: login.php");
}else
    header("Location: login.php");
```

Figura 12: Chiudo la sessione per Logout.

Per gestire il Logout, invece, dovremo utilizzare un'altra funzione, **session destroy()**, in cui la sessione e tutte le variabili di sessione verranno eliminate. Una volta fatto ciò, si verrà reindirizzati alla pagina del Login per inserire le credenziali ed eventualmente controllare le nuove credenziali inserite.

4.3 Interfaccia Personale

4.3.1 Sezione della sala

Dopo aver preparato il Database, mi sono occupato dell'interfaccia del personale. Come detto in precedenza, dovrà essere molto semplice. Verranno quindi create due pagine principali, la pagina Home per visualizzare gli ordini che sarà in costante aggiornamento e la pagina per effettuare gli ordini.

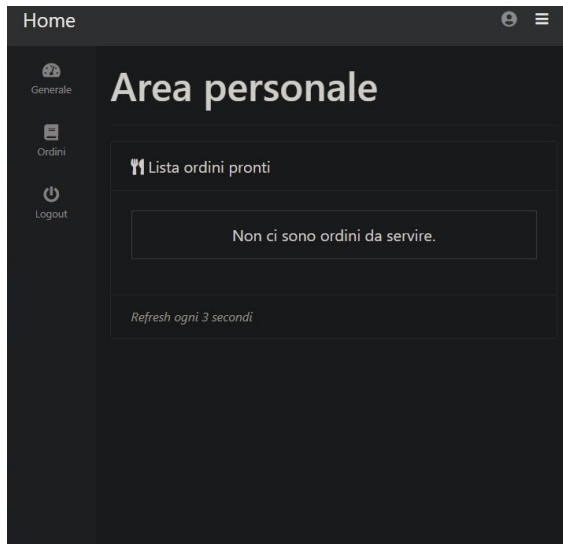


Figura 13: Interfaccia Home

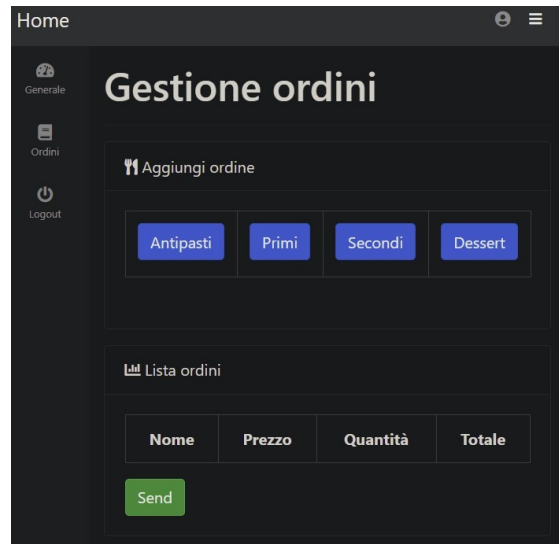


Figura 14: Interfaccia Ordini

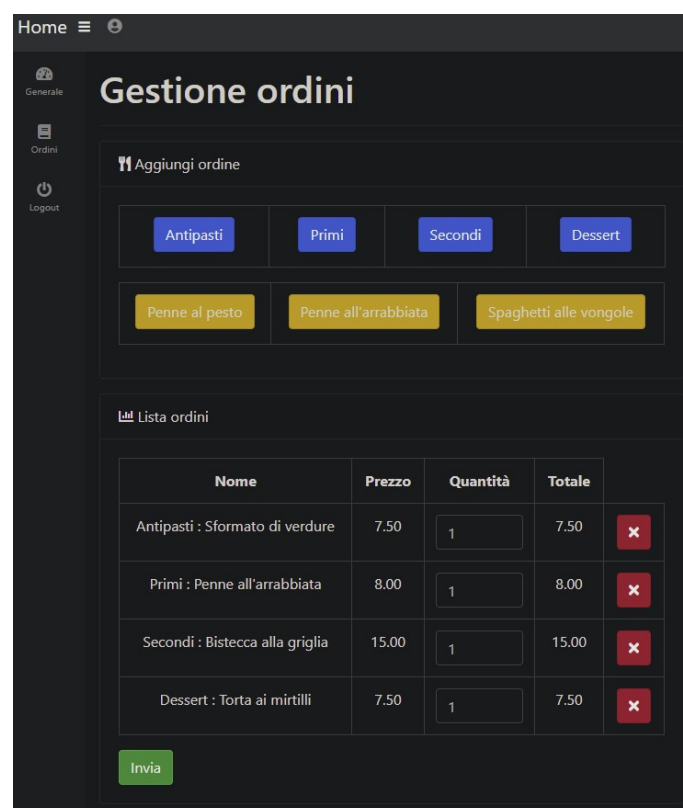


Figura 15: Ordine da inviare.

Dopo l'inserimento di qualche piatto, questi verranno aggiunti alla lista. Dopo aver confermato l'ordine, premendo "Invia", questo verrà inviato al Database tramite una Query.

```

function insertOrderDetailQuery($orderId, $itemId, $quantity) {
    global $sqlconnection;
    $addOrderQuery = "INSERT INTO tbl_orderdetail (orderId, itemId ,quantity, stato) VALUES ('{$orderId}', '{$itemId}', {$quantity}, 'waiting')";

    if ($sqlconnection->query($addOrderQuery) === TRUE) {
        echo "Inserito.";
    } else {
        echo "something wong";
        echo $sqlconnection->error;
    }
}

function insertOrderQuery($orderId, $totale) {
    global $sqlconnection;
    $addOrderQuery = "INSERT INTO tbl_order (orderId ,stato, total) VALUES ('{$orderId}', 'waiting', {$totale})";

    if ($sqlconnection->query($addOrderQuery) === TRUE) {
        echo "Inserito.";
    } else {
        echo "Errore";
        echo $sqlconnection->error;
    }
}

```

Figura 16: Funzioni per avere codice più pulito.

```

// se clicco bottone send e sono stati valorizzati piatto e quantità
if (isset($_POST['sentorder'])) {
    if (isset($_POST['itemId']) && isset($_POST['itemqty'])) {

        $arrItemID = $_POST['itemId'];
        $arrItemQty = $_POST['itemqty'];

        //controllo che num quantità = numero itemid per sicurezza
        if (count($arrItemID) == count($arrItemQty)) {
            //ricavo numero di ordini
            $arrlength = count($arrItemID);

            //incremento l'ID ogni volta che devo aggiungere un nuovo ordine
            $currentOrderID = getLastID("orderId", "tbl_order") + 1;
            echo "<p> . $currentOrderID . "</p>";

            //per ogni piatto, richiamo la funzione per inviarli al Database
            for ($i = 0; $i < $arrlength; $i++) {
                insertOrderDetailQuery($currentOrderID, $arrItemID[$i], $arrItemQty[$i]);
            }

            //calcolo il totale di tutto l'ordine
            $querytotale = "SELECT SUM(tbl_orderdetail.quantity * tbl_menuitem.price) AS Totale
                            FROM tbl_orderdetail LEFT JOIN tbl_menuitem USING(itemId);";

            $totale = $sqlconnection->query($querytotale);
            //ricavo la prima riga della query, quindi il totale dell'ordine
            $row = $totale->fetch_row();
            $totale = $row[0] . $row[1];

            // richiamo la funzione dell'ordine dando come parametro il totale ricavato e ID
            $ris = insertOrderQuery($currentOrderID, $totale);
            header("Location: index.php");
            exit();
        } else {
            echo "Errore";
        }
    }
}

```

Figura 17: Richiamo le funzioni per inviare ordini al DB.

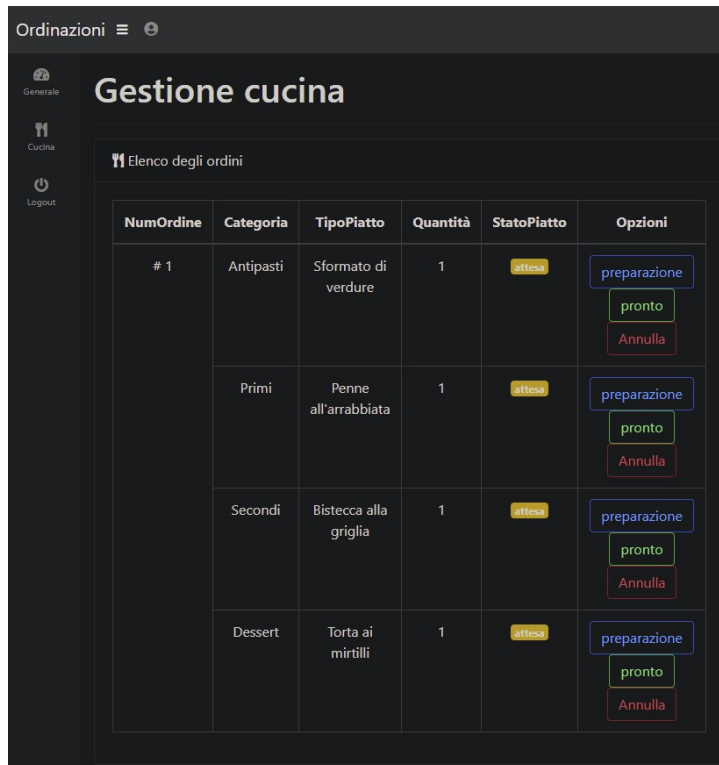
		orderID	orderDetailID	itemId	quantity	stato
	Modifica Copia Elimina	1	1	3	1	attesa
	Modifica Copia Elimina	1	2	12	1	attesa
	Modifica Copia Elimina	1	3	23	1	attesa
	Modifica Copia Elimina	1	4	33	1	attesa

Figura 18: Piatti ordinati nel Database

		orderID	stato	total
	Modifica Copia Elimina	1	attesa	38.00

Figura 19: Ordine nel Database

4.3.2 Sezione della cucina



The screenshot shows a web application interface for kitchen management. It features a sidebar with navigation links for 'Generale', 'Cucina', and 'Logout'. The main area is titled 'Gestione cucina' and contains a section 'Elenco degli ordini'. Below this is a table with columns: NumOrdine, Categoria, TipoPiatto, Quantità, StatoPiatto, and Opzioni. The table lists four items, all with a quantity of 1 and a status of 'attesa' (pending). Each item has three buttons in the 'Opzioni' column: 'preparazione' (blue), 'pronto' (green), and 'Annulla' (red).

NumOrdine	Categoria	TipoPiatto	Quantità	StatoPiatto	Opzioni
# 1	Antipasti	Sformato di verdure	1	attesa	preparazione pronto Annulla
	Primi	Penne all'arrabbiata	1	attesa	preparazione pronto Annulla
	Secondi	Bistecca alla griglia	1	attesa	preparazione pronto Annulla
	Dessert	Torta ai mirtilli	1	attesa	preparazione pronto Annulla

Figura 20: Ordini ricevuti dalla cucina.

In base al bottone cliccato verrà modificato lo stato del piatto nel Database. Questo sarà possibile tramite una funzione che invierà una richiesta POST al file **editStato.php**, che si occuperà di gestire la modifica degli stati. Come parametro della funzione avremo un oggetto **Button** da cui ricaveremo il valore per capire il bottone cliccato dall'utente e l'ID del piatto. Come dati da inviare, invece, verranno selezionati l'ID del piatto e lo stato in cui il piatto dovrà essere modificato, ricavato dalla funzione.

```
//per cambiare lo stato del piatto
function editstato(objBtn, orderdetailID) {
    var stato = objBtn.value;

    $.ajax({
        url: "editstato.php",
        type: 'POST',
        data: {
            orderdetailID: orderdetailID,
            stato: stato
        },
        success: function(output) {
            refreshTableOrder();
        }
    });
}
```

Figura 21: Funzione per cambiare stato del piatto.

```

$displayOrderQuery = "
SELECT tbl_order.orderID,tbl_orderdetail.orderdetailID, tbl_menu.menuName,
tbl_orderdetail.itemID, tbl_menuitem.menuItemName, tbl_orderdetail.quantity, tbl_orderdetail.stato
FROM tbl_order
LEFT JOIN tbl_orderdetail ON tbl_order.orderID = tbl_orderdetail.orderID
LEFT JOIN tbl_menuitem ON tbl_orderdetail.itemID = tbl_menuitem.itemID
LEFT JOIN tbl_menu ON tbl_menuitem.menuID = tbl_menu.menuID
WHERE tbl_orderdetail.stato IN ( 'attesa','preparazione','pronto')";

$orderRow = $orderResult->fetch_array(MYSQLI_ASSOC);

```

Figura 22: Query per prendere tutti i piatti non completati.

Questa Query prende tutti i piatti nel Database che non sono ancora stati serviti e poi viene eseguito una funzione di MySQLi (un'estensione per gestire le operazioni con il Database). I dati vengono estratti sotto forma di **Array associativi**, in cui ad ogni elemento corrisponde una chiave e dei valori associati a quella chiave. Le chiavi saranno i nome dei campi della tabella, per esempio nomePiatto, IDPiatto e Stato mentre i valori associati come ad esempio PastaAlSugo, 1, Pronto.

Questa funzione ci permetterà di prendere il risultato della Query, un Array di ordini, e dividerla in record. Sarà così più gestibile e si potranno estrarre man mano tutti i campi necessari.

```

$orderRow = $orderResult->fetch_array(MYSQLI_ASSOC);
if ($orderResult = $sqlconnection->query($displayOrderQuery)) {
    $currentspan = 0;

    //se non sono presenti ordini
    if ($orderResult->num_rows == 0) {
        echo "<tr><td class='text-center' colspan='7' >Niente ordini per il momento</td></tr>";
    } else {
        //per ogni ordine "grosso"
        while ($orderRow = $orderResult->fetch_array(MYSQLI_ASSOC)) {

            $rowspan = getCountID($orderRow["orderID"], "orderID", "tbl_orderdetail");

            if ($currentspan == 0)
                $currentspan = $rowspan;

            echo "<tr>";
            //stampo numero ordine
            if ($currentspan == $rowspan) {
                echo "<td rowspan=" . $rowspan . "># " . $orderRow['orderID'] . "</td>";
            }
            //per stampare i singoli piatti
            echo
            "<td>" . $orderRow['menuName'] . "</td>" .
            "<td>" . $orderRow['menuItemName'] . "</td>" .
            "<td class='text-center'>" . $orderRow['quantity'] . "</td>";

```

Figura 23: Ciclo finchè ci sono elementi nell'array.

Qui viene estratto il campo "orderID" dal record, viene stampato con accanto ci saranno anche il tipo, nome e quantità del piatto. Questi verranno visualizzati nella sezione delle ordinazioni. Ora bisognerà aggiungere i bottoni per modificare lo stato.


```

//in base al bottone cliccato, cambio lo stato.
switch ($orderRow['stato']) {
    //se è in attesa allora posso mettere i bottoni preparazione o pronto
    case 'attesa':
        echo "<button onclick='editstato(this," . $orderRow['orderdetailID'] . ")' class='btn btn-outline-primary' value = 'preparazione'>preparazione</button>";
        echo "<button onclick='editstato(this," . $orderRow['orderdetailID'] . ")' class='btn btn-outline-success' value = 'pronto'>pronto</button>";
        break;
    case 'preparazione':
        //dalla preparazione posso solo mettere pronto
        echo "<button onclick='editstato(this," . $orderRow['orderdetailID'] . ")' class='btn btn-outline-success' value = 'pronto'>pronto</button>";
        break;
    case 'pronto':
        //se è pronto posso solo eliminare il piatto dalla lista ordini
        echo "<button onclick='editstato(this," . $orderRow['orderdetailID'] . ")' class='btn btn-outline-warning' value = 'finito'>Clear</button>";
        break;
}

```

Figura 24: Dispongo bottoni in base allo stato piatto (OrderRow).

Verrà utilizzato uno **Switch Case** che permetterà di inserire i bottoni in base alla situazione del piatto. Verrà utilizzato **OrderRow**, che indicherà il piatto corrente. Verrà estratto il campo **stato** dal record. In base al risultato fornito (attesa, preparazione o pronto), verranno collocati i bottoni rimanenti.

Se dovessero arrivare altri ordini, verranno collocati in successione all'ultimo.

Gestione cucina

🍴 Elenco degli ordini

NumOrdine	Categoria	TipoPiatto	Quantità	StatoPiatto	Opzioni
# 1	Antipasti	Sformato di verdure	1	pronto	Clear Annulla
	Primi	Penne all'arrabbiata	1	pronto	Clear Annulla
	Secondi	Bistecca alla griglia	1	preparazione	pronto Annulla
	Dessert	Torta ai mirtilli	1	preparazione	pronto Annulla
# 2	Antipasti	Insalata di mare	1	attesa	preparazione pronto Annulla
	Primi	Spaghetti alle vongole	1	attesa	preparazione pronto Annulla

Figura 25: Ordini ricevuti dalla cucina.

			orderID	orderDetailID	itemID	quantity	stato
☐	✎ Modifica	📋 Copia	🗑 Elimina	1	1	3	1 pronto
☐	✎ Modifica	📋 Copia	🗑 Elimina	1	2	12	1 pronto
☐	✎ Modifica	📋 Copia	🗑 Elimina	1	3	23	1 preparazione
☐	✎ Modifica	📋 Copia	🗑 Elimina	1	4	33	1 preparazione
☐	✎ Modifica	📋 Copia	🗑 Elimina	2	5	2	1 attesa
☐	✎ Modifica	📋 Copia	🗑 Elimina	2	6	13	1 attesa

Figura 26: Modifiche apportate al Database.

Fino ad ora ci sono state solo modifiche relative agli stati dei piatti sul Database. Ora bisognerà visualizzare gli ordini pronti nella sala. Per fare ciò passeremo con il metodo GET, l'ID dell'ordine e dei degli ordini dei piatti. In questo caso, visto che non stiamo inviando dati sensibili, possiamo utilizzare il metodo GET senza preoccupazioni.

Gli ordini pronti verranno controllati ogni volta che ci sarà il refresh, così da poter monitorare tutti gli ordini pronti ad essere serviti.

Verrà caricato il contenuto della risposta della chiamata GET nella tabella creata nell'HTML, che fino ad ora era vuota.

```
<div class="row">
  <div class="col-lg-9">
    <div class="card mb-3">
      <div class="card-header">
        <i class="fas fa-utensils"></i>
        Lista ordini pronti
      </div>
      <div class="card-body">
        <table id="orderTable" class="table table-striped table-bordered width=" 100%" cellspacing="0">
        </table>
      </div>
      <div class="card-footer small text-muted"><i>Refresh ogni 3 secondi</i></div>
    </div>
  </div>
</div>
```

Figura 27: Tabella ordini HTML.

```
<script type="text/javascript">
  //refresho e invio "pronto" alla pagina displayorder
  function refreshTableOrder() {
    $("#orderTable").load("displayorder.php?cmd=pronto");
  }
  //se la pagina è pronta per script js allora refresh pagina x vedere se sono arrivati ordini
  $(document).ready(function() {
    refreshTableOrder();
  });
  // refresh ogni 5 secondi
  setInterval(function() {
    refreshTableOrder();
  }, 5000);
</script>
```

Figura 28: Funzione per caricare i dati con Refresh.

Quando la pagina è pronta, ogni 5 secondi viene fatto un refresh che controlla eventuali cambiamenti di stato nel Database. Nel caso in cui ci siano dei piatti con stato **pronto** saranno visualizzati nella tabella. I dati verranno passati tramite metodo GET, come detto in precedenza.


```
//visualizza gli ordini nella home dello staff se sono pronti
if ($_GET['cmd'] == 'pronto') {

    //visualizzo gli ordini pronti
    $latestReadyQuery = "SELECT tbl_orderdetail.orderID, tbl_orderdetail.orderdetailID, tbl_menuitem.menuItemName
                        FROM tbl_orderdetail LEFT JOIN tbl_menuitem USING(itemID)
                        WHERE tbl_orderdetail.stato IN ('pronto', 'finito')";

    if ($result = $sqlconnection->query($latestReadyQuery)) {
        if ($result->num_rows == 0) {
            echo "<tr><td class='text-center'>Non ci sono ordini da servire.</td></tr>";
        }
        //mando orderID e orderdetailID tramite GET alla pagina editstato per cambiare stati
        while ($latestOrder = $result->fetch_array(MYSQLI_ASSOC)) {
            echo "<tr><td><i class='fas fa-bullhorn' style='color:green;'>
            </i><b> Order #" . $latestOrder['orderID'] . ": " . $latestOrder['menuItemName'] . "</b> Pronto ad essere servito.
            <a href='editstato.php?orderdetailID=" . $latestOrder['orderdetailID'] . "&orderID=" . $latestOrder['orderID'] . "'>
            <i class='fas fa-check float-right'></i></a></td></tr>";
        }
    }
}
```

Figura 29: Visualizzazione ordini pronti ad essere serviti.

Ora si prenderanno gli ordini dei piatti che hanno stato **pronto** tramite una Query. Poi tramite un ciclo WHILE, per ogni ordine ricavato dalla Query, verrà inserito l'ordine del piatto con OrderID, NomePiatto. Inoltre bisognerà aggiungere un bottone per far sì che la sala possa confermare di aver ricevuto e servito l'ordine. Così da eliminare dalla coda gli ordini completati.



Figura 30: Ricevuta dell'ordine alla sala.

Man mano che i piatti saranno pronti, verranno visualizzati nella sala così potranno essere serviti. Cliccando il bottone verrà inviata una richiesta GET alla pagina **EditStato.php** e come parametri verranno dati l>ID del piatto e dell'ordine.

I piatti selezionati verranno contrassegnati come completati nel Database.

			orderID	orderDetailID	itemID	quantity	stato
<input type="checkbox"/>	Modifica	Copia Elimina	1	1	3	1	pronto
<input type="checkbox"/>	Modifica	Copia Elimina	1	2	12	1	pronto

Figura 31: Modifiche apportate al Database.

Inoltre quando tutti gli ordini dei singoli piatti saranno completati, anche l'ordine complessivo verrà contrassegnato come completato. Per fare ciò, ci saranno dei controlli in cui se tutti gli ordini con un certo ID sono nello stato "completato" allora anche l'ordine complessivo sarà completato.

```
//prendo orderdetailID del piatto così lo contrassegno come completato (bottone ok)
if (isset($_GET['orderdetailID']) && isset($_GET['orderID'])) {
    $stato = "Completed";
    //ricavo i dati
    $orderdetailID = $sqlconnection->real_escape_string($_GET['orderdetailID']);
    $orderID = $sqlconnection->real_escape_string($_GET['orderID']);
    //dichiaro query per cambiare stato dell'ordine
    $addorderQuery = "UPDATE tbl_orderdetail SET stato = '{$stato}'
    | | | | | WHERE orderdetailID = '{$orderdetailID}'";
    ";

    //se tutti gli ordini sono finiti allora termino anche l'ordine main
    //ritorna true se c'è almeno un prodotto in lavorazione
    $checkOrderQuery = "SELECT tbl_orderdetail.orderID, tbl_orderdetail.orderdetailID, tbl_orderdetail.stato
    | | | | | FROM tbl_orderdetail INNER JOIN tbl_order USING (orderID)
    | | | | | WHERE tbl_orderdetail.stato = 'Completed'
    | | | | | AND tbl_orderdetail.orderID = $orderID";
    ";

    //conto righe totali per l'ordine corrente
    $risnormal = $sqlconnection->query("SELECT * FROM tbl_orderdetail INNER JOIN tbl_order USING (orderid)
    | | | | | WHERE tbl_orderdetail.orderID = $orderID");
    $rows = $risnormal->num_rows;
}
```

Figura 32: EditStato Parte 1.

Ora si potranno ricavare i valori degli ID e ci serviranno una serie di Query. Una che conta il numero di piatti ordinati con stato completato e numero ordine prefissato; una per contare solo il numero di piatti con un numero ordine prefissato. Se questi due forniranno come risultato lo stesso numero di piatti significa che il numero di ordini è uguale al numero di ordini completati e quindi tutti gli ordini dei piatti sono stati terminati.

```
if ($sqlconnection->query($addorderQuery)) {
    //ritorna il numero di piatti completati, idorder e stato con stesso orderID
    $rischeck = $sqlconnection->query(
    "SELECT COUNT(tbl_orderdetail.orderdetailID) AS Conta, tbl_orderdetail.orderID, tbl_orderdetail.stato
    FROM tbl_orderdetail
    WHERE tbl_orderdetail.stato = 'Completed'
    AND tbl_orderdetail.orderID = $orderID"
    );

    $orderRow = $rischeck->fetch_array(MYSQLI_ASSOC);
    //ricavo solo il numero di piatti
    $row_cnt = $orderRow["Conta"];
    //quando il totale di piatti ordinati è uguale al totale di piatti completati
    //posso segnare l'ordine totale a completato.
    if ($rows == $row_cnt) {
        $finishOrderQuery = "UPDATE tbl_order SET stato = 'Completed'
        | | | | | WHERE orderID = $orderID";
        ";
        $sqlconnection->query($finishOrderQuery);
    }
    header("Location: index.php");
} else {
    echo "Errore";
    echo $sqlconnection->error;
}
```

Figura 33: EditStato Parte 2.

Con l'operatore IF verrà controllato che almeno un prodotto venga aggiornato allo stato pronto. In quel caso verrà fatto il controllo appena citato per verificare che tutti i piatti siano completati. Ora viene creata la Query per modificare lo stato dell'ordine complessivo e dopo la modifica si verrà rediretti alla schermata Home.

🍴 Elenco degli ordini					
NumOrdine	Categoria	TipoPiatto	Quantità	StatoPiatto	Opzioni
# 1	Antipasti	Sformato di verdure	1	pronto	Clear Annulla
	Primi	Penne all'arrabbiata	1	pronto	Clear Annulla
	Secondi	Bistecca alla griglia	1	pronto	Clear Annulla
	Dessert	Torta ai mirtilli	1	pronto	Clear Annulla
# 2	Antipasti	Insalata di mare	1	preparazione	pronto Annulla
	Primi	Spaghetti alle vongole	1	preparazione	pronto Annulla

Figura 34: Tutti gli ordini pronti.

Area personale

🍴 Lista ordini pronti

🔔 Order #1: Sformato di verdure Pronto ad essere servito. ✓

🔔 Order #1: Penne all'arrabbiata Pronto ad essere servito. ✓

🔔 Order #1: Bistecca alla griglia Pronto ad essere servito. ✓

🔔 Order #1: Torta ai mirtilli Pronto ad essere servito. ✓

Refresh ogni 3 secondi

Figura 35: Gli ordini visualizzati nella schermata Home.

Ora tutti i piatti dell'ordine 1 sono disponibili ad essere serviti. Man mano che verranno serviti e contrassegnati come serviti, non si vedranno più nella sezione ordini visto che ormai non ci sarebbe alcuna utilità a mostrarli.

<div><div><div><div></div><div></div><div></div></div><div></div></div></div>					orderID	orderDetailID	itemID	quantity	stato
<div><div><div></div><div></div><div></div></div><div></div></div>	<div><div></div><div></div><div></div></div> Modifica	<div><div></div><div></div><div></div></div> Copia	<div><div></div><div></div><div></div></div> Elimina	1	1	3	1	Completed	
<div><div><div></div><div></div><div></div></div><div></div></div>	<div><div></div><div></div><div></div></div> Modifica	<div><div></div><div></div><div></div></div> Copia	<div><div></div><div></div><div></div></div> Elimina	1	2	12	1	Completed	
<div><div><div></div><div></div><div></div></div><div></div></div>	<div><div></div><div></div><div></div></div> Modifica	<div><div></div><div></div><div></div></div> Copia	<div><div></div><div></div><div></div></div> Elimina	1	3	23	1	Completed	
<div><div><div></div><div></div><div></div></div><div></div></div>	<div><div></div><div></div><div></div></div> Modifica	<div><div></div><div></div><div></div></div> Copia	<div><div></div><div></div><div></div></div> Elimina	1	4	33	1	Completed	
<div><div><div></div><div></div><div></div></div><div></div></div>	<div><div></div><div></div><div></div></div> Modifica	<div><div></div><div></div><div></div></div> Copia	<div><div></div><div></div><div></div></div> Elimina	2	5	2	1	preparazione	
<div><div><div></div><div></div><div></div></div><div></div></div>	<div><div></div><div></div><div></div></div> Modifica	<div><div></div><div></div><div></div></div> Copia	<div><div></div><div></div><div></div></div> Elimina	2	6	13	1	preparazione	

Figura 36: Tutti i piatti sono contrassegnati come completati.

Dopo averli completati, anche nel Database verranno segnati completati. Nel momento in cui tutti saranno completati anche l'ordine complessivo sarà completato.

<div><div><div></div><div></div><div></div></div></div>				orderID	stato	total	
<div><div><div></div></div></div>	<div><div><div></div></div></div>	Modifica	<div><div><div></div></div></div> Copia	<div><div><div></div></div></div> Elimina	1	Completed	38.00
<div><div><div></div></div></div>	<div><div><div></div></div></div>	Modifica	<div><div><div></div></div></div> Copia	<div><div><div></div></div></div> Elimina	2	preparazione	61.50

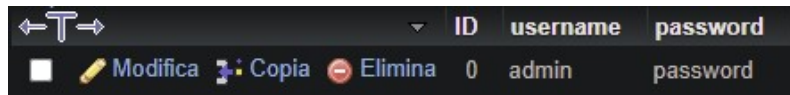
Figura 37: L'ordine complessivo 1 è completo.

5 Interfaccia Admin

L'interfaccia Admin verrà implementata per gestire il personale, il menu e tutte le componenti importanti del sito del ristorante. I compiti principali dell'amministratore saranno di poter aggiungere, modificare ed eliminare il menù (compresi i piatti e i tipi di piatti) e il personale. Inoltre deve poter visualizzare gli ordini in corso ed il personale disponibile.

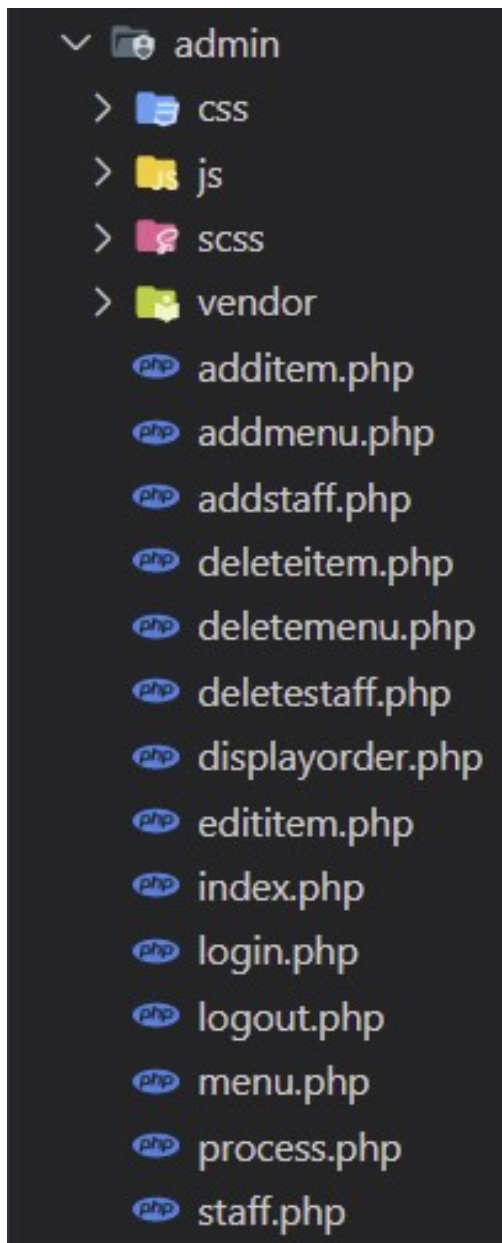
5.1 Database

Il Database sarà lo stesso con la sola differenza dell'utilizzo della tabella Admin che conterrà le credenziali per gli amministratori.



ID	username	password
0	admin	password

5.2 Struttura progetto



Admin: Gestione interfaccia Admin.

CSS: Gestione parte CSS.

JS Gestione parte JavaScript.

SCSS: Gestione parte SCSS.

VENDOR: Bootstrap.

AddItem: Aggiunta piatti.

AddMenu: Aggiunta tipo piatto .

AddStaff: Aggiunta personale.

DeleteItem: Elimina piatto.

DeleteMenu: Elimina tipo piatto.

DeleteStaff: Elimina personale.

DisplayOrder: Visualizza ordini.

EditItem: Modifica piatto.

Index: Pagina Home.

Login: Pagina di Login.

Logout: Gestione Logout.

Menu: Sezione Menu.

Process: Gestione Login.

Staff: Sezione personale.

5.3 Login-Logout

La sezione di Login è quasi identica a quella dell'interfaccia del personale, l'unico fattore che differisce da essa è che le credenziali vengono controllate nella **tabella Admin**, invece della tabella del Personale. Inoltre la tabella Admin non sarà collegata alla tabella ruolo, dato che non se ne ha la necessità.

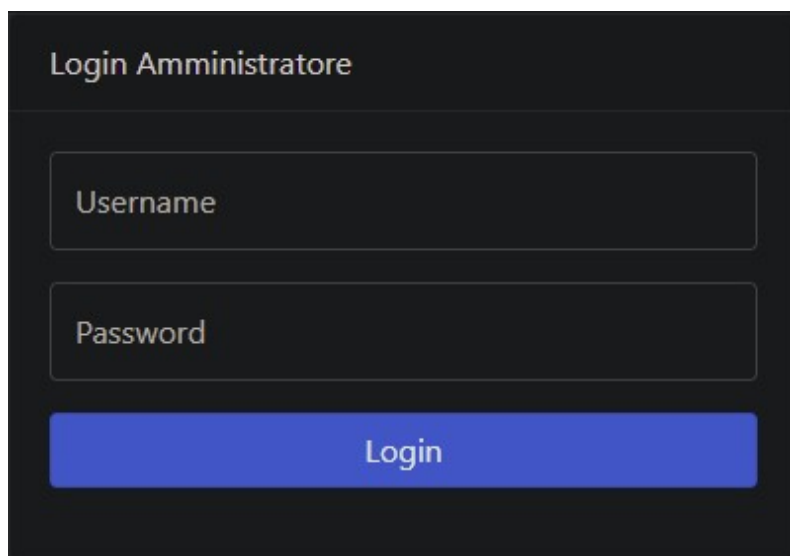
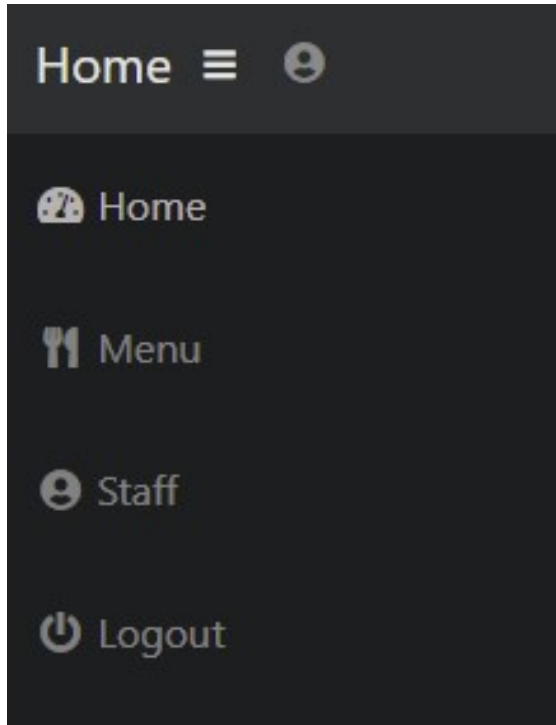


Figura 38: Pagina Login.

5.4 Struttura piattaforma



L'interfaccia Admin: è divisa in sezioni:

- **Sezione Home:**
Visualizzazione complessiva della situazione.
- **Sezione Menu:**
Gestione aggiunta, modifica ed eliminazione elementi menù.
- **Sezione Personale:**
Gestione aggiunta, modifica ed eliminazione personale.
- **Sezione Logout:**
Logout e fine sessione.

5.5 Interfaccia Home

Qua si avrà il resoconto dei piatti in lavorazione e la lista del personale con il ruolo che copre. Per visualizzarli basterà eseguire Query per visualizzare i piatti e una per il personale come in precedenza.

```
<table table class="table table-bordered text-center" width="100%" cellpadding="0">
  <tr>
    <td><b>Nome</b></td>
    <td><b>Ruolo</b></td>
  </tr>
</table>
<?php
  $displayStaffQuery = "SELECT username, role FROM tbl_staff";
  if ($result = $sqlconnection->query($displayStaffQuery)) {
    while ($staff = $result->fetch_array(MYSQLI_ASSOC)) {
      echo "<tr>";
      echo "<td>{$staff['username']}</td>";
      echo "<td>{$staff['role']}</td>";
      echo "</tr>";
    }
  }
?>
</table>
```

Figura 39: Visualizza personale.

Pannello Amministratore

🍽️

Lista ordini pronti

Ordine #	Tipo	Nome Piatto	Quantità	Stato
# 1	Antipasti	Sformato di verdure	1	pronto
	Primi	Penne all'arrabbiata	1	pronto
	Secondi	Bistecca alla griglia	1	pronto
	Dessert	Torta ai mirtilli	1	pronto

Refresh ogni 5 secondi

👤

Lista Personale

Nome	Ruolo
Cameriere	waiter
Cuoco	chef

Figura 40: Sezione Home.

5.6 Interfaccia Menu

Per gestire il Menu verrà utilizzata una tabella che visualizzerà i tipi di piatti ed i loro corrispondenti piatti. Ci saranno due bottoni principali, uno per aggiungere ed uno per rimuovere.

```
additem.php > ...
<?php

//Nuovo piatto
if (isset($_POST['addItem'])) {
    //se sono valorizzati i campi
    if (!empty($_POST['itemName']) && !empty($_POST['itemPrice']) && !empty($_POST['menuID'])) {
        //Ricavo i dati da inserire
        $itemName = $sqlconnection->real_escape_string($_POST['itemName']);
        $itemPrice = $sqlconnection->real_escape_string($_POST['itemPrice']);
        $menuID = $sqlconnection->real_escape_string($_POST['menuID']);
        //Query per aggiungere
        $addItemQuery = "INSERT INTO tbl_menuitem (menuID ,menuItemName ,price)
        VALUES ({$menuID} , '{$itemName}' , {$itemPrice})";
        //se la query va a buon fine, redirectione alla home
        if ($sqlconnection->query($addItemQuery) === TRUE) {
            header("Location: menu.php");
            exit();
        } else {
            echo "Errore";
            echo $sqlconnection->error;
        }
    } else {
        echo "Errore";
    }
}
```

Ad esempio per aggiungere un nuovo piatto verrà richiamata la pagina **AddItem.php** in cui si prenderanno i parametri attraverso metodo POST e successivamente avverrà la creazione della Query. In questo caso sarà una Query per l'aggiunta di un nuovo record al Database. Se non ci saranno problemi si verrà rediretti alla pagina dei Menu, così da poter vedere le modifiche apportate.

Gestione dei Menu

Operazioni possibili: Aggiunta, Modifica, Eliminazione

 Lista Menu

Nuovo tipo piatto

 Antipasti

Aggiungi Elimina

#	Nome Piatto	Prezzo	Opzioni
1	Vitello tonnato	10.00	Modifica Elimina
2	Insalata di mare	11.00	Modifica Elimina
3	Sformato di verdure	7.50	Modifica Elimina

Come visto in precedenza, per queste operazioni basterà effettuare una richiesta di modifica al Database con una Query. In base all'operazione necessaria, creare la Query di conseguenza e richiamare la Query attraverso una pagina PHP che verrà anch'essa richiamata dal bottone che verrà inserito nella pagina HTML.

#	Nome	Ruolo	Option
1	Cameriere	Waiter	Elimina
2	Cuoco	Chef	Elimina

Password di default per nuovo personale : Abc123

Aggiungi Personale

Waiter +

Figura 41: Sezione Menu.

Per **modificare** il ruolo di un lavoratore, basterà selezionare il ruolo e si visualizzerà la lista di ruoli dalla quale potrà scegliere il ruolo desiderato.

Per **eliminare** invece basterà cliccare sul bottone Elimina.

Infine per **aggiungere** si potrà cliccare sul ruolo per selezionare come visto precedentemente e inserire lo Username. La password di default per gli utenti sarà Abc123 e potrà essere modificata successivamente.

6 Conclusione

Ci sono state varie scelte e ipotesi formulate, cercando comunque di rimanere il più generico possibile. Questo perché non si conoscono precisamente le intenzioni del cliente e di conseguenza è sempre meglio rimanere sul generico.

In questo modo il codice è facilmente modificabile e in base alle esigenze del cliente si potranno implementare nuove funzionalità.

Il progetto è stato realizzato con certi strumenti e linguaggi di programmazione. Può essere migliorato con una conoscenza più ampia degli strumenti utilizzati e quindi anche con un'esperienza maggiore. Dato che non è un sito da esporre all'esterno, ho cercato di evitare decorazioni o implementare sistemi pesanti così da avere un sito più performante e reattivo.

Il tempo a disposizione per fare questo progetto era abbastanza limitato. Per poterlo mettere sul mercato servirebbero alcune modifiche che però dipenderebbero dalla richiesta del cliente.