
ITTS Plesso “Enrico Fermi” - PISTOIA

a.s. 2020/2021

Relazione di informatica:

Esercitazione CRUD

Proff.

Ing. Pasquale Silvestro

David Caramelli

Alunni: Giacomo Orsucci & Sami Boutabaa

Firma:

Giacomo
Orsucci

Scritto con i tasti

Sami
BOUTABA

Pistoia (PT), consegna per il 10/04/2021

Sommario

Credenziali per accedere ad admin.php	2
Obiettivo	2
Strumenti e linguaggi utilizzati	3
Strumenti	3
Linguaggi e formati utilizzati	3
Cenni teorici	3
I linguaggi interpretati	3
HTML	4
CSS	4
Bootstrap	5
JavaScript	5
JQuery	6
Il formato JSON	6
PHP	7
Le sessioni in PHP	7
SQL	9
Analisi funzionale	10
Flusso del programma	10
Idea iniziale	11
Idea iniziale dell'aspetto grafico del sito	11
Analisi tecnica	14
Struttura del CRUD	15
Schema concettuale del database	16
Schema logico del database	17
Creazione delle tabelle e tipi scelti	17
JQuery (AJAX) e JSON	18
Upload delle immagini	19
Gestione dei login	20
Gestione degli url	23
Soluzione dei problemi con la validazione dei dati	23
Manuale utente	24
Manuale per il cliente	24
Manuale per l'amministratore	27
login	27
Amministrazione	28
Aggiunta di un nuovo ingrediente	29

Aggiunta di un nuovo prodotto	29
Modifica dei prodotti presenti	30
Gestione degli ordini	31
Criticità riscontrabili e cosa fare	32
Test data set/debug	32
Conclusioni ed osservazioni personali	32
Sitografia e bibliografia	33

Credenziali per accedere ad admin.php

- 1) username: Orsu password: Orsu123
- 2) username: Bota password: Bota123

I username non sono case sensitive, la password sì.

Obiettivo

Sviluppare un'applicazione web PHP che permetta di effettuare online gli ordini alla pizzeria “Pizza express”. Come richiesto dalla traccia ogni ordine deve presentare:

- Numero ordine (univoco)
- Data dell'ordine
- Nominativo

Ogni Indirizzo deve essere composto da:

- Via, civico, città

E deve essere presente una lista di ordinazioni composta da:

- Descrizione del prodotto
- Prezzo
- Quantità

Inoltre la traccia richiede che siano previste le funzionalità di amministrazione di inserimento delle pizze e degli ingredienti che contengono, che devono poi poter essere ordinate dall'apposita pagina cliente. Si richiede infine che sia presente un filtro sugli ingredienti delle pizze (lato cliente) e che ad ogni pizza sia associata un'immagine relativa (salvata tramite link nel db).

Strumenti e linguaggi utilizzati

Strumenti

- Visual studio code
- Visual studio live share per la collaborazione in tempo reale sul codice
- WampServer (per MySQL e la virtualizzazione del server)

Linguaggi e formati utilizzati

- HTML
- CSS
- PHP
- JavaScript (tra cui la libreria JQuery ed AJAX)
- JSON
- SQL (MySQL)

Cenni teorici

I linguaggi interpretati

I linguaggi **interpretati** sono linguaggi che, per l'appunto, sono interpretati ed eseguiti **“real-time”** su qualsiasi macchina che presenti il relativo interprete. L'interprete, a differenza di un compilatore per linguaggi compilati, non traduce il codice sorgente in linguaggio macchina, ma lo esegue **“passo-passo”**, riga per riga. I linguaggi interpretati hanno il vantaggio di essere eseguibili su qualsiasi macchina che presenti l'apposito interprete **senza che debbano essere ricompilati** e che quindi venga rigenerato il relativo file sorgente ogni volta (come invece accade per i linguaggi compilati, che necessitano anche dell'apposito compilatore per ogni diversa tipologia di macchina). Questo aspetto però, comporta anche una **minore velocità** di esecuzione determinata dalla necessità dell'interprete di **analizzare** le istruzioni riga per riga a partire dal livello sintattico, di **identificare** le azioni da eseguire (eventualmente trasformando i nomi simbolici delle variabili coinvolte nei corrispondenti indirizzi di memoria), ed **eseguirle**. Al contrario, le istruzioni del codice compilato, già in linguaggio macchina, vengono caricate e instantaneamente eseguite dal processore.

In compenso però, l'interpretazione di un programma **può essere più rapida** del ciclo compilazione/esecuzione. Questa differenza può costituire un **vantaggio** durante lo **sviluppo**,

specialmente se questo viene condotto con tecniche di fast prototyping, o durante il debugging. Inoltre, la maggior parte degli interpreti consentono all'utente di agire sul programma in esecuzione **sospendendolo, ispezionando o modificando** i contenuti delle sue **variabili**, in modo più flessibile e potente di quanto generalmente si può ottenere, per il codice compilato, da un debugger. Tra i linguaggi interpretati che abbiamo utilizzato per la realizzazione delle seguenti esercitazioni troviamo HTML, CSS e JavaScript lato client e PHP lato server.

HTML



gestibili in JavaScript.

HTML, **HyperText Markup Language**, è un linguaggio di **markup** lato client. Non è un linguaggio di programmazione, ma di markup ed ha lo scopo di definire la **struttura**, “lo scheletro” delle pagine web definendone la composizione e gli elementi. La sintassi del HTML è definita dal **World Wide Web Consortium** (W3C). Attualmente l’ultima versione di HTML è la 5, nata con l’esigenza di fornire **nativamente** i servizi di Adobe Flash e simili e di garantire una **maggiori compatibilità** tra i diversi browser. Sono stati infine implementati anche **piccoli controlli** sugli inserimenti degli utenti prima solo

CSS



CSS o **Cascade Style Sheet** è il linguaggio usato per definire lo **stile** delle pagine web. Come da nome è un linguaggio “a cascata”, quindi le regole di stile che vengono dopo **si aggiungono** alle precedenti o **sostituiscono** ad esse.

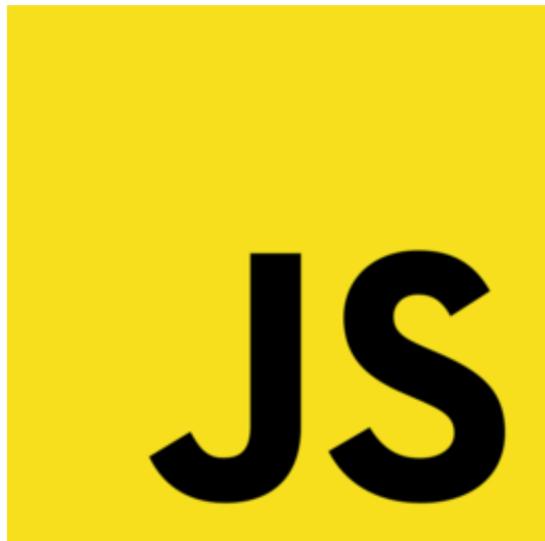
Bootstrap



Bootstrap è un **framework CSS** utilizzato per il **design front-end** e per creare pagine web dal design accattivante, **interattive** ed adattabili ad un'ampia gamma di prodotti (Desktop, laptop, mobile).

Bootstrap permette l'utilizzo di molteplici **classi di stile predefinite** per la personalizzazione degli elementi del proprio sito web ed ha una **completa compatibilità** con CSS e JavaScript aggiuntivi. Alcuni elementi Bootstrap implementano già del codice JavaScript. L'ultima release stabile di Bootstrap è la 4.6, attualmente la versione 5 si trova in una fase di beta.

JavaScript



JavaScript è un linguaggio di **scripting interpretato**, **debolmente tipizzato**, **client-side**, orientato agli **oggetti** ed utilizzato soprattutto per la gestione degli eventi e per i controlli sui dati inseriti dagli utenti. Javascript solitamente viene usato come **linguaggio integrato** all'interno di altro codice (per esempio HTML) e non stand-alone. La **gestione dei controlli** sui dati degli utenti con Javascript **lato client** permette di risparmiare notevoli quantità di traffico internet e **risorse server** che altrimenti verrebbero impiegate "inutilmente" ogni volta che l'utente inserisce un dato errato.

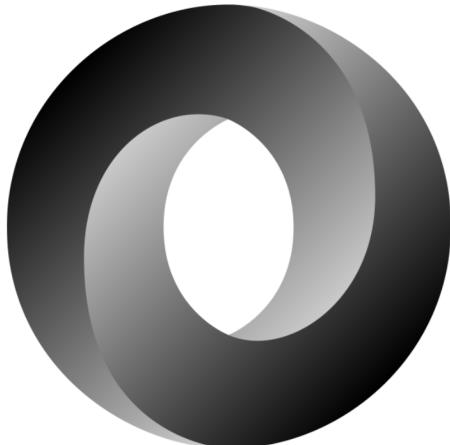
JQuery



JQuery è una **libreria JavaScript** che nasce con l'obiettivo di **semplificare** la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine

HTML, nonché semplificare l'uso di funzionalità AJAX, la **gestione degli eventi** e la manipolazione dei CSS. Per la seguente esperienza è risultato particolarmente utile il suo utilizzo per implementare il **controllo dei dati** inseriti dall'utente e per far **comunicare** client e server.

Il formato JSON



JSON (**JavaScript Object Notation**) è un semplice **formato per lo scambio di dati**. Per le persone è facile da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi. Si **basa** su un sottoinsieme del linguaggio di Programmazione **JavaScript**. JSON è un **formato di testo** completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni comuni ai più diffusi linguaggi di programmazione, come: C, C++, C#, Java, JavaScript, Perl, Python, e molti altri. Questa caratteristica fa di JSON un linguaggio **ideale per lo scambio di dati**. JSON è basato su due strutture:

- Un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.
- Un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

Queste sono strutture di dati universali. Virtualmente tutti i linguaggi di programmazione moderni li supportano in entrambe le forme. Nella realizzazione del CRUD il formato JSON è stato usato per scambiare dati tra PHP e JavaScript.

PHP



PHP o **Hypertext Preprocessor** è un linguaggio di **scripting di alto livello ed interpretato**. Originariamente nato per la creazione di pagine web dinamiche, oggi viene usato principalmente per la **scrittura del codice server-side**. PHP può anche essere usato per scrivere script a riga di comando o

applicazioni stand-alone con interfaccia grafica. PHP è un linguaggio che presenta una **sintassi molto simile a quelle del C** e, per certi versi, a quella del Perl. Il PHP è caratterizzato da una **debole tipizzazione** e presenta un'ottima **integrazione** con molti diversi **DBMS**. Questo rende, insieme alla grande disponibilità di **API**, il PHP un linguaggio molto gettonato per la creazione di gestionali web. Inoltre il PHP è integrabile con moltissimi altri linguaggi e dispone di **numerosi wrapper**. PHP è un linguaggio interpretato ed in quanto tale è compatibile con le più disparate piattaforme ed implementa sia una programmazione di tipo procedurale che ad oggetti. La sua ultima versione, ad oggi, è la 8.

Le sessioni in PHP

Le sessioni in PHP possono essere usate per **passare un dato** da una pagina all'altra e per **gestire** il processo di **autenticazione** degli utenti. Le sessioni permettono di memorizzare le informazioni dinamicamente tra le diverse pagine di navigazione in modo **diverso dai cookie**. Infatti, quest'ultimi lavorano sul client e memorizzano informazioni su di esso, mentre le prime lavorano lato server. Entrambi sono metodi usati per passare informazioni da una pagina all'altra ed entrambi hanno una **durata prestabilita**. Ma, i cookie usano il PC dell'utente per memorizzare le informazioni quindi se l'utente ha disabilitato i cookie sul browser, la pagina non funziona. D'altra parte però, le sessioni usano più risorse del server. Inoltre, le sessioni non sono lette dai motori di ricerca e questo influenza **l'analisi SEO** (Search Engine Optimization).

Tramite le sessioni è possibile “salvare” un’informazione fornita dall’utente nella pagina A per poi leggerla nella pagina B come descritto dalla schema seguente:



Le sessioni vengono inizializzate tramite l’istruzione **session_start()**. Alla sessione viene associato un numero **ID univoco** per distinguerla dalle altre sessioni degli altri utenti. Una volta inizializzata la sessione, può essere usata per **scrivere e leggere i dati**.

Attenzione, E’ necessario inserire **session_start()** in tutte le pagine che usano la sessione, prima delle altre funzioni che gestiscono la sessione. Quando l’istruzione **session_start()** trova una sessione già aperta, continua ad usare quest’ultima. Soltanto se non la trova ne crea una nuova.

Ed è proprio grazie a questo meccanismo che è possibile passare i dati da una pagina all’altra.

Una volta fatta partire la sessione è possibile utilizzarla in maniera simile ad un array associativo, si possono infatti memorizzare dati all’interno di essa, in un’ipotetica pagina A, mediante l’istruzione:

```
$_SESSION['username'] = $username;
```

Dati che poi possono essere “ripresi” in un’ipotetica pagina B nel seguente modo:

```
$username = $_SESSION['username'];
```

Questo meccanismo può risultare utile nel caso in cui, per esempio, la pagina B sia una pagina ad accesso limitato (magari accessibile solo dagli amministratori del nostro gestionale). In tal caso si potrebbero prevedere tutti i controlli sul login nella pagina A che, manda lo username dell’utente alla pagina B solo se esso riesce a loggarsi. A sua volta quindi, la pagina B lascerà entrare l’utente solo se si è loggato e quindi **`$_SESSION['username']`**; esiste, altrimenti lo reindirizzerà nuovamente alla pagina A di login.

Inoltre, è buona norma cancellare le variabili di sessione e distruggere quest'ultima tutte le volte che si vuole chiudere la sessione utilizzata.

Eliminazione singolo dato dalla sessione

```
unset($_SESSION['username']);
```

Eliminazione di tutti i dati della sessione

```
session_unset();
```

Terminazione della sessione

```
session_destroy();
```

SQL

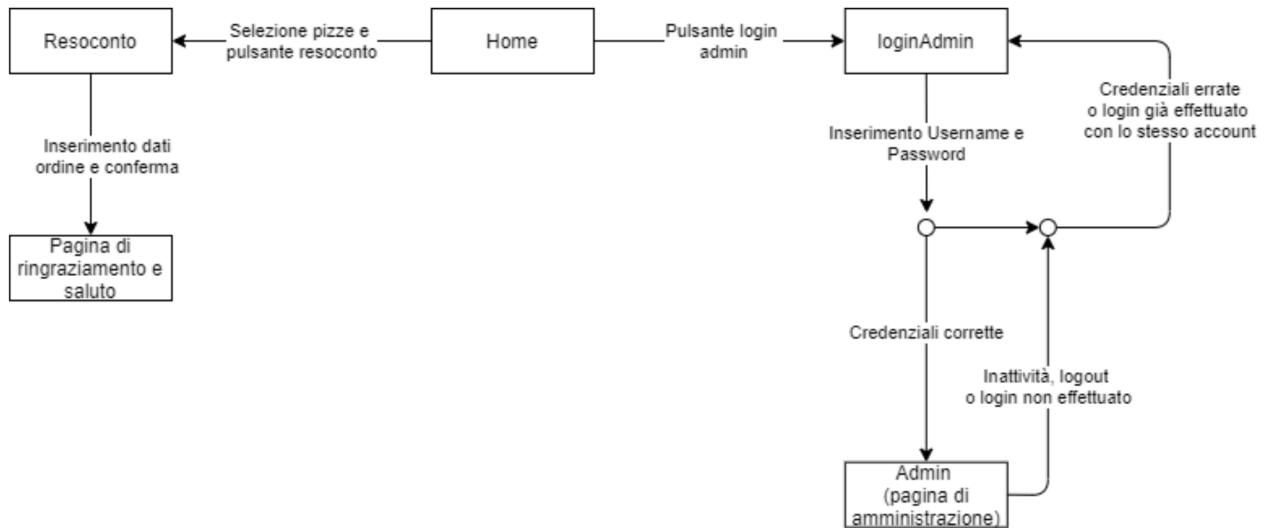


SQL o **Structured Query Language** è un linguaggio standardizzato per database creati secondo il **modello relazionale** ed è di solito utilizzato sfruttando DBMS come MySQL, MariaDB ecc. Il linguaggio SQL è stato progettato per permettere le seguenti operazioni:

- creare e modificare schemi di database (DDL = **Data Definition Language**);
- inserire, modificare e gestire dati memorizzati (DML = **Data Manipulation Language**);
- interrogare i dati memorizzati (DQL = **Data Query Language**);
- creare e gestire strumenti di controllo e accesso ai dati (DCL = **Data Control Language**).

Analisi funzionale

Flusso del programma



La **prima pagina** che viene presentata all'utente è la pagina “**Home**” ospitante il menù e la selezione delle pizze (in generale dei prodotti) che si vogliono ordinare. Il cliente, una volta selezionate le pietanze che intende ordinare, premendo il pulsante “**resoconto**” viene portato alla pagina di “**Resoconto**” in cui gli vengono chiesti i dati necessari per concludere l'ordine. Premendo sull'apposito tasto di conferma l'ordine viene **invia**to e l'utente ringraziato con un'apposita pagina e **reindirizzato** alla homepage. Sempre dalla pagina “**Home**” l'amministratore del gestionale può eseguire l'accesso all'apposita pagina di amministrazione premendo il pulsante di **login per l'admin**. A questo punto verrà presentata la **pagina di login** e verranno chieste le credenziali. Se le credenziali sono corrette e non si sta cercando di effettuare login multipli contemporaneamente con lo stesso account l'amministratore viene fatto accedere alla pagina “**Admin**”, diversamente viene riportato nuovamente alla pagina “**LoginAdmin**”. Nel caso in cui però non si riesca ad accedere perchè in precedenza si è **chiuso il browser senza effettuare il logout** ed ora risulta che si stia cercando di effettuare degli accessi multipli, prima di essere rimandati alla pagina di login si ha la possibilità di effettuare la **disconnessione** per poi **loggare normalmente** una volta reindirizzati su “**loginAdmin**”. Infine anche nel caso di logout o inattività per un certo lasso di tempo l'amministratore viene **slogato** e riportato alla pagina di login.

Idea iniziale

La nostra idea era di creare un sito per le ordinazioni **rapido, fruibile e semplice da utilizzare**.

Lato cliente abbiamo quindi deciso fin da subito di non prevedere una corposa pagina di presentazione della pizzeria (al massimo una piccolissima introduzione) perché volevamo creare qualcosa di **immediato da usare**, sul modello dell'app delle ordinazione del McDonald's. A tal fine prevedevamo già di utilizzare le **cards** bootstrap. Allo stesso modo sul modello della pagina **"Home"**, per la pagina di amministrazione, prevedevamo già di utilizzare comunque le **cards** perché volevamo rendere la **gestione facile**. Per quanto riguarda la pagina **"Resoconto"** anche in questo caso già prevedevamo di mostrare una tabella con il resoconto dell'ordine ed una form per l'inserimento dei dati del cliente.

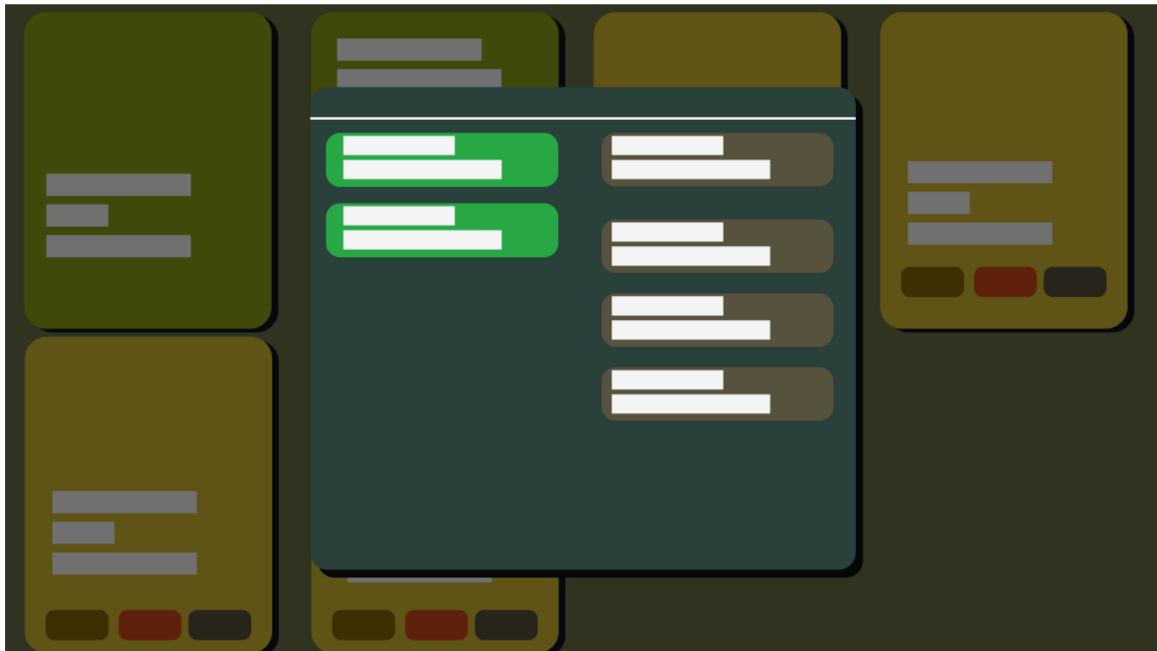
Idea iniziale dell'aspetto grafico del sito

Per avere un'idea iniziale e generale della **grafica** che avrebbe dovuto avere il gestionale finito abbiamo realizzato alcune bozze con **Adobe Illustrator**, anche per capire come avremmo dovuto realizzare il **sito** perché avesse le caratteristiche sopra illustrate.



Siamo partiti definendo la bozza della **pagina di amministrazione** (qui riportata). Al riguardo avevamo già previsto le **card modificabili** con i prodotti presi dal database, la **card di aggiunta** di una nuova pizza e la **card di gestione degli ingredienti**.

Anche la **Modal Box per la gestione degli ingredienti** per ogni prodotto era già stata pensata fin da subito perchè volevamo che la gestione dei prodotti risultasse il più possibile **carina e semplice** da utilizzare:



Abbiamo poi pensato alla realizzazione della pagina di home per permettere al cliente di effettuare le ordinazioni in maniera **semplice e veloce**. Anche per la homepage avevamo previsto ancora una volta le **card** (non modificabili), in maniera **simile** alla pagina di **amministrazione**:



Infine avevamo progettato la **pagina di resoconto** come una semplice pagina contenente una **tabella** con il **riepilogo dell'ordine** ed un **form** di inserimento dei **dati dell'utente**.



Analisi tecnica

Al fine di sfruttare **JQuery e bootstrap** è necessario includere, nel tag “**<head>**” gli script ed i link relativi o scaricare ed includere nel progetto gli appositi pacchetti. Noi abbiamo optato per la **prima soluzione** e come si può notare nella pagina “**admin.php**” abbiamo aggiunto il seguente script per utilizzare la **libreria JQuery**:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

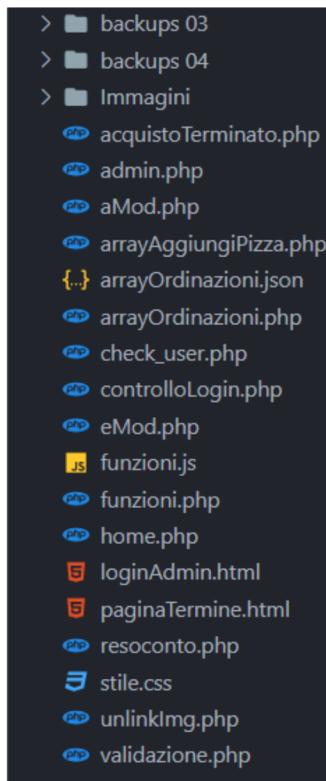
ed i seguenti link e script per sfruttare il **framework bootstrap**:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

Per poi utilizzare le **icone** reperibili al seguente [indirizzo link](#) abbiamo incluso il seguente script:

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.0/font/bootstrap-icons.css">
```

Struttura del CRUD



Il nostro **CRUD** si compone di un **database** definito in linguaggio **SQL** e tramite il DBMS **MySQL** ed il lato **server** è **virtualizzato** mediante **WAMP Server**. Nel dettaglio però, la cartella della nostra esercitazione, “**PizzaExpress**” è composta dai vari **backup** eseguiti per sicurezza durante la fase di sviluppo, dalla cartella “**Immagini**” contenente le immagini dei prodotti caricate dall’amministratore al momento dell’aggiunta dei nuovi prodotti e da vari file PHP, JavaScript, HTML e CSS. Vediamoli nel dettaglio.

File PHP:

- **home.php**: è la pagina mostrata per prima ed utilizzabile dall’utente per effettuare gli ordini.
- **admin.php**: è la pagina di amministrazione protetta da credenziali ed accessibile solamente agli amministratori del gestionale.
- **resoconto.php**: è la pagina di resoconto dell’ordine che viene mostrata al cliente.
- **acquistoTerminato.php**: è la pagina contenente le query da effettuare al momento della conclusione dell’ordine.
- **aMod.php**: è il file contenente le query necessarie per l’aggiunta al DB degli ingredienti contenuti nelle relative pizze.
- **eMod.php**: è il file contenente le query necessarie per l’eliminazione dal DB degli ingredienti contenuti nelle relative pizze.
- **arrayAggiungiPizza.php**: è il file contenente le istruzioni necessarie alla creazione dell’array che poi verrà riempito o svuotato nel relativo file JSON in base agli ingredienti aggiunti o eliminati dall’utente all’atto dell’aggiunta di un nuovo prodotto.
- **funzioni.php**: è il file contenente funzioni PHP condivise dai vari file del progetto.
- **validazione.php**: è il file contenente varie query da eseguire se la validazione dei dati inseriti nella pagina di amministrazione ha avuto successo.

- **controlloLogin.php**: è il file contenente le query di controllo per l'accesso degli utenti alla pagina di amministrazione.
- **check_user.php**: è il file contenente i controlli sullo stato di attività dell'utente loggato nella sezione di amministrazione.
- **unlinkImg.php**: è il file contenente il codice necessario all'eliminazione dell'immagine di un nuovo prodotto nel caso in cui l'inserimento non vada a buon fine.
- **arrayOrdinazioni.php**: è il file per la presentazione della modal contenente gli ordini eseguiti o meno lato amministrazione.

File JSON:

- **arrayOrdinazioni.json**: è il file contenente l'array con lo stato degli ordini (in attesa o consegnato).

File JavaScript:

- **funzioni.js**: è il file contenente tutte le funzioni javascript e quindi, Ajax, utilizzate.

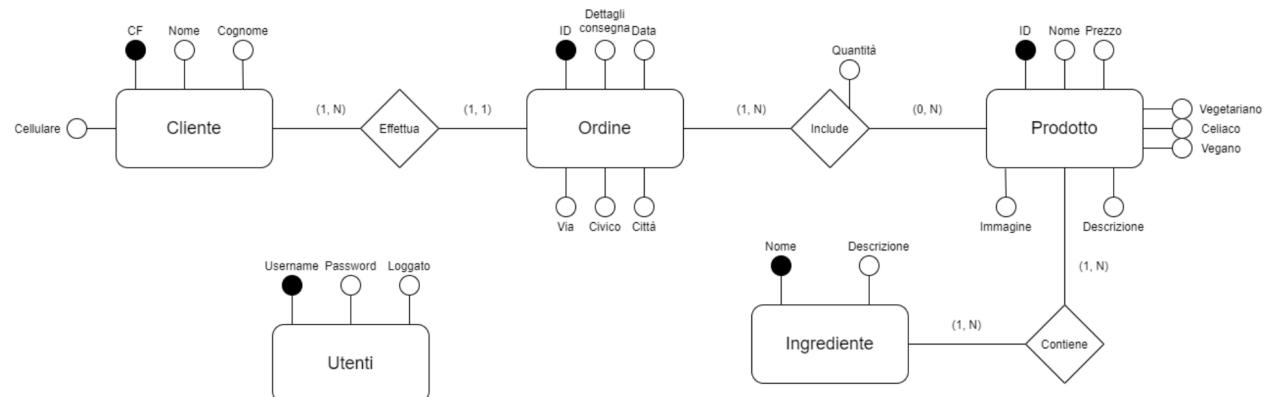
File CSS:

- **stile.css**: è il file contenente le regole di stile aggiuntive al bootstrap.

File HTML:

- **loginAdmin.html**: è la pagina di richiesta delle credenziali di login per accedere alla sezione di amministrazione.
- **paginaTermine.html**: è la pagina mostrata all'utente una volta concluso l'ordine.

Schema concettuale del database



Riguardo al database non c'è molto da dire perché non sono state fatte **scelte particolari** e lo schema risulta **chiaro**, l'unica precisazione necessaria riguarda la tabella “Utenti” e si può trovare nella seguente [sezione](#) del documento. Inoltre si evidenzia come si sia scelto di dare una

definizione **molto generica** ai prodotti trattati per permettere la vendita di più prodotti diversi e **non solo pizze**.

Schema logico del database

Cliente(CF, Nome, Cognome, Cellulare)

Ordine(ID, CF*, DettagliConsegna, Via, Civico, Citta, Data)

Include(IDOrdine*, IDProdotto*, Quantita)

Prodotto(ID, Nome, Prezzo, Immagine, Descrizione, Vegetariano, Celiaco, Vegano)

Contiene(IDProdotto*, NomelIngredient*)

Ingrediente(Nome, Descrizione)

Utenti(Username, Password, Loggato)

Creazione delle tabelle e tipi scelti

cliente(cf varchar(16) primary key, nome varchar(30) not null, cognome varchar(30) not null, cellulare varchar(10) not null)

ordine(id int primary key AUTO_INCREMENT, cf varchar(16), dettagliconsegna varchar(200), data date not null, via varchar(38) not null, civico int not null, città varchar(35) not null, foreign key (cf) references cliente(cf))

prodotto(id int primary key AUTO_INCREMENT, nome varchar(35) not null, prezzo float not null, descrizione varchar(100), vegetariano boolean not null, celiaco boolean not null, vegano boolean not null, immagine text not null)

ingrediente(nome varchar(35) primary key, descrizione varchar(100))

contiene(idProdotto int, nomelIngredient varchar(35), primary key(idProdotto, nomelIngredient), foreign key (idProdotto) references prodotto(id), foreign key (nomelIngredient) references ingrediente(nome))

include(idOrdine int, idProdotto int, quantita int not null, primary key(idOrdine, idProdotto), foreign key (idProdotto) references prodotto(id), foreign key (idOrdine) references ordine(id))

utenti(username varchar(30) primary key, password md5(32), loggato boolean)

La lunghezza dei varchar è stata determinata **di caso in caso** in base alle lunghezze necessarie al relativo campo. Il tipo di dato che può risultare più insolito è il tipo della password, **md5**. Il tipo **md5** è un tipo di dato **criptato** che assicura la sicurezza della password. Per cui, al momento

della registrazione di un nuovo utente la password deve essere inserita crirandola mediante la funzione: ***md5("password")***. Si precisa inoltre che tutti gli **id** che potevano essere gestiti mediante il tipo “**Auto_increment**” sono stati gestiti con quest’ultimo perché lato amministrazione non viene richiesto di poter specificare gli id dei prodotti e questo ci permette di **demandare la loro gestione al sistema, facilitando** le operazioni di amministrazione.

JQuery (AJAX) e JSON

```
function ajaxMover2(nome, array) {  
    array.push(nome);  
  
    array=JSON.stringify(array);  
  
    var file = 'arrayAggiungiPizza.php';  
  
    fetch(file, {  
        method: 'post',  
        body: JSON.stringify({jsonData: array}),  
        Headers:{'Content Type': 'application/json; charset=utf 8'}  
    }).then(function () {  
        refresh(0);  
    })  
    .catch(err => console.error(err));  
}
```

La **funzione javascript ed AJAX “ajaxMover2”** soprastante viene richiamata quando l’amministratore, nell’atto di aggiungere un prodotto al database, **definisce gli ingredienti** che lo compongono. Dato che la pizza che l’amministratore sta aggiungendo nel database **ancora non esiste**, non sarebbe stato possibile caricare direttamente gli ingredienti nel database e non avrebbe avuto senso dato che l’utente può benissimo decidere **di non aggiungere** più quel prodotto nel mentre. Quindi si è deciso di passare, tramite il formato **JSon**, un array contenente gli **ingredienti** che l’admin ha **aggiunto fino a quel momento** ed a cui vengono aggiunti mano a mano che l’amministratore li aggiunge al nuovo prodotto mediante la funzione sopra riportata. Array che, sempre dal JSon, verrà letto dal metodo **PHP “StampalngredientiAggiungiDB”** per mostrare gli ingredienti non compresi nella pizza ed aggiungibili e dal metodo **“StampalngredientiAggiungiDB2”** per mostrare gli ingredienti contenuti nell’array e quindi, già aggiunti alla pizza. Nel caso in cui si elimini un ingrediente dal prodotto che si sta aggiungendo, e quindi dall’array, viene invece invocato il metodo Javascript **“ajaxMover3”**.

```

function refresh(id) {
    var id = '#mod2' + id;
    $(id).load('admin.php ' + id, id);
}

```

La funzione **JavaScript ed AJAX** soprastante invece si occupa di aggiornare solo una parte della pagina di amministrazione, in particolare le **modal box**. E' stato necessario usufruire dell'**AJAX** a tal fine perché tramite il **PHP** è possibile ricaricare (tramite submit) l'**intera pagina**, ma non solo un elemento di essa e questo risultava sgradevole all'utente dato che così facendo la **modal box** si sarebbe **chiusa e solo successivamente riaperta**. Invocando l'apposita funzione **AJAX** (JavaScript), al contrario, possiamo indicare, specificando pagina ed id, l'elemento specifico da ricaricare.

Upload delle immagini

Javascript

```

$(".imgAdd").click(function () {
    $(this).closest(".row").find('.imgAdd').before([
        <div class="col-sm-2 imgUp"><div class="imagePreview"></div><label class="btn btn-primary">Upload</label>
    ]);
    $(document).on("click", ".i.del", function () {
        $(this).parent().remove();
    });
    $(function () {
        $(document).on("change", ".uploadFile", function () {
            var uploadFile = $(this);
            var files = !this.files ? this.files : [];
            if (!files.length || !window.FileReader) return; //controllo sul supporto e la selezione del file da caricare

            if (/^image/.test(files[0].type)) { // solo file di immagini sono ammessi
                var reader = new FileReader(); // inizializzazione del file reader
                reader.readAsDataURL(files[0]); // lettura del file locale

                reader.onloadend = function () { // settaggio dell'anteprima dell'immagine caricata
                    uploadFile.closest(".imgUp").find('.imagePreview').css("background-image", "url(" + this.result + ")");
                }
            }
        });
    });
});

```

Il codice **JavaScript** soprastante, come si può leggere, si occupa di gestire le anteprime delle immagini caricate per l'aggiunta di un nuovo prodotto ed i primi controlli sul file caricato.

PHP

```
//funzione di upload delle immagini al momento dell'aggiunta di un prodotto dalla pagina di amministrazione
function uploadImage()
{
    $target_dir = "Immagini/";
    $target_file = $target_dir . basename($_FILES['fileToUpload']['name']);
    list($width, $height) = getimagesize($_FILES["fileToUpload"]["tmp_name"]);

    $uploadOk = 1;
    $imageFileType = pathinfo($target_file, PATHINFO_EXTENSION);
    // Controlla se il file è un'immagine
    if (isset($_POST["a"])) {
        $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);

        if ($check !== false) {
            $uploadOk = 1;
        } else {
            return "caricareImg";
            $uploadOk = 0;
        }
    }
    // Controlla se l'immagine è già stata caricata sul server
    if (file_exists($target_file)) {
        return "Esiste";
        $uploadOk = 0;
    }
    // Controlla le dimensioni dell'immagine
    if ($_FILES["fileToUpload"]["size"] > 2000000) {
        return "Big";
        $uploadOk = 0;
    }
    // Controllo sul formato delle immagini
    if (
        $imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
        && $imageFileType != "gif"
    ) {
        return "type";
        $uploadOk = 0;
    }
    // Controllo sulla riuscita del caricamento dell'immagine
    if ($uploadOk == 0) {
        return "noImg";
        // Se è tutto ok carica l'immagine
    } else {
        if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {

            return $target_file;
        } else {
            return "errGenNellUpload";
        }
    }
}
```

Il codice **PHP** soprastante invece, si occupa sempre dell'**upload delle immagini**, ma ha il compito di **controllare** meglio l'immagine caricata e quindi, se è già presente nella cartella delle immagini, se è del formato e delle dimensioni giuste ecc. Se tutti i controlli vanno **a buon fine** l'immagine viene **caricata** nell'apposita cartella del server ed il link nel relativo campo del database.

Gestione dei login

Al fine di **proteggere la pagina di amministrazione** e renderla accessibile esclusivamente agli admin, è stata aggiunta una tabella “**utenti**” al database contenente “**username**” (primary key), “**password**” (crittografata mediante la funzione **md5()**) e “**loggato**” (valore **booleano**. vale 1 se l'utente è loggato e 0 altrimenti). Il procedimento del login segue il flusso qua schematizzato ed è gestito nel file “**controlloLogin.php**” che, verifica se l'utente è autorizzato a loggare e se è già loggato su un altro browser o dispositivo. Se il login va a buon fine, l'utente viene reindirizzato sulla pagina “**admin.php**”, che a sua volta reindirizza gli utenti non loggati alla pagina “**loginAdmin.html**”. Altrimenti viene riportato alla pagina “**loginAdmin.html**”.

Una volta che l'amministratore si trova in “**Admin.php**”, può essere disconnesso per due motivi:

1. perché effettua il logout esplicitamente premendo sull'apposito bottone
2. per inattività

Logout volontario

```
if(isset($_POST['logout']))
    logoutBottone($conn, $_SESSION['username']);
```

L'utente preme il **bottone di logout** e viene invocata l'apposita funzione che termina la sessione, pulisce il contenuto delle sue variabili e setta lo **stato della connessione** sul database a **disconnesso** (loggato = 0).

```
function logoutBottone($conn, $username)
{
    $query = "UPDATE utenti SET loggato = 0 WHERE username='$username'";
    $result = mysqli_query($conn, $query);

    $_SESSION = array();
    unset($_SESSION);

    session_destroy();
    echo "Disconnessione riuscita, arrivederci!";
    echo "<script>
        window.location.href='admin.php';
    </script>";
}
```

Logout per inattività

In “**Admin.php**” è presente il seguente codice che, quando viene aggiornata la pagina, controlla da quanto tempo (in secondi) l'utente è **inattivo**. Se l'utente è stato inattivo per un tempo inferiore al tempo necessario definito per la disconnessione il “**timer**” viene azzerato. Altrimenti l'utente viene disconnesso e riportato alla **pagina di login**.

```
if(isset($_SESSION['ultimo_login'])){
    if(time() - $_SESSION['ultimo_login'] > 600){
        logoutAuto($conn, $_SESSION['username']);
    }
    else{
        $_SESSION['ultimo_login'] = time();
    }
}?
```

Ma il codice soprastante controlla il tempo di inattività solamente quando viene aggiornata la pagina di amministrazione, per controllare **continuamente** il tempo di inattività dell’utente si è ricorsi nuovamente all’**AJAX** con il codice seguente:

```
<script type="text/javascript">

    setInterval(function(){
        check_user();
    }, 2000);

    function check_user(){
        jQuery.ajax({
            url: 'check_user.php',
            type: 'post',
            data: 'type = ajax',
            success: function(result){
                if(result == "1"){
                    window.location.href='admin.php';
                }
            }
        });
    }

</script>
```

che richiama, ogni 2 secondi, il file “**check_user.php**” che, si occupa di verificare il tempo di inattività dell’utente:

```
<?php

require 'funzioni.php';

$host = "localhost";
$user = "root";
$pass = "";
$db = "PizzaExpress";
$ids;

$conn = @mysqli_connect($host, $user, $pass, $db);

if (mysqli_connect_errno()) {
    die("Connessione fallita: " . mysqli_connect_error());
}
session_start();

if(isset($_SESSION['username'])){

    if(time() - $_SESSION['ultimo_login'] > 600){
        echo 1;
        logoutAuto($conn, $_SESSION['username']);
    }
}

?>
```

Se l’utente è **inattivo** per un periodo superiore al tempo specificato in secondi (in questo caso 600, 10 minuti) il codice ritorna il valore “**1**” che, verificato dal codice **AJAX** di “**Admin.php**”, porta al **refresh della pagina e la conseguente disconnessione** determinata dal controllo sul tempo di inattività al momento del refresh.

Gestione degli url

Ogni **pagina** che **non** deve essere **raggiunta direttamente dall'utente** **reindirizza** quest'ultimo altrove, a seconda della sezione in cui stava navigando e da cui ha cercato di accedere direttamente a qualche **url**. Questo controllo è stato implementato sfruttando o delle **costanti** o controllando se degli elementi della **pagina precedente** da cui l'utente deve necessariamente passare per arrivare ad una specifica pagina sono **settati o meno**. Per esempio, se l'utente si trova in “home.php” e cerca di raggiungere direttamente tramite url “funzioni.php”, in quest’ultima pagina è presente il seguente controllo:

```
//controllo per non far accedere direttamente l'utente tramite url  
if (!defined("CHECK_RICHIESTA")) {  
    header("Location: home.php");  
}
```

che verifica se la **costante**, definita nelle pagine che usano le funzioni di “**funzioni.php**”, è stata definita. Se la costante non è stata definita (e **non lo sarà mai** quando l'utente esegue la ricerca diretta tramite url) l'utente viene reindirizzato a “**home.php**”. In questo modo l'accesso a “**funzioni.php**” è permesso alle sole pagine che ne utilizzano le funzioni.

Soluzione dei problemi con la validazione dei dati

Avendo creato le pagine di **amministrazione** e di **resoconto** in **PHP** perché abbiamo bisogno di ottenere dati dal database, necessitiamo di richiamare il **JavaScript dal PHP** per effettuare i controlli sui dati inseriti dagli utenti. Questo può creare problemi nel passaggio di dati tra i due linguaggi dato che il primo è **lato server**, il secondo **lato client**. Inoltre si sono riscontrati problemi a causa della **non tipizzazione** dei dati dei due linguaggi che, restii a comunicare tra di loro, creavano confusione con i tipi dei dati passati rendendoli di fatto **inutilizzabili**. Per risolvere questo problema ci si è rivolti alla libreria JavaScript **JQuery** ed alle relative funzionalità **AJAX**. Questo perché tutti i nostri problemi nascevano al momento della restituzione della variabile di validazione (true se era andata a buon fine e false altrimenti) **dal JavaScript al PHP**. Sfruttando **AJAX** possiamo invece evitare il passaggio di variabili e richiamare, una volta effettuati tutti i controlli necessari sui dati, il file **PHP** contenente la funzione che vogliamo eseguire. Infine sono anche stati **gestiti i caratteri speciali** (che non possono essere inseriti dall'utente) che causano problemi con i linguaggi usati, in particolare il **singolo apice ed il back slash**.

Manuale utente

Manuale per il cliente



La prima pagina presentata è “**home.php**”, presentata di default o raggiungibile tramite url. Per prima cosa viene mostrata la foto del locale che, **scorrendo** in basso, **si dissolve** e lascia il posto al menu dei prodotti ordinabili.

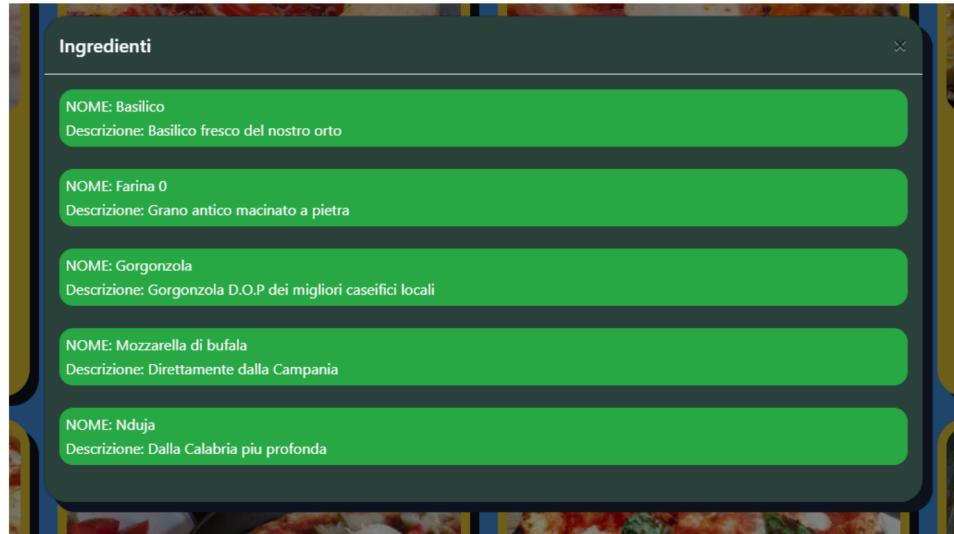
The screenshot shows a grid of four pizza options on a blue background:

- Napoli**: Prezzo: 7 €. A volte pochi ingredienti fanno la differenza. Forza Napoli. Vegetariano 1, Vegano 0, Celiaco 0. Ingredienti: [button]
- Carciofi e burrata**: Prezzo: 7 €. Il morbido incontra lo spinoso in una accoppiata eccezionale. Vegetariano 0, Vegano 0, Celiaco 0. Ingredienti: [button]
- Margherita senza glutine**: Prezzo: 7 €. La classica adatta a tutti. Vegetariano 0, Vegano 0, Celiaco 0. Ingredienti: [button]
- 4 Stagioni**: Prezzo: 8 €. Quattro opposti uniti in un'unica terra. Vegetariano 1, Vegano 0, Celiaco 0. Ingredienti: [button]

At the bottom left is a small image of a slice of Margherita pizza. At the bottom are navigation buttons: "login admin", "FILTRA INGREDIENTI", and "PAGAMENTO".

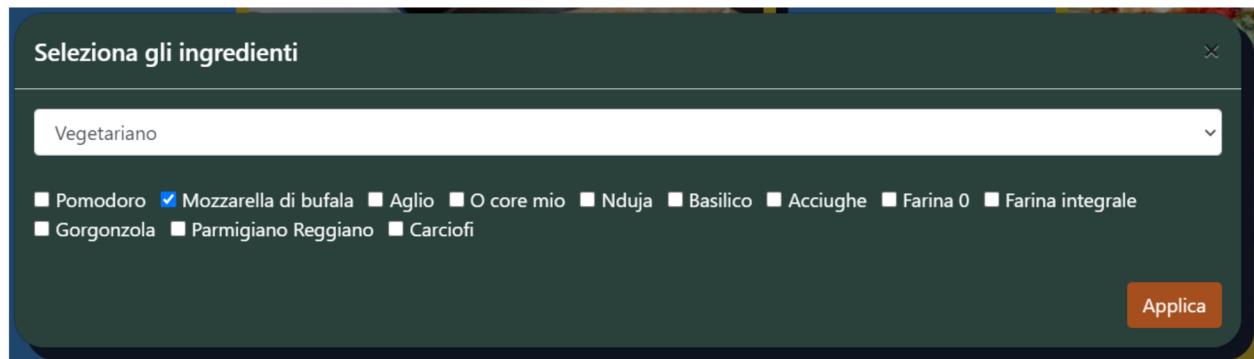
Scendendo troviamo quindi tutto il **menu** del ristorante con le **immagini** dei prodotti, il **nome**, la **descrizione** e la **tipologia** (vegetariano, vegano e celiaco). Per selezionare le pietanze che si intendono ordinare è sufficiente **aumentare o diminuire** la quantità desiderata tramite le

apposite frecce. La barra in basso, **il footer**, rimane **sempre presente** dopo aver scrollato in basso dall'immagine iniziale per permettere di accedere alle sue funzionalità senza dover raggiungere ogni volta la parte più bassa della pagina.

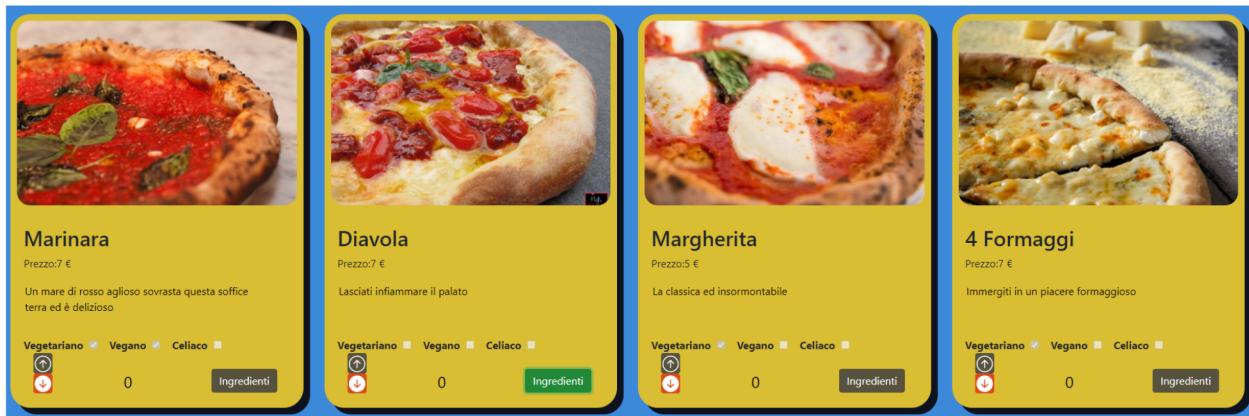


Premendo sul pulsante “**Ingredienti**” del prodotto interessato è possibile consultare la **lista degli ingredienti** che contiene.

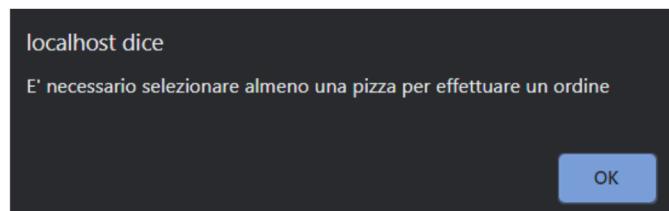
È inoltre possibile applicare un **filtro** sui prodotti selezionando la **tipologia di prodotto** a cui siamo interessati (vegano, celiaco, vegano e celiaco ecc.) e gli **ingredienti che deve includere** (pomodoro, mozzarella). Sarà quindi possibile **ricercare** gli ingredienti ed i prodotti che ci **aggradano**, ad esempio una pizza vegetariana con la mozzarella di bufala.



Applicando il filtro ci verranno mostrati solo i **prodotti con le caratteristiche ricercate**, la pagina verrà **ricaricata** e sarà possibile continuare ad **effettuare** i propri **acquisti**.

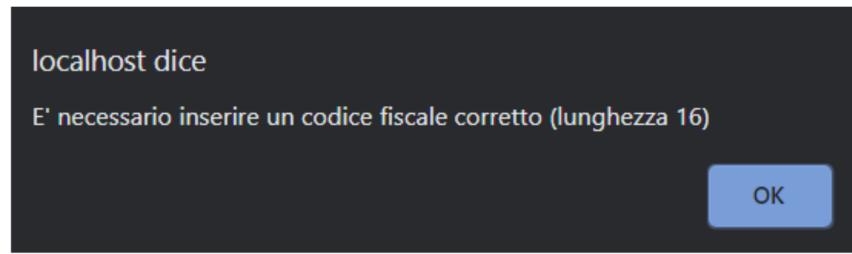


Una volta selezionate le pietanze che si intendono ordinare è sufficiente premere il pulsante “**Pagamento**” del footer per essere portati alla pagina “**resoconto.php**” e concludere l’ordine. Nel caso in cui non sia stata selezionata alcuna pietanza si fa notare all’utente tramite un alert e si ricarica la pagina home.



INSERISCI I DATI DELL'ORDINE			RESOCONTO ORDINE		
Nome		Cognome		Pizza	
Città		Via	Civico	Quantità	Prezzo parziale
Giacomo		Orsucci		Carciofi e burrata	1 1 X 7€ : 7 €
Montecatini Terme		Corso Roma	30	Margherita senza glutine	1 1 X 7€ : 7 €
I tuoi dati sensibili verranno utilizzati solamente allo scopo di fornirti il nostro servizio. Ci teniamo alla tua privacy!					
Dettagli Ordine Ho fame					
<input type="button" value="Conferma ordine"/>					

Nella tabella a **destra** si può consultare il **riepilogo dell’ordine**. Se si cambia idea sulle pietanze che si intende ordinare è sufficiente **tornare nella home** e rieseguire l’ordine, altrimenti manca solo da inserire i **propri dati** e confermare l’ordine premendo sul pulsante “**conferma ordine**”. L’inserimento dei dati è attentamente **guidato** e ogni qual volta l’utente esegue un errore gli viene **notificato** tramite un alert. La descrizione è **facoltativa** e potrebbe contenere la data e l’ora di consegna, altrimenti la consegna verrà eseguita il prima possibile (il nome “PizzaExpress” non è casuale).



Se tutti gli inserimenti andranno a **buon** fine l'utente verrà ringraziato e **reindirizzato** alla **homepage**.



Manuale per l'amministratore

login

Per **accedere** alla pagina “**admin.php**” ed amministrare il sito è prima necessario effettuare il **login** dalla pagina “**loginAdmin.html**”. La pagina è raggiungibile o tramite il pulsante “**login admin**” presente nella homepage o direttamente tramite url.

A screenshot of a login form set against a dark olive-green background. The form consists of two input fields: one for "USERNAME" and one for "PASSWORD", both with placeholder text ("Inserisci l'username" and "Password" respectively). In the bottom right corner of the form area, there are two blue rectangular buttons labeled "Login" and "Annulla" (Cancel).

Nel caso in cui l'utente **sbagli** le credenziali viene **notificato** l'errore ed ha la possibilità di tornare alla precedente pagina di login.

Identificazione non riuscita: nome utente o password errati
Torna alla pagina di [login](#)

Nel caso in cui invece l'utente sia **già connesso** tramite un altro browser **non** sarà possibile effettuare un altro **login contemporaneamente**. Si consiglia di **verificare** se si è già connessi tramite altri browser e di disconnettersi. Ma, nel caso in cui sia stato chiuso il browser senza effettuare la disconnessione, è necessario effettuare il **logout** tramite il pulsante mostrato e successivamente sarà possibile eseguire il **login normalmente**.

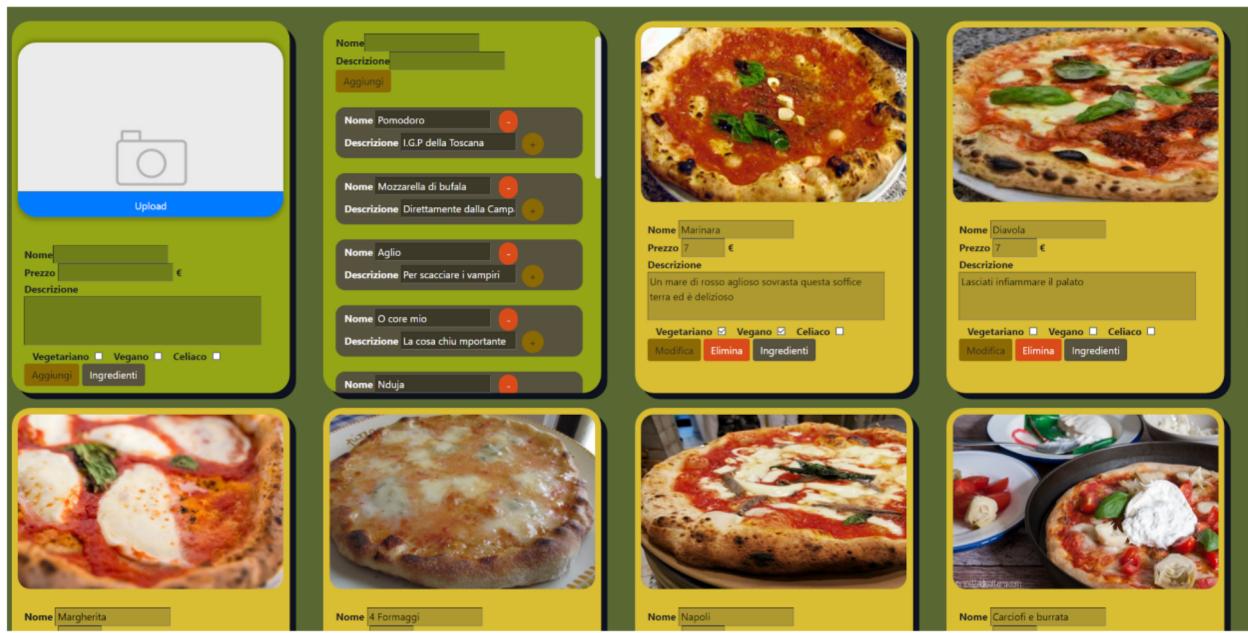
Siamo spiacenti, non si possono eseguire più connessioni contemporaneamente con lo stesso utente.
Torna alla pagina di [login](#).

Se hai chiuso il browser senza sloggare ed ora non riesci più ad accedere premi il pulsante sottostante.
Prima però assicurati di non essere loggato contemporaneamente su altri browser o schede dello stesso browser.

[Logout](#)

Logout effettuato, ora puoi tornare alla pagina di [login](#) e connetterti normalmente

Amministrazione



Una volta acceduti alla **pagina di amministrazione** vengono mostrate le pietanze presenti nel database (interamente modificabili) ed è possibile gestire tutti i prodotti e gli ingredienti.

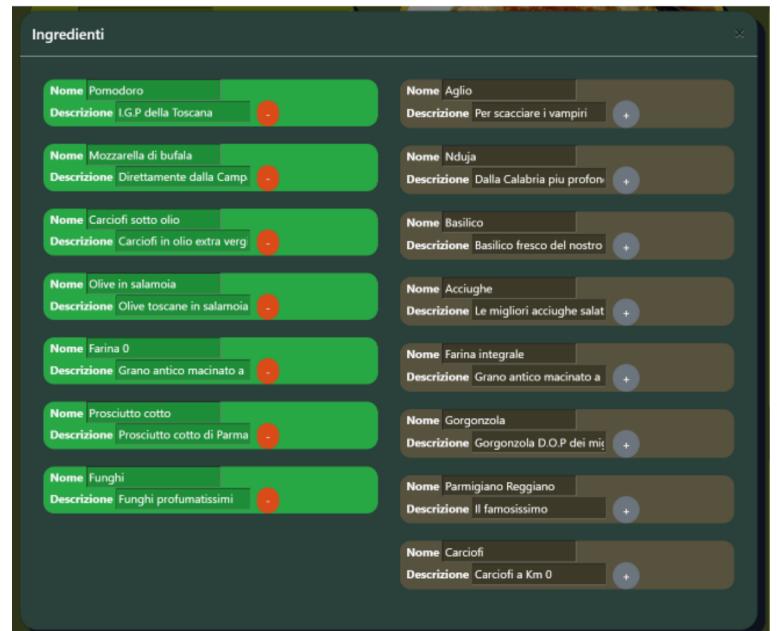
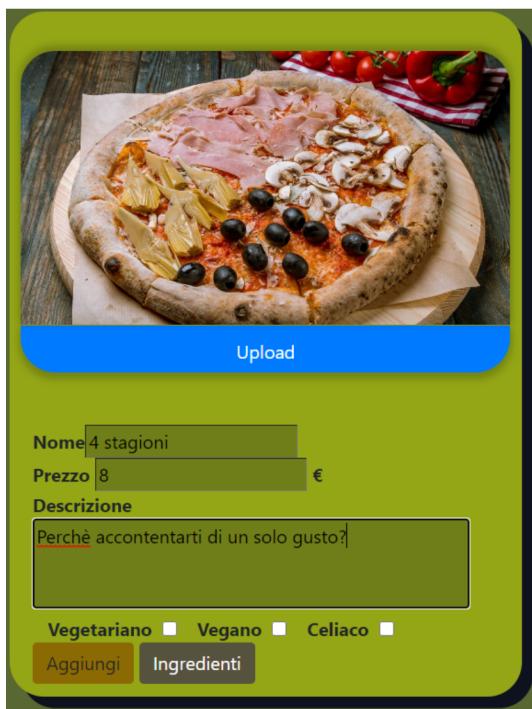
Aggiunta di un nuovo ingrediente



Nella seguente card è possibile **aggiungere nuovi prodotti** indicando il **nome** e la **descrizione** e premendo su “**Aggiungi**”. Nel caso in cui si vogliano gestire quelli già presenti è possibile **modificare** la descrizione e premere sul “+” per rendere la modifica effettiva, o eliminare quelli non più usati. Nel caso in cui si voglia modificare il **nome** è necessario **eliminare** l’ingrediente e ricrearlo.

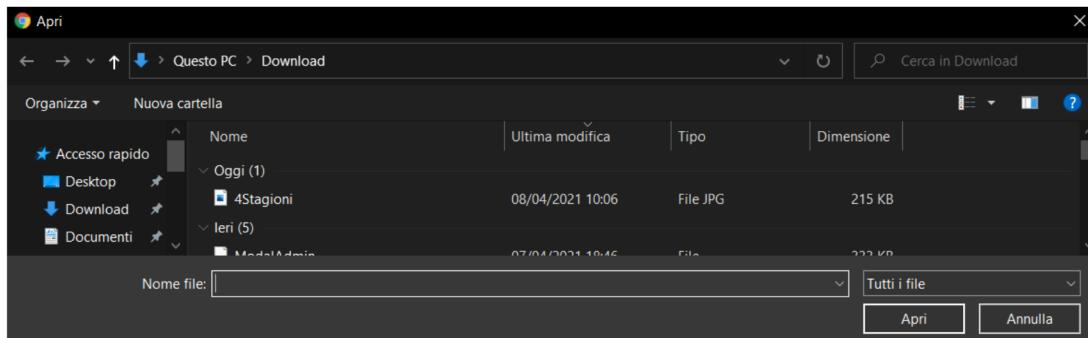
Aggiunta di un nuovo prodotto

Aggiunta degli ingredienti al prodotto



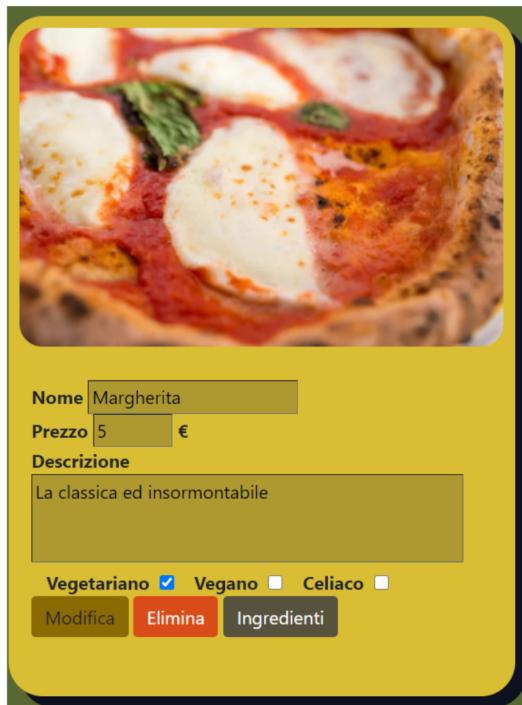
Per aggiungere un nuovo prodotto è necessario **caricare l’immagine** relativa cliccando sull’icona con la scritta “**upload**”, indicare il **nome**, il **prezzo**, la **descrizione**, la **tipologia** e gli ingredienti contenuti premendo sul bottone “**Ingredienti**” nella prima **card a sinistra**. Una volta inserite le informazioni relative al nuovo prodotto è sufficiente premere “**Aggiungi**” per completare l’inserimento.

Upload dell'immagine



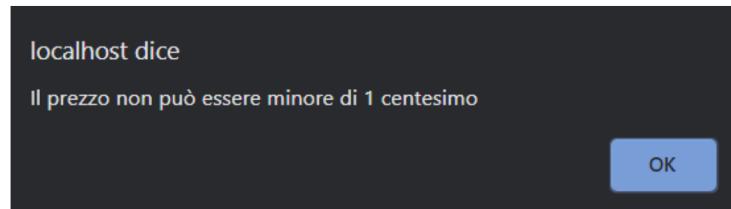
*formati particolari o immagini troppo grandi non sono supportati

Modifica dei prodotti presenti



Ogni **prodotto** può essere gestito tramite la relativa **card**. E' possibile modificarne il **nome**, il **prezzo**, la **descrizione**, il **tipo** e gli **ingredienti** contenuti, ma **non l'immagine**. In tal caso è necessario eliminare la pietanze tramite l'apposito bottone. La modifica degli ingredienti si presenta nello stesso modo **dell'aggiunta** degli **ingredienti** al momento dell'aggiunta di un nuovo prodotto. Premendo su “+” verrà aggiunto l'ingrediente e premendo su “-” verrà tolto. Una volta eseguite le dovute modifiche premendo su “**modifica**” il cambiamento verrà reso **effettivo**.

In caso di **errore** verrà notificato all'utente tramite un **alert** e, una volta che quest'ultimo lo avrà visualizzato e premuto su “**ok**”, la pagina verrà ricaricata senza aver effettuato la modifica o l'aggiunta del nuovo prodotto.



Gestione degli ordini

ORDINI

Premendo sul seguente pulsante è possibile aprire la modal per la **gestione degli ordini**.



Tramite la **modal** soprastante è possibile consultare gli **ordini**, i loro **dettagli** e settarne lo stato (**consegnato o meno**). Premendo “+” si aggiunge l'ordine selezionato a quelli **consegnati**.

Premendo “-” si riporta un ordine dallo stato di **consegnato** allo stato di **attesa**. Sulla sinistra troviamo gli ordini consegnati, a destra quelli ancora da consegnare.

Pizze Ordinate

Tramite il pulsante “**Pizze ordinate**” è possibile consultare la lista delle pizze comprese in ogni ordinazione.



Infine è possibile aggiornare la **modal** in ogni momento tramite il pulsante di aggiornamento.

Criticità riscontrabili e cosa fare

Normalmente non ci sono **criticità riscontrabili**, c'è solo una piccola "noia" da evidenziare. Se si ha un errore sull'inserimento dei **dati** al momento dell'aggiunta di un nuovo prodotto la **pagina** verrà **ricaricata**, ma l'inserimento non sarà effettuato ed il prodotto dovrà essere caricato interamente da capo. Infine si **consiglia** caldamente di aggiornare la pagina se si riscontrano dei problemi e, prima di fare qualsiasi altra azione, **aggiornare** la pagina se si è inattivi da diverso tempo al fine di evitare possibili problemi con il **logout** automatico.

Test data set/debug

Il sito, sia lato **cliente** che lato **amministrazione**, è stato **testato** in tutte le sue funzionalità sia dai programmati che da persone esterne al progetto. L'unico problema riscontrato riguarda il **logout automatico** che, a volte non avviene o che può creare problemi nel caso in cui si aggiunga una pizza e subito si venga disconnessi premendo "**Aggiungi**". Inoltre, **occasionalmente**, qualche elemento grafico potrebbe risultare alterato, ma **niente** che vada ad influenzare il funzionamento della piattaforma o **che renda l'esperienza dell'utente sgradevole**. Il sito è interattivo e supporta diversi tipi di device, ma **sporadicamente** potrebbe presentare **problemi di ridimensionamento**.

Conclusioni ed osservazioni personali

Il risultato finale dell'esercitazione rispecchia e va oltre quello che avevamo pensato di realizzare durante la fase di progettazione. Sono state aggiunte **funzionalità extra** come la gestione dei login nella pagina di amministrazione e si è cercato di rendere la grafica il più possibile **carina** per permettere un uso **efficace**, ma anche **piacevole** del sito. Durante lo svolgimento dell'esercitazione sono state incontrate varie **difficoltà**, inizialmente soprattutto nella **validazione dei dati**, ma sono state superate anche grazie all'apprendimento ed all'utilizzo di altre tecnologie come la libreria JQuery (e quindi l'utilizzo di Ajax) ed il formato JSON. Inoltre anche i **cookies** hanno fornito molti elementi di disturbo perché non sempre si aggiornavano correttamente e frequentemente è stato necessario **forzare**, in un modo o nell'altro, il loro **aggiornamento**. Si è costantemente cercato di curare ogni aspetto nel **minimo dettaglio** ed a tal fine è stata introdotta anche la gestione del **crop delle immagini** dei prodotti presi dal database per evitare che risultassero sgranate. Infine ci teniamo a segnalare, per possibili **sviluppi futuri**, come la piattaforma potrebbe essere arricchita ulteriormente tramite l'introduzione dei **metodi di pagamento**, della **gestione degli utenti** (aggiunta e modifica di nuovi utenti e assegnazione dei permessi) dalla pagina di **amministrazione** (per ora disponibile solo tramite console) o l'aggiunta di una pagina, sempre lato amministrazione, riportante tutte le **statistiche di vendita** di interesse

per il locale e lo **svuotamento dei dati** non più utili (come i vecchi ordini) dal database. Il **codice** è stato scritto in maniera mista in **italiano** ed **inglese** ed è stato **commentato** per evidenziarne la funzione e spiegarne le parti più complicate.

Sitografia e bibliografia

Forum e siti vari

[Stack Overflow](#), [Wikipedia](#), forum, siti vari e disparati

BootStrap

[Getbootstrap](#) , [W3Schools](#)

Sessioni PHP

[HTML.it](#) , [Documento PDF](#) , [Video YouTube](#)

PHP

[PHP.net](#)

Dispense fornite dal professore Pasquale Silvestro, aiuto dei professori e confronto con la classe.