

Diario di lavoro

Luogo	SAMT
Data	08.01.2019

Lavori svolti

Il progetto l'ho iniziato ufficialmente al pomeriggio, perché la mattina li abbiamo divisi e scelti con i compagni e dopodiché ho avuto la presentazione per cui non sono riuscito a lavorare. La prima cosa che ho fatto è stata leggere attentamente il QDC e scrivermi le domande che mi sono sorte. Essendo che il mio docente responsabile non era presente ho inviato il file con le domande e immagino faremo il colloquio la settimana prossima, alla prima lezione di progetti. Dopodiché ho iniziato a fare le varie analisi, ho anche cercato un software per poter fare un gantt decente, con anche segnate le ore impiegate, e non a giorni interi, ma ancora non sono riuscito a trovarlo e quindi a fare il gantt. Oltre a questo ho creato una repository su GitHub e ho inserito tutto ciò che ho fatto per adesso.

Problemi riscontrati e soluzioni adottate

Non ho riscontrati problemi, oltre a non aver trovato un programma decente per il gantt.

Punto della situazione rispetto alla pianificazione

Non ho ancora una pianificazione

Programma di massima per la prossima giornata di lavoro

Completare le analisi e riuscire a fare un gantt.

Diario di lavoro

Luogo	SAMT
Data	09.01.2019

Lavori svolti

Questa mattina ho speso ancora un po' di tempo per cercare un software per fare i gantt ma non ho trovato niente che riesca a esportarli in modo comodo e comprensibile e quindi ho optato per continuare ad utilizzare "Gantt Project" anche se da solo la possibilità di mettere la durata di lavoro in giorni e non è il massimo, ma è la migliore che c'è.

Dopo questo ho riguardato le risposte alle domande che avevo fatto al docente ieri. Dopo averle analizzate sono riuscito a capire meglio cosa devo fare e a iniziare a impostare qualche requisito. Oltre ai requisiti ho anice completato il gantt preventivo, anche se mi ha preso un bel po' della giornata perché dovevo capire bene cosa dovevo fare per poter completare in modo completo quest'ultimo.

Il gantt si può trovare sul git al sito:

<https://github.com/giairommauro/ProgettoVenditaPiccolaNegozianti>. Il gantt è visualizzabile in più modi, sia come immagine, che come file di gantt project, ma anche nella documentazione, tutti questi file sono scaricabili direttamente dal sito d qui sopra, nella documentazione in più sono spiegate le parti "speciali" del gantt.

Questo sito è il github ufficiale del mio progetto in cui verranno inseriti la maggior parte dei documenti, le uniche cose che non ci saranno sono i file e la struttura per il sito internet, perché essendo che cambia ogni volta potrebbe risultare complicato reinserirli ogni volta, ma se dovessi trovare un metodo semplice per importarli inserirò anche quelli.

Oggi ho anche completato l'analisi dei costi e dei mezzi, in modo che ora l'unica analisi che mi manca da completare è quella dei requisiti.

Problemi riscontrati e soluzioni adottate

Non ho riscontrati problemi.

Punto della situazione rispetto alla pianificazione

Sono in linea con il gantt oggi completato.

Programma di massima per la prossima giornata di lavoro

Completare le analisi e iniziare la progettazione.

Diario di lavoro

Luogo	SAMT
Data	15.01.2019

Lavori svolti

Questa mattina ho fatto inizialmente il colloquio con il docente in cui mi sono stati tolti gli ultimi dubbi sul progetto, e così ho completato il file delle domande, visualizzabile sul git al link https://github.com/giairomauro/ProgettoVenditaPiccolaNegozianti/tree/master/Informazioni_progetto.

Una volta finito il colloquio mi sono portato avanti con l'analisi dei requisiti fino a completarlo. Dopodiché ho iniziato a lavorare ai mockup delle pagine del sito, come mostrato qui sotto.

Gestione vendita

[Mappa](#) [Carrello](#)

Categoria



Prodotto XY

Categoria
Prezzo
Quantità

Figura 1 Pagina iniziale

Questa è la pagina iniziale del sito, in cui c'è una lista di tutti i prodotti e la possibilità di ricercarli tramite nome o categoria. All'interno della scheda del prodotto c'è la possibilità, grazie ad un bottone, di inserirlo nel carrello.

Sopra alla lista c'è la barra di navigazione tramite la quale si può accedere alla mappa dei negozi o al carrello.

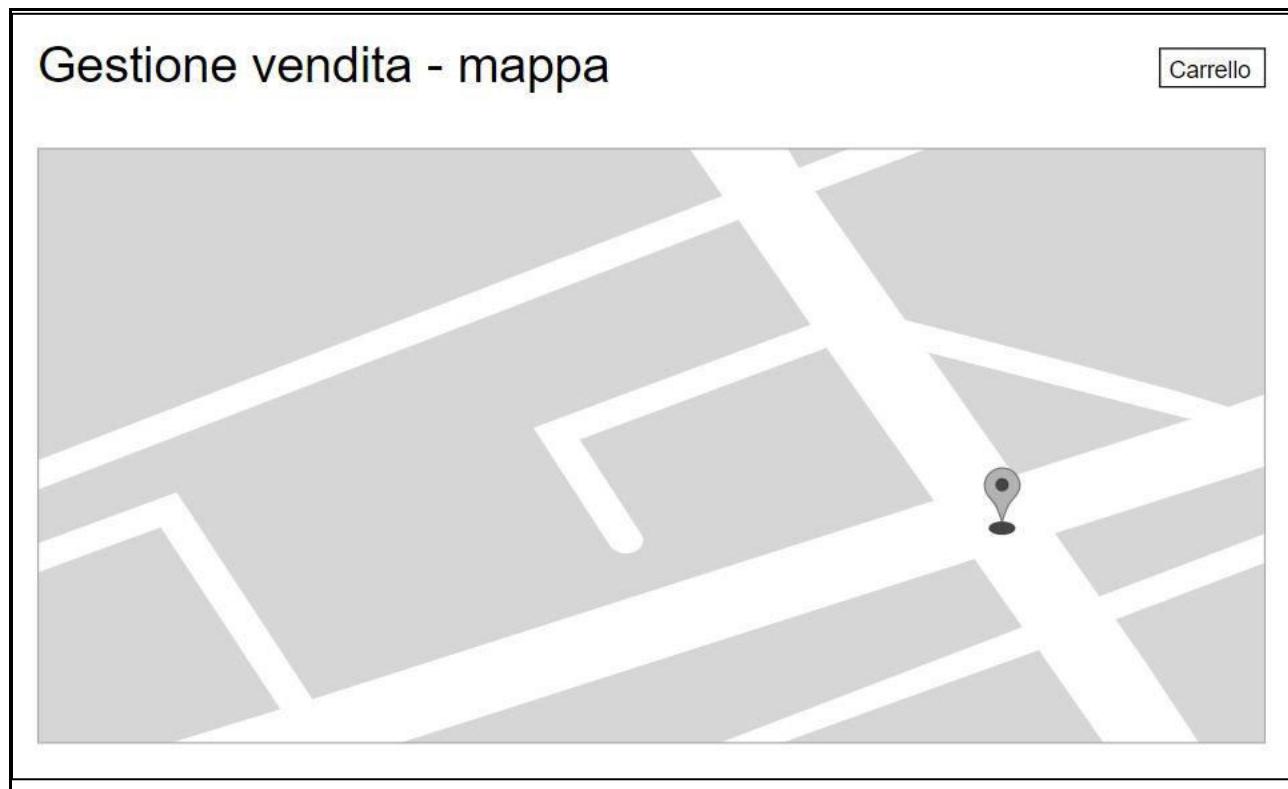


Figura 2 Pagina mappa

Quella mostrata qui sopra è la pagina della mappa, in cui è mostrata una mappa sopra la quale è possibile cliccare dei punti di segnalazione che apriranno un “popup” che mostra le informazioni del negozio che sta lì

Gestione vendita - carrello

[Mappa](#)

Prodotto XY

Categoria
Prezzo
Quantità
Consegna

Nome Stabile
Indirizzo
Telefono
Mail

Prezzo: X Fr.

[Compra](#)

[Stampa](#)

Figura 3 Pagina carrello

L'ultima pagina a cui i clienti hanno accesso è la pagina del carrello, che viene svuotata ogni volta che si acquista o si esce dal browser, ed è strutturata in questo modo, viene mostrato il prodotto con le relative informazioni e al fianco le informazioni del negozio che l'ha messo in vendita.

The screenshot shows a split-screen login form. On the left side, under the heading "Login", there are fields for "Email Address" and "Password", followed by a "Login" button and a "Registrati" link. On the right side, under the heading "Utente ospite", there are fields for "Nome", "Cognome", "Indirizzo email", and "Numero di telefono". Below these fields is a checkbox labeled "Non sono un robot" with a checked mark. At the bottom right of the right section is a "Avanti" button.

Figura 4 Pagina login o utente ospite

Una volta che l'utente clicca su "compra" se non ha ancora fatto il login viene mostrata una pagina in cui può fare il login o accedere come ospite, come mostrato nell'immagine, e se non dovesse essere registrato viene data la possibilità di farlo.

Gestione vendita - registrazione

Nome

Cognome

Indirizzo email

Numero di telefono

Non sono un robot

Avanti

Figura 5 Pagina di registrazione

Questa è la pagina in cui l'utente può registrarsi.

Gestione vendita - login

Indirizzo email

Password

Avanti

Figura 6 Pagina login Negozianti/Amministratori

Gli utenti amministratori o i negozianti hanno la necessità di accedere alle loro pagine quando entrano nella loro pagina dedicata.

Gestione vendita - negozi

Negozi X [Modifica](#) [Archivia](#)

Negozi Y [Modifica](#) [Archivia](#)



Aggiungi negozio

Figura 7 Pagina negozi amministratore

L'amministratore una volta che accede inizialmente vede una pagina con la lista di tutti i negozi e cliccandoci sopra appare un popup che ne mostra le informazioni. Oltre a questo può lavorare con i negozi o aggiungere di nuove.

Gestione vendita - modifica

Negozi

[Modifica](#)

Negoziante

[Modifica](#)

Figura 8 Pagina modifica negozi

La pagina di modifica dei negozi permette di modificare le informazioni del negozio stesso o del negoziante che lo tiene, questo è fatto diviso per impedire che bisogna modificare tutto se si vuole cambiare solo un'informazione in uno dei due punti.

Gestione vendita - aggiunta	
Negozi	Negoziante
<input type="text" value="Nome"/>	<input type="text" value="Nome"/>
<input type="text" value="Indirizzo"/>	<input type="text" value="Cognome"/>
<input type="text" value="Telefono"/>	<input type="text" value="Indirizzo email"/>
<input type="text" value="Mail"/>	<input type="text" value="Numero di telefono"/>
	<input type="text" value="Password"/>
	<input type="button" value="aggiungi"/>

Figura 9 pagina aggiunta negozio

La pagina di creazione di nuovi corsi è molto simile quella dell'aggiunta dei negozi ma in questo caso si aggiunge tutto assieme e non si separano negozio e negoziante e in più si può decidere la password dell'utente, che poi gli andrà convocata una volta creato.

Gestione vendita - prodotti

Categoria ▼ Cerca



Prodotto XY

Categoria
Prezzo
Quantità
Consegna Modifica

+ Aggiungi prodotto

Figura 10 Pagina gestione prodotti

Se il negoziante accede al suo sito invece avrà la possibilità di vedere e cercare tutti i prodotti che ha inserito all'interno del sito, e potrà anche lui decidere se modificare un prodotto o aggiungerne di nuovi.

Prodotto XY

Nome
Categoria
Prezzo
Quantità
Consegna

Modifica



Figura 11 PopUp modifica prodotto

Se si clicca sul bottone “modifica” in un prodotto appare un popup che permette di modificare le informazioni del determinato prodotto, tra cui anche l’immagine mostrata sul sito.

Gestione vendita - Creazione prodotto

Prodotto XY

Nome
Categoria
Prezzo
Quantità
Consegna

Inserisci

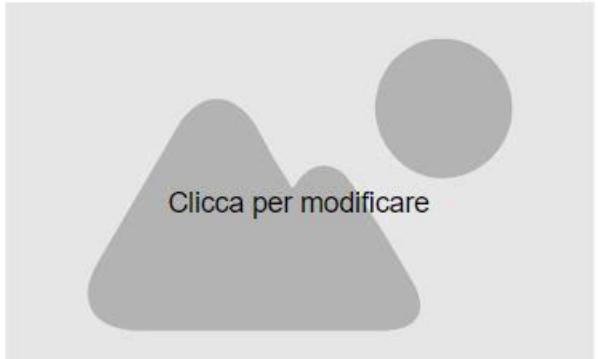


Figura 12 Pagina creazione prodotto

Se si prova ad aggiungere un nuovo prodotto viene visualizzata una nuova pagina, simile al popup ma con la possibilità di aggiungere un nuovo prodotto, quindi con i campi vuoti, e non modificarne uno esistente. Se si lascia l’immagine vuota nel sito verranno mostrati solo i dati ma senza errori.

Una volta finito il mockup l’ho inserito nella documentazione e ho iniziato a cercare un template che fosse comodo per il mio sito e addatto.

Problemi riscontrati e soluzioni adottate

Non ho riscontrati problemi.

Punto della situazione rispetto alla pianificazione

Sono in linea con il gantt.

Programma di massima per la prossima giornata di lavoro

Fare il diagramma E-R.

Diario di lavoro

Luogo	SAMT
Data	16.01.2019

Lavori svolti

Sta mattina ho iniziato a lavorare allo schema E-R.

Mentre lo facevo mi sono ricordato che ancora non avevo concluso i test case nella documentazione e di conseguenza ho finito quelli. Ho fatto solo quelli dei requisiti obbligatori perché quelli opzionali li farò so se effettivamente li implementerò. Dopo averlo fatto ho aggiornato anche la documentazione sul github inserendo quella appena modificata.

Dopodiché ho continuato l'E-R.

Sono riuscito a finire lo schema E-R a brutta e poi l'ho riportato a bella con "visio", poi l'ho sia esportato come pdf sia salvato un'immagine da inserire nella documentazione.

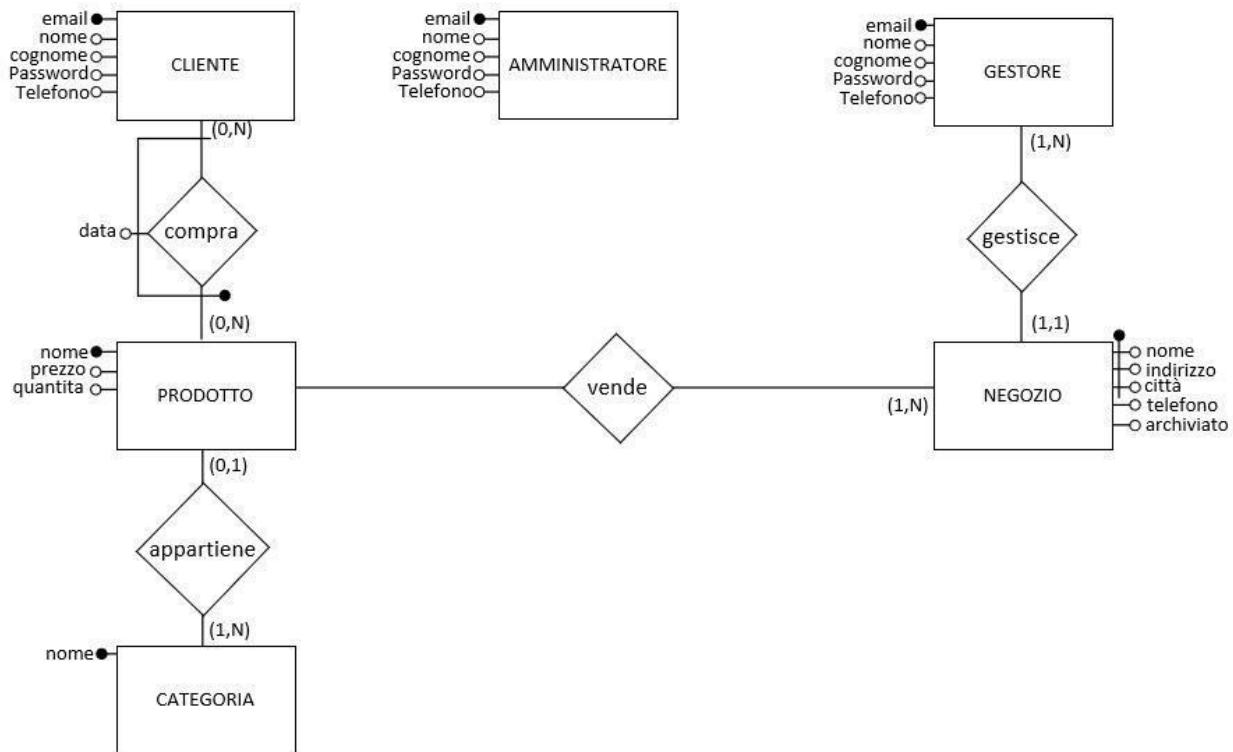


Figura 1 Schema E-R

Questo è lo schema completato, con tutte le tabelle che servono per gestire il sito internet da parte di tutti gli utenti.

Lo schema è spiegato più dettagliatamente nella documentazione, visualizzabile dal github.

Problemi riscontrati e soluzioni adottate

Non ho riscontrati problemi.

Punto della situazione rispetto alla pianificazione

Sono in linea con il gantt.

Programma di massima per la prossima giornata di lavoro

Creare il database.

Diario di lavoro

Luogo	SAMT
Data	22.01.2019

Lavori svolti

La mattina ho caricato tutto sul github e ho modificato la struttura e il file "README.md" per poter accedere anche alle ultime cose aggiunte.

Dopodiché ho riguardato con il docente lo schema E-R e gli ho posto le ultime domande riguardanti esso. Dopo aver analizzato lo schema il docente mi ha dato alcuni consigli su come strutturare le ultime cose, e quindi lo schema ha ricevuto ulteriori modifiche.

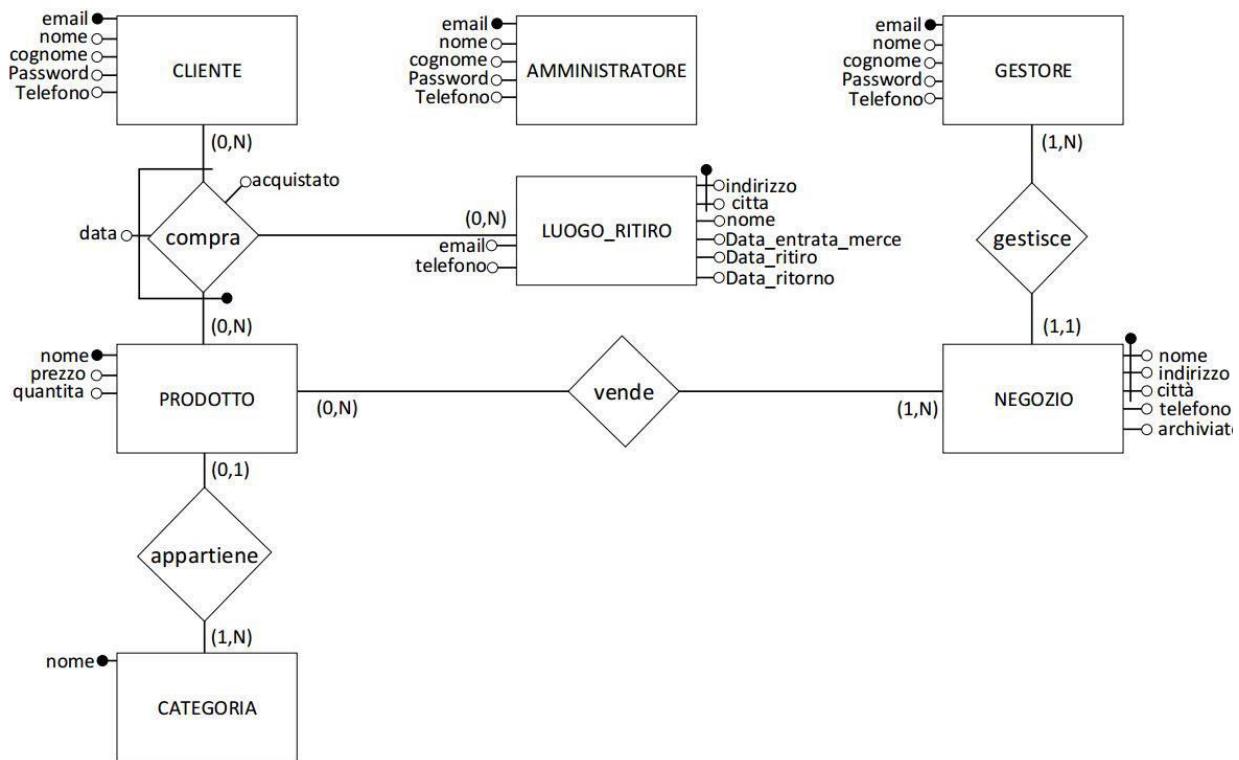


Figura 1 schema E-R prima versione

Questo è lo shcema E-R modificato, si può notare che è stata aggiunta la cardinalità da "PRODOTTO" verso "NEGOZIO", essendo che è stato deciso che lo stesso prodotto può ssere venduto da più negozi.

Oltre a questo è stata aggiunta un'altro attributo alla tabella compra, ovvero "acquistato" questo è un booleano che spiega se i prodotti sono stati comprati o meno, se sono stati comprati entra in funzione l'altra tabella aggiunta "LUOGO_RITIRO" che contiene tutte le informazioni dei luoghi in cui si devono ritirare i prodotti acquistati.

Una volta finito.

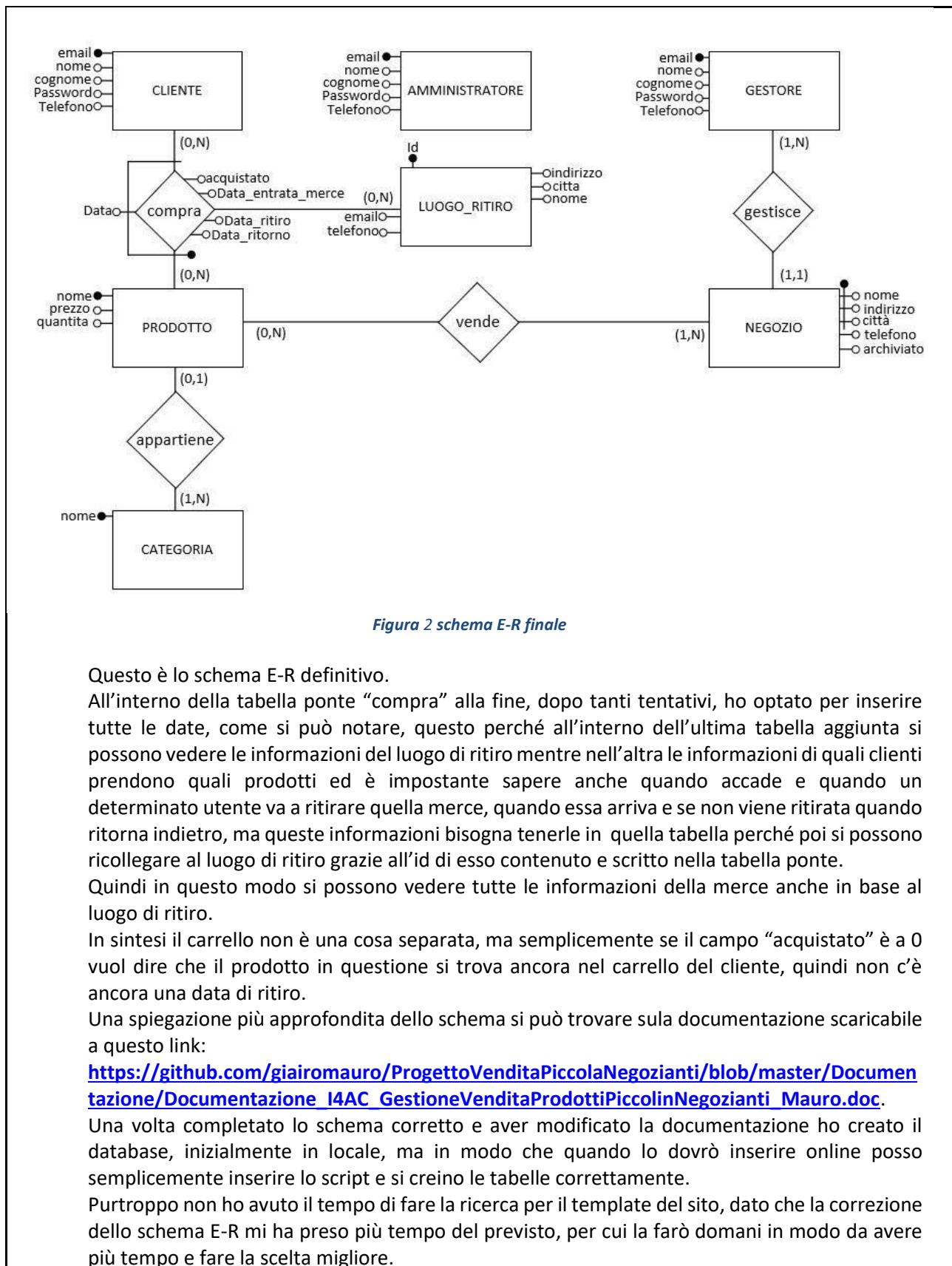


Figura 2 schema E-R finale

Questo è lo schema E-R definitivo.

All'interno della tabella ponte "compra" alla fine, dopo tanti tentativi, ho optato per inserire tutte le date, come si può notare, questo perché all'interno dell'ultima tabella aggiunta si possono vedere le informazioni del luogo di ritiro mentre nell'altra le informazioni di quali clienti prendono quali prodotti ed è impostante sapere anche quando accade e quando un determinato utente va a ritirare quella merce, quando essa arriva e se non viene ritirata quando ritorna indietro, ma queste informazioni bisogna tenerle in quella tabella perché poi si possono ricollegare al luogo di ritiro grazie all'id di esso contenuto e scritto nella tabella ponte.

Quindi in questo modo si possono vedere tutte le informazioni della merce anche in base al luogo di ritiro.

In sintesi il carrello non è una cosa separata, ma semplicemente se il campo "acquistato" è a 0 vuol dire che il prodotto in questione si trova ancora nel carrello del cliente, quindi non c'è ancora una data di ritiro.

Una spiegazione più approfondita dello schema si può trovare sulla documentazione scaricabile a questo link:

https://github.com/giairommauro/ProgettoVenditaPiccolaNegoziante/blob/master/Documentazione/Documentazione_I4AC_GestioneVenditaProdottiPiccoliNegoziante_Mauro.doc

Una volta completato lo schema corretto e aver modificato la documentazione ho creato il database, inizialmente in locale, ma in modo che quando lo dovrò inserire online posso semplicemente inserire lo script e si creino le tabelle correttamente.

Purtroppo non ho avuto il tempo di fare la ricerca per il template del sito, dato che la correzione dello schema E-R mi ha preso più tempo del previsto, per cui la farò domani in modo da avere più tempo e fare la scelta migliore.

Problemi riscontrati e soluzioni adottate

Non ho riscontrati problemi, se non i vari cambiamenti e controlli che ho fatto al database, quindi non erano veri e propri errori avuti ma problemi o complicazioni che sarebbero potuti sorgere se lo impostavo nel modo errato.

Punto della situazione rispetto alla pianificazione

Sono in linea con il gantt.

Programma di massima per la prossima giornata di lavoro

Iniziare l'implementazione delle pagine web.

Diario di lavoro

Luogo	SAMT
Data	23.01.2019

Lavori svolti

Questa mattina la prima cosa che ho fatto è stata cercare dei template, inizialmente ho cercato quello per il login, essendo ciò che devo implementare oggi, e anche un eventuale template per lo shop.

Ho trovato entrambi, per il login ho deciso di utilizzare un template che permette di switchare nella stessa pagina login e registrazione.

Il template l'ho scaricato da questo sito: <https://codepen.io/Gibbu/pen/ZKYYZW>.

Dopo averlo preso ho controllato per il template dello shop, che poi riadatterò anche per le pagine dell'amministratore del gestore, ed ho optato per uno da shop trovato a questo link: <https://colorlib.com/wp/template/essence/>.

Inizialmente ho avuto alcuni problemi ad adattare il sito al mio template MVC, essendo che i link per i vari file css e JavaScript da caricare non venivano presi, e ho dovuto cambiare i percorsi inserendoli con la struttura MVC, come mostra l'immagine qui sotto.

```
<!-- Core Style CSS -->
<link rel="stylesheet" href="php echo URL ?&gt;application/views/shop/css/core-style.css"
&lt;link rel="stylesheet" href="<?php echo URL ?&gt;application/views/shop/style.css"&gt;</pre

```

Figura 1 percorsi per css

Quelli mostrati sopra sono 2 esempi di come ho dovuto cambiare i percorsi presenti nelle pagine che ho utilizzato, dovevo prendere la costante on la cartella base e arrivare fino alle cartelle da aprire.

Un ulteriore applicazione che ho attuato è stata creare dei file "header" e "footer" sia per la pagina principale che per la pagina del login e registrazione, questo perché così è tutto più ordinato e ogni pagina contiene la sua parte e se dovessi utilizzare lo stesso header o footer per più pagine diventa molto più comodo averli in file separati.

Per aprire le pagine con la struttura che utilizzo devo caricare le pagine con delle classi e delle funzioni, ognuna delle quali fa qualcosa.

```
2 class Home
3 {
4     /**
5      * Function to open the buyer's login page
6      */
7     public function index()
8     {
9         require_once 'application/models/connection.php';
10        $connection = new Connection(servername: "localhost", username: "root", password: "", dbName: "ripetizioni"
11        $connection->sqlConnection();
12
13        require 'application/views/_templates/header.php';
14        require 'application/views/shop/index.php';
15        require 'application/views/_templates/footer.php';
16    }
}
```

Figura 2 apertura della pagina principale

Nell'immagine è mostrata la funzione "index" della classe "Home", che si occuperà di tutti i lavori sulle pagine dello shop, la funzione in questione stabilisce la connessione, l'ho preparata

ora anche se mi servirà più avanti, e dopodiché apre tutte le pagine utili per aprire l'index dello shop, ovvero header, footer e pagina principale.

La stessa cosa è fatta anche per la pagina di login e registrazione, ma questa pagina ho dovuta dividerla di più, perché in un caso deve esserci anche la possibilità di registrarsi altrimenti no. Quindi ho creato 2 funzioni una che apre tutta la pagina completa e una che crea solo la parte del login, che mostra l'immagine che segue.

```
11 public function index()
12 {
13     require_once 'application/models/connection.php';
14     $connection = new Connection( 'servername: "localhost", username: "root", password: "", dbName: "ripetizioni" );
15     $connection->sqlConnection();
16
17     require 'application/views/login_register/static/header.php';
18     require 'application/views/login_register/switch.php';
19     require 'application/views/login_register/login.php';
20     require 'application/views/login_register/register.php';
21     require 'application/views/login_register/static/footer.php';
22 }
23
24 /**
25 * Function to open the login page
26 */
27 public function openLogin($msg = "")
28 {
29     require_once 'application/models/connection.php';
30     $connection = new Connection( 'servername: "localhost", username: "root", password: "", dbName: "ripetizioni" );
31     $connection->sqlConnection();
32     require 'application/views/login_register/static/header.php';
33     require 'application/views/login_register/login.php';
34     require 'application/views/login_register/static/footer.php';
35 }
```

Figura 3 apertura pagine login/registrazione

Per la pagina di login ho dovuto fare alcune modifiche al css, per renderla più adatta ai colori della pagina principale, quindi l'ho riadattata come mostrato dalle immagini che seguono.

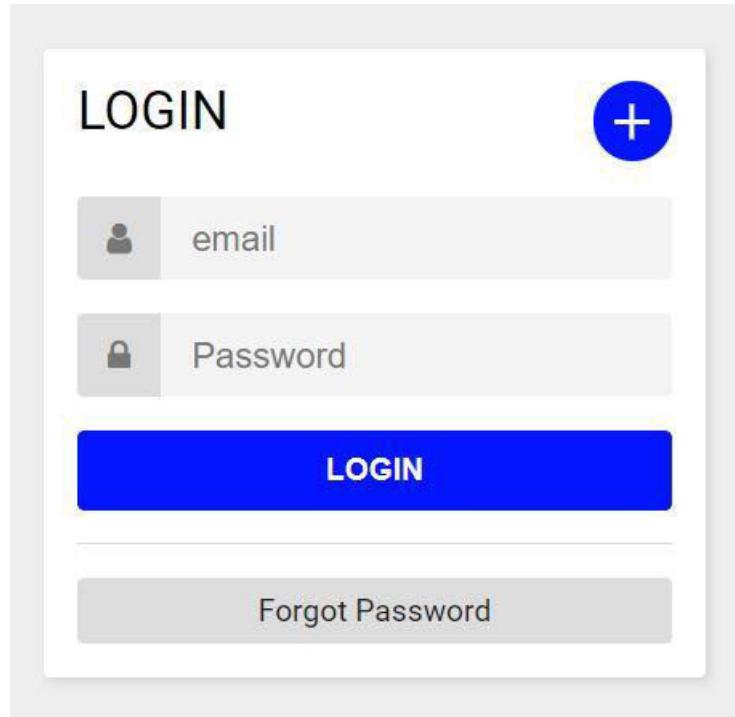


Figura 4 pagina di login

Questa è la pagina di login, ho sostituito la parte verde originale con il colore dei bottoni della pagina principale e quando si va sopra essi si trasforma nel rosso di esse.

The screenshot shows a registration form titled "REGISTER" in bold capital letters at the top center. In the top right corner is a blue circular button with a white "X". Below the title are five input fields, each with a small icon on the left: a person icon for "name", a person icon for "surname", an envelope icon for "Email", a phone receiver icon for "phone", and a lock icon for "Password". Each input field has a placeholder text to its right. At the bottom of the form is a large blue rectangular button with the word "REGISTER" in white capital letters.

Figura 5 pagina registrazione

Quella mostrata sopra invece è la pagina di registrazione in cui ho cambiato i colori anche in questo caso e in più ho modificato il form aggiungendo i campi che mi servono.

Purtroppo la modifica delle informazioni per adattare la pagina di login mi ha preso più tempo del previsto, ma comunque se si è dovevo farla da zero mi avrebbe preso forse ancora più tempo dato il fatto che avrei dovuto utilizzare da 0 bootstrap e css e non sono il massimo in questi 2 campi, di conseguenza non sono riuscito a completare i controlli dei vari campi e l'aggiunta del controllo Captcha, ma ho saputo che un mio compagno ha avuto dei problemi di permessi aggiungendolo, quindi attenderò maggiori informazioni da lui prima di inserirlo, per sapere se è effettivamente possibile o si ha bisogno di permessi speciali.

A causa di questo sono anche indietro con l'effettiva registrazione e il login dell'utente al sito, ma per fortuna ho già le funzioni quasi pronte devo solo completarle con i dati dei miei campi e quindi non mi ci vorrà molto a farlo.

Problemi riscontrati e soluzioni adottate

Non ho avuto dei veri e propri problemi ma mi ha impiegato del tempo capire come funzionavano i vari file css e javascript dei template per poterli utilizzare, e ho anche avuto dei leggeri problemi con bootstrap per sapere come funzionano alcune classi nella versione installata del template essendo diversa da quella usata precedentemente.

Punto della situazione rispetto alla pianificazione

Sono leggermente in ritardo rispetto al gantt essendo che non sono riuscito a finire la pagina di login e dovrò finire la prossima lezione.

Programma di massima per la prossima giornata di lavoro

Completare i controlli della pagina login e registrazione e implementare l'effettiva registrazione al database e il login alla pagina, sia quella del cliente che quella del gestore.

Diario di lavoro

Luogo	SAMT
Data	28.01.2019

Lavori svolti

Questa mattina ho portato avanti la pagina di login e registrazione, prima di tutto mi sono occupato di gestire il controllo della mail nella registrazione.

Con i controlli JavaScript ho controllato che ciò che viene scritto rispetti l'espressione regolare rispettiva, per ogni campo c'è un'espressione regolare apposita, e la funzione controlla come mostrato nell'immagine qui sotto.

```

36     function convalidate(value, id, regexp) {
37
38         //Variabile del campo
39         var id = document.getElementById(id);
40
41         //Se il campo è vuoto e non rispetta l'espressione regolare
42         if (!regexp.test(value) || value == "") {
43
44             //Colora il testo in rosso e segna l'errore.
45             id.style.color = "red";
46             checkInput = false;
47             //Altrimenti
48         } else {
49             console.log(id.name);
50             if(id.name == "emailR"){
51                 //Colora il testo di nero
52                 id.style.color = "black";
53
54                 //Controlla la mail
55                 checkMail(value);
56             }else{
57                 //Colora il testo di nero
58                 id.style.color = "black";
59                 checkInput = true;
60             }
61         }
62     }

```

Figura 1 controllo contenuto campi JavaScript

Una volta fatta la funzione sono passato a completare la parte di login, essendo che è la parte più importante attualmente per poter andare avanti con il gant.

Per lavorare con la struttura MVC ho creato 3 classi diverse, una per ogni tipo di utente, essendo che bisogna fare una classe per ogni tabella, e di conseguenza ho creato una per ogni tipo di utente. Oltre a questo ho anche dovuto creare 3 funzioni diverse per il login, ed ognuna fa riferimento ad un model diverso.

Quella mostrata qui sotto è la funzione che va a richiedere alla sua classe un controllo del login e riceve un campo

```
40     public function loginD()
41     {
42         if ($_SERVER["REQUEST_METHOD"] == "POST") {
43
44             //Prendo la classe model
45             require_once 'application/models/dealer.php';
46             $dealer = new Dealer();
47
48             //Prendo le variabili passate dal post
49             $email = $_POST["email"];
50             $pass = $_POST["pass"];
51             $pass = hash( algo: 'sha512', $pass);
52
53             //Se entrambi i campi non sono vuoti
54             if (!strcmp($email, str2: "") && !strcmp($pass, str2: "")) {
55
56                 //controllo il login
57                 $var = ($dealer->checkLogin($email, $pass, table: "gestore"));
```

Oltre a questo non sono riuscito a fare perché mi sono un po' perso con alcuni controlli della registrazione e di conseguenza non sono riuscito a completare il login.

Problemi riscontrati e soluzioni adottate

Ho avuto alcuni rallentamenti dati dalle espressioni regolari e i controlli per la registrazione, ho dovuto riguardare le espressioni regolari nel sito <https://regexpr.com/> per rinfrescarmi la memoria, e alcuni problemi fatto a questo mi hanno fatto perdere del tempo.

Punto della situazione rispetto alla pianificazione

Sono in ritardo rispetto al gantt essendo che non sono riuscito ancora a finire il login e avrei dovuto finirlo la scorsa lezione, ma dovrei comunque riuscire a recuperare, quindi il ritardo non è un grosso problema.

Programma di massima per la prossima giornata di lavoro

Completare la pagina di login e registrazione iniziare la pagina di vendita dei prodotti.

Diario di lavoro

Luogo	SAMT
Data	29.01.2019

Lavori svolti

Questa mattina ho completato il login, ho finto la funzione che ritorna i valori in base ai dati inseriti, ovvero se i dati dell'utente sono validi o meno, come spiegato sotto.

```

34 if($sql->rowCount() > 0) {
35     // Ciclo tutti i valori
36     while ($row = $sql->fetch()) {
37         //Controllo che la password sia corretta
38         if (!strcmp($pass, $row["password"])) {
39             // LOGIN VENDITORE
40             $_SESSION['customer'] = $mail;
41             $checkSession = "customer";
42         } else {
43             // PASSWORD ERRATA
44             $_SESSION['wrongPass'] = "wrongPass";
45             $checkSession = "wrongPass";
46         }
47     }
48     //Altrimenti
49 } else{
50     // UTENTE INESISTENTE
51     $_SESSION['noUser'] = "noUser";
52     $checkSession = "noUser";
53 }
54
55 //Chiudo la sessione
56 $conn = null;
57 //Ritorno il valore
58 switch ($checkSession) {
59     case "customer":
60         return $_SESSION['customer'];
61     case "wrongPass":
62         return $_SESSION['wrongPass'];
63     case "noUser":
64         return $_SESSION['noUser'];
65     default:
66         return "ERRORE";
67 }

```

Figura 1 model login

La parte iniziale dell'immagine viene subito dopo aver eseguito la query con il comando “\$sql->execute()” e si occupa di controllare che la query abbia prodotto dei risultati, in caso contrario crea la variabile di sessione in cui viene mostrato che non ci sono utenti. Se invece la query da dei risultati con la mail passata, li cicla, ovvero prende l'unico disponibile, e fa un controllo con la

password passata, a dipendenza del risultato crea una variabile di sessione adatta. Se la password è corretta crea la variabile con la mail altrimenti con un “messaggio di errore”.

Una volta fatto chiudo la connessione, mettendo la variabile di essa a null, poi con uno switch controllo e a dipendenza del valore della variabile di controllo passo quella di sessione adeguata.

Una volta fatto questo nella funzione controller vengono svolti ulteriori controlli per poi riaprire la pagina corretta.

```

40     public function loginD()
41     {
42         if ($_SERVER["REQUEST_METHOD"] == "POST") {
43
44             //Prendo la classe model
45             require_once 'application/models/dealer.php';
46             $dealer = new Dealer();
47
48             //Prendo le variabili passate dal post.
49             $email = isset($_POST["email"]) ? $_POST["email"] : null;
50             $pass = isset($_POST["pass"]) ? $_POST["pass"] : null;
51             //$pass = hash('sha512', $pass);
52
53             //Se entrambi i campi non sono vuoti
54             if ($email != null && $pass != null) {
55
56                 //controllo il login
57                 $var = ($dealer->checkLogin($email, $pass));
58
59                 //Se viene ritornata la mail apre la pagina dell'utente
60                 if(!strcmp($var, $email)){
61                     header( string: "location: ". URL );
62                     //Altrimenti torna alla pagina di login
63                 }else{
64                     header( string: "location: ". URL ."login/index");
65                 }
66             }
67         }
68     }

```

Figura 2 controller login

La funzione dopo aver preso i dati e averli inviati al model, immagazzina il ritorno in una variabile, che poi viene controllata, se è la mail allora apre la pagina dell’utente, che momentaneamente non è completa per cui porta alla pagina principale, ma se non è come la mail ritorna alla pagina di login in cui verrà mostrato il problema, questa stessa funzione è stata copiata per 3 volte, una per ogni tipo di utente, ed è praticamente uguale, l’unica differenza è che fanno riferimento a dei model diversi e aprono pagine diverse.

Una volta fatto il login se dovesse esserci un errore esso viene segnalato con un popup che mostra un messaggio di errore.

Questo sistema l’ho creato con un template preso online a questo link:
<https://codyhouse.co/gem/simple-confirmation-popup> in questo link si può sia vedere la demo che andare alla pagina i github in cui sono presenti tutti i file utili per il popup.

Qui sotto è mostrato come appare a schermo il popup una volta mostrato, il bottone non è centrato al 100% ma è un problema su cui ho deciso di non perdere troppo tempo essendo che il resto funziona perfettamente e sono già indietro.

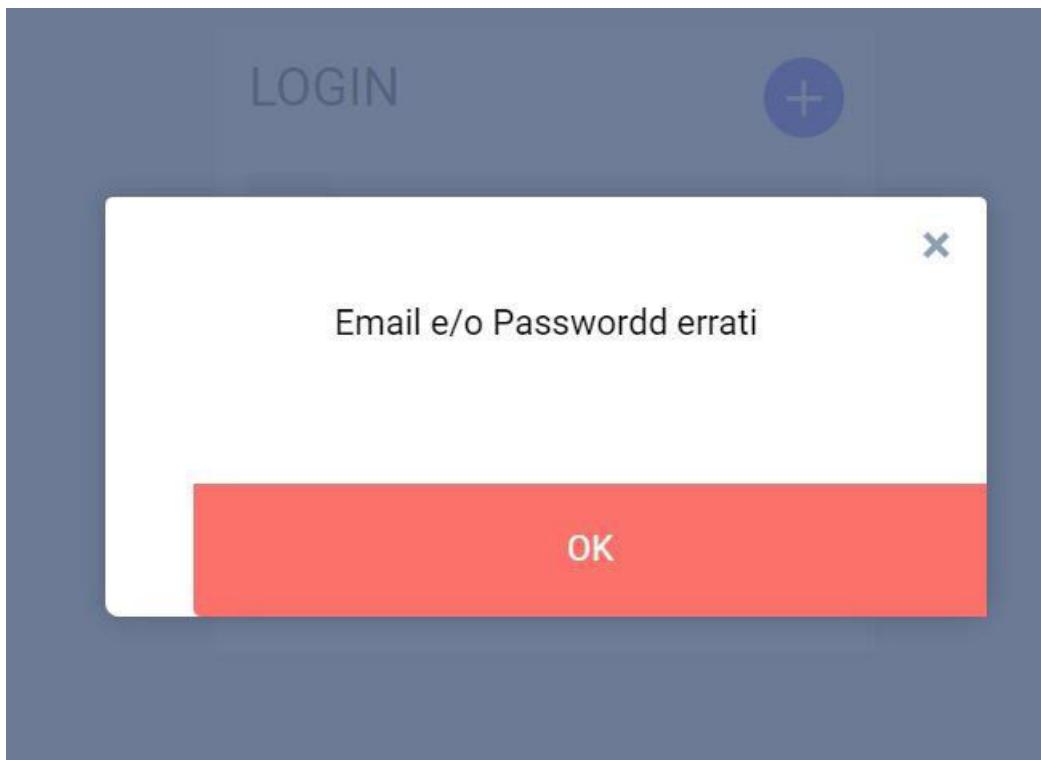


Figura 3 popup login errato

Inizialmente per funzionare viene parte il codice PHP mostrato nell'immagine sottostante.

```
6      <?php
7      if(isset($_SESSION['wrongPass'])){
8          print "<script type='text/javascript'>
9              linkClick();
10             </script>";
11
12         unset($_SESSION['wrongPass']);
13     } else if(isset($_SESSION['noUser'])){
14         print "<script type='text/javascript'>
15             linkClick();
16             </script>";
17
18         unset($_SESSION['noUser']);
19     }
20
21 ?>
```

Figura 4 PHP attivazione popup

Il codice controlla inizialmente se una delle due variabili di sessione esiste, in caso di risposta positiva stampa uno script che fa partire una funzione nel file JavaScript apposito per il popup, la funzione è quella mostrata nell'immagine che segue.

Una volta finito il codice viene disattivata la variabile di sessione in modo che se si aggiorna la pagina non viene mostrato nuovamente, e inutilmente, il popup.

Come ultima cosa ho ultimato la registrazione inserendo l'invio della mail ma ancora riscontra alcuni problemi per cui lo completerò definitivamente domani.

```

1  $(document).ready(function($) {
2
3      //close popup
4      $('#cd-popup').on('click', function(event) {
5          if( $(event.target).is('.cd-popup-close') || $(event.target).is('.cd-popup') || $(event.target).is('.ok-close') ) {
6              event.preventDefault();
7              $(this).removeClass( 'is-visible' );
8          }
9      });
10     //close popup when clicking the esc keyboard button
11     $(document).keyup( e: function(event) {
12         if(event.which=='27') {
13             $('#cd-popup').removeClass( 'is-visible' );
14         }
15     });
16 });
17
18 function linkClick(){
19     $('#cd-popup').addClass( 'is-visible' );
20 }
```

Figura 5 JQuery popup

La funzione che viene richiamata è quella in fondo, che va a rendere visibile il div per il popup e una volta fatto si vede il popup come mostrato nella foto di sopra, sopra a quella funzione ci sono altre due funzioni che partono a determinati eventi, la prima viene attivata in caso vengano premuti la "x" in alto a destra o il bottone ok e la seconda quando viene cliccato al di fuori del popup, ed entrambi hanno lo stesso compito, chiudono, o rendono invisibile, la schermata mostrando di nuovo il login.

Problemi riscontrati e soluzioni adottate

Inizialmente ho avuto dei problemi nella query, perché prima provavo a fare tutti e tre i login nella stessa funzione quindi dovevo decidere la tabella da inserire nella query **"SELECT * FROM :table WHERE email LIKE :mail"** e doveva essere diversa per ogni tipo di utente, ma il metodo mostrato nella stringa, ovvero mettere anche la tabella come variabile, non è utilizzabile con PDO, che è la classe che ho deciso di utilizzare per la comunicazione con il database.

Per ovviare a questo problema ho optato per creare il metodo delle 3 funzioni spiegato nella sezione precedente, in questo modo ogni funzione ha una query che fa riferimento alla tabella dell'utente proprio.

Un altro problema che ho avuto è stato con l'inserimento del popup. Essendo che l'ho preso da un template, ho dovuto estrapolare solo ciò che mi serviva, e quindi ho dovuto anche riadattare tutti i file come è stato per la pagina di login, quindi questo mi ha preso un po' di tempo.

Un altro problema è stato l'adattamento del codice JavaScript, essendo che il template originale faceva apparire la schermata al click di un bottone il tutto era gestito all'evento "onclick" del link, ma essendo che non avevo un link ma doveva apparire quando si tornava alla pagina ho dovuto riadattarlo.

Inizialmente ho provato a creare un link nascosto che tramite php se viene settata la variabile di sessione viene cliccata, da codice con JavaScript con il codice

"document.getElementById('id').click()" i modo da scatenare l'evento di JQuery, ma questo non funzionava essendo che il click da programma non attiva JQuery, cosa di cui mi sono accorto a mie spese, e quindi ho dovuto gestirlo in un altro modo, e ho optato per il metodo mostrato nella sezione precedente.

Punto della situazione rispetto alla pianificazione

Sono in sempre un po' in ritardo, ho finito oggi il login.

Programma di massima per la prossima giornata di lavoro

Portare avanti la pagina di messa in vendita dei prodotti.

Diario di lavoro

Luogo	SAMT
Data	30.01.2019

Lavori svolti

A prima cosa che ho fatto oggi è stata completare la parte di registrazione, ma ancora non ho implementato i controlli dei vari campi.

Quello che ho fatto è stato implementare delle funzioni che vanno ad inserire i dati passati dall'utente nel database, per fare questo però ho dovuto dividere il tutto in 3 funzioni perché alcune parti vanno divise per i 3 utenti ma altre sono uguali.

```

41     public function post(){
42         //Controllo ci sia la chiamata in POST
43         if ($_SERVER["REQUEST_METHOD"] == "POST") {
44             echo "LOG";
45             //Prendo le variabili
46             $name = $_POST["name"];
47             $surname = $_POST["surname"];
48             $mail = $_POST["emailR"];
49             $phone = $_POST["phone"];
50             $pass = $_POST["password"];
51             $pass = hash( algo: 'sha512', $pass);
52
53             //Tipo dell'utente del presente
54             $type = $_POST["type"];
55
56             switch ($type){
57                 case "customer":
58                     //Richiamo la funzione di inserimento dell'utente
59                     $this->createC($name, $surname, $mail, $phone, $pass);
60                 case "dealer":
61                     //Richiamo la funzione di inserimento dell'utente
62                     $this->createD($name, $surname, $mail, $phone, $pass);
63                 case "admin":
64                     //Richiamo la funzione di inserimento dell'utente
65                     $this->createA($name, $surname, $mail, $phone, $pass);
66             }
67         }else{
68             header( string: "location: javascript://history.back() ");
69         }
70     }

```

Figura 1 Metodo post

Quella sopra mostrata è la prima e la più lunga funzione, è quella che viene richiamata all'action del form di registrazione, e si occupa di prendere tutti i dati passati, tra cui anche il tipo di utente, e in base a quest'ultimo richiamare la funzione corretta, grazie allo switch finale. Come si può notare c'è un primo controllo (righe 43; 67-69) che si occupa di controllare che la funzione venga richiamata con il POST, in caso contrario ritorna alla pagina precedente, per evitare che qualcuno

entri in quella funzione manualmente tramite link, questo controllo è fatto su tutte le funzioni a cui serve.

```

11     public function createC($name, $surname, $mail, $phone, $pass)
12     {
13         require_once 'application/models/customer.php';
14         $customer = new Customer();
15
16         if (strcmp($name, "")) || strcmp($surname, "") || strcmp(
17             $customer->addUser($name, $surname, $mail, $phone, $pass));
18     }
19
20         $this->sendMail($name, $surname, $mail);
21
22         header( string: "location: ". URL ."login/loginPageC");
23     }

```

Figura 2 metodo creazione utenti

Questo metodo è quello richiamato dalla funzione vista precedentemente, e si occupa di prendere la variabile per lavorare con i clienti ed aggiungere l'utente con i dati e nella funzione apposita. Prima di questo c'è un controllo per vedere che nessun campo richiesto sia vuoto, non è tutto presente nell'immagine a causa della lunghezza. Alla fine della funzione prima di aprire la pagina finale viene richiamata una funzione che invia una mail all'utente appena registrato che è mostrato qui sotto, e si occupa semplicemente di creare un messaggio e inviarlo grazie al metodo apposito, la mail che invia il messaggio è quella del vecchio progetto perché sarebbe stato insensato e troppo lungo creare una nuova.

```

30     public function sendMail($name, $surname, $mail){
31
32         //Messaggio da inviare
33         $msn = "'$name $surname' Sei stato correttamente registrato";
34         $msn = $msn. "\nAccedi da qui http://localhost:8042/MVC/login/index";
35
36         //Invio il messaggio
37         $headers = "From: progettotoripetizioni@gmail.com";
38         $a = mail($mail, subject: "NON RISPONDERE A QUESTA MAIL", $msn, $headers);
39     }

```

Figura 3 metodo di invio mail

Le 3 funzioni di creazione dell'utente sono pressoché uguali, cambia solo la classe a cui fanno riferimento.

Una volta finito questo ho guardato tra le pagine del template e deciso quali sono le migliori per fare le pagine dei gestori ed ho optato per quella di "shop" e quella di "dettaglio dei prodotti" per i nuovi prodotti da inserire, e sono le 2 pagine mostrate, in parte, qui sotto.



Figura 4 pagina template di shop

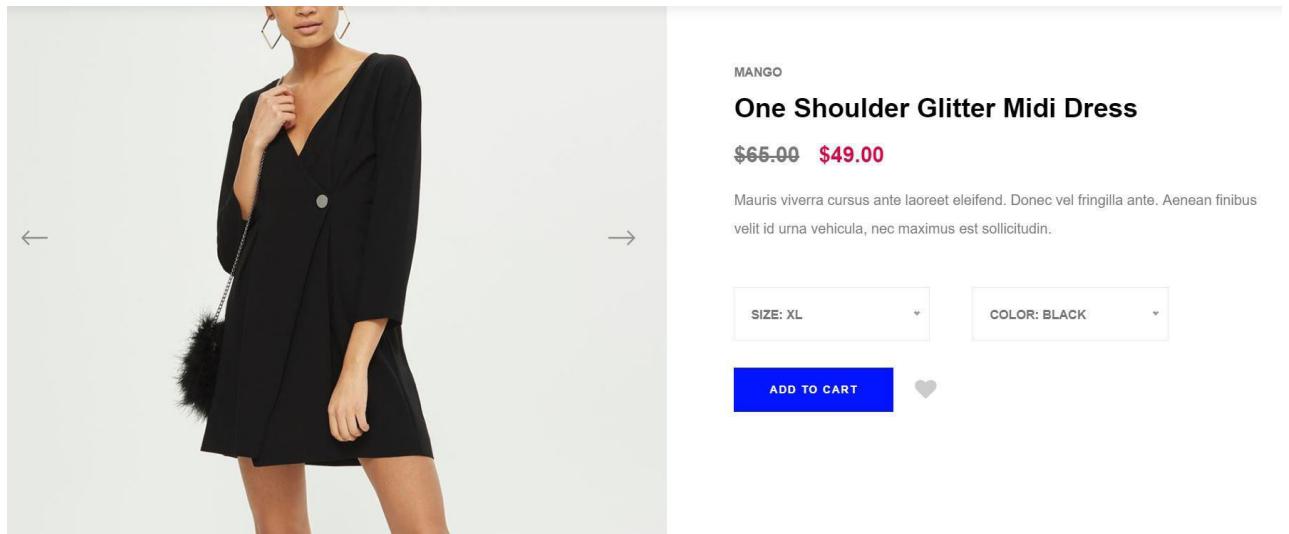


Figura 5 pagina template dettagli prodotto

Dopo aver isolato queste 2 pagine ho anche implementato la parte di logout, mi è bastato prendere uno dei 3 bottoni in alto, inserire l'immagine di logout e linkarla alla funzione adatta. L'immagine qui sotto è quella utilizzata per il logout.

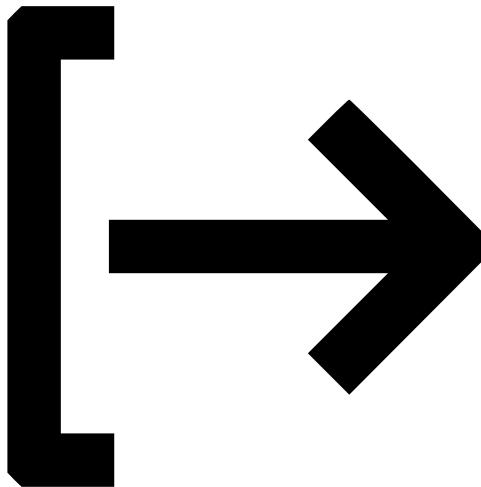


Figura 6 immagine logout

Questo bottone è stato impostato per attivare la funzione di logout quando premuto, che semplicemente distrugge la sessione e torna alla pagina iniziale, come mostrato nell'immagine sottostante.

```
5      class Logout
6      {
7          public function dealer()
8          {
9              session_destroy();
10             header( string: "location: http://localhost:8042/MVC/dealer");
11         }
12     }
```

Figura 7 funzione di logout

Come si può vedere anche per questa classe ho dovuto creare 3 diverse funzioni perché ogni logout ritorna ad una pagina diversa, quella del cliente alla pagina del negozio, quella del gestore al login suo e quella dell'admin al suo.

Problemi riscontrati e soluzioni adottate

Ho avuto un problemino con la registrazione, perché la funzione che doveva alla fine tornare alla pagina di login faceva riferimento alla funzione sbagliata, e quindi non tornava indietro ma rimaneva bloccato lì, mi è bastato inserire il link giusto all'interno della funzione “header”.

Punto della situazione rispetto alla pianificazione

Oggi ho iniziato la preparazione della pagina della messa in vendita degli oggetti.

Programma di massima per la prossima giornata di lavoro

Portare avanti la pagina di messa in vendita dei prodotti.

Diario di lavoro

Luogo	SAMT
Data	04.02.2019

Lavori svolti

Questa mattina essendo che avevo solo queste 2 ore ho deciso di utilizzarle per portare a termine i controlli dei campi nel form della registrazione, in modo che se si scrivono campi errati viene segnalato colorando in rosso il testo errato come mostrato nell'immagine qui sotto. Le prime 2 immagini mostrano il controllo applicato ai campi ce possono contenere solo lettere, come ad esempio nome e cognome.

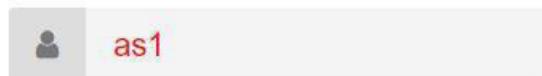


Figura 1 campo solo lettere errato

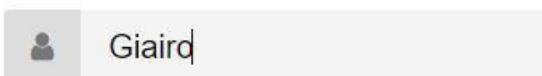


Figura 2 campo solo lettere corretto

Le foto che seguono mostrano il campo del numero di telefono, per questo vengono accettati i numeri gli spazi e il più solo all'inizio, se si dovesse mettere un più in un punto diverso viene segnalato l'errore come qualsiasi altro caso sbagliato.

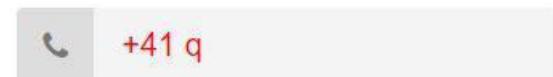


Figura 3 campo numero di telefono errato



Figura 4 campo numero di telefono corretto

I campi mostrati qui sotto mostrano il controllo della mail in cui bisogna inserire il seguente tipo di stringa nell'input: "Stringa + @ + stringa + . + stringa", all'interno di "stringa" può andarci qualsiasi cosa.



Figura 5 campo email errato



Figura 6 campo email corretto

L'ultimo campo da controllare è quello della password, per questo è un po' più complicato perché l'errore non viene segnalato colorando il testo, di conseguenza ho sviluppato un altro metodo, scrivere l'errore sotto ai campi, essendo che ci sono 2 campi per la password, una normale e una di verifica. I requisiti per la password sono sia lettere che numeri e da 8 a 25 caratteri e quelle mostrate qui sotto sono i messaggi di errore possibili che vengono mostrati.

A screenshot of a web form showing two password input fields. The first field has a lock icon and contains a password with dots. The second field also has a lock icon and contains the text 'Conferma password'. Below the fields, a red error message reads 'Password non valida (8-25 caratteri, sia lettere che numeri)'.

Figura 7 password non valida

ERRORE: password diverse

Figura 8 password non coincidenti

A screenshot of a web form showing two password input fields. Both fields have lock icons and contain different passwords represented by dots. Below the fields is a large blue 'REGISTER' button.

Figura 9 password corrette

Come si può anche notare dall'ultima immagine se i dati sono tutti corretti viene attivato il bottone che permette la registrazione, se invece ci sono segnalati dei messaggi di errore esso rimane disattivo. C'è la possibilità che il bottone sia attivo con dei campi vuoti, ma in quel caso se viene cliccato darà errore al campo vuoto perché sono tutti "required".

Tutti i controlli sono fatti con JavaScript grazie a delle funzioni. Nella funzione di convalida generale mostrata nel diario del 28.01.2019 si nota che se passa il controllo della mail si apre una nuova funzione, questa si occupa di fare una richiesta con Ajax ad una funzione che ritorna tutti i dati degli utenti nel database e compara la mail a quella scritta per controllare che non esista già, e se dovesse esistere segnala un errore alla fine del form.

```

if(id.name == "emailR"){
    //Colora il testo di nero
    id.style.color = "black";

    //Controlla la mail
    checkMail(value);
}

```

Figura 10 convalida email

```

66     function checkMail(mail){
67         var mailExists = document.getElementById('mailExists');
68         var type = document.getElementById('type').value;
69
70         xhttp.onreadystatechange = function() {
71             if (this.readyState === 4 && this.status === 200) {
72
73                 //Prendo tutti i valori passati dalla pagina
74                 var results = JSON.parse(xhttp.responseText);
75
76                 //Controllo che l'email inserita non esista già nel database
77                 for(var i = 0; i < results.length; i++){
78
79                     //Se la mail esiste già cambio la variabile di controllo e smetto di controllare segnalando il messaggio d'errore
80                     if(results[i]['email'] == mail){
81                         mailExists.hidden = false;
82                         checkMailE = false;
83                         break;
84                     //Se la mail non esiste nasconde il messaggio d'errore e cambio la variabile di controllo.
85                     }else{
86                         checkMailE = true;
87                         mailExists.hidden = true;
88                     }
89                 }
90
91                 //Se la mail non esiste va avanti
92                 if(checkMailE){
93                     confirm();
94                     //se la mail esiste
95                 }else{
96
97                 }
98             }
99         }
100        xhttp.open( method: "POST", url: "/MVC/registration/checkMail/", async: true);
101        xhttp.setRequestHeader( name: 'Content-type', value: 'application/x-www-form-urlencoded');
102        xhttp.send( body: "type="+ type);
103    }
}

```

Figura 11 funzione controllo unicità mail

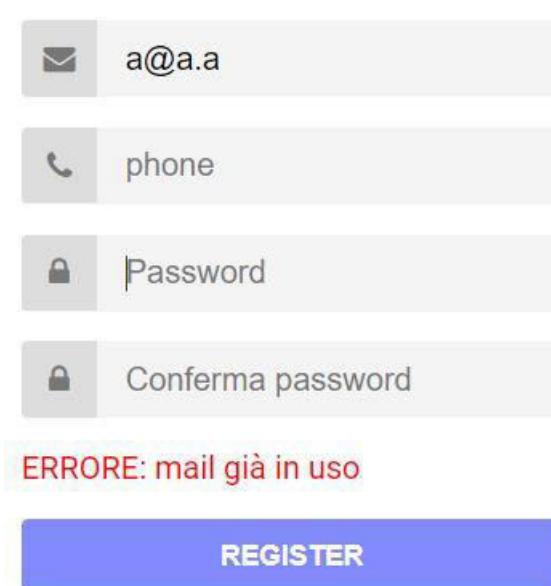


Figura 12 email già esistente

Le funzioni vengono prese da model e controller che si occupano solo di questo.

```
75     public function checkMail(){
76         if($_SERVER['REQUEST_METHOD'] == 'POST'){
77
78             $user = "";
79
80             switch ($_POST['type']){
81                 case "customer":
82                     require_once 'application/models/customer.php';
83                     $user = new Customer();
84                     break;
85                 case "dealer":
86                     require_once 'application/models/customer.php';
87                     $user = new Customer();
88                     break;
89                 case "admin":
90                     require_once 'application/models/customer.php';
91                     $user = new Customer();
92                     break;
93             }
94
95             //Stampo con json tutti gli utenti
96             header( string: 'Content-Type: application/json' );
97             echo json_encode($user->getUsers());
98         }else{
99             header( string: "location: javascript://history.back()" );
100        }
101    }
```

Figura 13 controller checkMail

La funzione sopra mostrata è quella richiamata da Ajax e si occupa solo di creare un oggetto della classe richiesta in base alla registrazione che si vuole fare e poi richiamare la funzione nella determinata classe.

```
140     public function getUsers() {
141         //Connetto al database
142         $conn = $this->connection->sqlConnection();
143
144         //Prendo i dati dell'utente in base alla mail
145         $sql = $conn->prepare( statement: "SELECT * FROM cliente");
146         $sql->execute();
147
148         $dataArray = array();
149
150         //Se ci sono dei valori
151         if($sql->rowCount() > 0){
152
153             // while che prende tutti i dati
154             while($row = $sql->fetch()) {
155                 array_push( &array: $dataArray, $row);
156             }
157         } else {
158             $conn = null;
159             return false;
160         }
161
162         $conn = null;
163         return $dataArray;
164     }
```

Figura 14 model getUsers

La funzione “getUsers” sopra mostrata è quella che si occupa di prendere tutti i dati degli utenti di una determinata tabella, per ogni tipo di utente i dati vengono presi dalla propria funzione, e quindi la funzione è pressoché la stessa cambia solo quel dettaglio.

Problemi riscontrati e soluzioni adottate

Ho avuto un problema all'inizio con il css, perché, come spiegate precedentemente nei diari, ho spostato le cartelle di css e JavaScript delle pagine principali in una cartella unica in modo che tutti facciano riferimento a quello, ma a quanto pare non si era salvato il cambiamento di “src” nel tag link e quindi non riusciva a caricare lo stile delle pagine.

Oltre a questo non ho avuto grandi problemi se non alcune modifiche fatte durante la scrittura dei codici per i controlli della registrazione, ma seguente la sezione precedente tutto risulta funzionante.

Punto della situazione rispetto alla pianificazione

Oggi ho iniziato semplicemente completato la registrazione.

Programma di massima per la prossima giornata di lavoro

Portare avanti la pagina di messa in vendita dei prodotti.

Diario di lavoro

Luogo	SAMT
Data	05.02.2019

Lavori svolti

La prima cosa che ho fatto questa mattina è stata correggere degli errori trovati ho segnalati dal docente. Il primo stava nel database, nella tabella dei prodotti mancava l'immagine, allora ho inserito un campo che andrebbe a essere riempito con il percorso in cui trovare l'immagine per quel determinato prodotto, al posto di inserire le immagini direttamente nel database e appesantirlo.

Quello qui sotto è il nuovo schema E-R modificato.

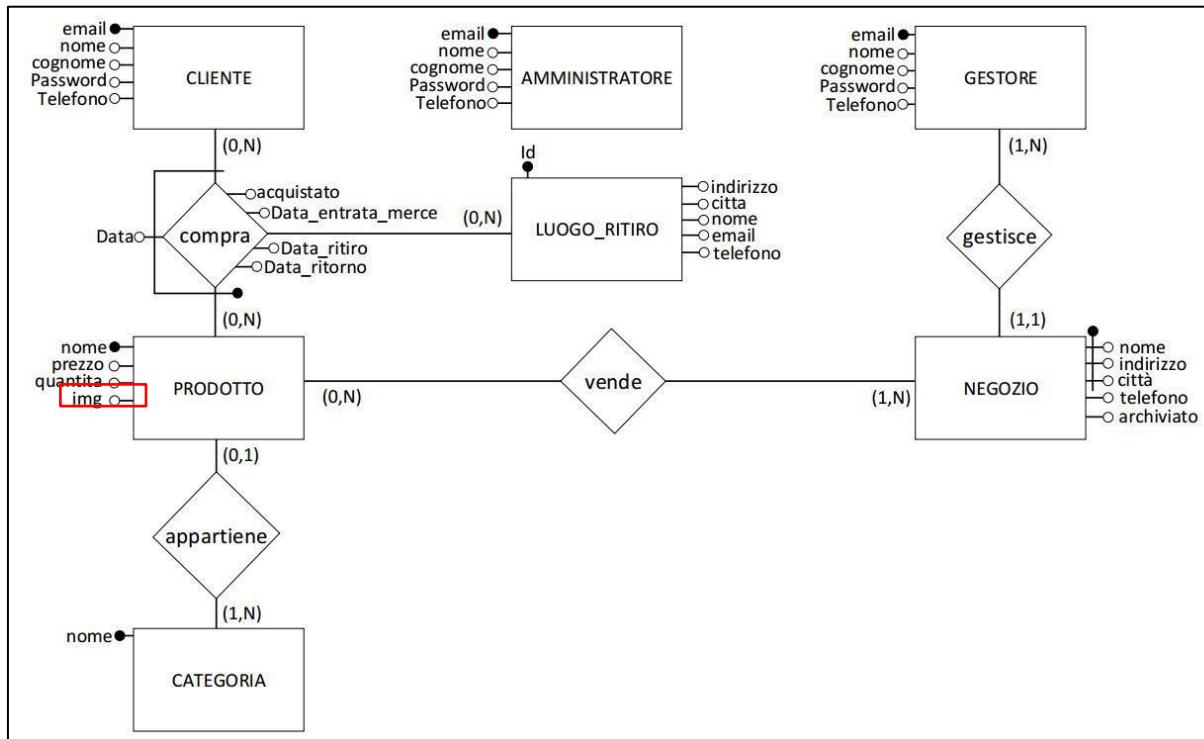


Figura 1 schema E-R con immagine

L'altro errore da mettere a posto mi è stato segnalato dal mio docente, all'interno della funzione della registrazione c'era un'incongruenza fuori dall'if.

```

11     public function createC($name, $surname, $mail, $phone, $pass)
12     {
13         require_once 'application/models/customer.php';
14         $customer = new Customer();
15
16         if (strcmp($name, "") == 0 || strcmp($surname, "") == 0 || strcmp($mail, "") == 0 || strcmp($phone, "") == 0 || strcmp($pass, "") == 0)
17             $customer->addUser($name, $surname, $mail, $phone, $pass);
18
19
20         $this->sendMail($name, $surname, $mail);
21
22         header('location: ' . URL . "login/loginPageC");
23     }

```

Figura 2 vecchia funzione per la creazione utente

La funzione inizialmente faceva il controllo dei campi e si assicurava che avessero un valore e se non l'avevano non creava l'utente, ma comunque sia inviava la mail, e questo è errato dal momento che non viene creato l'utente ma viene inviata l'email che è stato creato, quindi la semplice correzione che ho applicato è stata inserire il richiamo del metodo "sendMail" all'interno del controllo de campi.

Una volta fatti questi cambiamenti ho esposto alcuni dubbi che mi erano venuti al docente sull'amministratore ee i suoi compiti riguardo alla risposta **"le tabelle dei parametri / configurazioni (tempi di consegna, luoghi di consegna, prodotti da mostrare pagina web, percentuali che il sito trattiene dal prezzo per il servizio"** che mi aveva dato e abbiamo risolto che i tempi di consegna e i luoghi vengono anche gestiti da lui, ma principalmente è il gestore a segnarlo, come è giusto che sia, e possono essere modificati, e l'amministratore può vedere se vengono cambiati troppe volte e può andare a discuterne con il negoziante. I prodotti che vengono mostrati nelle pagine invece vengono decisi dall'utente invece che dall'amministratore, mentre le percentuali che il sito trattiene dall'amministratore in una pagina a parte. Dato ciò ho dovuto fare un ulteriore modifica al database, per introdurre la percentuale che il sito trattiene, quindi l'E-R per ora definitivo è quello mostrato sotto.

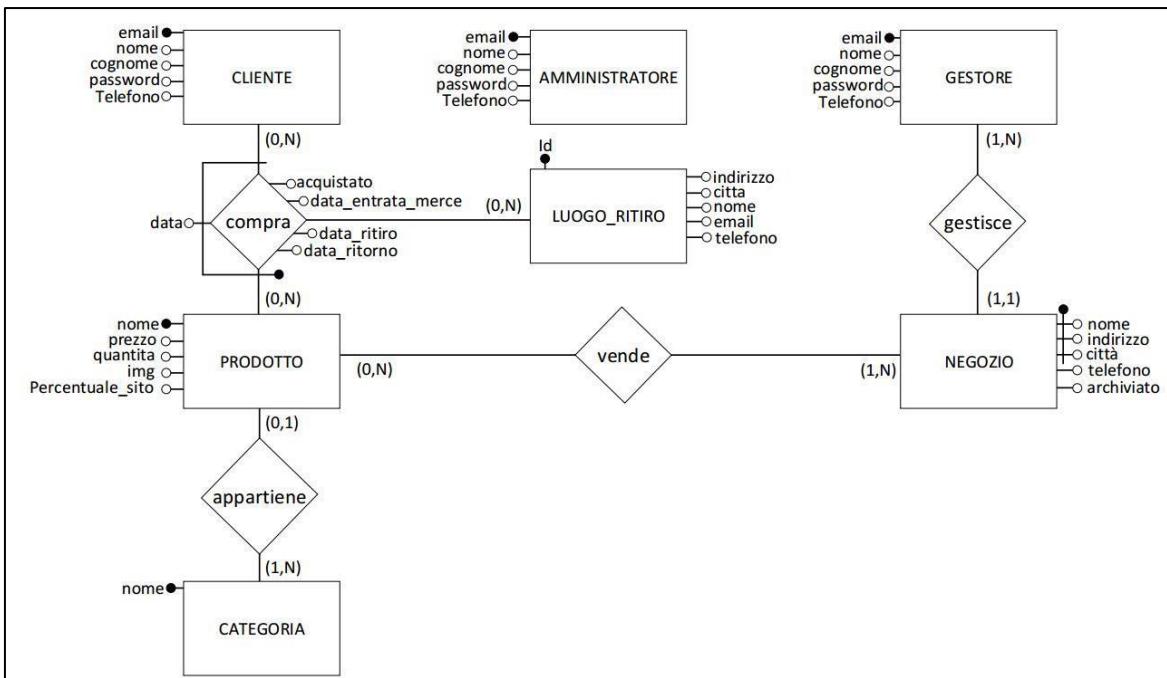


Figura 3 schema E-R con percentuale sito

Una volta completato ho messo il sito online, per farlo ho utilizzato il software “total commander” che è poi lo stesso software che uso per lavorare sui file online. Semplicemente ho fatto l’accesso all’ftp con le credenziali datemi dal professor Barchi e ho spostato al suo interno tutte le cartelle e i file della struttura, ovvero “application” con il contenuto e i file “.htacces” e “index.php”. Una volta fatto ho dovuto riadattare le informazioni, prima di tutto ho dovuto cambiare il nome della cartella del progetto nel file “.htaccess” come mostrato nell’immagine che segue.

```
17 | RewriteBase /gestionevendita2018/
```

Figura 4 cartella progetto online “.htaccess”

Dopo aver cambiato questo ci sono altri 2 cambiamenti da fare, il primo appartiene al database, essendo che faccio riferimento ad un database non più locale ho dovuto cambiare le variabili di base con i criteri di accesso al database con quelli datimi dal professore.

L’ultima cosa, ma non meno importante, da modificare è la variabile globale “URL” che si trova nel file “config.php” e bisogna dargli il valore del sito base a cui accedere, nel mio caso “<http://samtinfo.ch/gestionevendita2018/>”, come mostrato nello screenshot che segue.

```
20 | define('URL', 'http://samtinfo.ch/gestionevendita2018/');
```

Figura 5 variabile URL config.php

Dopo aver fatto questo ho iniziato a lavorare alla pagina di mostra dei prodotti per i venditori. La prima cosa che ho fatto è stata modificare la lista delle categorie a sinistra, facendogli inserire automaticamente le categorie presenti nel database, per fare questo ho utilizzato Ajax, in una funzione faccio la richiesta e poi con la risposta che mi arriva compongo la lista delle categorie, la funzione è quella mostrata qui sotto.

```

8     function getCategories() {
9
10    xhttp.onreadystatechange = function () {
11      if (this.readyState === 4 && this.status === 200) {
12
13        //Prendo i valori passati dal server e li metto in un array
14        var obj = JSON.parse(xhttp.responseText);
15
16        //Prendo il corpo della tabella in cui inserire le righe
17        var tbody = document.getElementById( elementid: 'menu-content2' );
18
19        //Inserisco tutti le righe con i relativi corsi.
20        for (var i = 0; i < obj.length; i++) {
21          //Creo la riga i cui inserire i campi e gli metto le informazioni
22          var li = document.createElement( tagName: "li");
23          li.setAttribute( qualifiedName: "data-toggle", value: "collapse");
24          //li.setAttribute("class", "collapsed");
25
26          //Creo una nuova colonna e inserisco il dato, per ogni informazioni richiesta
27          var a = document.createElement( tagName: "a");
28          a.appendChild(document.createTextNode(obj[i]["nome"]));
29          a.setAttribute( qualifiedName: "href", value: "#");
30          li.appendChild(a);
31          console.log(tbody);
32          //Inserisco la riga nella tabella
33          tbody.appendChild(li);
34        }
35      }
36    }
37    xhttp.open( method: "POST", url: "/gestionevendita2018/home/getCategories", async: true);
38    xhttp.send();
39  }

```

Figura 6 funzione JavaScript inserimento categorie

Questa funzione va a fare la richiesta al controller Home, che contiene le funzioni utilizzabili da tutti, e prende quella per le categorie.

Questa funzione va a richiamare dalla classe “Category”, che è quella che si occupa di lavorare nel database con la tabella delle categorie, e prende la funzione che le ritorna tutte.

```

71     public function getCategories(){
72
73       if ($_SERVER["REQUEST_METHOD"] == "POST") {
74
75         //Prendo la classe model
76         require_once 'application/models/category.php';
77         $category = new Category();
78
79         $ccategories = $category->getCategories();
80
81         //Stampo con json i valori dei coach
82         header( string: 'Content-Type: application/json');
83         echo json_encode($ccategories);
84       }else{
85         header( string: "location: javascript://history.back() ");
86       }
87     }

```

Figura 7 Controller getCategories

La funzione model invece va a prendere tutti i dati contenuti nella tabella, ovvero i nomi di tutte le categorie essendo quello l’unico campo esistente, e le ritorna in un array.

```
17     public function getCategories(){
18         //Connetto al database
19         $conn = $this->connection->sqlConnection();
20
21         //Prendo i dati dell'utente in base alla mail
22         $sql = $conn->prepare( statement: "SELECT * FROM categoria");
23
24         $dataArray = array();
25         //Eseguo la query
26         $sql->execute();
27
28         // Ciclo tutti i valori
29         while ($row = $sql->fetch()) {
30             array_push( &array: $dataArray, $row);
31         }
32         $conn = null;
33         return $dataArray;
34     }
```

Figura 8 model getCategories

Alla fine le categorie vengono inserite a destra di tutto e sono cliccabili, ma ancora non sono state implementate per cui non modificano la lista dei prodotti.



Figura 9 visualizzazione categorie

Dopo questo ho fatto la visualizzazione dei prodotti, più o meno il sistema è lo stesso, la differenza sostanziali, oltre alla tabella e di conseguenza la tabella, è la visualizzazione, perché è una struttura di “Div”, ben definiti, quindi il model e il controller sono praticamente uguali a quelli della categoria con le differenze delle classi che richiamano e la tabella a cui fa riferimento, come mostrato nelle immagini qui sotto.

```

92     public function getProducts(){
93
94         if ($_SERVER["REQUEST_METHOD"] == "POST") {
95
96             //Prendo la classe model
97             require_once 'application/models/product.php';
98             $product = new Product();
99
100            $products = $product->getProducts();
101
102            //Stampo con json i valori dei coach
103            header( string: 'Content-Type: application/json');
104            echo json_encode($products);
105        }else{
106            header( string: "location: javascript://history.back()");
107        }
108    }

```

Figura 10 controller getProducts

```

17     public function getProducts(){
18         //Connetto al database
19         $conn = $this->connection->sqlConnection();
20
21         //Prendo i dati dell'utente in base alla mail
22         $sql = $conn->prepare( statement: "SELECT * FROM prodotto");
23
24         $dataArray = array();
25         //Eseguo la query
26         $sql->execute();
27
28         // Ciclo tutti i valori
29         while ($row = $sql->fetch()) {
30             array_push( &array: $dataArray, $row);
31         }
32         $conn = null;
33         return $dataArray;
34     }

```

Figura 11 model getProducts

La funzione i JavaScript invece è un o più complicata ed è spiegata qui sotto.

```

50     var divbody = document.getElementById( elementId: 'prodContainer');

```

Figura 12 variabile div iniziale

```

53     for (var i = 0; i < obj.length; i++) {
54         //Prendo i div principale per i prodotti
55         var divContainer = document.createElement( tagName: "div");
56         divContainer.setAttribute( qualifiedName: "class", value: "col-12 col-sm-6 col-lg-4");
57         var divWrapper = document.createElement( tagName: "div");
58         divWrapper.setAttribute( qualifiedName: "class", value: "single-product-wrapper");

```

Figura 13 variabili div

La prima immagine mostra una variabile che contiene il div più grande con tutti i prodotti al suo interno. Dopodiché creo 2 div con delle classi specifiche che andranno a contenere le informazioni del prodotto.

```

60
61
62
63
64
65
66
67
68
69
    //Creo la riga i cui inserire i campi e gli metto imposto la classe
    var div = document.createElement( tagName: "div");
    div.setAttribute( qualifiedName: "class", value: "product-img");

    //Creo un immagine imposto il source con il dato preso ee inserisco il div nel contenitore padre
    var img = document.createElement( tagName: "img");
    var src = "http://samtinfo.ch/gestionevendita2018/"+obj[i]["img"] ;
    img.setAttribute( qualifiedName: "src", src);
    div.appendChild(img);
    divWrapper.appendChild(div);

```

Figura 14 inserimento immagine

Quest'immagine mostra la parte in cui creo un div con la classe per contenere le immagini, poi creo l'immagine imposto il percorso, che va a prendere nell'FTP e lo inserisco nel div.

```

71
72
73
74
75
76
77
78
79
    //Creo un nuovo div che contiene le informazioni e lo metto nel padre
    var div = document.createElement( tagName: "div");
    var a = document.createElement( tagName: "a");
    a.setAttribute( qualifiedName: "href", value: "#");
    var h6 = document.createElement( tagName: "h6");
    h6.appendChild(document.createTextNode(obj[i]['nome']));
    a.appendChild(h6);
    div.appendChild(a);
    divWrapper.appendChild(div);

```

Figura 15 inserimento titolo

Con la parte di codice mostrata sopra vado a creare un ulteriore di con un link, che in futuro poterà alla pagina che mostra i dettagli del prodotto, e il titolo di esso al suo interno, e anche questo viene aggiunto al div padre.

```

82
83
84
85
    var p = document.createElement( tagName: "p");
    var prezzo = obj[i]['prezzo'] +" Fr.";
    p.appendChild(document.createTextNode(prezzo));
    divWrapper.appendChild(p);

```

Figura 16 inserimento prezzo

Con questo creo un paragrafo da inserire e ci metto il prezzo, formattato prime (variabile "prezzo").

```

88
89
90
        divContainer.appendChild(divWrapper);
        divbody.appendChild(divContainer);
        console.log(divbody);

```

Figura 17 unione finale

Una volta inserito tutto dentro il div "divWrapper" inserisco quest'ultimo nel div padre e il tutto in quello preso all'inizio. Questo percorso viene fatto in un for, come si è visto nella seconda riga mostrata, che passa tutti i prodotti in modo che tutti vengano inseriti con il formato corretto.



corda

10 Fr.

Figura 18 mostra prodotto

Questa è la visualizzazione di un prodotto d'esempio inserito hard coded da me.

Problemi riscontrati e soluzioni adottate

Non ho avuto particolari problemi

Punto della situazione rispetto alla pianificazione

Oggi sono arrivato a buon punto con la pagina della messa in vendita dei prodotti.

Programma di massima per la prossima giornata di lavoro

Completare la pagina di messa in vendita dei prodotti.

Diario di lavoro

Luogo	SAMT
Data	06.02.2019

Lavori svolti

Questa mattina come prima cosa ho dovuto mettere a posto un problemino che c'era con il login dell'venditore, come siegato nella sezione "Problemi riscontrati e soluzioni adottate". Dopodiché ho lavorato alla pagina di mostra del dettaglio del prodotto, inizialmente ho preso quella originale e l'ho modificata in modo che vengano mostrate solo le informazioni necessarie, come mostrato nella figura qui sotto.

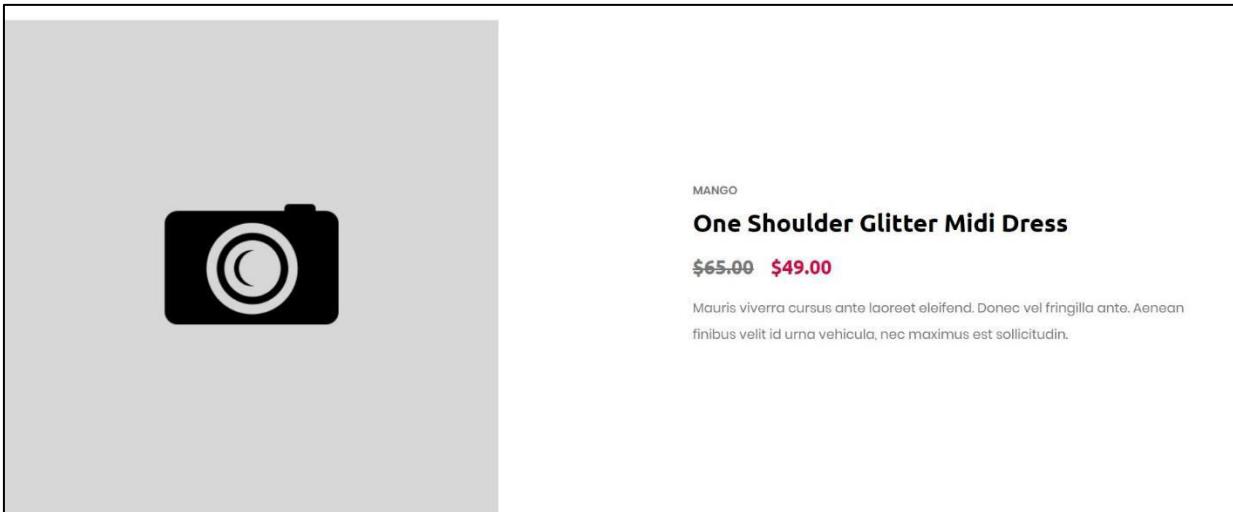


Figura 1 base pagina inserimento prodotto

Ci sono ancora delle informazioni del template, perché ancora non è completo, ma ho intanto inserito l'immagine di base, che è vuota essendo che deve poter permettere all'utente di cliccargli e inserire quella del prodotto che si vuole inserire. Gli altri campi saranno modificati in modo che siano degli input in cui inserire i dati e poi un bottone in fondo con cui inserire il prodotto.

La stessa pagina verrà poi modificata e riadattata anche per la modifica dei dati dei prodotti esistenti, al posto degli input vuoti ci saranno degli input i dati del prodotto selezionato.

Per fare in modo che l'immagine sia un input file in cui scegliere l'immagine ho controllato alla pagina "<https://stackoverflow.com/questions/2855589/replace-input-type-file-by-an-image>".

Alla fine la grafica della pagina sono riuscito a completarla, come mostrato sotto.

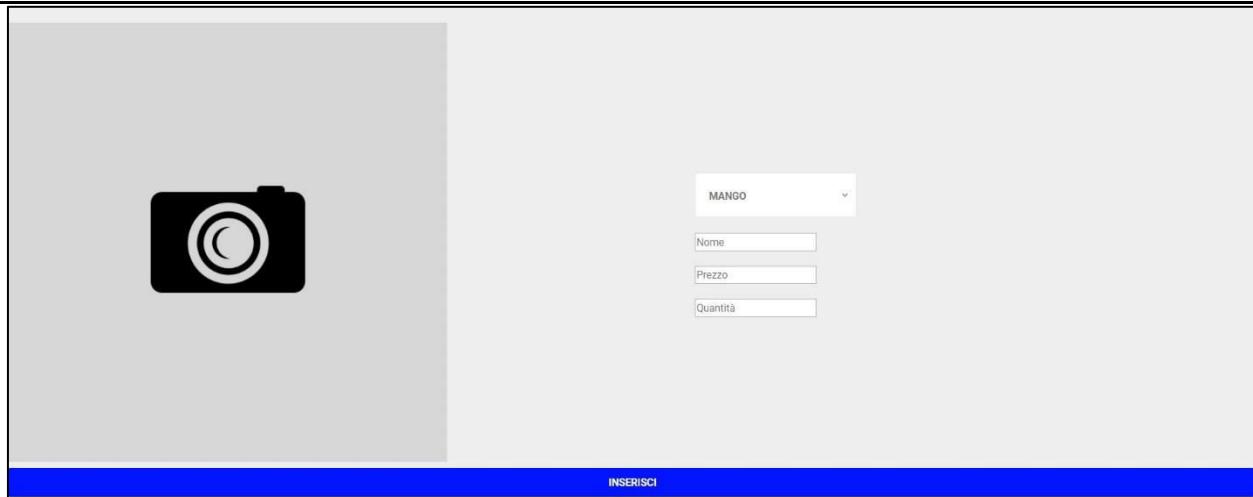


Figura 2 pagina inserimento prodotto

Nella pagina è implementata solo la parte grafica, non ci sono ancora controlli e la funzione del bottone non è ancora implementata.

Problemi riscontrati e soluzioni adottate

Un problema che ho avuto appena ho iniziato è stato con il login del venditore, non perché non funzionasse, ma non funzionava nel modo corretto, ovvero era sempre accessibile, anche se la sessione veniva distrutta, e di conseguenza tutte le sessioni, si riusciva comunque ad accedere, questo perché mancava il controllo all'interno della funzione controller che apre la pagina, così l'ho inserito.

```
26 if(isset($_SESSION['dealer'])) {  
27     require 'application/views/_templates/header.php';  
28     require 'application/views/dealer/static/header.php';  
29     require 'application/views/dealer/shop.php';  
30     require 'application/views/_templates/footer.php';  
31 } else{  
32     $this->index();  
33 }
```

Figura 3 controllo sessione apertura pagine

Il controllo si occupa di assicurarsi che la sessione sia attiva e la variabile esista, ma se non è così riapre la pagina di base, ovvero quella di login.

Oltre a questo però ho dovuto anche cambiare la funzione per il login, perché nel model c'era un controllo errato. Nella dichiarazione della variabile di sessione dopo che il login va a buon fine questa veniva inserita con l'indice sbagliato, di conseguenza il controllo finale non combaciava e dava sempre errore.

```
37 if (!strcmp($pass, $row["password"])) {  
38     // LOGIN VENDITORE  
39     $_SESSION['dealer'] = $mail;  
40     $checkSession = "dealer";  
41     break;
```

Figura 4 controllo email corretta login

```
53     switch ($checkSession) {  
54         case "dealer":  
55             return $_SESSION['dealer'];  
56         case "wrongPass":  
57             return $_SESSION['wrongPass'];  
58         default:  
59             return "ERRORE";  
60     }  
}
```

Figura 5 switch login venditore

L'errore era alle righe 39 e 40, perché venivano dichiarati con i nomi scorretti, ovvero "manager" al posto di "dealer", e quindi nel controller dello switch non trovava mai corretto e dava errore, dando errore le pagine non mi veniva mai aperta perché la variabile di sessione non esiste e quindi mi ritornava sempre nella pagina di login.

Ora ho messo a posto, come mostrato nelle immagini superiori, semplicemente inserendo i valori e indici coretti nel

Punto della situazione rispetto alla pianificazione

Oggi sono arrivato avanzato molto con la pagina della messa in vendita dei prodotti.
Ho creato la grafica della pagina di inserimento dei prodotti.

Programma di massima per la prossima giornata di lavoro

Completare la pagina di messa in vendita dei prodotti.

Diario di lavoro

Luogo	SAMT
Data	11.02.2019

Lavori svolti

Tutto quello che ho fatto questa mattina è stato impostare i controlli dei campi per l'inserimento dei nuovi prodotti e fare la funzione che li inserisce.
La funzione dei controlli è quasi uguale a quella dei controlli della registrazione ma non c'è il controllo della mail speciale e le espressioni regolari sono solo per le lettere e i numeri.

```

20     var regLetters = /^[ \u00c0-\u017E a-zA-Z\']+$/;
21     var regNumbers = /^[0-9\']+$/;
22     function convalidate(value, id, regexp) {
23
24         //Variabile del campo
25         var id = document.getElementById(id);
26
27         //Se il campo è vuoto e non rispetta l'espressione regolare
28         if (!regexp.test(value)) {
29
30             //Colora il testo in rosso e segna l'errore.
31             id.style.color = "red";
32             checkInput = false;
33             //Altrimenti
34         } else {
35             //Colora il testo di nero
36             id.style.color = "black";
37             checkInput = true;
38         }
39         confirm();
40     }
41
42 /**
43 * Funzione che conferma e attiva o disattiva il bottone
44 */
45 function confirm() {
46     var submit = document.getElementById( elementId: "insertProduct");
47
48     if(checkInput){
49         submit.disabled = false;
50     }else{
51         submit.disabled = true;
52     }
53 }
```

Figura 1 Funzione di convalida dei campi inserimento prodotti

Come si può notare c'è anche la funzione di conferma che controlla solo le la variabile di controllo è passata o meno. Anche per questo tutti i campi sono richiesti.

Durante l'implementazione dell'inserimento dei dati ho avuto dei problemi, spiegati nella sezione "Problemi riscontrati e soluzioni adottate".

Purtroppo l'inserimento dei dati non è ancora completo perché non sono riuscito a passare anche i dati dell'immagine, essendo che è un po' più complicato che con gli input normali.

Problemi riscontrati e soluzioni adottate

Ho avuto un problema durante l'inserimenti delle categorie esistenti nell'apposito select, questo perché non è un vero e proprio select ma con un contenitore, "div", che si va a generare alla creazione della pagina grazie ad un file JavaScript del template e crea quella visualizzazione, a causa di questo i dati non venivano passati con la richiesta POST. Per fare in modo che i dati passassero al select di base ho dovuto andare a modificare leggermente la creazione del div nel file JS originale e lavorare con JavaScript. Quello che ho fatto una volta modificato il div, semplicemente aggiungendo un id per lavorarci meglio, è stato creare una funzione in JavaScript che ogni volta che il valore di quest'ultimo cambiava andava ad inserire del select principale i valori, come mostra l'immagine qui sotto.

```
214 function setCategory(id){  
215     var category = document.getElementById(id);  
216  
217     //Elimino l'ultimo valore se ce ne sono più di 2  
218     var length = category.length;  
219     console.log(length);  
220     if(length >= 2){  
221         category.removeChild(category.lastElementChild);  
222     }  
223     //Creo una nuova option  
224     var option = document.createElement( tagName: "option");  
225  
226     //Inserisco all'interno della variabile il valore del paragrafo creato alla selezione della categoria  
227     option.setAttribute( qualifiedName: "value", document.getElementById( elementId: "current").innerHTML);  
228  
229     //Inserisco l'option creata e la selezione  
230     category.appendChild(option);  
231     category.getElementsByTagName( qualifiedName: 'option')[1].selected = 'true';  
232 }
```

Figura 2 funzione valore select categoria

Nelle prime righe controllo se c'è un valore oltre al primo di base con scritto "categoria" e se c'è viene eliminato in modo che possa essere rimpiazzato con la nuova scelta, essendo che questa pagina viene richiamata ogni volta che il valore della categoria cambia. Dopodiché creo un optiuon e inserisco il valore che è nel paragrafo e contiene la categoria scelta, il paragrafo viene creato e modificato dalla stessa funzione che crea il div dal select, poi metto l'option nel select e lo seleziono.

Con questa funzione richiamata ad ogni "onchange" del select riesco a passare tramite POST anche la categoria scelta, l'ultimo "problema" che mi rimane da completare è il passaggio dei dati dell'immagine.

Punto della situazione rispetto alla pianificazione

Ho quasi finito la pagina degli inserimenti dei nuovi prodotti, quindi sono molto indietro rispetto alla progettazione iniziale.

Programma di massima per la prossima giornata di lavoro

Completare la pagina di messa in vendita dei prodotti.

Diario di lavoro

Luogo	SAMT
Data	12.02.2019

Lavori svolti

Questa mattina ho completato l'inserimento dei dati del prodotto all'interno del database. È stato un processo semplice, a parte un problemino, spiegato nella sezione "Problemi riscontrati e soluzioni adottate". Per inserire dati vengono richiamati il controller e il model appropriati, spiegati qui sotto.

```

11  public function insertProduct()
12  {
13      if ($_SERVER["REQUEST_METHOD"] == "POST") {
14
15          require_once 'application/models/product.php';
16          $product = new ProductModel();
17
18          // Connessione all'ftp
19          $connFTP = ftp_connect( host: "efof.ftp.infomaniak.com");
20          $login = ftp_login($connFTP, username: "efof_gestvend", password: "GestVend_Admin_2018");
21
22          $image = isset($_FILES['imageQuestion'])? $_FILES['imageQuestion'] : null; //Prendo il nome del file
23          $name = $image['name'];
24
25          //Prendo il percorso temporaneo del file e gli cambio nome
26          $tmpName = $image['tmp_name'];
27          $newName = 'application/img/' . $name;
28          rename($tmpName, $newName);
29
30          //Imposto i permessi per il file
31          ftp_chmod($connFTP, mode: 0664, $newName);
32
33          $category = isset($_POST["category"])? $_POST["category"] : null;
34          $title = isset($_POST["title"])? $_POST["title"] : null;
35          $prize = isset($_POST["prize"])? $_POST["prize"] : null;
36          $quantity = isset($_POST["quantity"])? $_POST["quantity"] : null;
37
38          //Se entrambi i campi non sono vuoti
39          if ($category != null && $title != null && $prize != null && $quantity != null && $image != null) {
40
41              $var = $product->insertProduct($category, $title, $prize, $quantity, $newName, $newName);
42          }
43
44          header( string: "location: ". URL . "dealer/details");
45      }else{
46          header( string: "location: javascript://history.back()");
47      }
48  }

```

Figura 1 model inserimento prodotto

In questo caso il model svolge un ruolo abbastanza importante, prima stabilisce una connessione con l'FTP (righe 19-20) e dopodiché prende il file che è stato caricato, questo file viene salvato in un file temporaneo, infatti il passaggio successivo (righe 26-28) si occupa di prendere il valore del percorso del file e sostituirlo con il percorso nel ftp in cui inserire il file. Una volta fatto deve impostare i permessi per il file per tutti gli utenti (riga 31), la modalità "0664" imposta lettura e scrittura per il creatore del file e solo lettura per gli altri utenti, in modo che dal sito si possano visualizzare le immagini. La struttura dei permessi nel file è quella mostrata sotto.

UNIX permissions	0664
Owner	<input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write <input type="checkbox"/> Run <input type="checkbox"/> Setuid
Group	<input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write <input type="checkbox"/> Run <input type="checkbox"/> Setgid
Others	<input checked="" type="checkbox"/> Read <input type="checkbox"/> Write <input type="checkbox"/> Run <input type="checkbox"/> Sticky bit

Figura 2 permessi file FTP

Dopo aver impostato tutto prendo i valori degli altri campi inseriti e se sono validi richiamo la funzione del controller per inserire i dati. Una volta fatto tutto riapro la pagina di inserimento.

```

40     public function insertProduct($category, $title, $price, $quantity, $image){
41         //Connetto al database
42         $conn = $this->connection->sqlConnection();
43
44         //Prendo i dati dell'utente in base alla mail
45         $sql = $conn->prepare("INSERT INTO prodotto(nome_categoria, nome, prezzo, quantita, img) values(:categoria, :nome, :prezzo, :quantita, :img)");
46         $sql->bindParam(':categoria', $category, PDO::PARAM_STR);
47         $sql->bindParam(':nome', $title, PDO::PARAM_STR);
48         $sql->bindParam(':prezzo', $price, PDO::PARAM_INT);
49         $sql->bindParam(':quantita', $quantity, PDO::PARAM_INT);
50         $sql->bindParam(':img', $image, PDO::PARAM_STR);
51
52         //Eseguo la query
53         if(!$sql->execute()){
54             $conn = null;
55             $_SESSION['ExistData'] = true;
56             return $_SESSION['ExistData'];
57         }
58
59         $conn = null;
60         $_SESSION['created'] = true;
61         return $_SESSION['created'];
62     }

```

Figura 3 controller inserimento prodotto

Il controller si occupa semplicemente di inserire i dati passati nel database, sempre on PDO. La cosa particolare è alla fine, se l'inserimento non va a buon fine (righe 53-57) allora chiudo la connessione e ritorno una variabile di sessione per segnalare l'errore, mentre se va tutto a buon fin imposto un'altra variabile che segnala la corretta razione del prodotto nel database. La variabile di sessione creata nel controller viene usata nella pagina di inserimento per segnalare o meno l'errore, come nel caso del login, ma in questa pagina non si può applicare il popup, quindi ho dovuto gestirlo in modo diverso.

Figura 4 errore prodotto già esistente

Nel caso ci sia l'errore viene segnalato sotto al form on la scritta mostrata, ma se tutto va a buon fine allora vado nella pagina principale.

Un'ultima cosa che ho dovuto modificare nella pagina è stata il fatto che quando viene selezionata un'immagine viene mostrata al posto dell'immagine di base, ed è semplicemente bastato inserire una funzione che modifica la source dell'immagine con quello nuovo all'"onchange" dell'input, come mostrato qui sotto.

```

239  function setImage(file) {
240      //Controllo che abbia selezionato un file
241      if (file.files && file.files[0]) {
242
243          //creo un lettore di file
244          var reader = new FileReader();
245
246          //Quando viene caricato un file
247          reader.onload = function (e) {
248              //Modifico l'attributo src dell'immagine inserendo quello del file selezionato
249              $('#image')
250                  .attr('src', e.target.result)
251          };
252
253          //Setto i dati del file per l'URL
254          reader.readAsDataURL(file.files[0]);
255      }
256  }

```

Figura 5 mostra immagine selezionata

Un ulteriore cosa che ho dovuto gestire è il fatto che non vengano mostrati tutti i prodotti esistenti ma solo quelli che lui viene nel determinato negozio, per questo ho dovuto creare una nuova funzione che faccia una join per prendere solo i dati adatti, la funzione è la stessa l'unica differenza è la query, che è la seguente:

```

SELECT * from prodotto p
inner join vende v on v.nome_prodotto = p.nome
inner join negozio n on n.nome = v.nome_negozio
    and n.indirizzo = v.indirizzo_negozio
    and n.citta = v.citta_negozio
    WHERE n.email_gestore = :email

```

La query qui mostrata va a prendere tutto dalla tabella prodotto controllando anche le altre tabelle, partendo dal fondo, prima controlla che prende le informazioni dal negozio gestito dall'utente corrente, dopodiché prende tutti i valori nella tabella ponte "vende" che corrispondono a quelli del negozio e infine quelli in "prodotto" che corrispondono alla tabella vende, in questo modo vengono ritornati tutti i dati delle 3 tabella che corrispondono all'utente attuale.

L'ultima cosa che ho fatto è stata fare in modo che vengano mostrati i prodotti di una determinata categoria quando essa viene selezionata nella lista a sinistra, e vengono mostrati tutti se viene cliccato "Tutto".

Per farlo ho creato 2 nuove funzioni, uno che fa la richiesta per prendere i prodotti adatti mentre l'altra per inserire le informazioni, in JavaScript, la 3 funzione contiene il for con l'inserimento dei div mentre le altre 2 sono come quella mostrata qui sotto con la differenza nella funzione che richiamano.

```

141     function getProductsByCategory(category) {
142         xhttp.onreadystatechange = function () {
143             if (this.readyState === 4 && this.status === 200) {
144                 //Prendo i valori passati dal server e li metto in un array
145                 var obj = JSON.parse(xhttp.responseText);
146
147                 //Richiamo la funzione per inserire i prodotti
148                 insertProducts(obj);
149             }
150         }
151         xhttp.open( method: "POST", url: "/gestionevendita2018/product/getProductsByCategory", async: true);
152         xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
153         xhttp.send( body: "&category="+ category);
154     }

```

Figura 6 funzione mostra prodotti

Questa funzione va a richiamare quella che prende i prodotti in base alla categoria e gli passa la categoria richiesta, viene richiamata al click sul link della categoria.

```

32     public function getProductsByCategory() {
33
34         if ($_SERVER["REQUEST_METHOD"] == "POST") {
35
36             //Prendo la classe model
37             require_once 'application/models/product.php';
38             $product = new ProductModel();
39
40             //Prendo il valore della categoria
41             $category = isset($_POST["category"]) ? $_POST["category"] : null;
42
43             $categories = $product->getProductsByCategory($category);
44
45             //Stampo con json i valori dei coach
46             header( string: 'Content-Type: application/json');
47             echo json_encode($categories);
48         } else{
49             header( string: "location: javascript://history.back() ");
50         }
51     }

```

Figura 7 model selezione prodotti con categoria

La funzione sopra mostrata si occupa di prendere dal controller apposito nella classe “ProductModel” tutti i prodotti in base alla categoria passata e stamparlo come JSON i modo che Ajax possa prenderlo e riutilizzarlo.

```

64     public function getProductsByCategory($category){
65         //Prendo tutti i prodotti dell'utente corrente
66         $data = $this->getDealerProducts();
67
68         //Creo un array vuoto se faccio passare tutti i dati presi
69         $dataArray = array();
70         foreach ($data as $value){
71
72             //Se la categoria del prodott è corretta lo inserisco nell'array
73             if($value['nome_categoria'] == $category){
74                 array_push( &array: $dataArray, $value);
75             }
76         }
77
78         //Ritorno l'array
79         return $dataArray;
80     }

```

Figura 8 model che seleziona i prodotti in base alla categoria

La funzione non va a fare la query ma prende tutti i prodotti dalla funzione creata in precedenza e dopodiché li filtra con un foreach per ritornare, in un array, solo quelli della categoria richiesta.

Problemi riscontrati e soluzioni adottate

Prima di far funzionare tutto ho avuto un problemino con il passaggio dell'immagine tramite il form, questo perché per passare dei file bisogna inserire un attributo speciale nel tag <form>, come mostrato nella foto.

```
12 <form action="=php echo URL ?&gt;product/insertProduct" method="POST" enctype="multipart/form-data"&gt;</pre

```

La parte da inserire per farlo funzionare è “**enctype="multipart/form-data"**”. Un altro problema che ho trovato è la visualizzazione delle categorie nelle pagine in cui mostro anche i prodotti. A quanto pare avendo 2 funzioni che utilizzano Ajax l'unica che viene presa è l'ultima richiamata, mentre la prima chiamata POST non va a buon fine, per questo motivo nella pagina vengono mostrati tutti i prodotti ma non le categorie. Purtroppo questo problema non sono riuscito a metterlo a posto neanche con l'aiuto del professor Sartori, e di conseguenza ancora non è impostato.

Dopo un po' però ho trovato un modo, ovvero richiamare la funzione alla fine dell'inserimento delle categorie in modo che solo dopo vengano inseriti i prodotti e non allo stesso modo, in questo modo funziona.

Punto della situazione rispetto alla pianificazione

Ho finito la mostra dei prodotti quindi mi sono portato avanti.

Programma di massima per la prossima giornata di lavoro

Completare la pagina di modifica de prodotti inseriti.

Diario di lavoro

Luogo	SAMT
Data	13.02.2019

Lavori svolti

Oggi ho dedicato la mattinata a completare la pagina degli utenti, ovvero la mostra di tutti i prodotti e le 2 pagine principali.

Per mostrare i prodotti nella pagina "shop" ho semplicemente ripreso il codice della pagina del venditore, l'ho messa nella pagina dei clienti, e ho fatto riferimento alla funzione che prende tutti i prodotti e non solo alcuni, per la gestione delle categorie ho dovuto passare un nuovo argomento con il POST in modo che se è impostata allora vuol dire che è il venditore che richiede i prodotti, in quel caso deve prendere solo i suoi prodotti.

Il controllo è stato applicato alle funzioni come mostrato di seguito.

```

43     if(isset($_POST["dealer"])) {
44         $categories = $product->getProductsByCategory($category, true);
45     }else{
46         $categories = $product->getProductsByCategory($category);
47     }

```

Figura 1 controllo venditore controller

Nel controller viene controllato se è stata impostata la variabile che segnala il venditore, e in base alla risposta passa o meno al model un true per mostrarlo anche a lui.

```

64     public function getProductsByCategory($category, $dealer = false) {
65         if($dealer) {
66             //Prendo tutti i prodotti dell'utente corrente
67             $data = $this->getDealerProducts();
68         }else{
69             //Prendo tutti i prodotti dell'utente corrente
70             $data = $this->getProducts();
71         }

```

Figura 2 controllo venditore model

Nel model di conseguenza ho aggiunto un parametro da ricevere, che di default è false in modo che se viene portato a true vuol dire che è stato richiamato da un venditore e prende i prodotti per lui altrimenti li prende tutti.

Questi controlli sono fatti solo quando si richiedono i prodotti di una determinata categoria, perché quando si richiedono tutti viene richiamata la funzione giusta direttamente. Un ulteriore modifica che ho fatto è stata nella query di selezione, anche quando prendo tutti i prodotti ho fatto una join per ricevere tutte le informazioni anche del negozio e il venditore, in modo da poterle utilizzare senza fare troppi richiami.

Oltre a impostare questo ho anche rimodificato la pagina principale, in modo che renda accessibile solo la pagina di shop e non tutte, essendo abbastanza inutili.

Mi sono anche accorto che nell'inserimento dei prodotti da parte dei vendori non viene segnalato quale venditore e quale negozio mettono in vendita il prodotto, in più la chiave primaria del prodotto, ovvero il nome, non è abbastanza, perché questo impone a tutti i

negozi di avere lo stesso prezzo e la stessa quantità di prodotti, e questo non è corretto, per cui ne ridiscuterò con il mio docente per rivedere questa parte.

L'ultima cosa che ho fatto nella pagina principale è stata impostare il fatto che vengano mostrate tutte le categorie e al click portino alla pagina dei prodotti. Per farlo ho implementato una nuova funzione JavaScript che prenda le categorie e le inserisca nella pagina, come mostrato nell'immagine sotto.

```
71  function getCategoriesHome() {
72
73      xhttp.onreadystatechange = function () {
74          if (this.readyState === 4 && this.status === 200) {
75
76              //Prendo i valori passati dal server e li metto in un array
77              var obj = JSON.parse(xhttp.responseText);
78
79              //Prendo il corpo della tabella in cui inserire le righe
80              var divbody = document.getElementById('divContainer');
81
82              //Inserisco tutti le righe con i relativi corsi.
83              for (var i = 0; i < obj.length; i++) {
84
85                  //Creo i div e la riga finale e inserisco tutto
86                  var div = document.createElement( tagName: "div");
87                  div.setAttribute( qualifiedName: "class", value: "col-12 col-sm-6 col-md-4");
88                  var divIntern = document.createElement( tagName: "div");
89                  divIntern.setAttribute("class", "single_catagory_area d-flex align-items-center justify-content-center bg-img");
90                  var divFinal = document.createElement( tagName: "div");
91                  divFinal.setAttribute("class", "catagory-content");
92                  var a = document.createElement( tagName: "a");
93                  a.setAttribute( qualifiedName: "href", value: "<?php echo URL ?>home/shop");
94                  a.appendChild(document.createTextNode(obj[i]["nome"]));
95
96                  divFinal.appendChild(a);
97                  divIntern.appendChild(divFinal);
98                  div.appendChild(divIntern);
99                  divbody.appendChild(div);
100             }
101         }
102     }
103     xhttp.open( method: "POST", url: "/gestionevendita2018/home/getCategories", async: true);
104     xhttp.send();
105 }
```

Figura 3 inserimento categorie pagina principale

Problemi riscontrati e soluzioni adottate

Ho dovuto mettere a posto un problema con il login, essendo che avevo lo stesso codice dei popup nei 3 login ne ho messo uno solo nel file che viene richiamato per tutti ovvero l'header, ma nel rifare i test mi sono accorto che non caricava quello dei dati errati.

Prima i fare questo avevo aggiunto un secondo popup che segnala che l'utente non esiste, e per attivarlo gli ho aggiunto una classe e ho inserito una nuova funzione JavaScript che richiama quello, quindi ora il richiamo delle funzioni e le funzioni risultano come mostrato nelle immagini sottostanti.

```
6 <?php
7     if(isset($_SESSION['wrongPass'])) {
8         print "<script type='text/javascript'>
9             linkClickWrongPass();
10            </script>";
11        unset($_SESSION['wrongPass']);
12    } else if(isset($_SESSION['noUser'])) {
13        print "<script type='text/javascript'>
14            linkClickNoUser();
15            </script>";
16        unset($_SESSION['noUser']);
17    }
18
19 ?>
```

Figura 4 php per mostra popup

```
18     function linkClickWrongPass() {
19         $('.wrongPass').addClass('is-visible');
20     }
21
22     function linkClickNoUser() {
23         $('.noUser').addClass('is-visible');
24     }
```

Figura 5 funzioni JS mostra popup

Si può notare che in base alla variabile di sessione impostata viene aperto un popup diverso.

Punto della situazione rispetto alla pianificazione

Ho finito la mostra dei prodotti dappertutto quindi mi sono portato avanti.

Devo completare in modo ottimale la parte d'inserimento dei prodotti, ovvero inserendo anche quale negozio e venditore lo imposta.

Programma di massima per la prossima giornata di lavoro

Implementare la ricerca e il carrello.

Diario di lavoro

Luogo	SAMT
Data	18.02.2019

Lavori svolti

La prima cosa che ho fatto questa mattina è stata aggiornare il github inserendo tutte le nuove modifiche, guardare

<https://github.com/giairomauro/ProgettoVenditaPiccolaNegozianti> per vedere tutti gli aggiornamenti fatti.

Questa mattina ho avuto solo 2 ore di lezione di progetto, perché le ultime 2 sono state prese da una presentazione di una scuola superiore. A causa di questo ho pensato che sarebbe stato più sensato portare avanti solo la documentazione, essendo che continuare ad implementare mi avrebbe preso troppo tempo e non sarei neanche riuscito a andare troppo avanti.

Inizialmente ho scritto l'abstract, completo di tutto, informazioni di cui ero al corrente di dover inserire e anche quelle dettemi dopo l'ultimo progetto svolto.

Poi ho completato lo scopo, essendo che ricontrollando mi sono accorto che mancavano delle informazioni

Problemi riscontrati e soluzioni adottate

Non ho avuto problemi

Punto della situazione rispetto alla pianificazione

Ho finito la mostra dei prodotti dappertutto quindi mi sono portato avanti.

Devo completare in modo ottimale la parte d'inserimento dei prodotti, ovvero inserendo anche quale negozio e venditore lo imposta.

Programma di massima per la prossima giornata di lavoro

Implementare la ricerca e il carrello.

Diario di lavoro

Luogo	SAMT
Data	19.02.2019

Lavori svolti

Oggi inizialmente ho dovuto risolvere un problema con il login i vendori, mi sono accorto che il popup in caso l'utente richiesto non esiste non partiva, questo perché nella classe model non gestivo il caso in cui non esistevano dati.

Una volta risolto questo problema ho iniziato a lavorare alla ricerca.

La prima cosa che ho fatto è stata portare il campo di ricerca dall'header alla zona in cui vengono mostrati i prodotti, per evitare che l'utente faccia una ricerca nella pagina errata.

Una volta spostato ho anche dovuto modificare il file "core-style.css", per adattare lo stile del campo, essendo che prima era fato in modo che lo modifichi solo se era all'interno dell'header della pagina.

Dopo questo ho anche dovuto fare in modo che quando venga scatenato l'evento del form in cui è contenuto l'input, form da cui non ho potuto toglierlo per lo style, non ricarichi la pagina ma richiami semplicemente una funzione JavaScript. Per far questo ho creato una nuova funzione che disattivi l'evento, come mostrato di sotto.

```

213     function disableForm() {
214         console.log("as");
215         document.getElementById('buttonSearch').addEventListener( type: "click", listener: function(event) {
216             event.preventDefault();
217         })
    
```

Figura 1 funzione disabilitazione form

Questa funzione viene richiamata subito dopo quella che mostra le categorie e i prodotti. Dopo aver adattato la funzione ho creato la funzione JavaScript che va a ricercare tutti i prodotti che contengono nel nome la stringa cercata.

Per farlo però ho dovuto prima creare una variabile globale, che controlla che sia selezionata o meno una categoria, questa funzione la dichiaro all'inizio del file con valore false e la modifico nella funzione che prende tutti i prodotti mettendola sempre uguale a "false" e in quella che prende in base alla categoria uguagliandola alla categoria ricercata.

La funzione JavaScript è la seguente.

```

153     function searchProduct(value){
154         xhttp.onreadystatechange = function () {
155             if (this.readyState === 4 && this.status === 200) {
156
157                 //Prendo i valori passati dal server e li metto in un array
158                 var obj = JSON.parse(xhttp.responseText);
159
160                 //Richiamo la funzione per inserire i prodotti
161                 insertProducts(obj);
162             }
163         }
164         xhttp.open( method: "POST", url: "/gestionevendita2018/product/searchProducts", async: true);
165         xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
166         xhttp.send( body: "category="+ searchCategory +"&value="+ value);
167     }
    
```

Figura 2 Funzione di ricerca dei prodotti

Come si può notare anche questa funzione va a richiamare quella che inserisca i prodotti nella pagina, "insertProducts", dopo averli presi. A differenza delle altre questa funzione passa 2 argomenti con il POST, il valore della categoria e la stringa da ricercare nei prodotti da prendere.

La funzione va a richiamare il controller della classe product per prendere i dati.

```

112     public function searchProducts(){
113         //Se la funzione è richiamata come POST
114         if ($_SERVER["REQUEST_METHOD"] == "POST") {
115
116             //Prendo la classe model
117             require_once 'application/models/product.php';
118             $product = new ProductModel();
119
120             //Prendo le variabili passate dal POST
121             $category = isset($_POST["category"]) ? $_POST["category"] : null;
122             $value = isset($_POST["value"]) ? $_POST["value"] : null;
123
124             if($value != null){
125                 $products = $product->searchProducts($category, $value);
126             }
127
128             //Stampo con json i valori presi
129             header( string: 'Content-Type: application/json' );
130             echo json_encode($products);
131             //Altrimenti
132         }else{
133             //Ritorno alla pagina precedente
134             header( string: "location: javascript://history.back() " );
135         }
136     }

```

Figura 3 funzione che cerca tutti i prodotti in base alla stringa

La funzione prende i valori di post, come al solito, e dopodiché richiama il model passandogli i valori utili per lavorare.

La funzione nel model è simile a quella che prende i prodotti in base alla categoria, ovvero non fa una query diretta al database ma si basa su funzioni che lo fanno già.

```

119     public function searchProducts($category, $valueS) {
120         if($category === "false") {
121             //Prendo tutti i prodotti dell'utente corrente
122             $data = $this->getProducts();
123         }else{
124             //Prendo tutti i prodotti dell'utente corrente
125             $data = $this->getProductsByCategory($category);
126         }
127
128         //Creo un array vuoto e faccio passare tutti i dati presi
129         $dataArray = array();
130
131         foreach ($data as $value){
132             //Se il valore cercato è contenuto nel nome del prodotto passato
133             if(gettype(strpos($value['nome_prodotto'], $valueS)) != 'boolean'){
134
135                 //Inserisco il prodotto nell'array
136                 array_push( &array: $dataArray, $value);
137             }
138         }
139
140         //Ritorno l'array
141         return $dataArray;
142     }

```

Figura 4 model funzione ricerca prodotti

La funzione controlla se è già definita una categoria, se non lo è, ovvero il valore della variabile è “false” allora prende tutti i prodotti esistenti, altrimenti prende solo quelli con la categoria richiesta, il controllo è fatto su una stringa anche se si setta la variabile con mee un booleano in JavaScript, perché essendo che dopo gli do un valore in forma di stringa il programma lo trasforma automaticamente in una stringa.

Una volta presi i prodotti si analizzano e si controlla che il valore ricercato sia contenuto nel nome del prodotto che si passa nel foreach (righe 131-133). Per il controllo utilizzo la funzione “strpos()” che controlla a che posizione il valore nel secondo argomento è contenuto nella stringa passata come primo argomenti, se non è presente ritorna un booleano “false” se è presente ritorna il numero della posizione, per questo controllo che il tipo non sia booleano. Una volta conclusa anche la ricerca ho fatto in modo che venisse segnalato il link premuto, colorando il testo in blu. Pr farlo mi è bastato aggiungere una classe a tutti i link con la funzione “setAttribute” quando creo le categorie, e ho modificato l’inizio delle funzioni che vengono richiamate al click sui link.

```

116     function getProducts(link = null) {
117         //Se viene inserito l'el link
118         if(link != null) {
119             //Modifico la selezione del link cliccato e resetto gli altri
120             link.setAttribute("style", "color:#0315FF;");
121
122             //Controllo tutti gli elementi
123             for (i = 0; i < document.getElementsByClassName( classNames: "categories").length; i++) {
124                 //Se l'elemento è diverso da quello cliccato allora lo coloro di nero
125                 if (link != document.getElementsByClassName( classNames: "categories")[i]) {
126                     document.getElementsByClassName( classNames: "categories")[i].setAttribute( qualifiedName: "style", value: "color:FFFFFF;" );
127                 }
128             }
129         }

```

Figura 5 colorazione link cliccato

Ho aggiunto un argomento alla funzione che può essere null, come nel caso in cui non venga richiamata da un link. Se il link non è nullo allora li colora e prende tutti gli altri e li colora di nero, grazie al for su tutti i link con la classe impostata all'inserimento.

Dopo tutto questo ho impostato le basi per poter creare l'implementazione del carrello, ovvero ho creato la funzione in JavaScript, la classe Cart che andrà a contenere le informazioni dei carrelli temporanei, e la funzione nel controller dei clienti. Non ho foto perché ancora non c'è niente scritto ma sono solo istanziate le funzioni.

Infine ho messo a posto il problema con il database per quanto riguarda la chiave primaria di "prodotto", per evitare che tutti i negozi debbano avere la stessa quantità allo stesso prezzo di un determinato prodotto.

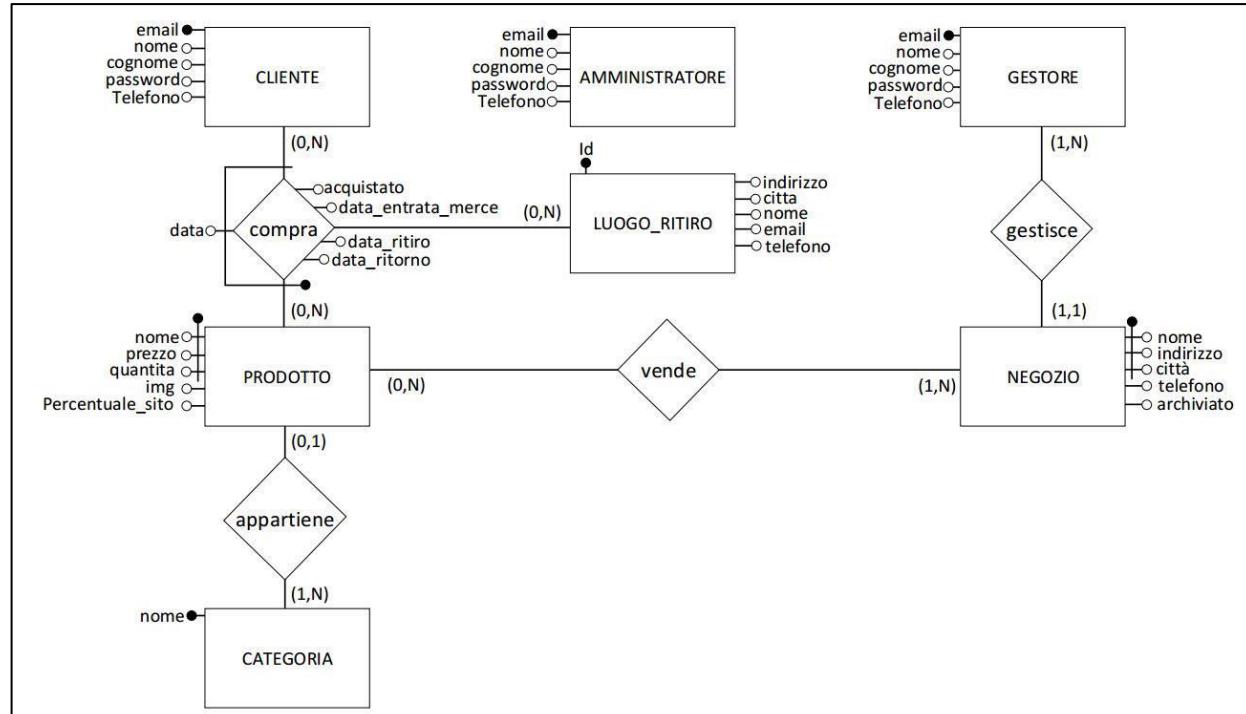


Figura 6 schema E-R con modifica chiave "prodotto"

Come chiave ne ho composta una con il nome il prezzo e la quantità, per fare in modo che lo stesso prodotto possa avere prezzo e quantità multiple.

La modifica è stata effettuata anche nel database. Per farlo però ho dovuto svuotare la tabella ponte "vende".

Tutto quello che è stato fatto è visibile nel sito <http://samtinfo.ch/gestionevendita2018/>.

Problemi riscontrati e soluzioni adottate

Ho avuto problemi con il css durante lo spostamento dell'input di ricerca, perché esso faceva riferimento ad un sistema che avevo modificato, ovvero il campo dentro l'header e dentro altri div e di conseguenza mi dava degli errori e non veniva visualizzato correttamente, mi è bastato però entrare nel file di css "core-style.css" e cercare "search" in modo che mi mostrava tutti i punti in cui viene modificato quell'input, dopodiché ho potuto modificarlo eliminando la classe dell'header.

Oltre a questo ho anche dovuto modificare il form, non direttamente esso ma il fatto che ricaricava la pagina al submit, come mostrato nella sezione precedente.

Un altro problema che ho avuto è stato con il controllo della variabile della categoria, perché inizialmente l'ho impostata come booleana e dava "true" o "false", ma poi mi sono accorto che mi serviva il valore di essa e allora è diventata una stringa, ma pensavo che se non cambiava rimaneva "false" e controllabile come un booleano, invece JavaScript lo imposta come stringa da subito. Di conseguenza ho dovuto cambiare il controllo, come mostrato nella sezione precedente sotto l'immagine "**Figura 7 model funzione ricerca prodotti**".

Punto della situazione rispetto alla pianificazione

Ho finito la ricerca dei prodotti tramite il nome.

Devo completare in modo ottimale la parte d'inserimento dei prodotti, ovvero inserendo anche quale negozio e venditore lo imposta.

Programma di massima per la prossima giornata di lavoro

Implementare il carrello.

Diario di lavoro

Luogo	SAMT
Data	20.02.2019

Lavori svolti

Questa mattina volevo iniziare a lavorare al carrello, ma prima ho dovuto mettere a posto alcuni problemi legati al cambiamento fatto alla fine della giornata di ieri al database come spiegato nella sezione che segue.

Dopo averlo fatto mi sono accorto che mancava la mostra dei dettagli del prodotto quando si clicca su di esso, allora ho optato per cercare un modal da potergli inserire, una volta fatto l'ho inserito nella pagina e ho continuato a lavorare al carrello senza implementarlo, perché non mi sembrava una priorità assoluta.

Una volta fatto ho cambiato la funzione “getProducts” che prende tutti i prodotti, facendo in modo che prenda i valori solo dei prodotti, perché in quel caso era inutile che prendeva tutti i dati anche dei vendori, i dati completi servono quando prende il prodotto singolo per mostrarlo o inserirlo nel carrello, quindi ho semplicemente cambiato la query, come di seguito.

SELECT * from prodotto

La select precedente non è stata eliminata ma spostata in una nuova funzione, “getProduct”, che prende i dati dei prodotti singoli.

```

43     public function getProduct($name, $price, $quantity){
44         //Connetto al database
45         $conn = $this->connection->sqlConnection();
46
47         //Prendo i dati dell'utente in base alla mail
48         $sql = $conn->prepare("SELECT * from prodotto p
49             inner join vende v on v.nome_prodotto = p.nome
50             inner join negozio n on n.nome = v.nome_negozio and n.indirizzo = v.indirizzo_negozio and n.citta = v.citta_negozio
51             WHERE p.nome LIKE :nome AND p.prezzo LIKE :prezzo AND p.quantita LIKE :quantita");
52         $sql->bindParam(':nome', $name, PDO::PARAM_STR);
53         $sql->bindParam(':prezzo', $price, PDO::PARAM_INT);
54         $sql->bindParam(':quantita', $quantity, PDO::PARAM_STR);
55
56         $dataArray = array();
57         //Eseguo la query
58         $sql->execute();
59
60         // Ciclo tutti i valori
61         while ($row = $sql->fetch()) {
62             array_push($array, $row);
63         }
64         $conn = null;
65         return $dataArray;
66     }

```

Figura 1 model select prodotti singoli

La funzione riceve i valori che compongono la chiave dell'elemento, e fa la ricerca, prendendo tutti i valori, in base a quelli passati, dopodiché inserisce tutto in un array e lo ritorna.

Dopo aver modificato questo sono andato avanti con il file JavaScript, essendo che se non era completo quello non si può andare avanti.

Prima di tutto ho fatto in modo che alla creazione il link passi il valore del suo name, il link sarebbe quello da cliccare per inserire i valori dentro il carrello, e il valore passato è una concatenazione delle chiavi dell'oggetto in questo formato “nome.prezzo.quantità”.

Una volta che questo valore arriva alla funzione viene decodificata da un'altra che ritorna i 3 valori in un array, come mostrato sotto.

```

295     function decode(value) {
296         return value.split(".");
297     }

```

Figura 2 funzione di decodifica

Una volta decodificato la funzione richiamata va, grazie ad ajax, ad inserire il valore nel database.

```

303     function addToCart(codedKeys) {
304         //Divido la chiave composta nei valori
305         var keys = decode(codedKeys);
306
307         xhttp.onreadystatechange = function () {
308             if (this.readyState === 4 && this.status === 200) {
309                 console.log(xhttp.responseText);
310             }
311         }
312         xhttp.open( method: "POST", url: "/gestionevendita2018/customer/addToCart", async: true);
313         xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
314         xhttp.send( body: "&name="+ keys[0] +"&price="+ keys[1] +"&quantity="+ keys[2]);
315
316
317     }

```

Figura 3 funzione JavaScript di aggiunta al carrello

La funzione richiama un controller, del cliente, che prende i valori che gli vengono passati, ovvero nome prezzo e quantità de prodotto (riga 314) e poi inseriscono i dati richiesti all'interno della tabella ponte “compra” che contiene tutti i dati dei prodotti acquistati, o semplicemente messi nel carrello, dell'utente.

```

22     public function addToCart()
23     {
24         //Se la sessione è aperta apro le pagine altrimenti no
25         if(isset($_SESSION['customer'])) {
26
27             //Prendo la classe model
28             require_once 'application/models/product.php';
29             require_once 'application/models/buy.php';
30             $product = new ProductModel();
31             $buy = new BuyModel();
32
33             //Prendo le variabili passate dal POST
34             $name = isset($_POST["name"])? $_POST["name"] : null;
35             $price = isset($_POST["price"])? $_POST["price"] : null;
36             $quantity = isset($_POST["quantity"])? $_POST["quantity"] : null;
37             $prod = "";
38
39             //Se entrambi i campi non sono vuoti
40             if ($name != null && $price != null&& $quantity != null) {
41                 $buy->insertData($name, $_SESSION['customer']);
42             }
43             //Altrimenti
44             }else{
45
46             }
47     }

```

Figura 4 controller aggiunta al carrello.

Come si può notare, la funzione aggiunge al carrello solo se ha fatto il login un utente, in caso contrario non funziona, questo perché la parte in cui un utente a caso inserisce i valori nel carrello non l'ho ancora implementato.

La funzione model che viene richiamata si occupa semplicemente di inserire i dati all'interno del database nella tabella apposita.

```

13     public function insertData($prodName, $custMail) {
14
15         //Connetto al database
16         $conn = $this->connection->sqlConnection();
17
18         //echo "Connected successfully";
19         $sql = $conn->prepare("INSERT INTO compra (nome_prodotto, email_cliente, data)
20             VALUES (:prodName, :custMail, now())");
21         $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
22         $sql->bindParam(':custMail', $custMail, PDO::PARAM_STR);
23
24         //Eseguo la query
25         $sql->execute();
26
27         //Se ci sono dei valori
28         if($sql->rowCount() > 0) {
29             $conn = null;
30             return true;
31         } else {
32             $conn = null;
33             return false;
34         }
35     }

```

Figura 5 model inserimento dati tabella "compra"

Come si può notare la parte del carrello non è ancora completata, mancano ancora la segnalazione degli elementi nel carrello, la mostra di essi, e tutta la parte dell'utente che non ha fatto il login. Un'altra cosa che devo implementare è il fatto che venga segnalato se il prodotto è effettivamente stato inserito o meno nel database.

Problemi riscontrati e soluzioni adottate

Il primo problema che ho avuto era con la mostra dei prodotti, perché ieri avevo eliminato tutto il contenuto della tabella ponte “vende” per modificare la chiave dei prodotti, ed essendo che la query per prendere i prodotti va riferimento anche a quella tabella per prendere i dati in modo completo allora ho dovuto riempirla nuovamente per poter mostrare i prodotti.

Punto della situazione rispetto alla pianificazione

Ho implementato parte del carrello.

Devo completare in modo ottimale la parte d'inserimento dei prodotti, ovvero inserendo anche quale negozio e venditore lo imposta.

Programma di massima per la prossima giornata di lavoro

Completare l'implementazione del carrello.

Diario di lavoro

Luogo	SAMT
Data	25.02.2019

Lavori svolti

Questa mattina inizialmente ho dovuto aggiustare un problemino con la mostra dei prodotti alla ricerca come mostrato nella sezione “Problemi riscontrati e soluzioni adottate”.

Una volta fatto mi sono accorto che era scaduta la mia prova gratuita di phpStorm, e di conseguenza ho cercato una soluzione, e ho trovato la versione per studenti che è gratuita, di conseguenza ho effettuato la registrazione con la mail di scuola e ho scaricato quella, in modo che non mi si chiuda l’editor ogni mezzora. Questo mi ha preso un po’ di tempo perché ho dovuto registrarmi e il sito ha dato un po’ di problemi a prendere la mail di scuola.

Una volta completata in modo corretto la tabella “compra” ho dovuto cambiare la query che inserisce i dati al suo interno, per inserire anche prezzo e quantità del prodotto.

```

21     public function insertData($prodName, $prodPrice, $prodQuantity, $custMail){
22
23         //Connetto al database
24         $conn = $this->connection->sqlConnection();
25
26         //echo "Connected successfully";
27         $sql = $conn->prepare("INSERT INTO compra (nome_prodotto, prezzo_prodotto, quantita_prodotto, email_cliente, data)
28             VALUES (:prodName, :prodPrice, :prodQuantity, :custMail, now())");
29         $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
30         $sql->bindParam(':prodPrice', $prodPrice);
31         $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
32         $sql->bindParam(':custMail', $custMail, PDO::PARAM_STR);
33
34         //Eseguo la query
35         $sql->execute();
36
37         //Se ci sono dei valori
38         if($sql->rowCount() > 0) {
39             $conn = null;
40             return true;
41         } else {
42             $conn = null;
43             return false;
44         }
45     }

```

Figura 1 funzione inserimento dati nella tabella compra

Una volta completata la funzione di inserimento ho iniziato a lavorare alla mostra dei prodotti nel carrello della pagina. Ma prima ho aggiunto una nuova colonna nella tabella chiamata “quantità_richiesta”, che contiene un numero che si incrementa ogni volta che l’utente inserisce quel prodotto nel carrello, per fare in modo che ne possa ordinare più di uno fino al massimo disponibile. Dopo aver fatto questo ho cambiato anche lo schema E-R e ho dovuto modificare la funzione di inserimento dei dati nel database un’altra volta.

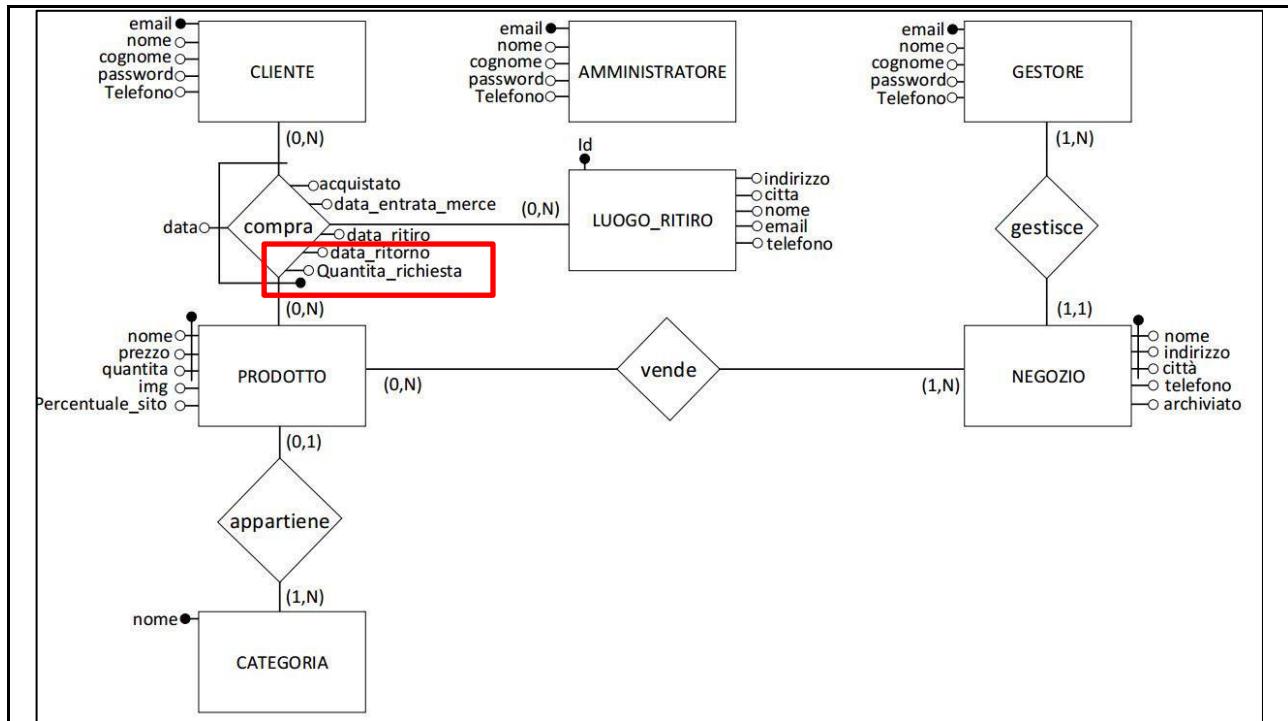


Figura 2 schema E-R con quantita_richiesta

Come si può notare ho semplicemente aggiunto un campo nella tabella ponte “compra”. Dopodiché ho dovuto modificare ulteriormente l'inserimento dei dati nella funzione, per fare in modo che un utente possa inserire nella tabella, ovvero il carrello, più volte lo stesso prodotto.

```

27     $quantity = 0;
28     $buyCount = $this->getDataByProduct($prodName, $prodPrice, $prodQuantity);
29     if(count($buyCount) > 0){
30
31         //Setto la nuova quantità e modifco il campo nella tabella
32         $quantity = $buyCount['quantita_richiesta'] + 1;
33         $sql = $conn->prepare("UPDATE compra SET quantita_richiesta = :quantity, data = now() WHERE nome_prodotto LIKE :prodName
34         AND prezzo_prodotto LIKE :prodPrice AND
35         quantita_prodotto LIKE :prodQuantity AND email_cliente LIKE :custMail");
36         $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
37         $sql->bindParam(':prodPrice', $prodPrice);
38         $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
39         $sql->bindParam(':custMail', $custMail, PDO::PARAM_STR);
40         $sql->bindParam(':quantity', $quantity, PDO::PARAM_INT);
41     }else{
42
43         //se il campo è nuovo inserisco la nuova riga
44         $quantity = 1;
45         $sql = $conn->prepare("INSERT INTO compra (nome_prodotto, prezzo_prodotto, quantita_prodotto, email_cliente, data, quantita_richiesta
46         VALUES (:prodName, :prodPrice, :prodQuantity, :custMail, now(), :quantity)");
47         $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
48         $sql->bindParam(':prodPrice', $prodPrice);
49         $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
50         $sql->bindParam(':custMail', $custMail, PDO::PARAM_STR);
51         $sql->bindParam(':quantity', $quantity, PDO::PARAM_INT);
52     }

```

Figura 3 controllo prodotto inserito

Prima di fare la query viene controllato se il prodotto è stato già inserito nel carrello dal cliente, questo viene fatto dalla funzione “getDataByProduct” (riga 28) che controlla se esiste già nella tabella quel prodotto messo dall'utente corrente e solo nel carrello, in caso positivo, quindi se la funzione ritorna un valore nell'array, modifica il valore della quantità richiesta incrementandolo, altrimenti inserisce il valore nuovo.

Qui sotto è mostrata la funzione che prende i dati.

```

96     public function getDataByproduct($prodName, $prodPrice, $prodQuantity){
97
98         //Connetto al database
99         $conn = $this->connection->sqlConnection();
100
101        //echo "Connected successfully";
102        $sql = $conn->prepare("SELECT * FROM compra WHERE nome_prodotto LIKE :prodName
103        AND prezzo_prodotto LIKE :prodPrice AND quantita_prodotto LIKE :prodQuantity AND email_cliente LIKE :custMail");
104        $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
105        $sql->bindParam(':prodPrice', $prodPrice);
106        $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
107        $sql->bindParam(':custMail', $_SESSION['customer'], PDO::PARAM_STR);
108
109        //Eseguo la query
110        $sql->execute();
111
112        $dataArray = array();
113        //Se ci sono dei valori
114        if($sql->rowCount() > 1) {
115
116            // Ciclo tutti i valori
117            while ($row = $sql->fetch()) {
118                array_push( $array, $dataArray, $row);
119            }
120            //Se c'è un solo valore lo inserisco
121        }else if($sql->rowCount() == 1) {
122            $dataArray = $sql->fetch();
123        }
124        $conn = null;
125        return $dataArray;
126    }
127

```

Figura 4 model getDataByProduct

Come si vede la funzione ritorna un array, nel caso non dovessero esserci i dati richiesti di conseguenza ritorna un array vuoto.

Questa è stata l'ultima cosa che ho fatto, quindi purtroppo non sono ancora riuscito a completare il carrello.

Problemi riscontrati e soluzioni adottate

Il primo problema che ho avuto è stato la mostra dei prodotti dopo la ricerca, perché avendo cambiato i dati che prende la query, ovvero quando ho messo che prende solo i dati del prodotto, e quindi ho dovuto cambiare l'indice nell'array che gli arriva alla funzione model, mettendo solo "nome" al posto di "nome_prodotto" al controllo dei dati.

Un secondo problema che ho avuto è stato con le foreign key, perché nella tabella "compra" ho dovuto inserire le 2 colonne "prezzo_prodotto" e "quantita_prodotto" che sono le 2 chiavi che ho aggiunto la scorsa lezione, e senza quelle 2 non potevo prendere il prodotto corretto dopo aver inserito i dati nella tabella per mostrarli nel carrello. Per farlo ho dovuto cancellare la tabella e ricrearla, perché è il metodo più rapido avendo già salvato in un file la creazione della tabella, mi è bastato copiarla e incollarla nel database.

Ho avuto dei problemini con la modifica delle query, ho dovuto provarli hardcoded su "phpMyAdmin" per controllare dove era l'errore e spesso era un campo inserito male o un dato non valido che passava.

Punto della situazione rispetto alla pianificazione

Ho messo a posto il carrello ma non è ancora completo.

Devo completare in modo ottimale la parte d'inserimento dei prodotti, ovvero inserendo anche quale negozio e venditore lo imposta.

Programma di massima per la prossima giornata di lavoro

Completare l'implementazione del carrello.

Diario di lavoro

Luogo	SAMT
Data	26.02.2019

Lavori svolti

La prima cosa che ho fatto questa mattina è stata modificare le funzioni della tabella "compra", per renderle più efficienti e ottimizzarle, in modo che sia anche più facile lavorarci. La funzione che ho modificato è stata quella che prende i dati in base al prodotto, innanzitutto gli mando la mail dell'utente tramite argomento, ma non obbligatoria, in modo che se dovesse servire prendere i dati solo dai prodotti sia possibile, dopodiché ho cambiato il modo in cui prende i prodotti, ovvero non fa un altro riferimento al database ma ad una funzione già esistente e poi ne controlla i dati.

```

144     public function getDataByproduct($prodName, $prodPrice, $prodQuantity, $custMail = null) {
145
146         //Se richiesto prendo i dati dell'utente
147         $products = array();
148         if($custMail != null)
149             $products = $this->getDataByMail($custMail);
150         else
151             $products = $this->getData();
152
153         $dataArray = array();
154
155         //Se ci sono dei valori
156         if(array_key_exists( key: 0,$products) && is_array($products[0])) {
157
158             // Ciclo tutti i valori
159             foreach ($products as $value) {
160                 //Se passa il prodotto richiesto
161                 if ($value['nome_prodotto'] == $prodName && $value['prezzo_prodotto'] == $prodPrice
162                     && $value['quantita_prodotto'] == $prodQuantity) {
163
164                     //inserisco il valore nell'array
165                     array_push( &array: $dataArray, $value);
166                 }
167             }
168             //Se c'è un solo valore lo inserisco
169         }else if(array_key_exists( key: 0,$products) && !is_array($products[0])) {
170             //Se passa il prodotto richiesto
171             if ($products['nome_prodotto'] == $prodName && $products['prezzo_prodotto'] == $prodPrice
172                 && $products['quantita_prodotto'] == $prodQuantity) {
173
174                 $dataArray = $products;
175             }
176         }
177         $conn = null;
178     }
179 }
```

Figura 1 model `getDataByProduct` per la tabella "compra"

La funzione, come mostrato, prende i dati con in base alla mail o meno, a dipendenza del campo passato, e dopodiché se esistono già dei dati controlla quanti sono, se è il primo allora controllo se è del prodotto richiesto e passo o meno il valore, se sono più di uno li controllo e passo solo quelli del prodotto richiesto, se non ce ne sono passo un array vuoto.
Un ulteriore modifica che ho fatto è stata nella funzione che inserisce i dati.

```

54     //Imposto la nuova quantità del prodotto
55     require_once 'application/models/product.php';
56     $product = new ProductModel();
57     $product->setQuantity($prodName, $prodPrice, $prodQuantity, $prodQuantity - 1);

```

Figura 2 modifica quantità prodotto

Dopo aver inserito o modificato la riga nella tabella prendo il prodotto e tramite la funzione nella foto mostrata (riga 57) modifico il valore della quantità di quel prodotto, semplicemente prendendo il valore attuale e diminuendolo di 1.

La funzione che modifica la quantità è mostrata qui sotto.

```

140     public function setQuantity($prodName, $prodPrice, $prodQuantity, $quantity){
141         //Connetto al database
142         $conn = $this->connection->sqlConnection();
143
144         //Imposto la nuova quantità
145         $sql = $conn->prepare("UPDATE prodotto SET quantita = :quantity WHERE nome LIKE :prodName
146             AND prezzo LIKE :prodPrice
147             AND quantita LIKE :prodQuantity");
148         $sql->bindParam(':quantity', $quantity, PDO::PARAM_INT);
149         $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
150         $sql->bindParam(':prodPrice', $prodPrice);
151         $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
152
153         //Eseguo la query
154         if(!$sql->execute()){
155             $conn = null;
156             return false;
157         }
158
159         $conn = null;
160         return true;
161     }

```

Figura 3 model setQuantity prodotto

La modifica viene effettuata con la query mostrata (righe 145-151) e grazie ai dati passati. Fare questa funzione mi ha dato un problema con la query quindi completarla mi ha preso del tempo.

Un'altra modifica che ho dovuto fare è stata nel file JavaScript, per fare in modo che anche se e non si aggiorna la pagina cambi comunque il valore passato della chiave di un determinato prodotto.

```

325
326     //Diminuisco di uno il valore della quantità del prodotto nel nome
327     keys[2]--;

```

Figura 4 modifica nome link prodotto

La funzione dopo aver inserito i dati e modificato la quantità del prodotto va a decrementare il valore della quantità da inserire nel nome del link, h è quello che viene codificato e decodificato e contiene i dati della chiave del prodotto.

Una volta fatto questo ho aggiunto un controllo prima di inserire i prodotti nella lista.

```

220     for (var i = 0; i < obj.length; i++) {
221         //Se c'è ancora il prodotto
222         if(obj[i]['quantita'] > 0) {

```

Figura 5 controllo disponibilità prodotto

Dopo aver preso i dati prima di inserire il prodotto controllo che ci sia, ovvero che la quantità sia maggiore di 0 (riga 222) e solo in quel caso iniziano a creare i componenti da inserire.

La modifica viene attuata in modo provvisorio, ne discuterò con il docente per quando applicarla, perché potrebbe avere più senso farla solo all'acquisto dei prodotti, perché se un utente si mette tutti i prodotti nel carrello senza mai comprarli non saranno più disponibili agli altri, ma se vengono cambiati solo all'acquisto c'è il rischio che 2 utenti comprino l'ultimo prodotto nello stesso momento e uno dei 2 avrà un errore.

Una volta completato questo ho iniziato a lavorare per fare in modo che vengano inseriti nella mostra del carrello i prodotti aggiunti.

Per farlo ho utilizzato JavaScript e dopo aver inserito i dati nel database prendo i dati dal carrello e "codifico" i valori con la funzione apposita nel formato mostrato nello scorso diario.

```

343     function getCart() {
344         xhttp.onreadystatechange = function () {
345             if (this.readyState === 4 && this.status === 200) {
346
347                 //Prendo i valori passati dal server e li metto in un array
348                 var obj = JSON.parse(xhttp.responseText);
349
350                 if(typeof (obj[0]) === "object"){
351                     for(var i = 0; i < obj.length; i++){
352                         var codedKeys = encode(obj[i]['nome_prodotto'], obj[i]['prezzo_prodotto'], obj[i]['quantita_prodotto']);
353
354                         //richiamo la funzione che inserisce i prodotti
355                         modifyCart(codedKeys);
356                     }
357                 }else{
358                     var codedKeys = encode(obj['nome_prodotto'], obj['prezzo_prodotto'], obj['quantita_prodotto']);
359
360                     //richiamo la funzione che inserisce i prodotti
361                     modifyCart(codedKeys);
362                 }
363             }
364         }
365         xhttp.open( method: "POST", url: "/gestionevendita2018/customer/getCart", async: true);
366         xhttp.send();
367     }

```

Figura 6 JS per prenderei valori del carrello

Dopo aver preso i valori controllo se ce ne sono più di uno o uno per prendere i valori nel modo corretto, e poi richiamo un'altra funzione che va a modificare i valori della mostra del carello.

L'ultima funzione non è ancora completata quindi non ho fatto una foto, la farò domani e che completerò la mostra dei prodotti nel carrello nella pagina.

Problemi riscontrati e soluzioni adottate

Un errore che ho avuto è stato quando cambiavo la quantità del prodotto dopo averlo messo nel carrello, perché dovevo impostare le Foreign keys delle tabelle "vende" e "compra" legate alla tabella "prodotto" le impostazioni degli "on update", impostandole come "cascade", e "on delete", mettendo "no action".

Punto della situazione rispetto alla pianificazione

Ho portato avanti il carrello ma non è ancora completo.

Devo completare in modo ottimale la parte d'inserimento dei prodotti, ovvero inserendo anche quale negozio e venditore lo imposta.

Programma di massima per la prossima giornata di lavoro

Completare l'implementazione del carrello.

Diario di lavoro

Luogo	SAMT
Data	27.02.2019

Lavori svolti

La prima cosa che ho fatto stamattina è stato aggiornare il github, ovvero inserendo i file che mancano e quelli che già c'erano ma sono stati modificati, dopodiché ho modificato il file "Readme.md" permettendo una migliore navigazione nel progetto una volta aperto il git. Dopo ho creato il file "dati_progetto.docx" che contiene il link al sito e i dati degli utenti già creati per testarlo.

Una volta messo tutto a posto e aggiornato, ho continuato la funzione che va a mettere i prodotti nel carrello della pagina per poterli visualizzare. Prima ho modificato la funzione che prende i valori del carrello per fare in modo che inserisca tutto in un array, per controllarlo più correttamente nella funzione dopo.

```

343     function getCart() {
344         xhttp.onreadystatechange = function () {
345             if (this.readyState == 4 && this.status == 200) {
346
347                 //Prendo i valori passati dal server e li metto in un array
348                 var obj = JSON.parse(xhttp.responseText);
349
350                 //variabile che contiene tutte le chiavi
351                 var codedKeys = new Array();
352
353                 if(typeof (obj[0]) == "object") {
354                     for(var i = 0; i < obj.length; i++){
355                         var codedKey = encode(obj[i]['nome_prodotto'], obj[i]['prezzo_prodotto'], obj[i]['quantita_prodotto']);
356                         codedKeys.push(codedKey);
357                     }
358                 }else{
359                     var codedKey = encode(obj['nome_prodotto'], obj['prezzo_prodotto'], obj['quantita_prodotto']);
360                     codedKeys.push(codedKey);
361                 }
362                 //richiamo la funzione che inserisce i prodotti
363                 modifyCart(codedKeys);
364             }
365         }
366         xhttp.open( method: "POST", url: "/gestionevendita2018/customer/getCart", async: true);
367         xhttp.send();
368     }
}

```

Figura 1 funzione getCart

Prima di codificare i valori creo un array (riga 351) e dopodiché i valori li inserisco dentro esso, e alla fine lo passo con tutti i dati all'interno.

```
374     function modifyCart(codedKeys) {
375         //Decodifico tutte le chiavi
376         var keys = new Array();
377         for(var i = 0; i < codedKeys.length; i++) {
378             keys.push(decode(codedKeys[i]));
379         }
380
381         //Prendo il corpo del div in cui inserire i campi
382         var divBody = document.getElementById(elementId: 'cart-list');
383         divBody.innerHTML = "";
384
385         xhttp.onreadystatechange = function () {
386             if (this.readyState === 4 && this.status === 200) {
```

Figura 2 JavaScript modifyCart 1

All'inizio della funzione prendo tutti i valori delle chiavi le decodifico e le inserisco in un array.

Dopodiché prendo il corpo in cui inserire ogni prodotto.

Una volta controllo che sia andato a buon fine la richiesta al php.

```

389     var obj = JSON.parse(xhttp.responseText);
390
391     //Prendo i div principale per i prodotti
392     var divContainer = document.createElement( tagName: "div");
393     divContainer.setAttribute( qualifiedName: "class", value: "single-cart-item");
394
395     var a = document.createElement( tagName: "a");
396     a.setAttribute( qualifiedName: "class", value: "product-image");
397     var img = document.createElement( tagName: "img");
398     var src = "http://samtinfo.ch/gestionevendita2018/" + obj[0]["img"];
399     img.setAttribute( qualifiedName: "class", value: "cart-thumb");
400     img.setAttribute( qualifiedName: "src", src);
401     img.setAttribute( qualifiedName: "alt", obj[0]['nome_prodotto']);
402     img.setAttribute( qualifiedName: "title", obj[0]['nome_prodotto']);
403     a.appendChild(img);
404
405     var div = document.createElement( tagName: "div");
406     div.setAttribute( qualifiedName: "class", value: "cart-item-desc");
407     var span = document.createElement( tagName: "span");
408     span.setAttribute( qualifiedName: "class", value: "product-remove");
409     var i = document.createElement( tagName: "i");
410     i.setAttribute( qualifiedName: "class", value: "fa fa-close");
411     i.setAttribute( qualifiedName: "aria-hidden", value: "ture");
412     span.appendChild(i);
413     div.appendChild(span);
414
415     var span = document.createElement( tagName: "span");
416     span.setAttribute( qualifiedName: "class", value: "badge");
417     span.appendChild(document.createTextNode( data: "Prodotto"));
418     div.appendChild(span);

```

Figura 3 JavaScript modifyCart 2

```

419
420     var h6 = document.createElement( tagName: "h6");
421     h6.appendChild(document.createTextNode(obj[0]['nome_prodotto']));
422     div.appendChild(h6);
423
424     var p = document.createElement( tagName: "p");
425     p.setAttribute( qualifiedName: "class", value: "price");
426     p.appendChild(document.createTextNode( data: "prezzo: " + obj[0]['prezzo_prodotto']));
427     div.appendChild(p);
428
429     a.appendChild(div);
430     divContainer.appendChild(a);
431     divBody.appendChild(divContainer);
432
433 }
434 xhttp.open( method: "POST", url: "/gestionevendita2018/product/getProduct", async: true);
435 xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
436 xhttp.send( body: "&name=" + keys[0] + "&price=" + keys[1] + "&quantity=" + keys[2]);
437

```

Figura 4 JavaScript modifyCart 3

Le ultime 2 immagini mostrano la creazione delle parti della mostra del prodotto nel carrello, con le relative classi e informazioni che devono mostrare.

Le informazioni del prodotto vengono mostrate correttamente, come mostrato nell'immagine qui sotto.

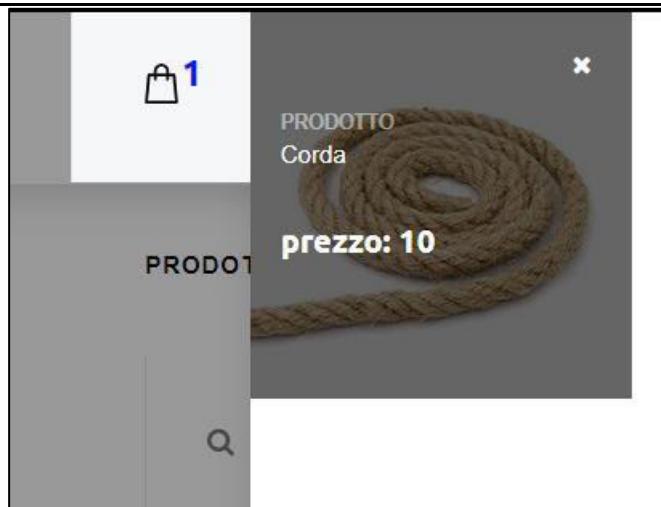


Figura 5 mostra info prodotto

Questo metodo però riscontra un problema, spiegato nella sezione “Problemi riscontrati e soluzioni adottate”.

Problemi riscontrati e soluzioni adottate

Il problema che ho riscontrato oggi è come uno già avuto in precedenza e riguarda la richiesta in post alla funzione che prende i prodotti del carrello e li inserisce nella pagina. Il problema è che la richiesta, essendo asincrona dato che la sincrona è deprecata, è troppo veloce e di conseguenza va a prendere solo l'ultima disponibile e non tutte quelle possibili. Ho pensato ad una possibile soluzione, ma non sono riuscito ad attuarla oggi per mancanza di tempo, essendo che fare la mostra mi ha preso molto tempo. La soluzione potrebbe essere di prendere già i prodotti direttamente nella funzione del controller, quindi già nella richiesta in “getCart” si potrebbe prendere i dati dal carrello dell’utente e dopodiché prendere i prodotti e passare direttamente quelli senza dover prendere tramite JavaScript i dati delle chiavi dei prodotti e richiederli con un'altra funzione.

Punto della situazione rispetto alla pianificazione

Ho portato avanti il carrello ma non è ancora completo.

Devo completare in modo ottimale la parte d'inserimento dei prodotti, ovvero inserendo anche quale negozio e venditore lo imposta.

Programma di massima per la prossima giornata di lavoro

Completare l'implementazione del carrello.

Diario di lavoro

Luogo	SAMT
Data	18.03.2019

Lavori svolti

La prima cosa che ho fatto è stata completare la mostra delle informazioni del carrello appena l'utente accede o quando inserisce un nuovo prodotto nel carrello.

Per fare questo, in JavaScript ho creato delle funzioni che se ne occupano.

```

371  function getCart(){
372      xhttp.onreadystatechange = function () {
373          if (this.readyState === 4 && this.status === 200) {
374
375              //Prendo i valori passati dal server e li metto in un array
376              var obj = JSON.parse(xhttp.responseText);
377
378              //variabile che contiene tutte le chiavi
379              var codedKeys = new Array();
380
381              if(typeof (obj[0]) == "object"){
382                  for(var i = 0; i < obj.length; i++){
383                      //richiamo la funzione che inserisce i prodotti
384                      modifyCart(obj[i]);
385                  }
386              }else{
387                  //richiamo la funzione che inserisce i prodotti
388                  modifyCart(obj);
389              }
390          }
391      }
392      xhttp.open( method: "POST", url: "/gestionevendita2018/customer/getCart", async: true);
393      xhttp.send();
394  }

```

Figura 1 Funzione getCart

La funzione che prende tutti i dati del carrello ha subito delle modifiche, non va più a codificare i dati ma passa direttamente i dati alla funzione che li inserisce nella pagina, questo perché la funzione “getCart” del controller passa direttamente i dati dei prodotti e non più quelli del carrello, come mostrato nell’immagine qui sotto.

```

81     public function getCart()
82     {
83         //Se la sessione è aperta apro le pagine altrimenti no
84         if(isset($_SESSION['customer'])) {
85
86             //Prendo la classe model
87             require_once 'application/models/buy.php';
88             require_once 'application/models/product.php';
89             $buy = new BuyModel();
90             $product = new ProductModel();
91
92             //Array che conterrà tutti i prodotti
93             $prodArray = array();
94
95             //Prendo i dati del carrello
96             $cartProducts = $buy->getDataByMail($_SESSION['customer']);
97
98             //Per ogni dato inserisco il prodotto relativo nell'array
99             foreach ($cartProducts as $value)
100                 array_push( $prodArray,
101                             $product->getProduct($value['nome_prodotto'], $value['prezzo_prodotto'], $value['quantita_prodotto']));
102
103             //Stampo con json l'array con i prodotti
104             header( string: 'Content-Type: application/json');
105             echo json_encode($prodArray);
106             //Altrimenti
107         }else{
108
109     }
110 }
```

Figura 2 funzione controller getCart

La funzione prende i dati dal carrello e dopo con quelli prende tutti i prodotti che l'utente ha. Oltre a questo anche quando viene aggiunto un prodotto al carrello viene richiamata un'altra funzione.

```

322     function addToCart(obj) {
323         //Divido la chiave composta nei valori
324         var keys = decode(obj.name);
325
326         //Se il prodotto è disponibile
327         if(keys[2] > 0) {
328             xhttp.onreadystatechange = function () {
329                 if (this.readyState === 4 && this.status === 200) {
330
331                     //Diminuisco di uno il valore della quantità del prodotto nel nome
332                     keys[2]--;
333                     obj.name = encode(keys[0], keys[1], keys[2]);
334
335                     //Prendo i valori che vanno mostrati nel carrello
336                     getCartProduct(keys[0], keys[1], keys[2]);
337                 }
338             }
339             xhttp.open( method: "POST", url: "/gestionevendita2018/customer/addToCart", async: true);
340             xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
341             xhttp.send( body: "&name=" + keys[0] + "&price=" + keys[1] + "&quantity=" + keys[2]);
342         }
343     }
```

Figura 3 funzione addToCart

La funzione controlla che il prodotto abbia un valore maggiore di 0 e in quel caso va a richiamare la funzione e prende i valori del carrello con quel prodotto, una volta fatto diminuisce il valore che è scritto nella pagina sul prodotto e dopodiché richiama la funzione che prende il prodotto e lo inserisce nel carrello.

```

351     function getCartProduct(name, price, quantity){
352         xhttp.onreadystatechange = function () {
353             if (this.readyState === 4 && this.status === 200) {
354
355                 //Prendo i valori passati dal server e li metto in un array
356                 var obj = JSON.parse(xhttp.responseText);
357
358                 //richiamo la funzione che inserisce i prodotti
359                 modifyCart(obj[0][0]);
360             }
361         }
362         xhttp.open( method: "POST", url: "/gestionevendita2018/customer/getCartProduct", async: true);
363         xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
364         xhttp.send( body: "&name=" + name + "&price=" + price + "&quantity=" + quantity);
365     }

```

Figura 4 funzione getCartProduct JS

La funzione si occupa di prendere il prodotto in base ai dati passati e passarlo alla funzione che li inserisce nella pagina.

Oltre alla modifica della funzione “getCart”, nella classe controller “Customer” ci sono altre funzioni nuove, la funzione che inserisce i dati nella tabella “addToCart” e quella che prende il prodotto singolo da aggiungere al carrello.

```

22     public function addToCart()
23     {
24         //Se la sessione è aperta apro le pagine altrimenti no
25         if(isset($_SESSION['customer'])) {
26
27             //Prendo la classe model
28             require_once 'application/models/buy.php';
29             $buy = new BuyModel();
30
31             //Prendo le variabili passate dal POST
32             $name = isset($_POST["name"])? $_POST["name"] : null;
33             $price = isset($_POST["price"])? $_POST["price"] : null;
34             $quantity = isset($_POST["quantity"])? $_POST["quantity"] : null;
35
36             //Se entrambi i campi non sono vuoti
37             if ($name != null && $price != null&& $quantity != null) {
38                 $buy->insertData($name, $price, $quantity, $_SESSION['customer']);
39             }
40             //Altrimenti
41             }else{
42
43             }
44     }

```

Figura 5 funzione addToCart

La funzione prende i valori della chiave del prodotto e poi inserisce le informazioni nella tabella grazie alla funzione model “insertData”, che ha anch’essa subito dei cambiamenti.

```

21     public function insertData($prodName, $prodPrice, $prodQuantity, $custMail){
22         //Connetto al database
23         $conn = $this->connection->sqlConnection();
24         $sql = null;
25
26         $quantity = 0;
27         $buyCount = $this->getDataByProduct($prodName, $prodPrice, $prodQuantity, $custMail);
28
29         //Controllo se il prodotto è già inserito
30         if(count($buyCount) > 0){
31
32             //Setto la nuova quantità e modifco il campo nella tabella
33             if(is_array($buyCount[0])){
34                 $quantity = $buyCount[0]['quantita_richiesta'] + 1;
35             }
36             else{
37                 $quantity = $buyCount['quantita_richiesta'] + 1;
38
39                 $sql = $conn->prepare("UPDATE compra SET quantita_richiesta = :quantity, data = now() WHERE nome_prodotto LIKE :prodName
40                 AND prezzo_prodotto LIKE :prodPrice AND
41                 quantita_prodotto LIKE :prodQuantity AND email_cliente LIKE :custMail");
42                 $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
43                 $sql->bindParam(':prodPrice', $prodPrice);
44                 $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
45                 $sql->bindParam(':custMail', $custMail, PDO::PARAM_STR);
46                 $sql->bindParam(':quantity', $quantity, PDO::PARAM_INT);
47             }else{
48
49                 //se il campo è nuovo inserisco la nuova riga
50                 $quantity = 1;
51                 $sql = $conn->prepare("INSERT INTO compra (nome_prodotto, prezzo_prodotto, quantita_prodotto, email_cliente, data, quantita_richiesta)
52                 VALUES (:prodName, :prodPrice, :prodQuantity, :custMail, now(), :quantity)");
53                 $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
54                 $sql->bindParam(':prodPrice', $prodPrice);
55                 $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
56                 $sql->bindParam(':custMail', $custMail, PDO::PARAM_STR);
57                 $sql->bindParam(':quantity', $quantity, PDO::PARAM_INT);
58             }
59
60             //Eseguo la query
61             $sql->execute();
62             print_r($sql->errorInfo()[2]);
63
64             //Imposto la nuova quantità del prodotto
65             require_once 'application/models/product.php';
66             $product = new ProductModel();
67             $product->setQuantity($prodName, $prodPrice, $prodQuantity, $prodQuantity - 1);
68
69             //Se ci sono dei valori
70             if($sql->rowCount() > 0){
71                 $conn = null;
72                 return true;
73             }
74             else{
75                 $conn = null;
76                 return false;
77             }
78         }
    
```

Figura 6 funzione model insertData parte 1

La funzione prima controlla se il dato esiste già, se è il caso allora non inserisce un nuovo dato ma modifica quello esistente inserendo la nuova quantità richiesta del relativo prodotto, altrimenti inserisce i nuovi dati.

```

46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
    
```

Figura 7 funzione model insertData parte 2

Oltre alla funzione che inserisce o modifica i dati ce n'è una che prende il prodotto richiesto da inserire nel carrello e lo stampa, in modo che grazie a JavaScript si possa inserire nella pagina, come mostrato nell'immagine che segue.

```

49     public function getCartProduct()
50     {
51         //Se la sessione è aperta apro le pagine altrimenti no
52         if(isset($_SESSION['customer'])) {
53
54             //Prendo la classe model
55             require_once 'application/models/product.php';
56             $product = new ProductModel();
57
58             //Prendo le variabili passate dal POST
59             $name = isset($_POST["name"]) ? $_POST["name"] : null;
60             $price = isset($_POST["price"]) ? $_POST["price"] : null;
61             $quantity = isset($_POST["quantity"]) ? $_POST["quantity"] : null;
62
63             //Array che conterrà tutti i prodotti
64             $prodArray = array();
65
66             //Per ogni dato inserisco il prodotto relativo nell'array
67             array_push( &array: $prodArray, $product->getProduct($name, $price, $quantity));
68
69             //Stampo con json l'array con i prodotti
70             header( string: 'Content-Type: application/json');
71             echo json_encode($prodArray);
72             //Altrimenti
73         }else{
74
75         }
76     }

```

Figura 8 funzione getCartProduct

Per inserire i dati, utilizzo la funzione “modifyCart” mostrata nel diario precedente a questo, ma con delle modifiche, perché oltre a mostrare i dati deve anche mostrare il numero di oggetti presenti all’interno del carrello, e se un oggetto è già presente non deve reinserirlo ma semplicemente modificare il numero di prodotti di quel tipo richiesti, come mostrato nell’immagine.

```

401     function modifyCart(obj) {
402
403         //Prendo il corpo del div in cui inserire i campi
404         var divBody = document.getElementById( elementId: 'cart-list' );
405
406         //Controllo se il prodotto è già nel carrello
407         if(!document.getElementById( elementId: "cart"+ obj['nome_prodotto'])) {
408
409             //Modifico la variabile del numero di oggetti
410             cartObjects += parseInt(obj['quantita_richiesta']);
411
412             //Prendo i div principale per i prodotti
413             var divContainer = document.createElement( tagName: "div");
414             divContainer.setAttribute( qualifiedName: "class", value: "single-cart-item");
415             divContainer.setAttribute( qualifiedName: "id", value: "cart" + obj['nome_prodotto']);
416
417             //Inserisco il link che conterrà tutto
418             var a = document.createElement( tagName: "a");
419             a.setAttribute( qualifiedName: "class", value: "product-image");
420
421             //Inserisco l'immagine
422             var img = document.createElement( tagName: "img");
423             var src = "http://samtinfo.ch/gestionevendita2018/" + obj["img"];
424             img.setAttribute( qualifiedName: "class", value: "cart-thumb");
425             img.setAttribute( qualifiedName: "src", src);
426             img.setAttribute( qualifiedName: "alt", obj['nome_prodotto']);
427             img.setAttribute( qualifiedName: "title", obj['nome_prodotto']);
428             img.setAttribute( qualifiedName: "id", value: "img" + obj['nome_prodotto']);
429             a.appendChild(img);
430
431             //Creo il contenitore dei testi
432             var div = document.createElement( tagName: "div");
433             div.setAttribute( qualifiedName: "class", value: "cart-item-desc");

```

Figura 9 funzione modifica del carrello 1

Inizialmente la funzione non passa più tutti i dati ma viene richiamata ogni volta quindi l'unica cosa che fa è inserire l'oggetto che gli viene passato tramite argomento. Un primo controllo che si può vedere, viene fatto per assicurarsi che l'oggetto con quell'id non esiste già, e se è così crea l'oggetto come mostrato in quella e nella foto che segue, con le varie informazioni e Id.

```

435     //Inserisco il bottone di cancellazione
436     var span = document.createElement( tagName: "span");
437     span.setAttribute( qualifiedName: "class", value: "product-remove");
438     var i = document.createElement( tagName: "i");
439     i.setAttribute( qualifiedName: "class", value: "fa fa-close");
440     i.setAttribute( qualifiedName: "aria-hidden", value: "true");
441     span.appendChild(i);
442     div.appendChild(span);
443
444     //Inserisco la categoria
445     var span = document.createElement( tagName: "span");
446     span.setAttribute( qualifiedName: "class", value: "badge");
447     span.appendChild(document.createTextNode(obj['nome_categoria']));
448     div.appendChild(span);
449
450     var h6 = document.createElement( tagName: "h6");
451     h6.appendChild(document.createTextNode(obj['nome_prodotto']));
452     div.appendChild(h6);
453
454     var p = document.createElement( tagName: "p");
455     var prezzo = obj['prezzo_prodotto'] + " Fr";
456     p.setAttribute( qualifiedName: "class", value: "price");
457     p.appendChild(document.createTextNode( data: "prezzo: " + prezzo));
458     div.appendChild(p);
459
460     var p = document.createElement( tagName: "p");
461     p.setAttribute( qualifiedName: "class", value: "price");
462     p.setAttribute( qualifiedName: "id", value: "quantità" + obj['nome_prodotto']);
463     p.appendChild(document.createTextNode( data: "quantità: " + obj['quantita_richiesta']));
464     div.appendChild(p);
465
466     a.appendChild(div);
467     divContainer.appendChild(a);
468     divBody.appendChild(divContainer);

```

Figura 10 funzione modifica del carrello 2

```

470 }else{
471
472     //Modifico la variabile del numero di oggetti
473     cartObjects++;
474
475     //Modifico la quantità del prodotto richiesto
476     document.getElementById( elementId: "quantity"+ obj['nome_prodotto']).innerHTML = "";
477     document.getElementById( elementId: "quantity"+ obj['nome_prodotto']).appendChild(
478         document.createTextNode( data: "quantità: " + obj['quantita_richiesta']));
479 }
480
481 //Scrivo il numero di oggetti
482 for(var i = 0; i < document.getElementsByClassName( classNames: "cartObjects").length; i++) {
483     document.getElementsByClassName( classNames: "cartObjects")[i].innerHTML = "";
484     document.getElementsByClassName( classNames: "cartObjects")[i].appendChild(document.createTextNode(cartObjects));
485 }
486

```

Figura 11 funzione modifica del carrello 3

Se invece l'oggetto esiste già viene aumentata la quantità di quest'ultimo e poi modificata all'interno del carrello.

Alla fine della funzione viene preso l'oggetto che mostra il carrello in alto a destra e viene scritto il numero dei prodotti inseriti.

Dopodiché ho messo a posto l'inserimento dei prodotti da parte dei venditori, facendo in modo che venga anche inserito nella tabella “vende” che, per l'appunto, un determinato negozio vende un certo prodotto.

Per farlo ho inserito un nuovo select nella pagina di inserimento che permetta di scegliere quale negozio vende il determinato prodotto, come mostrato nell'immagine sottostante.



Figura 12 pagina inserimento prodotti

L'implementazione di questo select mi ha preso un po' di tempo, perché a differenza della categoria al submit non deve passare solo il dato che l'utente vede ma anche l'indirizzo e la città per poter avere la chiave completa, di conseguenza ho utilizzato anche in questo caso il metodo di concatenare in una stringa i dati e metterli come id dei vari campi, e una volta selezionato un dato viene preso l'id di questo e inserito nel select come per la categoria. La funzione che prende i dati e li mette nel select è uguale a quella per la categoria ma fa riferimento a una tabella diversa.

```

312     function setShop(id){
313         var category = document.getElementById(id);
314
315         //Elimino l'ultimo valore se ce ne sono più di 2
316         var length = category.length;
317         if(length >= 2){
318             category.remove(1);
319         }
320         //Creo una nuova option
321         var option = document.createElement('option');
322
323         //prendo il tag del negozio selezionato
324         var li = document.getElementsByClassName('selected')[document.getElementsByClassName('selected').length - 1];
325
326         //Inserisco all'interno della variabile il valore del paragrafo creato alla selezione della categoria
327         option.setAttribute('value', li.id);
328
329         //Inserisco l'option creata e la seleziono
330         category.appendChild(option);
331         category.getElementsByTagName('option')[1].setAttribute('selected', true);
332     }

```

Figura 13 funzione di modifica del select dei negozi "setShop"

La funzione che si attiva una volta selezionato un negozio però è leggermente diversa da quella della categoria, perché va a prendere il valore, non dal paragrafo con id "current" che viene creato in automatico da template ma al punto che ha la classe "select", ovvero il negozio selezionato, in modo che viene inserito nel select, non solo il nome scritto a schermo ma tutta la chiave codificata.

Una volta fatto, la funzione che inserisce i dati nel database va a inserire anche i dati nella tabella ponte che mostra quali negozi vedono quali prodotti, come mostrato nelle 2 immagini

che seguono, la prima mostra la funzione controller e la seconda il model, per il model ho creato una nuova classe che faccia riferimento alla tabella “vende”.

```

114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
    $shop = isset($_POST["shop"]) ? explode( delimiter: ".", $_POST["shop"] ) : null;
    $sName = $shop[0];
    $sAddress = $shop[1];
    $sCity = $shop[2];

    //Se i campi che devono comparire non sono vuoti
    if ($title != null && $price != null && $quantity != null) {
        //Inserisco i dati del prodotto
        $var = $product->insertProduct($category, $title, $price, $quantity, $newName);

        //Se i campi che devono comparire non sono vuoti
        if ($sName != null && $sAddress != null && $sCity != null) {
            //Inserisco i dati del negozio
            $var = $sell->insertData($sName, $sAddress, $sCity, $title, $price, $quantity);
        }
    }
}

```

Figura 14 inserimento prodotto con controllo negozio

```

21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
    public function insertData($shopName, $shopAddress, $shopCity, $prodName, $prodPrice, $prodQuantity){
        //Connetto al database
        $conn = $this->connection->sqlConnection();

        //se il campo è nuovo inserisco la nuova riga
        $quantity = 1;
        $sql = $conn->prepare("INSERT INTO vende (nome_prodotto, prezzo_prodotto, quantita_prodotto, nome_negozio, ind
        VALUES (:prodName, :prodPrice, :prodQuantity, :shopName, :shopAddress, :shopCity)");
        $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
        $sql->bindParam(':prodPrice', $prodPrice);
        $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
        $sql->bindParam(':shopName', $shopName, PDO::PARAM_STR);
        $sql->bindParam(':shopAddress', $shopAddress, PDO::PARAM_STR);
        $sql->bindParam(':shopCity', $shopCity, PDO::PARAM_STR);

        //Eseguo la query
        $sql->execute();

        //Se la query va a buon fine
        if($sql->rowCount() > 0) {
            $conn = null;
            return true;
        } else {
            $conn = null;
            return false;
        }
    }
}

```

Figura 15 funzione insertData classe SellModel

Una volta finito questo mi manca, oltre al completamento della pagina de venditore, come spiegato nella sezione “Punto della situazione rispetto alla pianificazione”, devo fare in modo che se lo stesso prodotto con stesso prezzo e quantità esiste in 2 negozi e uno viene comprato non venga modificato per entrambi i negozi ma venga creata una nuova istanza per il negozio a cui è stato comprato con la quantità minore.

Problemi riscontrati e soluzioni adottate

Ho messo a posto il problema riscontrato la scorsa lezione, come spiegato all'inizio della sezione precedente, e dopodiché non ho riscontrato particolari problemi.

Punto della situazione rispetto alla pianificazione

Ho completato l'inserimento nel carrello.

Ho completato l'inserimento dei prodotti, in modo che venga anche segnalato chi li vende.

Mi manca la modifica dei prodotti già inseriti.

Programma di massima per la prossima giornata di lavoro

Completare la modifica dei prodotti da parte dei venditori.

Diario di lavoro

Luogo	SAMT
Data	20.03.2019

Lavori svolti

La prima ora della mattina non ho lavorato perché il professor Valsangiacomo ci ha spiegato un po' come funziona il foglio riassuntivo del progetto da fare.

Dopodiché ho fatto un test sul sito, da part di un utente e ho notato che c'era un problema con l'inserimento nel carrello, questo era dato semplicemente dal fatto che per errore avevo eliminato i dati del costruttore della classe model "buy". Un altro problema di cui mi sono accorto è il fatto che nella schermata principale anche al login dell'utente non veniva mostrato il carrello, questo perché mancava il richiamo alla funzione che lo apre, quindi l'ho aggiunta dopo la funzione "getCategoriesHome" che viene richiamata in quella pagina.

Dopo aver messo a posto questi piccoli problemini ho continuato con la pagina del venditore, ho iniziato a lavorare sul modal, per fare in modo che all'aggiunta o alla modifica dei prodotti non vada su un'altra pagina ma appaia su quella principale un modal che permetta di lavorare. Per farlo ho preso un esempio di modal presente sul sito

https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_modal e l'ho riadattato inserendo i dati che deve contenere. Nell'immagine che segue è mostrato il modal per l'aggiunta dei prodotti.



```

32 <div id="addModal" class="modal" style="z-index: 101;">
33
34     <!-- Modal content -->
35     <div class="modal-content">
36         <span id="close">&times;</span>
37         <p>Il sito si prende il 10% del guadagno</p>
38         <form action="php echo URL ?dealer/insertProduct" method="POST" enctype="multipart/form-data">
39             <!-- ##### Single Product Details Area Start ##### -->
40             <section class="single_product_details_area d-flex align-items-center sBody">
41
42                 <!-- Single Product Thumb -->
43                 <div class="single_product_thumb clearfix">
44                     <label for="file-input">
45                         
46                     </label>
47                     <input type="file" id="file-input" name="imageQuestion" onchange="setImage(this)" accept="image/*" required>
48                 </div>
49
50                 <!-- Single Product Description -->
51                 <div class="single_product_desc clearfix">
52                     <h5>Inserimento prodotto</h5>
53
54                     <select name="category" id="category" onchange="setCategory(this.id)" required>
55                         <option disabled selected>Categoria</option>
56                     </select><br><br><br>
57                     <select name="shop" id="shop" onchange="setShop(this.id)" required>
58                         <option disabled selected>Negozio</option>
59                     </select><br><br><br>
60                     <input type="text" name="title" id="title" placeholder="Nome" onkeyup="validate(this.value, this.id, regLet)">
61                     <input type="number" name="prize" id="prize" placeholder="Prezzo" min="0" onkeyup="validate(this.value, this.id, regNum)">
62                     <input type="number" name="quantity" id="quantity" placeholder="Quantità" min="0" onkeyup="validate(this.value, this.id, regNum)">
63                 </div>
64                 <div class="formDiv">
65                     <input type="submit" id="insertProduct" value="INSERISCI" disabled/>
66                 </div>
67             </section>
68             <!-- ##### Single Product Details Area End ##### -->
69         </form>

```

Figura 1 modal aggiunta prodotti

La parte che non è mostrata all'interno dell'immagine è uguale a quella nella pagina esterna, e alla fine del form vengono semplicemente chiuso i 2 div. Questa parte va inserita all'interno

della pagina principale, per fare in modo che sia presente anche se non visibile, e bisogna aggiungere anche il css e il JavaScript presenti al link sopra citato.

Dopo c'è la parte di modifica dei dati che è praticamente uguale con la differenza nella denominazione degli id dei vari campi e i dati che non sono preimpostati, perché verranno preimpostati una volta aperto.

Dopo aver fatto questo ho iniziato a lavorare alla modifica dei dati dei prodotti da parte del venditore, per farlo ho concatenato tutti i dati utili come id del nome del prodotto, e al click della scritta passo "this" ovvero tutto il tag.

Purtroppo ancora non sono riuscito a finirlo.

Problemi riscontrati e soluzioni adottate

Non ho avuto particolari problemi

Punto della situazione rispetto alla pianificazione

Ho fatto in modo che la modifica e l'aggiunta dei prodotti venga fatto su un modal e non una pagina esterna.

Mi manca da completare la modifica dei prodotti già inseriti.

Programma di massima per la prossima giornata di lavoro

Completare la modifica dei prodotti da parte dei venditori.

Diario di lavoro

Luogo	SAMT
Data	25.03.2019

Lavori svolti

Inizialmente ho completato la modifica dei dati del prodotto che può fare il venditore sulla sua pagina, una volta cliccato sul nome del prodotto si apre un popup che permette di modificarne i dati ed eventualmente salvare. Questa pagina dava un problema nell'inserimento dei dati all'interno dei select, come spiegato nella sezione che segue questa.
Una volta fatto ho creato una funzione che elimina i dati dei select, e viene richiamata quando si chiude il popup.

```

181  function deleteCategoriesShop(ulC, ulS) {
182      //Faccio passare tutti i valori del select dall'ultimo e li cancello
183      // lasciando il primo
184      var ulClength = ulC.getElementsByTagName("li").length;
185      for (var i = ulClength-1; i > 0; i--) {
186          ulC.removeChild(ulC.childNodes[i]);
187      }
188
189      var ulSlength = ulS.getElementsByTagName("li").length;
190      for (var i = ulSlength-1; i > 0; i--) {
191          ulS.removeChild(ulS.childNodes[i]);
192      }
193  }

```

Figura 1 deleteCategoriesShop eliminazione dati select

La funzione riceve le 2 liste e in base a quanti dati ci sono dentro grazie ai child elimina tutti i dati tranne il primo, perché è quello che segnala il titolo del select e non viene inserito dinamicamente ma è scritto hardcoded essendo sempre uguale.

Qui sotto è mostrata la parte della funzione che va ad inserire tutti i dati nei campi del popup di modifica dei prodotti.

```

288     function detailsProduct(obj) {
289         var data = obj.id.split(".");
290
291         //Prendo il corpo del div in cui inserire i campi
292         var divbody = document.getElementById( elementId: 'modSBody' );
293
294         //Creo la riga i cui inserire i campi e gli metto imposto la classe
295         var div = document.createElement( tagName: "div" );
296         div.setAttribute( qualifiedName: "class", value: "single_product_thumb clearfix");
297
298         //Inserisco l'immagine del prodotto
299         var img = document.getElementById( elementId: "modImage" );
300         img.src = "http://samtinfo.ch/gestionevendita2018/" + data[0] + "." + data[1];
301
302         //Imposto le liste di categoria e negozi
303         getCategories( list: false, ul[2]);
304         setTimeout( handler: function(){
305             getShop(window.ul[3]);
306         }, timeout: 400);
307
308         //Prendo i div e inserisco i valori li imposto come selected
309         var divUl = document.getElementsByClassName( classNames: "nice-select");
310         divUl[2].childNodes[0].innerHTML = data[2];
311         divUl[3].childNodes[0].innerHTML = data[3];
312
313         //Inserisco il nome del prodotto
314         var name = document.getElementById( elementId: "modTitle");
315         name.setAttribute( qualifiedName: "value", data[4]);
316
317         //Inserisco il prezzo del prodotto
318         var price = document.getElementById( elementId: "modPrice");
319         price.setAttribute( qualifiedName: "value", data[5]);
320
321         //Inserisco il prezzo del prodotto
322         var quantity = document.getElementById( elementId: "modQuantity");
323         quantity.setAttribute( qualifiedName: "value", data[6]);
324
325         document.getElementById( elementId: 'modifyModal').style.display = "block";

```

Figura 2 detailsProduct, mostra dei dettagli del prodotto

Anche in questa funzione utilizzo il timeout (righe 304-306) per aprire le funzioni della categoria e dei negozi, dopodiché inserisco, grazie sempre ai child, i dati nei campi in modo da far capire quali sono quelli attuali.

La modifica non sono ancora riuscito ad ultimarla perché la correzione dei problemi mi ha preso diverso tempo.

Problemi riscontrati e soluzioni adottate

Un primo problema che ho avuto è stato con l'inserimento dei dati nella pagina di modifica del prodotto, per i select non riuscivo ad inserire le informazioni nella lista, ovvero, venivano inseriti i dati ma subito tolti. Dopo un po' ho capito il problema. Per modificare i select, vengono usate 2 funzioni, una che imposta le categorie e una i negozi, quella dei negozi è quella mostrata qui sotto.

```

141     function getShop(ul) {
142         xhttp.onreadystatechange = function () {
143             if (this.readyState == 4 && this.status == 200) {
144
145                 //Prendo i valori passati dal server e li metto in un array
146                 var obj = JSON.parse(xhttp.responseText);
147
148                 var divUl = document.getElementsByClassName( classNames: "nice-select");
149                 divUl[1].setAttribute( qualifiedName: "style", value: "z-index:0;");
150
151                 //Inserisco tutte le categorie in optio e poi nel select.
152                 for(var i = 0; i < obj.length; i++){
153                     var li = document.createElement( tagName: "li");
154                     li.setAttribute( qualifiedName: "class", value: "option selected focus");
155                     li.setAttribute( qualifiedName: "style", value: "z-index:3;");
156                     li.setAttribute( qualifiedName: "data-value", value: obj[i]["nome"] +"."+ obj[i]["indirizzo"] +"."+ obj[i]["citta"]);
157                     li.setAttribute( qualifiedName: "id", value: obj[i]["nome"] +"."+ obj[i]["indirizzo"] +"."+ obj[i]["citta"]);
158                     li.appendChild(document.createTextNode(obj[i]["nome"]));
159                     ul.appendChild(li);
160                 }
161             }
162         }
163         xhttp.open( method: "POST", url: "/gestionevendita2018/dealer/getShopsByDealer", async: true);
164         xhttp.send();
165     }

```

Figura 3 funzione getShop per riempire il select di negozi

La funzione riceve tramite argomento il campo che deve riempire, poi con i dati presi dalla richiesta al database lo riempie.

Il problema era che la funzione veniva richiamata sia all'apertura del popup di modifica, per il campo in quella sezione, che alla fine della funzione che prende i dati delle categorie, di conseguenza andava in conflitto e riempiva la prima volta ma poi veniva richiamata la funzione ed essendo che i vari tag “ul” hanno gli stessi nomi svuotava quello riempito in precedenza e riempiva il nuovo. Per risolvere il problema mi è bastato inserire il richiamo della funzione per il select dell'inserimento, assieme a quello delle categorie, come mostrato qui sotto.

```

19         $('#switch').click(function() {
20             modal.style.display = "block";
21             getCategorys( list: false);
22             getShop(ul[1]);
23         });

```

Figura 4 apertura inserimento prodotti

Dopo aver fatto questa modifica però è tornato il problema di Ajax, che è troppo veloce e di conseguenza funziona solo l'ultimo richiamo, ma sono riuscito a fare in modo che prima di far partire la seconda funzione attenda un po', con il metodo “setTimeout” mostrato nell'immagine sotto, che aspetta 500 millisecondi e dopo esegue ciò che c'è al suo interno.

```

19         $('#switch').click(function() {
20             modal.style.display = "block";
21             getCategorys( list: false, ul[0]);
22             setTimeout( handler: function(){
23                 getShop(ul[1]);
24             }, timeout: 500);
25         });

```

Figura 5 apertura inserimento prodotti

L'ultimo problema che ho dovuto risolvere è stata la chiusura del popup della modifica. Questo problema era dato dal fatto che in JQuery andavo a chiudere solo il modal dell'aggiunta quindi mi è bastato aggiungere il secondo modal come mostra l'immagine sotto e tutto funziona correttamente.

```
15 $(function() {
16     // Get the modal
17     var addmodal = document.getElementById( elementId: 'addModal' );
18     var modifyModal = document.getElementById( elementId: 'modifyModal' );
19
20     $('#switch').click(function() {
21         addmodal.style.display = "block";
22         getCategorys( list: false, ul[0] );
23         setTimeout( handler: function(){
24             getShop(ul[1]);
25         }, timeout: 500 );
26     });
27
28     // When the user clicks on <span> (x), close the modal
29     $('.close').click(function() {
30         addmodal.style.display = "none";
31         deleteCategoriesShop(ul[0], ul[1]);
32         modifyModal.style.display = "none";
33         deleteCategoriesShop(ul[2], ul[3]);
34     });
35
36     // When the user clicks anywhere outside of the modal, close it
37     window.onclick = function(event) {
38
39         if (event.target == addmodal || event.target == modifyModal) {
40             addmodal.style.display = "none";
41             deleteCategoriesShop(ul[0], ul[1]);
42             modifyModal.style.display = "none";
43             deleteCategoriesShop(ul[2], ul[3]);
44         }
45     };
46 })
```

Figura 6 JQuery apertura e chiusura modal

Punto della situazione rispetto alla pianificazione

Ho completato la parte del venditore.

Devo ancora gestire il caso in cui 2 negozi facciano riferimento allo stesso prodotto e in uno dei 2 diminuisca la quantità.

Sono ancora in ritardo di una settimana.

Programma di massima per la prossima giornata di lavoro

Iniziare la pagina dell'amministratore.

Diario di lavoro

Luogo	SAMT
Data	26.03.2019

Lavori svolti

Questa mattina ho fatto un colloquio con il docente responsabile e abbiamo riguardato il sito e visto le parti fondamentali ancora mancanti, trovabili nel file “annotazioni” nella cartella “informazioni_progetto” della cartella in GitHub del progetto, a questo link https://github.com/giairomauro/ProgettoVenditaPiccolaNegoziante/blob/master/Informazioni_progetto/Annotazioni.docx.

Una volta finito ho completato la parte di modifica dei prodotti da parte dei vendori, semplicemente creando la funzione di modifica nel model delle tabelle “prodotto” e “vende” e facendogli arrivare i dati corretti per modificare i dati nella tabella, come mostrano le immagini che seguono.

```

140 public function modifyProduct()
141 {
142     //Se la funzione è richiamata come POST
143     if ($_SERVER["REQUEST_METHOD"] == "POST") {
144
145         //Prendo la classe model
146         require_once 'application/models/product.php';
147         $product = new ProductModel();
148         require_once 'application/models/sell.php';
149         $sell = new SellModel();
150
151         // Connessione all'ftp
152         $connFTP = ftp_connect( host: "efof.ftp.infomaniak.com");
153         $login = ftp_login($connFTP, username: "efof_gestvend", password: "GestVend_Admin_2018");
154
155         //Prendo le variabili passate dal post.
156         $name = null;
157         $image = isset($_FILES['imageQuestion'])? $_FILES['imageQuestion'] : null; //Prendo il nome del file
158         if($image['name'] != "") {
159             $name = $image['name'];
160
161             //Prendo il percorso temporaneo del file e gli cambio nome
162             $tmpName = $image['tmp_name'];
163             $newName = 'application/img/' . $name;
164             rename($tmpName, $newName);
165
166             //Imposto i permessi per il file
167             ftp_chmod($connFTP, mode: 0664, $newName);
168         }
169
170         //Prendo le variabili passate dal POST
171         $category = isset($_POST["category"])? $_POST["category"] : null;
172         $title = isset($_POST["title"])? $_POST["title"] : null;
173         $price = isset($_POST["prize"])? $_POST["prize"] : null;
174         $quantity = isset($_POST["quantity"])? $_POST["quantity"] : null;
175         $oldData = isset($_POST["oldData"])? explode( delimiter: ".", $_POST["oldData"]) : null;

```

Figura 1 funzione modifyProduct di modifica

```

177 //Prendo i dati del negozio
178 $shop = isset($_POST["modShop"])? explode( delimiter: ".", $_POST["modShop"]) : null;
179
180 print_r($shop);
181 $sName = $shop[0];
182 $sAddress = $shop[1];
183 $sCity = $shop[2];
184
185 //Se i campi che devono comparire non sono vuoti
186 if ($title != null && $price != null && $quantity != null) {
187     //Inserisco i dati del prodotto
188     $product->modifyProduct($category, $title, $price, $quantity, $newname, $oldData);
189
190     //Controllo se è cambiato il negozio
191     if($sName != $oldData[4]){
192         $sell->modifyData($sName, $sAddress, $sCity, $title, $price, $quantity, $oldData);
193     }
194 }
195
196 header( string: "location: ". URL ."dealer/home");
197 //Altrimenti
198 }else{
199     //Ritorno alla pagina precedente
200     header( string: javascript://history.back() );
201 }
202 }

```

Figura 2 funzione modifyProduct 2

```

57 public function modifyData($shopName, $shopAddress, $shopCity, $prodName, $prodPrice, $prodQuantity, $oldData){
58     //Connetto al database
59     $conn = $this->connection->sqlConnection();
60
61     //se il campo è nuovo inserisco la nuova riga
62     $quantity = 1;
63     $sql = $conn->prepare("UPDATE vendita SET nome_negozio = :shopName, indirizzo_negozio = :shopAddress, citta_negozio = :shopCity
64     WHERE nome_prodotto = :oldProdName AND prezzo_prodotto = :oldProdPrice AND quantita_prodotto = :oldProdQuantity AND
65     nome_negozio = :oldShopName AND indirizzo_negozio = :oldShopAddress AND citta_negozio = :oldShopCity");
66     $sql->bindParam(':shopName', $shopName, PDO::PARAM_STR);
67     $sql->bindParam(':shopAddress', $shopAddress, PDO::PARAM_STR);
68     $sql->bindParam(':shopCity', $shopCity, PDO::PARAM_STR);
69     $sql->bindParam(':oldProdName', $oldData[7], PDO::PARAM_STR);
70     $sql->bindParam(':oldProdPrice', $oldData[8]);
71     $sql->bindParam(':oldProdQuantity', $oldData[9], PDO::PARAM_INT);
72     $sql->bindParam(':oldShopName', $oldData[4], PDO::PARAM_STR);
73     $sql->bindParam(':oldShopAddress', $oldData[5], PDO::PARAM_STR);
74     $sql->bindParam(':oldShopCity', $oldData[6], PDO::PARAM_STR);
75
76     //Eseguo la query
77     $sql->execute();
78
79     //Se la query va a buon fine
80     if($sql->rowCount() > 0){
81         $conn = null;
82         return true;
83     } else {
84         $conn = null;
85         return false;
86     }
87 }

```

Figura 3 funzione modifyData della tabella vendita

Queste funzioni grazie a dei controlli modificano solo i dati che devono, non vanno a modificare dati che restano uguali, oppure se sono uguali e vengono riscritti MySQL sen accorge lo lascia come sono.

Una volta finito questo ho iniziato a mettere a posto la pagina dei clienti.

La prima cosa che ho fatto è stata fare in modo che di fianco ai prodotti nel carrello venga mostrato il prezzo totale, per farlo ho dato un id allo span che avrebbe contenuto il prezzo, creato una variabile globale nel file JavaScript e fatto in modo che aumenti ogni volta che viene aggiunto un nuovo prodotto nel carrello. Le immagini qui sotto mostrano i passaggi spiegati sopra in ordine.

```

22 <h2>Sommario</h2>
23 <ul class="summary-table">
24   <li><span>Totale:</span> <span id="totPrice"></span></li>
25 </ul>

```

Figura 4 span prezzo totale carrello

```

10 //Variabile del prezzo totale
11 var totPrice = 0;

```

Figura 5 variabile prezzo totale

```

411 //Modifico la variabile del numero di oggetti e del prezzo
412 cartObjects += parseInt(obj['quantita_richiesta']);
413 totPrice += parseInt( obj['quantita_richiesta'] * obj['prezzo_prodotto']);

```

Figura 6 modifica variabile prezzo totale

```

475 //Modifico la variabile del numero di oggetti
476 cartObjects++;
477 totPrice += parseInt(obj['prezzo_prodotto']);

478 //Modifico la quantità del prodotto richiesto
479 document.getElementById( elementId: "quantity"+ obj['nome_prodotto']).innerHTML = "";
480 document.getElementById( elementId: "quantity"+ obj['nome_prodotto']).appendChild(
481   document.createTextNode( data: "quantità: " + obj['quantita_richiesta']));
482 }

483 }

484

485 //Scrivo il numero di oggetti
486 for(var i = 0; i < document.getElementsByClassName( classNames: "cartObjects").length; i++) {
487   document.getElementsByClassName( classNames: "cartObjects")[i].innerHTML = "";
488   document.getElementsByClassName( classNames: "cartObjects")[i].appendChild(document.createTextNode(cartObjects))
489 }

490

491 //Scrivo il prezzo totale nel sommario
492 document.getElementById( elementId: 'totPrice').innerHTML = totPrice +"Fr.";

```

Figura 7 modifica e inserimento prezzo totale

La modifica del prezzo viene fatta 2 volte, la prima per la prima volta che il prodotto viene inserito nel carrello, quindi somma il prezzo moltiplicato alla quantità richiesta dal cliente, mentre la seconda aggiunge soltanto il prezzo perché lo fa quando un prodotto che c'è già viene aggiunto una volta.

L'immagine qui sotto mostra il risultato finale nel carrello con tutti i dati, anche se ancora non si vedono tutti i dati di alcuni prodotti.

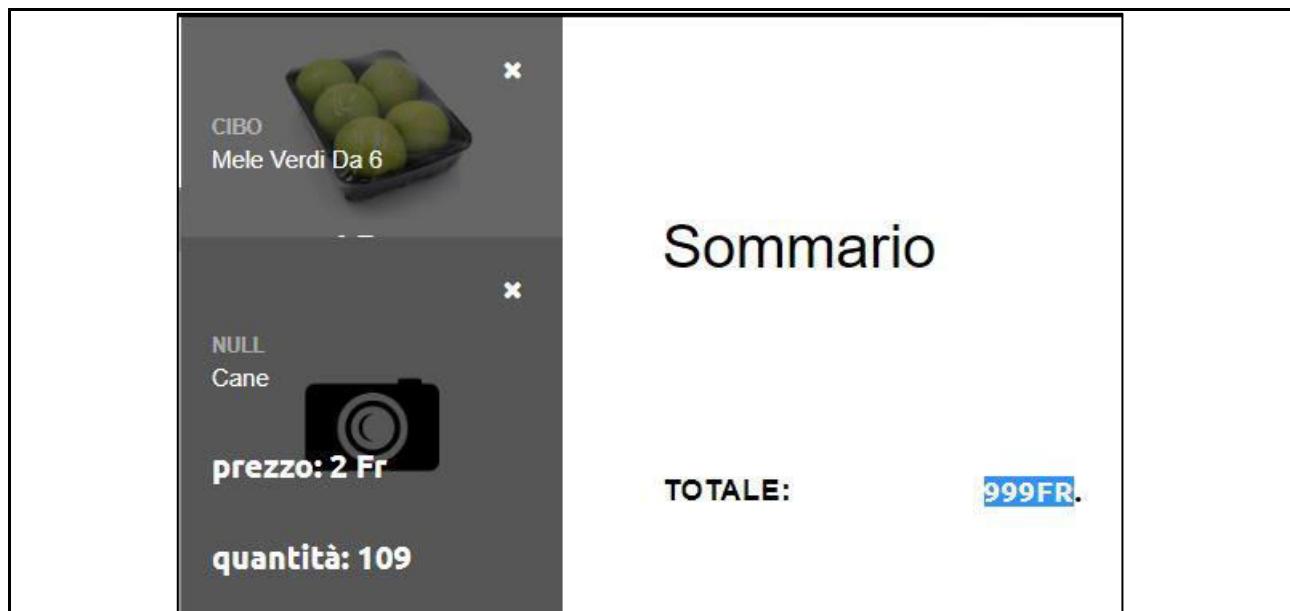


Figura 8 mostra prezzo totale carrello

Dopo questo non sono riuscito più ad andare molto avanti.

Problemi riscontrati e soluzioni adottate

La parte di modifica mi ha dato del filo da torcere per via dei dati da prendere, perché deve prendere dei dati già scritti anche se non vengono modificati, di conseguenza ho dovuto aggiungere diversi controlli e diverse informazioni, tra cui togliere ai select e all'immagine il fatto che sia richiesto, perché se lo era dava errore se non veniva modificato.

Oltre a questo ho anche dovuto implementare la modifica della tabella "vende" nel caso in cui venga modificato il negozio che vende il determinato prodotto.

Punto della situazione rispetto alla pianificazione

Devo completare la pagina dei clienti.

Devo ancora gestire il caso in cui 2 negozi facciano riferimento allo stesso prodotto e in uno dei 2 diminuisca la quantità.

Sono molto in ritardo.

Programma di massima per la prossima giornata di lavoro

Completare pagina dei clienti.

Diario di lavoro

Luogo	SAMT
Data	01.04.2019

Lavori svolti

Questa mattina ho completato in modo più completo l'inserimento dei dati nel modal che mostra i dettagli del prodotto.

Per farlo ho creato una funzione a parte che va a eliminare i dati del prodotto dal modal di dettaglio, e viene richiamata appena si chiude, in qualsiasi modo, sia premendo sulla "X" sia cliccando fuori dal modal stesso.

La funzione è quella mostrata qui sotto, che prende i dati e li elimina.

```

40     function deleteDetails(){
41
42         //Prendo tutti gli elementi dello shop li elimino
43         var shops = document.getElementsByClassName( classNames: "shop" );
44         for (var i = 0; i < shops.length; i++){
45             shops[i].parentNode.removeChild(shops[i]);
46         }
47
48         //Svuoto la categoria del prodotto
49         document.getElementById( elementId: "category" ).innerHTML = "";
50
51         //Svuoto la categoria del prodotto
52         document.getElementById( elementId: "title" ).innerHTML = "";
53
54         //Svuoto la categoria del prodotto
55         document.getElementById( elementId: "price" ).innerHTML = "";
56
57         //Svuoto la categoria del prodotto
58         document.getElementById( elementId: "quantity" ).innerHTML = "";
59     }

```

Figura 1 funzione deleteDetails per eliminare i dettagli dal modal

Dopo questo ho lavorato al select che mostra i luoghi di ritiro nel carrello, per poterli scegliere e selezionare da dove andare a ritirare i prodotti prima di fare il check out.

Per farlo ho inserito il select, come nelle pagine dei vedoriti, e dopodiché una funzione JavaScript che prende dal database i dati dei luoghi e li inserisce all'interno, come mostrato dall'immagine che segue.

```

454     function setPickupShop() {
455         xhttp.onreadystatechange = function () {
456             if (this.readyState === 4 && this.status === 200) {
457
458                 //Prendo i valori passati dal server e li metto in un array
459                 var obj = JSON.parse(xhttp.responseText);
460
461                 //Prendo la lista in cui inserire i dati
462                 var ul = document.getElementsByClassName( classNames: "list") [0];
463
464                 //Inserisco tutte le categorie in option e poi nel select.
465                 for(var i = 0; i < obj.length; i++){
466                     var li = document.createElement( tagName: "li");
467                     li.setAttribute( qualifiedName: "class", value: "option focus");
468                     li.setAttribute( qualifiedName: "style", value: "z-index:0;");
469                     li.setAttribute( qualifiedName: "data-value", obj[i][ "nome"]);
470                     li.appendChild(document.createTextNode(obj[i][ "nome"]));
471                     ul.appendChild(li);
472                 }
473             }
474         }
475         xhttp.open( method: "POST", url: "/gestionevendita2018/customer/getPickupShop", async: true);
476         xhttp.send();
477     }

```

Figura 2 JS setPickupShop inserisce i dati del luogo di ritiro

La funzione controller è molto simile ad altre già create, perché si occupa solo di prendere i dati richiamando la funzione apposita nel model.

```

115     public function getPickupShop()
116     {
117         //Se la sessione è aperta apro le pagine altrimenti no
118         if(isset($_SESSION[ 'customer' ])) {
119
120             //Prendo la classe model
121             require_once 'application/models/pickupshop.php';
122             $pickupShop = new PickupShopModel();
123
124             //Array che conterrà tutti i prodotti
125             $dataArray = array();
126
127             //Prendo i dati del carrello
128             $shops = $pickupShop->getData();
129
130             //Per ogni dato inserisco il prodotto relativo nell'array
131             foreach ($shops as $value)
132                 array_push( &array: $dataArray, $value);
133
134             //Stampo con json l'array con i prodotti
135             header( string: 'Content-Type: application/json');
136             echo json_encode($dataArray);
137             //Altrimenti
138         } else{
139
140     }
141 }

```

Figura 3 controller getPickupShop richiede i dati del luogo di ritiro e li da

Per prendere i dati dal database, ho dovuto creare una nuova classe, perché faccio riferimento una tabella, "luogo_ritiro", a cui non avevo mai acceduto prima, di conseguenza mi serviva una nuova classe, come al solito ho istanziato il costruttore che richiama la classe "connection" come tutti gli altri model.

La funzione che prende i dati dal database e li ritorna è la seguente.

```
17     public function getData()
18     {
19         //Connetto al database
20         $conn = $this->connection->sqlConnection();
21
22         //Prendo i dati dell'utente in base alla mail
23         $sql = $conn->prepare("SELECT * FROM luogo_ritiro");
24
25         //Eseguo la query
26         $sql->execute();
27
28         $dataArray = array();
29         //Se ci sono dei valori
30         if ($sql->rowCount() > 1) {
31
32             // Ciclo tutti i valori
33             while ($row = $sql->fetch()) {
34
35                 array_push( &array: $dataArray, $row);
36             }
37         } else if ($sql->rowCount() == 1) {
38             array_push( &array: $dataArray, $sql->fetch());
39         }
40         $conn = null;
41         return $dataArray;
42     }
```

Figura 4 model getData prende i dati del luogo di ritiro dal database

Questo è tutto quello che ho fatto questa mattina di progetto.

Problemi riscontrati e soluzioni adottate

Non ho avuto particolari problemi.

Punto della situazione rispetto alla pianificazione

Devo completare la pagina del carrello, ovvero il checkout e la grandezza delle immagini.

Devo ancora gestire il caso in cui 2 negozi facciano riferimento allo stesso prodotto e in uno dei 2 diminuisca la quantità.

Programma di massima per la prossima giornata di lavoro

Completare pagina dei clienti, implementando il checkout.

Diario di lavoro

Luogo	SAMT
Data	02.04.2019

Lavori svolti

Oggi ho iniziato a lavorare all'aggiunta, la diminuzione e l'eliminazione dei prodotti direttamente dal carrello.

Per prima cosa ho dovuto aggiungere dei simboli per farlo all'interno del carrello, quindi nella funzione che va ad inserirci i dati ho aggiunto 2 parti dove veniva inserita la x che elimina completamente il prodotto.

```

577 var aProd = document.createElement( tagName: "a");
578 aProd.setAttribute( qualifiedName: "name", value: obj['nome_prodotto'] +".."+ obj['prezzo_prodotto'] +".."+ obj['quantita_prodotto']);
579 aProd.setAttribute( qualifiedName: "onclick", value: "addToCart(this)");
580 var span = document.createElement( tagName: "span");
581 span.setAttribute( qualifiedName: "class", value: "product-remove mr-5");
582 var i = document.createElement( tagName: "i");
583 i.setAttribute( qualifiedName: "class", value: "fa fa-plus");
584 i.setAttribute( qualifiedName: "aria-hidden", value: "ture");
585 span.appendChild(i);
586 aProd.appendChild(span);
587 div.appendChild(aProd);

588 //Inserisco il bottone di diminuzione
589 var aProd = document.createElement( tagName: "a");
590 aProd.setAttribute( qualifiedName: "name", value: obj['nome_prodotto'] +".."+ obj['prezzo_prodotto'] +".."+ obj['quantita_prodotto']);
591 aProd.setAttribute( qualifiedName: "onclick", value: "delOneFromCart(this)");
592 var span = document.createElement( tagName: "span");
593 span.setAttribute( qualifiedName: "class", value: "product-remove mr-4");
594 var i = document.createElement( tagName: "i");
595 i.setAttribute( qualifiedName: "class", value: "fa fa-minus");
596 i.setAttribute( qualifiedName: "aria-hidden", value: "ture");
597 span.appendChild(i);
598 aProd.appendChild(span);
599 div.appendChild(aProd);

600 //Inserisco il bottone di eliminazione
601 var aProd = document.createElement( tagName: "a");
602 aProd.setAttribute( qualifiedName: "name", value: obj['nome_prodotto'] +".."+ obj['prezzo_prodotto'] +".."+ obj['quantita_prodotto']);
603 aProd.setAttribute( qualifiedName: "onclick", value: "delFromCart(this)");
604 var span = document.createElement( tagName: "span");
605 span.setAttribute( qualifiedName: "class", value: "product-remove");
606 var i = document.createElement( tagName: "i");
607 i.setAttribute( qualifiedName: "class", value: "fa fa-close");
608 i.setAttribute( qualifiedName: "aria-hidden", value: "ture");
609 span.appendChild(i);
610 aProd.appendChild(span);
611 div.appendChild(aProd);

```

Figura 1 inserimento bottoni aggiunta diminuzione eliminazione prodotti

Tutti i contenitori delle immagini dei prodotti sono all'interno di un link che ha come nome le chiavi, concatenate, del prodotto in questione, e quando vengono cliccati aprono ognuno il metodo apposito per ciò che deve fare.

Il metodo per aggiungere il prodotto è già creato, ho solo dovuto fare quelli dell'eliminazione di un pezzo del prodotto e del prodotto completo.

Le funzioni implementate in JavaScript sono le seguenti.

```

415     function delOneFromCart(obj) {
416
417         //Divido la chiave composta nei valori
418         var keys = decode(obj.name);
419
420         //Se il prodotto è disponibile
421         if(keys[2] > 0) {
422             xhttp.onreadystatechange = function () {
423                 if (this.readyState === 4 && this.status === 200) {
424
425                     //Diminuisco di uno il valore delle quantità a tutti i nomi richiesti
426                     var name = encode(keys[0], keys[1], keys[2]);
427                     var x = document.getElementsByName(name);
428                     keys[2]++;
429                     for(var i = x.length - 1; i >= 0; i--) {
430                         x[i].name = encode(keys[0], keys[1], keys[2]);
431                     }
432
433                     //Modifico la variabile del numero di oggetti
434                     cartObjects--;
435                     totPrice -= parseInt(keys[1]);
436                     String(keys[1]);
437
438                     //Prendo i valori che vanno mostrati nel carrello
439                     getCartProduct(keys[0], keys[1], keys[2]);
440                 }
441             }
442             xhttp.open( method: "POST", url: "/gestionevendita2018/customer/modifyCart", async: true);
443             xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
444             xhttp.send( body: "&name=" + keys[0] + "&price=" + keys[1] + "&quantity=" + keys[2] + "&action=delOne")
445         }
446     }

```

Figura 2 delOneFromCart elimina un pezzo del prodotto

Le funzioni richiamano tutte lo stesso controller, h è lo stesso della funzione di aggiunta dei prodotti nel carrello. La funzione dell'eliminazione completa del prodotto non è ancora completata, il prodotto viene cancellato, ma non viene mostrato direttamente l'eliminazione dal carrello, bisogna ricaricare la pagina.

Il controller della funzione ha subito a sua volta dei cambiamenti, per essere utilizzata da tutte e 3 le funzioni, prendendo l'ultimo elemento inviato con il post, che raffigura l'azione da svolgere.

```

23     public function modifyCart()
24     {
25         //Se la sessione è aperta apro le pagine altrimenti no
26         if(isset($_SESSION['customer'])) {
27
28             //Prendo la classe model
29             require_once 'application/models/buy.php';
30             $buy = new BuyModel();
31
32             //Prendo le variabili passate dal POST
33             $name = isset($_POST["name"])? $_POST["name"] : null;
34             $price = isset($_POST["price"])? $_POST["price"] : null;
35             $quantity = isset($_POST["quantity"])? $_POST["quantity"] : null;
36             $action = isset($_POST["action"])? $_POST["action"] : null;
37
38             //Se entrambi i campi non sono vuoti
39             if ($name != null & $price != null & $quantity != null & $action != null) {
40                 $buy->modifyData($name, $price, $quantity, $_SESSION['customer'], $action)
41             }
42             //Altrimenti
43         }else{
44
45     }
46 }

```

Figura 3 modifyCart modifica il carrello aggiungendo diminuendo o eliminando i prodotti

L'unica differenza della funzione, oltre al nome che cambia da “addToCart” a “modifyCart” è che prende il valore dell'azione e lo passa al model che lavora con il database.

```

54     switch ($action) {
55         case "add":
56             $quantity = $this->addQuantity($buyCount);
57
58             //Imposto la nuova quantità del prodotto
59             require_once 'application/models/product.php';
60             $product = new ProductModel();
61             $product->setQuantity($prodName, $prodPrice, $prodQuantity, $prodQuantity - 1);
62             $prodQuantity--;
63             break;
64         case "delOne":
65             $quantity = $this->removeQuantity($buyCount);
66
67             //Imposto la nuova quantità del prodotto
68             require_once 'application/models/product.php';
69             $product = new ProductModel();
70             $product->setQuantity($prodName, $prodPrice, $prodQuantity, $prodQuantity + 1);
71             $prodQuantity++;
72             break;
73         default:
74
75             //Imposto la nuova quantità del prodotto
76             require_once 'application/models/product.php';
77             $product = new ProductModel();
78             $newQuantity = isset($buyCount[0]['quantita_richiesta'])? $prodQuantity + $buyCount[0]['quantita_richiesta']
79             : $prodQuantity + $buyCount['quantita_richiesta'];
80             $product->setQuantity($prodName, $prodPrice, $prodQuantity, $newQuantity);
81     }

```

Figura 4 switch controllo action

Nel model dopo il controllo del numero di oggetti c'è uno switch che controlla l'azione e a dipendenza di qual è svolge un'azione diversa, nei primi 2 casi vengono richiamate funzioni che modificano la quantità, mentre nell'ultimo caso, ovvero quello in cui viene eliminato completamente il prodotto, viene impostata una nuova quantità che è la vecchia più tutta la quantità richiesta.

Le funzioni richiamate esternamente sono le seguenti.

```

13     public function addQuantity($buyCount) {
14
15         //Setto la nuova quantità e modifco il campo nella tabella
16         if(is_array($buyCount[0]))
17             $quantity = $buyCount[0]['quantita_richiesta'] + 1;
18         else
19             $quantity = $buyCount['quantita_richiesta'] + 1;
20
21         return $quantity;
22     }
23
24     public function removeQuantity($buyCount) {
25
26         //Setto la nuova quantità e modifco il campo nella tabella
27         if(is_array($buyCount[0]))
28             $quantity = $buyCount[0]['quantita_richiesta'] - 1;
29         else
30             $quantity = $buyCount['quantita_richiesta'] - 1;
31
32         return $quantity;
33     }

```

Figura 5 funzioni modifica quantità

La prima funzione si occupa di aumentare la quantità richiesta da inserire e la seconda di diminuirla.

Ho utilizzato questo metodo perché altrimenti avrei dovuto creare 2 funzioni con lo stesso codice per sql ma con la differenza minima mostrata nelle funzioni sopra, quindi è molto più efficiente fare in questo modo.

L'ultimo modifica che ho fatto è il controllo se la quantità richiesta viene portata a 0, perché in quel caso non bisogna semplicemente riscriverla ma eliminare direttamente l'istanza dal database, perché risulterebbe un dato inutile essendo inutilizzato.

Di conseguenza l'ho gestito come mostrato.

```

83     if($quantity == 0){
84         $sql = $conn->prepare("DELETE FROM compra WHERE nome_prodotto LIKE :prodName
85                               AND prezzo_prodotto LIKE :prodPrice AND
86                               quantita_prodotto LIKE :prodQuantity AND email_cliente LIKE :custMail");
87         $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
88         $sql->bindParam(':prodPrice', $prodPrice);
89         $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
90         $sql->bindParam(':custMail', $custMail, PDO::PARAM_STR);
91     }else{
92         $sql = $conn->prepare("UPDATE compra SET quantita_richiesta = :quantity, data = now() WHERE nome_prodotto LIKE :prodNa
93                               AND prezzo_prodotto LIKE :prodPrice AND
94                               quantita_prodotto LIKE :prodQuantity AND email_cliente LIKE :custMail");
95         $sql->bindParam(':quantity', $quantity, PDO::PARAM_INT);
96         $sql->bindParam(':prodName', $prodName, PDO::PARAM_STR);
97         $sql->bindParam(':prodPrice', $prodPrice);
98         $sql->bindParam(':prodQuantity', $prodQuantity, PDO::PARAM_INT);
99         $sql->bindParam(':custMail', $custMail, PDO::PARAM_STR);
100    }
101 }

```

Figura 6 scelta modifica o eliminazione prodotto dal carrello

Come si può vedere c'è un controllo che nel caso in cui la quantità sia pari a 0 viene eliminato altrimenti modificato il dato.

Problemi riscontrati e soluzioni adottate

Ho avuto alcuni problemi dopo l'inserimento dell'eliminazione dei dati dalla tabella "compra" perché ho dovuto modificare diverse parti di codice per riadattarla, perché a quel punto avevo anche cambiato il nome della funzione.

Un altro problema l'ho riscontrato con i bottoni, perché inizialmente avevo strutturato nel modo errato l'inserimento da parte di JavaScript di essi, ma come mostrato nell'immagine della sezione precedente è corretto. In più c'era il problema che quando premo uno dei bottoni veniva modificato solo il nome di quello, e non andava bene essendo che nel nome dei bottoni c'è la chiave primaria del prodotto, quindi ho dovuto fare in modo che quando ne premo uno si modificavano tutti inserendo la chiave con i dati corretti, come mostrato dalla figura sotto.

```
387         var name = encode(keys[0], keys[1], keys[2]);  
388         var x = document.getElementsByName(name);  
389         keys[2]--;  
390         for(var i = x.length - 1; i >= 0; i--) {  
391             x[i].name = encode(keys[0], keys[1], keys[2]);  
392         }
```

Figura 7 modifica chiavi prodotto

Il lavoro che viene eseguito è il seguente, prendo tutti i campi con il nome vecchio, modifico la quantità della chiave, dopodiché faccio passare tutti i dati dall'ultimo, essendo che dopo la modifica spariscono dall'array essendo cambiato il nome, e poi cambio il nome di quel tag. Ho avuto un problema con questa parte che mi ha reso diverso tempo perché inizialmente facevo passare i valori dal primo ma giustamente una volta cambiato il nome l'array diminuiva e quindi andava in conflitto, ma in questo modo funziona correttamente.

Punto della situazione rispetto alla pianificazione

Devo completare la modifica della quantità dei prodotti, non è ancora completa al 100%, dopodiché inizierò il checkout.

Devo ancora gestire il caso in cui 2 negozi facciano riferimento allo stesso prodotto e in uno dei 2 diminuisca la quantità.

Programma di massima per la prossima giornata di lavoro

Completare modifica quantità prodotti nel carrello, implementando il checkout.

Diario di lavoro

Luogo	SAMT
Data	03.04.2019

Lavori svolti

Questa mattina l'ho usata quasi interamente a completare la modifica dei dati nel carrello, ovvero aggiungere o togliere prodotti direttamente da lì.

Per farlo ho dovuto modificare diverse classi, tra cui anche quelle modificate ieri, e altre più vecchie, per poter implementare il funzionamento che mi serve.

La prima modifica che ho fatto è stata alla classe già cambiata ieri.

La modifica che ho applicato sta nel dividere lo switch che controlla l'azione che l'utente vuole eseguire, come mostrano le immagini sotto.

```

56     switch ($action) {
57         case "add":
58             $quantity = $this->addQuantity($buyCount);
59             break;
60         case "delOne":
61             $quantity = $this->removeQuantity($buyCount);
62             break;
63     }

```

Figura 1 modifica quantità richiesta

```

97     switch ($action) {
98         case "add":
99
100            //Imposto la nuova quantità del prodotto
101            $product->setQuantity($prodName, $prodPrice, $prodQuantity, $prodQuantity - 1);
102            $prodQuantity--;
103            break;
104        case "delOne":
105
106            //Imposto la nuova quantità del prodotto
107            $product->setQuantity($prodName, $prodPrice, $prodQuantity, $prodQuantity + 1);
108            $prodQuantity++;
109            break;
110        default:
111
112            //Imposto la nuova quantità del prodotto
113            $newQuantity = isset($buyCount[0]['quantita_richiesta']) ? $prodQuantity + $buyCount[0]['quantita_richiesta']
114                : $prodQuantity + $buyCount['quantita_richiesta'];
115            $product->setQuantity($prodName, $prodPrice, $prodQuantity, $newQuantity);
116    }

```

Figura 2 modifica quantità prodotto

La modifica è stata applicata perché la modifica della quantità richiesta va fatta prima, ma quella del prodotto non può essere modificata prima, altrimenti la query viene fata sul dato sbagliato, quindi le righe mostrate nella seconda immagine vanno inserite appena prima della riga “\$sql->execute();” in modo da far applicare tutto nel modo corretto.

Un'altra modifica che bisogna attuare per far funzionare il tutto è stata all'interno della funzione “getProduct” nella classe model “ProductModel” come mostrato sotto.

```
53     if(count($buy->getDataByProduct($name, $price, $quantity)) > 0) {
54         //Prendo i dati dell'utente in base alla mail
55         $sql = $conn->prepare("SELECT * from prodotto p
56                               right outer join compra c on p.nome = c.nome_prodotto and p.prezzo = c.prezzo_prodotto and p.quantita = c.quantita_prodotto
57                               left outer join vende v on v.nome_prodotto = p.nome
58                               left outer join negozio n on n.nome = v.nome_negozi and n.indirizzo = v.indirizzo_negozi and n.citta = v.citta_negozi
59                               WHERE p.nome LIKE :name AND p.prezzo LIKE :prezzo AND p.quantita LIKE :quantita");
60         $sql->bindParam(':name', $name, PDO::PARAM_STR);
61         $sql->bindParam(':prezzo', $price, PDO::PARAM_INT);
62         $sql->bindParam(':quantita', $quantity, PDO::PARAM_STR);
63     } else {
64         //Prendo i dati dell'utente in base alla mail
65         $sql = $conn->prepare("SELECT * from prodotto p
66                               left outer join vende v on v.nome_prodotto = p.nome
67                               left outer join negozio n on n.nome = v.nome_negozi and n.indirizzo = v.indirizzo_negozi and n.citta = v.citta_negozi
68                               WHERE p.nome LIKE :name AND p.prezzo LIKE :prezzo AND p.quantita LIKE :quantita");
69         $sql->bindParam(':name', $name, PDO::PARAM_STR);
70         $sql->bindParam(':prezzo', $price, PDO::PARAM_INT);
71         $sql->bindParam(':quantita', $quantity, PDO::PARAM_STR);
72     }
73 }
```

Figura 3 getProduct controllo prodotto richiesto

Nella modifica è stato aggiunto un controllo per assicurarsi se c'è o meno un prodotto nella tabella "compra", questo è stato fatto perché la query inizialmente con le join prendeva anche i valori dalla tabella ponte, ma se non ci sono dati non ritorna niente, ed essendo questa funzione utilizzata per altre cose oltre che per il carrello deve prendere i dati anche se non ci sono.

Problemi riscontrati e soluzioni adottate

Tutti i problemi che ho avuto sono legati alla modifica della quantità di prodotti nel carrello.

Punto della situazione rispetto alla pianificazione

Oggi ho completato la parte del carrello, ma senza l'eliminazione completa del prodotto, questo perché così non perdo altro tempo in quello ma mi porto avanti.
Devo ancora gestire il caso in cui 2 negozi facciano riferimento allo stesso prodotto e in uno dei 2 diminuisca la quantità.

Programma di massima per la prossima giornata di lavoro

Implementare il checkout.

Diario di lavoro

Luogo	SAMT
Data	08.04.2019

Lavori svolti

Oggi ho iniziato la pagina dell'amministratore.
L'amministratore ha la possibilità di aggiungere e gestire i negozi e i relativi gestori di essi.
Per implementare questa funzionalità ho per prima cora creato una pagina, molto simile a quella del venditore, ma in questo caso non vengono mostrate le categorie ma solo i negozi.
L'immagine qui sotto mostra la base della pagina dell'amministratore.

Progetto vendita - gestore

[→]

PRODOTTI

NEGOZI TROVATI



Copyright ©2019 All rights reserved | This template is made with ❤ by [Colorlib](#)

Figura 1 pagina iniziale amministratore

Una volta impostata la pagina ho aggiunto il modal che si va ad occupare della registrazione, il cui codice è mostrato nelle immagini che seguono.

```

4   <div id="addModal" class="modal" style="...">
5
6     <!-- Modal content -->
7     <div class="modal-content">
8       <span class="close"><x></span>
9       <h5>Inserimento negozio</h5>
10      <form action="php echo URL ?&gt;admin/insertShop" method="POST" enctype="multipart/form-data"&gt;
11        &lt;!-- ##### Single Shop Details Area Start ##### --&gt;
12        &lt;section class="single_product_details_area d-flex align-items-center sBody"&gt;
13
14          &lt;!-- Single Shop Description --&gt;
15          &lt;div class="single_product_desc clearfix"&gt;
16            &lt;h6&gt;Negozio&lt;/h6&gt;
17            &lt;span&gt;Nome&lt;/span&gt;
18            &lt;input type="text" placeholder="Nome" id="shopName" name="shopName" onkeyup="convalidate(this.value, this.id, regLet
19            &lt;span&gt;Indirizzo&lt;/span&gt;
20            &lt;input type="text" placeholder="Indirizzo" id="shopAddress" name="shopAddress" onkeyup="convalidate(this.value, this
21            &lt;span&gt;Città&lt;/span&gt;
22            &lt;input type="text" placeholder="Città" id="shopCity" name="shopCity" onkeyup="convalidate(this.value, this.id, regLe
23            &lt;span&gt;Telefono&lt;/span&gt;
24            &lt;input type="text" placeholder="Telefono" id="shopPhone" name="shopPhone" onkeyup="convalidate(this.value, this.id,
25          &lt;/div&gt;
26
27          &lt;!-- Single Shop Description --&gt;
28          &lt;div class="single_product_desc clearfix"&gt;
29            &lt;h6&gt;Venditore&lt;/h6&gt;
30            &lt;span&gt;Nome&lt;/span&gt;
31            &lt;input type="text" placeholder="Nome" id="name" name="name" onkeyup="convalidate(this.value, this.id, regLetters)" r
32            &lt;span&gt;Cognome&lt;/span&gt;
33            &lt;input type="text" placeholder="Cognome" id="surname" name="surname" onkeyup="convalidate(this.value, this.id, regLe
34            &lt;span&gt;Telefono&lt;/span&gt;
35            &lt;input type="text" placeholder="Telefono" id="phone" name="phone" onkeyup="convalidate(this.value, this.id, regPhone
36            &lt;span&gt;Email&lt;/span&gt;
37            &lt;input type="text" placeholder="Email venditore" id="mail" name="mail" onkeyup="convalidate(this.value, this.id, reg
38            &lt;span&gt;Password&lt;/span&gt;
39            &lt;input type="password" placeholder="Password" id="password" name="password" onkeyup="checkPassword(this.value)" requ
40            &lt;span&gt;Conferma password&lt;/span&gt;
41            &lt;input type="password" placeholder="Conferma password" id="confirm-password" name="password" onkeyup="confirm()" di
42          &lt;/div&gt;
43
44          &lt;!-- Messaggio password invalida --&gt;
45          &lt;div class="single_product_desc clearfix"&gt;
46            &lt;p id="invalid-pass" style="color: red;" hidden&gt;Password non valida (8-25 caratteri, sia lettere che numeri)&lt;/p&gt;
47            &lt;p id="mailExists" style="color: red;" hidden&gt;ERRORE: mail già in uso&lt;/p&gt;
48            &lt;p id="noPassMatch" style="color: red;" hidden&gt;ERRORE: password diverse&lt;/p&gt;
49            &lt;p id="InvalidInput" style="color: red;" hidden&gt;ERRORE: 1 o più dati inseriti errati o mancanti&lt;/p&gt;
50            &lt;p id="invalidInput"&gt;TUTTI I CAMPI SONO OBBLIGATORI!&lt;/p&gt;
51          &lt;/div&gt;
52
53          &lt;div class="formDiv"&gt;
54            &lt;input type="submit" id="register-submit" value="INSERISCI" disabled/&gt;
55          &lt;/div&gt;
56        &lt;/section&gt;
57        &lt;!-- ##### Single Shop Details Area End ##### --&gt;
58      &lt;/form&gt;
59    &lt;/div&gt;
60
61  &lt;/div&gt;
</pre

```

Il codice mostra che la base del modal, come le classi e la struttura, è la stessa dei modal creati in precedenza per i prodotti, la differenza è il contenuto, in questo caso vengono inseriti i dati per fare la registrazione del prodotto e del venditore che lo gestisce.

Essendo una registrazione la maggior parte del codice era già esistente, mi è bastato rinominare nomi e id dei componenti in modo corretto e poi riportare le funzioni JavaScript utilizzate nella pagina di registrazione dei compratori.

Una volta fatto ho dovuto creare 3 funzioni in PHP, 2 da model e una controller.

La prima funzione l'ho creata nella classe controller "Admin" ed è quella che viene richiamata dal form al click del bottone submit. La funzione si occupa di prendere i dati inseriti dall'utente e passarli alle relative funzioni che le inseriscono nel database, come mostrato dalla foto qui sotto.

```

85     require_once 'application/models/shop.php';
86     $shop = new ShopModel();
87     require_once 'application/models/dealer.php';
88     $dealer = new Dealer();
89
90     //Prendo le variabili passate dal POST del negozio
91     $shopName = isset($_POST["shopName"]) ? $_POST["shopName"] : null;
92     $shopAddress = isset($_POST["shopAddress"]) ? $_POST["shopAddress"] : null;
93     $shopCity = isset($_POST["shopCity"]) ? $_POST["shopCity"] : null;
94     $shopPhone = isset($_POST["shopPhone"]) ? $_POST["shopPhone"] : null;
95
96     //Prendo le variabili passate dal POST del venditore
97     $name = isset($_POST["name"]) ? $_POST["name"] : null;
98     $surname = isset($_POST["surname"]) ? $_POST["surname"] : null;
99     $phone = isset($_POST["phone"]) ? $_POST["phone"] : null;
100    $mail = isset($_POST["mail"]) ? $_POST["mail"] : null;
101    $password = isset($_POST["password"]) ? $_POST["password"] : null;
102    $password = hash( algo: 'sha512', $password);
103
104    //Se i campi che devono comparire non sono vuoti
105    if ($shopName != null && $shopAddress != null && $shopCity != null && $shopPhone != null && $mail != null)
106        //Inserisco i dati del negozio
107        $shop->insertShop($shopName, $shopAddress, $shopCity, $shopPhone, $mail);
108    }
109
110    //Se i campi che devono comparire non sono vuoti
111    if ($mail != null && $name != null && $surname != null && $password != null && $phone != null) {
112        //Inserisco i dati del venditore
113        $dealer->insertDealer($mail, $name, $surname, $password, $phone);
114    }
115
116    header( string: "location: ". URL ."admin/home");

```

Figura 2 registerShopDealer registrazione negozio e venditore

L'immagine mostra solo la parte interno all'"if" che controlla che ci sia il metodo POST, il controllo presente in ogni funzione.

Come si può notare la funzione fa riferimento a 2 diversi model, e 2 funzioni di essi, questo perché non deve registrare solo un nuovo negozio, ma anche salvare un nuovo venditore per quel negozio. Le funzioni non hanno niente di speciale, si occupano semplicemente di inserire i dati presi come argomento all'interno delle tabelle, come mostrato dalle raffigurazioni che seguono.

```

17     public function insertDealer($email, $name, $surname, $password, $phone){
18         //Connetto al database
19         $conn = $this->connection->sqlConnection();
20
21         //Prendo i dati dell'utente in base alla mail
22         $sql = $conn->prepare("INSERT INTO gestore(email, nome, cognome, password, telefono)
23             values(:email, :name, :surname, :pass, :phone)");
24         $sql->bindParam(':email', $email, PDO::PARAM_STR);
25         $sql->bindParam(':name', $name, PDO::PARAM_STR);
26         $sql->bindParam(':surname', $surname, PDO::PARAM_STR);
27         $sql->bindParam(':pass', $password, PDO::PARAM_STR);
28         $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
29
30         //Se ci sono dei valori
31         if($sql->execute()) {
32             return true;
33         } else {
34             return $sql->errorInfo();
35         }
36         $conn = null;
37     }

```

Figura 3 insertDealer funzione di inserimento del venditore

```

17     public function insertShop($name, $address, $city, $phone, $email){
18         //Connetto al database
19         $conn = $this->connection->sqlConnection();
20
21         //Inserisco i dati del nuovo negozio
22         $sql = $conn->prepare("INSERT INTO negozio(nome, indirizzo, citta, telefono, email_gestore)
23             values(:name, :address, :city, :phone, :email)");
24         $sql->bindParam(':name', $name, PDO::PARAM_STR);
25         $sql->bindParam(':address', $address, PDO::PARAM_STR);
26         $sql->bindParam(':city', $city, PDO::PARAM_STR);
27         $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
28         $sql->bindParam(':email', $email, PDO::PARAM_STR);
29
30         //Se ci sono dei valori
31         if($sql->execute()) {
32             echo "LO";
33         }else {
34             print_r($sql->errorInfo());
35         }
36         $conn = null;
37     }

```

Figura 4 insertShop inserimento di un negozio e il suo venditore

Una volta fatto questo ho impostato la visualizzazione dei negozi all'interno della pagina. La struttura di base è la stessa di quella dei prodotti ma in questo caso vengono solo scritti i titoli, che mostrano il nome del negozio, e un link per archiviare il negozio, come mostrato sotto.

```

182     function getData(){
183         xhttp.onreadystatechange = function () {
184             if (this.readyState === 4 && this.status === 200) {
185
186                 //Prendo i valori passati dal server e li metto in un array
187                 var obj = JSON.parse(xhttp.responseText);
188
189                 //Prendo il corpo del div in cui inserire i campi
190                 var divbody = document.getElementById( elementId: 'shopContainer' );
191                 divbody.innerHTML = "";
192
193                 //Inserisco tutti le righe con i relativi dati.
194                 for (var i = 0; i < obj.length; i++) {
195
196                     //Prendo i div principale per i prodotti
197                     var divContainer = document.createElement( tagName: "div");
198                     divContainer.setAttribute( qualifiedName: "class", value: "col-12 col-sm-6 col-lg-4");
199                     var divWrapper = document.createElement( tagName: "div");
200                     divWrapper.setAttribute( qualifiedName: "class", value: "single-product-wrapper");
201
202                     //Creo un nuovo div che contiene le informazioni e lo metto nel padre
203                     var div = document.createElement( tagName: "div");
204                     div.setAttribute( qualifiedName: "class", value: "product-description");
205                     // noinspection JSUnresolvedDeclaration
206                     var a = document.createElement( tagName: "a");
207                     a.setAttribute( qualifiedName: "href", value: "#");
208                     var h6 = document.createElement( tagName: "h6");
209                     h6.style = "margin-left: 10px";
210                     h6.setAttribute( qualifiedName: "id", value: obj[i]['nome'] + "." + obj[i]['indirizzo'] + "." + obj[i]['citta']);
211                     h6.setAttribute( qualifiedName: "onclick", value: "shopDetails(this)");
212                     h6.appendChild(document.createTextNode(obj[i]['nome']));
213                     a.appendChild(h6);
214                     div.appendChild(a);

```

```

218     var a = document.createElement( tagName: "a");
219     a.setAttribute( qualifiedName: "href", value: "#");
220     var prezzo = obj[i]['prezzo'] + " Fr.";
221     a.style = "margin-left: 10px; font-size: 15px;";
222     a.appendChild(document.createTextNode( data: "archivia"));
223     div.appendChild(a);
224     divWrapper.appendChild(div);

225
226     //aggiungo tutti i div a quello primario
227     divContainer.appendChild(divWrapper);
228     divbody.appendChild(divContainer);
229 }
230 }
231 }
232 xhttp.open( method: "POST", url: "/gestionevendita2018/admin/getShops", async: true);
233 xhttp.send();
234

```

Figura 5 shopDetails inserimento

Le immagini mostrano la funzione JavaScript che va a prendere i dati dei negozi e li inserisce nel div corretto. La funzione controller e model sono molto semplici, si occupano di andare a prendere i dati dello shop selezionato e del gestore di esso, e poi passarli alla funzione JavaScript.

Quelle mostrate sono le funzioni model e controller che prendono i dati.

```

76     public function getShops()
77     {
78         //Connetto al database
79         $conn = $this->connection->sqlConnection();
80
81         //Prendo i dati dei negozi
82         $sql = $conn->prepare("SELECT nome, indirizzo, citta FROM negozio");
83
84         //Eseguo la query
85         $sql->execute();
86
87         $dataArray = array();
88         //Se ci sono dei valori
89         if ($sql->rowCount() > 1) {
90             // Ciclo tutti i valori
91             while ($row = $sql->fetch()) {
92
93                 array_push( &array: $dataArray, $row);
94             }
95         } else if ($sql->rowCount() == 1) {
96             array_push( &array: $dataArray, $sql->fetch());
97         }
98         $conn = null;
99         return $dataArray;
100    }

```

Figura 6 model getShops che prende le informazioni dei negozi

```

40
41
42     public function getShops(){
43         //Controllo che la funzione entri in POST
44         if ($_SERVER["REQUEST_METHOD"] == "POST") {
45             //Controllo che sia aperta la sessione e abbia fatto il login un amministratore
46             if (isset($_SESSION['admin'])) {
47                 //Prendo la classe model
48                 require_once 'application/models/shop.php';
49                 $shop = new ShopModel();
50
51                 $shops = $shop->getShops();
52
53                 //Stampo con json i valori dei coach
54                 header( string: 'Content-Type: application/json' );
55                 echo json_encode($shops);
56             } else{
57                 header( string: "location: javascript://history.back()" );
58             }
59         }
60     }

```

Figura 7 controller getShops che recupera i dati e li stampa

Una volta presi e stampati i dati in JSON la funzione JS mostrata sopra inserisce i dati all'interno della pagina.

Una volta fatto ho implementato il form nel modal per modificare i file. Il modal è quasi uguale a quello per aggiungere ma senza gli input della password, che non si può modificare, e con degli input nascosti in più che contengono le variabili delle chiavi del negozio e del gestore, per fare in modo che se vengono modificati riesce a prenderli lo stesso.

La prima cosa che viene fatta è riempire gli input con i dati del negozio e del gestore richiesti, questo con una funzione JavaScript che li prende, come mostrato sotto.

```

237     function shopDetails(tag) {
238         var data = tag.id.split(".");
239
240         //Prendo il corpo del div in cui inserire i campi
241         var divbody = document.getElementById( elementId: 'modSBody' );
242         xhttp.onreadystatechange = function () {
243             if (this.readyState === 4 && this.status === 200) {
244
245                 //Prendo i valori passati dal server e li metto in un array
246                 var obj = JSON.parse(xhttp.responseText);
247
248                 //Cambio i dati del negozio
249                 var shopName = document.getElementById( elementId: "modShopName" );
250                 shopName.setAttribute( qualifiedName: "value", obj[0][0] );
251                 var shopAddress = document.getElementById( elementId: "modShopAddress" );
252                 shopAddress.setAttribute( qualifiedName: "value", obj[0]['indirizzo'] );
253                 var shopCity = document.getElementById( elementId: "modShopCity" );
254                 shopCity.setAttribute( qualifiedName: "value", obj[0]['citta'] );
255                 var shopPhone = document.getElementById( elementId: "modShopPhone" );
256                 shopPhone.setAttribute( qualifiedName: "value", obj[0][3] );
257
258                 //Cambio i dati del gestore
259                 var name = document.getElementById( elementId: "modName" );
260                 name.setAttribute( qualifiedName: "value", obj[0]['name'] );
261                 var surname = document.getElementById( elementId: "modSurname" );
262                 surname.setAttribute( qualifiedName: "value", obj[0]['cognome'] );
263                 var phone = document.getElementById( elementId: "modPhone" );
264                 phone.setAttribute( qualifiedName: "value", obj[0]['telefono'] );
265                 var mail = document.getElementById( elementId: "modMail" );
266                 mail.setAttribute( qualifiedName: "value", obj[0]['email'] );
267
268                 document.getElementById( elementId: 'modifyModal' ).style.display = "block";
269             }
270         }
271         xhttp.open( method: "POST", url: "/gestionevendita2018/admin/getShopDealer", async: true );
272         xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded" );
273         xhttp.send( body: "&name="+ data[0] +"&address="+ data[1] +"&city="+ data[2] );

```

Figura 8 shopDetails inserimento dei dati del venditore e del suo negozio

La funzione controller richiamata a riga 268 prende semplicemente i dati dal model, che invece fa una query con una doppia join per prendere tutti i dati sia del negozio che del venditore. Le immagini mostrano le 2 funzioni in questione.

```
65     public function getShopDealer() {
66         //Controllo che la funzione entri in POST
67         if ($_SERVER["REQUEST_METHOD"] == "POST") {
68
69             //Controllo che sia aperta la sessione e abbia fatto il login un amministratore
70             if (isset($_SESSION['admin'])) {
71
72                 //Prendo la classe model
73                 require_once 'application/models/shop.php';
74                 $shop = new ShopModel();
75
76                 //Prendo le variabili passate dal POST del negozio
77                 $name = isset($_POST["name"]) ? $_POST["name"] : null;
78                 $address = isset($_POST["address"]) ? $_POST["address"] : null;
79                 $city = isset($_POST["city"]) ? $_POST["city"] : null;
80
81                 //Se i campi che devono comparire non sono vuoti
82                 if ($name != null && $address != null && $city != null) {
83                     $shops = $shop->getShopDealer($name, $address, $city);
84                 }
85
86                 //Stampo con json i valori dei coach
87                 header( string: 'Content-Type: application/json');
88                 echo json_encode($shops);
89             } else{
90                 header( string: "location: javascript://history.back()");
91             }
92             } else{
93                 header( string: "location: javascript://history.back()");
94             }
95         }
96     }
```

Figura 9 controller getShopsDealer prende i valori del negozio e del venditore

```

105     public function getShopDealer($name, $address, $city)
106     {
107         //Connetto al database
108         $conn = $this->connection->sqlConnection();
109
110         //Prendo i dati dei negozi
111         $sql = $conn->prepare("SELECT * FROM negozio n
112                             right outer join gestore g on g.email = n.email_gestore
113                             WHERE n.nome LIKE :nameR AND n.indirizzo LIKE :addressR AND n.citta LIKE :cityR
114                             union
115                             SELECT * FROM negozio n
116                             left outer join gestore g on g.email = n.email_gestore
117                             WHERE n.nome LIKE :nameL AND n.indirizzo LIKE :addressL AND n.citta LIKE :cityL");
118         $sql->bindParam(':nameR', $name, PDO::PARAM_STR);
119         $sql->bindParam(':addressR', $address, PDO::PARAM_STR);
120         $sql->bindParam(':cityR', $city, PDO::PARAM_STR);
121         $sql->bindParam(':nameL', $name, PDO::PARAM_STR);
122         $sql->bindParam(':addressL', $address, PDO::PARAM_STR);
123         $sql->bindParam(':cityL', $city, PDO::PARAM_STR);
124
125         //Eseguo la query
126         $sql->execute();
127
128         $dataArray = array();
129         //Se ci sono dei valori
130         if ($sql->rowCount() > 1) {
131             // Ciclo tutti i valori
132             while ($row = $sql->fetch()) {
133
134                 array_push( &array, $dataArray, $row);
135             }
136         } else if ($sql->rowCount() == 1) {
137             array_push( &array, $dataArray, $sql->fetch());
138         }
139         $conn = null;
140         return $dataArray;
141     }

```

Figura 10 model getShopDealer prende i dati del negozio e del suo gestore dal database e le ritorna

Le ultime funzioni per ultimare la modifica dei dati sono quelle che si occupano di modificare effettivamente i dati nel database. Come al solito la funzione controller della classe Admin, come tutti gli altri controller di questa pagina, sono semplicemente da passaggio per modificare i file, come mostrato nell'immagine.

```

193     require_once 'application/models/shop.php';
194     $shop = new ShopModel();
195     require_once 'application/models/dealer.php';
196     $dealer = new Dealer();
197
198     //Prendo le variabili passate dal POST del negozio
199     $shopName = isset($_POST["modShopName"]) ? $_POST["modShopName"] : null;
200     $shopAddress = isset($_POST["modShopAddress"]) ? $_POST["modShopAddress"] : null;
201     $shopCity = isset($_POST["modShopCity"]) ? $_POST["modShopCity"] : null;
202     $shopPhone = isset($_POST["modShopPhone"]) ? $_POST["modShopPhone"] : null;
203
204     //Prendo le variabili passate dal POST del venditore
205     $name = isset($_POST["modName"]) ? $_POST["modName"] : null;
206     $surname = isset($_POST["modSurname"]) ? $_POST["modSurname"] : null;
207     $phone = isset($_POST["modPhone"]) ? $_POST["modPhone"] : null;
208     $mail = isset($_POST["modMail"]) ? $_POST["modMail"] : null;
209
210     //Prendo i vecchi valori delle chiavi passati dal POST
211     $oldMail = isset($_POST["oldMail"]) ? $_POST["oldMail"] : null;
212     $oldShopName = isset($_POST["oldShopName"]) ? $_POST["oldShopName"] : null;
213     $oldShopAddress = isset($_POST["oldShopAddress"]) ? $_POST["oldShopAddress"] : null;
214     $oldShopCity = isset($_POST["oldShopCity"]) ? $_POST["oldShopCity"] : null;
215
216     //Se i campi che devono comparire non sono vuoti
217     if ($mail != null && $name != null && $surname != null && $phone != null && $oldMail != null) {
218         //Inserisco i dati del venditore
219         $dealer->modifyDealer($mail, $name, $surname, $phone, $oldMail);
220     }
221
222     //Se i campi che devono comparire non sono vuoti
223     if ($shopName != null && $shopAddress != null && $shopCity != null && $shopPhone != null && $mail != null && $oldShopName != null) {
224         //Inserisco i dati del negozio
225         $shop->modifyShop($shopName, $shopAddress, $shopCity, $shopPhone, $mail, $oldShopName, $oldShopAddress, $oldShopCity);
226     }
227
228     header( string: "location: ". URL . "admin/home");

```

Figura 11 controller modifyShop, prende i dati e richiama le funzioni che modificano dati

L'immagine, come alcune precedenti, mostra solo il codice dopo il controllo del POST.
Le 2 funzioni che modificano i dati sono quasi uguali se non per i dati che inseriscono, che sono per forza di cose diverse, e sono mostrate nelle immagini che seguono.

```

47     public function modifyDealer($email, $name, $surname, $phone, $oldMail){
48         //Connetto al database
49         $conn = $this->connection->sqlConnection();
50
51         //Prendo i dati dell'utente in base alla mail
52         $sql = $conn->prepare("UPDATE gestore set email = :email, nome = :name, cognome = :surname, telefono = :phone
53                               WHERE email LIKE :emailW");
54         $sql->bindParam(':email', $email, PDO::PARAM_STR);
55         $sql->bindParam(':name', $name, PDO::PARAM_STR);
56         $sql->bindParam(':surname', $surname, PDO::PARAM_STR);
57         $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
58         $sql->bindParam(':emailW', $oldMail, PDO::PARAM_STR);
59
60         //Se ci sono dei valori
61         if($sql->execute()) {
62             return true;
63         } else {
64             return $sql->errorInfo();
65         }
66         $conn = null;
67     }

```

Figura 12 model modifyDealer, modifica i dati con i nuovi ricevuti

```

48     public function modifyShop($name, $address, $city, $phone, $email, $oldName, $oldAddress, $oldCity){
49         //Connetto al database
50         $conn = $this->connection->sqlConnection();
51
52         //Prendo i dati dell'utente in base alla mail
53         $sql = $conn->prepare("UPDATE negozio set nome = :name, indirizzo = :address, citta = :city, telefono = :phone, email_gestore = :email
54             WHERE nome LIKE :nameW AND indirizzo LIKE :addressW AND citta LIKE :cityW");
55         $sql->bindParam(':name', $name, PDO::PARAM_STR);
56         $sql->bindParam(':address', $address, PDO::PARAM_STR);
57         $sql->bindParam(':city', $city, PDO::PARAM_STR);
58         $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
59         $sql->bindParam(':email', $email, PDO::PARAM_STR);
60         $sql->bindParam(':nameW', $oldName, PDO::PARAM_STR);
61         $sql->bindParam(':addressW', $oldAddress, PDO::PARAM_STR);
62         $sql->bindParam(':cityW', $oldCity, PDO::PARAM_STR);
63
64         //Se ci sono dei valori
65         if($sql->execute()) {
66             return true;
67         }else {
68             return $sql->errorInfo();
69         }
70         $conn = null;
71     }

```

Figura 13 model modifyShop, modifica i dati con i nuovi ricevuti

Dopo questo ho lavorato alla creazione del file pdf, essendo che la parte precedente è stata molto veloce perché era per la maggior parte codice già precedente implementato.

Per fare il pdf ho messo in un form tutto il contenuto del carrello e ho inserito degli input nascosti che contenevano i dati da portare nel pdf, ovvero il nome, il prezzo e la quantità del prodotto, la quantità e il prezzo totali e la scelta del negozio di ritiro.

Il form va a richiamare una funzione nel controller “customer” che prende i dati e crea il pdf.

```

163     public function doCheckout(){
164         //Se la sessione è aperta apro le pagine altrimenti no
165         if(isset($_SESSION['customer'])) {
166
167             //Prendo le variabili passate dal POST
168             $name = isset($_POST["name"])? $_POST["name"] : null;
169             $price = isset($_POST["price"])? $_POST["price"] : null;
170             $quantity = isset($_POST["quantity"])? $_POST["quantity"] : null;
171             $totPrice = isset($_POST["totPrice"])? $_POST["totPrice"] : null;
172             $cartObjects = isset($_POST["cartObjects"])? $_POST["cartObjects"] : null;
173             $pickupShop = isset($_POST["pickupShop"])? $_POST["pickupShop"] : null;
174
175             if($name != null && $price != null && $quantity != null && $totPrice != null && $cartObjects != null && $pickupShop != null)
176
177                 //Prendo i dati del negozio di ritiro
178                 $pickupShopInfos = explode( delimiter: ".", $pickupShop);
179
180                 require('application/fpdf/fpdf.php');
181
182                 // crea l'istanza del documento
183                 $pdf = new FPDF();
184                 $pdf->AddPage();
185
186                 //---Header---
187                 // Arial bold 15
188                 $pdf->SetFont('Arial', 'B', 15);
189                 // Dati utente
190                 $pdf->Cell(30, 10, $_SESSION['customer'], 0, 0, 'L');
191                 //Data odierna
192                 $pdf->Cell(130);
193                 $pdf->Cell(30, 10, date( format: "Y/m/d"), 0, 0, 'R');
194                 // Vado a capo
195                 $pdf->Ln(20);

```

```

199     $pdf->SetFont('Arial', 'B', 30);
200     // Muove verso destra
201     $pdf->Cell(80);
202     // Titolo in riquadro
203     $pdf->Cell(30, 10, 'Il suo ordine', 0, 0, 'C');
204     // Interruzione di linea
205     $pdf->Ln(20);
206
207     //Metto l'indice
208     // Seleziona Arial 15
209     $pdf->SetFont('Arial', '', 16);
210     // Indice
211     $pdf->Cell(30, 10, 'Prodotto', 0, 0, 'L');
212     $pdf->Cell(50);
213     $pdf->Cell(30, 10, 'Quantità', 0, 0, 'C');
214     $pdf->Cell(50);
215     $pdf->Cell(30, 10, 'Prezzo', 0, 0, 'C');
216     // Interruzione di linea
217     $pdf->Ln(20);
218
219     //Traccio una linea
220     $pdf->SetDrawColor(180, 180, 180);
221     $pdf->SetLineWidth(1);
222     $pdf->Line(10, 70, 200, 70);
223     // Interruzione di linea
224     $pdf->Ln(10);
225
226     //Variabile per la prossima riga
227     $nextLine = 70;
228     for ($i = 0; $i < count($name); $i++) {
229         //Inserisco i prodotti
230         $pdf->Cell(30, 10, $name[$i], 0, 0, 'L');
231         $pdf->Cell(50);
232         $pdf->Cell(30, 10, $quantity[$i], 0, 0, 'C');
233         $pdf->Cell(50);
234         $pdf->Cell(30, 10, $price[$i] . 'Fr.', 0, 0, 'C');
235         // Interruzione di linea
236         $pdf->Ln(20);
237         $nextLine += 21;
238     }
239
240     //Traccio una linea
241     $pdf->SetDrawColor(180, 180, 180);
242     $pdf->SetLineWidth(1);
243     $pdf->Line(10, $nextLine, 200, $nextLine);
244     // Interruzione di linea
245     $pdf->Ln(10);
246
247     //Metto il totale
248     // Seleziona Arial 15
249     $pdf->SetFont('Arial', '', 16);
250     // Indice
251     $pdf->Cell(30, 10, 'Totale', 0, 0, 'L');
252     $pdf->Cell(50);
253     $pdf->Cell(30, 10, $cartObjects, 0, 0, 'C');
254     $pdf->Cell(50);
255     $pdf->Cell(30, 10, $totPrice . 'Fr.', 0, 0, 'C');
256     // Interruzione di linea
257     $pdf->Ln(10);
258     $nextLine += 30;
259
260     $pdf->SetDrawColor(180, 180, 180);
261     $pdf->SetLineWidth(1);
262     $pdf->Line(10, $nextLine, 200, $nextLine);
263     // Interruzione di linea
264     $pdf->Ln(10);
265
266     //Metto il totale
267     // Seleziona Arial 15
268     $pdf->SetFont('Arial', '', 16);
269     // Indice
270     $pdf->Cell(30, 10, 'Negozio', 0, 0, 'L');
271     $pdf->Cell(50);
272     $pdf->Cell(30, 10, $pickupShopInfos[0], 0, 0, 'C');
273     // Indirizzo
274     $pdf->Ln(10);
275     $pdf->Cell(80);
276     $pdf->Cell(30, 10, $pickupShopInfos[1], 0, 0, 'C');
277     $pdf->Cell(25);
278     $pdf->Cell(30, 10, $pickupShopInfos[2], 0, 0, 'C');
279     // Contatti
280     $pdf->Ln(10);
281     $pdf->Cell(80);
282     $pdf->Cell(30, 10, $pickupShopInfos[3] . "..." . $pickupShopInfos[4], 0, 0, 'C');
283     $pdf->Cell(25);
284     $pdf->Cell(30, 10, $pickupShopInfos[5], 0, 0, 'C');
285     // Interruzione di linea
286     $pdf->Ln(10);
287     $nextLine += 30;
288
289

```

```

292     $pdf->SetY(-35);
293     // Arial italic 8
294     $pdf->SetFont('Arial','I',10);
295     // Numero della pagina
296     $pdf->Cell(0,10,'Progetto vendita piccoli negozi',0,0,'L');

297     //Salvo il pdf
298     $pdf->Output();
299 }else{
300     header( string: "location: ". URL );
301 }
302 }else{
303     header( string: "location: ". URL );
304 }
305 }
306 }
```

Figura 14 controller doCheckout funzione che crea il pdf

La funzione dopo aver preso i dati controlla che effettivamente siano creati e non nulli ma in caso lo sono ritorna alla pagina iniziale.

La visualizzazione definitiva del file pdf è mostrata nell'immagine che segue.



Figura 15 file pdf definitivo

Si può notare che c'è un errore nella scritta "Quantità" questo perché il charset non riesce a leggere la "à" e scrive quell'altra lettera.

Problemi riscontrati e soluzioni adottate

Ho avuto alcuni problemi nello scrivere le query, inizialmente perché sbagliavo il nome dei campi ma poi anche con la riga per prendere i dati del negozio singolo con il suo gestore, questo perché provavo ad utilizzare una "full outer join" che in mysql non è più utilizzabile.

Punto della situazione rispetto alla pianificazione

Oggi ho completato la pagina dell'amministratore e la creazione del pdf
Devo ancora gestire il caso in cui 2 negozi facciano riferimento allo stesso prodotto e in uno dei 2 diminuisca la quantità.

Programma di massima per la prossima giornata di lavoro

Completare la documentazione.

Diario di lavoro

Luogo	SAMT
Data	09.04.2019

Lavori svolti

La prima cosa che ho fatto questa mattina è stato creare le funzioni per archiviare i negozi. Innanzitutto ho aggiunto un evento all'onclick del link “archiviato” quando vengono mostrati i negozi nella pagina dell'amministratore passando alla funzione richiamata l'id del link che contiene i valori della chiave del negozio concatenate con un “.”, dopodiché ho creato la funzione richiesta.

```

295  function fileShop(id){
296      var id = id.split(".");
297      console.log(id);
298      xhttp.onreadystatechange = function () {
299          if (this.readyState === 4 && this.status === 200) {
300              location.reload();
301          }
302      }
303      xhttp.open( method: "POST", url: "/gestionevendita2018/admin/fileShop", async: true);
304      xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
305      xhttp.send( body: "&name="+ id[0] +"&address="+ id[1] +"&city="+ id[2]);
306  }

```

Figura 1 JS fileShop che richiama la funzione per archiviare

La funzione in JavaScript scomponete l'id prendendo la chiave separata e poi richiama la funzione controller passandogli i dati della chiave.

```

240     public function fileShop()
241     {
242         //Se la funzione è richiamata come POST
243         if ($_SERVER["REQUEST_METHOD"] == "POST") {
244
245             //Prendo la classe model
246             require_once 'application/models/shop.php';
247             $shop = new ShopModel();
248
249             //Prendo le variabili passate dal POST
250             $name = isset($_POST["name"]) ? $_POST["name"] : null;
251             $address = isset($_POST["address"]) ? $_POST["address"] : null;
252             $city = isset($_POST["city"]) ? $_POST["city"] : null;
253
254             if($name != null && $address != null && $city != null){
255                 $shop->fileShop($name, $address, $city);
256             }
257
258             header( string: "location: ". URL ."admin/home");
259         }else{
260             //Ritorno alla pagina precedente
261             header( string: "location: javascript://history.back()");
262         }
263     }

```

Figura 2 controller fileShop funzione che prende i dati e richiama il model per archiviare

La funzione controller nella classe “Admin” si occupa solo di prendere i dati e passarli al model per archiviare quel determinato negozio.

```

180     public function fileShop($name, $address, $city) {
181         //Connetto al database
182         $conn = $this->connection->sqlConnection();
183
184         //Prendo i dati dell'utente in base alla mail
185         $sql = $conn->prepare("UPDATE negozio set archiviato = 1
186             WHERE nome LIKE :name AND indirizzo LIKE :address AND citta LIKE :city");
187         $sql->bindParam(':name', $name, PDO::PARAM_STR);
188         $sql->bindParam(':address', $address, PDO::PARAM_STR);
189         $sql->bindParam(':city', $city, PDO::PARAM_STR);
190
191         //Se ci sono dei valori
192         if($sql->execute()) {
193             return true;
194         }else {
195             return $sql->errorInfo();
196         }
197         $conn = null;
198     }

```

Figura 3 model fileShop funzione che archivia il negozio

La funzione model si occupa di archiviare il negozio, ovvero portare il campo booleano “archiviato” a 1 in modo che il negozio con le caratteristiche passate non possa più essere visualizzato a meno che non viene rimodificato dal database.

Dopodiché ho risolto un problema e ho continuato la documentazione.

La documentazione è quasi completata, mi mancano da finire le conclusioni, però oggi ho completato l’implementazione e a parte dei test.

Problemi riscontrati e soluzioni adottate

Un problema che ho riscontrato è stato con il carrello, inizialmente alle volte quando si aggiungevano o toglievano dei prodotti, soprattutto se era l'ultimo prodotto disponibile, dava errore, non si toglieva il prodotto o si aggiungeva per errore. Per risolvere il problema mi è bastato modificare il controllo nella funzione "getProduct" della classe model "ProductModel". Per risolvere bastava modificare il controllo di esistenza del prodotto nel database, perché prima faceva il controllo solo sul prodotto, e di conseguenza se lo stesso prodotto era posseduto da 2 clienti non riusciva ad eliminarlo perché passava il controllo, ora il controllo è come mostrato nell'immagine sotto.

```
46     public function getProduct($name, $price, $quantity, $custMail = null){  
47         //Connetto al database  
48         $conn = $this->connection->sqlConnection();  
49  
50         require_once 'application/models/buy.php';  
51         $buy = new BuyModel();  
52  
53         if(count($buy->getDataByProduct($name, $price, $quantity, $custMail)) > 0) {
```

Figura 4 modifica controllo esistenza prodotto carrello getProduct

La mail dell'utente viene passata come argomento opzionale della funzione, quindi ho anche dovuto modificare il richiamo della funzione stessa.

Punto della situazione rispetto alla pianificazione

Oggi ho completato il sito e portato avanti la documentazione.

Programma di massima per la prossima giornata di lavoro

Completare la documentazione.

Diario di lavoro

Luogo	SAMT
Data	10.04.2019

Lavori svolti

Oggi ho semplicemente completato la documentazione, inserendo le ultime parti che mancavano, ovvero la sitografia e il glossario. Poi ho fatto un ultimo giro di test per assicurarmi che tutto funzioni correttamente.
Dopodiché ho unito tutti i diari ho stampato le cose rilegate. Una volta fatto questo ho masterizzato la cartella contenente gli allegati e il codice sorgente in un DVD e ho inserito anche quello nella rilegatura.

Problemi riscontrati e soluzioni adottate

Non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Oggi ho completato, rilegato consegnato tutto, in linea con il gantt consuntivo.

Programma di massima per la prossima giornata di lavoro

-