Diario di lavoro

Luogo	SAMT
Data	19.02.2019

Lavori svolti

Oggi inizialmente ho dovuto risolvere un problema con il login i venditori, 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( elementId: '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 $value){
132             //Se il valore cercato è contenuto nel nome del prodotto passato
133             if(gettype(stripos($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 “stripos()” 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'del 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.

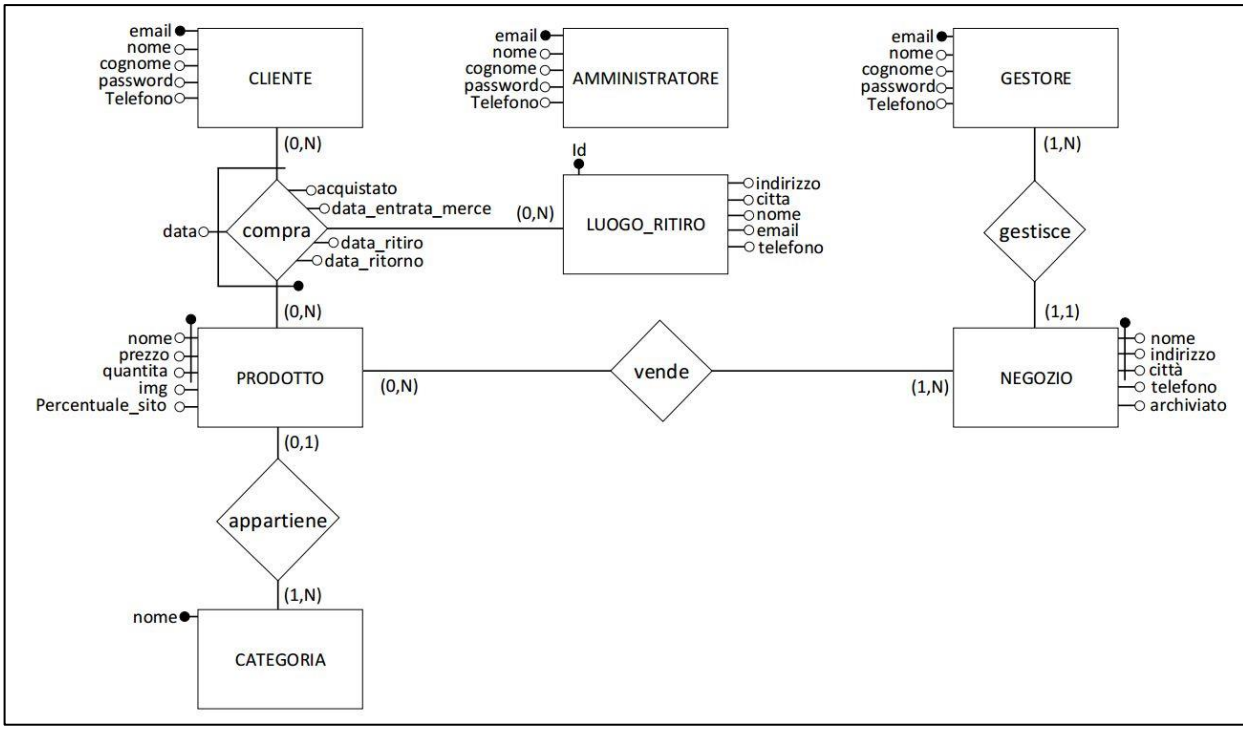

Figura 6 schema E-R con modifica chiave "prodotto"

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.