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 scompone l'id prendendo la chiave separata e poi richiama la funzione controller passandogli i dati della chiave.

```php
public function fileShop()
{
    //Se la funzione è richiamata come POST
    if ($_SERVER["REQUEST_METHOD"] == "POST") {

        //Prendo la classe model
        require_once 'application/models/shop.php';
        $shop = new ShopModel();

        //Prendo le variabili passate dal POST
        $name = isset($_POST["name"])? $_POST["name"] : null;
        $address = isset($_POST["address"])? $_POST["address"] : null;
        $city = isset($_POST["city"])? $_POST["city"] : null;

        if($name != null && $address != null && $city != null){
            $shop->fileShop($name, $address, $city);
        }

        header( string: "location: ". URL ."admin/home");
    }else{
        //Ritorno alla pagina precedente
        header( string: "location: javascript://history.back()");
    }
}
```

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.

```php
public function fileShop($name, $address, $city){
    //Connetto al database
    $conn = $this->connection->sqlConnection();

    //Prendo i dati dell'utente in base alla mail
    $sql = $conn->prepare("UPDATE negozio set archiviato = 1
                WHERE nome LIKE :name AND indirizzo LIKE :address AND citta LIKE :city");
    $sql->bindParam(':name', $name, PDO::PARAM_STR);
    $sql->bindParam(':address', $address, PDO::PARAM_STR);
    $sql->bindParam(':city', $city, PDO::PARAM_STR);

    //Se ci sono dei valori
    if($sql->execute()) {
        return true;
    }else {
        return $sql->errorInfo();
    }
    $conn = null;
}
```

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, quini 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.