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.

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.

```
<div id="addModal" class="modal" style="...">

    <!-- Modal content -->
    <div class="modal-content">
        <span class="close">×</span>
        <h5>Inserimento negozio</h5>
        <form action="<?php echo URL ?>admin/insertShop" method="POST" enctype="multipart/form-data">
            <!-- ##### Single Shop Details Area Start ##### -->
            <section class="single_product_details_area d-flex align-items-center sBody">

                <!-- Single Shop Description -->
                <div class="single_product_desc clearfix">
                    <h6>Negozio</h6>
                    <span>Nome</span>
                    <input type="text" placeholder="Nome" id="shopName" name="shopName" onkeyup="convalidate(this.value, this.id, regLet
                    <span>Indirizzo</span>
                    <input type="text" placeholder="Indirizzo" id="shopAddress" name="shopAddress" onkeyup="convalidate(this.value, this
                    <span>Città</span>
                    <input type="text" placeholder="Città" id="shopCity" name="shopCity" onkeyup="convalidate(this.value, this.id, regLe
                    <span>Telefono</span>
                    <input type="text" placeholder="Telefono" id="shopPhone" name="shopPhone" onkeyup="convalidate(this.value, this.id,
                </div>

                <!-- Single Shop Description -->
                <div class="single_product_desc clearfix">
                    <h6>Venditore</h6>
                    <span>Nome</span>
                    <input type="text" placeholder="Nome" id="name" name="name" onkeyup="convalidate(this.value, this.id, regLetters)" r
                    <span>Cognome</span>
                    <input type="text" placeholder="Cognome" id="surname" name="surname" onkeyup="convalidate(this.value, this.id, regLe
                    <span>Telefono</span>
                    <input type="text" placeholder="Telefono" id="phone" name="phone" onkeyup="convalidate(this.value, this.id, regPhone
                    <span>Email</span>
                    <input type="text" placeholder="Email venditore" id="mail" name="mail" onkeyup="convalidate(this.value, this.id, reg
                    <span>Password</span>
                    <input type="password" placeholder="Password" id="password" name="password" onkeyup="checkPassword(this.value)" requ
                    <span>Conferma password</span>
                    <input type="password" placeholder="Conferma password" id="confirm-password" name="password" onkeyup="confirm()"  di
                </div>

                <!-- Messaggio password invalida -->
                <div class="single_product_desc clearfix">
                    <p id="invalid-pass" style="color: red;" hidden>Password non valida <br>(8-25 caratteri, sia lettere che numeri)</p>
                    <p id="mailExists" style="color: red;" hidden>ERRORE: mail già in uso</p>
                    <p id="noPassMatch" style="color: red;" hidden>ERRORE: password diverse</p>
                    <p id="invalidInput" style="color: red;" hidden>ERRORE: 1 o più dati inseriti errati o mancanti</p>
                    <p id="invalidInput">TUTTI I CAMPI SONO OBBLIGATORI</p>
                </div>

                <div class="formDiv">
                    <input type="submit" id="register-submit" value="INSERISCI" disabled/>
                </div>
            </section>
            <!-- ##### Single Shop Details Area End ##### -->
        </form>
    </div>

</div>
```

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.

```php
require_once 'application/models/shop.php';
$shop = new ShopModel();
require_once 'application/models/dealer.php';
$dealer = new Dealer();

//Prendo le variabili passate dal POST del negozio
$shopName = isset($_POST["shopName"])? $_POST["shopName"] : null;
$shopAddress = isset($_POST["shopAddress"])? $_POST["shopAddress"] : null;
$shopCity = isset($_POST["shopCity"])? $_POST["shopCity"] : null;
$shopPhone = isset($_POST["shopPhone"])? $_POST["shopPhone"] : null;

//Prendo le variabili passate dal POST del venditore
$name = isset($_POST["name"])? $_POST["name"] : null;
$surname = isset($_POST["surname"])? $_POST["surname"] : null;
$phone = isset($_POST["phone"])? $_POST["phone"] : null;
$mail = isset($_POST["mail"])? $_POST["mail"] : null;
$password = isset($_POST["password"])? $_POST["password"] : null;
$password = hash( algo: 'sha512', $password);

//Se i campi che devono comparire non sono vuoti
if ($shopName != null && $shopAddress != null && $shopCity != null && $shopPhone != null && $mail != null)
    //Inserisco i dati del negozio
    $shop->insertShop($shopName, $shopAddress, $shopCity, $shopPhone, $mail);
}

//Se i campi che devono comparire non sono vuoti
if ($mail != null && $name != null && $surname != null && $password != null && $phone != null) {
    //Inserisco i dati del venditore
    $dealer->insertDealer($mail, $name, $surname, $password, $phone);
}

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.

```php
public function insertDealer($email, $name, $surname, $password, $phone){
    //Connetto al database
    $conn = $this->connection->sqlConnection();

    //Prendo i dati dell'utente in base alla mail
    $sql = $conn->prepare("INSERT INTO gestore(email, nome, cognome, password, telefono)
        values(:email, :name, :surname, :pass, :phone)");
    $sql->bindParam(':email', $email, PDO::PARAM_STR);
    $sql->bindParam(':name', $name, PDO::PARAM_STR);
    $sql->bindParam(':surname', $surname, PDO::PARAM_STR);
    $sql->bindParam(':pass', $password, PDO::PARAM_STR);
    $sql->bindParam(':phone', $phone, PDO::PARAM_STR);

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

Figura 3 insertDealer funzione di inserimento del venditore

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

    //Inserisco i dati del nuovo negozio
    $sql = $conn->prepare("INSERT INTO negozio(nome, indirizzo, citta, telefono, email_gestore)
        values(:name, :address, :city, :phone, :email)");
    $sql->bindParam(':name', $name, PDO::PARAM_STR);
    $sql->bindParam(':address', $address, PDO::PARAM_STR);
    $sql->bindParam(':city', $city, PDO::PARAM_STR);
    $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
    $sql->bindParam(':email', $email, PDO::PARAM_STR);

    //Se ci sono dei valori
    if($sql->execute()) {
        echo "LO";
    }else {
        print_r($sql->errorInfo());
    }
    $conn = null;
}
```

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.

```javascript
function getData(){
    xhttp.onreadystatechange = function () {
        if (this.readyState === 4 && this.status === 200) {

            //Prendo i valori passati dal server e li metto in un array
            var obj = JSON.parse(xhttp.responseText);

            //Prendo il corpo del div in cui inserire i campi
            var divbody = document.getElementById( elementId: 'shopContainer');
            divbody.innerHTML = "";

            //Inserisco tutti le righe con i relativi dati.
            for (var i = 0; i < obj.length; i++) {

                //Prendo i div principale per i prodotti
                var divContainer = document.createElement( tagName: "div");
                divContainer.setAttribute( qualifiedName: "class", value: "col-12 col-sm-6 col-lg-4");
                var divWrapper = document.createElement( tagName: "div");
                divWrapper.setAttribute( qualifiedName: "class", value: "single-product-wrapper");

                //Creo un nuovo div che contiene le informaazioni e lo metto nel padre
                var div = document.createElement( tagName: "div");
                div.setAttribute( qualifiedName: "class", value: "product-description");
                // noinspection JSDuplicatedDeclaration
                var a = document.createElement( tagName: "a");
                a.setAttribute( qualifiedName: "href", value: "#");
                var h6 = document.createElement( tagName: "h6");
                h6.style = "margin-left: 10px";
                h6.setAttribute( qualifiedName: "id", value: obj[i]['nome'] + "." + obj[i]['indirizzo'] + "." + obj[i]['citta'])
                h6.setAttribute( qualifiedName: "onclick", value: "shopDetails(this)");
                h6.appendChild(document.createTextNode(obj[i]['nome']));
                a.appendChild(h6);
                div.appendChild(a);
```

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

                //aggiungo tutti i div a quello primario
                divContainer.appendChild(divWrapper);
                divbody.appendChild(divContainer);
            }
        }
    };
    xhttp.open( method: "POST", url: "/gestionevendita2018/admin/getShops", async: true);
    xhttp.send();
}
```

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.

```
    public function getShops()
    {
        //Connetto al database
        $conn = $this->connection->sqlConnection();

        //Prendo i dati dei negozi
        $sql = $conn->prepare("SELECT nome, indirizzo, citta FROM negozio");

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

        $dataArray = array();
        //Se ci sono dei valori
        if ($sql->rowCount() > 1) {
            // Ciclo tutti i valori
            while ($row = $sql->fetch()) {

                array_push( &array: $dataArray, $row);
            }
        } else if ($sql->rowCount() == 1) {
            array_push( &array: $dataArray, $sql->fetch());
        }
        $conn = null;
        return $dataArray;
    }
```

Figura 6 model getShops che prende le informazioni dei negozi

```php
public function getShops(){
    //COntrollo che la funzione entri in POST
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        //COntrollo che sia aperta la sessione e abbia fatto il login un amministratore
        if (isset($_SESSION['admin'])) {
            //Prendo la classe model
            require_once 'application/models/shop.php';
            $shop = new ShopModel();

            $shops = $shop->getShops();

            //Stampo con json i valori dei coach
            header( string: 'Content-Type: application/json');
            echo json_encode($shops);
        }else{
            header( string: "location: javascript://history.back()");
        }
    }else{
        header( string: "location: javascript://history.back()");
    }
}
```

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 shopAdress = document.getElementById( elementId: "modShopAddress");
252              shopAdress.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]['nome']);
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.

```php
public function getShopDealer(){
    //COntrollo che la funzione entri in POST
    if ($_SERVER["REQUEST_METHOD"] == "POST") {

        //COntrollo che sia aperta la sessione e abbia fatto il login un amministratore
        if (isset($_SESSION['admin'])) {

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

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

            //Se i campi che devono comparire non sono vuoti
            if ($name != null && $address != null && $city != null) {
                $shops = $shop->getShopDealer($name, $address, $city);
            }

            //Stampo con json i valori dei coach
            header( string: 'Content-Type: application/json');
            echo json_encode($shops);
        }else{
            header( string: "location: javascript://history.back()");
        }
    }else{
        header( string: "location: javascript://history.back()");
    }
}
```

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

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

    //Prendo i dati dei negozi
    $sql = $conn->prepare("SELECT * FROM negozio n
                           right outer join gestore g on g.email = n.email_gestore
                           WHERE n.nome LIKE :nameR AND n.indirizzo LIKE :addressR AND n.citta LIKE :cityR
                           union
                           SELECT * FROM negozio n
                           left outer join gestore g on g.email = n.email_gestore
                           WHERE n.nome LIKE :nameL AND n.indirizzo LIKE :addressL AND n.citta LIKE :cityL");
    $sql->bindParam(':nameR', $name, PDO::PARAM_STR);
    $sql->bindParam(':addressR', $address, PDO::PARAM_STR);
    $sql->bindParam(':cityR', $city, PDO::PARAM_STR);
    $sql->bindParam(':nameL', $name, PDO::PARAM_STR);
    $sql->bindParam(':addressL', $address, PDO::PARAM_STR);
    $sql->bindParam(':cityL', $city, PDO::PARAM_STR);

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

    $dataArray = array();
    //Se ci sono dei valori
    if ($sql->rowCount() > 1) {
        // Ciclo tutti i valori
        while ($row = $sql->fetch()) {

            array_push( &array: $dataArray, $row);
        }
    } else if ($sql->rowCount() == 1) {
        array_push( &array: $dataArray, $sql->fetch());
    }
    $conn = null;
    return $dataArray;
}
```

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.

```php
require_once 'application/models/shop.php';
$shop = new ShopModel();
require_once 'application/models/dealer.php';
$dealer = new Dealer();

//Prendo le variabili passate dal POST del negozio
$shopName = isset($_POST["modShopName"])? $_POST["modShopName"] : null;
$shopAddress = isset($_POST["modShopAddress"])? $_POST["modShopAddress"] : null;
$shopCity = isset($_POST["modShopCity"])? $_POST["modShopCity"] : null;
$shopPhone = isset($_POST["modShopPhone"])? $_POST["modShopPhone"] : null;

//Prendo le variabili passate dal POST del venditore
$name = isset($_POST["modName"])? $_POST["modName"] : null;
$surname = isset($_POST["modSurname"])? $_POST["modSurname"] : null;
$phone = isset($_POST["modPhone"])? $_POST["modPhone"] : null;
$mail = isset($_POST["modMail"])? $_POST["modMail"] : null;

//Prendo i vecchi valori delle chiavi passati dal POST
$oldMail = isset($_POST["oldMail"])? $_POST["oldMail"] : null;
$oldShopName = isset($_POST["oldShopName"])? $_POST["oldShopName"] : null;
$oldShopAddress = isset($_POST["oldShopAddress"])? $_POST["oldShopAddress"] : null;
$oldShopCity = isset($_POST["oldShopCity"])? $_POST["oldShopCity"] : null;

//Se i campi che devono comparire non sono vuoti
if ($mail != null && $name != null && $surname != null && $phone != null && $oldMail != null) {
    //Inserisco i dati del venditore
    $dealer->modifyDealer($mail, $name, $surname, $phone, $oldMail);
}

//Se i campi che devono comparire non sono vuoti
if ($shopName != null && $shopAddress != null && $shopCity != null && $shopPhone != null && $mail != null && $oldShopName !
    //Inserisco i dati del negozio
    $shop->modifyShop($shopName, $shopAddress, $shopCity, $shopPhone, $mail, $oldShopName, $oldShopAddress, $oldShopCity);
}

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.

```php
public function modifyDealer($email, $name, $surname, $phone, $oldMail){
    //Connetto al database
    $conn = $this->connection->sqlConnection();

    //Prendo i dati dell'utente in base alla mail
    $sql = $conn->prepare("UPDATE gestore set email = :email, nome = :name, cognome = :surname, telefono = :phone
                WHERE email LIKE :emailW");
    $sql->bindParam(':email', $email, PDO::PARAM_STR);
    $sql->bindParam(':name', $name, PDO::PARAM_STR);
    $sql->bindParam(':surname', $surname, PDO::PARAM_STR);
    $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
    $sql->bindParam(':emailW', $oldMail, PDO::PARAM_STR);

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

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

```php
public function modifyShop($name, $address, $city, $phone, $email, $oldName, $oldAddress, $oldCity){
    //Connetto al database
    $conn = $this->connection->sqlConnection();

    //Prendo i dati dell'utente in base alla mail
    $sql = $conn->prepare("UPDATE negozio set nome = :name, indirizzo = :address, citta = :city, telefono = :phone, email_gestore = :emai
                WHERE nome LIKE :nameW AND indirizzo LIKE :addressW AND citta LIKE :cityW");
    $sql->bindParam(':name', $name, PDO::PARAM_STR);
    $sql->bindParam(':address', $address, PDO::PARAM_STR);
    $sql->bindParam(':city', $city, PDO::PARAM_STR);
    $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
    $sql->bindParam(':email', $email, PDO::PARAM_STR);
    $sql->bindParam(':nameW', $oldName, PDO::PARAM_STR);
    $sql->bindParam(':addressW', $oldAddress, PDO::PARAM_STR);
    $sql->bindParam(':cityW', $oldCity, PDO::PARAM_STR);

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

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.

```php
public function doCheckout(){
    //Se la sessione è aperta apro le pagine altrimeenti no
    if(isset($_SESSION['customer'])) {

        //Prendo le variabili passate dal POST
        $name = isset($_POST["name"])? $_POST["name"] : null;
        $price = isset($_POST["price"])? $_POST["price"] : null;
        $quantity = isset($_POST["quantity"])? $_POST["quantity"] : null;
        $totPrice = isset($_POST["totPrice"])? $_POST["totPrice"] : null;
        $cartObjects = isset($_POST["cartObjects"])? $_POST["cartObjects"] : null;
        $pickupShop = isset($_POST["pickupShop"])? $_POST["pickupShop"] : null;

        if($name != null && $price != null && $quantity != null && $totPrice != null && $cartObjects != null && $pickupShop != null)

            //Prendo i dati del negozio di ritiro
            $pickupShopInfos = explode( delimiter: ".", $pickupShop);

            require('application/fpdf/fpdf.php');

            // crea l'istanza del documento
            $pdf = new FPDF();
            $pdf->AddPage();

            //---Header---
            // Arial bold 15
            $pdf->SetFont('Arial', 'B', 15);
            // Dati utente
            $pdf->Cell(30, 10, $_SESSION['customer'], 0, 0, 'L');
            //Data odierna
            $pdf->Cell(130);
            $pdf->Cell(30, 10, date( format: "Y/m/d"), 0, 0, 'R');
            // Vado a capo
            $pdf->Ln(20);
```

```php
$pdf->SetFont('Arial', 'B', 30);
// Muove verso destra
$pdf->Cell(80);
// Titolo in riquadro
$pdf->Cell(30, 10, 'Il suo ordine', 0, 0, 'C');
// Interruzione di linea
$pdf->Ln(20);

//Metto l'indice
// Seleziona Arial 15
$pdf->SetFont('Arial', '', 16);
// Indice
$pdf->Cell(30, 10, 'Prodotto', 0, 0, 'L');
$pdf->Cell(50);
$pdf->Cell(30, 10, 'Quantità', 0, 0, 'C');
$pdf->Cell(50);
$pdf->Cell(30, 10, 'Prezzo', 0, 0, 'C');
// Interruzione di linea
$pdf->Ln(20);

//Traccio una linea
$pdf->SetDrawColor(180, 180, 180);
$pdf->SetLineWidth(1);
$pdf->Line(10, 70, 200, 70);
// Interruzione di linea
$pdf->Ln(10);

//Variabile per la prossima riga
$nextLine = 70;
for ($i = 0; $i < count($name); $i++) {
    //Inserisco i prodotti
    $pdf->Cell(30, 10, $name[$i], 0, 0, 'L');
    $pdf->Cell(50);
    $pdf->Cell(30, 10, $quantity[$i], 0, 0, 'C');
    $pdf->Cell(50);
    $pdf->Cell(30, 10, $price[$i] . 'Fr.', 0, 0, 'C');
    // Interruzione di linea
    $pdf->Ln(20);
    $nextLine += 21;
}

//Traccio una linea
$pdf->SetDrawColor(180, 180, 180);
$pdf->SetLineWidth(1);
$pdf->Line(10, $nextLine, 200, $nextLine);
// Interruzione di linea
$pdf->Ln(10);

//Metto il totale
// Seleziona Arial 15
$pdf->SetFont('Arial', '', 16);
// Indice
$pdf->Cell(30, 10, 'Totale', 0, 0, 'L');
$pdf->Cell(50);
$pdf->Cell(30, 10, $cartObjects, 0, 0, 'C');
$pdf->Cell(50);
$pdf->Cell(30, 10, $totPrice . 'Fr.', 0, 0, 'C');
// Interruzione di linea
$pdf->Ln(10);
$nextLine += 30;
$pdf->SetDrawColor(180, 180, 180);
$pdf->SetLineWidth(1);
$pdf->Line(10, $nextLine, 200, $nextLine);
// Interruzione di linea
$pdf->Ln(10);

//Metto il totale
// Seleziona Arial 15
$pdf->SetFont('Arial', '', 16);
// Indice
$pdf->Cell(30, 10, 'Negozio', 0, 0, 'L');
$pdf->Cell(50);
$pdf->Cell(30, 10, $pickupShopInfos[0], 0, 0, 'C');
// Indirizzo
$pdf->Ln(10);
$pdf->Cell(80);
$pdf->Cell(30, 10, $pickupShopInfos[1], 0, 0, 'C');
$pdf->Cell(25);
$pdf->Cell(30, 10, $pickupShopInfos[2], 0, 0, 'C');
// Contatti
$pdf->Ln(10);
$pdf->Cell(80);
$pdf->Cell(30, 10, $pickupShopInfos[3] ."." $pickupShopInfos[4], 0, 0, 'C');
$pdf->Cell(25);
$pdf->Cell(30, 10, $pickupShopInfos[5], 0, 0, 'C');
// Interruzione di linea
$pdf->Ln(10);
$nextLine += 30;
```

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

a@a.a 2019/04/08

Il suo ordine

Prodotto	QuantitÃ	Prezzo
cane	1	2Fr.
corda	2	10Fr.
Totale	3	22Fr.

Negozio
Ritira tutto
Via gelsomino 1 Lugano
ritutto@gmail.com 0912839482

Progetto vendita piccoli negozi

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.