

Relazione di Tecnologie Web

Michele Lorenzo

2021/2022

Indice

1	Introduzione	2
2	Funzionalità	3
2.1	Login/Logout	3
2.2	Registrazione	3
2.3	Gestione del contenuto dell'utente	4
3	Caratteristiche	6
3.1	Usabilità	6
3.2	Interazione/Animazione	6
3.3	Sessioni	6
3.4	Interrogazioni del DB	7
3.5	Validazione dei dati di input	7
3.6	Sicurezza	8
3.7	Presentazione	9
4	Front-end	10
5	Back-end e comunicazione front/back-end	11
5.1	Schema del DB	11
5.2	Descrizione delle funzioni remote	11

1 Introduzione

Il tema scelto per la realizzazione del progetto di Tecnologie Web è quello di un sito e-commerce chiamato 'reBuy', dove l'utente può vendere e comprare telefoni usati previa registrazione.

Il sito è strutturato in diverse pagine che presentano, ad eccezione del login e della registrazione, una barra di navigazione che permette di navigare tra di esse:

- Pagina home
- Pagina di login
- Pagina di registrazione
- Carrello della spesa
- Pagina contenente la lista degli ordini
- Pagina per aggiungere un nuovo prodotto
- Pagina di conferma dell'ordine

Tutte le pagine includono i file 'header.html' e 'footer.html'.

2 Funzionalità

2.1 Login/Logout

Prima di poter usufruire delle pagine del sito bisogna effettuare un login tramite mail, password e tipo di utente (buyer o seller).

Ogni pagina del sito include un banner che effettua una query Ajax al server per controllare se l'utente ha effettuato l'accesso, lato server questo avviene controllando se l'id dell'utente è impostato nella sessione corrente. Se è stata spuntata in fase di login la checkbox 'Remember me', l'id dell'utente viene memorizzato nei cookies permettendo così di non effettuare l'accesso ogni volta (la sessione viene inizializzata in automatico sfruttando tale id).

```
if (isset($_COOKIE["userId"])) {  
    $user = new User();  
  
    try {  
        $user->selectFromDBWithId($_COOKIE["userId"]);  
        Util::sessionClean();  
  
        $_SESSION["id"] = $user->getId();  
        $_SESSION["firstName"] = $user->getFirstName();  
        $_SESSION["isSeller"] = $user->getIsSeller();  
  
        Util::printJson(array("firstName" => $user->getFirstName(),  
                               "isSeller" => $user->getIsSeller())  
        );  
    } catch (PDOException $e) {  
        Util::printJson(  
            array("error"=>"500: Server Error:" . $e->getMessage())  
        );  
    }  
}
```

Per effettuare il logout è presente un tasto apposito nel banner che effettua la seguente operazione per pulire e reinizializzare la sessione:

```
if (isset($_SESSION)) {  
    session_unset();  
    session_destroy();  
    session_start();  
}
```

2.2 Registrazione

La registrazione avviene tramite un apposito form presente nella pagina 'sign-up.php' dove viene chiesto di inserire: nome, cognome, data di nascita, sesso, email, numero di telefono, tipo di utente (Buyer o Seller) e password. Se la validazione dei dati è andata a buon fine i dati vengono inviati in formato json


al server tramite una query Ajax e sfruttando il metodo POST. Il server a sua volta effettua nuovamente un controllo dei dati e li inserisce nel database:

```
INSERT INTO users (first_name, last_name, birthday, gender,
email, phone_number, seller, users.password) VALUES (?, ?, ?, ?,
?, ?, ?, ?)
```

In caso di dati non validi o di errori, l'utente sarà informato attraverso appositi alert.

2.3 Gestione del contenuto dell'utente

Il contenuto degli utenti può essere aggiunto tramite un'apposita pagina 'new-product.php' che si occuperà di validare e inserire i dati del prodotto nel database. Può essere eliminato, nel caso in cui non fosse già stato venduto, tramite 'added-products-list.php' che stampa a video tutti i prodotti aggiunti dall'utente con l'apposito tasto:

	Samsung S21 VF456JNHG 6/128, Like New	699.00€	Remove
------------------------------------------------------------------------------------	---------------------------------------------	---------	------------------------

```
static function deleteProduct($id) {
    $conn = DBConfig::DBConnect();
    $qId = $conn->quote($id);
    $conn->exec("DELETE FROM products WHERE id = $qId");
}
```

Gli ordini degli utenti sono memorizzati in una apposita tabella nel database e, i prodotti acquistati in tale ordine, vengono associati ad esso. E' possibile visualizzare gli ordini nella pagina 'orders.php'.

Nella pagina home è possibile visualizzare tutti i prodotti venduti con la possibilità di applicare dei filtri di ricerca. Codice php per selezionare i prodotti applicando i filtri:

```
$conn = DBConfig::DBConnect();
$filters = array();
$where = " WHERE price BETWEEN " . $priceMin . " AND " . $priceMax;

if (!empty($brand)) $filters[] = " brand = " . $conn->quote($brand);
if (!empty($condition))
    $filters[] = " products.condition = " . $conn->quote($condition);
if (!empty($name))
    $filters[] = " name LIKE "
        . $conn->quote("%" . htmlspecialchars($name) . "%");
if (!empty($excludeOrdered)) $filters[] = " products.order IS NULL";

foreach ($filters as $filter) {
```

```
    $where .= (" AND" . $filter);
}

$stmt = $conn->prepare(
    "SELECT products.id, brand,
       products.name,
       model,
       year_of_production,
       img,
       size,
       users.first_name,
       users.last_name,
       products.condition,
       price
    FROM products JOIN users ON products.seller = users.id"
    . $where
);

$stmt->execute();
$this->productList = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

3 Caratteristiche

3.1 Usabilità

In 'reBuy' ad una azione dell'utente con il sito corrisponde sempre un feedback che permette di capire se l'azione è andata a buon fine oppure si è verificato un errore, spesso gli errore sono segnalati da alert rossi:

⚠ Please upload an image first.

Gli errori di validazione dei form, invece, sono segnalati mostrando un messaggio con testo di colore rosso sotto l'input che ha causato tale errore:

Name

This field is required.

3.2 Interazione/Animazione

Nel sito sono presenti diverse animazioni che sono state realizzate tramite i metodi messi a disposizione da jQuery:

- **fadeOut**: quando un prodotto viene eliminato dal carrello o dalla lista di prodotti da vendere viene applicato un effetto di dissolvenza.
- **slideToggle**: quando nella lista degli ordini vogliamo visualizzare i prodotti di uno specifico ordine, cliccando sulla freccia apposita, i prodotti vengono visualizzati con un effetto lento di slide verso il basso, viceversa se si clicca nuovamente la freccia.
- **animate**: quando aggiungiamo al carrello della spesa un nuovo prodotto viene applicata un'animazione che cambia l'opacità del prodotto, seguita dal cambiamento dello sfondo e dell'icona del pulsante.
Quando cambiamo pagina nella lista di prodotti viene applicata un'animazione che riporta l'utente in cima alla pagina.

```
$('html, body').animate({ scrollTop: 0 }, 'fast');
```

3.3 Sessioni

Ogni pagine del sito a un suo controller php, ogni controller prima di effettuare qualsiasi operazione richiama il metodo 'sessionStart()' della classe 'Util' che inizializza la sessione:

```
if(!isset($_SESSION)) session_start();
```

3.4 Interrogazioni del DB

Il sito presenta vari form che permettono di effettuare query di inserimento oppure query di selezione molto complesse sui prodotti, come quelle riportate nella sezione precedente di questo documento.

Tutte le query sono state eseguite tramite la classe PDO di php e i risultati, se ce ne sono, sono stati memorizzati in oggetti appositi (product, orderList, ecc.). Alcune delle query utilizzate nel sito:

```
/* Query per aggiungere un prodotto a un ordine */
UPDATE products SET products.order = ? WHERE id = ?;

/* Query per selezionare tutti i prodotti non venduti */
SELECT products.id, brand,
       products.name,
       model,
       year_of_production,
       img,
       size,
       users.first_name,
       users.last_name,
       products.condition,
       price
FROM products JOIN users ON products.seller = users.id
WHERE products.order IS NULL;
```

3.5 Validazione dei dati di input

La validazione dei dati di input è stata eseguita sia lato client che lato server. Per la validazione lato client è stata usata una libreria chiamata 'jQuery validation' che mi ha permesso di eseguire controlli accurati sui dati inseriti da parte dell'utente:

```
$("#form[name='signUp']").validate({
  rules: {
    firstName: "required",
    lastName: "required",
    email: {
      required: true,
      email: true
    },
    birthdayDate: "required",
    phoneNumber: {
      required: true,
      number: true
    },
    password: {
      required: true,
      minlength: 8,
      maxlength: 16
    }
  }
});
```

```

    },
    passwordCheck: {
        required: true,
        equalTo: "#password"
    }
},
messages: {
    passwordCheck: {
        equalTo: "Please enter the same password."
    },
    termsOfUse: "Please accept the Terms & Condition before submitting."
},
errorClass: "text-danger",
submitHandler: form => {
    if (checkData())
        signUp();
}
});

```

Lato server sono stati implementati ulteriori controlli sui dati passati dal client, questi controlli sono meno complessi rispetto a quelli effettuati lato client.

```

function checkData() {
    return !empty($this->firstName)
        && !empty($this->lastName)
        && !empty($this->birthday)
        && !empty($this->gender)
        && !empty($this->email)
        && !empty($this->phoneNumber)
        && !empty($this->password)
        && filter_var($this->email, FILTER_VALIDATE_EMAIL);
}

```

3.6 Sicurezza

Prima di inserire i dati degli utenti all'interno di query sql, vengono effettuati dei controlli per evitare attacchi come XSS e SQL injection. Per tali controlli sono stati utilizzati metodi php come 'htmlspecialchars' e della classe PDO il metodo 'quote'.

```

# htmlspecialchars
function sanitize() {
    $this->firstName = htmlspecialchars($this->firstName);
    $this->lastName = htmlspecialchars($this->lastName);
    $this->gender = htmlspecialchars($this->gender);
    $this->email = filter_var($this->email, FILTER_SANITIZE_EMAIL);
    $this->phoneNumber = htmlspecialchars($this->phoneNumber);
    $this->password = htmlspecialchars($this->password);
}

```



```
# quote
$conn = DBConfig::DBConnect();

$qFirstName = $conn->quote($this->firstName);
$qLastName = $conn->quote($this->lastName);
$qBirthday = $conn->quote($this->birthday);
$qGender = $conn->quote($this->gender);
$qEmail = $conn->quote($this->email);
$qPhoneNumber = $conn->quote($this->phoneNumber);
$qIsSeller = $conn->quote($this->isSeller);
$qPassword = $conn->quote($this->password);
```

Per garantire la sicurezza degli utenti, le password sono state memorizzate nel database come codici hash utilizzando l'algoritmo 'bcrypt' e il metodo php 'password_hash'.

3.7 Presentazione

Il sito è stato realizzato utilizzando le classi css di 'bootstrap 5' e le icone di 'bootstrap icons', questo ha permesso di realizzare un sito con un layout chiaro, intuitivo, flessibile e responsive.

Le pagine principali del sito adottano colori chiari tendenti al bianco con scritte nere per non comprometterne la leggibilità, mentre bottoni, link, footer e barra di navigazione adottano colori più scuri.

Welcome to Rebuy store

On our site you can sell and buy used phones.

PRODUCT SEARCH

Brand

Condition


Model Name

Min Max

€ €

Apply Filters

Clear all



Samsung S21

model: VF456JNHG

Year of production: 2018


size: 6/128 GB

seller: Michele Lorenzo

condition: Like New

699.00€

Add to cart



Samsung S21

model: VF456JNHG

Year of production: 2018


size: 6/128 GB

seller: Michele Lorenzo

condition: Like New

699.00€

Add to cart



Samsung S21

model: VF456JNHG

Year of production: 2018

size: 6/128 GB

seller: Michele Lorenzo

condition: Like New

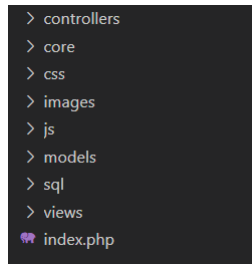
699.00€

Add to cart

4 Front-end

E' stato adottato uno stile unobtrusive, quindi si è prestata particolare attenzione nella divisione di presentazione, contenuto e comportamento. Lo scambio di dati tra client e server avviene solamente tramite query Ajax e i dati sono scambiati utilizzando il formato Json.

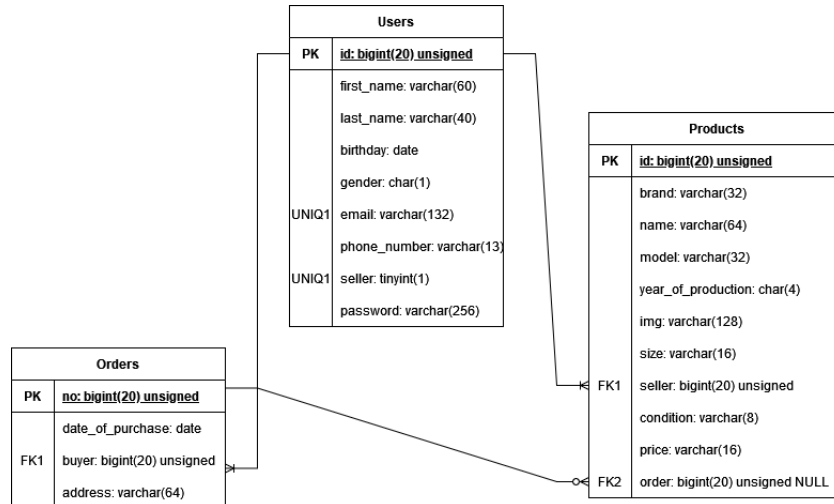
Le cartelle sono state organizzate secondo il pattern MVC:



- **controllers:** contiene i controller php che gestiscono le richieste Ajax inviate dalla corrispondente pagina javascript lato client.
- **core:** contiene classi utili per il server che vengono richiamate più volte all'interno del codice back-end, come ad esempio dati per la connessione con il DBMS.
- **css:** foglio di stile del sito (presentazione).
- **images:** contiene le immagini utilizzate per la realizzazione del sito e una cartella che contiene le immagini caricate dall'utente.
- **js:** file javascript per la gestione delle corrispondenti viste (comportamento).
- **models:** contiene i modelli degli oggetti presenti nel sito (prodotti, ordini ecc.) e i metodi per interagire con il database.
- **sql:** backup dei dati del database.
- **views:** contiene le viste (le pagine) del sito, all'interno è presente una cartella 'html' che contiene codice HTML richiamato in più pagine (header, banner ecc.).

5 Back-end e comunicazione front/back-end

5.1 Schema del DB



5.2 Descrizione delle funzioni remote

Ogni pagina ha il suo controller che gestisce diverse query Ajax e i modelli si occupano delle interazioni con il database. Nel seguente elenco verranno mostrate le richieste Ajax per ogni controller e le corrispondenti risposte da parte del server. In caso di errore viene lanciata un'eccezione, 'PDOException' nel caso di errori con il DB e 'InvalidArgumentException' nel caso in cui la validazione non termina con successo. Le eccezioni vengono catturate nei blocchi 'try-catch' e restituite in formato json:

```
catch (PDOException $e) {
    Util::printJson(array("error" => "500: Server Error: " . $e->getMessage()));
}
```

- **AddedProductListController.php**: presenta due funzioni. 'Get products' che restituisce la lista dei prodotti dell'utente. Output:

```
[{ id: 30,
  brand: "Samsung",
  name: "S21",
  model: "VF456JNHG",
  year_of_production: "2018",
  img: "images/products/Samsung_S21.png",
  size: "6/128",
  seller: 9,
  condition: "Like New",
  price: "699.00",
  order: null }]
```

'Delete product' che permette di eliminare un prodotto in vendita. Output:

```
{ response: '200: OK' }
```

- **BannerController.php:** presenta due funzioni. 'Check login' che permette di controllare se l'utente è loggato. Output:

```
{  firstName: 'Michele',  
    isSeller: 1 }
```

'Logout' pulisce la sessione corrente ed effettua il redirect alla pagina home. Output:

```
{ redirect: 'views/home-page.php' }
```

- **HomeController.php:** presenta tre funzioni. 'Get products' restituisce tutti i prodotti. Output identico a 'get products' di 'AddedProductList.php'.
'Add cart' che aggiunge un prodotto al carrello della spesa. Non viene restituito nessun output.
'Filter' restituisce tutti i prodotti con opportuni filtri applicati. Output identico a 'get products' di 'AddedProductList.php'.
- **LoginController.php:** crea un nuovo utente, controlla e sanifica i dati, inserisce i dati all'interno del DB ed effettua il redirect. Output identico a quello di 'logout' in 'BannerController.php'.
- **NewProductController.php:** presenta tre funzioni. 'Check is seller' controlla se l'utente è un venditore, se non lo è viene effettuato un redirect. Se l'utente è un venditore l'output è identico a 'delete product' in 'AddedProductListController.php', se invece non lo è, l'output è identico a 'logout' in 'BannerController.php'.
'Add product' crea un nuovo prodotto, controlla e sanifica i dati e lo aggiunge al DB. Output:

```
{ success: 'Product added successfully' }
```

L'ultima funzione serve per il caricamento dell'immagine all'interno della cartella 'images/products' del server. Output:

```
{ success: 'File upload successful' }
```

- **OrdersController.php:** presenta due funzioni. 'Get orders' carica gli ordini dal DB in base all'utente e li restituisce. Output:

```
[{ no: 2,  
  date_of_purchase: "2022-08-23",  
  buyer: 9,  
  address: "Via Pessinetto 12, 10149, Torino"  }]
```

'Get products' carica i prodotti dal DB in base al numero dell'ordine e li restituisce. Output identico a 'get products' di 'AddedProductList.php'.

- **ShoppingCartController.php:** presenta tre funzioni. 'Redirect' crea un nuovo ordine, controlla e sanifica i dati e li inserisce nel DB. Successivamente aggiorna il numero dell'ordine in tutti i prodotti acquistati ed effettua il redirect alla pagina di conferma dell'ordine. Output:

```
{ redirect: 'views/confirmed-order.php' }
```

'Get products' carica dal DB i prodotti in base all'id e li restituisce. Output identico a 'get products' di 'AddedProductList.php'.

'Delete product' cancella un prodotto dal carrello e dai cookies. Non viene restituito nessun output.

- **SignUpController.php:** Crea un nuovo utente, controlla e sanifica i dati, li inserisce nel DB ed effettua un redirect alla pagina di login. Output:

```
{ redirect: 'views/login.php' }
```