Stefano Chiavazza matricola: 835588

Tema del sito: Il sito offre agli utenti la possibilità di comprare e vendere oggetti. Ogni utente può quindi sia vendere che comprare.

Profilo: Per ogni utente è presente una pagina di profilo. In questa pagina vengono mostrate le informazioni di base sull'utente (username, e-mail e breve descrizione). Le informazioni posso essere cambiate. Vengono poi presentati tutti i prodotti messi in vendita dall'utente e le recensioni che l'utente ha ricevuto.

Prodotto: Ogni prodotto ha un titolo, e un prezzo (deciso dall'utente che lo mette in vendita). Inoltre, può anche avere una breve descrizione e un'immagine. L'utente che ha messo in vendita il prodotto può modificare queste informazioni anche in un secondo momento. Gli altri utenti possono aggiungere il prodotto al carello se non è ancora stato comprato da qualche altro utente.

Recensione: Un utente può scrivere una recensione su un altro utente (solo una). La recensione consiste in un voto da 0 a 5 e da un breve testo (opzionale). L'utente che ha scritto la recensione può rimuoverla se lo desidera.

Carrello: L'utente può aggiungere prodotti al proprio carrello se non sono già stati acquistati e può poi decidere di acquistare i prodotti aggiunti.

Accesso: Per registrarsi l'utente deve fornire username e-mail e password. Username e password non possono essere già utilizzare da un altro utente. Per accedere poi basta inserire username e password.

Ricerca: È possibile cercare dei prodotti. Verranno mostrati tutti i prodotti che nel titolo contengono la stringa cercata. È possibile poi raffinare la ricerca specificando prezzo minimo e massimo e l'ordine in cui mostrare i prodotti (prezzo crescente o decrescente).

Database:

Il database è composto da quattro tabelle: Carts, Products, Users e userReviews.

La tabella Users contiene le informazioni sugli utenti registrati ed è così definita:

```
Users (
        ID int(11) NOT NULL AUTO_INCREMENT,
        Username varchar(32) NOT NULL,
        Password varchar(128) NOT NULL,
        email varchar(128) NOT NULL,
        reg_date date NOT NULL,
        image varchar(1024),
        info varchar(1024),
        PRIMARY KEY (ID),
        UNIQUE KEY username,
        UNIQUE KEY email
)
```

La tabella Products contiene le informazioni sui prodotti:

```
Products (
        `ID` int(11) NOT NULL AUTO_INCREMENT,
        `title` varchar(128) NOT NULL,
        `description` varchar(1024) DEFAULT NULL,
```

```
`price` int(11) NOT NULL,
`image` varchar(1024) DEFAULT NULL,
`ownerID` int(11) NOT NULL,
`sold` tinyint(1) DEFAULT '0',
PRIMARY KEY (`ID`),
KEY `ownerID` (`ownerID`),
FOREIGN KEY (`ownerID`) REFERENCES `Users` (`ID`)
```

)

La tabella Carts contiene le informazioni sui carrelli degli utenti:

Carts (

```
`productID` int(11) NOT NULL,
`userID` int(11) NOT NULL,
PRIMARY KEY (`productID`,`userID`),
KEY `userID` (`userID`),
CONSTRAINT `Carts_ibfk_1` FOREIGN KEY (`productID`) REFERENCES `Products` (`ID`) ON DELETE    CASCADE,
CONSTRAINT `Carts_ibfk_2` FOREIGN KEY (`userID`) REFERENCES `Users` (`ID`) ON DELETE CASCADE
```

)

Infine, la tabella userReview contiene le recensioni degli utenti:
userReviews (

```
`writerID` int(11) NOT NULL,
`reviewedID` int(11) NOT NULL,
`text` varchar(1024) DEFAULT NULL,
`rating` int(11) NOT NULL,
`pub_date` date NOT NULL,
PRIMARY KEY (`writerID`,`reviewedID`),
KEY `reviewedID` (`reviewedID`),
CONSTRAINT  FOREIGN KEY (`writerID`) REFERENCES `Users` (`ID`),
CONSTRAINT  FOREIGN KEY (`reviewedID`) REFERENCES `Users` (`ID`)
```

)

Backend:

index: Quando non viene richiesto nessun dato specifico il server ritorna il file index.html con i relativi fogli di stile e codici javaScript. Tutto l'html e il javascript per il sito sono giá contenuti all'interno. Le altre chiamate infatti ritornano solo dati in formato json.

Ajax: tutte le chiamate ajax ritornano un file JSON della seguente forma:

```
{
    state: 0,            // 0 ha avuto successo, 1 la richiesta non è andata a buon fine
    message: "Success",      //un messaggio, che può essere di errore o no
    data:  {…}               //eventuali dati ritornati dalla richiesta
}
```

'ajax/login': metodo POST, parametri: **username, password**
se la combinazione di username a password esiste restituisce:

```
{
    "state": 0,
```

```
        "message": "User info",
        "data":    {
                "ID": "2",
                "username": "stefano",
                "email": "Stefano@mail.com",
                "date": "2018-12-16",
                "rating": null,
                "image": "image.jpg",
                "info": "description"
        }
}
```

'ajax/signup': metodo POST, parametri: **username, password, e-mail**

se l'inserimento è andato a buon fine restituisce le informazioni sull'utente esattamente come per login.

'ajax/logout': metodo GET

Cancella i dati di sessione e ritorna un messaggio di successo

```
{
        "state": 0,
        "message": "Logout successful",
        "data": null
}
```

'ajax/profile'?id=2: metodo GET, parametri: **id**

se esiste un utente con l'id specificato ritorna le informazioni sull'utente
se non è specificato nessun id, ma l'utente è autenticato tramite sessione ritorna le informazioni sull'utente autenticato (stesso formato di login e signup).

'ajax/profile/update': metodo POST, parametri: **userID, username, e-mail, image, info**

se l'id dell'utente da modificare è uguale al id memorizzato nella sessione, aggiorna le informazioni sull'utente con quelle passate e ritorna le nuove informazioni in formato JSON.

'ajax/products?id=2: metodo GET, parametri: **id**

se l'id corrisponde ad un utente ritorna tutti i prodotti che ha messo in vendita l'utente
sintassi dell'output JSON:

```
{
        "state": 0,
        "message": "List of products",
        "data":    [
                {
                "     ID": "36",
                "     title": "DELL laptop",
                "     description": "Old dell laptop but still functioning",
                "     price": "200",
                "     ownerID": "2",
                "     ownerName": "stefano",
                "     image": "image.jpg",
                "     sold": false
                },
                {...} . . .

        ]
}
```

‘ajax/products/add’: metodo POST, parametri: **title, price, [description], [image]**

se l’utente è autenticato aggiungi alla lista dei prodotti in vendita dall’utente un prodotto con le informazioni passate. Title e price sono necessari, le altre informazioni sono opzionali.
Se l’inserimento ha avuto successo ritorna un messaggio di successo, altrimenti ritorna un messaggi che descrive l’errore riscontrato. Non ritorna dati.

‘ajax/products/delete’: metodo POST, parametri: **productID**

se l’utente è autenticato tramite sessione e il prodotto corrispondente al id passato è associato al suo account, rimuove il prodotto. In caso di errore ritorna un messaggio che descrive l’errore.

‘ajax/products/update’: metodo POST, parametri: **ID, title, price, description, image, sold**

se l’utente è autenticato e ha il permesso, modifica le informazioni del prodotto corrispondete a ID. Ritorna un messaggio di successo o di errore se la modifica dei dati non è andata a buon fine.

‘ajax/products/search?search=phone&maxPrice=300&minPrice=120&desc=1’: metodo GET, parametri: **search, [maxPrice], [minPrice], [desc]**

Ritorna la lista di prodotti che nel titolo contengono la stringa *search* e che rispettano i gli eventuali limiti di prezzo imposti. La lista può essere ordinata in ordine crescente o decrescente in base al prezzo. Ritorna una lista con lo stesso formato di ‘ajax/products?id=2’ e un messaggio in caso di errore.

‘ajax/cart?userID=2’: metodo GET, parametri: **userID**

se *userID* corrisponde al id dell’utente autenticato, allora ritorna le informazioni sui prodotti nel carrello. I prodotti hanno lo stesso formato JSON di *‘ajax/products?id=2’.*

‘ajax/cart/add’: metodo POST, parametri: **userID, productID**

se *userID* corrisponde al id di sessione, quindi l’utente è autenticato e sta modificando il proprio carrello allora aggiunge il prodotto corrispondente a *productID* al carrello. Un utente no può aggiungere un proprio prodotto o un prodotto già venduto. Ritorna la nuova lista di prodotti presenti nel carrello.

‘ajax/cart/remove’: metodo POST, parametri: **userID, productID**

se l’utente ha il permesso rimuovi il prodotto specificato dal carrello. Ritorna la nuova lista di prodotti nel carrello.

‘ajax/cart/buy’: metodo POST, parametri: **userID**

se l’utente è autenticato, imposta tutti i prodotti nel carrello dell’utente come venduti (sold=1) e rimuove tutti i prodotti dal carrello. Ritorna una lista (vuota) dei prodotti nel carrello.

‘ajax/reviews’: metodo GET, parametri: **reviewedID**

ritorna tutte le recensioni ricevute dall’utente corrispondente a *reviewedID.* L’output ha il seguente formato JSON:


```
{
"     state": 0,
"     message": "User reviews",
"     data":  [
            {
"                 writerID": "1",
"                 writerUsername": "ab",
"                 reviewedID": "2",
"                 rating": "0",
"                 pub_date": "2019-01-03",
"                 text": "testo della recensione",
                  "writer_image": "profile_img.jpg"
            }
      ]
```

}

'ajax/reviews/add': metodo POST, parametri: **writerID, reviewedID, rating, text**
se l'utente è autenticato, aggiunge una recensione all'utente con id: *reviewedID.* Ritorna la nuova lista di recensioni.

'ajax/reviews/remove': metodo POST, parametri: **writerID, reviewedID**
se l'utente è autenticato, rimuove la recensione corrispondente e ritorna la nuova lista di recensioni.

Tutte le richieste passano per *'index.php' e 'controller/Controller.php',* quest'ultimo si occupa di decidere quale controller chiamare. Il sito necessita del modulo *mod_rewrite* per gestire le richieste ajax.

A seconda della risorsa richiesta viene chiamato il *Controller* corrispondente. Ad esempio, se viene richiesto *'ajax/products?id=2'* viene chiamata la funzione *getAllProducts*, contenuta in *controller/ProductController.php.* Tutte le funzioni per gestire una funzionalità sono raggruppate nella stessa classe controller. Il controller, tramite sessioni, controlla se i l'utente ha il permesso di accedere a quella risorsa e poi richiede i dati necessari al model.

Il modello, una volta controllata la validità dei dati passati, comunica con il database e ritorna i dati al controller. Ad esempio, se viene richiesta la lista di prodotti, ritorna una lista di oggetti Product (definito in *'model/Products/Product.php'*) che contengono le informazioni richieste. Se il modello riscontra degli errori, lancia un'eccezione che viene poi gestita dal Controller.

Il controller crea poi un'istanza della classe Response (definita in *'contoller/Response.php')* a cui passa i dati ritornati dal model. Tutte le classi dei dati del model (*Products, Users, ecc..*) implementano l'interfaccia JsonSerializable. La classe Response si occupa di rappresentare gli oggetti passati in formato JSON e di inviare la risposta.

Front end:

Librerie utilizzate: jquery, handlebars.

Come già detto, tutto il codice html, javaScript e css viene trasferito durante la prima chiamata. Il codice javaScript è contenuto nella cartella js e il codice css nella cartella css.
Il sito utilizza gli hash nel url per decidere quale pagina mostrare all'utente.
Il codice contenuto in Router.js si occupa di ascoltare per cambiamenti nel hash e decidere che pagina visualizzare. Le possibili pagine sono: */login, /signup, /profile, /products/search.*
Una volta decisa la pagina da visualizzare viene chiamato il codice javaScript corrispondente.

Nella pagina del profilo vengono mostrate le informazioni sull'utente, i prodotti messi in vendita, le recensioni e il carrello. Ogni funziona del sito è gestita separatamente dalle altre e per ognuna c'è un file javaScript dedicato. In generale, dopo aver fatto la richiesta al server e aver ricevuto i dati, viene generato il codice html con le informazioni ricevute tramite *handlebars*, che si occupa anche di fare l'escape dei caratteri pericolosi. Infine, il codice generato viene inserito nella pagina.

A livello di interfaccia, il sito può avere un layout a tre, due o una colonna a seconda della dimensione dello schermo utilizzato. In alto è presente una barra di navigazione con alcune azioni e la funzione di ricerca. Il contenuto principale della pagina e mostrato nella colonna centrale. La colonna di destra è dedicata al carrello, è possibile aggiungere i prodotti al carrello utilizzando un bottone oppure trascinando il prodotto nell'area del carrello. Il drag and drop è implementato tramite le funzioni html5. La colonna di sinistra è utilizzata per la ricerca avanzata.