Relazione di progetto
Tecnologie Web

912404 - Marco Edoardo Santimaria

Gennaio 2022

Indice

1 Introduzione

1.1 Tema del sito

Il sito realizzato si chiama UniShare, e come suggerisce il titolo, ha come scopo permettere a utenti universitari di mettere in vendita appunti.

1.2 Sezioni principali

Il sito è composto da 4 sezioni principali:

- Homepage: questa è la pagina di arrivo sul sito. contiene un form per cercare documenti e una descrizione (riempita con testo fittizio) del team che ha realizzato il sito;
- Search: questa pagina è dedicata alla consultazione del catalogo di appunti in vendita sul sito;
- Upload: questa pagina è dedicata alla messa in vendita (e quindi all'upload) di appunti;
- Userpage: Questa è la pagina dell'utente (oltre a essere la più complessa), fornisce tutte le informazioni dell'utente, e permette la consultazione degli appunti in vendita e acquistati;

2 Funzionalità

2.1 Login / Logout

Nel momento in cui un utente arriva sul sito, può vedere la homepage senza essere loggato, ma per usare qualsiasi altra funzionalità, l'utente deve aver effettuato l'accesso. questo viene ottenuto con un controllo fatto nella funzione javascript `loginCheck()` disponibile globalmente in tutto il sito e richiamata in ogni pagina ad eccezione della homepage. se la chiamata restituisce uno stato diverso da "Logged" allora l'utente viene rediretto alla pagina di accesso. All'utente viene esposto il login come una pagina html con un form da compilare mentre il logout è un pulsante nella navbar che richiama la funzione logout().

2.2 Registrazione

La registrazione è fornita tramite form raggiungibile sia tramite pulsante nella pagina di login, sia come pulsante nella navbar (a condizione che l'utente non sia loggato). Premuto il pulsante "Registrati", oppure premuto il pulsante invio (bindato alla funzione register() tramite jquery) viene inviata una chiamata all'api di registrazione inviando un oggetto json. in caso di successo, si viene rediretti alla homepage del sito, altrimenti viene mostrato un errore. I dati vengono validati lato client, in particolare la email viene validata con una espressione regolare. Per le altre informazioni, si verifica che non siano vuote e che le due password coincidano. Lato server vengono effettuati altri controlli.

2.3 Ruoli

Sono previsti due tipi di ruoli all'interno del sito. il primo è utente normale e il secondo è amministratore. Le differenze si vedono nella pagina user, dove se l'utente è amministratore ha più funzionalità. Questo è ottenuto tramite interazione js e php. In particolare la pagina user contiene di default i template delle interfacce degli oggetti visibili agli utenti normali insieme a quelli visibili solo agli admin. Quando la pagina è stata caricata viene chiamata la funzione `checkUserType();` che, andrà a eliminare tutti gli elementi marcati con la classe `admin-only`, nel caso l'utente non sia amministratore. Uno "svantaggio" di questo approccio è che nel codice js rimane anche la parte che gestisce i componenti usabili solo dagli amministratori. In ogni caso, le api per le funzioni di amministratore sono utilizzabili solo da essi grazie a controlli lato server.

2.4 Contenuto generato dall'utente

A ogni utente (che sia amministratore o utente semplice) è concesso di caricare appunti da metter in vendita tramite la pagina upload. Inoltre, una volta che un utente ha acquistato un appunto, potrà lasciare una recensione dell'appunto, che verrà mostrata come numero di stelle, nel momento in cui viene effettuata una ricerca. Il documento caricato sarà poi disponibile alla vendita, e sarà visualizzabile nella pagina search.

3 Caratteristiche

3.1 Usabilità

L'usabilità è stata testata e comprovata, grazie a delle sperimentazioni pratiche effettuate su parenti che non hanno capacità informatiche di alto livello (utenti di competenze medio basse), proponendo loro dei task da compiere sul sito e valutando la facilità con cui riuscivano a interagire con il sito, oltre a raccogliere il loro feedback e a modificare il sito di conseguenza. Considerando che il target del sito, sono utenti con competenze medio alte (universitari), è plausibile immaginare che il sito sia stato reso di semplice utilizzo per l'utente target. Infine l'utilizzo di bootstrap, risolve ogni problema relativo al visual design.

3.2 Interazione / Animazione

Le componenti del sito sono animate con metodi jquery. è stato fatto un uso intensivo dei metodi fadeIn() e fadeOut() di jquery. In particolare nella pagina di upload del documento, è presente una animazione .hide() di jquery dove l'oggetto viene fatto scivolare a sinistra per poi mostrare la pagina di successo. Infine nella dashboard della user page, è possibile fare drag&drop dei widget da aggiungere, in modo da fornire una interfaccia intuitiva alla personalizzazione della pagina stessa.

3.3 Sessioni

Le sessioni vengono iniziate nelle prime linee della pagina api/index.php, rendendo disponibile per tutta l'esecuzione del backend la variabile di sessione `$_SESSION["username"]`. in particolare, se questa è settata, so che ho un utente loggato e che quindi posso utilizzare il valore della variabile come riferimento all'utente che ha effettuato la richiesta. Se invece la variabile non è impostata, allora vado a terminare l'esecuzione. Nell'api logout (api/res/API_LOG_OUT.php),provvedo a eliminare la variabile e a riavviare la sessione php.

3.4 Interrogazioni a db

All'utente viene reso possibile interagire con il database tramite l'utilizzo di form html. L'utente infatti è in grado di: registrarsi, accedere, ricercare appunti, acquistare appunti, caricare appunti e infine di lasciare recensioni per appunti acquistati. Esistono anche delle interrogazioni "indirette", ovvero quelle effettuate dalla dashboard quando va a riempire i widget della dashboard con delle informazioni variabili in base all'utente collegato. Lato server le interrogazioni vengono gestite con la libreria pdo di php.

3.5 Validazione input e sicurezza

La sicurezza nelle query è garantita dal metodo `bindValue()` di PDO. Questo metodo infatti garantisce che l'input che viene inserito a database sia sicuro. inoltre sempre a lato server, viene controllato che tutti i dati sono stati passati dal client e che non siano vuoti.I controlli effettuati lato client, invece, sono minimali: si verifica solamente che le email siano valide (usando una espressione regolare) e infine si verifica che tutti i campi abbiano un valore non vuoto.

3.6 Presentazione

Il sito utilizza un layout con al massimo due colonne. I colori sono tutti simili (Questo in quanto mi sono limitato a usare i colori definiti da boostrap), e nelle piccole parti dove ho scelto io combinazioni di colori ho posto attenzione a non generare combinazioni di difficile lettura. Le immagini sono poche e chiare in modo che siano funzionali allo scopo senza andare a disturbare l'utilizzo del sito. L'usabilità è stata verificata, chiedendo a parenti di testare il sito e fornire un feedback sulla facilità di svolgere i task da me proposti.

4 Front-end

4.1 Separazione presentazione/contenuto/comportamento

- **Presentazione**: Tutto lo stile del sito, è definito o nelle librerie boostrap oppure nel file `src/css/custom.css`, e viene successivamente incluso nelle pagine con gli opportuni tag html;
- **Contenuto**: Le pagine del sito internet sono completamente realizzate in html. Non vengono usati include di php, ma i Server Side Inlcudes.
- **Comportamento**: Per quanto riguarda il javascript, è contenuto tutto nella cartella `src/js/` e viene incluso a seconda della pagina con i tag corretti. Non vi è alcun codice javascript incluso all'interno della pagina html. Nel codice html non viene mai usato l'attributo dei tag `onclick`, ma viene usato il metodo `.click()` definito dalla libreria jQuery.

Lato server, il codice php (tutto contenuto nella cartella `api`) restituisce sempre un oggetto json e mai un segmento di codice html. Il codice javascript si occupa di costruire gli oggetti DOM da aggiungere alla pagina, in base al codice JSON ricevuto.

La completa separazione delle tre componenti, rende completamente indipendenti le parti tra di loro, per quanto riguarda il server su cui sono posizionati: idealmente, sarebbe possibile distribuire il front-end tramite una cdn, e salvare il database e il lato server su server completamente diversi.

4.2 Soluzioni cross plattform

Il sito è compatibile con la maggior parte di browser (non è stato testato internet explorer per ovvie ragioni), essendo basato su tecnologie che hanno incorporato al loro interno sistemi per garantire la retro compatibilità quali bootstrap e jquery. Il sito è stato inoltre sviluppato solo in versione desktop, e sebbene sia stato usato bootstrap per lo stile del sito, vi sono alcune modifiche agli stili che rendono il sito non pienamente mobile friendly.

4.3 Organizzazione file e cartelle di progetto

il progetto è organizzato con la seguente gerarchia di cartelle:

- **Root folder**: contiene le pagine html in cui l'utente potrà navigare
  - **api**: contiene il codice php del backend
    - **res**: contiene il codice delle singole api
  - **uploads**: contiene i documenti pdf caricati dagli utenti
  - **src** : contiene i file di supporto al Front end
    - **css**: contiene il foglio di stile personalizzato che va a modificare alcuni comportamenti della libreria bootstrap
    - **js**: contiene il codice javascript delle pagine di front end
    - **html**: contine le sezioni di codice html riusato tra le varie pagine. Queste sezioni di html vencono incluse con dei server side includes
    - **media**: contiene tutti i file multimediali (quali immagini) usati dal sito

4.4 Soluzioni html/css/javascript degne di nota

- Nell'intero sito esistono pochissimi form. La maggior parte delle interazioni viene effettuata con chiamate ajax implementate con la libreria jquery. Questa chiamata viene mascherata dalla funzione

  ```
  ajaxCall(api:String, payload:JSON, onSuccess:function(data), onError:function(data);
  ```

  Questo è necessario in quanto, per mantenere uniformità nella comunicazione front end e back end, in entrambe le direzioni vengono inviati degli oggetti JSON.

- La pagina utente, è realizzata in più sezioni, ognuna della quale non richiede un refresh della pagina, ma viene mostrata tramite manipolazione del DOM tramite js. Vi è infine una dashboard per ogni utente, composta di 4 widget, i quali possono essere cambiati da ogni utente.

5 Back-end

5.1 Architettura generale delle classi e delle funzioni php

Il backend ha come punto di partenza il file index.php. questo è l'endpoint di tutte le api. Una volta ricevuta una richiesta, essa viene trattata, andando a verificare che l'api sia presente nell'array dichiarato nel file res/apiList.php. Una volta verificato ciò viene incluso il file corrispondente e chiamata la funzione che andrà a servire la richiesta. Questo è fatto per rendere estremamente facile l'estensione delle funzionalità del backend, in quanto basta creare un nuovo file e inserirlo nella cartella api/res. Inoltre rende l'esecuzione del codice migliore in quanto non tutto il codice deve essere caricato e parsificato. Ogni api è composta di un metodo che richiede due parametri:

- **$conn**: e l'oggetto pdo della connessione
- **$payload**: è un array associativo contenente il json decodificato della richiesta del client

5.2 Back-end e comunicazione front/back-end e descrizione delle funzioni remote

La comunicazione front end viene effettuata con oggetti json in entrambi i sensi. le richieste vengono effettuate come richiesta **POST** all'url:

```
https://<endpoint>/api/index.php
```

Il motivo dell'utilizzo del solo metodo post è dato dal ragionamento che, poiché devo inviare un oggetto json, non ha senso aggiungerlo come parametro dell'url. Inoltre così facendo, la richiesta non viene scritta nei log file di apache, garantendo quindi una maggiore privacy. In ogni caso, il file index.php è stato strutturato per poter integrare altri metodi HTTP in maniera estremamente veloce e senza dover modificare il codice già scritto.
Le richieste hanno il seguente formato (NB: alcune api hanno payload vuoto in quanto non hanno bisogno di comunicare informazioni al server in eccesso a quelle che già ha):

```
{
    "api": <requested_api>,
    "payload":{
        <key> : <value>,
        .
        .
        <key_n> : <value_n>
    }
}
```

Le risposte invece hanno formato differente ma sono comunque un oggetto JSON in caso di richiesta di informazioni. Se invece le API devono effettuare solo qualche calcolo, senza restituire informazioni, l'output del server può assumere due forme: `{"Ok":"Done"}` in caso di successo oppure `{"Error":<error_message>}` dove error message viene impostato all'errore sql in caso che la variabile globale php `__DEBUG__`, definita nel file api/config.php, sia TRUE oppure a un messaggio generico se FALSE. L'interfaccia delle singole API è disponibile per ogni api, presente nella cartella `/api/res` come descrizione della funzione.

5.3 Schema del database

Lo schema del database è il seguente:

- appunti(idappunti, Nome, Path, uploadDate, price, insegnamento_scuola, user, tipoAppunti, nomeDocente, visible)
- users(email, Name, Surname, password, UserType)
- dashboard(user, obj1, obj2, obj3, obj4)
- recensione(idRecensione, valore, appunto, acquisto)
- acquisto(ID_acquisto, user, appunto)
- insegnamento(idinsegnamento, Nome, Scuola)
- scuola(idScuola, nomeScuola)

Con i seguenti vincoli relazionali:
appunti.user referenzia user.email
appunti.insegnamento_scuola referenzia insegnamento.idInsegnamento
insegnamento.scuola referenzia scuola.idScuola
dashboard.user referenzia user.email
acquisto.user referenzia user.email
acquisto.appunto referenzia appunti.idappunti
recensione.appunto referenzia appunti.idappunti
recensione.acquisto referenzia acquisto.ID_acquisto