Relazione progetto
Corso di Tecnologie Web

Daniele Pautasso
MATR 795817

URL: https://gitlab2.educ.di.unito.it/st148488/tweb.git

Introduzione

Nome del sito: Playlog (o play.log)

Tema del sito:
Il fine principale del sito è quello di fornire un servizio di "libreria virtuale" videoludica che permetta agli utenti (presumibilmente giocatori) di mantenere un registro dei titoli posseduti/giocati, facile non solo da aggiornare ma anche da condividere.
Un qualsiasi visitatore può cercare un gioco in base al titolo e visualizzarne le informazioni essenziali. Inoltre un utente registrato ha la possibilità di aggiungere tali giochi ad una delle liste che il sito mette a sua disposizione.
Tali liste sono: *Owned*, i titoli che l'utente possiede ma che non ha ancora iniziato; *Playing*, quelli a cui sta attualmente giocando; *Finished*, quelli che ha completato; *Dropped*, quelli che per qualche ragione ha iniziato ma abbandonato (temporaneamente o definitivamente); *Wishlist*, quelli che vorrebbe acquistare.
Infine, è presente un sistema di "followers-followed" che consente agli utenti registrati di "seguire" i profili di altri utenti, così da risalire velocemente ai titoli di loro interesse.

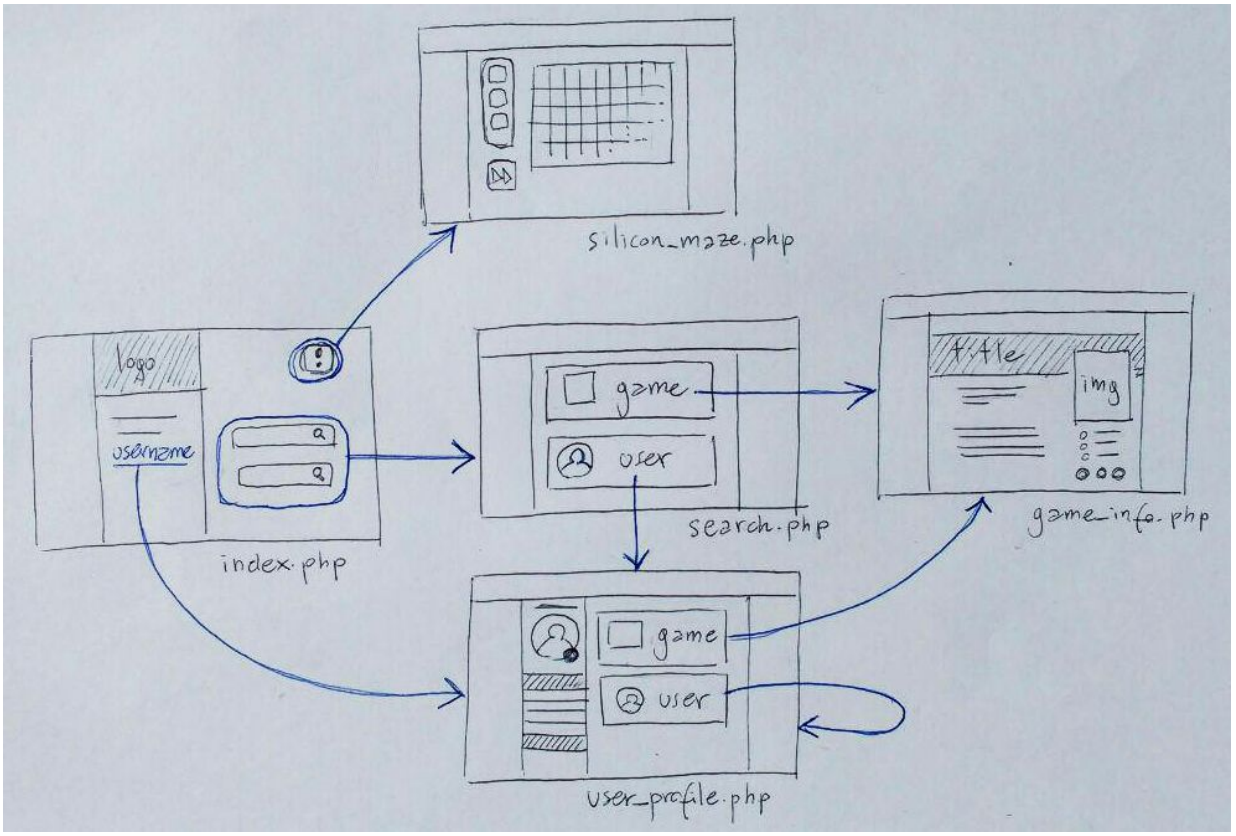
Sezioni principali:

- Home page: pagina principale, nella quale l'utente può accedere o registrarsi al sito (corrisponde ad index.php)
- Search: pagina in cui vengono mostrati i risultati delle ricerche relative ai giochi o agli utenti (corrisponde a search.php)
- Game Info: pagina in cui l'utente visualizza le informazioni relative ad uno specifico gioco e, se autenticato, può aggiungerlo ad una delle proprie liste (corrisponde a game_info.php)
- User Profile: il profilo di uno specifico giocatore: ogni utente autenticato può gestire sia i propri giochi che gli utenti che sta seguendo. Visitando il profilo di un altro utente, è possibile aggiungerlo all'elenco dei giocatori seguiti. (corrisponde a user_profile.php)
- Game - Silicon Maze: pagina in cui l'utente può giocare ad un semplice gioco realizzato in JavaScript, tentando di migliorare il proprio punteggio (corrisponde a silicon_maze.php)

Schema di navigazione e mock-up:
Si è cercato di strutturare il sito in modo tale che ciascuna sezione sia facilmente raggiungibile dalle altre. Elemento comune a tutte le pagine (escluso index.php) è l'header:

Come intuibile, Home riporta sempre alla home page. Link1 e Link2 riportano ancora una volta alla home, ma in più mostrano rispettivamente i form di login e signup nel caso l'utente

non sia autenticato; in caso contrario, Link1 porta al profilo dell'utente e Link2 diviene la funzione di logout, che reindirizza alla home.
Sono presenti inoltre due search bar, destinate alla ricerca dei giochi e degli utenti, che indirizzano entrambe alla pagina search.php.
Di seguito si illustrano brevemente i collegamenti tra le pagine, omettendo quelli relativi all'header appena descritto.



Dettagli tecnici - Front End

Soluzioni cross-platform:
Il sito è stato testato con i browser Mozilla Firefox e Google Chrome. I file CSS associati ad ogni pagina cercano di garantire una minima compatibilità con le risoluzioni desktop più diffuse, pur non essendo pensati espressamente anche per i dispositivi mobili. Si è fatto un grande ricorso alle misure espresse in percentuale, limitando il più possibile le misure in pixel.

Organizzazione file e cartelle:
L'organizzazione di file e cartelle rispecchia il tentativo di adottare quanto più possibile uno stile unobtrusive. I file php che si trovano nella root directory 'progetto' generano le pagine HTML che compongono il sito. Le directory sono invece organizzate secondo la logica seguente:

- **js**: directory contenente i vari script JavaScript associati alle omonime pagine php
- **img**: directory contenente tutte le immagini utilizzate dalle varie sezioni del sito.

- **css**: directory contenente i vari fogli di stile CSS associati alle omonime pagine php.
- **lib**: raggruppa, suddivisi in base alla finalità, i file che definiscono le funzioni php utilizzate da una o più pagine.

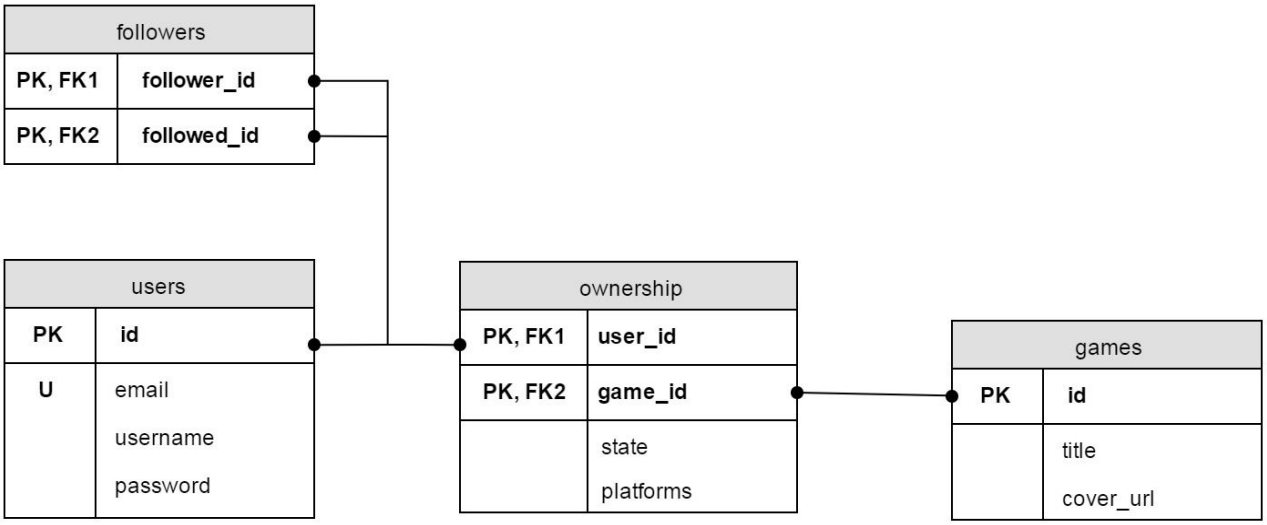
Dettagli tecnici - Back End

Architettura generale funzioni php:
Come precedentemente accennato, tutte le funzioni php sono contenute nella directory 'lib' e suddivise in base alla loro logica funzionale. Ad esempio, tutte le funzioni che hanno a che fare con l'autenticazione si trovano in 'autentication'; le funzioni che si occupano della visualizzazione dei profili utente sono definite in 'users/profile.php' mentre quelle che aggiornano lo stato delle varie liste in 'users/ownership.php', e così via.
Un ruolo particolare viene svolto dai file presenti nella cartella 'request_handlers', che contengono le porzioni di codice php destinate a gestire le chiamate Ajax effettuate da diverse pagine del sito, alle quali rispondono restituendo uno stream JSON.

Database "remoto":
Vista la mole di informazioni relative ai giochi necessarie al funzionamento del sito ed essendo inverosimile mantenere un database realizzato ex novo, si è optato per fare ricorso ad un database esterno già esistente, gestito da IGDB.com. Quest'ultimo fornisce una API grazie alla quale è possibile interrogare il database in questione tramite opportune richieste HTTP, ottenendo risposte in formato JSON. Tali informazioni vengono poi analizzate e processate da apposite funzioni php ed infine utilizzate all'interno del sito stesso (direttamente o dopo averle inviate come risposta ad una richiesta Ajax). L'esempio più complesso in questo senso è fornito dalla funzione get_game_info() - definita in 'games/get_info.php' - grazie alla quale vengono recuperate, tramite più interrogazioni del database di IGDB.com, le informazioni dettagliate relative ad un particolare titolo.
Maggiori dettagli sulla struttura e sulle funzionalità offerte possono essere trovati al link: *http://igdb.github.io/api/about/welcome/*

Database "locale":
Viene inoltre mantenuto un database ad uso esclusivo del sito per la memorizzazione dei dati relativi agli utenti registrati (tabella 'users'), allo stato delle diverse liste di ciascun utente (tabella 'ownership') ed alle relazioni di follower/followed (tabella 'followers'). Nonostante le informazioni relative ai giochi non siano contenute interamente nel suddetto database locale, esiste comunque una tabella 'games'. Tale scelta è dovuta all'elevato costo delle interrogazioni al database remoto: informazioni quali titolo e URL della cover del gioco sono necessarie anche durante la visualizzazione e la gestione dei profili utente, ed effettuare una interrogazione remota per ogni azione di questo tipo sarebbe estremamente inefficiente. Di conseguenza, ogni qualvolta un utente aggiunge un titolo ad una delle proprie liste viene effettuata una query al database locale per verificare se tale gioco è già presente. Se non lo è, una nuova entry viene creata: ciò rende il successivo recupero delle informazioni sopra citate decisamente più rapido. In caso contrario, vengono semplicemente aggiornati i campi titolo e URL della cover (per adeguarsi ad eventuali modifiche). La tabella games funziona quindi come una sorta di "chace" che rende la gestione dei profili utente molto più semplice e veloce.

Schema del database locale:



Per scongiurare il pericolo di SQL Injection, ogni query viene eseguita per mezzo delle funzioni prepare(), bind() e execute() messe a disposizione da PDO. Inoltre, le password degli utenti non sono conservate in chiaro per motivi di sicurezza.

Descrizione delle funzioni remote:
Tutte le richieste Ajax effettuate dalle pagine sono indirizzate ad uno dei file nella directory 'request_handlers'. Ognuno di questi file solitamente gestisce un sottoinsieme ben definito delle operazioni remote possibili per evitare di importare grandi porzioni di codice php contenenti funzioni non utilizzate in un determinato contesto. Inoltre, tale approccio consente la modifica di alcune classi di funzioni senza che un eventuale errore infici il corretto funzionamento delle altre sezioni del sito.
Le richieste fanno tutte uso del metodo POST ed hanno in comune un parametro 'function' che permette all'handler di scegliere la funzione da richiamare. La maggior parte delle risposte includono, oltre agli eventuali dati, anche un campo 'success': a seconda dell'azione intrapresa esso assume un valore che può essere utilizzato lato client per verificare l'esito dell'operazione stessa.
A titolo di esempio, viene illustrata brevemente la sequenza di azioni effettuate per richiedere in modo asincrono una lista di giochi.

Richiesta Ajax in 'search.js':

```
new Ajax.Request("lib/request_handlers/search_handler.php",
        {
                method: "post",
                parameters:  {   function: "search_games_by_title" , //funzione da richiamare
                                 gametitle: gametitle },   //titolo gioco da cercare
                onSuccess: show_games_list,   // riceve JSON, mostra lista aggiornando DOM
                onFailure: handle_failure
        }
);
```

Gestione nel file ‘lib/request_handlers/search_handler.php’:

```
switch($_POST[“function”]){
	case "search_games_by_title":
		$res = search_games_by_title($_POST["gametitle"]); //effettua richiesta a DB remoto
		if(isset($res)){
			if(!empty($res))
				$res[] = array('success' => 1);  //richiesta ok, lista non vuota di giochi
			else
				$res[] = array('success' => 0);  //richiesta ok, ma lista vuota
		}
		else $res = array( '0' => array('success' => -1) );  //errore richiesta a DB remoto
		echo json_encode($res);  //converto l’array ed invio i risultati come JSON
		break;
	//gestione altri casi dello switch
}
```

Sintassi output JSON in caso di successo e lista di giochi non vuota :

```
[{id: 1070, name: "Super Mario World", rating: 92.27 cover: {...} },
 {id: 1330, name: "Super Mario Bros", rating: 88.34, cover: {...} },
 ... ,
 {success: 1}]
```

Tale soluzione è di particolare rilevanza nelle fasi di registrazione/login, in quanto il valore restituito nel campo ‘success’ viene utilizzato per fornire un feedback significativo all’utente. E’ infatti possibile distinguere, ad esempio, i casi in cui la registrazione non va a buon fine in quanto già presente un account associato alla email specificata da quelli in cui invece si verifica un qualche errore lato server, quindi indipendente dall’utente.
In generale, le condizioni di errore sono gestite usando un approccio simile a quello appena descritto.