Progetto per AmMinistrazione di Sistema

# Descrizione

Il sito web si configura come una *galleria espositiva online*, che permette ai suoi *proprietari / autori*, di *publicizzare e vendere i loro lavori*. L’idea è che questo sito web rappresenti *l’attività di un gruppo di fotografi professionisti*, che vogliono esporre e vendere i loro lavori, senza alcuna intermediazione, tramite un loro sito web dedicato. Il sito fa quindi da interfaccia fra gli utenti finali, che andranno ad acquistare i lavori, gli autori delle foto, e le figure gestionali annesse per la gestione logistica e strutturale.

# Utenti del sito

**L’utente finale** è quella figura a cui è dedicato lo spazio per ***l’acquisto***. L’utente , potrà scegliere se acquistare ***una determinata immagine*,** in *formato digitale* (acquistando *i diritti d’uso)*, oppure se ordinare i vari articoli *in formato galleria*, per un’esposizione pubblica o privata. Avrà la possibilità di ***registrare un******account***, per acquisti futuri o immediati, che riassumerà tutte le informazioni che ha concesso al sito in termini di dati anagrafici e di recapito, nonchè l’elenco di attività collegate al sito. L’account utente è visionabile in una ***sezione del sito dedicata***, comprendente 3 pagine riassuntive (*Profilo, Pagamenti, Ordini).* ***L’acquisto*** non è mediato da un ***carrello*** in quanto, per prezzi così importanti, si è scelto un metodo di acquisto più riflessivo, limitato ad un singolo articolo per volta.

Per la ***gestione del sito*** sono previste due figure: il *proprietario* e il *web Admin*.

Il **proprietario** non si occupa delle funzioni gestionali del sito, ha solo *controllo del database* e *dello spazio nel server web*. La separazione, posta in questi termini, è stata pensata per *limitare le possibilità d’azione di un eventuale amministratore esterno* alle funzionalità predefinite che gli sono state concesse. **Il web Admin**, invece, è una figura che si può occupare sia del ***lato logistico***, che del ***lato gestionale****.* Ha a disposizione *4 sezioni di gestione*, dalla quale può visionare e interagire coi dati relativi a: *utenti, ordini effettuati, metodi di pagamento registrati, articoli visibili (e non) all’interno della galleria principale*.

Il **web Admin** ottiene quindi *uno spaccato completo delle attività del sito*, avendo poi la possibilità di **modificare** o **eliminare** dati incongrui. *Non è stato previsto, da parte del web Admin,* ***l’inserimento*** *di nuove voci* all’interno del database, in maniera tale che possa limitarsi a gestire e modificare unicamente le voci già presenti all’interno del database *(gestite dal proprietario).* L’eventuale modifica di qualsiasi voce presente all’interno del database è tracciata in automatico dalle più comuni implementazioni di SQL (per cui non sono presenti voci a riguardo se non laddove hanno una funzione informativa anche per il web Admin).

**Gli autori** delle foto, che *coincidono col proprietario*, dovranno procedere ***manualmente*** all’inserimento delle immagini, prima all’interno dello spazio web del server, poi all’interno del database. In una successiva implementazione si può pensare ad un’interfaccia per velocizzare l’operazione.

# Requisiti del progetto

1. **Utilizzo di HTML e CSS**

Il sito web è strutturato in ***HTML5***, facendo largo uso dei nuovi tag semantici nella strutturazione di ogni pagina interna al sito. Il codice HTML5 è stato validato, secondo gli standard del w3c, per tutte le sezioni del sito.

Le viste sono strutturate con l’ausilio del ***CSS3***. Ho usato animazioni in alcuni punti del sito, mentre ho avuto diversi problemi nell’implementare un design che fosse scalabile su più risoluzioni differenti e su schermi non widescreen. Il risultato è un codice css non validato per via dei tag proprietari implementati dai broswer, non aderenti allo standard.

1. **Utilizzo di PHP e MySQL**

Tutto il sito poggia su una base di ***PHP*** che gestisce *sessioni*, *visualizzazione*, *permessi*, *ruoli* e *funzionalità* degli utenti. E’ strutturato secondo il pattern MVC e fa uso, in alcuni contesti, di funzioni d’ausilio (chiamate helper functions) per portare a termine compiti specifici, anche oltre le funzionalità base del pattern (es SendMail.PHP)

Il ***database (SQL)*** è stato progettato esplicitamente per il sito, e presenta 6 tabelle (WebAdmin, LoggedUser, Ordine, Immagine, CreditCard, Bonifico) che riassumono tutte le entità (e le relative associazioni) su cui si struttura il sito. *Nella cartella Database è presente sia lo schema concettuale del database sia l’esport dell’implementazione dello schema logico.*

1. **Utilizzo del pattern MVC**

Il ***pattern MVC*** struttura completamente il sito web, garantendo un’interazione completa fra le entità che lo compongono, le logiche di gestione e la visualizzazione finale di tutti i dati annessi. Nello specifico, la pagina ***index.php*** fa da *punto unico d’accesso*, istanziando, in relazione alla risorsa richiesta, lo specifico controller dedicato alla gestione. Il risultato dell’interazione fra controller e model è direttamente visionabile all’interno della pagina ***master.php***, che si occupa d’includere *tutte le viste* necessarie alla composizione finale della vista selezionata.

Il ***model,*** include sia la versione php delle entità già presenti nel database che alcune funzioni ed entità relative alle funzionalità specifiche del sito. Include:

* **User** – che rappresenta la superclasse utente(generico) prevista dallo schema concettuale del DB
* **Authenticated user** – che rappresenta l’utente loggato
* **Web Admin** – che rappresenta un amministratore del sito
* **Pagamenti** – realizzata in supporto al meccanismo di gestione delle transazioni, serve per memorizzare le preferenze dell’utente, durante lo svolgimento della transazione. (il risultato è utile come riepilogo sia al server che all’utente)
* **Formati** – che rappresenta i formati disponibili relativi all’immagine desiderata
* **RaccoltaImmagini** – che permette il raggrupamento di tutte le immagini presenti nel sito secondo Album tematici. (Funzionalità non prevista nella progettazione del DB)
* **Immagine**
* **Ordine**
* **CreditCard**
* **Bonifico –** che rappresentano le entità presenti nel DB, istanziabili in PHP
* **DB –** che memorizza tutte le coordinate necessarie alla connessione col DB

A queste entità si associano delle funzioni d’ausilio in PHP, necessarie per l’interazione Controller-Model; sono le funzioni:

* **UserConstructor** – che permette d’istanziare, registrare, cancellare utenti
* **PicturesCollectionConstructor** – che permette di creare delle raccolte di foto con nome
* **OrderIDGenerator** – che elabora un’id unico per ogni transazione
* **AdminHelper** – che fornisce all’admin tutti i servizi necessari allo svolgimento delle sue funzioni

***Il controller*** è in realtà strutturato secondo più controller differenti, suddivisi in base al contesto gestionale di cui si devono occupare. I controller, interagiscono fra loro attraverso la modifica di oggetti del model e, nel caso del login e della gestione delle transazioni, attraverso oggetti PHP superglobals, necessari per memorizzare lo stato dell’ utente fra una richiesta e l’altra. Sono inclusi:

* **LoginController** – che si occupa della gestione della sessione utente, autenticando e convalidando le sessioni, oppure terminandole.
* **UserController** – che gestisce viste e modelli legati sia al profilo dell’utente standard che al menù di gestione dell’amministratore
* **AboutController** – che gestisce viste e modelli legati alla sezione WHO WE ARE
* **WorksController –** che gestisce sia la galleria delle immagini presenti sul sito che le transazioni d’acquisto (in ausilio alle funzioni di gestione aggiuntive)
* **OtherSectionsController –** che gestisce viste e modelli legati a tutte le altre pagine del sito, fra cui registrazione, index, contatti, ecc

1. **Almeno due ruoli (indicare quali sono)**

* [Utenti del sito](#_Utenti_del_sito)

1. **Almeno una transazione (indicare la classe dove è implementata)**

***La transazione*** è implementata attraverso una classe dedicata, TransactionHandler, tramite la funzione starNewTransaction (***TransactionHandler ::starNewTransaction($datiOrdine)***). La funzione si occupa prima di generare un id casuale per l’ordine, in modo che non si sovrapponga ad altre ordinazioni. Subito dopo effettua la transazione con l’eventuale banca online e (controllo non implementato) e istanzia un prepared statement, secondo il pattern della query da effettuare. Dopo aver eseguito lo statement, decrmenta la disponibilità dell’oggetto dal database, e riabilita l’autocommit. In caso di errore la transazione è annullata e genera un codice d’errore che riporta alla funzione in cui l’errore si è verificato.

1. **Almeno una funzionalità ajax (indicare in quale script si trovi)**

All’interno dello script .js presente in ***Javascript/AjaxLoginMgmt.js*** si trova uno script ***JQuery*** utilizzato per la gestione dei form di login tramite *ajax:*

i dati del form sono inviati tramite AJAX al punto unico d’accesso, index.php, con la richiesta di gestione da parte del LoginController (che deve autenticare l’utente). Il server comunica l’esito della procedura un oggetto XMLHttpRequest con codice di stato 202 in caso di successo, o 401 in caso di mancata autenticazione. Ricevuto l’esito, lo script del client non ricarica la pagina ma elabora un menù apposito per gli utenti loggati (altrimenti restituisce un alert con un messaggio d’errore).