Universitá Politecnica delle Marche

Corso di Tecnologie Web

Docente: Prof. Alessandro Cucchiarelli

Raffaello. Il social network.

Studente:
Nicola Fioranelli
(Gruppo 68)

Matricola:
1061904

Gennaio 2017

RAFFAELLO

Indice

1 Progetto

La diffusione sempre maggiore dei dispositivi che permettono l'accesso al web ha modificato le modalitá di fruizione dei contenuti remoti da un'ottica one-way ad una visione interattiva.

In questo senso il progetto affidatoci per il corso di Tecnologie Web vuole essere un social network per la gestione di una comunitá on-line. Per realizzarlo ho impiegato diversi strumenti tecnologici che analizzeró nel dettaglio successivamente.

1.1 Progettazione del Database

Ogni applicazione web ha bisogno di poter memorizzare informazioni per riutilizzarle durante le elaborazioni delle richieste del Client. Per far ció é necessario disporre di un database come, per esempio, MySQL.

Dall'analisi delle specifiche di progetto ho ottenuto il seguente schema logico (Fig. 1) con l'unica soluzione della view 'elencoutenti' che semplifica l'utilizzo del DB con Zend.

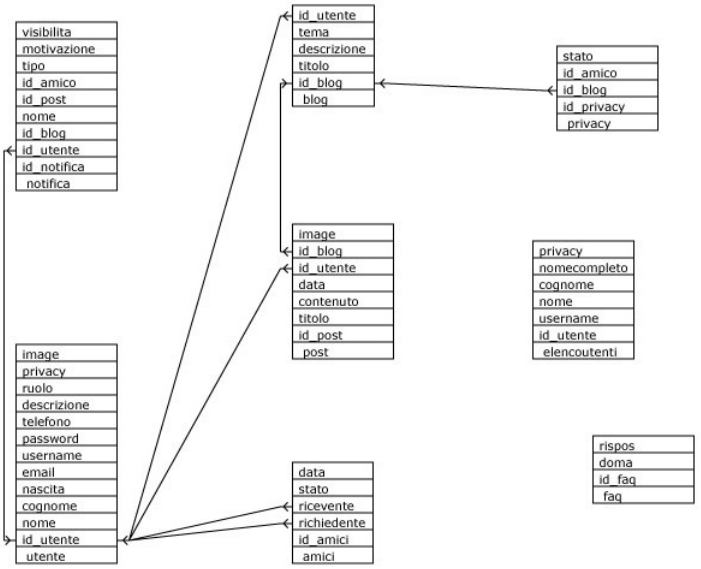


Figura 1: Diagramma logico database

1.2 Zend e il design pattern MVC

Il problema della scalabilitá di una web app é molto rilevante in quanto le modalitá di visualizzazione delle informazioni, la grafica o il database possono cambiare nel

corso del tempo; ecco perché si rende necessario poter disporre di una architettura modulare che consenta di intervenire solo nei campi di cui necessitiamo.

Un design pattern risulta essere una soluzione progettuale ad un problema ricorrente e nel modo specifico il modello MVC si pone come esempio di architettura utilizzabile; in esso infatti sono tre le strutture in cui si divide la web app.

- **Model:** é responsabile della gestione del database, recupera i dati su richiesta del Controller e li fornisce alla View;
- **View:** implementa l'interfaccia grafica della web app;
- **Controller:** é responsabile della gestione della comunicazione con il Model e delle interazioni dell'utente tramite la View.

Un framework é invece uno strumento che semplifica il lavoro dei programmatori durante lo sviluppo dell'applicazione, fornendo per esempio librerie per le operazioni piú comuni da realizzare.

Una soluzione che racchiude il pattern MVC e lo sviluppo sotto PHP é Zend Framework, presentato durante il corso e utilizzato in questo progetto nella sua **versione 1.12.13**.

1.3 Articolazione del progetto

Seguendo le specifiche di progetto ho realizzato la web app "Raffaello" cosí strutturata. (Fig. 2)

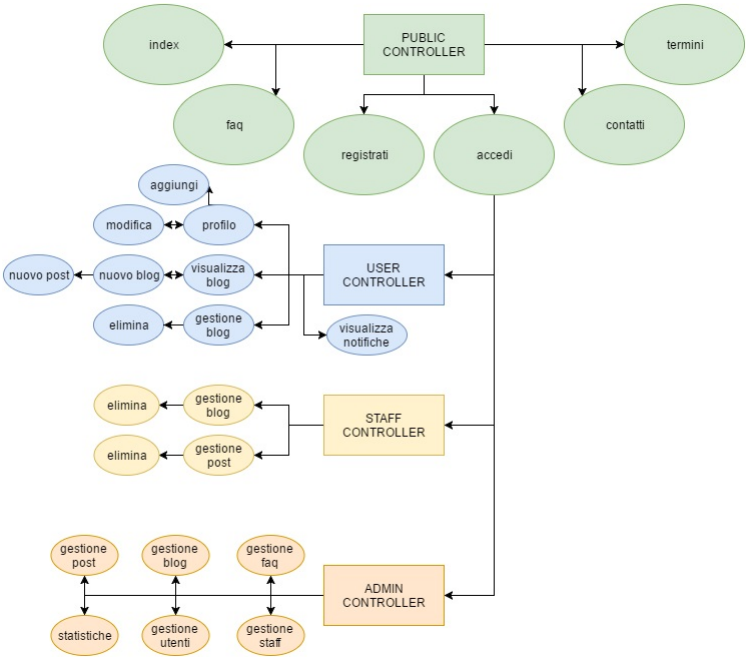


Figura 2: Schema dei link.

Chiavi esterne e Model In piú di un'occasione durante lo sviluppo del progetto é stato necessario avere dati presenti su tabelle diverse nel database collegate tra loro attraverso chiavi esterne (*foreign key*).

Il problema sorge peró quando il Model richiede i dati di una tabella che contiene delle chiavi esterne alle quali si vuole accedere; l'approccio standard a tale tipo di problema é l'utilizzo del `join` nelle operazioni di `select`. In questo caso, si é voluto invece evitare il `join`, abbastanza complesso da gestire in Zend, con una soluzione che "alleggerisce" il lavoro del Model e lascia al Controller il compito di ricostruire i dati da passare alla View.

Si prenda come esempio il problema di visualizzare sulla home dell'utente i dati relativi alle richieste di amicizia; per semplicitá di gestione la corrispondente tabella nel database ha solo l'id dell'utente che richiede o a cui arriva l'amicizia come chiave esterna ma non il nome e cognome.

Seguendo l'idea sopra descritta il Model corrispondente ha il metodo `elencoAmiciById($id)` che fa una semplice `select` nel database:

```
public function elencoAmiciById($id)
{
return $this->tabella->fetchAll($this->tabella->select()->
    where("ricevente =  ? ", $id)->where("stato = 'standby'")
    );
}
```

sará invece il Controller a costruire il vettore dei dati da passare alla View, recuperando anche le informazioni dalla tabella Utente mediante l'id trovato con il precedente metodo:

```
$amiciModel = new Application_Model_Amici();
$utentiModel = new Application_Model_Utente();
$idUtente = $this->utenteCorrente->current()->id_utente;
$rowset = $amiciModel->elencoAmiciById($idUtente);
$amicidata = array();
$i = 0;
foreach ($rowset as $data) {
  $amicidata[$i]['id_utente'] = $data->richiedente;
  $temp = $utentiModel->elencoUtenteById($data->richiedente
     );
  $amicidata[$i]['richiedente'] = ucwords($temp->current()
     ->nome . " " . $temp->current()->cognome);
  $amicidata[$i]['id_amici'] = $data->id_amici;
  $i++;
}
$this->view->assign("amiciSet", $amicidata);
```

Customizzazione delle Zend Form Come giá accennato in precedenza Zend Framework, offre delle classi predefinite da poter utilizzare nel proprio codice, so-

prattutto per le funzioni maggiormente utilizzate. Un'esempio é la classe Zend Form che implementa la form di immissione dati, inserita nel progetto per svariati usi.

Tuttavia di default tali form restituiscono, nei casi in cui i validatori dei diversi elementi non accettino i dati inseriti per svariati motivi, un messaggio di errore predefinito in inglese. É sembrato ovvio, dunque, customizzare gli errori delle form sia per quanto riguarda il layout sia per la lingua.

Quanto al primo aspetto esso é stato risolto con l'inserimento di un foglio di stile (`error.css`) nella cartella `public` contenente le direttive grafiche da utilizzare. Diversa, invece, la soluzione adottata per la lingua; infatti si é provveduto a includere il file `Lingua.php` all'interno di una form per la traduzione degli errori in italiano(solo in `Registrati.php` in quanto il comando `include_once` non necessita l'inserimento in ogni form).

Paginatore Per visualizzare contenuti molto estesi ma al tempo stesso offrire una buona fruibilitá all'utente si utilizza il paginatore. Tale funzione é disponibile nelle classi di Zend Framework come `Zend_Paginator` e ne é stato fatto uso per la visualizzazione dei post all'interno di un blog.

Il paginatore puó accettare tipi di dato diversi ma in questo caso ho utilizzato come struttura dati l'array.

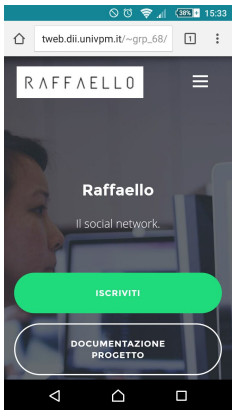
1.4 Tecnologie utilizzate

Sweet Alert e interazione Ajax Nel normale funzionamento client-server il workflow tradizionale é rappresentato da una chiamata a livello client a cui segue l'invio delle risorse necessarie elaborata dal server; questo tipo di comunicazione viene detto sincrono. Tuttavia esistono delle tecnologie che permettono di creare una comunicazione asicrona: é il caso delle chiamate Ajax implementate mediante jQuery.

L'utilizzo di tali tecnologie é stato implementato negli sweet alert[1] inseriti nelle fasi di cancellazione blog e post nei tre livelli di utenza.

jQuery Validation Ancora una volta seguendo la stessa idea descritta sopra é stata implementata una validazione asincrona.
Infatti mediante la libreria JQuery Validation, si é potuto controllare la bontá dei dati inseriti nelle form ancora prima dell'invio al server. É quindi l'interprete Javascript a occuparsi della validazione in prima battuta, indicando a "run time" i campi contenenti errori, a cui segue, dopo la 'submit' la validazione lato server.

[1]Documentazione del progetto Sweet Alert disponibile all'indirizzo `http://t4t5.github.io/sweetalert/`



Bootstrap Un ultimo strumento, altrettanto importante in quanto permette di avere una user experience piú gradevole, é stato adottare un template che utilizzasse il framework Bootstrap, una raccolta di modelli di progettazione basati su HTML e CSS e con alcune estensioni opzionali di Javascript (come lo slider nella home-page dell'area pubblica della web app).

Bootstrap é anche responsive ció significa che il layout delle pagine web si regola dinamicamente, tenendo conto delle caratteristiche del dispositivo utilizzato, sia esso desktop, tablet o telefono cellulare.

2 Guida all'uso

Raffaello (Fig. 3) é un social network che permette a tutti i suoi iscritti di comunicare mediante la creazione di post all'interno di blog tematici e la creazione di gruppi di amici. Le informazioni personali degli utenti sono al sicuro grazie a livelli di privacy personalizzabili sia per la visibilitá del proprio profilo personale sia per i blog pubblicati.

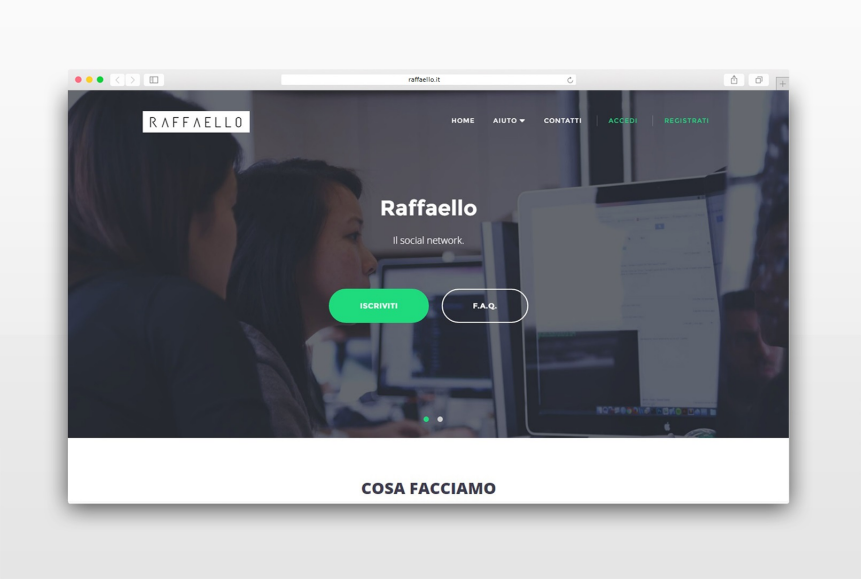


Figura 3: Home page del sito

2.1 Livello di utenza pubblico

Dalla homepage ciascun fruitore puó trovare informazioni circa le condizioni per l'utilizzo della piattaforma, le FAQ e le due form dedicate alla registrazione dell'utente e l'altra all'accesso alle proprie aree riservate.

2.2 L'utente

L'utente standard accede alle funzioni di pubblicazione una volta effettuata la procedura di login. Per coerenza con l'idea di social network dopo il primo accesso (non essendo presenti blog) il sistema chiede di pubblicare un primo blog e di conseguenza un primo post al suo interno. La pagina principale permette di tenere sotto controllo le richieste di amicizia e le notifiche(ed eventualmente nasconderle).

La navbar (Fig. 4) consente di avere a disposizione in qualsiasi momento tutte le funzionalitá fra le quali:

- la gestione degli amici (di conseguenza la possibilitá di eliminazione di una persona dal proprio gruppo);
- la gestione dei blog (dove é possibile eliminare un blog o accedere per ogni blog alla visibilitá dello stesso per ciascun amico: l'icona rossa a forma di croce indica che per quell'utente il blog non sará visibile, quella verde il contrario);
- la modifica profilo (attraverso la quale si potranno cambiare i propri dati personali nella form dedicata; lasciando vuoto il campo password essa non verrá modificata e lasciando vuoto il campo immagine del profilo verrá lasciata l'immagine di default);
- da ultimo le impostazioni della privacy del profilo.

Cliccando infine sul proprio nome sulla navbar si accede alla pagina del profilo con la propria anagrafica.

A disposizione per tutti gli utenti la funzione di logout.

Figura 4: Navbar personalizzata

2.3 Lo staff

Lo staff ha la possibilitá di consultare tutti i blog di tutti gli utenti eliminandoli dopo l'inserimento di un'adeguata motivazione cosí come per tutti i post.

2.4 L'amministratore

Da ultimo l'amministratore che oltre a svolgere le operazioni disponibili per lo staff puó gestire tutti gli utenti e ha a disposizione strumenti statistici per ogni utenza.

Nella sua area riservata é parso interessante dotare l'amministratore della possibilitá di gestire dinamicamente le FAQ: é quindi possibile creare nuove domande e risposte da fornire agli utenti nell'area pubblica.

La gestione degli utenti infine suddivisa per categoria, utente standard e staff, per quest'ultima esiste anche la possibilitá di inserire una nuova persona. Sulla gestione degli utenti inoltre é possibile accedere al profilo di ciascun iscritto dove consultare anche i dati statistici sulle richieste di amicizia e gli amici che ha.

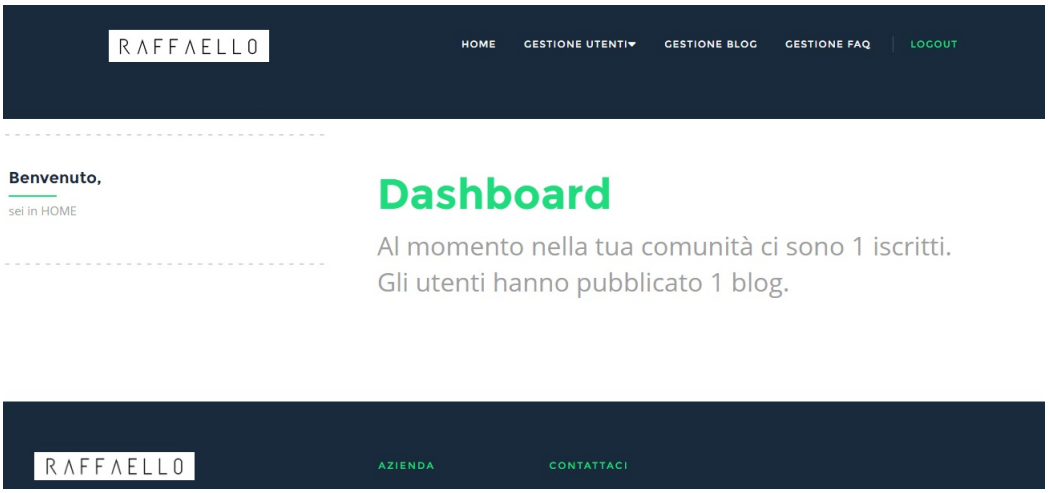


Figura 5: Dashboard amministratore