POLITECNICO DI BARI

Dipartimento di Ingegneria Elettrica e dell'Informazione (DEI)

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Tema d'anno
in
Linguaggi e Tecnologie Web

FaceFly

Sistema per la raccomandazione dei posti in aereo in base ai profili personali estratti da social network

Docente del corso:
Prof. Ing. Floriano Scioscia
Prof.ssa Ing. Eufemia Tinelli

Studenti:
Di Leo Carlo
Salatino Angelo Antonio
Spina Micaela

Anno Accademico 2012/2013

Sommario

Indice delle figure

Indice dei codici

1 Introduzione

FaceFly é un portale web che dà la possibilità ad ogni suo utente di scegliere il proprio posto nell'aereo vicino ad altri passeggeri con i quali possieda degli interessi in comune.

Questo portale è stato realizzato come progetto di fine corso per la disciplina di Linguaggi e Tecnologie Web.

Con la rapida diffusione di voli low cost, molte persone preferiscono spostarsi con gli aerei, anche tra due località molto vicine tra loro. Inoltre, spesso per motivi di lavoro, si é portati a viaggiare da soli. Questo potrebbe portare i passeggeri a dover sopportare viaggi sull'aereo in completa solitudine, o, peggio ancora, ogni passeggero corre il rischio di sedersi accanto a qualcuno che abbia interessi o gusti diametralmente opposti ai suoi: in entrambi i casi, il viaggio risulterebbe estremamente sgradevole e noioso. Per prevenire il verificarsi di ciascuna di queste situazioni, si é pensato di sviluppare il portale FaceFly.

FaceFly è in grado di estrarre gli interessi dei suoi utenti da Facebook, il social network più usato al mondo, e di utilizzarli al fine di effettuare un match con tutti gli interessi degli altri passeggeri presenti su un dato volo. In questo modo il portale segnala ad ogni utente quali sono i passeggeri con i quali possiede il maggior grado di affinità, in modo tale che egli possa sfruttare tale informazione per scegliere il proprio posto nell'aereo vicino a coloro che gli assomigliano maggiormente. Ad ogni modo, l'utente può anche decidere di non avvalersi di queste informazioni e scegliere uno o più posti in maniera del tutto personale.

L'obiettivo di FaceFly è stimolare la conversazione tra passeggeri vicini, rendendo il viaggio confortevole, meno stressante e dando anche l'impressione che sia di breve durata.

La presente relazione è strutturata in questo modo: dopo una preliminare introduzione sul lavoro svolto ed un breve excursus di quelle che sono state le idee fondamentali per lo sviluppo del portale FaceFly, segue la sezione "Progettazione", il cui scopo principale è la descrizione, ad un livello teorico e qualitativo, della struttura del portale, dei suoi più importanti componenti, e delle funzionalità offerte. In seguito, sarà possibile trovare la parte di "Implementazione", che ha l'obiettivo di mostrare la struttura del portale al lettore in modo dettagliato e tecnico, che possa chiarire esattamente come è stato realizzato il lavoro, quali strumenti si sono utilizzati per lo sviluppo, quali le tecnologie adottate, le scelte effettuate nel corso del lavoro e le motivazioni legate a tali scelte. Infine, a concludere questa relazione vi sarà una sezione di "Sviluppi futuri" che presenti possibili migliorie apportabili al sistema in un futuro, per chiunque abbia intenzione di espandere ulteriormente FaceFly.

2 Idee sullo sviluppo

La funzionalità del portale si basa su un assunzione fondamentale: il sito non è un *reseller* di biglietti aerei per compagnie terze e/o affiliate ma fornisce solo il servizio di prenotazione dei posti a sedere negli aerei associati a voli di compagnie che si presuppone, a monte, abbiano deciso di avvalersi di tale servizio. Partendo da questo presupposto, l'utente per poter utilizzare il portale deve essere già in possesso del biglietto, a corredo del quale saranno stati distribuiti un ID della prenotazione e una password, che consentiranno l'accesso al sito. Inseriti i dati richiesti e successivamente loggati, si verrà ridiretti sulla pagina che gestisce l'aereo del volo nel quale è stata fatta la prenotazione, e al contempo verrà chiesto di condividere alcune informazioni, ovvero i Likes su Facebook. E' opportuno specificare che ad ogni ID prenotazione assegnato ad un utente, possono corrispondere più posti a sedere effettivamente prenotati all'interno dell'aereo: questo significa che, al momento del login sul portale, l'utente dovrà scegliere necessariamente tanti posti quanti sono previsti dal codice identificativo della sua prenotazione. Questa precisazione va fatta, e si basa essenzialmente sull'osservazione che, attualmente, i maggiori siti di vendita di biglietti aerei online forniscono all'utenza l'opportunità di acquistare, in un'unica transazione, quanti biglietti si vogliono.

3 Progettazione

3.1 Introduzione

Una delle idee principali su cui si basa questo progetto è che ci siano due classi di utenti. La prima classe individua coloro che hanno acquistato il biglietto aereo e vogliono fruire del servizio, la seconda classe è formata dagli amministratori del sistema ai quali saranno forniti dei permessi speciali che gli consentano di effettuare operazioni differenti sul portale.
Per sviluppare questa idea si è dovuto ricorrere alla creazione di due sezioni differenti, una per ogni classe d'utente, giustificata dal fatto che i casi d'uso dei due utenti sono nella quasi totalità differenti, come mostrato dal diagramma dei casi d'uso in Figura 1.
Dalla stessa figura si nota come l'attore generico Utente si specializzi in Amministratore e in Cliente, interagenti per mezzo di alcuni casi d'uso con gli attori Volo, News, Commenti, Prenotazione e Aereo. Qui di seguito verranno descritte la sezione amministratore (par. 3.2) e la sezione utente (par. 3.3) che descriveranno da cosa sono composte tali sezioni. Infine nel paragrafo 3.4 verrà descritto il sistema software con l'ausilio del diagramma delle componenti.

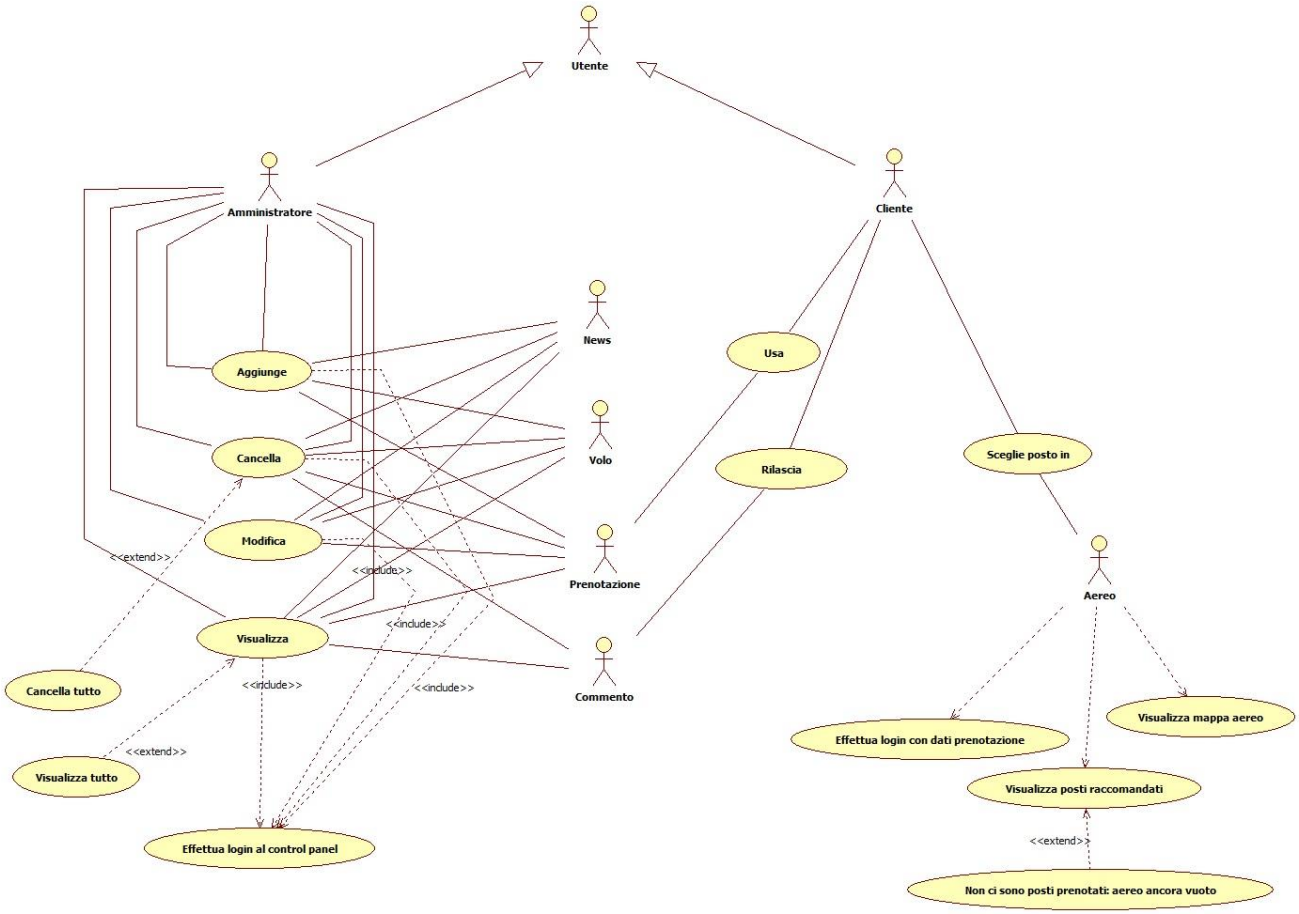

Figura 1: Diagramma dei casi d'uso di FaceFly

3.2 Sezione amministrazione

Intuibilmente l'amministratore è colui che si occupa di manutenere il sistema, gestendo quindi tutto ciò che concerne i voli presenti nel database, le prenotazioni associate ad essi, i commenti rilasciati circa la qualità del servizio percepita dai fruitori del sistema, le news da mostrare nella homepage; l'amministratore ha anche la possibilità di controllare gli altri amministratori, eventualmente cancellandone o inserendone di nuovi.
L'utente amministratore per poter utilizzare questa sezione dovrà fare accesso al pannello di controllo dell'intero portale. Tale pannello di controllo fornisce una suite di funzionalità che permettono di gestire il sito. Le credenziali di amministratore, che quindi danno la possibilità di accesso al sito, si ottengono durante l'installazione del sistema oppure attraverso un altro amministratore già registrato al sistema che si assume la responsabilità di aggiungerne un altro.
Con dettaglio, in questa versione del portale, le funzionalità inserite e disponibili sono le seguenti:

- la gestione dei voli, ovvero la sezione attraverso cui l'amministratore del sito può aggiungere, visualizzare, modificare, eliminare voli dal database;
- la gestione delle prenotazioni, ovvero la sezione attraverso cui l'amministratore del sito può aggiungere, visualizzare, modificare, eliminare prenotazioni dal database, nell'ambito dei soli voli disponibili al momento della prenotazione;
- la gestione delle news, ovvero la sezione attraverso cui l'amministratore del sito può aggiungere, visualizzare, modificare, eliminare news dal database, relative agli ultimi voli aggiunti, o a quelli a partenza ravvicinata che possiedono ancora posti liberi;
- la gestione dei commenti degli utenti, ovvero la sezione attraverso cui l'amministratore del sito può visualizzare, eliminare i commenti inseriti da qualsiasi utente circa il servizio fornito da FaceFly;
- la gestione del proprio account, ovvero la sezione che consente all'amministratore del sito di visualizzare i propri dati, eventualmente modificarli, aggiungere ulteriori amministratori o effettuare il logout.

A corredo delle precedenti, sono state inserite delle funzionalità secondarie che non hanno alcun fine di gestione del sito, ma servono solo per effettuare ricerche all'interno del database. In questa versione del portale, le funzionalità secondarie sono le seguenti:

- la ricerca dei voli presenti nel database sulla base di una determinata data;
- la ricerca dei voli "in scadenza", ovvero quelli la cui data di partenza è vicina, tuttavia presentano ancora posti disponibili per la prenotazione;
- la visualizzazione delle ultime modifiche effettuate sul database in termini degli ultimi voli aggiunti;
- la ricerca di tutti i possibili itinerari tracciabili nel nostro database specificandone luogo di partenza e luogo di arrivo;
- il controllo dello stato di uno specifico volo.

La sezione amministrazione fornisce al portale un elevato dinamismo perché tende a ridurre la presenza di pagine statiche e consente la manutenzione del sito anche ai "non esperti" del settore.

3.3 Sezione utente

Nella sezione utente, differentemente da quanto accade nella sezione amministratore, è necessario accedere con un ID ticket e una password, che saranno validi per la sola sessione di prenotazione dei posti. Nelle assunzioni fatte precedentemente, si è deciso che questi dati per l'accesso al sito vengano forniti direttamente all'atto dell'acquisto del biglietto, dalla compagnia aerea stessa. L'accesso effettuato in seguito alla prenotazione mostrerà solo lo status dei posti sull'aereo.
Ciò premesso, a login effettuato da parte di un utente, il sistema da subito è a conoscenza del codice identificativo del volo su cui è stata fatta la prenotazione e la quantità dei posti che sono stati prenotati. Questi dati sono necessari per preparare la homepage che consentirà di effettuare la prenotazione. Tuttavia, prima di impaginare la homepage, il sistema richiede che venga effettuato l'accesso a Facebook, il quale chiederà il consenso al trattamento dei dati, per scaricare la lista dei *Likes* per mezzo di un oggetto JSON.
A consenso ottenuto viene caricata la homepage e in background viene avviata una richiesta asincrona alla Servlet (XMLHttpRequest). Come parametro di tale chiamata c'è l'oggetto JSON dei likes dell'utente ricevuti da Facebook. La servlet con tutti i dettagli del caso sarà descritta nel paragrafo 4.3.
Nell'attesa che la servlet elabori e restituisca i risultati, l'utente può comunque interagire con la pagina web caricata, la quale presenta la mappa dell'aereo con i posti occupati in giallo, quelli liberi in blu, e quelli selezionati in verde. Cliccando su dei posti occupati, in apposite sezioni verranno mostrati i risultati dell'utente che ha occupato quel determinato posto evitando di mostrare usernames o altri dati sensibili per rispettare la privacy degli utenti che hanno già usufruito del servizio.
Al termine dell'elaborazione dei dati da parte della servlet, si riceveranno i risultati, sulla base dei quali verrà generato un rank tra l'utente loggato e gli utenti già prenotati per quel volo (ogni dettagli in merito è presente nel paragrafo 4.4). In questa circostanze vi è l'eccezione sulla prima prenotazione per un determinato volo, in quanto non essendoci precedenti prenotazioni non è possibile fornire un rank.
Per quanto concerne la fase finale, quindi la prenotazione, l'utente non è in alcun modo vincolato alla scelta del posti vicino all'utente con cui possiede la maggior affinità. Il sistema progettato ambisce alla sola raccomandazione.
Il diagramma di sequenza (Figura 2) mostra il workflow necessario per prenotare i posti a sedere a partire dalla fase di login.
Quando l'utente conferma nel database verranno salvati tutti i dati necessari per rendere effettiva la fase di prenotazione. Nella tabella del volo in corrispondenza dei posti prenotati verranno salvati l'username di Facebook (trattato con la massima riservatezza) e l'oggetto JSON dei risultati ottenuti dalla servlet cosicché possa essere usato per generare il ranking per gli utenti successivi. Un esempio di risultato ottenuto dalla servlet è presente nel Codice 5.

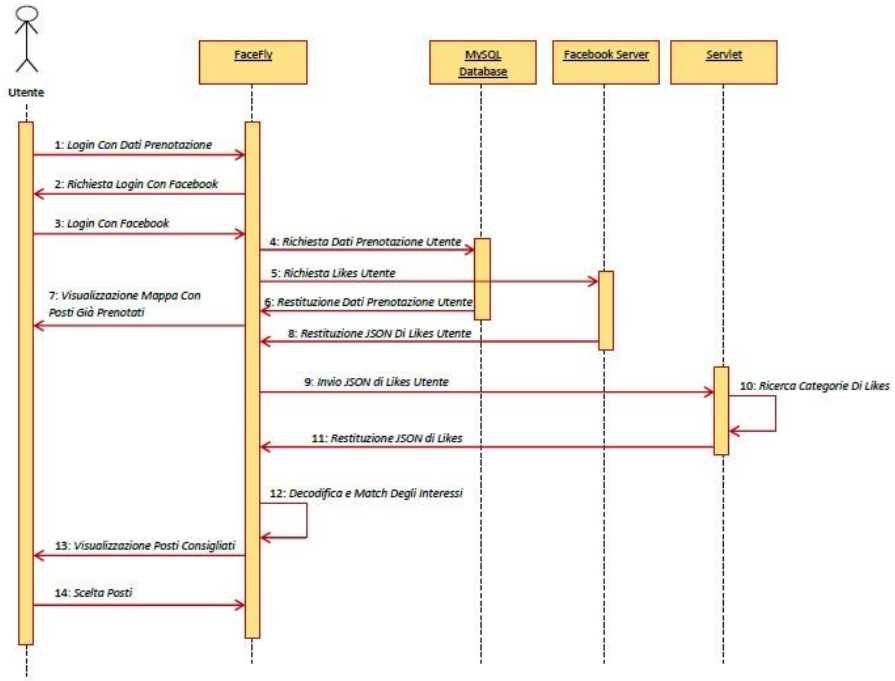

Figura 2: Diagramma di sequenza del caso d'uso della prenotazione dei posti da parte dell'utente

3.4 Sistema software

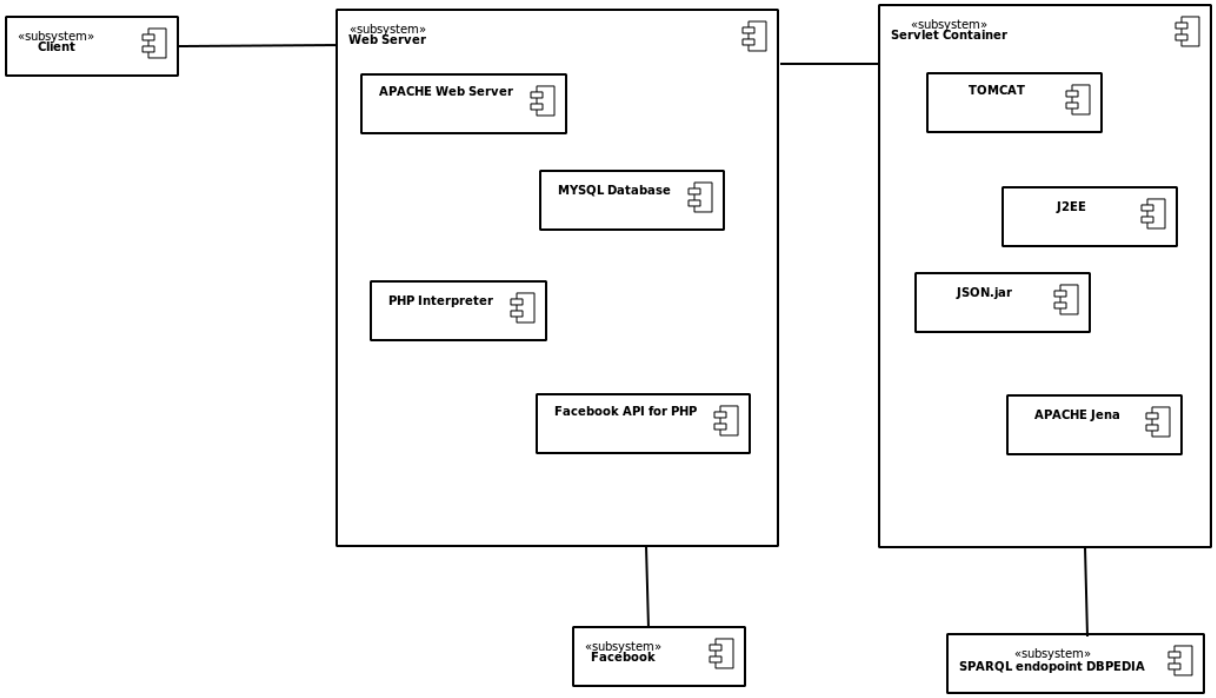

Figura 3: Diagramma delle componenti.

Come si denota dalla Figura 3, possiamo suddividere il nostro sistema in due sottosistemi, Web Server e Servlet Container, che si interfacciano con 3 entità esterne al sistema: riceve le richieste dal browser del client, richiede e riceve dati da Facebook e effettua query SPARQL all'endpoint SPARQL di DBPEDIA.

Per lo sviluppo del sistema si è utilizzato la piattaforma XAMPP che fornisce alcuni dei moduli più essenziali in un unico pacchetto.
Il modulo APACHE Web Server astrae tutta la gestione delle richieste HTTP, protocollo con cui il client comunica col Web Server, permettendoci di concentrarci sul processamento della richiesta stessa.
Per la creazione delle pagine dinamiche, nonché per la gestione dei dati chiesti e/o ottenuti da Facebook si è utilizzato il linguaggio server-side PHP e il PHP Interpreter fornito da XAMPP.
Il DBMS utilizzato è MySQL, perché free e perché presente in XAMPP. In questa applicazione il DBMS gira sulla stessa macchina del Web Server, ma è comunque possibile migrare su un'altra macchina.
Le Facebook Graph Api permettono l'accesso ai nodi del grafo Facebook astraendo la parte di comunicazione con il server e accedendo alle informazioni utilizzando lo standard JSON.
Il Web Server e la servlet comunicano utilizzando il protocollo HTTP, ma anche in questo caso la gestione di basso livello della comunicazione ci viene nascosta grazie all'utilizzo della Java 2 Enterprise Edition, che fornisce API per lo sviluppo di applicazioni web a più alto livello. L'utilizzo del linguaggio Java ci permette inoltre di creare un'applicazione più scattante e l'utilizzo di numerose librerie per la gestione dei dati.
Tomcat è un servlet-container che fornisce un approccio semplificato al caricamento della servlet, nonché supporta le funzionalità di un web server.
Per lo scambio dei dati tra application server e web server viene utilizzato il formato JSON, quindi ci siamo serviti di una libreria per Java per la gestione come Arraylist del formato JSON.
Infine il modulo APACHE Jena viene utilizzato per l'invio delle query all'endpoint SPARQL di DBPEDIA. Anche in questo caso buona parte della comunicazione viene nascosta allo sviluppatore.

4 Implementazione

4.1 Tecnologie usate

Il portale FaceFly ha permesso l'uso di molte tecnologie web attuali, cosa gradita se fatta a livello accademico i cui scopi principali sono l'apprendimento. Per la realizzazione del progetto ci si è serviti del linguaggio HTML per formattare i documenti. Essendo le pagine non totalmente statiche, perché variano in funzione dell'utente, ci si è serviti di un linguaggio server side come il PHP. Con il linguaggio PHP vengono gestite tutte le query al database, gestione degli utenti, gestione dei voli, gestione degli amministratori, e così via. Il linguaggio CSS è stato usato per definire la formattazione delle pagine in modo tale da separare nettamente il contenuto informativo dalle direttive di presentazione. Il linguaggio XML è stato usato nella versione RSS come sistema per la gestione delle news. Il JavaScript è stato usato come linguaggio client-side per eseguire script in locale che non richiedono l'intervento del server come la richiesta di login, animazioni, e così via. Il jQuery sottoforma di libreria é stato usato per velocizzare e semplificare l'implementazione di alcune funzioni JavaScript. Il linguaggio SQL è stato usato per accedere al database, così da poter creare le tabelle, popolarle e interrogarle. Gli oggetti JSON sono gli strumenti usati per la ricezione della struttura dati serializzata da parte di Facebook e la comunicazione tra l'applicazione web e la servlet. Tale servlet è stata interamente sviluppata in Java la quale include la libreria Jena per gestire la comunicazione della servlet con l'endpoint SPARQL di DBPEDIA. Il linguaggio SPARQL è stato di conseguenza utilizzato per interrogare DBPEDIA. La servlet utilizza inoltre una libreria per la codifica e la decodifica di oggetti JSON.
Come web server sono stati utilizzati Apache HTTP Server e Apache Tomcat (descritta nel par. 4.3.2). Il primo ci ha permesso di ospitare le pagine in formato HTML e PHP (caricando l'opportuno modulo), mentre il secondo ci ha permesso di ospitare applicazioni sviluppate in Java.

4.2 Facebook API

4.2.1 Introduzione

Nella sezione sviluppatori di Facebook, ed in particolare la sezione web (Facebook APIs), ci sono cinque modalità di interazione tra Facebook e il proprio sito web. Il primo è chiamato Login, il quale rende semplice la connessione tra gli utenti e il proprio sito web, velocizzando il processo di registrazione e costruire un sistema funzionale in pochi minuti.
La seconda api è il Graph API che presenta una vista semplice e consistente del grafo sociale di Facebook, rappresentando uniformemente gli oggetti nel grafo e le connessioni tra di loro.
La terza api è chiamata FQL, che rappresenta un linguaggio di query di Facebook e permette di usare un'interfaccia SQL-style per richiedere i dati esposti dal Graph API. Fornisce caratteristiche aggiunte non disponibili in Graph API, come l'annidamento delle query.
La quarta api è quella dell'Open Graph, che consente ad applicazioni di aggiungere contenuti attraverso un api strutturata e fortemente tipizzata.
L'ultima ma non meno importante è la Legacy REST API che permette di interagire con il sito web di Facebook programmando delle richieste HTTP.
Stando agli obiettivi preposti per la progettazione del portale, dopo un'attenta analisi si è optato per l'adozione delle Graph API. Tale scelta è giustificabile con le note di Facebook, il quale mette spesso in risalto che Graph API, FQL e Legacy REST API hanno funzionalità sovrapposte perché tutte consentono l'accesso al grafo, ma le ultime due sono in via di deprecazione. Viene fatto notare anche che applicazioni "future proof" dovrebbero contenere le Graph API.
Qui di seguito verrà data una brevissima descrizione della Graph API, cosi da rendere chiari alcuni concetti propedeutici per comprendere come sia stata realizzatta l'app per l'aquisizione dei likes.

4.2.2 Perché i likes?

Il like di Facebook è lo strumento più veloce è più utilizzato all'interno del social network. Attraverso il like un utente esprime una propria di preferenza, indica le cose che più gli interessano. Quando un utente si iscrive a Facebook, inizialmente è disorientato e non conosce le potenzialità sull'uso del like. A regime, quando l'utente comprende la sua potenzialità e scegliendo con accuratezza dove usare il like non fa altro che trasmettere la sua personalità sul web. Questo spiega perché molte applicazioni odierne basate sui social network si basano spesso sui likes di Facebook. I likes forniscono perciò un valido strumento per fare profilazione e quindi pilotare le pubblicità.

4.2.3 Le Graph API

Facebook dal 2010 a questa parte si presenta sotto la veste di un "social graph". Questo nome prende spunto anche dalle strutture matematiche quali i grafi dove ci sono i nodi e le relazioni tra di essi. Nel contesto di Facebook i cui nodi sono elementi quali: utenti, fan page, gruppi, foto e video, collegati tra di loro per mezzo di archi che rappresentano le relazioni tra essi. Ad esempio, due nodi utenti possono essere collegati tra loro da una connessione, che indica una relazione di amicizia. Ogni oggetto nel grafo sociale è caratterizzato da un identificativo univoco ID, e dunque sarà possibile accedere alle proprietà di tale oggetto semplicemente richiedendole a: *http://graph.facebook.com/ID*.
Un oggetto utente, ad esempio, che ha come ID 0000, avrà proprietà recuperabili all'indirizzo: *http://graph.facebook.com/0000*.
Alternativamente, per un qualsiasi nodo, se ne possono richiamare le proprietà anche sfruttando il relativo username, piuttosto che l'ID. Tutte le risposte a richieste di questo tipo sono oggetti JSON. Tuttavia, talvolta, interrogando le Graph API con richieste non autorizzate, si ottengono risposte indesiderate, come quella mostrata nel Codice 1.

```
{
"error": {
      "type": "QueryParseException",
      "message": "An active access token must be used to query information about the current user."
         }
}
```

Codice 1: Esempio di messaggio di errore da parte di Facebook, inserito in un oggetto JSON

Questo succede perché alcune richieste alle Graph API, per motivi di sicurezza, hanno bisogno di un access token per essere autenticate e per produrre un output che non dia errore. Tale token è ottenibile richiedendo un codice di autorizzazione che, in seguito, rimanderemo al server, che in risposta ci fornirà un access token valido. Inoltre, spesso l'ottenimento dell'access token è comunque fondamentale, in quanto in linea di massima le Graph API consentono di accedere esclusivamente alle informazioni pubbliche di un oggetto: se invece si vuole fare accesso a informazioni specifiche, quali ad esempio l'indirizzo mail di un utente, è necessario richiedere un permesso specifico.
E' inoltre possibile usare le API per leggere solo campi specifici, ottenere l'immagine di ogni oggetto, leggere i metadati di un oggetto e ottenere gli update in tempo reale di ogni cambiamento. Si può richiedere in una singola query informazioni di più oggetti indicando più ID come parametro "ids": ad esempio, *http://graph.facebook.com/?ids=ID1,ID2*.

4.2.4 Creare un'app di Facebook

Per creare un app è necessario registrarsi come sviluppatore Facebook, collegarsi alla pagina developers e successivamente in apps[1]. Successivamente creare una nuova app, inserendo un nome e un namespace validi. Ad avvenuta conferma, si viene re-diretti verso la pagina dei settings della propria applicazione dove si richiedono le informazioni base (Figura 4), e come integrare la propria app con Facebook (Figura 5).

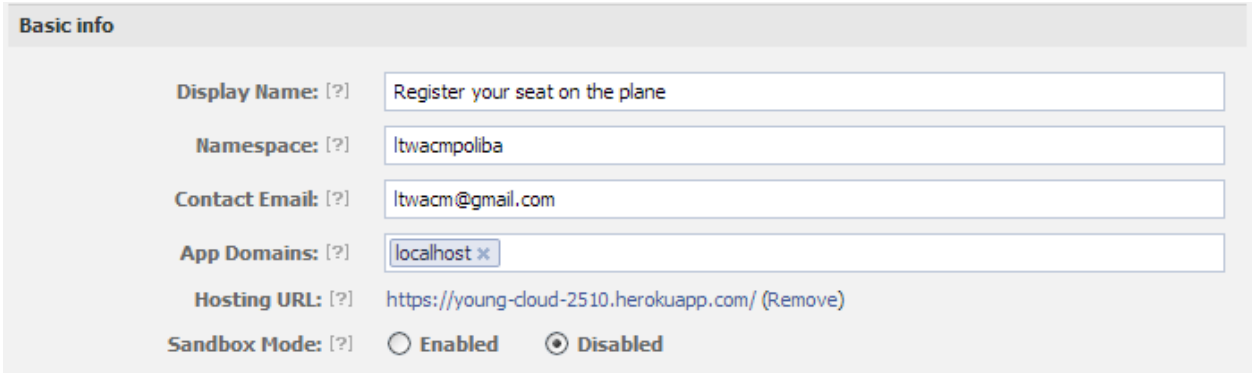

Figura 4: Basic info dell'app di Facebook.

[1] https://developers.facebook.com/apps

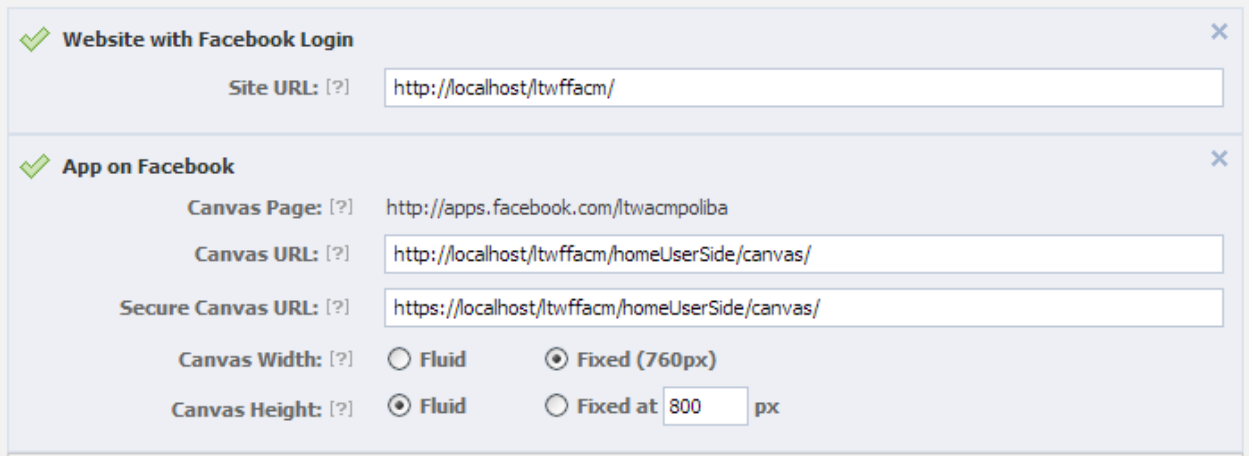

Figura 5: Integrazione tra l'app e Facebook

Nella sezione Permessi dell'applicazione si impostano tutti i permessi utente che dovrà fornire per autenticarsi all'app. I permessi i questione sono i likes (**user_likes**) e il proprio nome utente (contenuto in **user_about_me**) come mostrato in Figura 6.

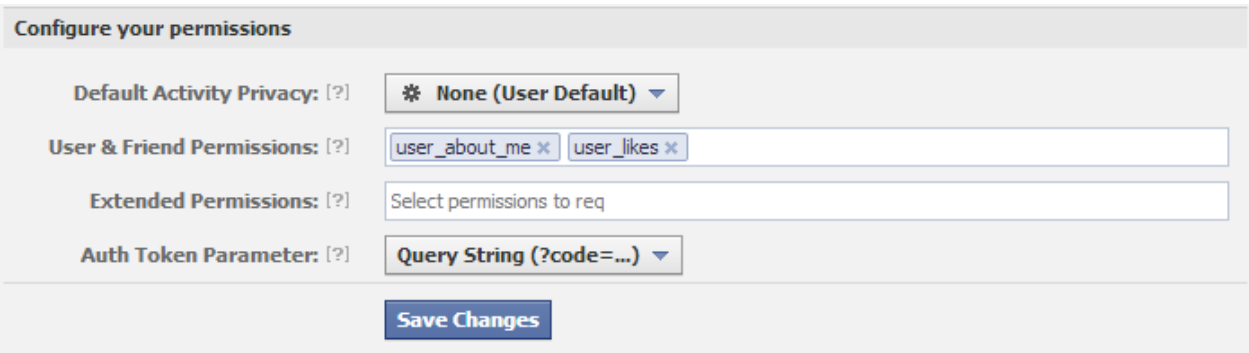

Figura 6: Configurazione dei permessi per l'app.

Completata la fase di setting e ritornando nella sezione Basic, si dovrebbe notare che l'applicazione è live, come mostrato nella Figura 7.

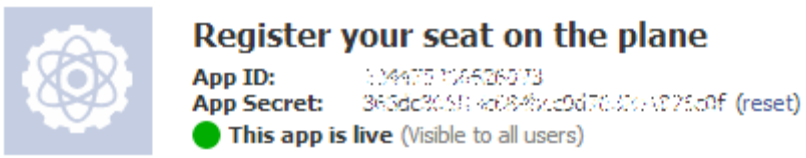

Figura 7: Status dell'applicazione

Per completare è necessario copiare e incollare l'**App ID** e **App Secret** nelle API fornite nella sezione Facebook SDK for PHP[2].

Per testare l'applicazione è stato usato il Graph API Explorer[3] che dopo aver selezionato l'applicazione, permette di richiedere un token valido per l'autenticazione e successivamente di fare un GET alla Graph API per richiedere i dati al proprio profilo. Un esempio ridotto dei risultati ottenuti è presente nel Codice 5.

[2] https://developers.facebook.com/docs/reference/php/
[3] https://developers.facebook.com/tools/explorer

```
{
 "id": "633501194",
 "name": "Angelo Antonio Salatino",
 "likes": {
  "data": [
   {
    "category": "Tv show",
    "name": "House",
    "id": "7608631709",
    "created_time": "2013-02-27T01:39:13+0000"
   },
   {
    "category": "Athlete",
    "name": "Gianluigi Buffon",
    "id": "280290668726284",
    "created_time": "2012-05-17T08:04:03+0000"
   },
   {
    "category": "Actor/director",
    "name": "Cobie Smulders",
    "id": "170075663114862",
    "created_time": "2012-04-17T21:59:46+0000"
   },
   {
    "category": "Musician/band",
    "name": "Caparezza",
    "id": "262849827085497",
    "created_time": "2011-10-21T12:12:24+0000"
   },
        ...,
   {
    "category": "Tv show",
    "name": "Fringe",
    "id": "15576613667",
    "created_time": "2011-03-02T19:14:17+0000"
   }
   ]
 }
}
```


Codice 2: Esempio di oggetto JSON restituito dalle API di Facebook

4.3 Servlet

4.3.1 Introduzione

Per processare i likes ottenuti da Facebook e estrarre gli interessi è stata implementata una piccola servlet.
Una servlet è un'applicativo scritto in Java la cui classe principale implementa l'interfaccia javax.servlet.GenericServlet o javax.servelt.HTTPServlet (quest'ultima utilizza il protocollo http per la comunicazione col web-server).
Si sarebbe potuto utilizzare ancora una volta il linguaggio PHP che è stato usato per le pagine dinamiche nel resto del progetto ma si è scelto di usare il linguaggio Java sia per dividere il web-server dalla parte applicativa, sia per ottenere uno speed-up delle prestazioni in quanto il linguaggio java è in parte compilato mentre il linguaggio PHP è interamente interpretato.
Una servlet necessita di un componente particolare, il Web Container (o Servlet Container) tra le altre cose gestisce le richieste in entrata e le risposte della servlet e può gestire più servlet contemporaneamente.
Vari sono gli applicativi presenti che offrono il servizio di Web Container, tra questi c'è Tomcat. Si è scelti Tomcat perché è uno dei Servlet Container più diffusi al mondo, è gratuito ed è presente nel framework Xampp insieme ad Apache Web Server, MySQL e PHP.
Qui di seguito verrà data una breve descrizione su Apache Tomcat, successivamente si forniranno dettagli su come si imposta una servlet e come essa sia stata implementata affinché fosse in grado di eseguire query SPARQL per ottenere i risultati mostrati alla fine di questa sezione.

4.3.2 Apache Tomcat

In Tomcat è molto semplice inserire nuove applicazioni. Ogni applicazione è contenuta a partire dal path $CATALINA_HOME/webapps dove $CATALINA_HOME è la variabile d'ambiente contenente il path della cartella d'installazione di Tomcat.
Anche se il deploy può essere fatto manualmente, creando la cartella che conterrà la servlet e tutta la struttura, è consigliabile creare, a partire dal progetto della servlet, il file .war (Web application ARchive) corrispondente e permettere a Tomcat di effettuare il deployment in maniera automatica.
Tomcat effettua il deploy dei file .war inseriti inseriti nella cartella $CATALINA_HOME/webapps all'avvio (deploy statico) se l'attributo "deployOnStartup" è settato a true. E' possibile effettuare il deploy anche senza spegnere Tomcat (deploy dinamico) se l'attributo "autodeploy" è settato a true.
Infine è possibile servirsi del Tomcat Manager che fornisce una GUI per l'inserimento delle servlet.

4.3.3 Impostazioni

Per l'implementazione della servlet ci siamo serviti di NetBeans perché non aveva senso spostarsi in un'altro ambiente dato che tutto il portale è stato implementato con esso.
Inoltre NetBeans permette la creazione di una servlet in maniera semplice ed automatica, permettendo all'utilizzatore finale di dover implementare il solo metodo *processRequest(HttpServletRequest request,HttpServletResponse response)* nel quale abbiamo inglobato la logica applicativa.
Per implementare una servlet utilizzando Netbeans occorre creare un nuovo progetto di web-server, settare il server utilizzato (Tomcat in questo caso) e creare una nuova classe che implementi l'interfaccia di una servlet.
Per il progetto ci si è inoltre serviti delle librerie fornite dal progetto *Apache Jena* per interrogare *DBpedia* e di una libreria per gestire in maniera semplificata il formato JSON utile alla comunicazione con la web-server.

4.3.4 Implementazione

Alla servlet viene inviata una richiesta HTTP, più precisamente una POST poiché la quantità di dati da processare può superare i limiti di una GET. Il body della richiesta contiene i likes, codificati come stringa di array di oggetti JSON e il nome dell'utente che si sta processando. Quest'ultima informazione serve al fine di compilare un file di log utile per eventuali analisi dei dati o debug in caso di failure .

La stringa contenente i likes viene analizzata e decodificata con l'aiuto di una libreria che permette, in Java, di gestire gli oggetti JSON.

In questa fase viene fatta una prima scrematura dei dati, eliminando quei likes che hanno dei duplicati.

I dati vengono quindi divisi in due categorie sfruttando i campi "Movie" e "TV Show" che Facebook mette a disposizione: tutti i likes con categoria "Movie" e "TV Show" verranno processati insieme in quanto afferenti alla stessa macro categoria mentre tutti gli altri likes verranno processati come possibili persone (pochi lo saranno, ma Facebook non offre garanzie per discernere).

Vengono effettuate le relative query SPARQL e infine i risultati vengono nuovamente codificati sottoforma di stringa JSON e inviati nuovamente alla web-server.

4.3.5 Query SPARQL

Di seguito vengono riportate le query SPARQL utilizzate per interrogare DBPEDIA.

Dato N il numero di likes dell'utente, vengono effettuate N query a DBPEDIA, ciascuna dipendente dalla macro categoria del like. Per quanto riguarda la macro categoria Movie - TV Show la query utilizzata nel Codice 3.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ontology: <http://dbpedia.org/ontology/>
PREFIX dcterms: <http://purl.org/dc/terms/>

SELECT distinct ?subject
WHERE{
	{
		?film rdf:type ontology:Film .
		?film dcterms:subject ?subject .
		?film rdfs:label ?label .
		?label <bif:contains> "' movieList.get(i)'"
	}
	UNION
	{
		?film rdf:type ontology:TelevisionShow .
		?film dcterms:subject ?subject .
		?film rdfs:label ?label .
		?label <bif:contains> "' movieList.get(i)'"
	}
}
```

Codice 3: Query SPARQL per la richiesta dei generi dei film da DBpedia

Nella prima parte vengono inclusi i prefissi dei namespace contenenti la descrizione delle risorse, delle classi e delle proprietà utilizzate ed è comune a qualsiasi query.

La query estrae tutte le risorse appartenenti alla classe film o serie tv, quindi la loro categoria appartenente al namespace dcterms e cerca per i risultati il cui nome contiene il nome della pagina a cui l'utente ha messo il like. Vengono considerate solo le categorie che tuttavia sono tante e non enumerabili. Perciò all'interno di ciascun risultato viene ricercata una particolare parola chiave che rappresenterà il nostro genere.

I generi utilizzati sono: Action, Comedy, Family, History, Mystery, War, Adventure, Crime, Fantasy, Western, Animation, Noir, Documentary, Drama, Musical, Romance, Thriller.

Per quanto riguarda la macro categoria Persone, il comportamento è analogo alla query precedente. La query utilizzata è mostrata nel Codice 4.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ontology: <http://dbpedia.org/ontology/>

SELECT distinct ?type
WHERE{
        ?people rdf:type owl:Thing .
        ?people rdf:type ?type .
        ?people rdfs:label ?label .
        ?type rdfs:subClassOf ontology:Person .
        ?label <bif:contains> "' peopleList.get(i)'"
    }
```

Codice 4: Query SPARQL per la richiesta delle persone da DBpedia

Le categorie di persone considerate sono: OfficeHolder, MilitaryPerson, Philosopher, FictionalCharacter, Judge, Model, PlayboyPlaymate, PokerPlayer, Scientist, Politician, Architect, OrganisationMember, SoccerManager, CollegeCoach, Astronaut, Ambassador, Journalist, Artist, Athlete, Monarch, Cleric, Royalty, Criminal.

4.3.6 Risultati restituiti

Per il formato di risposta e quindi dei risultati da restituire si è deciso di utilizzare un oggetto JSON, in modo tale che ci sia un'uniformità tra i dati inviati in fase di richiesta e di risposta. I vettori memorizzati saranno quindi convertiti con il metodo *add* della classe *JSONArray* in tale formato e inviati al richiedente. Un esempio di oggetto JSON inviato al richiedente è mostrato nel Codice 5.


```
[{"Crime":0.069767445,"Romance":0.046511628,"Animation":0.023255814,"Comedy":0.093023255,"Mystery":0.046511628,"Western":0.023255814,"Noir":0.0,"Action":0.093023255,"Adventure":0.046511628,"History":0.046511628,"War":0.046511628,"Fantasy":0.069767445,"Musical":0.023255814,"Family":0.023255814,"Thriller":0.093023255,"Drama":0.18604651,"Documentary":0.069767445},{"Journalist":0.0,"PokerPlayer":0.016949153,"Astronaut":0.0,"CollegeCoach":0.06779661,"Model":0.0,"Politician":0.050847456,"OfficeHolder":0.033898305,"Architect":0.0,"Royalty":0.0,"Cleric":0.016949153,"Ambassador":0.0,"Philosopher":0.016949153,"Monarch":0.016949153,"Criminal":0.016949153,"Athlete":0.33898306,"Judge":0.0,"SoccerManager":0.033898305,"Scientist":0.016949153,"FictionalCharacter":0.084745765,"MilitaryPerson":0.033898305,"PlayboyPlaymate":0.0,"Artist":0.20338982,"OrganisationMember":0.050847456}]
```


Codice 5: Esempio di oggetto JSON restituito dalla servlet per un dato utente

4.4 Ranking

4.4.1 Introduzione

Il ranking è una tra le parti più importanti del portale, perché va a calcolare l'affinità tra l'utente che si è appena loggato e gli utenti già prenotati nel sistema per quel dato volo. In questa versione del portale si è deciso di misurare l'affinità tra due utenti attraverso la distanza euclidea. La fase di ranking è ancora un problema aperto perché per ottenere una determinata stabilità sono necessari molti campioni su cui effettuare dei test e analisi dei dati. In versioni successive è possibile che tale ranking si possa basare su altri meccanismi.

4.4.2 Funzionamento

Per comprendere il suo funzionamento è necessario riportarci al momento in cui viene caricata la homepage, ovvero quando il sistema carica dal database, con esattezza dalla tabella del volo, tutti gli oggetti JSON degli utenti già prenotati andando a collezionarli in un nuovo array JSON, come mostrato nel Codice 6. Si rammenta che tutto ciò avviene contemporaneamente all'invio della richiesta asincrona alla servlet.

```
[{"spot":"A1","result":JSON_A1},
{"spot":"A2","result":JSON_A2},
...,
{"spot":"xy","result":JSON_xy}]
```

Codice 6: Oggetto JSON costruito a partire da tutti gli oggetti JSON degli utenti prenotati per un dato volo

Tale array è stato così strutturato, per avere un riferimento tra un dato oggetto JSON_xy caricato dal database e l'ID del posto associato. Come già anticipato nel paragrafo 3.3 tale oggetto JSON è il risultato ottenuto dalla servlet (mostrato nel Codice 5) che viene direttamente memorizzato nel database, non mostrando alcun riferimento all'ID del posto.

Quando la servlet restituisce i risultati dell'utente loggato, essendo in possesso di tutti i dati necessari, il sistema effettua una chiamata alla funzione *matchingInterests_v2*.

A tale funzione vengono passati due oggetti:

- Array JSON degli utenti prenotati al volo;
- Oggetto JSON dell'utente, risultato dell'elaborazione della servlet.

Fondamentalmente tale funzione legge di quanti elementi è composto l'array JSON, per ogni elemento chiama la funzione *compareWithArray_v2* (mostrata nel Codice 7), passandogli l'oggetto JSON dell'utente loggato (jsonUser) oltre all'oggetto JSON memorizzato nell'array per un determinato ID posto (jsonDb). La funzione *compareWithArray_v2* non fa altro che calcolare la distanza euclidea, restituirne il risultato, che poi verrà memorizzato in un opportuno array associativo con l'ID posto nella funzione chiamante.

```
function compareWithArray_v2(jsonDb,jsonUser){
    var sum = 0;
    var dist = 0;
    for(var i = 0; i < jsonUser.length; i ++){
       var obj= jsonUser[i];
       var obj2= jsonDb[i];
        for (var property in obj) {
                sum += Math.pow(obj[property]-obj2[property],2);
            }
            dist = dist + Math.sqrt(sum);
    }
        dist = dist/2;
    return dist;
}
```

Codice 7: Funzione che calcola la distanza euclidea tra due oggetti JSON degli utenti

4.5 Database

4.5.1 Introduzione

Per la realizzazione del database alla base del progetto, ci si è serviti di MySQL.
My SQL è il database relazionale open-source più diffuso al mondo e rappresenta una soluzione ottimale per chiunque sia in cerca di un database veloce, flessibile, affidabile e, soprattutto, gratuito. Esso non va confuso con SQL, in quanto SQL è un linguaggio di interrogazione strutturata sviluppato da IBM, mentre MySQL è un "programma" che sfrutta SQL per interagire con i dati. MySQL inoltre non è esso stesso un database, ma è un sistema di gestione, o più precisamente un DBMS (Database Management System). Un DBMS può contenere e gestire più database gestendo simultaneamente le richieste di eventi diversi. MySQL è open-source, ovvero non solo è gratuito, ma permette anche di avere a disposizione il codice sorgente del programma e quindi è possibile modificarlo adattandolo alle specifiche esigenze del programmatore. E' molto diffuso e utilizzato soprattutto nelle applicazioni per il Web, principalmente insieme a Php, ma spesso anche con altri linguaggi di scripting come Asp, Cgi e così via.
Nel progetto si è scelto di usare MySQL con phpMyAdmin, un'applicazione Php libera che consente di amministrare in modo semplificato database di MySQL tramite un qualunque browser. L'applicazione è indirizzata sia agli amministratori del database sia agli utenti, in quanto è in grado di gestire i permessi prelevandoli direttamente dal database MySQL. Nel nostro caso, essa viene sfruttata esclusivamente dagli amministratori di FaceFly, gli unici ad avere il diritto di modificare le tabelle del database, che si chiama my_ltwacm. Per poter sfruttare phpMyAdmin è necessario avere in precedenza installato e configurato MySQL e Apache, e averli attivati. Per effettuare tutto ciò, si possono utilizzare pacchetti già preconfezionati che installano tutto in un singolo passaggio, come ad esempio XAMPP.
Il database alla base di FaceFly presenta la seguenti tabelle:

- Login: tabella nella quale viene mantenuta traccia di tutti gli amministratori del sistema. In particolare, ne vengono conservati i seguenti dati:
  - Id dell'amministratore (chiave primaria);
  - Nome;
  - Cognome;
  - Indirizzo e-mail;
  - Username per accedere al Control Panel di FaceFly;
  - Password per accedere al Control Panel di FaceFly;
  - Livello permessi (livello massimo, pari a 7).
- Voli: tabella nella quale si mantiene traccia di tutti i voli gestibili da FaceFly. In particolare, le informazioni dei voli che vengono salvate sono:
  - Codice del volo (chiave primaria);
  - Compagnia di viaggio;
  - Data della partenza;
  - Orario della partenza;
  - Orario previsto per l'arrivo;
  - Luogo della partenza;
  - Luogo di destinazione;
  - Numero della mappa dell'aereo associato al volo;
  - Numero dei posti della mappa considerata.

- Per ogni riga della tabella Voli, esisterà nel database una tabella che abbia nome pari al codice del volo della riga considerata. Ad esempio, per una entry della tabella voli che abbia codice AF1000, vi sarà nel database una tabella AF1000, contenente i seguenti campi:
    - Id posto (chiave primaria);
    - Id di Facebook dell'utente che ha prenotato quel particolare posto;
    - Stato del posto (occupato/non occupato);
    - Consigli associati al posto.
- Prenotazioni: tabella con la quale si tiene traccia delle prenotazioni associate ad un determinato volo. In particolare, le informazioni di cui si mantiene una copia nel database, sono:
    - Id della prenotazione (chiave primaria);
    - Password associata alla prenotazione;
    - Codice del volo per il quale si è effettuata la prenotazione;
    - Id di Facebook dell'utente che ha prenotato il/i posto/i;
    - Numero posti prenotati;
    - Flag che indica lo stato di prenotazione di un posto(prenotato/non prenotato).
- Commenti_utenti: tabella nella quale vengono salvati tutti i commenti inseriti dagli utenti utilizzatori del sito, per esprimere un feedback sulla qualità del servizio offerto da FaceFly. I campi di questa tabella sono:
    - Indirizzo e-mail dell'utente che rilascia il commento (chiave primaria);
    - Titolo del commento;
    - Testo del commento.

E' infine opportuno aggiungere che le tabelle Login, Voli, Prenotazioni e Commenti_utenti vengono create contemporaneamente alla prima installazione del database di FaceFly. In seguito a tale installazione, viene anche richiesto il salvataggio del primo amministratore di sistema, le cui informazioni costituiranno la prima entry della tabella Login. Successivamente, l'amministratore così loggato nel sistema, potrà accedere correttamente al Control Panel e aggiungere voli alla tabella Voli, e conseguenti prenotazioni alla tabella Prenotazioni, nell'ambito dei voli esistenti nel database. Potrà gestire le informazioni relative a voli e/o prenotazioni aggiunte in qualsiasi momento, modificandole o rimuovendole del tutto. La rimozione di un volo causerà, perché il vincolo di integrità referenziale del database venga rispettato, la conseguente rimozione di ogni prenotazione associata ad esso. L'amministratore potrà anche aggiungere altri amministratori al sistema, o rimuoverne alcuni già presenti. Potrà anche modificare il suo account, o eliminarlo del tutto. Un'ulteriore possibilità data all'amministratore di FaceFly è la visualizzazione e possibile rimozione di uno o tutti i commenti inseriti da comuni utenti che si siano interfacciati con il sito. Tali utenti infatti, potrebbero, dopo avere avuto a che fare con il sito FaceFly, sentire l'esigenza di rilasciare un proprio feedback relativo alla qualità del servizio, aggiungibile nel form del sito presente nella sezione Testimonials. E' altresì opportuno specificare che, essendo l'indirizzo e-mail dell'utente chiave primaria della tabella Commenti_utenti, nella quale vengono memorizzati tutti i commenti, ogni singolo utente potrà aggiungere uno e un solo commento.
Di seguito viene fornito uno schema concettuale del database finora descritto.

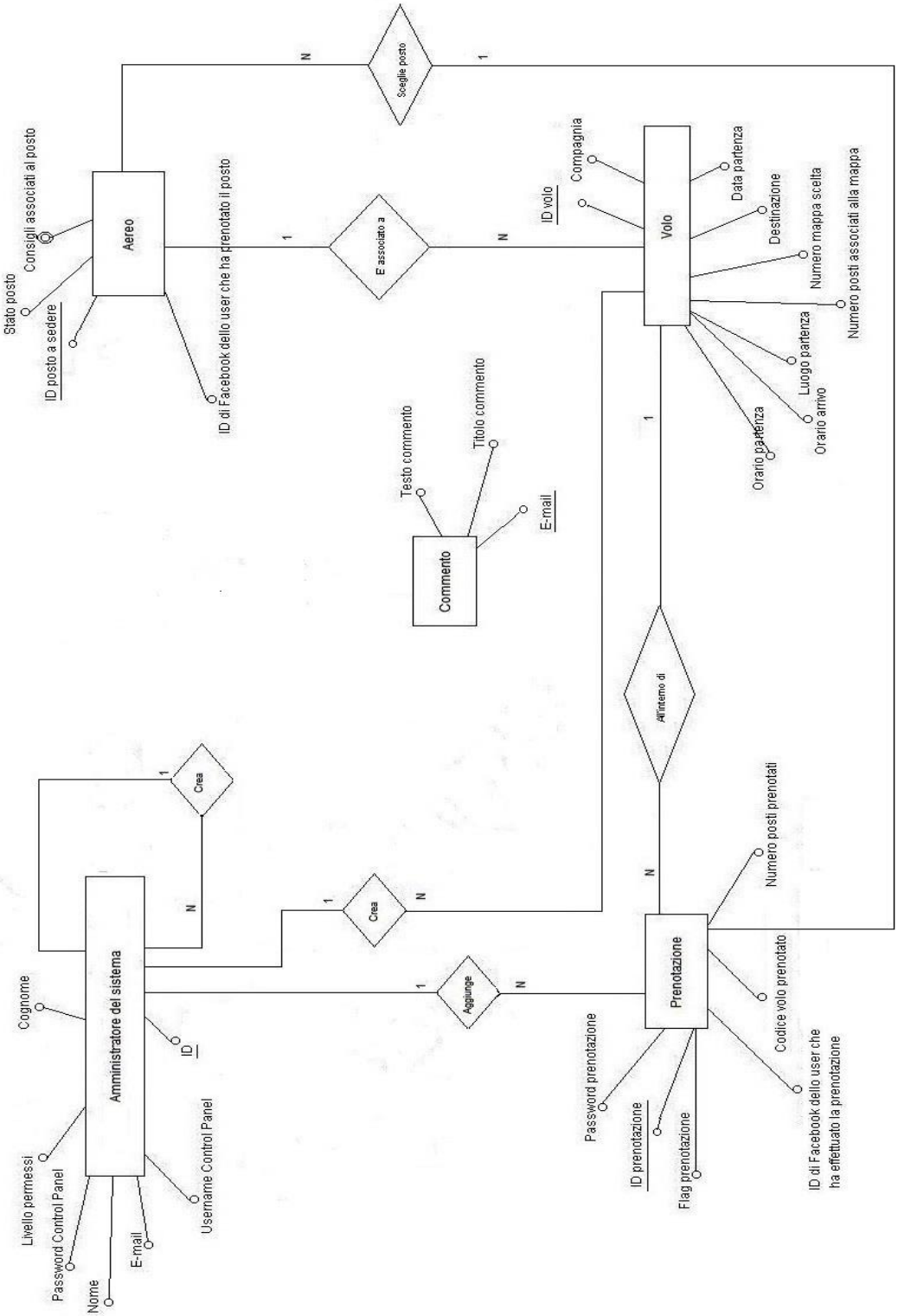

Figura 8: Modello Entità-Relazioni del database progettato per il database progettato

5 Implementazioni future e migliorie

L'attuale stato del portale non lo rende competitivo e pronto per essere messo sul web e far si che le compagnie aeree stipulino delle convenzioni. Alcuni sviluppi futuri e migliorie da apportare al sistema potrebbero essere:

- L'aumento della sicurezza del portale, evitare dunque che il portale stesso possa essere usato da intermediario per il recupero di dati sensibili da parte di terzi;
- L'introduzione di altri social network come LinkedIn e Twitter, così da dare la possibilità ad utenti non iscritti a Facebook di poter utilizzare il sistema;
- Il miglioramento della veste grafica dell'intero sito, rendendola più comoda alla vista dell'utente;
- La possibilità ad un utente, che abbia già scelto e confermato i posti a sedere, di poter modificare la propria scelta, o di poter effettuare la propria scelta in più tranche nel caso abbia prenotato più di un posto, nel caso peggiore garantendo all'utente la possibilità di effettuare tanti accessi al sistema quanti sono i posti a sedere prenotati (nel caso voglia scegliere ogni singolo posto in una fase a sé stante);
- La realizzazione di un'analisi dei dati, grazie all'ausilio di beta testers, in modo da migliorare gli algoritmi di raccomandazione.