Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria "Enzo Ferrari"

Corso di Laurea in Ingegneria Informatica

PROGETTO E SVILUPPO DI UNA WEB-APPLICATION PER LA NOTIFICA DI MESSAGGI SU PORTALI AZIENDALI

Relatore:
Prof.ssa Sonia Bergamaschi

Tutor Aziendale:
Ing. Andrea Prandini

Candidato:
Marco Giovanetti

Anno Accademico 2013–2014

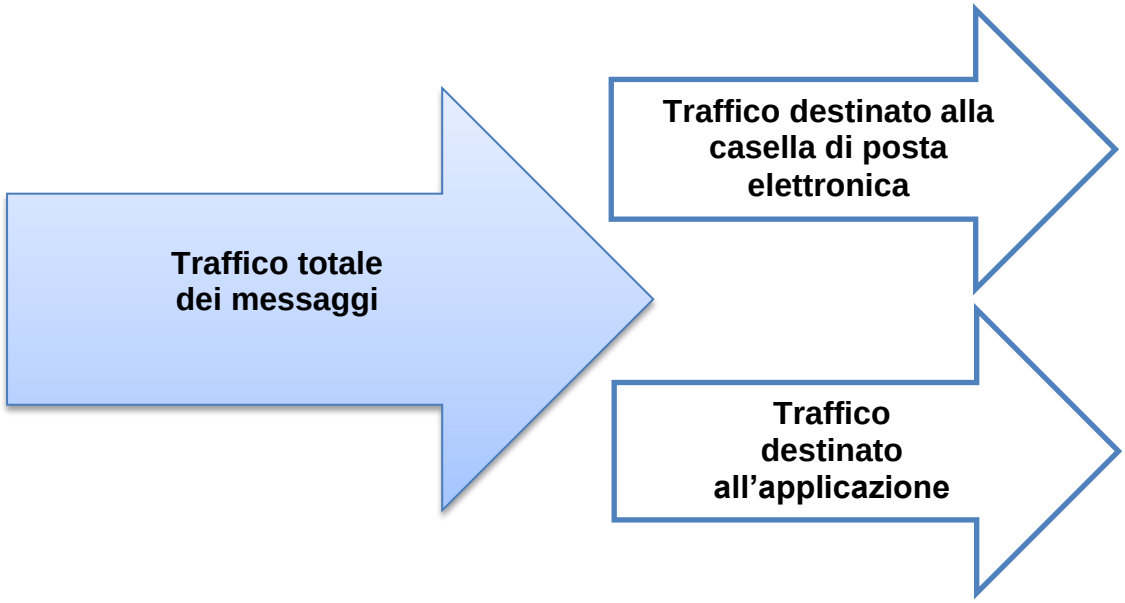
Indice

Introduzione

Il presente elaborato è stato sviluppato in seguito all'esperienza maturata durante l'attività di tirocinio formativo presso l'azienda Quix s.r.l. di Soliera (MO), della durata di tre mesi e svolto da Marzo a Giugno dell'anno corrente.

L'azienda è la sviluppatrice del software "Qorder fashion", ossia una soluzione B2B (Business to Business) nata per gestire l'intero processo d'acquisizione ordini Clienti nel settore della Moda, e dell'applicazione "Fashion Touch", presente nell'Apple Store, che è la versione mobile del software citato in precedenza per la raccolta ordini B2B da iPad.

L'obiettivo prefissato di questa tesi è la progettazione e lo sviluppo di un sistema di notifica aziendale integrato sui portali aziendali gestiti dal team di Quix, basato sull'architettura e le funzionalità tipiche di alcuni famosi social network, che permetta la deviazione di parte del traffico, generato dai vari sistemi ed utenti che risiedono sul portale, su di esso senza dover utilizzare client di posta elettronica intermedi.



Lo scopo di questa applicazione web è quindi quello di offrire agli utenti di uno stesso portale aziendale un'alternativa alla posta elettronica per lo scambio di brevi comunicazioni.

Panoramica dei capitoli

Capitolo 1: "Funzionalità e tecnologie usate"

In questo capitolo sono presentati a grandi linee il progetto e le sue funzionalità, l'ambiente di sviluppo e i linguaggi utilizzati nella creazione della web-application, ognuno di essi accompagnato da una breve descrizione.

Capitolo 2: "L'applicazione al suo interno"

In questo capitolo sono descritte la struttura del database, come la web-application si interfaccia ad esso e l'architettura delle servlet, ovvero le classi Java il cui ruolo è quello di Controller tra il database (il Model) e la parte grafica (la View).

Capitolo 3: "La realizzazione della parte grafica"

In queste pagine viene descritto il progetto della parte grafica dell'applicazione, dapprima presentando brevemente il linguaggio HTML5 e il framework Bootstrap 3.0.1, e poi facendo un focus sulle schermate vere e proprie che si possono incontrare durante l'utilizzo della web-app.

Capitolo 4: "Conclusioni ed eventuali implementazioni future"

In questo capitolo si descrivono i risultati ottenuti e le eventuali funzionalità che potranno essere implementate nel progetto.

CAPITOLO 1 – *Funzionalità e tecnologie usate*

1.1 – Il progetto

La creazione di questo progetto nasce dall'esigenza di ridurre il traffico mail tra dipendenti e, in generale, utenti di una azienda che possiedono un account su uno stesso portale aziendale. Ogni giorno possono essere scambiate anche decine di comunicazioni via mail, con il risultato di ritrovarsi la casella di posta elettronica piena di messaggi che, molto spesso, vengono eliminati senza neppure essere letti ma basandosi solamente sull'oggetto del messaggio. Attraverso l'uso di questa web-application sarebbe possibile deviare parte del traffico su questo sistema di notifica, in modo che un utente possa leggere o meno le comunicazioni che gli interessano, semplicemente evitando di leggere le altre e senza dover ogni giorno dover fare una scrematura delle email da aprire o meno per mantenere ordine nella casella di posta elettronica.

Inoltre, tenendo conto della tendenza degli ultimi anni di navigare in internet sempre più spesso da dispositivi mobili, si è scelto di creare l'intera web-application con un layout in grado di ridisegnarsi ed adattarsi alle dimensioni dello schermo da cui viene visualizzata, in modo che sia sempre ottimizzata per l'interazione e la navigazione da dispositivi mobili.

Il progetto, costruito interamente partendo da zero, è costituito dalla parte grafica, creata utilizzando i linguaggi HTML5 e CSS3, integrati con le classi messe a disposizione dal framework Bootstrap e arricchita tramite funzioni jQuery e JavaScript, e dalla parte interna, costruita seguendo l'architettura MVC (*Model View Controller*), che separa i compiti fra i componenti software che interpretano tre ruoli principali:

- il *controller* riceve i comandi dell'utente e attua le azioni specificate sul model;
- il *model* fornisce i metodi per accedere ai dati utili all'applicazione (Ad esempio metodi per l'accesso al database);
- il *view* visualizza i dati contenuti nel model e si occupa dell'interazione con gli utenti del sistema.

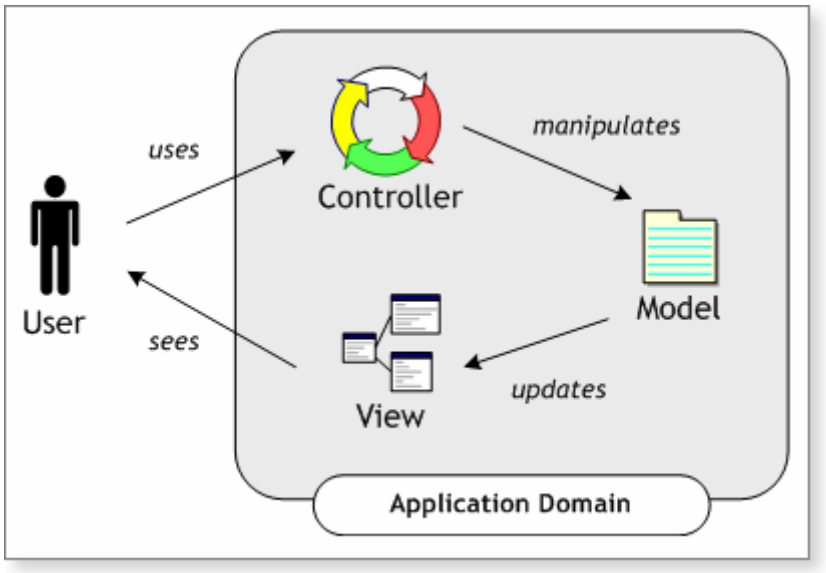


Figura 1: Schema MVC

Lo schema precedente rappresenta l'interazione fra i tre sottosistemi definiti dal pattern MVC; nel momento in cui l'utente interagisce con l'interfaccia grafica (ad esempio facendo click su un bottone), il relativo Controller riceve l'evento e applica una strategia per modificare i dati risiedenti all'interno del Model. Nel caso in cui il Model venga modificato, esso notificherà l'avvenuta modifica al Controller che a sua volta si occuperà di aggiornare la View. Il Controller agisce dunque come mediatore, comunicando alla View le modifiche allo strato del Model in modo che essa possa mostrare visivamente le conseguenze di tali modifiche.

Per quanto riguarda il core dell'applicazione, creato esclusivamente usando il linguaggio Java, esso si occupa dell'interazione tra la parte grafica e la parte dei dati, salvati e caricati per mezzo di un database MySql.

1.2 – L'ambiente di sviluppo: Eclipse

Eclipse è un ambiente di sviluppo integrato (*Integrated Development Environment, IDE*) open source, multilinguaggio e multipiattaforma sviluppato da un consorzio di grandi aziende (*IBM*, *Intel* e *HP* su tutti) chiamato *"Eclipse Foundation".* Eclipse è scritto in linguaggio Java e la sua interfaccia grafica si basa su SWT (*Standard Widget Toolkit*), un

toolkit grafico che sfrutta i widget forniti dal sistema operativo e che conferisce all'interfaccia utente del programma una elevata reattività.

I linguaggi supportati da Eclipse sono molteplici, tra i quali ricordiamo Java, C, C++ e HTML. Questa moltitudine di linguaggi supportati da Eclipse è possibile anche grazie ai vari plug-in, ovvero programmi non autonomi che interagiscono con l'IDE e che permettono di gestire ulteriori linguaggi, tra cui PHP, JavaScript e XML.

La versione di Eclipse utilizzata per lo sviluppo della web-application è *"Eclipse Indigo for Java EE Developers"*, cioè la versione "Enterprise Edition" per lo sviluppo di software per le imprese, compresi i servizi di rete e web.

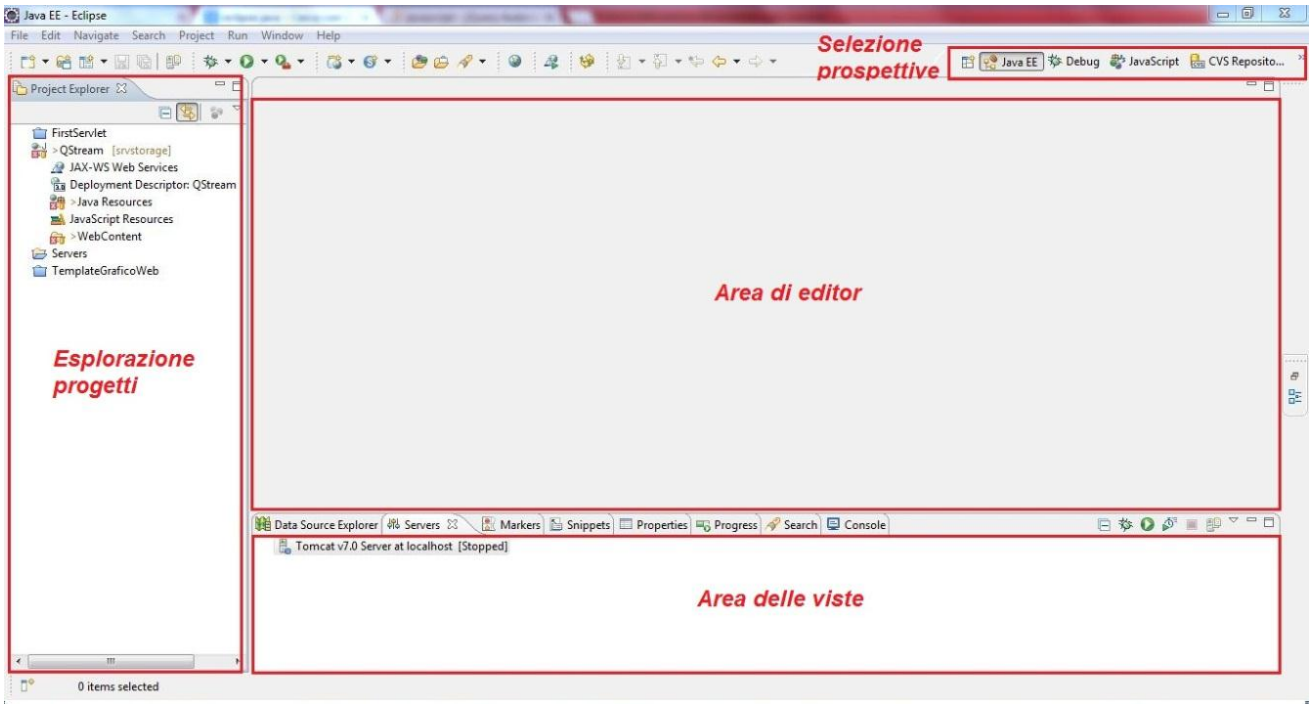


Figura 2: Interfaccia di Eclipse Indigo Java EE

L' Interfaccia di Eclipse EE è abbastanza intuitiva e presenta quattro sezioni principali, fondamentali per l'utilizzo dell'IDE. Sulla sinistra si trova la parte all'interno della quale si possono esplorare i vari progetti che si stanno seguendo. È possibile quindi espandere ognuno di questi per visualizzarne i relativi contenuti oppure, cliccando con il tasto destro del mouse su uno dei progetti scelti, si può far partire l'esecuzione, sia in modalità normale che in modalità "*debug*" per la ricerca di errori nascosti o di logica. In questa sezione sono incluse anche le configurazioni dei vari server che vengono usati per i test

dell'applicazione in fase di sviluppo. Infine al centro la finestra è divisa in due parti: quella superiore permette di creare, visualizzare e modificare codice di programmazione attraverso un editor di testo che implementa l'evidenziazione della sintassi del codice, mentre quella inferiore permette di accedere alle varie viste che possono essere scelte in Eclipse. Alcune di queste, due su tutte la tab "*Console*" e la tab "*Servers*", sono molto importanti in fase di sviluppo dell'applicazione in quanto la prima viene ampiamente utilizzata come strumento ausiliario per il "log" del programma, ovvero una procedura che registra operazioni significative su un file di testo o, come in questo caso, nella console, ogni volta che vengono eseguite e la seconda permette un rapido accesso alle impostazioni e alle configurazioni dei server, oltre che permettere l'avvio o l'arresto di ognuno di essi senza dover accedere alla relativa console o al percorso dell'eseguibile.

1.3 – Linguaggi utilizzati

Durante lo sviluppo del progetto ho avuto modo di studiare ed in seguito utilizzare vari linguaggi di programmazione. Qui sotto si può osservare un elenco, integrato con una breve descrizione, di alcuni dei linguaggi maggiormente usati:

- ***Java:*** Java è un linguaggio di programmazione ad oggetti, sviluppato durante la prima metà degli anni '90 dalla Sun Microsystems e da un team di sviluppatori con a capo James Gosling, progettato specificatamente per essere il più possibile indipendente dalla piattaforma su cui viene eseguito. Ciò è possibile grazie al fatto che, in fase di compilazione, viene generato un nuovo codice, detto "*bytecode*", che è interpretato in fase di esecuzione da una *Java Virtual Machine – JVM*, ovvero un programma che è in grado di interpretare il bytecode.

- ***JavaScript:*** JavaScript è un linguaggio di scripting con una sintassi vicina al linguaggio Java, comunemente usato per la creazione e l'arricchimento di siti web. Fu sviluppato da Brendan Eich, dipendente della Netscape Communications, nel 1995. Anche se il nome potrebbe trarre in inganno, non intercorre una vera e propria relazione tra Java e JavaScript, se non nella sintassi (ereditata da entrambi dal C). Una delle caratteristiche principali di JavaScript è il fatto di essere un linguaggio interpretato, ovvero il codice non è compilato ma interpretato

direttamente da un programma, detto interprete Quest'ultimo risiede tipicamente nel browser che si sta usando. Un'altra caratteristica di JavaScript è che non è un linguaggio stand-alone, ovvero deve essere integrato all'interno di un programma ospite che gli fornisca un'API (*Application Programming Interface*) ben definita, in modo che il codice ospitato sia in grado di accedere ad operazioni specifiche. L'uso principale di JavaScript in ambito di web-application è la scrittura di funzioni integrate nelle pagine HTML che interagiscono con il DOM (*Document Object Model*) del browser per compiere determinate azioni, altrimenti non possibili con il solo HTML statico, come ad esempio controllare i valori nei campi di input.

- ***HTML5:*** HTML (*HyperText Markup Language*) è un linguaggio usato in informatica per la formattazione di testo sotto forma di pagine web. Esso descrive le modalità di impaginazione e di layout del contenuto della pagina in base all'utilizzo di determinati tag di formattazione, che stabiliscono il ruolo del relativo elemento contrassegnato e ne definiscono vari attributi, come ad esempio colore, dimensione, font del testo e la posizione relativa all'interno della pagina. I tag si definiscono racchiudendone il nome all'interno di parentesi angolari (Es. <*div*>Testo da visualizzare nella pagina.<*/div*>). Attualmente la versione più diffusa del linguaggio HTML è la 4.0.1, mentre la versione 5 del noto linguaggio di markup è ancora in fase di sviluppo e standardizzazione e non ancora supportata dalla maggior parte dei browser.

- ***CSS3:*** Il CSS (*Cascade Style Sheet*) è un linguaggio utilizzato per definire la formattazione di fogli HTML, XML e XHTML. L'uso di CSS permette una programmazione più chiara e facile, mantenendo separati i contenuti dalla formattazione, e permettendo il riuso di gran parte del codice. Un altro enorme vantaggio dei CSS è che il loro utilizzo permette di creare pagine molto meno pesanti (in termini di bit di dati), perché le regole CSS possono essere incluse in un file esterno alle pagine HTML che rimane memorizzato nella cache del browser, riducendo ulteriormente la quantità di dati che i server devono trasmettere al client ad ogni richiesta.

Oltre quelli precedentemente descritti, è stato utilizzato anche il linguaggio SQL per la creazione di query per il salvataggio, la ricerca e l'ottenimento dei dati dal database.

1.4 – Framework

Per la realizzazione del progetto si è inoltre scelto di utilizzare due framework, ovvero una serie di librerie di codice usabili dal programmatore per velocizzare e facilitare lo sviluppo del codice. Alla base del framework c'è il concetto di riusabilità del codice, poiché lo scopo è appunto quello di fornire al programmatore una serie di funzioni già scritte ed in questo modo riutilizzabili a piacere.

I framework utilizzati sono elencati di seguito, accompagnati da una breve descrizione riguardante il loro uso e le loro potenzialità:

- ***jQuery 1.10.2:*** jQuery è un framework *cross-browser* in linguaggio JavaScript, il cui scopo è quello di facilitare la programmazione lato client di pagine HTML mascherando le differenze implementative che possono intercorrere tra i vari browser. Non a caso il suo slogan è *"write less, do more",* ovvero "scrivi meno, fai di più", in quanto bastano poche righe di codice per effettuare operazioni che richiederebbero svariate istruzioni utilizzando linguaggi diversi. Con jQuery è possibile manipolare HTML, CSS e DOM, oltre agli elementi interni di jQuery, come liste o code di oggetti. Inoltre questo framework è espandibile tramite alcuni plug-in, ovvero librerie che estendono le sue funzionalità. Tra questi vi sono, ad esempio, *jQueryUI*, una libreria di funzioni JavaScript che fornisce interazioni, animazioni ed effetti avanzati al programmatore, e *jQueryMobile*, utile per la creazione di siti web responsive ed applicazioni accessibili da qualunque dispositivo mobile.

- ***Bootstrap 3.0.1:*** Bootstrap è un framework grafico open source, sviluppato all'interno di Twitter, per la creazione di interfacce grafiche complete e volte alla visualizzazione su dispositivi mobili, come tablet e smartphone. Al suo interno contiene modelli HTML, CSS e alcune funzioni JavaScript che permettono lo sviluppo di siti web *responsive*, ossi capaci di adattarsi e ridisegnarsi a seconda dello schermo del dispositivo dal quale si visita la pagina.

2.1 – L'architettura del database

Alla base dell'applicazione vi è un database MySql, creato utilizzando lo Storage Engine *MyISAM*. Gli Storage Engine sono librerie per il database MySql che svolgono compiti riguardanti la gestione fisica dei dati, quali scrittura e lettura dei record e indicizzazione dei dati. MyISAM, engine standard fino alla versione 5.5 di MySql, è un motore di immagazzinamento molto veloce che non richiede molte risorse, sia RAM che su disco. Uno svantaggio di questo Storage Engine è che non supporta le transazioni, ovvero insieme di operazioni che rende i dati permanenti se eseguite con successo, altrimenti li riporta allo stato precedente l'inizio della transazione nel caso di un errore qualsiasi.
Il database è composto da otto tabelle, delle quali le due principali sono quella degli utenti (account) e quella dei messaggi (tweet). Qui sotto si può vedere come è stato strutturato lo schema del database, con le relative relazioni e legami tra le varie tabelle.

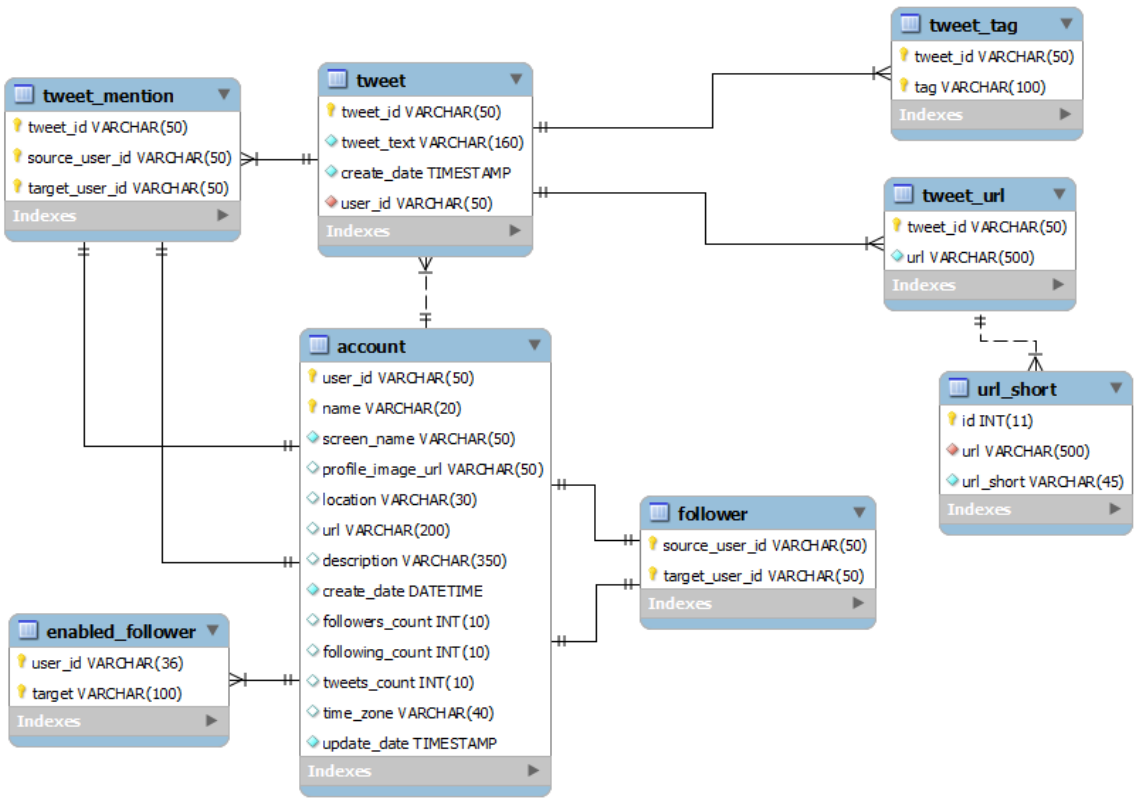


Figura 3: Diagramma EER delle tabelle del database

Le otto tabelle che costituiscono il database non sono però le uniche che agiscono all'interno del sistema, poiché esso si avvale anche di uno schema esterno messo a punto dall'azienda Quix e chiamato "*PUMa – Professional User Manager*", utilizzato nei portali aziendali da loro gestiti per rappresentare l'utente del sistema e che, nel mio caso, ho usato per estendere con maggiori proprietà l'entità account, permettendomi di assegnare ad ogni utente del sistema un gruppo di appartenenza (ad esempio il gruppo clienti sarà indicato come *"GROUP_CUSTOMERS"*) ed un ruolo, che può essere impostato dall'amministratore di sistema per ogni utente e può essere "avanzato" o "standard ("*stream_adv"* e "*stream_std"* rispettivamente). All'interno dell'entità *t_user* sono anche memorizzate le credenziali di accesso al sistema, gestito dal protocollo *"CAS – Central Authentication Service"*, cioè un protocollo di *"Single Sign-On" (SSO).* Esso è una struttura di controllo di accesso di sistemi di software indipendenti ma legati tra loro, tramite il quale un utente può accedere una volta all'applicazione e ottenere l'accesso a tutti i sistemi interni senza che venga richiesto un nuovo login per ciascuno di essi. Anche questo servizio viene utilizzato dall'azienda come protocollo di accesso ai portali aziendali da essa gestiti. Di seguito si può vedere un'immagine che mostra la relazione tra le tabelle di "PUMa" utilizzate nella web-application.

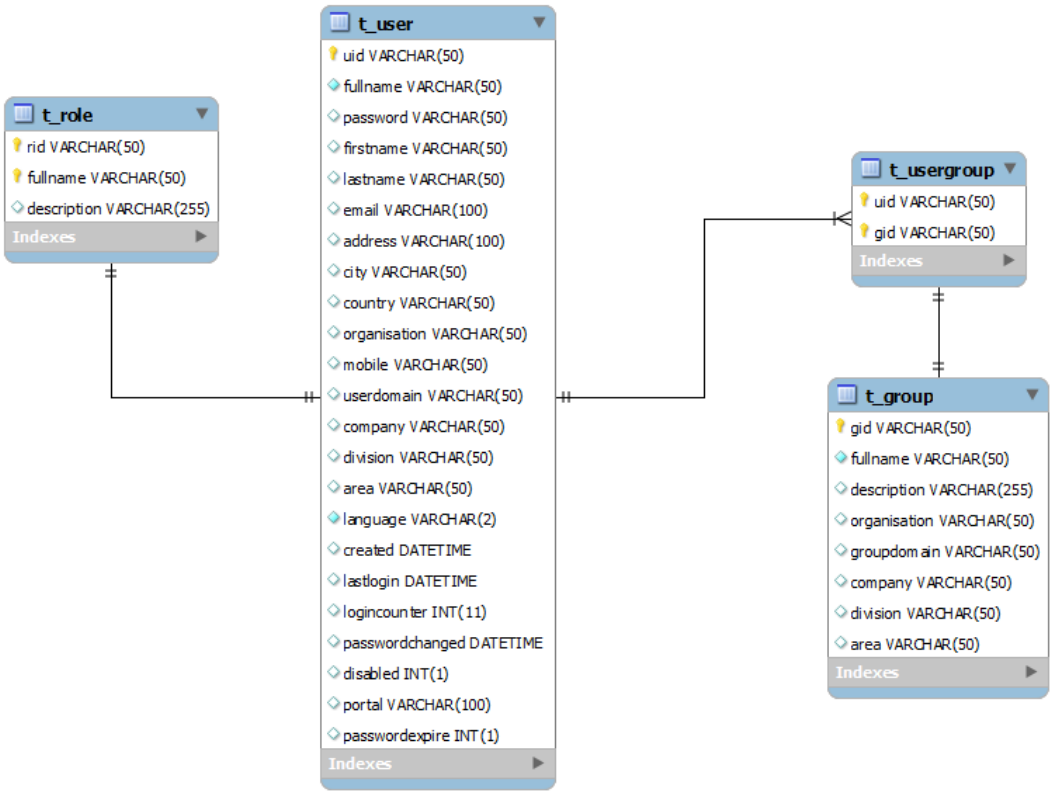


Figura 4: Diagramma EER delle tabelle di *PUMa* utilizzate dalla web-application

Proseguendo nella descrizione del database, incontriamo le due tabelle principali dello schema che si trova alla base della web-application:

- **Tabella *Account:*** è la tabella principale del database, poiché mantiene i dati dell'attore principale del sistema, ovvero l'utente. Questa tabella è estesa dalla tabella *t_user* di "Puma", alla quale si lega per mezzo del *uid*, coincidente con il *name* dell'account. Al suo interno troviamo due primary key: lo *user_id* e il *name*, che sono univoci nell'intero sistema. Continuando a scorrere la tabella troviamo lo *screen_name*, ovvero il nome modificabile dall'utente e visibile agli altri, la *profile_image_url*, cioè la colonna della tabella in cui è salvato il nome dell'immagine del profilo, relativamente ad un url prefissato nella web-application, la colonna *url*, dove possiamo reperire l'url univoco per ogni utente, i campi *followers_count, following_count e tweets_count*, adibiti al mantenimento delle informazioni relative al numero di utenti che seguono, sono seguiti e il numero dei tweet rispettivamente dell'utente considerato. Oltre a queste colonne principali troviamo quelle di *create_date* e *update_date*, in cui sono salvate le date di creazione e di ultimo aggiornamento dell'utente, quella di *description*, che permette il salvataggio di una piccola descrizione modificabile dall'utente e quella di *time_zone*, che mantiene il record riguardo il fuso orario dell'account. La tabella di Account si interfaccia con le seguenti tabelle: *tweet*, per la creazione del messaggio online, la tabella di *follower*, che mantiene la relazione di following tra due utenti identificati tramite il loro user id, la tabella di *tweet_mention*, che memorizza la relazione tra due utenti nel caso di *"mention",* ovvero la menzione di un utente, tramite il suo nome utente, in un messaggio, e, nel caso l'account possegga i privilegi di utente avanzato, con la tabella di *enabled_follower*, che mantiene la relazione di un utente avanzato con i gruppi di utenti che possono eventualmente aggiungerlo alla loro lista di follower.

- **Tabella *Tweet:*** è la tabella designata per il mantenimento delle informazioni dei messaggi che sono stati postati online dai vari account del sistema. La tabella è identificata tramite una primary key che identifica in modo univoco il messaggio all'interno del sistema, il *tweet_id*. Scorrendo la tabella troviamo altre colonne, tra le quali quella di *tweet_text*, al cui interno è salvato il testo del messaggio, quella di *user_id*, contenente lo user_id dell'utente che ha creato il messaggio, e la colonna

create_date, che mantiene la data di creazione del messaggio. La tabella tweet è legata alla tabella *tweet_mention*, per il mantenimento della relazione di una "*mention*" tra due utenti con l'id del tweet in cui essa è presente, alla tabella *tweet_tag*, che memorizza l'id del tweet in cui compare un "*hashtag*", ovvero una parola chiave che può essere utilizzata da altri utenti per ricercare messaggi inerenti ai loro interessi, e l'hashtag stesso e legata alla tabella *tweet_url*, che si occupa di mantenere la relazione tra un tweet, sempre tramite il suo id, ed un url in esso presente.

Oltre a queste due, nello schema ci sono altre tabelle secondarie, molte delle quali sono già state descritte nella descrizione delle relazioni con le tabelle principali. Tra le tabelle non ancora citate troviamo quella di *url_short*, che è adibita al mantenimento della relazione che intercorre tra un url inserito all'interno di un messaggio e il suo corrispondente "*short url*", ovvero una trasformazione dell'url originale in una forma abbreviata che permetterà di rendere illeggibile l'url condiviso fuori dalla applicazione, oltre che occupare meno spazio sul database al momento del salvataggio.

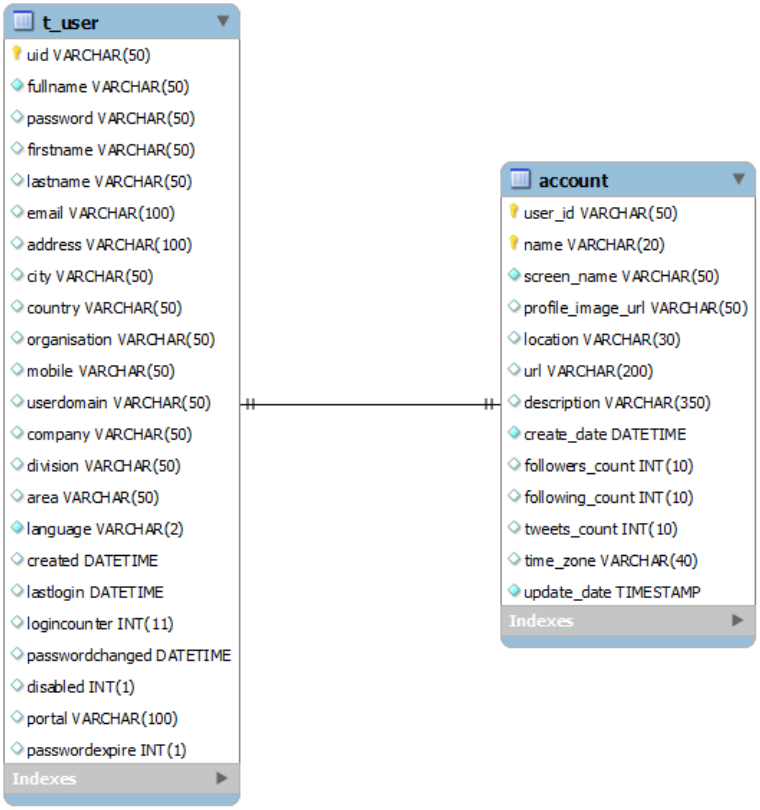


Figura 5: Diagramma EER della relazione tra t_user di PUMA e l'entità Account

Nella figura precedente si può osservare che l'entità t_user di "PUMa" è legata all'entità Account del database alla base della web-application tramite una relazione uno ad uno tra il *fullname* della prima tabella e il *name* della seconda. Grazie a questo legame, l'entità che si trova alla base del sistema progettato viene estesa e arricchita dagli attributi presenti in "PUMa", grazie ai quali è stato possibile implementare diverse logiche di visualizzazione delle pagine in base alla privacy ed al ruolo assegnato all'account.

2.2 – L'interfaccia tra database e Java

Per l'interazione tra il database e la web-application è stato usato il driver JDBC (Java DataBase Connectivity) che consente l'accesso ai dati da qualsiasi programma scritto in linguaggio Java, indipendentemente dal tipo di DBMS utilizzato. JDBC è l'insieme delle API che definiscono le specifiche per l'accesso ai database, le cui principali interfacce sono: Driver, Connection, Statement, PreparedStatement e ResultSet. L'oggetto di tipo Connection consente di ottenere una connessione con un database. La classe DriverManager, appartenente alla interfaccia Driver, è incaricata di scegliere l'opportuno driver registrato e, attraverso l'URL e le credenziali di accesso al database passati come parametri, istanziare l'oggetto Connection. Per ottenere la connessione ad un database MySql usando il driver JDBC, la stringa di codice da utilizzare è:

```
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost/it_quix",
"userName", "userPassword");
```

L'oggetto Statement, invece, permette di effettuare query in linguaggio SQL, attraverso la connessione precedentemente creata. Qualora lo stesso Statement SQL sia utilizzato più volte, è consigliato servirsi, in alternativa, dell'oggetto PreparedStatement che aumenta l'efficienza dell'esecuzione della query e l'allocazione delle risorse. Inoltre l'uso di PreparedStatement previene dal fenomeno chiamato "*SQL Injection*", ovvero l'immissione di codice maligno all'interno di una query SQL. Per esempio, se non avessimo alcuna validazione dei campi di login, un utente malintenzionato potrebbe inserire nel campo username la stringa *" **' ; shutdown –** "*, con la conseguenza di disconnettere il database rendendo inaccessibile l'applicazione. Grazie al PreparedStatement ciò non è possibile, in quanto le stringhe SQL vengono passate già elaborate ad esso. Infine, il ResultSet, è

l'oggetto che raccoglie i dati richiesti all'interno dello Statement e li inserisce in una struttura dati opportuna. Il metodo getType("NomeColonna"), dove Type indica il tipo di dato che si vuole estrarre dal ResultSet e che deve essere concorde al tipo di dato di "NomeColonna", restituisce il dato contenuto all'interno della colonna specificata. Nel caso il dato fosse mancante non viene lanciata alcuna eccezione e l'esecuzione continua. Per poter utilizzare il driver, si dovrà quindi aprire una connessione con il database tramite l'interfaccia Connection, creare un PreparedStatement per interrogare la base dati e utilizzare eventualmente il ResultSet restituito per gestire i risultati ottenuti dalla query SQL.

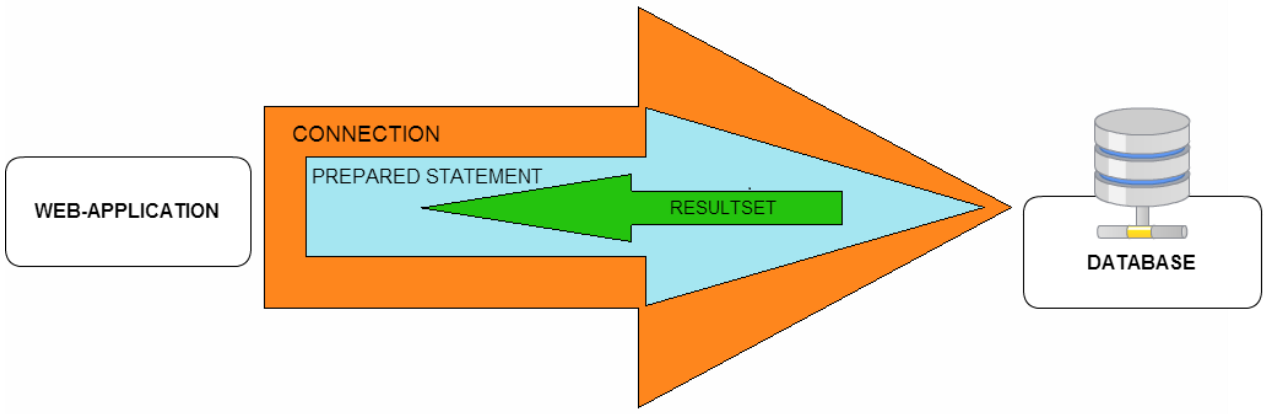


Figura 6: Schema del funzionamento della connessione tra web-application e database

Sempre seguendo l'architettura Model-View-Controller, l'intera interfaccia tra database e applicazione è gestita da un unico package, chiamato *DAO*, che è fondamentalmente una rappresentazione astratta delle tabelle del database. I *DAO – Data Access Objects* sono classi che forniscono metodi per l'accesso al database e permettono una netta separazione tra le componenti di un'applicazione, come in questo caso tra Model e Controller. La classe astratta "*AbstractDAO*", estesa da tutte le classi del package DAO, contiene al suo interno vari metodi utili per creare connessioni e chiudere i vari oggetti coinvolti nella comunicazione tra database e applicazione.

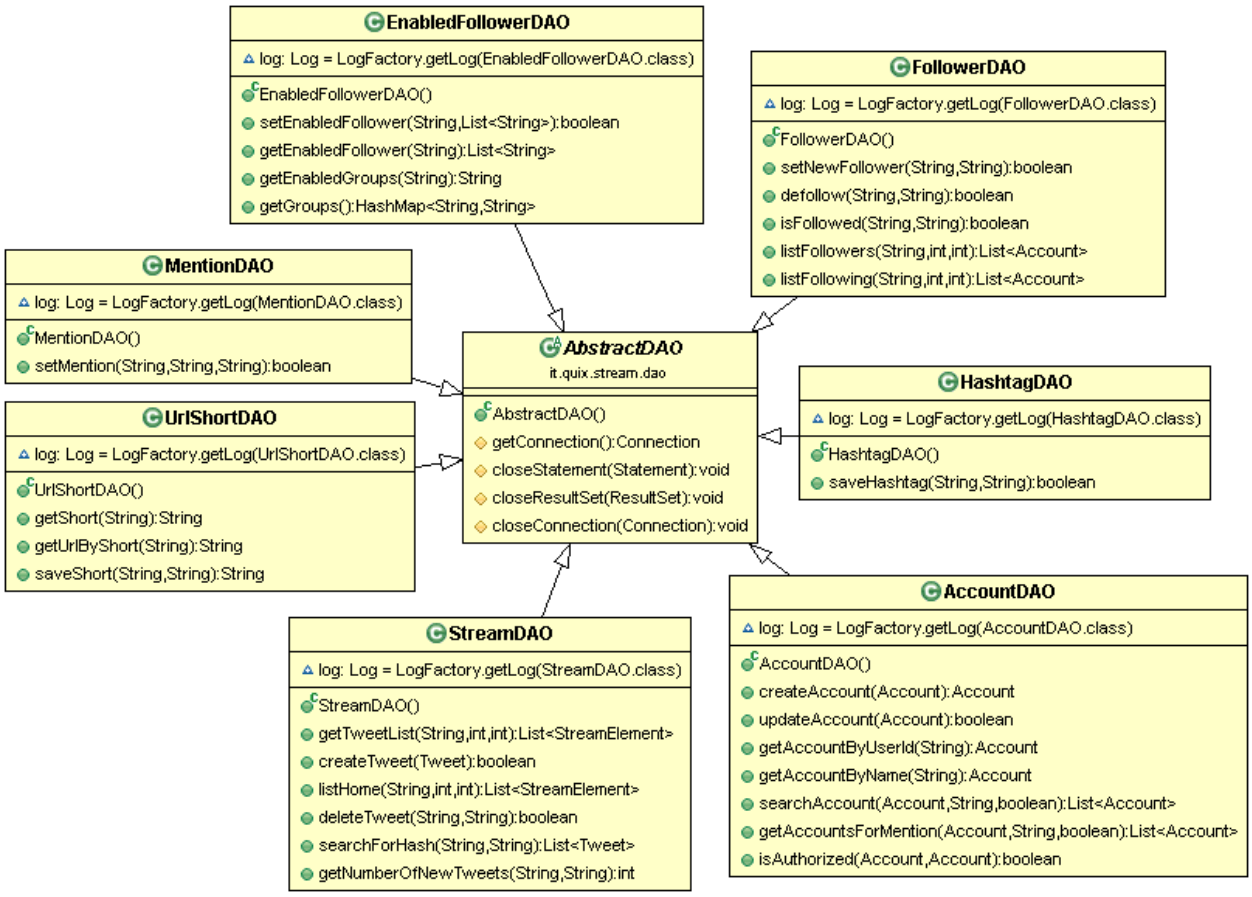


Figura 7: Class Diagram del package it.quix.DAO

2.3 – L'architettura delle servlet

In accordo con il modello MVC, che prevede la separazione dei compiti tra i vari componenti software, si è scelto di utilizzare le *servlet* come controller delle varie parti del sistema. Le servlet sono oggetti in linguaggio Java che operano all'interno di un web server e che sono usate per la creazione di web-application e pagine web dinamiche. In questo caso specifico, le varie servlet create nell'applicazione hanno il compito di gestire eventuali richieste da parte del client, effettuare elaborazioni dei dati ricevuti (ad esempio la ricerca di "mention" o url all'interno di un messaggio inviato da un utente o la creazione di una short-url a partire da una url) e di fornire una risposta al client tramite protocollo HTTP.

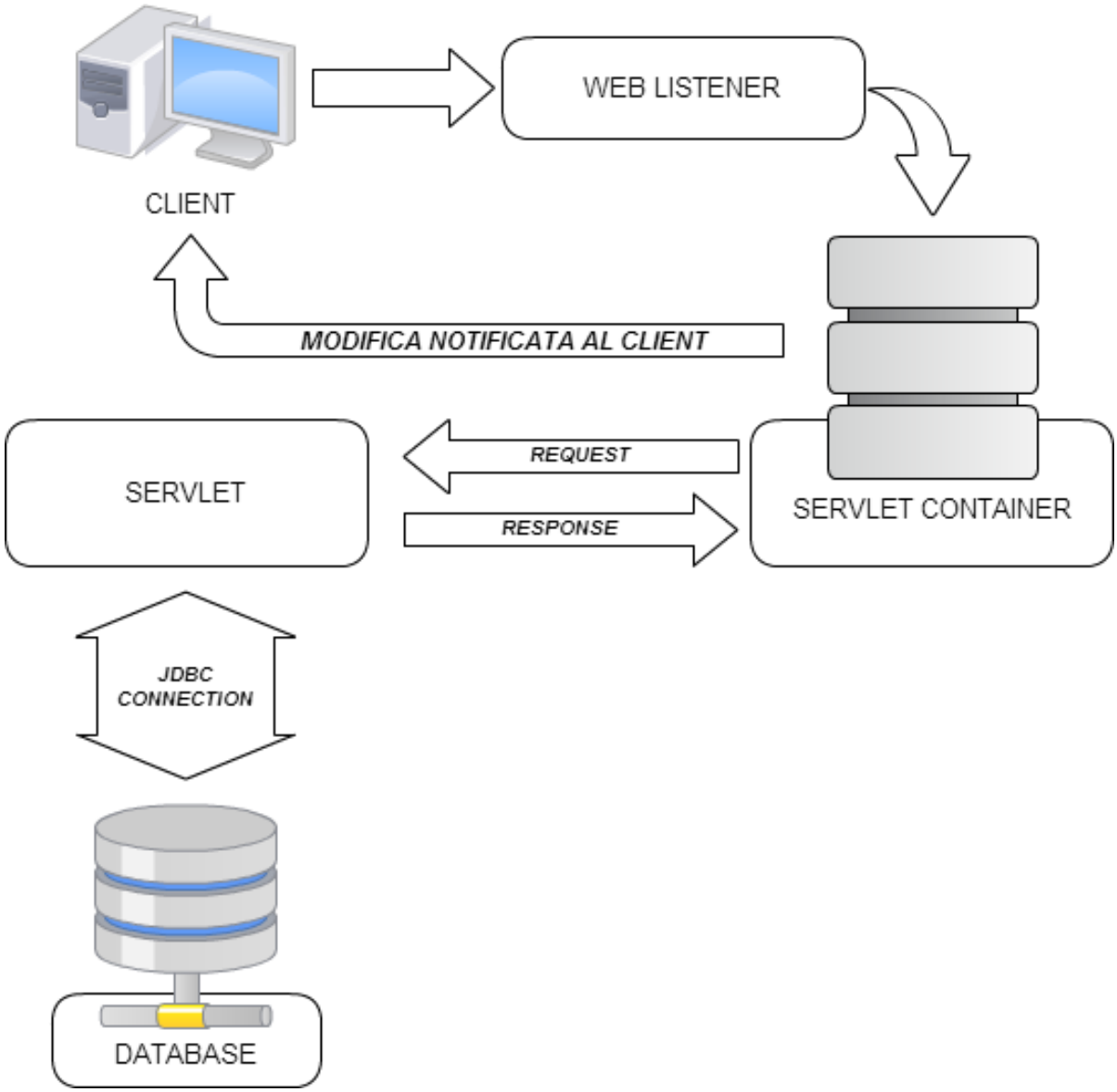


Figura 8: Schema dell'interazione Client - Servlet - Database

La struttura base di una servlet è di seguito rappresentata:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class ExampleServlet
 */
@WebServlet(url="/ExampleServlet")
```

```
public class ExampleServlet extends HttpServlet {

    public ExampleServlet() {
        super();
        // Auto-generated constructor stub
    }

      protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
            // Auto-generated method stub
      }

      protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
            // Auto-generated method stub
      }

}
```

I package `java.io.IOException` e `javax.servlet.ServletException` mettono a disposizione del programmatore vari metodi per l'intercettazione degli errori durante l'esecuzione del programma. Il package `javax.servlet.annotation.WebServlet` permette invece l'uso dell'annotazione `@WebServlet(url="/ExampleServlet")` per definire un componente servlet all'interno dell'applicazione. Questa tecnica è stata implementata in sostituzione del mapping della servlet all'interno del file web.xml. Il parametro url all'interno dell'annotazione è obbligatorio, in quanto specifica a che indirizzo potrà essere acceduta la servlet, mentre gli altri parametri sono opzionali. Continuando nella descrizione della struttura base troviamo gli ultimi tre package, ovvero `javax.servlet.http.HttpServlet`, `javax.servlet.http.HttpServletRequest` e `javax.servlet.http.HttpServletResponse`, che forniscono le classi e le interfacce per scrivere una servlet che gestisca il protocollo HTTP e per gestire, in ordine, la richiesta del client (che contiene ad esempio i dati inseriti dall'utente per essere elaborati) e la risposta del server (che può contenere dati da mostrare, come ad esempio risultati di elaborazione). All'interno della classe, che estende l'interfaccia HttpServlet, sono implementati due metodi: *doGet()* e *doPost()*. Essi richiedono come parametri gli oggetti HttpServletRequest e HttpServletResponse, ossia la richiesta inviata dal client e la risposta del server, e permettono di gestire i metodi GET e POST del protocollo HTTP. Questi due metodi possono essere incanalati in un unico metodo chiamato *service().* Questa funzione, che richiede come parametri gli oggetti HttpServletRequest e HttpServletResponse come i metodi precedentemente visti, identifica il tipo di richiesta del protocollo HTTP (GET, POST, ecc) e invoca il corrispondente metodo Servlet *doGet(), doPost(), ecc*.

All'interno della web-application sono state create svariate servlet per la gestione decentralizzata delle possibili richieste del Client. Si va dalla servlet che gestisce il login a quella incaricata di creare la homepage personalizzata a seconda di quale utente abbia effettuato l'accesso, componendo una lista dei messaggi visibili dall'account, fino ad arrivare alle servlet che gestiscono la visualizzazione di un profilo ricercato e a quella che gestisce il logout.

Ogni servlet estende una classe astratta, chiamata "*AbstractSecurityServlet*", che si occupa di controllare che l'utente sia connesso al sistema. Se questo è verificato, essa fornisce due metodi (*getAccount()* e *setAccount())* alle classi dalle quali è estesa che, se invocati, permettono di ritornare o di settare in sessione un'entità Account contente le informazioni riguardati l'utente che è correntemente connesso al sistema.

Figura 9: La classe astratta AbstractSecurityServlet

Invece, nel caso che l'utente non sia in sessione (perché non ancora loggato o perché è scaduta la sessione stessa), interviene un filtro che ridirige alla pagina di login. I filtri sono classi il cui compito è quello di filtrare e compiere azioni sui contenuti che risiedono all'interno delle "request" e "response" provenienti dalle servlet. Per questo motivo un filtro agisce intercettando una richiesta subito prima che raggiunga la servlet oppure subito dopo che quest'ultima ha creato una risposta. I filtri si distinguono dalle servlet per due motivi principali:

- Non creano realmente una risposta, al contrario delle servlet;
- Non implementano i metodi caratteristici delle servlet (come doGet() e doPost()), ma quelli tipici de i filtri (doFilter() su tutti).

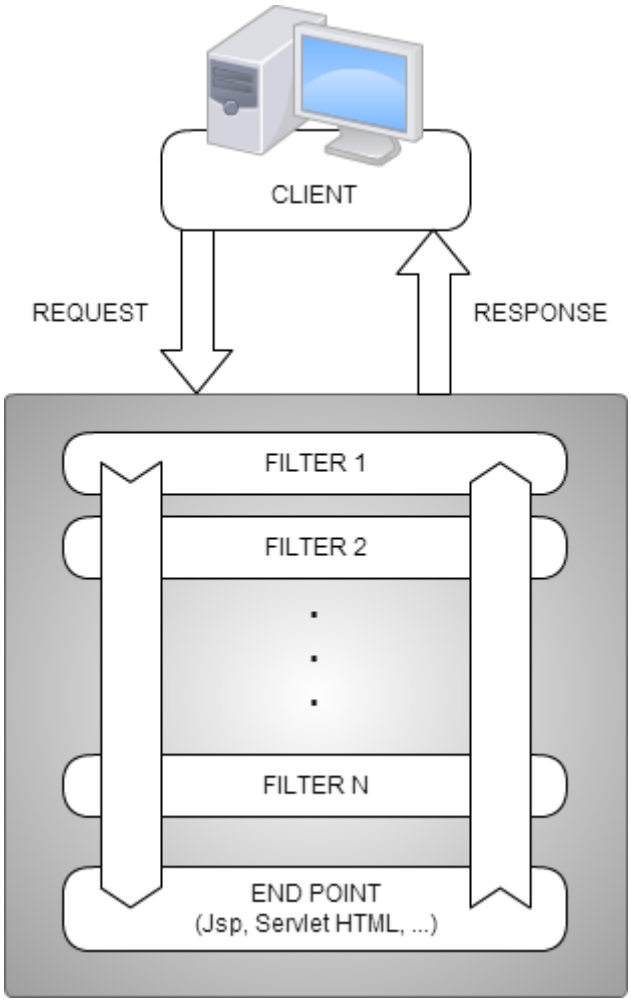


Figura 10: Schema dell'interazione tra filtri e servlet

La struttura base di un filtro è mostrata di seguito:

```
import java.io.IOException;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class Filter implements javax.servlet.Filter {

	public void destroy() {

	}

	public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {

		// Inoltra la richiesta alla catena di filtri
		chain.doFilter(request, response);
```

```
    }
    public void init(FilterConfig fConfig) throws ServletException {
    }
}
```

All'interno del metodo *doFilter* si deve inserire il codice che il filtro deve eseguire quando viene richiamato. Gli altri due metodi presenti, ereditati dall'interfaccia `javax.servlet.Filter`, ovvero *init()* e *destroy()*, vengono richiamati rispettivamente alla creazione e alla distruzione dell'istanza del filtro e il codice al loro interno viene eseguito subito prima e subito dopo il metodo *doFilter()*.

Le uniche servlet che non estendono la classe astratta *AbstractSecurityServlet*, ma ugualmente soggette a filtraggio da parte del filtro precedentemente descritto, sono quelle di login e logout, poiché non necessitano di conoscere lo stato di accesso dell'utente in quanto sono loro che regolano l'accesso o l'uscita dell'utente dal sistema.

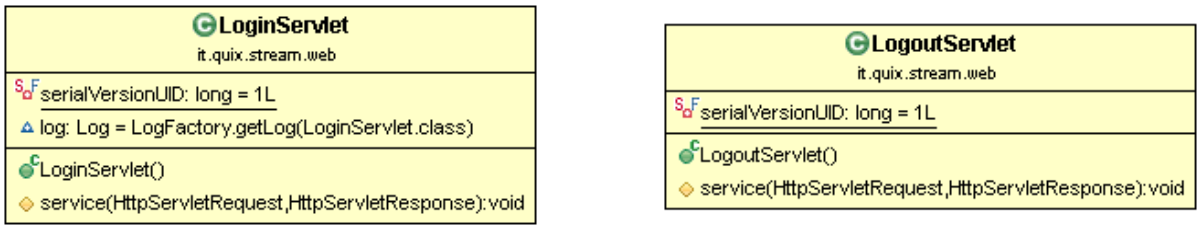


Figura 11: Servlet di Login e Logout con relativi metodi

CAPITOLO 3 – *La realizzazione della parte grafica*

3.1 – Alla base dell'interfaccia grafica

L'intera grafica dell'applicazione è stata costruita seguendo le regole del *Responsive Web Design,* ossia il sistema è stato progettato in modo da ridisegnarsi a seconda del tipo di schermo dal quale viene visualizzato. Ciò è stato possibile soprattutto grazie al framework *Bootstrap 3.0,* che semplifica notevolmente questa tecnica di progettazione. Al suo interno infatti contiene classi CSS e funzioni JavaScript da utilizzare per creare pagine che si adattano automaticamente alle variazioni delle dimensioni dello schermo. Alla base del layout responsive implementato da Bootstrap c'è il concetto di griglia, ovvero un insieme di righe e colonne che permettono la massima fluidità nel ridisegnamento della finestra. Essa è divisa in 12 colonne, la cui larghezza è espressa in percentuale: 12 colonne occupano il 100% dello spazio messo a disposizione, 6 colonne il 50% e così via. Per la costruzione della pagina seguendo questo standard si utilizzano principalmente due classi di Bootstrap: la classe "*row*", che permette di definire una nuova riga, e la classe *"col-*-*",* dove al posto degli asterischi vanno sostituite rispettivamente la minima dimensione dello schermo a partire dal quale la colonna assumerà lo spazio assegnato e lo spazio stesso che deve occupare all'interno della riga. Nel primo caso le dimensioni dello schermo assumibili sono quattro: "*xs",* per schermi fino a 767px*, "sm",* per schermi da 768px a 991px*, "md",* per schermi da 992px a 1199px *e "lg"* per schermi con larghezza maggiore di 1200px, mentre al posto del secondo asterisco i valori ammessi vanno da 1 a 12 (Ad esempio, una colonna dichiarata tramite la classe *col-md-6* occuperà metà dello spazio a disposizione in uno schermo di larghezza compresa tra 992px e 1199px). Sempre in accordo con il concetto di fluidità della griglia, se non diversamente dichiarato, essa tenderà ad occupare tutta la larghezza della viewport. Per evitare questo si può racchiudere la dichiarazione della griglia all'interno di un div caratterizzato dalla classe *"container".* Qui sotto è mostrato il corrispondente codice per creare due contenitori affiancati che occupano metà dello spazio a disposizione ciascuno.

```
<div class="container">
 <div class="row">
  <div class="col-md-6">
   Primo Div
  </div>
  <div class="col-md-6">
   Secondo Div
  </div>
 </div>
</div>
```

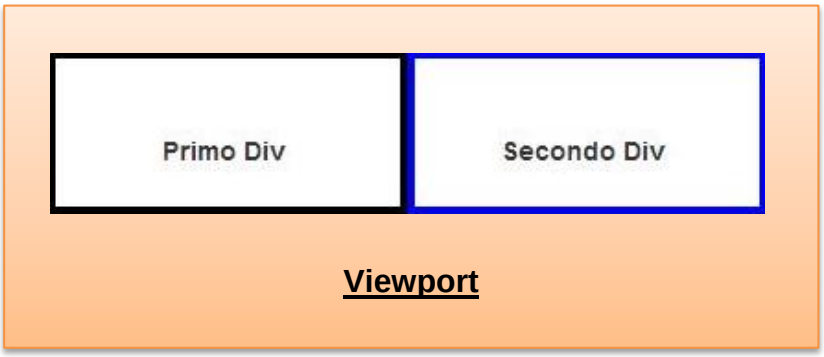


Figura 12: Parte di codice e risultato finale

3.2 – Views

L'applicazione è composta di sette schermate principali, tramite le quali l'utente può interagirvi. Di seguito vediamo una breve descrizione per ognuna di queste view:

3.2.1 – View della pagina di Login

La vista di Login è la prima pagina chiamata all'apertura della web-application. Nel caso che l'utente sia già autenticato all'interno del portale dove risiede l'applicazione, questa view non verrà visualizzata poiché il sistema riconosce che l'account ha precedentemente effettuato il login tramite il CAS e quindi non è necessario farlo nuovamente.

Figura 13: Vista di Login

Questa vista è abbastanza intuitiva e permette all'utente l'inserimento delle proprie credenziali. Oltre ai campi per l'inserimento di username e password, sono presenti il link da cliccare nel caso di password dimenticata, che permette l'accesso ad una procedura tramite la quale è possibile ritrovare o ricreare la chiave di accesso perduta, ed una checkbox che permette il salvataggio di un cookie sul pc che manterrà i dati d'accesso per 90 giorni, prima di invalidarsi.

Figura 14: Procedura di recupero password

3.2.2 – View della homepage

In caso di login effettuato, l'utente sarà automaticamente reindirizzato alla Home Page a lui riservata. Questa pagina è creata dinamicamente ogni volta che viene richiamata, tramite url o ricarica effettuata tramite il browser, ed è divisa in tre parti principali: sulla sinistra sono presenti i riquadri che mostrano il profilo utente riassunto nelle sue parti principali (immagine profilo, nome visualizzato, nome utente, locazione e descrizione) e il riquadro per la creazione di un nuovo messaggio da postare online. Sulla destra è invece presente il pannello che contiene il flusso dei tweet che l'utente può visualizzare. Questo stream è composto dai messaggi scritti dall'utente stesso, quelli provenienti dagli utenti che segue e quelli che lo menzionano.

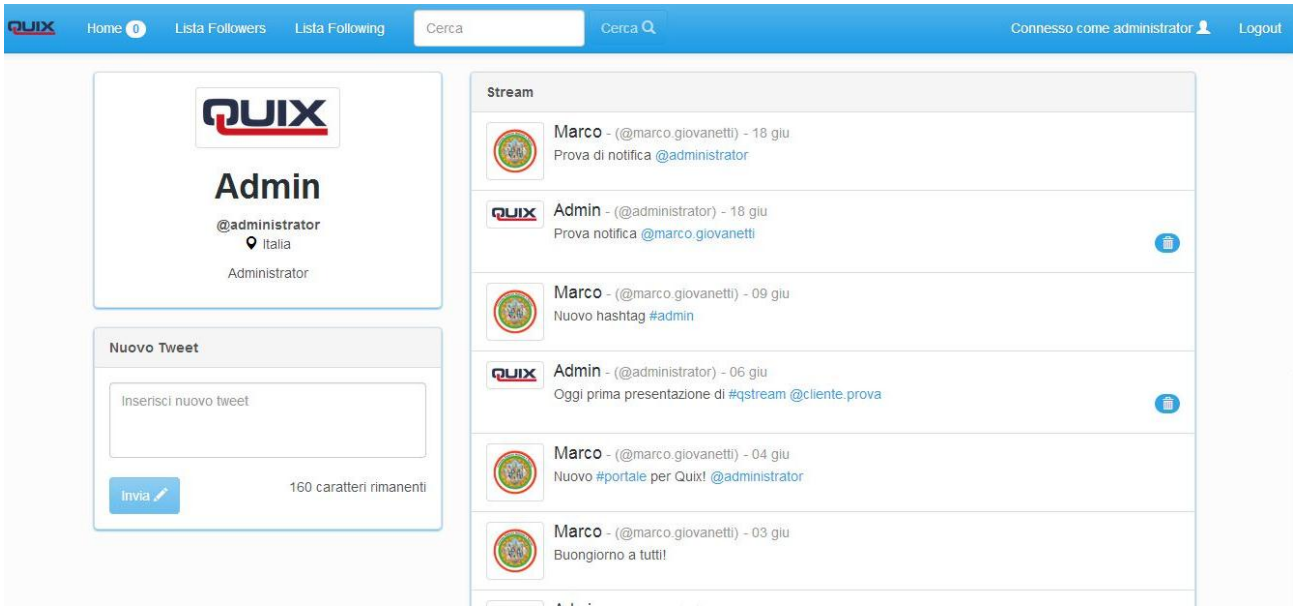


Figura 15: Vista di homepage

In alto, presente in qualsiasi pagina ad esclusione di quelle di login e di errore, è stata inserita una barra di navigazione, utile per riuscire a spostarsi in modo veloce tra le varie pagine della web-application. Essa infatti permette di ritornare alla homepage tramite il tasto “Home”, visualizzare la lista dei propri follower o degli account che l’utente connesso segue, effettuare la ricerca di un utente o di un hashtag, visualizzare il proprio profilo cliccando sul proprio nome utente sulla destra e di effettuare il logout dal sistema. A fianco della scritta Home nel bottone della navbar, ad ogni login verrà indicato il numero dei messaggi postati dall’ultimo login effettuato, in modo da dare un’indicazione all’utente del numero dei messaggi da visualizzare. All’interno della pagina corrente vi sono vari elementi cliccabili, oltre a quelli presenti nella barra di navigazione e già descritti.

Considerando il riquadro di riassunto dell’utente correntemente connesso, è possibile accedere al proprio profilo cliccando sul nome dell’utente. È inoltre possibile visualizzare un ingrandimento dell’immagine del profilo facendo click sulla corrispondente immagine.

Passando al riquadro di nuovo tweet, risulta che, al caricamento della pagina, il tasto di invio del messaggio è disabilitato. Questo è stato fatto per evitare che vengano inviati messaggi senza alcun testo al loro interno. Per scrivere un tweet è sufficiente iniziare a digitare nell’apposita area di testo e il tasto si abiliterà, permettendo all’utente di postare il

messaggio appena creato. È possibile creare post lunghi al massimo 160 caratteri. Nel caso questa misura non venisse rispettata, il tasto si disabiliterà in automatico ed indicherà all'utente il numero di caratteri da cancellare per soddisfare il vincolo di lunghezza. Nel conteggio dei caratteri non sono considerate le reali dimensioni delle url che eventualmente vengono inserite, poiché, grazie al metodo di *shortening* implementato nel sistema, esse vengono traformate in *short url* lunghe nove caratteri. In caso di inserimento di una url in un tweet il conteggio dei caratteri mancanti si bloccherà, riprendendo non appena verrà digitato uno spazio.

Figura 16: Funzionalità del pannello di creazione del messaggio: conteggio caratteri e suggerimento del nome utente nel caso di una mention

Per quanto riguarda il pannello contenente il flusso di tweet sono presenti vari elementi con i quali è possibile interagire. Cliccando sull'immagine del profilo o sul nome dell'autore del tweet scelto, si verrà reindirizzati alla pagina dell'account selezionato, che verrà descritta più avanti. In corrispondenza dei messaggi creati dallo stesso utente che si trova sulla home page, apparirà un link in basso a destra che permette l'eliminazione del messaggio stesso dal sistema. In corrispondenza della riga in cui compaiono le informazioni relative all'autore del post, è possibile visualizzare la data e l'ora precise in cui il tweet è stato creato fermando il puntatore in corrispondenza della data abbreviata. Oltre a questi componenti, che si trovano sempre in corrispondenza di ogni messaggio, è possibile trovare anche altre parti all'interno del testo del tweet. Infatti se compaiono menzioni ad altri utenti, url o hashtag all'interno del post, esse vengono trasformate in link che puntano ad ognuno dei diversi elementi (per esempio, se nel testo di un post appare una menzione all'utente administrator, indicata dall' espressione @administrator, essa verrà sostituita da un link che punterà alla pagina dell'account administrator). Il caricamento del flusso avviene in maniera dinamica tramite l'uso di uno "*scroll infinito*",

ovvero una tecnica utilizzata in moltissimi social network, che permette di caricare i contenuti poco per volta e solo all'avvenimento di uno scroll fino alla fine della pagina. In questo modo si evita di caricare quantità potenzialmente grandi di dati senza che interessino veramente all'utente. Per instanziare questa funzionalità è stata creata una funzione con jQuery che, tramite una chiamata "AJAX" (*Asynchronous JavaScript and XML*), preleva i dati dal database per mezzo di una servlet e aggiorna dinamicamente la pagina con i messaggi richiesti. AJAX è una tecnica usata molto spesso nella creazione di applicazioni web client-side, poiché grazie ad essa è possibile modificare parti della pagina senza dover ricaricarla completamente, evitando così un consumo eccessivo di risorse. L'intera interfaccia grafica è stata creata sfruttando la tecnica *responsive*, ossia è in grado di adattarsi in base alle dimensioni dello schermo. Di seguito si può osservare la schermata di home page come viene visualizzata accendendovi da un tablet:

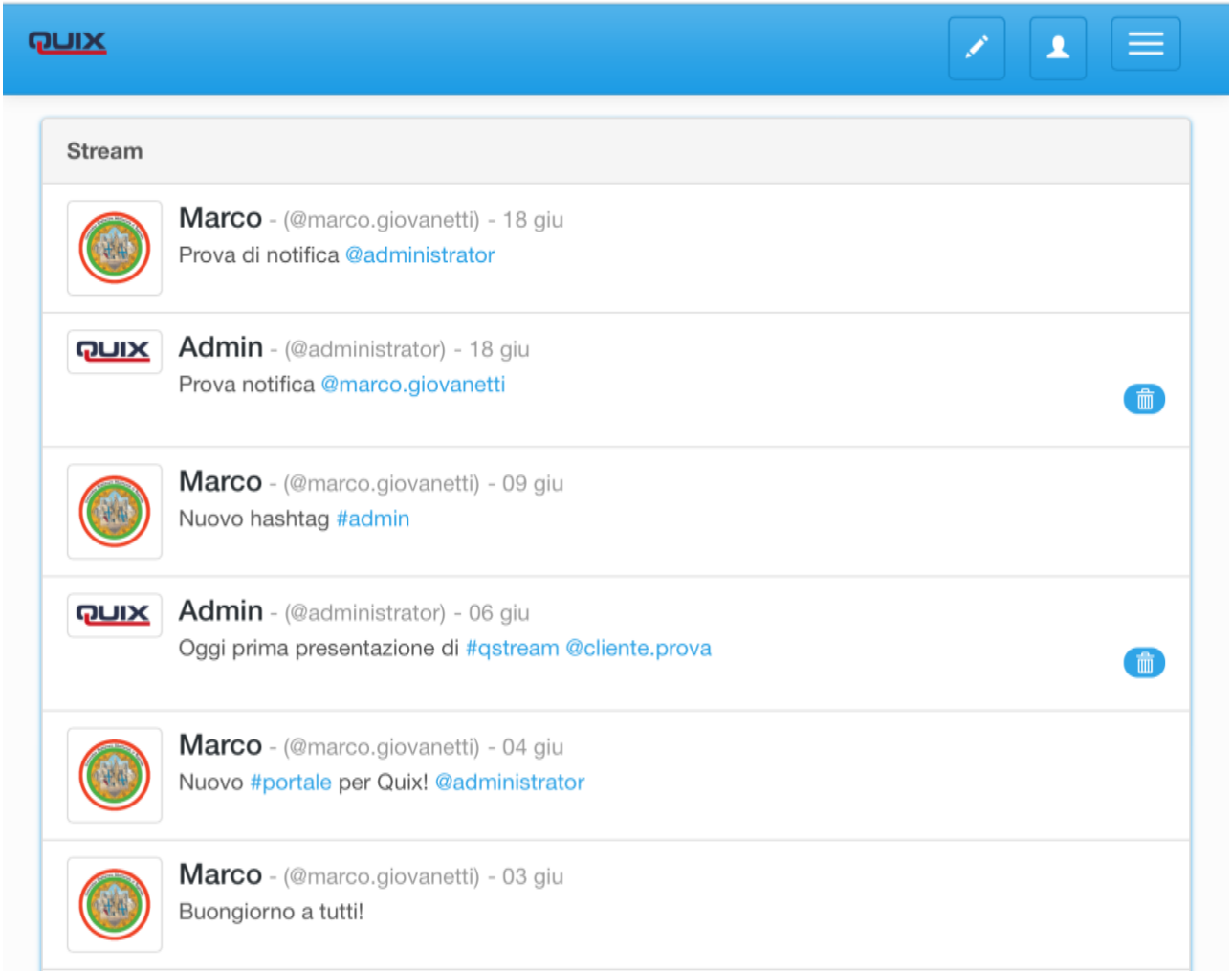


Figura 17: Homepage visualizzata su tablet

I cambiamenti che più si notano sono la riorganizzazione della barra di navigazione, nella quale gli elementi presenti nelle precedenti schermate sono accessibili cliccando il pulsante all'estrema destra, e l'aggiunta di due bottoni con icone che prima non erano presenti. Il pulsante rappresentato dall'icona a forma di matita permette la discesa di una tendina al cui interno si trova l'area di testo e che permette di scrivere un nuovo tweet. L'altro bottone con l'immagine stilizzata di un account permette invece di visualizzare, sempre tramite una tendina, il riquadro di riassunto delle informazioni dell'utente. In questo modo, il pannello contente il flusso di post rimane l'unico inizialmente presente nella pagina.

3.2.3 – View della pagina di visualizzazione account

Quando si accede al profilo di un utente, tramite il click del suo username nella home o per mezzo del suo url personalizzato, viene creata una pagina contenente le informazioni complete relative all'utente che si vuole vedere. Ciò è visibile solo se si hanno le autorizzazioni necessarie per visitare l'utente.

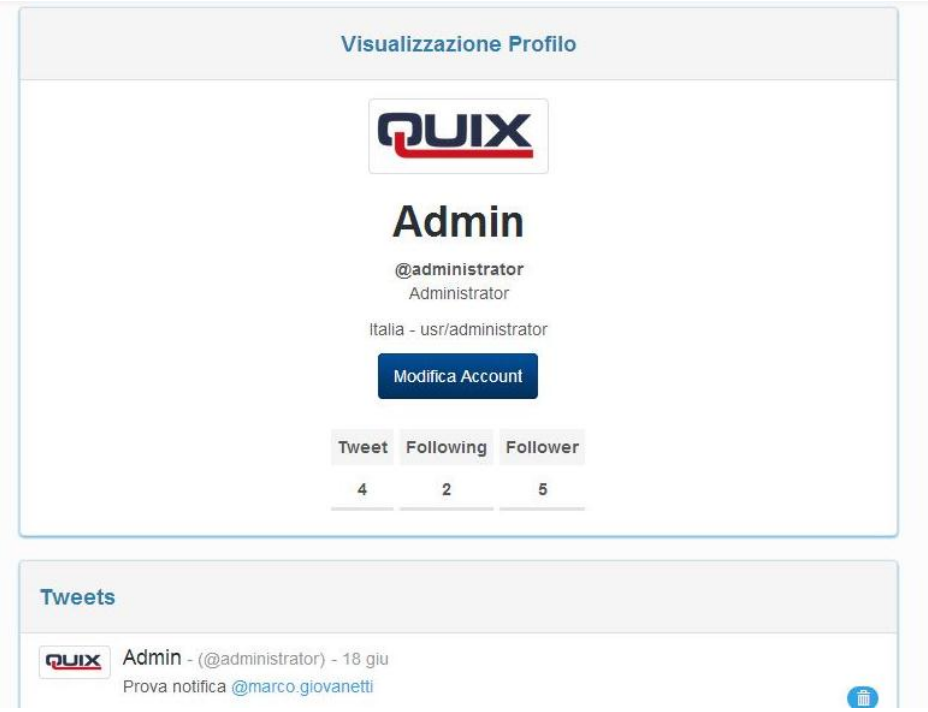


Figura 18: Vista profilo dell'utente connesso

Tramite questa schermata si può avere un'idea più completa riguardo alle informazioni dell'utente. Oltre alle proprietà citate precedentemente, come il nome visualizzato, il nome utente, la descrizione e la locazione, all'interno di questa pagina si possono visualizzare anche l'url personalizzato dell'account, relativamente ad un indirizzo internet fissato, e le informazioni relative al numero di tweet, di utenti seguiti (following) e di utenti che seguono l'utente (follower). Questa vista cambia a seconda che il profilo che si sta visualizzando sia dell'utente che è correntemente connesso o un utente diverso da quest'ultimo. Nel primo caso, come si può vedere dall'immagine sopra, sotto alle informazioni del profilo si trova un bottone che permette di accedere alla pagina di modifica dell'account, che verrà mostrata più avanti. Nel secondo caso, al posto del bottone "Modifica Account", vi è un bottone diverso che indica se l'utente di cui si sta visualizzando il profilo è seguito o meno dall'utente connesso correntemente. Nel caso l'utente non sia seguito, il bottone che comparirà permetterà di instaurare una nuova relazione di following tra l'account loggato e quello scelto. Viceversa, se il profilo che si sta visitando non compare tra la lista di follower dell'utente collegato, il bottone "Segui" verrà sostituito da quello che permette di eliminare la relazione di following che intercorre tra i profili.

Figura 19: Pulsanti per l'azione di following (utente seguito e non seguito rispettivamente)

La creazione della relazione di following tra i due utenti è effettuata tramite una chiamata AJAX che, richiamando una servlet, inserisce nella tabella adeguata del database gli user_id del profilo che segue e di quello seguito nelle corrispondenti righe. Se è eseguita con successo, la funzione cambia il pulsante e aggiorna il numero di follower di entrambi gli account. Anche in questa pagina risultano cliccabili gli elementi presentati nella

schermata di home, come ad esempio link, menzioni, hashtag e l'immagine di profilo, che è possibile ingrandire cliccandoci sopra. Inoltre questa pagina, come le altre, è stata progettata in ottica responsive e quindi è in grado di adattarsi alle differenti grandezze dello schermo del dispositivo in cui è richiamata.

Come descritto nelle precedenti pagine, se lo schermo dal quale viene visualizzata la pagina è più piccolo di una certa soglia, la schermata si ridisegna per una migliore navigazione da dispositivi mobili. Come si può notare dalla figura, la barra di navigazione è collassata in un unico pulsante, tramite il quale è possibile accedere alle varie voci del menu. Inoltre è stato aggiunto un tasto, con l'icona di una casa, che permette il reindirizzo alla pagina di home.

3.2.4 – View della pagina di ricerca

Tramite la barra di navigazione è anche possibile effettuare ricerche di utenti o hashtag. Nel primo caso si può trovare un profilo utilizzando come chiave di ricerca il nome utente o il nome visualizzato, mentre nel secondo basta utilizzare una qualsiasi stringa e verranno restituiti tutti i messaggi al cui interno compare la parola ricercata preceduta dal simbolo "#".

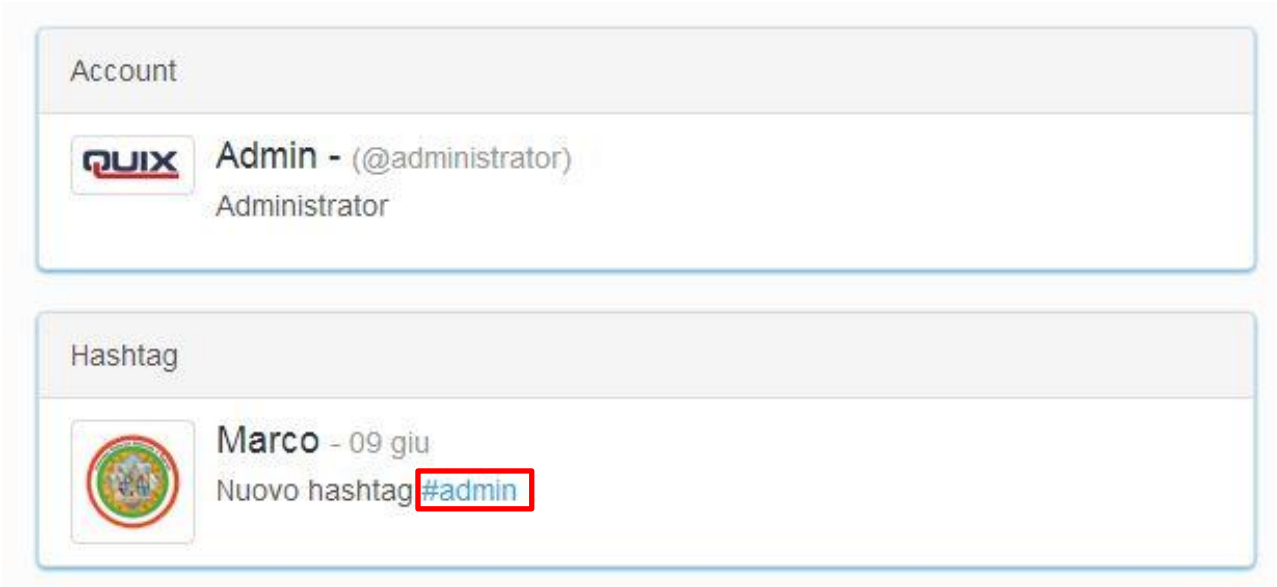


Figura 20: View di ricerca tramite stringa "admin"

Questa pagina raccoglie i dati ricercati sul database per mezzo di due query SQL distinte. La prima interrogazione sul database si occupa di ricercare tutti gli account il cui nome utente o il nome visualizzato contiene al suo interno la stringa ricercata. Per quanto riguarda la seconda, invece, essa si occupa della ricerca di tutti i tweet contenenti un hashtag corrispondente a quello cercato.

3.2.5 – View della pagina di modifica dell'account

Visitando il proprio profilo e cliccando il bottone "Modifica Account" si può accedere alla pagina di modifica dei dati dell'utente. In questa pagina è possibile cambiare l'immagine del profilo, lo stato di provenienza e la descrizione. Oltre a queste proprietà, modificabili da qualunque utente, se l'account è di ruolo avanzato comparirà un'ulteriore attributo che permetterà di scegliere i gruppi di utenti che saranno abilitati a seguire il profilo. Questa è una delle più importanti proprietà presenti nel sistema, in quanto la scelta dei profili abilitati è affidata totalmente all'account avanzato e non è settata in alcun modo di default.

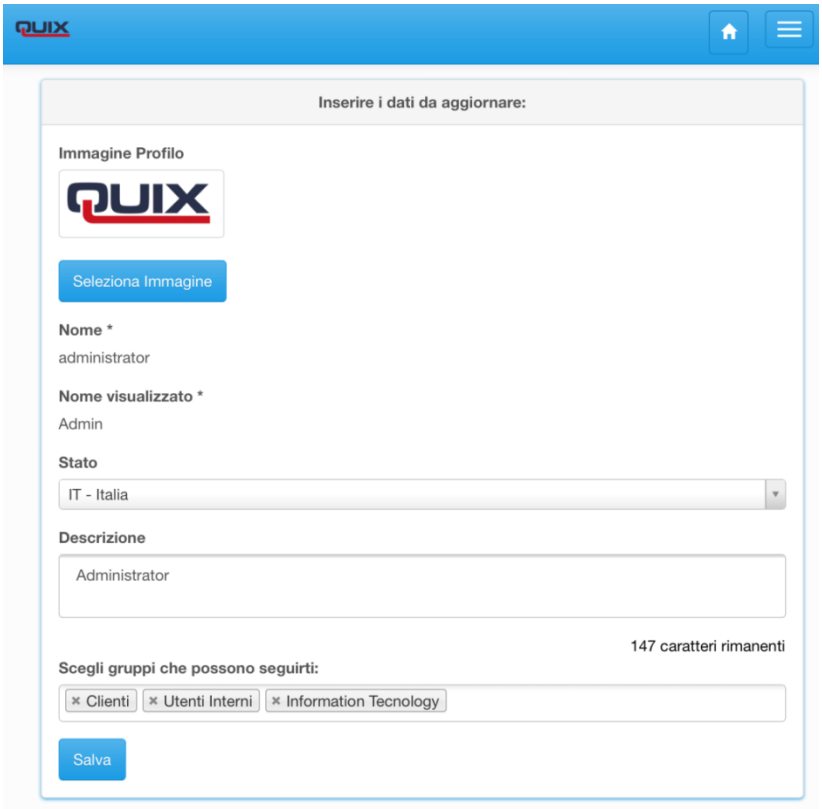


Figura 21: Vista della pagina di modifica account

Per cambiare l'immagine del profilo si dovrà cliccare per prima cosa il bottone "Seleziona Immagine", quindi selezionare l'immagine scelta dal filesystem. A questo punto, l'immagine sarà caricata in memoria e verrà visualizzata un'anteprima. Sarà poi necessario premere il tasto "Salva" perché la modifica sia effettivamente registrata sul database. Quest'ultima operazione sarà ricordata all'utente per mezzo di un piccolo popup che comparirà alla creazione dell'anteprima. L'ulteriore operazione di dover premere il bottone "Salva" dopo aver visualizzato l'anteprima è stato inserito per evitare che vengano salvati ogni volta i dati di un' immagine che poi potrebbe essere sostituita.

Esiste inoltre una limitazione sulla lunghezza della descrizione dell'utente, che è di 160 caratteri. Superato il limite, non sarà permesso salvare il profilo a meno che non vengano cancellati i caratteri in numero sufficiente a rientrare nel limite.

Per quanto riguarda la scelta dei gruppi abilitati, la ricerca di questi viene fatta tramite una chiamata AJAX sul database che restituisce un suggerimento dopo la digitazione delle prime lettere e permette di selezionare uno tra i gruppi selezionabili. Nel caso non esista nessun gruppo che inizia per le lettere digitate, non sarà possibile scegliere una voce dall'elenco.

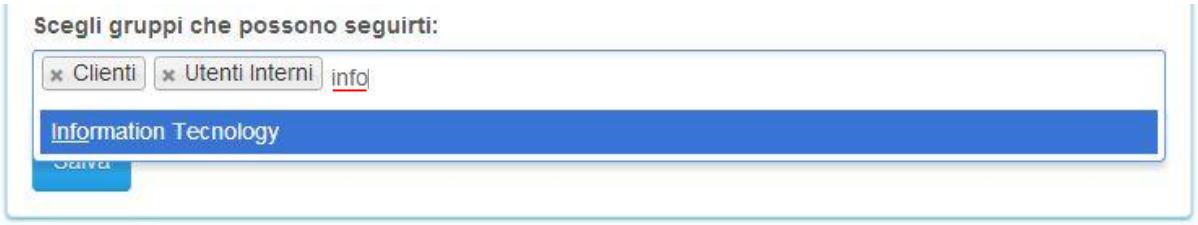


Figura 22: Ricerca del gruppo contenente la stringa "info"

3.2.6 – Cenni sulle viste delle pagine di visualizzazione dei follower e following

Oltre alle pagine già citate, si può inoltre accedere a due pagine che permettono la visualizzazione della lista di utenti seguiti e che seguono il proprio profilo. Ciò è possibile grazie ai due bottoni, presenti nella barra di navigazione, indicati con le etichette “Lista Follower” e “Lista Following”.

Figura 23: Pagina di lista Follower

L’immagine precedente mostra gli utenti che seguono il profilo correntemente loggato. La pagina di lista dei following è equivalente dal punto di vista grafico, con le uniche differenze nelle indicazioni testuali presenti nella pagina (Ad esempio, il titolo sarà Following e non Follower).

3.2.7 – Cenni sulle viste delle pagine di errore e di contenuto non disponibile

Oltre alle schermate già presentate, la web-application prevede altre due pagine che sono visualizzate solamente in casi speciali. La prima è la pagina che viene richiamata nel caso ci sia stato un errore in fase di esecuzione.

Figura 24: View della pagina di errore

Gli errori che provocano la visualizzazione di questa schermata sono sostanzialmente due: il primo caso in cui si può incorrere potrebbe essere una distrazione nella battitura nella url che si vuole richiamare, per esempio digitando *qstream/hoem* invece di *qstream/home*, che è appunto l'indirizzo corretto. Il secondo caso, mostrato in figura, è invece frutto di un errore imprevisto e non gestibile all'interno dell'applicazione, come, per esempio, l'impossibilità di raggiungere il database. In questo caso, la pagina che si apre permette all'utente di inviare una segnalazione via mail all'amministratore del sistema, anche se molto spesso il problema è risolvibile provando a ricaricare la pagina entro pochi minuti.

La seconda schermata, visibile nella figura della pagina successiva, è richiamata invece nel caso che si cerchi di accedere ad un profilo privato.

Figura 25: View della pagina di contenuto non accessibile

Questa situazione si presenta soprattutto quando un utente standard cerchi di visualizzare le informazioni riguardanti un profilo con stesso ruolo o un utente avanzato che non lo ha incluso tra i possibili follower. L'errore che viene intercettato corrisponde al codice 403 del protocollo HTTP, in cui il server ha ricevuto e compreso la richiesta ma rifiuta l'accesso alla pagina indicata nell'url.

Conclusioni ed eventuali implementazioni future

Il tirocinio effettuato presso Quix s.r.l. mi ha permesso di acquisire conoscenze approfondite nello sviluppo delle web-application.

Nelle prime settimane ho svolto uno studio personale sui linguaggi e i framework che avrei dovuto utilizzare nello svolgimento del progetto. Inoltre i dipendenti dell'azienda mi hanno guidato nella realizzazione dell'applicazione di base, mostrandomi come vengono organizzati gli applicativi all'interno del gruppo di lavoro aziendale (gerarchia delle cartelle, organizzazione delle classi, ecc.).

Nelle settimane successive e per tutta la durata del tirocinio è stata importante la ricerca personale effettuata tramite internet e manuali per il superamento di problemi che si sono presentati durante lo sviluppo del progetto e questo mi ha permesso di acquisire una buona tecnica di "problem solving". Non è inoltre mancato il supporto dei ragazzi dell'azienda, che mi hanno supportato e sostenuto durante le varie fasi dello sviluppo.

Il progetto svolto mi ha permesso di avere una panoramica completa e approfondita della varie fasi che compongono lo sviluppo di un software aziendale, dalla progettazione fino alla realizzazione vera e propria, consentendomi di mettere in pratica tutte le conoscenze acquisite in ambito universitario ed acquisendone di nuove.

L'applicazione può essere migliorabile ulteriormente come descritto di seguito:

- Aggiunta della possibilità di inviare messaggi privati ad un utente e non visibile ad altri;
- Aggiunta della possibilità di scegliere i singoli profili all'interno dei gruppi che sono abilitati a seguire il profilo;
- Aggiunta della possibilità di etichettare messaggi come "importanti";
- Ottimizzazione del sistema di notifica, con aggiornamento in tempo reale e invio di notifiche push nel caso sia inviato un messaggio importante al proprio profilo;
- Migliorie varie all'interfaccia grafica per rendere l'applicazione maggiormente "user-friendly".

Indice delle figure

Bibliografia

Sitografia

- http://www.quix.it
- http://en.wikipedia.org
- http://bootstrapdocs.com/v3.1.0/docs/components
- http://docs.oracle.com/javaee

Bibliografia

- Chris Anley, "*Advanced SQL injection in SQL Server applications*", Next Generation Security Software Insight Security Research (NGSSISR) Publication, 2002
- David McFarland, "*JavaScript & JQuery: The Missing Manual, Second Edition*", O'Reilly, 2012
- David Flanagan, "*JavaScript: The Definitive Guide, 6th Edition*", O'Reilly, 2011
- Alberto Rigenti, "*Analysis and Development of Functions in Rest Logic: Application to the "DataView" Web App*", Tesi di laurea triennale, relatore: Prof. Sonia Bergamaschi.