Università degli studi di Padova

Arya - Distributore di contenuti nei social network

Relazione su tirocinio breve svoltosi presso Mediacom Service srl

Laurendo Marco Massarotto
Relatore Sergio Congiu

Corso di Laurea in Ingegneria Informatica DM 509/99

Data di Laurea 29/03/2011

Anno accademico 2010/2011

Indice

1. Introduzione (Comune)

1.1. Descrizione degli obiettivi

L'obbiettivo dello stage presso Mediacom Service srl è stato quello di sviluppare una sorta di espansione per i siti da loro realizzati al fine di integrare la pubblicazione di contenuti nel sito con la pubblicazione nei social networks.

1.2. Descrizione del contesto organizzativo

Mediacom Service srl è una società specializzata nella realizzazione di siti internet, sia di tipo informativo che di e-commerce, per aziende.
La società ha realizzato un CMS di nome "Sapore" e con esso sono stati e vengono tutt'ora realizzati tutti i siti, personalizzandoli nelle funzionalità e variandone la grafica a seconda delle esigenze del cliente.
Ciò nonostante la struttura di base, ovvero il modo in cui il database è realizzato e la formattazione dei dati inseriti, è comune a tutti i siti.
Il CMS è anche molto modulare e questo permette di poter sviluppare un'applicazione che può essere facilmente e velocemente implementata sia nei siti in fase di realizzazione che in quelli realizzati in passato.

1.3. Organizzazione del lavoro di stage

Lo stage ha visto partecipazione di due stagisti, Marco Massarotto e Barbara Giardina.

Il lavoro è iniziato con una fase di studio della situazione attuale effettuata in comune dai due stagisti. Sono stati analizzati i linguaggi di programmazione da utilizzare, la struttura del CMS "Sapore" e le funzionalità messe a disposizione dei social network.

È stata poi progettata in comune la struttura base della comunicazione inter-client e il protocollo per la sua autenticazione.

La fase di lavoro comune è terminata con la definizione astratta della delle classi utilizzate nel back-end.

Con queste informazioni a disposizione, necessarie per la corretta suddivisione dei compiti e per l'obbligatoria compatibilità successiva del codice realizzato, il lavoro è stato assegnato nel seguente modo:

Barbara Giardina si è occupata della realizzazione di arya-connect.php, la componente del middle-end che estrae i dati dal CMS e invia i comandi al back-end, e della definizione di tutti i metodi di pubblicazione di tutti i social network, realizzando le varie classi che estendono la classe base socialcontenuta nel file social.php e implementando le api messe a disposizione dai diversi siti.

Marco Massarotto ha progettato e realizzato il front-end, che visualizza le informazioni all'utente e che ne raccoglie i comandi, la pagina command.php, che riceve le chiamate dal middle-end le verifica e le esegue, nonché del "motore" dell'autenticazione Oauth definito nelle classi social.php e socialoauth.php.

La fase finale di testing e di risoluzione dei problemi è stata effettuate in comune.

2. Analisi della situazione di partenza

2.1. Contesto aziendale

2.1.1. Aspettative accertate(comune)

Con un bacino complessivo di oltre 800 milioni di utenti i Social Network sono sempre più utilizzati a scopi pubblicitari da società che vogliono far conoscere il proprio marchio e avvicinarsi ai propri clienti per essere al corrente delle loro opinioni e dei loro suggerimenti.

Per garantire visibilità ogni società deve poter disporre di una pagina in ognuno dei social network principali. Affinché la pagina del marchio raggiunga una certa popolarità e non venga dimenticata dai suoi sostenitori è necessario che sia sempre aggiornata e che vi siano pubblicati frequentemente nuovi contenuti.

Internet però, lo sappiamo, non è mai uniforme ed è in continuo mutamento.
Nuovi social network nascono e muoiono ogni giorno e non è possibile prevedere a priori quali raggiungeranno il successo e quali saranno presto abbandonati.

Oltre al grande numero di social network ai quali un'azienda deve partecipare si aggiunge il problema della non standardizzazione delle metodologie di pubblicazione. Il risultato di tutto ciò è che gli addetti preposti all'inserimento delle notizie sono costretti di volta in volta ad eseguire operazioni lunghe e complicate per poter svolgere il loro compito ed inserire un contenuto in tutte le pagine da loro amministrate diffuse in diversi social, oltre che "nell'obbligatorio" sito internet aziendali.

Diviene quindi necessario realizzare uno strumento che sia in grado di pubblicare autonomamente lo stesso contenuto su diversi social network e che sia sufficientemente modulare da poter seguire l'evoluzione della rete.

ARYA dovrà quindi permettere l'inserimento delle notizie desiderate nei social più famosi e permettere l'implementazione di nuovi social in maniera trasparente per l'utente finale, che dovrà occuparsi semplicemente di pubblicare la notizia nel suo sito principale e di avviare il comando di ripubblicazione al programma.

2.1.2. Tecnologia utilizzata e Dati esistenti

CMS "Sapore"(Marco)

Il CMS, acronimo di Content Management System, è un software che organizza e gestisce la pubblicazione di contenuti in un sito web eliminando la necessità per l'amministratore del sito di conoscere il linguaggio di programmazione in cui è scritto.
Nonostante i CMS non siano stati concepiti per il Web oggi il loro utilizzo più diffuso è rivolto proprio alla gestione di siti web, soprattutto se sono di grandi dimensioni e richiedono un frequente aggiornamento.

Tecnicamente parlando, un CMS è un'applicazione lato server che si appoggia su un database sufficientemente articolato per lo stoccaggio dei contenuti; è suddivisa in due parti: la sezione di amministrazione, che serve ad organizzare e supervisionare la produzione dei contenuti, e la sezione applicativa, che l'utente usa per fruire dei contenuti e delle applicazioni del sito.
L'amministratore del CMS gestisce dal proprio terminale, tramite un pannello di interfaccia e controllo, i contenuti da inserire o modificare e le modalità di pubblicazione.

Un software, per potersi definire CMS, deve fornire le seguenti funzionalità:

- Identificazione degli utenti di amministrazione e dei relativi ruoli di produzione o fruizione dell'informazione
- Assegnazione di responsabilità e permessi a differenti categorie di utenti per distinti tipi di contenuti (in un progetto complesso il prodotto finito non è frutto del lavoro del singolo, che pertanto non ha possibilità o esigenza di intervenire in tutti gli ambiti)
- Definizione delle attività di workflow, cioè formalizzazione di un percorso per l'assemblaggio del prodotto finale che, in quanto frutto di produzione frammentaria, deve acquisire la sua unitarietà sottostando a opportune procedure di supervisione.
- Tracciamento e gestione delle versioni del contenuto
- Pubblicazione del contenuto
- Definizione del palinsesto editoriale

Il software "Sapore" sviluppato e utilizzato da Mediacom Service srl soddisfa tutti questi requisiti. Fornisce infatti un sistema di gestione semplice ed immediato con possibilità di creare uno o più utenti amministrativi anche con diverse competenze e aree d'accesso.
È ovviamente possibile pubblicare contenuti e questa operazione può essere suddivisa in sottofasi, ad esempio:

- inserimento dell'articolo
- creazione della galleria di immagini
- verifica
- etc..

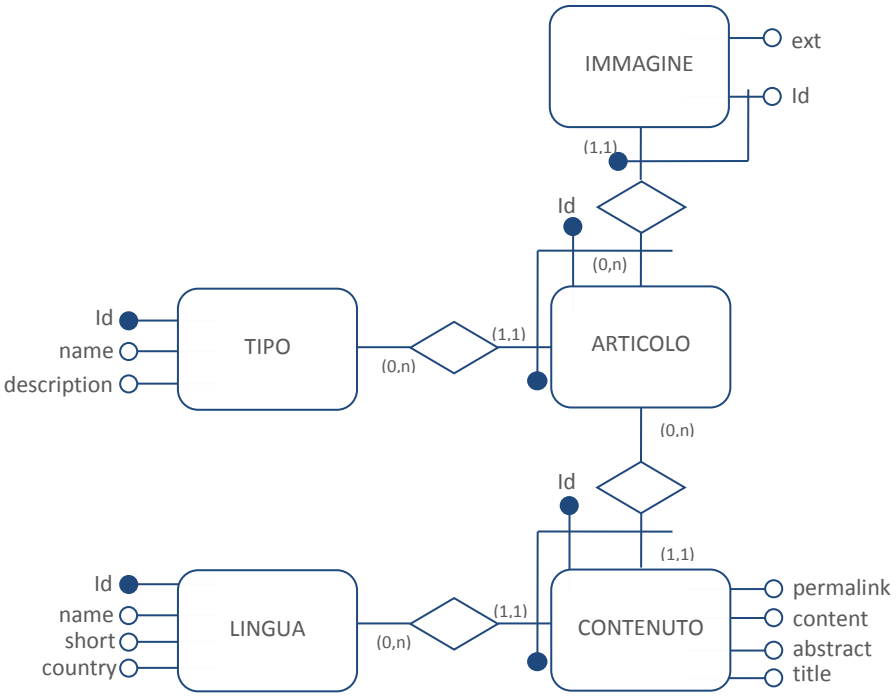
Infine i vari contenuti possono essere organizzati in macro aree e ordinati a piacimento al fine di poter definire il palinsesto editoriale desiderato.

Dal punto di vista prettamente tecnico "Sapore" si basa su una struttura a template.
Ogni tipologia di contenuti ha un proprio template, ovvero la semplice definizione in html della visualizzazione grafica della pagina nella quale vengono inseriti dei Tag specifici in sostituzione dei contenuti veri e propri.
La pagina che si occupa di visualizzare il template interpreterà i tag e li sostituirà con i contenuti effettivamente richiesti prelevandoli dal database e inserendoli nelle apposite strutture grafiche predefinite.

BASI DI DATI PREESISTENTI (BARBARA)

Per la comprensione del lavoro svolto è necessario dare una panoramica delle basi di dati preesistenti.
Verrà illustrata, anche per motivi di copyright , solo la parte di database di utilità per il nostro progetto.

Le entità che vengono usate sono tutte quelle che si riferiscono alle notizie inserite nel sito principale dell'azienda.
I tipi di notizie (entità "tipo") che di cui ci occuperemo sono tre, "message" "event" e "album"; un articolo inserito (entità "articolo") può appartenere ad un solo tipo. Un articolo può contenere da 0 a n immagini (presenti nell'entità "immagine") e avrà o meno un contenuto (entità "contenuto"). Vista la possibilità di inserire lo stesso articolo in più lingue possono esserci più entità "contenuto" collegate all'entità "articolo"; questo implica che per ottenere un certo articolo con il contenuto in italiano da inserire nei social network bisognerà indicare la lingua cercata.



2.2. Analisi Social Network

2.2.1. Analisi Generale (Comune)

Una rete sociale, più comunemente conosciuta con il termine inglese “social network”, consiste in un gruppo di persone collegate tra loro da legami sociali come ad esempio legami familiari, amicizie casuali, contatti di lavoro o relazioni sentimentali.

L’antropologo britannico Robin Dunbar nel 2007 ha introdotto in un suo studio un limite cognitivo teorico, chiamato numero di Dunbar, asserendo che il numero di persone con le quali un individuo è in grado di mantenere relazioni sociali stabili è limitato a 150; questo limite è funzione diretta della dimensione relativa della neocorteccia cerebrale ossia è un limite fisico e non psicologico. La stessa teoria asserisce inoltre che un numero di relazioni superiore al numero di Dunbar necessita di regole e leggi più restrittive per mantenere il gruppo stabile e coeso.

La versione di internet delle reti sociali è una delle forme più evolute della comunicazione in rete ed è anche un tentativo di violare il numero di Dunbar.

I social network sono una invenzione recente, il primo social network venne lanciato nel 1997 e si chiamava Six Degrees. I tempi però non erano ancora maturi e venne chiuso dopo soli tre anni. Internet come lo conosciamo oggi era agli albori e la gente non era ancora pronta ad un interazione così personale con il Word Wide Web.
Six Degrees però aveva aperto la strada e nel 2003, vennero pubblicati svariati social network, tra i quali LinkedIn e MySpace, anche se molti chiusero poco tempo dopo e molti altri non sono diventati famosi.
Gli utenti familiarizzarono molto velocemente con questo nuovi siti; collegarsi online diveniva il modo migliore per rimanere in contatto con amici e colleghi e allo stesso tempo rendeva possibile riscoprire vecchie amicizie e farne di nuove.

Grazie ai social network è inesorabilmente cambiato il modo di percepire amicizia e conoscenze e la distanza sociale fra le persone si è ridotta.
Quello che era successo all’avvento dei telefoni prima, e dei cellulari poi è accaduto di nuovo con l’avvento dei social network. Persone distanti fra di loro migliaia di chilometri possono essere mentalmente vicine come mai prima.

Anche il concetto di privacy è variato, un utente in un social network può imbattersi in pensieri, opinioni e fatti personali di un altro utente semplicemente entrando nella sua bacheca.

Noi ci occuperemo di sei social network differenti fra loro per interfaccia e funzionalità ma che condividono tra loro alcune caratteristiche fondamentali.

I Social Network principali sono, al momento, Facebook, Twitter, LinkedIn MySpace , Google Buzz e Yahoo! Pulse.

2.2.2. FACEBOOK (BARBARA)

Facebook è il più completo dei social network nonché il più famoso.
È infatti il secondo sito più visitato al mondo, preceduto solo da Google.

Ideato nel 2004 da Mark Zuckerberg per permettere il contatto tra i suoi colleghi universitari si è rapidamente evoluto fino a diventare una rete sociale che abbraccia trasversalmente tutti gli utenti di Internet.

Dall'agosto 2008 all'agosto 2009 gli utenti hanno subito una crescita esponenziale passando da 800.000 a 10.000.000. Nel febbraio 2010 gli utenti erano già 400.000.000 ed ora hanno superato la quota di 500.000.000. Se fosse una nazione sarebbe la terza al mondo dopo Cina ed India.
Il suo successo è stato tale da renderlo persino protagonista di un film.

Ogni utente crea un profilo che contiene informazioni di carattere personale, foto e interessi. Inoltre appartiene a una o più reti, gruppi di persone accumunate da caratteristiche simili come ad esempio nazionalità, scuola frequentate ecc.. Ogni utente ha inoltre una lista di amici.

In ogni momento nella sua bacheca compariranno commenti e notizie pubblicati dagli utenti che sono in quella lista.

Facebook consente di pubblicare commenti sotto forma di messaggio di stato o di nota, link, immagini e video e consente anche l'invio di messaggi privati che assumono la funzione di e-mail.

Si possono avere vari gradi di pubblicazione, nel menù di impostazioni sulla privacy chiunque può decidere a chi è permesso vedere determinate categorie di notizie.

Le scelte possibili sono "solo io", "solo amici", "amici di amici" e "tutti" ed è consentito anche bloccare delle persone specifiche presenti fra gli amici aggiungendo i nomi in un elenco.

È stata questa grande libertà di scelta, insieme alla semplicità d'utilizzo, che hanno consentito la grande diffusione di questa web application.

Da pochi anni è passata dall'essere solo uno strumento di incontro all'essere anche uno strumento pubblicitario aumentando così il suo fatturato e arrivando a superare il miliardo di euro nel 2010.

La società Facebook Inc. verrà a breve quotata in borsa.

2.2.3. MySpace (Barbara)

MySpace è stato creato nel 2003 da Tom Anderson e Chris DeWolfe, questo social ha permesso a molti gruppi e artisti come Arctic Monkeys, Lily Allen, i Belladonna, Mika e i Cansei de Ser Sexy di diventare famosi.

La sezione di MySpace dedicata all'Italia è stata lanciata ufficialmente nel maggio 2007.

Rispetto agli altri paesi, in Italia MySpace ha avuto una maggiore diffusione tant'è che il numero di utenti iscritti a MySpace Italia è superiore a 70.000; ha dovuto in ogni caso chiudere nel 2009 per un piano di ristrutturazione dovuto alla concorrenza degli altri social network.

Nel 2011 MySpace, messo in ginocchio da Facebook e Twitter, ha dovuto effettuare un ulteriore taglio della forza lavoro pari al 47%.

MySpace, come Facebook, consente ad ogni utente di avere una bacheca all'interno della quale sono presenti le sue informazioni personali e gli aggiornamenti da lui pubblicati che possono essere semplici commenti, link o foto.

2.2.4. Twitter (Marco)

Twitter è probabilmente il più semplice e immediato dei social network.

Creato nel marzo 2006 dalla Obvius Corporation di San Francisco, è un servizio che fornisce agli utenti una pagina personale aggiornata da loro stessi tramite messaggi, chiamati tweet, di lunghezza massima 140 caratteri.
Per questo motivo Twitter oltre che Social Network viene chiamato anche microblogging.

I tweet possono essere effettuati in svariati modi, consentono la pubblicazione di contenuti nella propri pagina in praticamente ogni situazione; è possibile inviare i nuovi aggiornamenti tramite il sito web, via SMS, con programmi di messaggistica istantanea, via e-mail oppure tramite varie applicazioni basate sulle API di twitter, ARYA sarà una di queste, e vengono visualizzati oltre che sulla bacheca dell'utente stesso anche sulle bacheche degli utenti che si sono registrati per riceverli.

Infatti un utente di Twitter può avere una lista di amici ma al contrario degli altri social per ricevere gli aggiornamenti di stato di un qualunque utente deve inserirlo in una lista apposita a meno che quest'ultimo non indichi esplicitamente il suo nome preceduto dal carattere @.Questa lista può contenere chiunque, questo implica che tutti i messaggi scambiati attraverso Twitter sono messaggi pubblici.

Il servizio è diventato così popolare grazie all'immediatezza e alla facilità di utilizzo; la società che lo gestisce non è quotata in borsa ma si stima valga circa un miliardo di dollari.

Il 30 aprile 2009 Twitter ha cambiato la sua interfaccia web con l'aggiunta di una barra di ricerca e un riassunto di temi di attualità, cioè le frasi più comuni che compaiono nel messaggio.

Grazie a questo ora ogni aggiornamento inviato a Twitter da qualsiasi parte del mondo può essere immediatamente indicizzato e utilizzato per la ricerca in tempo reale e con questa funzione Twitter è diventato un motore di ricerca per trovare ciò che sta accadendo ora.

Altra particolarità di Twitter è il fatto di poter etichettare parole o frasi precedendole dal tasto #; in questo modo quando quella parola o frase verrà cercata dagli utenti tutti i Tweet che la contengono verranno visualizzati.

Non permette di caricare ne immagini ne altri contenuti multimediali.

2.2.5. YAHOO! PULSE (BARBARA)

Dal nome si vede subito che **Yahoo! pulse** è il social network della società fornitrice di servizi internet Yahoo! Inc.

Yahoo! fu ideato nel 1994 da David Filo e Jerry Yang ed ebbe una storia simile a quella di Facebook con la fondamentale diversificazione che Facebook fu fondato da subito come social network mentre il social network di Yahoo! venne creato insieme a tanti altri servizi quali il motore di ricerca (Yahoo!), la casella e-mail (Yahoo! Mail), il famoso yahoo! answers dove le persone della comunità di Yahoo! possono fare domande e ricevere o dare risposte, il servizio di incontri (yahoo! incontri) e tanti altri servizi.

Dopo il boom avuto nei primi anni della creazione, dove il consiglio di amministrazione di Yahoo! si era allargato e questa era entrata in borsa, e dopo il rifiuto del consiglio di amministrazione di vendere la società alla Microsoft, avvenuto nel gennaio 2008, la Yahoo! Inc. ha avuto una rapida svalutazione passando da 31 dollari per azione del febbraio 2008 agli 11 del dicembre dello stesso anno, dove la sua discesa si è fermata forse anche grazie alle dimissioni di uno dei soci maggiori e fondatore Jerry Yang.

Da allora Yahoo! non ha più avuto grandi novità e il social network Yahoo! pulse, già non uno dei social network più popolari, è stato surclassato dai nuovi social Facebook , MySpace e Twitter.

Il funzionamento del social è molto simile a quello di MySpace e Facebook; anche qui l'utente avrà una bacheca dove postare i suoi messaggi e che, a meno che non li si blocchi cambiando le impostazioni sulla privacy, tutti gli altri utenti potranno vedere.

Yahoo consente di pubblicare commenti di stato, eventi e link. Per fare questo si appoggia a servizi web esterni invisibili all'utente.

2.2.6. LINKEDIN (MARCO)

LinkedIn è un servizio di Social Network impiegato principalmente in ambito professionale.

Fondato nel 2003 a Montain View in Calfornia da Reid Hoffman si è rapidamente diffuso grazie anche alla sua utilità.
LinkedIn, infatti, permette agli utenti registrati di mantenere una lista di persone ritenute affidabili in ambito lavorativo. L'uso che si può fare di questo programma è molteplice: si può ottenere di essere presentati a qualcuno che si conosce attraverso un contatto affidabile; un datore di lavoro può pubblicare offerte e ricercare potenziali candidati, così come una persona può cercare offerte di lavoro e grazie a questo servizio è anche in grado di cercare fra i propri contatti qualcuno che può metterli direttamente in contatto con il datore di lavoro.

Nel Gennaio 2009 LinkedIn contava circa 30 milioni di utenti che in poco più di un anno sono rapidamente raddoppiati arrivando a 68 milioni nel maggio 2010.
LinkedIn spazia su oltre 150 comparti economici e oltre 400 regioni economiche. È un social network molto diffuso in tutti i paesi del mondo, principalmente negli US, in India e nel Regno Unito, e cresce a velocità di 100.000 iscritti a settimana.

È possibile inserire solo messaggi di testo e link anche se, per l'uso che si fa di questo social network, sono sufficienti.

Grande limite di LinkedIn è invece il fatto di poter effettuare al massimo 5 richieste al giorno per l'invio di messaggi tramite API.

2.2.7. GOOGLE BUZZ (BARBARA)

Google Buzz è uno strumento di Social Networking e microblogging creato da Google ed implementato nel servizio di posta elettronica Gmail.

Ogni persona che si iscrive sul sito di Google avrà accesso a tutti i servizi di Google, compresi Google Buzz, Gmail, Google Docs e altri.

Google Buzz è stato presentato al pubblico il 9 febbraio 2010 e consente di integrare con link, foto e video le conversazioni che avvengono su Gmail.

Secondo i fondatori di Google non è un tentativo di competere con gli altri social network ma un'applicazione che aiuterà a colmare il gap tra lavoro e tempo libero dove per gap si intende il termine inglese che indica lo spazio che intercorre tra due cose discontinue.

Ogni utente ha la possibilità di "seguire" tutti gli altri utenti presenti nella sua rubrica semplicemente inserendoli nella "lista degli utenti seguiti", da quel momento tutti i loro commenti compariranno nella bacheca di Google Buzz. Funzionalità del tutto simile a Twitter.

Ogni utente della rubrica può mandare un messaggio sulla bacheca di un proprio contatto, anche se quest'ultimo non lo ha aggiunto nella sua lista, semplicemente inserendo nel messaggio il suo nome preceduto da @ ed il suddetto messaggio gli verrà inviato anche per e-mail.

3. TECNOLOGIE UTILIZZATE

3.1. LINGUAGGI DI PROGRAMMAZIONE(COMUNE)

3.1.1. HTML

DEFINIZIONE:

Anche se il titolo del capito è “linguaggi di programmazione” bisogna subito dire che l’HTML non è un linguaggio di programmazione, per lo meno non nel vero significato del termine.

L’HTML non ha meccanismi che consentono di prendere delle decisioni e non è in grado di compiere delle iterazioni né ha i costrutti propri della programmazione, per questi motivi l’HTML si dice linguaggio statico.

L’HTML è un linguaggio di markup, ossia un linguaggio che descrive attraverso dei tag le modalità di impaginazione, formattazione o visualizzazione grafica del contenuto di una pagina web.

Per i motivi descritti sopra una pagina scritta in puro HTML in un dato dispositivo si vedrà sempre allo stesso modo.

Tuttavia l’HTML supporta l’inserimento di script e oggetti esterni quali immagini o filmati.

Pur avendo una propria sintassi l’HTML non è così rigoroso nel farla rispettare, se ci si dimentica di chiudere un tag non verranno prodotti messaggi di errore ma semplicemente non verrà interpretato come tag e resterà visualizzato come semplice testo.

Un’altra importante caratteristica di HTML è che esso è stato creato per definire il contenuto logico e non l’aspetto finale del documento e questo perché i dispositivi che possono accedere ad un documento HTML sono molteplici e con capacità grafiche differenti. Questo significa che uno stesso documento verrà visto molte volte in modo diverso su due dispositivi differenti ma che potrà essere visto su tutti i dispositivi, anche i meno recenti.

Questo ha garantito la massima diffusione di internet anche se ha posto dei forti limiti agli sviluppatori delle pagine Web.

Attualmente i documenti HTML sono in grado di incorporare varie tecnologie che offrono la possibilità di aggiungere al documento ipertestuale controlli più sofisticati sulla resa grafica, di avere interazioni dinamiche con l’utente come ad esempio animazioni interattive e di aggiungere contenuti multimediali.

Si tratta di linguaggi come CSS, JavaScript o Java, o di altre applicazioni multimediali di animazione vettoriale o di streaming audio o video.

È possibile delegare la scrittura del codice HTML ad applicazioni specifiche che permettono al designer di occuparsi dell’aspetto grafico finale della pagina mentre il codice viene generato automaticamente questo però riduce il controllo sulla pulizia del codice.

In qualunque modo siano generati, i documenti HTML hanno estensione punto HTML (.html) o punto HTM (.htm). (Wikipedia, l'enciclopedia libera, p. HTTP)

STORIA:

“Il linguaggio HTML è stato sviluppato alla fine degli anni ottanta da Tim Berners-Lee al CERN di Ginevra assieme al noto protocollo HTTP che supporta invece il trasferimento di documenti in tale formato. Verso il 1994 ha avuto una forte diffusione in seguito ai primi utilizzi commerciali del web.

Nel corso degli anni, seguendo lo sviluppo di Internet, l'HTML ha subito molte revisioni, ampliamenti e miglioramenti, che sono stati indicati secondo la classica numerazione usata per descrivere le versioni dei software. Attualmente l'ultima versione disponibile è la versione 4.01, resa pubblica il 24 dicembre 1999. Dopo un periodo di sospensione, in cui il W3C si è focalizzato soprattutto sulle definizioni di XHTML (applicazione a HTML di regole e sintassi in stile XML) e dei fogli di stile (CSS), nel 2007 è ricominciata l'attività di specifica con la definizione, ancora in corso, di HTML5, attualmente allo stato di bozza (draft).” (Wikipedia, l'enciclopedia libera, p. HTTP)

VERSIONE UTILIZZATA:

La versione utilizzata per una data pagina HTML è definita nel doctype e compare nella prima riga della stessa.
Per i nostri documenti abbiamo deciso di utilizzare la versione //W3C//DTD HTML 4.01 TRANSITIONAL.
Nel documento queste specifiche appariranno in questo modo:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" >

Questa riga oltre a indicare la versione fornisce anche alcune informazioni sul contenuto.
Fa vedere che è un documento HTML pubblico che fa riferimento alle specifiche rilasciate dal W3C.

Il w3c è il consorzio ufficiale di Internet che sviluppa tecnologie per guidare la rete fino al massimo del suo potenziale.

Se un documento segue lo standart w3c vuol dire che è realmente accessibile da tutti gli utenti, indipendentemente dal dispositivo utilizzato.

Il segno – indica che stiamo usando specifiche non registrate all'ISO (organizzazione di standardizzazione internazionale).

Il documento fa riferimento a una DTD ("Document Type Definition" cioè "Definizione del tipo di documento") scritta in lingua inglese e la versione di HTML supportata è la 4.01 "transitional".

Per quel che riguarda l'HTML le indicazioni possibili sono Strict, Transitional o Frameset.

La strict è una DTD particolarmente rigorosa: esclude ogni elemento che riguarda il layout (la cui formattazione è affidata all'utilzzo dei CSS) e non è consentito l'uso degli elementi deprecati. Abbiamo deciso di non usarla per essere più liberi nella scrittura del testo.

La transitional è una versione temporanea, per consentire il passaggio da una specifica all'altra. Nella DTD transitional i tag deprecati sono ammesi. Questa DTD andrà bene nella maggior parte dei casi ed infatti è la più comunemente utilizzata.

La frameset è la DTD che riguarda i frames e verrà utilizzata nel frame presente nel CMS sapore.

3.1.2. PHP

DEFINIZIONE:

PHP è l'acronimo ricorsivo di *"PHP:Hypertext Preprocessor"*.

È un linguaggio la cui funzione fondamentale è quella di produrre codice HTML.
Poiché PHP è un linguaggio di programmazione lato Server, abbiamo la possibilità di analizzare situazioni differenti, come ad esempio l'input degli utenti o i dati contenuti in un database, e di decidere di produrre codice HTML condizionato ai risultati dell'elaborazione.
Quando il server riceve una richiesta per una pagina PHP la fa analizzare dall'interprete del linguaggio il quale restituisce un file contenente solo il codice che deve essere inviato al browser che sarà principalmente HTML ma che potrà contenere anche codice JavaScript, fogli di stile CSS o qualunque altro contenuto fruibile da un browser, come immagini e documenti Pdf. Questo è ciò che viene definito **Web dinamico**.

La dinamicità può essere lato client o lato server e PHP si occupa della seconda modificando la pagina prima che questa venga trasmessa all'utente. Per avere dinamicità lato client, in cui la pagina viene modificata dopo che è stata scaricata dal browser, viene utilizzato il codice Javascript, spiegato nel capitolo successivo.

La combinazione di queste due tecniche si può ottenere sfruttando il sistema chiamato AJAX che prevede di modificare dinamicamente i contenuti della pagina visualizzata, senza effettuare un refresh della stessa, caricandoli da pagine dinamiche PHP.

STORIA:

A metà degli anni Novanta il Web era ancora formato in gran parte da pagine statiche, cioè da documenti HTML il cui contenuto non poteva cambiare fino a quando qualcuno non interveniva manualmente a modificarlo. Con l'evoluzione di Internet, però, si cominciò a sentire l'esigenza di rendere dinamici i contenuti, cioè di far sì che la stessa pagina fosse in grado di proporre contenuti diversi, personalizzati in base alle preferenze degli utenti, oppure estratti da una base di dati (database) in continua evoluzione.

PHP nasce nel 1994, ad opera di Rasmus Lerdorf, come una serie di macro la cui funzione era quella di facilitare ai programmatori l'amministrazione delle homepage personali: da qui trae origine il suo nome, che allora significava appunto Personal Home Page. In seguito, queste macro furono riscritte ed ampliate fino a comprendere un pacchetto chiamato Form Interpreter (PHP/FI).

Essendo un progetto di tipo open source, ben presto si formò una ricca comunità di sviluppatori che portò alla creazione di PHP 3: la versione del linguaggio che diede il via alla crescita esponenziale della sua popolarità. Tale popolarità era dovuta anche alla forte integrazione di PHP con il Web server Apache (il più diffuso in rete), e con il database MySQL. Tale combinazione di prodotti, integralmente ispirata alla filosofia del free software, diventò ben presto vincente in un mondo in continua evoluzione come quello di Internet.

Alla fine del 1998 erano circa 250.000 i server Web che supportavano PHP: un anno dopo superavano il milione. I 2 milioni furono toccati in aprile del 2000, e alla fine dello stesso anno erano addirittura 4.800.000.
Il 2000 è stato sicuramente l'anno di maggiore crescita del PHP, coincisa anche con il rilascio della versione 4, con un nuovo motore (Zend) molto più veloce del precedente ed una lunga serie di nuove funzioni, fra cui quelle importantissime per la gestione delle sessioni.

La crescita di PHP, nonostante sia rimasta bloccata fra luglio e ottobre del 2001, è poi proseguita toccando quota 7.300.000 server alla fine del 2001, per superare i 10 milioni alla fine del 2002, quando è stata rilasciata la versione 4.3.0. La continua evoluzione dei linguaggi di programmazione concorrenti e l'incremento notevole dell'utilizzo del linguaggio anche in applicazioni enterprise ha portato la Zend a sviluppare una nuova versione del motore per supportare una struttura ad oggetti molto più rigida e potente.

Nasce così PHP 5, che si propone come innovazione nell'ambito dello sviluppo web open source soprattutto grazie agli strumenti di supporto professionali forniti con la distribuzione standard ed al grande sforzo di Zend che, grazie alla partnership con IBM, sta cercando di spingere sul mercato soluzioni di supporto enterprise a questo ottimo linguaggio. Lo sviluppo di PHP procede comunque con due progetti paralleli che supportano ed evolvono sia la versione 4 che la versione 5. Questa scelta è stata fatta poichè tuttora sono pochi i fornitori di hosting che hanno deciso di fare il porting dei propri server alla nuova versione del linguaggio.

Oggi PHP è un linguaggio completo di scripting, sofisticato e flessibile, che può girare praticamente su qualsiasi server Web, su qualsiasi sistema operativo (Windows o Unix/Linux, ma anche Mac, AS/400, Novell, OS/2 e altri), e consente di interagire praticamente con qualsiasi tipo di database (SQLite, MySQL, PostgreSQL, SQL Server, Oracle, SyBase, Access e altri). Si può utilizzare per i più svariati tipi di progetti, dalla semplice home page dinamica fino al grande portale o al sito di e-commerce. (Wikipedia, l'enciclopedia libera, p. PHP)

VERSIONE UTILIZZATA:

La versione di PHP da noi utilizzata e la 5.2.12

3.1.3. JAVASCRIPT

DEFINIZIONE:

JavaScript è un linguaggio di programmazione orientato agli oggetti comunemente usato nei siti web.

La caratteristica principale di JavaScript è quella di essere un linguaggio interpretato: il codice non viene compilato, ma interpretato (in JavaScript lato client l'interprete è incluso nel browser che si sta utilizzando). La sintassi è relativamente simile a quella del C, del C++ e del Java.

Il linguaggio definisce le funzionalità tipiche dei linguaggi di programmazione ad alto livello (strutture di controllo, cicli, ecc.) e consente l'utilizzo del paradigma object oriented seppur in maniera debole.

Ad esempio, il meccanismo dell'ereditarietà è più simile a quello del Self e del NewtonScript che a quello del linguaggio Java (che è un linguaggio fortemente orientato agli oggetti). Gli oggetti stessi ricordano più gli array associativi del Perl che gli oggetti di Java o del C++.

In JavaScript il codice viene eseguito direttamente sul client e non sul server.
Il vantaggio di questo approccio è che, anche con la presenza di script particolarmente complessi, il server non viene sovraccaricato a causa delle richieste dei clients.
Di contro, nel caso di script che presentino un sorgente particolarmente grande, il tempo per lo scaricamento può diventare abbastanza lungo. Un ulteriore svantaggio è che ogni informazione che presupponga un accesso a dati memorizzati in un database remoto deve essere rimandata ad un linguaggio che effettui esplicitamente la transazione, per poi restituire i risultati ad una o più variabili JavaScript; operazioni del genere richiedono il caricamento della pagina stessa. Con l'avvento di AJAX tutti questi limiti sono stati superati. (Wikipedia, l'enciclopedia libera, p. Javascript)

STORIA:

Fu originariamente sviluppato da Brendan Eich della Netscape Communications con il nome di Mocha e successivamente di LiveScript, ma in seguito è stato rinominato "JavaScript" ed è stato formalizzato con una sintassi più vicina a quella del linguaggio Java di Sun Microsystems. JavaScript è stato standardizzato per la prima volta tra il 1997 e il 1999 dalla ECMA con il nome ECMAScript. L'ultimo standard, del dicembre 1999, è ECMA-262 Edition 3, e corrisponde a JavaScript 1.5 ed è anche uno standard ISO. (Wikipedia, l'enciclopedia libera, p. Javascript)

VERSIONE UTILIZZATA:

JavaScript 1.5

3.1.4. SQL

DEFINIZIONE:

SQL sta per Structured Query Language ed è un linguaggio strutturato di interrogazione per basi di dati.

L'SQL è un linguaggio dichiarativo e per questo non richiede la stesura di sequenze di operazioni , cosa che invece serve per i linguaggi di programmazione imperativi quali Assembly e FORTRAN, ma le proprietà logiche delle informazioni ricercate.Esso si divide in tre sottoinsiemi

- Data Definition Language (DDL) - serve a creare, modificare o eliminare gli oggetti in un database. Sono i comandi DDL a definire la struttura del database e quindi dei dati ivi contenuti. Ma non fornisce gli strumenti per modificare i dati stessi: per tale scopo si usa il DML. L'utente deve avere i permessi necessari per agire sulla struttura del database e questi permessi vengono assegnati tramite il DCL.
- Data Manipulation Language (DML) - fornisce i comandi per inserire, modificare, eliminare o leggere i dati all'interno delle tabelle di un database. La struttura di questi dati deve già essere stata definita tramite il DDL. Inoltre, il permesso di accedere a tali dati deve essere assegnato all'utente tramite il DCL.
- Data Control Language (DCL) - serve a fornire o revocare agli utenti i permessi necessari per poter utilizzare i comandi DML e DDL, oltre agli stessi comandi DCL (che gli servono per poter a sua volta modificare i permessi su alcuni oggetti).

Per poter fare tutto questo l'SQL usa costrutti di programmazione denominati *query.*

Le istruzioni SQL possono essere inserite in linguaggi di programmazioni tradizionali come C e Java o anche in linguaggi del Web come PHP e Asp e permettono in questo modo di mettere in relazione le pagine interrogate con il database sottostante da cui prendere le informazioni.

STORIA:

L'SQL nasce nel 1974 ad opera di Donald Chamberlin, nei laboratori dell'IBM. Nasce come strumento per lavorare con database che seguano il modello relazionale. A quel tempo però si chiamava SEQUEL (la corretta pronuncia IPA è [ˈɛsˈkjuˈɛl], quella informale [ˈsiːkwəl]). Nel 1975 viene sviluppato un prototipo chiamato SEQUEL-XRM; con esso si eseguirono sperimentazioni che portarono, nel 1977, a una nuova versione del linguaggio, che inizialmente avrebbe dovuto chiamarsi SEQUEL/2 ma che poi divenne, per motivi legali, SQL. Su di esso si sviluppò il prototipo System R, che venne utilizzato da IBM per usi interni e per alcuni suoi clienti. Ma, dato il suo successo, anche altre società iniziarono subito a sviluppare prodotti basati su SQL. Nel 1981 IBM iniziò a vendere alcuni prodotti relazionali e nel 1983 rilasciò DB2, il suo DBMS relazionale diffuso ancor oggi. SQL divenne subito lo standard industriale per i software che utilizzano il modello relazionale.
L'ANSI lo adottò come standard fin dal 1986, senza apportare modifiche sostanziali alla versione inizialmente sviluppata da IBM. Nel 1987 la ISO fece lo stesso. Questa prima versione standard è denominata SQL/86. Negli anni successivi si realizzarono altre versioni, che furono SQL/89, SQL/92 e SQL/2003. Tale processo di standardizzazione mirava alla creazione di un linguaggio che funzionasse su tutti i DBMS (Data Base Management Systems) relazionali, ma questo obiettivo non fu raggiunto. Infatti, i vari produttori implementarono il linguaggio con numerose variazioni e, in pratica, adottarono gli standard ad un livello non superiore al minimo, definito dall'Ansi come Entry Level. (Wikipedia, l'enciclopedia libera, p. SQL)

VERSIONE UTILIZZATA:

MySQL 5.1.41 community server

3.1.5. CSS

DEFINIZIONE:

Il CSS (Cascading Style Sheets) è un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML. Le regole per comporre il CSS sono contenute in un insieme di direttive (Recommendations) emanate a partire dal 1996 dal W3C.

L'introduzione del CSS si è resa necessaria per separare i contenuti dalla formattazione e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine HTML che per gli utenti.

STORIA:

Dietro il semplice acronimo CSS (Cascading Style Sheets - Fogli di stile a cascata) si nasconde uno dei fondamentali linguaggi standard del W3C. La sua storia cammina su binari paralleli rispetto a quelli di HTML, di cui vuole essere l'ideale complemento. Da sempre infatti, nelle intenzioni degli uomini del Consortium, HTML, così come la sua recente evoluzione, XHTML, dovrebbe essere visto semplicemente come un linguaggio strutturale, alieno da qualunque scopo attinente la presentazione di un documento. Per questo obiettivo, ovvero arricchire l'aspetto visuale ed estetico di una pagina, lo strumento designato sono appunto i CSS. L'ideale perseguito da anni si può sintetizzare con una nota espressione: separare il contenuto dalla presentazione. La prima specifica ufficiale di CSS (CSS1) risale al dicembre del 1996. Nel maggio 1998 è stata la volta della seconda versione: CSS2. Niente stravolgimenti, ma molte aggiunte rispetto alla prima. CSS2 non è altro che CSS1 più alcune nuove proprietà, valori di proprietà e definizioni per stili non canonici come quelli rivolti alla stampa o alla definizione di contenuti audio. È attualmente allo stato di Working Draft la nuova specifica CSS3. (Wikipedia, l'enciclopedia libera, p. CSS)

VERSIONE UTILIZZATA:

La versione da noi utilizzata è la CSS2.

3.2. STRUTTURE DATI (BARBARA)

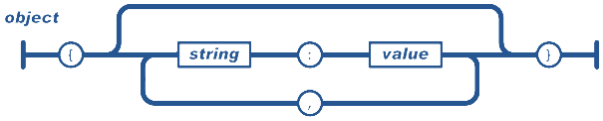
3.2.1. HTTP

HTTP (HyperText Transfer Protocol ossia protocollo di trasferimento di un ipertesto) è il protocollo maggiormente usato come sistema di trasmissione di informazioni sul web.
L'HTTP funziona con un meccanismo di richiesta/risposta: il client invia una richiesta ed il server risponde. Al contrario di altri protocolli una volta che la richiesta è stata soddisfatta le connessioni vengono chiuse e questo è molto utile dal momento che solitamente le pagine richieste contengono link a pagine che risiedono in altri server.
Il messaggio di richiesta è composto di tre parti: riga di richiesta, informazioni aggiuntive (header) e corpo del messaggio(body).
La riga di richiesta è composta da metodo, URI e versione.
L'URI (Uniform Resource Identifier), ossia l'oggetto della richiesta, cambia a seconda dell'API chiamata e del social network che la richiede e la versione attualmente è la 1.1.
Ci sono svariati metodi disponibili, i più utilizzati sono POST, HEADER, GET e DELETE.
GET è il metodo usato per ottenere il contenuto della risorsa, HADER viene usato per ottenere solo i campi dell'header della risorsa che contengono informazioni aggiuntive di utilità per l'applicazione quali ad esempio la data di modifica del file , POST è il metodo usato per inserire informazioni all'URI specificata e DELETE per eliminarle.

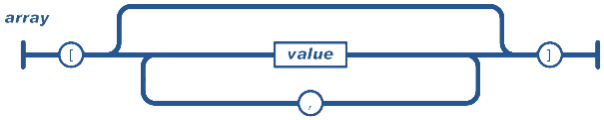
3.2.2. JSON

Json (JavaScript Object Notation) è un tipo di formato per lo scambio di dati. È immediato da capire e per le macchine risulta facile da generare e da analizzare. Si basa su due strutture: un insieme di coppie nome/valore e un elenco ordinato di valori.
Le strutture di dati basilari che lo compongono sono cinque, combinazioni di essere formano JSON.

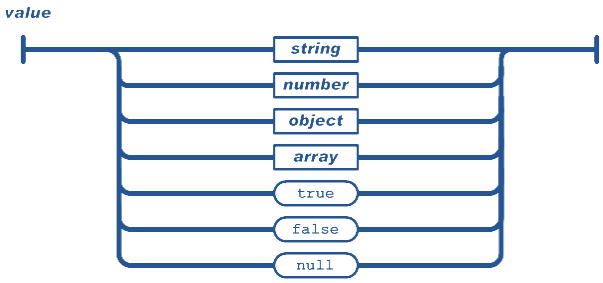
Oggetti, ossia una serie non ordinate di nomi/valori.



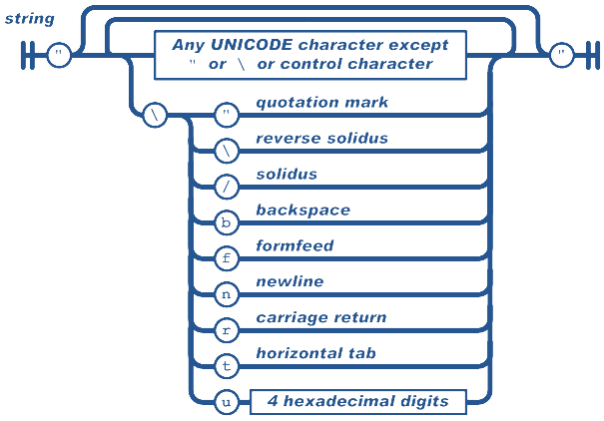
Array, ossia raccolte ordinata di valori.



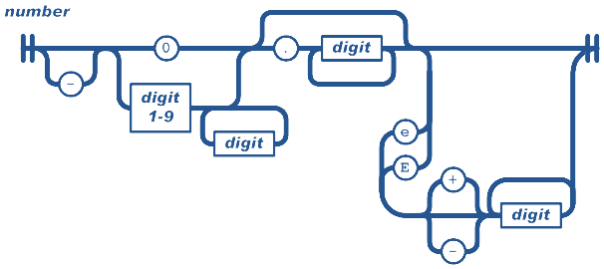
Valori che possono essere stringhe, numeri, caratteri booleani, oggetti o array .



Stringhe, ossia raccoltedi zero o più caratteri Unicode tra virgolette.



Numeri.



Per usare le informazioni contenute in una stringa strutturata in JSON, PHP mette a disposizione un comando molto utile chiamato *json_decode* che riceve in ingresso una stringa JSON e restituisce un'array di valori o uno oggetto. In caso di array per accedere al valore di un campo sarà sufficiente scrivere $nome_array[“nome_campo”]; in caso di oggetto per accedere al valore sarà sufficiente scrivere $nome_oggetto->nome_campo. Per creare una stringa JSON invece varrà fatto il processo contrario, si creerà un array di campi/valori e poi con il comando *json_encode* lo si trasformerà in stringa. (JSON.org)

3.2.3. XML

Un altro formato per lo scambio di dati altrettanto usato è XML (eXtensible Markup Language). XML è un metalinguaggio di markup, ossia un linguaggio marcatore che ha come scopo la definizione di altri linguaggi marcatori. Nel caso di XML questi linguaggi devono essere semplici.
L'XML ha una forte somiglianza con il linguaggio marcatore HTML e per questo motivo per molti risulterà di immediata comprensione. La due grande diversificazioni tra i due linguaggi sono la possibilità per l'XML di creare tag personalizzati (da qui la parola eXstensible ossia estensibile) e l'obbligo di chiudere tutti i tag. Altra cosa che diversifica i due linguaggi è lo scopo, mentre l'HTML definisce una grammatica per la descrizione e la formattazione delle pagine web il secondo è un metalinguaggio utilizzato per creare nuovi linguaggi atti a scrivere documenti strutturati.
Per poter essere correttamente interpretato un documento XML deve essere ben formato: deve obbligatoriamente contenere un prologo, che indica le specifiche del documento quali ad esempio versione e codifica, deve avere un unico elemento radice e tutti i tag devono essere bilanciati ossia opportunamente chiusi e nidificati.
Il primo tag in un documento XML è il tag radice e tutti gli altri elementi devono essere necessariamente nidificati all'interno di esso. I tag sono case sensitive <testo> sarà quindi diverso da <Testo> .

Di seguito la struttura basilare di XML.

```
<radice>
 <elemento>
  <subelemento>..valore...</subelemento>
 </elemento>
</radice>
```

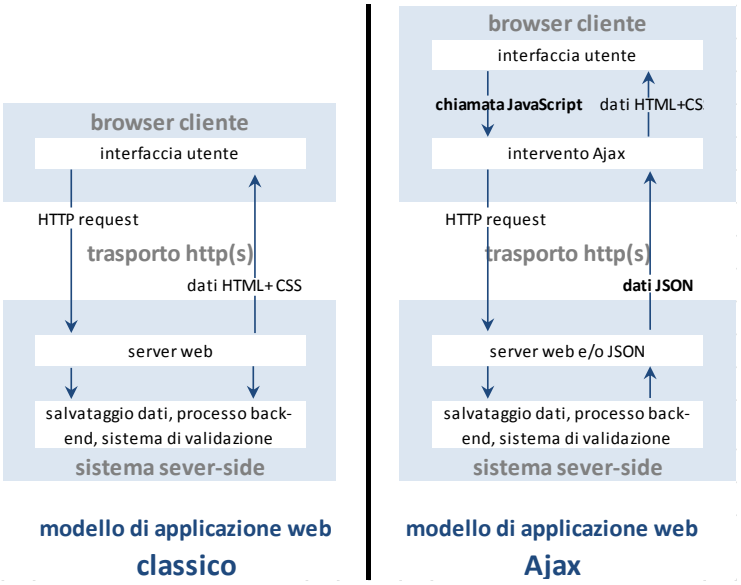
Per ottenere le informazioni contenute nei campi di una stringa strutturata in XML è necessario trasformare prima la stringa in un oggetto e poi trasformare l'oggetto in un array visto che il passaggio diretto non è possibile.
Per ottenere un oggetto da una stringa XML bisogna creare un nuovo oggetto di tipo *SimpleXMLElemen*t passando nel costruttore la stringa ricevuta. Per ottenere un array bisogna poi utilizzare la funzione *xmlobj2arr* da noi scritta all'interno del file *util.php* dandogli come parametro l'oggetto creato che restituirà un'array. Non si è potuta utilizzare la funzione insita in php che permette di passare da oggetti ad array in quanto essa ha dei problemi nella conversione degli oggetti *simpleXMLElement*.
La stringa da inviare verrà invece creata manualmente senza usare funzioni.

3.3. METODOLOGIE E PARADIGMI DI PROGRAMMAZIONE (MARCO)

3.3.1. AJAX

Seppur AJAX non sia un vero e proprio linguaggio di programmazione rimane un concetto degno di nota e di approfondimento.
AJAX è una "metodologia" di programmazione di pagine web che si avvale di tutti i linguaggi sopra illustrati al fine di rendere la fruizione di contenuti web da parte dell'utente veloci e fluidi, eliminando, o perlomeno riducendo, i tempi morti dati dai caricamenti delle pagine.

AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo per la realizzazione di applicazioni web interattive (Rich Internet Application).



Lo sviluppo di applicazioni HTML con AJAX si basa su uno scambio di dati in background fra browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente. AJAX è asincrono nel senso che i dati extra sono richiesti al server e caricati in background senza interferire con il comportamento della pagina esistente. Normalmente le funzioni richiamate sono scritte con il linguaggio JavaScript. Tuttavia, e a dispetto del nome, l'uso di JavaScript e di XML non è obbligatorio, come non è necessario che le richieste di caricamento debbano essere necessariamente asincrone.

AJAX è una tecnica multi-piattaforma utilizzabile su molti sistemi operativi, architetture informatiche e browser web, ed esistono numerose implementazioni open source di librerie e framework.

La tecnica Ajax utilizza una combinazione di:

- HTML (o XHTML) e CSS per il markup e lo stile;
- DOM (Document Object Model) manipolato attraverso un linguaggio ECMAScript come JavaScript o JScript per mostrare le informazioni ed interagirvi;
- l'oggetto Javascript XMLHttpRequest per l'interscambio asincrono dei dati tra il browser dell'utente e il web server. In alcuni framework Ajax e in certe situazioni può essere usato un oggetto *Iframe* invece di *XMLHttpRequest* per scambiare i dati con il server e, in altre implementazioni, tag <script> aggiunti dinamicamente (JSON);

In genere viene usato XML come formato di scambio dei dati, anche se di fatto qualunque formato può essere utilizzato, incluso testo semplice, HTML preformattato, JSON e perfino EBML. Questi file sono solitamente generati dinamicamente da script lato server (PHP).

Le applicazioni web che usano Ajax richiedono browser che supportano le tecnologie necessarie (quelle dell'elenco sopra).
Questi browser includono: Mozilla, Firefox, Opera, Konqueror, Safari, Internet Explorer e Chrome.

3.3.2. OAUTH

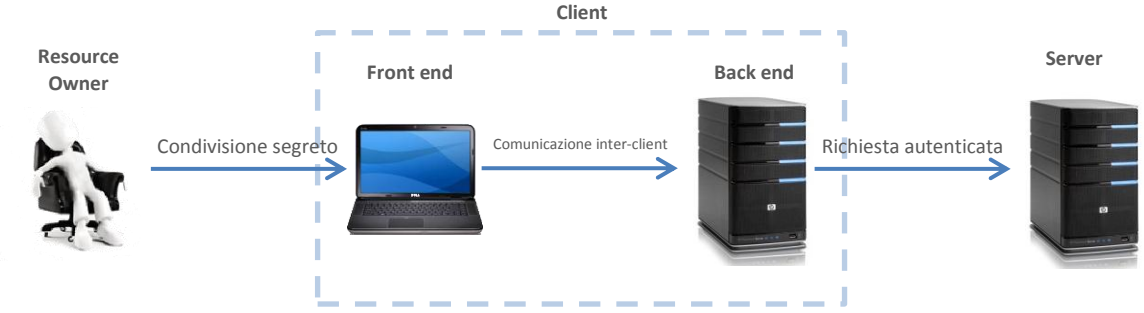
Nel tradizionale modello di autenticazione client/server l'utente utilizza le proprie credenziali per accedere a risorse conservate sul server.

Una risorsa protetta è una risorsa memorizzata o fornita dal server che richiede un'autorizzazione per l'accesso. Le risorse protette sono possedute e/o controllate dal *resource owner.* Le risorse protette possono essere di due tipi: dati (foto, documenti, contatti) o servizi (inviare un messaggio, caricare una foto, trasferire del denaro).
Oauth può essere usato con diversi protocolli di trasporto ma ha una definizione formale solo per le risorse HTTP(S).

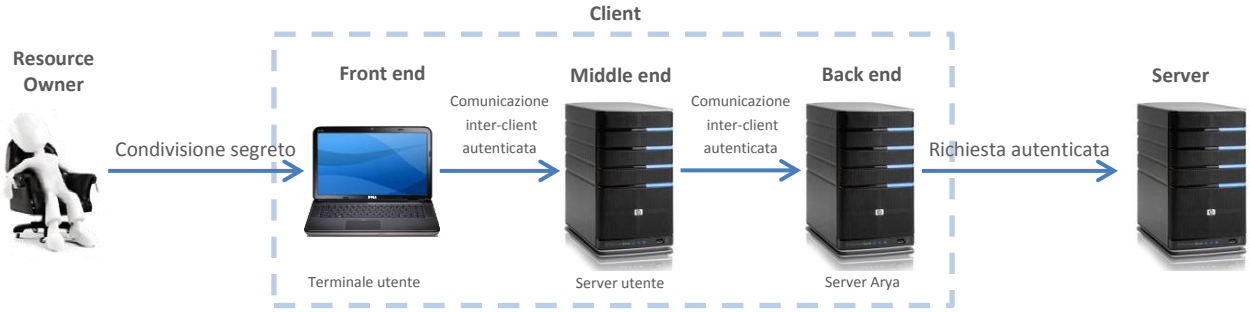
Nel modello di autenticazione oauth il client non è il proprietario delle risorse ma agisce per suo conto, accedendo a dati conservati sul server di proprietà del *resource owner*.
Questa è chiamata autorizzazione a 2 gambe (2 legged)

Per permettere al client di accedere a queste risorse il *resource owner* deve prima garantirgli il permesso. Questo è espresso sotto forma di un *token*, e di un *secret* corrispondente, senza che le credenziali private di accesso del proprietario vengano divulgate. Questo permette inoltre di creare dei token limitati nel tempo o nelle funzionalità, e che possono essere revocati in qualsiasi momento dal *resource owner*.

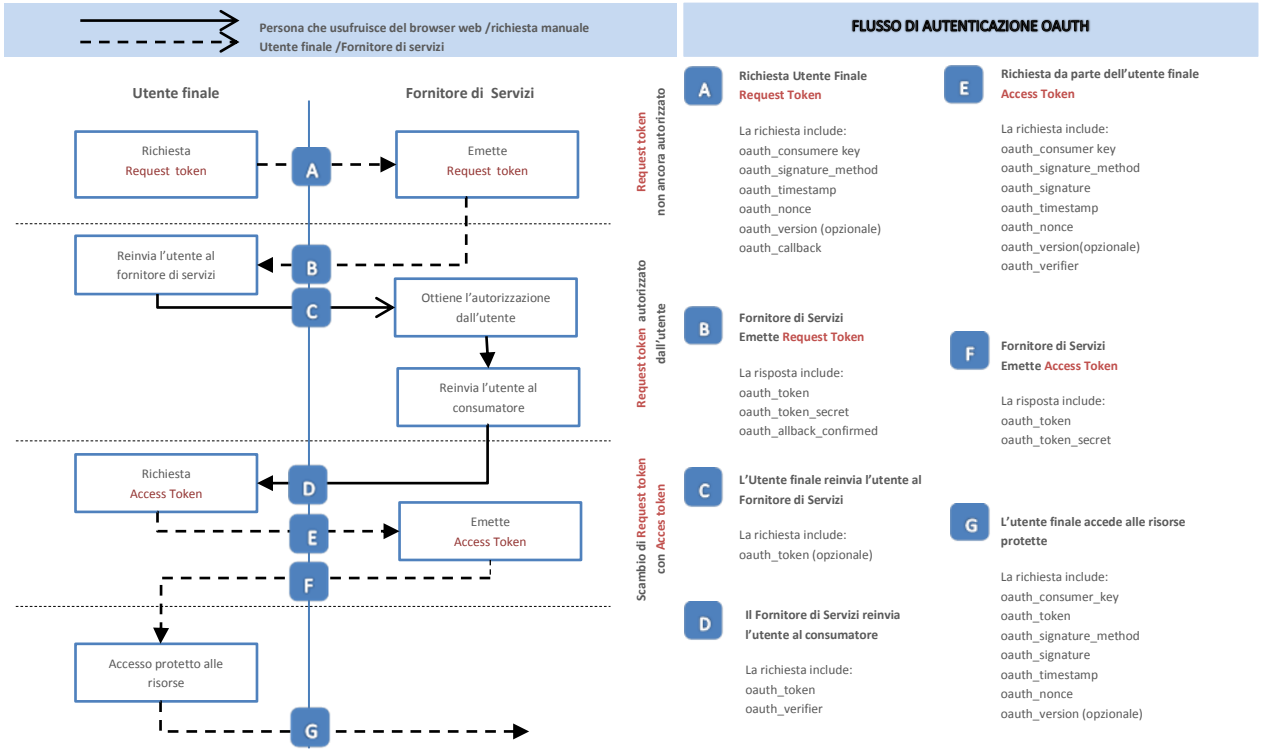
I client web-based in più hanno una divisione tra front-end (interfaccia) e back-end (elaborazione), anche se dal punto di vista logico questa soluzione non si differenzia dalla precedente, in quanto i due elementi agiscono come una singola entità e sempre per conto del titolare delle risorse.
Questa soluzione è chiamata autorizzazione a 3 gambe (3 legged).



Arya introduce un passaggio aggiuntivo alla versione precedente scomponendo ulteriormente l'oggetto client. Se nel caso precedente era il front-end ad utilizzare le risorse protette offerte dal server del social network, in questo caso invece è il middle-end a sfruttarle.
La comunicazione principale avviene quindi tra middle-end (server dell'utente) e back-end (server di arya) rispondendo agli ordini che il *resource owner* impartisce attraverso il front-end.
Nello specifico si sfruttano i soli servizi di pubblicazione offerti dal server al fine di inserire nella pagina utente del *resource owner* contenuti presenti nel middle-end.



Per quanto riguarda il processo di autenticazione oauth, dal punto di vista logico nemmeno questa soluzione si discosta dalle due precedenti poiché i tre componenti del client si comportano come un'unica entità che permette il collegamento tra il proprietario delle risorse e il fornitore di servizi.



In OAuth vi sono 3 tipi di credenziali

- Credenziali del client (consumer key and secret)
- Credenziali temporanee (request token and secret)
- Credenziali del token (access token and secret)

Le credenziali del client sono utilizzate per identificare il client. Permettono al server di raccogliere informazioni sui client che utilizzano i suoi servizi o per fornire al *resource owner* più informazioni sul client che richiede l'accesso alle risorse da lui controllate.

Le credenziali temporanee servono al client ed al server per istituire la richiesta di autorizzazione da fornire poi al *resource owner.*

Le credenziali del *token* sono utilizzate in sostituzione del nome utente e della password del *resource owner.* Il possessore, invece di condividere i suoi dati d'accesso con il client, autorizza il server a rilasciare questa classe speciale di credenziali al client. Esse rappresentano quindi il consenso di utilizzare le proprie risorse dato dal *resource owner* al client. Queste credenziali possono fornire un accesso limitato nel tempo oppure ad un limitato numero di risorse protette.

3.4. API UTILIZZATE (BARBARA)

3.4.1. STRUTTURA DI UN SOCIAL NETWORK

Al cuore di un Social Network vi è il "Grafo sociale" ovvero la rappresentazione informatica delle persone, viste come i nodi del grafo, e delle loro interconnessioni. Per accedere alle informazioni contenute nel grafo si usano strutture chiamate API.
API (Application Programming Interface) è l'acronimo che descrive l'insieme di istruzioni prestrutturate disponibili al programmatore tali da formare set di strumenti specifici per un determinato scopo. È un metodo per ottenere un'astrazione, che può essere tra hardware e programmatore o tra software di basso livello e software di alto livello.

Più specificatamente le API che riguardano il grafo (graph api) sono l'insieme di istruzioni utilizzate dalle web application per accedere alle informazioni contenute nel grafo sociale. Nel loro insieme forniscono una visione semplice e coerente del grafo rappresentando in maniera uniforme i nodi (e le loro caratteristiche) e le connessioni tra loro. Le API dei social network che vedremo usano un'architettura REST.

REST (REpresentational State Transfer) è un tipo di architettura software per i sistemi di ipertesto distribuiti, questo termine fu coniato nel 2000 da Roy Fielding uno dei principali autori delle specifiche HTTP.
È nell'uso comune pensare che un indirizzo internet possa indirizzare solo pagine web.
Il concetto fondamentale in REST è l'esistenza di risorse, intese come fonti di informazioni, a cui si può accedere tramite un identificatore globale, un URI(Universal Resource Identificator), ossia una stringa che identifica univocamente una risorsa generica come un indirizzo web ma anche un documento, un immagine, un file etc.
Gli URI rendono disponibili le risorse secondo vari protocolli, il più comunemente usato è l'HTTP.

Un'applicazione può interagire con una risorsa conoscendo l'identificatore della risorsa, ossia l'URI, e il metodo richiesto. ARYA userà POST per inserire informazioni nell'URI specificata e GET per restituire il contenuto della risorsa indicata dall'URI .L'applicazione deve in ogni caso conoscere il formato della risorsa, tipicamente HTML, XML o JSON.

I formati delle risorse consentono di dare una serializzazione di un oggetto sottoforma di stringa.
I formati usati dai social network interessati da ARYA usano solo i formati JSON e XML.
Alcuni supportano entrambi i formati, e la scelta deve venire opportunamente indicata nella richiesta, altri solo uno delle due.

In un social network ogni nodo ed ogni connessione hanno un ID unico, in genere una stringa alfanumerica; si può accedervi tramite l'URI che li riferisce e ricevendo, o inserendo, l'informazione in formato JSON o XML.

Per utilizzare alcune API che consentono l'accesso ad informazioni "riservate" è necessario fornire il permesso da parte dell'utente, ovvero l'*access token* ottenuto durante la fase di autorizzazione Oauth vista prima.

3.4.2. FACEBOOK

Come già detto ogni nodo, ossia ogni utente, di Facebook ha un ID unico. Su Facebook si può, alternativamente, usare l'username come ID.
La pagina da interrogare per accedere alle proprietà di ogni oggetto è *https://graph.facebook.com/ID* e si riceve una rappresentazione dell'oggetto in formato JSON.
Si possono inoltre esaminare le connessioni tra gli oggetti usando la struttura URL *https://graph.facebook.com/ID/CONNECTION_TYPE.*

È possibile inserire informazioni sul grafo di Facebook emettendo il metodo HTTP POST all' URI appropriato e fornendo delle credenziali di accesso valide per la richiesta in corso. Per poter pubblicare qualsiasi cosa sul grafo di Facebook è necessario un permesso esteso di pubblicazione chiamato *publish_stream* da richiedere in fase di richiesta di *access token*.

Molte operazioni di scrittura richiedono permessi estesi ulteriori al *publish_stream* che, eventualmente, verranno indicati nella spiegazione specifica delle API utilizzate.

Le api di inserimento utilizzate da ARYA per Facebook sono quelle necessarie ad inserire note, eventi, album e foto.

Argomento importante che interessa solo Facebook è l'amministrazione delle pagine. Ogni utente infatti ha la possibilità di creare ed amministrare più pagine oltre a suo profilo.

Qualunque di queste API può essere eseguita anche per conto di una qualunque delle pagine amministrate dall'utente, previa concessione del permesso esteso *manage_pages* in fase di autenticazione e successiva richiesta al server di un access token adatto.

PROFILO

L'oggetto che contiene le informazioni sull'utente è presente all'indirizzo *https://graph.facebook.com/me.*

Facendo una richiesta HTTP GET viene restituito un JSON così formato:

```
{
"id": "id utente",
"name": "nome completo utente",
"first_name": "nome utente",
"last_name": "cognome utente",
"link": "http://www.facebook.com/ID(username)",
"about": "frase significativa per l'utente inserita dallo stesso sulla propria bacheca",
"birthday": "gg/mm/aa",
"bio": " breve descrizione di se stesso inserita dall'utente sul proprio profilo ",
"quotes": "motto ",
"education":
  [
  1 :{
     "school":
       {
       "id": "id scuola 1",
       "name": "nome scuola 1"
       },
     "type": "tipo di scuola 1"
     },
  ...
  ],
"gender": "sesso dell'utente",
"interested_in":
  [
  1:{
     "inclinazione sessuale dell'utente"
     },
  ...
  ],
"relationship_status": "stato sentimentale dell'utente",
"significant_other":  # campo presente  solo se l'utente è impegnato sentimentalmente
  {
  "name": "nome del compagno",
  "id": "id del compagno"
  },
"religion": "inclinazione religiosa utente",
"political": "inclinazione politica utente",
"timezone": fuso orario utente,
"locale": "sigla stato di appartenenza utente",
"verified": true,
"updated_time": "data ultima modifica da parte dell'utente"
}
```

Questi sono solo alcuni dei parametri presenti nel profilo, quelli non presenti sono parametri con corpo nullo.

NOTA

La richiesta di inserimento di una nuova nota viene fatta con il metodo HTTP POST all'URI *http://graph.facebook.com/PROFILE_ID/notes.*

Per farlo bisognerà creare una stringa serializzata in formato JSON, creata con una struttura specifica, riportata qui sotto, richiesta dal sito stesso.

```
{
"access_token": "access token precedentemente richiesto",
" subject":"titolo del messaggio da inserire",
"message":"corpo del messaggio da inserire"
}
```

Il corpo della nota consente di inserire i principali tag HTML ed in questo modo consente di inserire anche foto e link direttamente nella nota.
I link permessi sono: <p> <a> <img> <h1> <i> <b> <u> <h2> <h3> <h4> <h5>

  . Verrà fatta automaticamente dall'applicazione, in fase di inserimento del corpo del messaggio, una pulizia degli eventuali tag non permessi.

Il vantaggio delle note è la possibilità di inserire più di un link nella stessa nota e foto strutturate nel modo che più si preferisce ma soprattutto la possibilità di inserire note con un messaggio pressoché infinito.

EVENTO

Per creare un evento oltre al permesso *publish_stream* è anche necessario il premesso *create_event.*
L'URI di interesse a cui inoltrare la richiesta è
http://graph.facebook.com/PROFILE_ID/events.

La stringa serializzata in formato JSON deve essere strutturata in modo adeguato, avrà dei campi obbligatori quali il titolo, la data di inizio dell'evento e la data di fine, oltre all'access token, e conterrà inoltre altri campi facoltativi.

```
{
 "access_token": "access token precedentemente richiesto",
 "name": "nome dell'evento",
 "description": "Descrizione evento*",
"start_time": "2010-03-14T14:00:00", #data inizio, unix timestamp
"end_time": "2010-03-14T17:30:00", #data fine, unix timestamp
"location": "luogo dell'evento",
"venue":
   {
   "street": "via"
   "city": "città",
   "state": "stato",
   "country": "stato",
   "latitude": latitudine,
   "longitude": longitudine
   },
"privacy": "OPEN", #le possibilità sono open o close, open è un evento aperto a tutti, close solo agli amici
}
```

ALBUM

Per creare un album non sono necessari altri permessi diversi da *publish_stream.*
L'URI in cui vogliamo inserire l'oggetto è
http://graph.facebook.com/PROFILE_ID/albums.

L'oggetto, serializzato in JSON per l'invio, sarà così strutturato.

```
{
"access_token": "access token precedentemente richiesto",
" name":"nome dell'album",
"message":"piccola descrizione dell'album"
}
```


È importante notare che nella creazione dell'album non vengono menzionate foto; l'utilizzo di questo comando creerà infatti soltanto la struttura in cui poi verranno effettivamente inserire le foto con l'API apposita.

FOTO

L'URI per l'inserimento di foto in un album già precedentemente creato è
http://graph.facebook.com/ALBUM_ID/photos.
È quindi ovvio che tutte le foto appartenenti ad uno stesso album avranno ALBUM_ID uguale.

```
{
"access_token": "access token precedentemente richiesto",
"message": "testo che verrà inserito come commento alla foto",
"nome_foto.ext": "@/percorso_foto/nome_foto.ext",
 "height": altezza foto in pixel,
 "width": larghezza foto in pixel
}
```

Le foto inserite nell'album appariranno nell'oggetto Album come connessioni; per controllare le foto effettivamente inserite basterà richiedere le informazioni sull'Album all'indirizzo *http://graph.facebook.com/ALBUM_ID*; ci verrà restituito una rappresentazione dell' oggetto in formato JSON con inserite anche le connessioni a tutte le foto inserite.

IMPERSONIFICAZIONE PAGINE

Vista la possibilità di avere più pagine oltre al profilo può essere utile poter operare per conto di esse. Per fare questo è necessario conoscere *l'access token* di ognuna delle pagine per cui si desidera operare poiché esse risultano per il server come entità distinte e non condividono l'*access token* dell'amministratore.

Mandando una richiesta HTTP GET all'url *http://graph.facebook.com/u_id/accounts* ci verrà restituita, in formato JSON, la lista delle pagine attualmente collegate all'id specificato e i relativi *access token* necessari ad operare per loro conto.


```
{
"data" :
  [
  0:
    {
     "name": "nome Pagina 1"
     "category": "categoria pagina 1"
     "id": id pagina 1
     "access_token" :access token pagina1
    },

  1:
    {
    "name": "nome Pagina 2"
    "category": "categoria pagina 2"
     "id": id pagina 2
     "access_token" :access token pagina2
    }
  ]
}
```

3.4.3. MySpace

L'URI che contiene le informazioni sugli oggetti di MySpace ha la seguente struttura basilare: *http://api.myspace.com/1.0/{category}/{personId}/{selector}* .
A questo inditizzo è possibile inserire e/o ottenere informazioni in formato JSON o XML; il formato JSON è il predefinito, mentre, per utilizzare il formato XML, è necessario inserire *?format=xml* alla fine dell'URI richiesta.
La categoria, *{category},* specifica il tipo di oggetto presente a quell'URI.
{personId} è l'id identificativo della persona a cui appartengono le informazioni, può essere un intero oppure *@me* che riferisce l'utente corrente.
{selector} individua il gruppo di individui per i quali si dedidera sapere le informazioni, i valori possibili sono @self che individua l'unica persona decisa in *{personId}, @all/@friends* che sono sinonimi e identificano tutte le persone amiche di *{personId}.*

Le API utilizzate da ARYA per MySpace sono quelle necessarie a prelevare le informazioni sul profilo dell'utente, quella necessaria per inserire link o eventi e le due necessarie ad inserire album e foto.

L'*access token* necessario per avere l'autorizzazione a inserire e richiedere informazioni su MySpace è inserito nell'header durante la richiesta HTTP GET o POST.

Profilo

Le api del profilo consentono di ottenere informazioni su persone e amici.
La richiesta HTTP GET va fatta all'URI *http://api.myspace.com/1.0/people/@me/@self* e le informazioni sul profilo in formato JSON sono strutturate come sotto.

```
{
"itemsPerPage" : numero di elementi presenti nella risposta
"numOmittedEntries" : numero di elementi ommessi ovvero presenti in altre pagine
"person" :
  {
  "displayName" : "nome visualizzato nel profilo"
   "hasApp" : true #false in caso l'utente non possieda applicazioni
   "id" : "id utente"
   "msUserType" : RegularUser #tipo di utente
  "name" :
    {
     "familyName" : Massarotto
     "givenName" : Marco
    }
  "profileUrl" :"link al profilo dell'utente"
  "thumbnailUrl": "link all'immagine dell'utente"
  }
"startIndex" : 0
"totalResults" : 1
}
```

LINK ED EVENTI

MySpace consente di strutturare personalmente le risorse da inserire. Questo si può fare con l'uso di Template ossia modelli in cui vengono inseriti contenuti diversi mantenendo invariata la struttura.
Le strutture da usare sarebbero due, una per i messaggi e una per gli eventi.
Purtroppo nel profilo può essere presente un solo contenuto per template per volta, l'inserimento di un una nuova notizia che utilizzi lo stesso template provoca la cancellazione del precedente. È stato necessario cercare dunque un altro modo di inserire le notizie mantenendo comunque una certa qualità nell'informazione.
Un aggiornamento recente ha consentito, in MySpace, l'inserimento di link simili a quelli inseriti negli altri social network. L'aggiornamento non avrebbe una struttura apposita per ogni tipo di notizia come invece era con i templates ma in questo modo è consentito l'inserimento di più notizie.

L'URI a cui inviare la stringa JSON strutturata nel modo riportato qui sotto è il seguente *http://api.myspace.com/1.0/statusmood/@me/@self*.

```
{
"sharedLinkInfo:
  {
  "description": "corpo del messaggio",
  "linkUrl": "link al messaggio originale",
  "thumbnailUrl": "link alla foto",
  "title":"titolo"
  },
"source":
  {
  "imageUrl": "link al logo dell'applicazione che inserisce il messaggio",
  "name": "nome applicazione"
  }
}
```

I messaggi e gli eventi differiranno solo per il contenuto inserito nel campo "*description*".
La presenza del campo *"ThrumbnailUrl"* è opzionale, in caso di mancata foto verrà quindi ommesso.

ALBUM

L'URI di base che consente di accedere e modificare le informazioni sugli album è la seguente *http://api.myspace.com/1.0/albums*.
Come avviene su Facebook la creazione dell'album creerà solo la "scatola" in cui le foto verranno poi inserite. Di fatto verrà creato un oggetto Album con un ID unico e delle informazioni che lo descrivono, poi le foto verranno inserite come oggetti con ID unici e collegate tramite *connessioni* all'album desiderato.

L'URI per l'inserimento dell'oggetto album è quindi la seguente *http://api.myspace.com/1.0/albums/@me/@self* e l'oggetto da inserire è rappresentato dalla stringa JSON strutturata nel modo seguente

```
{"
"caption"=>"titolo",
"mediaItemCount"=>0, #numero di foto presenti nell'album, alla creazione 0
"msPrivacyLevel"=>"everyone" #possibilità di me, friendsonly ed everyone
}
```

FOTO

Come abbiamo visto fin ora per caricare le foto negli altri social network è sufficiente indicare nel messaggio l'URI dell'immagine desiderata, questo è sufficiente poiché il server del social network va a reperirla ed a scaricarla in automatico.
MySpace, per l'inserimento delle foto negli album, usa un sistema di tipo push secondo il quale il messaggio da inviare all'URI per caricare la foto nell'album deve avere già all'interno la sequenza binaria dell'immagine insieme all'indicazione del formato in cui è rappresentata.
Sarà compito di ARYA quindi memorizzare lo stream dati, ovvero la sequenza binaria dell'immagine, e includerlo nel messaggio di invio all'URI specifica.
L'URI a cui inviare tutte le informazioni è la seguente
http://api.myspace.com/1.0/mediaItems/@me/@self/albumID

```
{
"type" : " image", #tipo di informazione inserita, in questo caso foto
"caption" : " testo da inserire sotto l'immagine"
}
```

Lo stream dati e l'estensione dell'immagine inviata verranno inseriti nel pacchetto e verranno inviati insieme alla stringa JSON.

3.4.4. TWITTER

Vista la semplicità di Twitter ARYA userà solo due API: quella che richiama il profilo dell'Utente e quella di inserimento dei Tweet.

PROFILO

L'URI a cui mandare la richiesta sarà strutturata in questo modo: *http://api.twitter.com/version/account/verify_credentials.format*.
Twitter supporta i formati JSON e XML. Il formato scelto verrà inserito al posto della parola chiave *format.* Per ottenere le informazioni sull'Utente dovremo quindi inviare la richiesta HTTP GET all'indirizzo *http://api.twitter.com/1/account/verify_credentials.json* ottenendo il seguente risultato


```
{
"name": "nome utente",
#inizio impostazioni dello stile del profilo utente
...
#fine impostazioni dello stile del profilo utente
"profile_image_url": "percorso immagine del profilo",
"location": "città di nascita dell'utente",
created_at": "Sat Feb 17 20:49:54 +0000 2007",
"url": "eventuale link al sito esterno dell' utente",
"description": "Breve descrizione dell'utente.",
"text": "Ultimo tweet inserito",
"lang": "en", #lingua del profilo
"time_zone": "fuso orario usato",
# inizio informazioni ininfluenti per il nostro  studio
"notifications": false,
"favourites_count": 95,
"contributors_enabled": false,
"utc_offset": -28800,
"id": 777925,
"protected": false,
"followers_count": 1025,
"verified": false,
"geo_enabled": true,
"friends_count": 294,
"statuses_count": 2924,
"status":
  {
  "coordinates":
    {
    "coordinates":
      [
      -122.40075845,
      37.78264991
      ],
    "type": "Point"
    },
  "favorited": false,
  "created_at": "Tue Jun 22 18:17:48 +0000 2010",
  "truncated": false,
  "contributors": null,
  "id": 16789004997,
  "geo":
    {
    "coordinates":
      [
      37.78264991,
      122.40075845
      ],
    "type": "Point"
    },
  "in_reply_to_user_id": null,
  "place": null,
  "source": "url sorgente",
  "in_reply_to_screen_name": null,
  "in_reply_to_status_id": null
  },
"screen_name": "themattharris",
"following": false
}
```

TWEET

L'URI di inserimento in twitter è strutturata in modo simile a quella di interrogazione *http://api.twitter.com/version/statuses/update.format* .

Per inserire un oggetto nel grafo dovremo inviare una richiesta HTTP POST all'URI *http://api.twitter.com/1/statuses/update.json* così strutturata:


```
{
"access_token": "access token precedentemente richiesto",
"status": "corpo del messaggio"
}
```

Si vede subito la semplicità di questo social network ma anche la limitatezza; infatti nel corpo del messaggio verrà inserito, prima del messaggio vero e proprio, il link al messaggio originale e nel caso di link più lungo di 140 caratteri (dimensione massima del messaggio) il Tweet inserito risulterà inutile ed incomprensibile.

3.4.5. YAHOO!PULSE

Per capire in che formato ci arriverà la risposta bisognerà analizzare l'header; ricordiamo che la risposta HTTP ha un header e un body, dentro il body sarà contenuto la stringa di nostro interesse con le informazioni richieste. Dentro il body le informazioni potranno essere in formato JSON o XML. La risposta XML è di default ed è possibile passare al formato JSON aggiungendo semplicemente *?format=json* in coda all'URL di richiesta.

PROFILO

L'oggetto che contiene le informazioni sull'utente si può richiedere all'indirizzo *http://social.yahooapis.com/v1/user/ID_UTENTE /profile?format=json*.

```
{
"guid" : "id dell'utente",
"birthYear" : anno di nascita dell'utente,
"birthdate" : data di nascita dell'utente,
"created" : data di creazione del profilo,
"displayAge" : anni utente,
"emails" :
  [
  0 :{
     "handle" : "e-mail utente",
     "id" : id e-mail,
     "type" : "tipo e-mail"
     },
  ...
  ]
"familyName" :"cognome utente",
"gender" :"sesso utente",
"givenName" :"nome utente",
"image" :
  {
  "height" :altezza immagine profilo,
  "imageUrl" :"link all'immagine del profilo",
  "size" :dimensione immagine profilo,
  "width" :"larghezza immagine prifilo"
  }
"interests" : #lista di array contenente tutti gli interessi dell'utente
  [
  0 :{
     "declaredInterests" : "nome interesse"
     "interestCategory" :"categoria di appartenenza dell'interesse"
     },
  ...
  ]
"lang" :"lingua utente"
"location" :"provenienza utente"
"memberSince" :"data di iscrizione dell'utente al social"
"nickname" :"soprannome"
"profileUrl" :"link al profilo dell'utente"
"status" :
  {
  "lastStatusModified" :data ultima modifica dello stato
  "linkTo" :"eventuale link presente nello stato"
  "message" :"stato utente"
  }
"updated" :"data ultimo aggiornamento profilo"
"isConnected" :"false"#indica se l'utente è collegato o meno
}
```

LINK, EVENTI E FOTO

L'URI da utilizzare per inserire, modificare o cancellare un aggiornamento di stato è la seguente *http://social.yahooapis.com/v1/user/{guid}/updates/{source}/{suid}?format=json.*
Le variabili tra parentesi graffe non sono di immediata comprensione come quelle viste in altri social network; *{guid}* Identifica l'utente che ha generato l'operazione, *{source}* identifica l'applicazione o il sito di Yahoo! che ha creato l'operazione, *{suid}* identifica in modo univoco l'operazione effettuata. Nella stringa da inviare sarà presente un campo *type* che indicherà il tipo di aggiornamento inserito. Le scelte possibili sono molteplici e per ogni tipo, al sito apposito, è presente una lista di campi facoltativi e di campi obbligatori da inserire nella stringa JSON da inviare. A seconda del tipo indicato nel campo *type* l'aggiornamento sarà visualizzato in maniera differente.
I due tipi di interesse per ARYA sono *eventcreate* e *share*; il primo visualizza la notizia indicando prima del nome dell'evento una scritta "ha partecipato a" mentre il secondo prima del titolo avrà la scritta "ha condiviso". I due tipi di inserimento hanno gli stessi campi obbligatori e quindi il messaggio da inviare avrà la stessa struttura di base in entrambi I casi. Le differenze quindi sono minime, sarà l'applicazione ARYA a strutturare il corpo del messaggio in maniera differente a seconda del fatto che il messaggio originale sia evento o notizia. Non è presente un tipo di aggiornamento che ci consenta di inserire album, è consentito in compenso di inserire una foto nell'aggiornamento. Mostreremo quindi l'album come notizia con il link che reindirizza l'utente all'album originale e il messaggio che descrive l'album, la foto inserita sarà la prima presente nell'album. È prevista una foto obbligatoria anche in caso di inserimento di notizia o evento, in caso di mancata foto nel messaggio originale verrà inserita una foto con il logo dell'azienda che ha creato la notizia.
Ecco quindi la stringa strutturata in JSON

```
{
"updates":
  [
  1:
    {
    "class": "app", #la classe a cui appartiene l'aggiornamento inserito. L'unico valore possibile è "app"
    "collectionType": "guid" , #l'unico valore possibile è "guid"
    "description": "corpo del messaggio",
    "suid": "Arya - ".time(),     #costrutto base per costruire il suid, identifica UNIVOCAMENTE l'aggiornamento
                                  appena effettuato
    "link": "link al messaggio originale",
    "source": "APP.APP_ID" , #identifica in modo univoco l'applicazione che ha effettuato l'operazione
    "pubDate": "data di pubblicazione", #un semplice timestamp
    "title": "titolo" ,
    "type": "tipo di aggiornamento", #eventcreate, share ma può essere anche ad esempio songplay o comment
    "collectionID": "ID utente" ,
    "imgURL": "link alla foto inserita",
    "imgWidth": "larghezza foto inserita" ,
    "imgHeight": "altezza foto inserita"
    } ,
  ...
  ]
}
```

3.4.6. LINKEDIN

LinkedIn ha un limite di cinque utilizzi al giorno per ogni API.

In LinkedIn le API si suddividono in due gruppi fondamentali: profile API e share API.

PROFILO

Le API del profilo (profile API) contengono le informazioni sul profilo di un membro di LinkedIn. Si può usare questa chiamata per richiedere due tipi di versioni del profilo: standard e pubblico.
Il profilo standard ha tutte le informazioni che il richiedente è in grado di vedere, che non sono cioè bloccate dalla privacy del richiesto.
Il profilo pubblico ritorna solo le informazioni decise dal proprietario del profilo richiesto che sono visibili a tutti.

Verrà richiesto il profilo standard, facendo la richiesta all'URI *http://api.linkedin.com/v1/people/~* , dove ~ ha la funzione che *@me* ha in MySpace e indica quindi l'utente corrente, otterremo una risposta codificata in XML, l'unico formato supportato da LinkedIn, strutturata in questo modo:

```
<person>
 <id>
 <first-name />
 <last-name />
 <headline>
 <location>
  <name>
  <country>
   <code>
  </country>
 </location>
 <industry>
 <distance>
 <relation-to-viewer>
  <distance>
 </relation-to-viewer>
 <num-recommenders>
 <current-status>
 <current-status-timestamp>
 <connections total="" >
 <summary/>
 <positions total="">
  <position>
   <id>
   <title>
   <summary>
   <start-date>
    <year>
    <month>
   </start-date>
   <is-current>
   <company>
    <name>
   </company>
  </position>
 <educations total="">
  <education>
   <id>
   <school-name>
   <degree>
   <start-date>
    <year>
   </start-date>
   <end-date>
    <year>
   </end-date>
  </education>
 </educations>
 <member-url-resources>
  <member-url>
   <url>
   <name>
  </member-url>
 <api-standard-profile-request>
  <url>
  <headers>
   <http-header>
    <name>
    <value>
   </http-header>
  </headers>
 </api-standard-profile-request>
 <site-standard-profile-request>
  <url>
 </site-standard-profile-request>
 <picture-url>
</person>
```

LINK, EVENTI E FOTO

Per pubblicare su LinkedIn si usano le Share API(API di condivisione). Si possono pubblicare brevi messaggi, alla maniere di Twitter, oppure URL con un titolo, un breve messaggio ed una foto opzionale.

Tra le due scelte è stato deciso ovviamente di usare il link sia per la pubblicazione di messaggi che di eventi ed album; in questo modo, vista la limitatezza del messaggio, il link consente di reindirizzare il visitatore al messaggio originale.

Sia il messaggio originale che l'evento originale non sempre contengono una foto ed in quel caso verrà inserita un'immagine di default con il marchio dell'azienda poiché l'inserimento della foto è obbligatoria.
L'album conterrà soltanto la prima immagine presente nell'album e come messaggio il commento all'album mentre l'evento nel messaggio avrà il luogo e la data di svolgimento.

Per condividere qualcosa bisogna codificarlo in XML e pubblicare un post HTTP al seguente URI: *http://api.linkedin.com/v1/people/~/shares.*

Tutti i tipi di inserimento quindi saranno strutturati allo stesso modo:

```
<share>
  <comment>ha pubblicato:</comment>
  <content>
    <title>titolo messaggio</title>
    <submitted-url>link messaggio originale</submitted-url>
    <submitted-image-url>url immagine presentazione</submitted-image-url>
    <description>breve messaggio</description>
  </content>
  <visibility>
    <code>anyone</code> #visibilità messaggio, può essere anyone oppure connection-only
  </visibility>
</share>
```

3.4.7. Google Buzz

Il formato specifico per gli indirizzi delle API di Google Buzz è il seguente:
https://www.googleapis.com/buzz/v1/resourceID?parameters
dove *resourceID* è identificativo di un'attività, commento o profilo personale e *parameters* sono tutti i paramentri da applicare a quella query.

Google buzz supporta il formato JSON purché specificato nell'URI, il formato di default è *atom* che è un formato molto simile all'XML.

Profilo

Si può ottenere il profilo di un utente inviando un HTTP GET opportunamente autenticata al seguente indirizzo *https://www.googleapis.com/buzz/v1/people/userId/@self/alt=json*
Nel nostro caso quindi userId sarà l'utente corrente e possiamo usare *@me/@self come su MySpace*.
La richiesta ci ritornerà JSON, come richiesto nelle specifiche, strutturato in questo modo:

```
{
"data":
  {
  "id": "id dell'utente",
  "displayName": "nome dell'utente",
  "profileUrl": "link al profilo dell'utente",
  "thumbnailUrl": "link ad un immagine di anteprima dell'utente",
  "urls":
    [
    1: {
       "value": "link al profilo dell'utente",
        "type": "profile"
       } ,
    ...
    ],
  "photos":
    [
    1: {
       "value": "link ad un immagine dell'utente",
       "type": "thumbnail" #tipo di immagine, in questo caso anteprima
       }
    ]
  }
}
```

LINK, EVENTI E FOTO

Al momento della realizzazione di ARYA e più particolarmente dell'implementazione delle classi di Google Buzz nel social network era consentito, tramite API, solo l'inserimento di link con un breve messaggio sottostante e foto di anteprima al link. Nel social network infatti era consentito l'inserimento di album con foto ma per fare questo Google Buzz si appoggiava ad un sito esterno invisibile all'utente e impossibile da realizzare vista la mancanza di API appropriate.

Abbiamo quindi decise di inserire Link, Eventi ed Album nello stesso modo fatto fin ora. In questo modo abbiamo dato la miglior visibilità possibile al messaggio con i mezzi a nostra disposizione. Link, Eventi ed Album verranno inseriti con la stessa API di inserimento, l'unica differenza starà nel messaggio. L'Evento nel messaggio avrà il titolo dell'evento, il luogo e la data; il link all'Album conterrà come descrizione la presentazione dell'Album e come foto la prima foto inserita nell'Album. Per ulteriori informazioni l'utente verrà rimandato al messaggio originale tramite il link.
Al contrario di LinkedIn in cui l'inserimento della foto nel link era obbligatoria su Google Buzz l'immagine è facoltativa, verrà quindi inserita l'immagine nel link solo se presente.

La richiesta di inserimento autenticata verrà fatta all'URI *https://www.googleapis.com/buzz/v1/activities/@me/@self/alt=json* ; Il testo rappresentato in JSON deve essere strutturato in questo modo:

```
{
"data":
  {
  "object":
    {
    "type": "note"
    "content": "titolo del messaggio", #testo del link
    "attachments":
      [
      1:{
        "type": "tipo di inserimento", #può essere 'photo' per inserire foto da inserire o 'article' per inserire solo testo
        "title": "titolo della foto",
        "links":
          {
          "enclosure":
            [
            1:{
              "href": "percorso della foto da inserire" ,
              "type": "image/jpeg"
              } ,
            ...
            ],
          "alternate":
            [
            1: {
              "href": "link al messaggio originale" ,
              "type": "text/html"
              } ,
            ...
            ],
          "preview":
            [
            1: {
              "href": "percorso della foto da inserire" ,
              "type": "image/jpeg"
              }
            ]
          }
        }
      ]
    }
  }
}
```

3.5.STRUTTURA DEL SITO(MARCO)

3.5.1. STRUTTURA DELLE CARTELLE (MARCO)

SERVER ARYA (BACK END) :

callback.php	***pagina di ritorna da un autenticazione***
command.php	***pagina di ricezione di comandi da middle-end***
getauth.php	***esporta le credenziali di un utente in account.php***
index.php	***interfaccia amministrativa di arya***
loginbox.css	***definizioni degli stili dell'interfaccia amministrativa***
<base>	
aryaauth.php	***classe per accedere agli utenti di arya***
db.php	***classe per accedere al database***
<config>	
config.php	***configurazioni specifiche di arya (debug, etc..)***
dbconfig.php	***credenziali di accesso al database***
socialconfig.php	***consumer key dei diversi socialnetwork***
<images>	
….	***diverse immagini usate nel sito***
<social>	
buzz.php	***classe per l'utilizzo di google buzz***
facebook.php	***classe per l'utilizzo di facebook***
linkedin.php	***classe per l'utilizzo di linkedin***
myspace.php	***classe per l'utilizzo di myspace***
social.php	***classe astratta di definizione di un social network***
socialutility.php	***classe di supporto***
twitter.php	***classe per l'utilizzo di twitter***
yahoo.php	***classe per l'utilizzo di yahoo***
<oauth>	
buzzoauth.php	***definizione dei parametri oauth di google buzz***
cryptutil.php	***classe di supporto per le operazioni di cifratura***
facebookoauth.php	***classe per l'autenticazione oauth di facebook***
fb_ca_chain_bundle.crt	***certificato di facebook per le connessioni ssl***
googleoauth.php	***definizione dei parametri oauth di google buzz***
liboauth.php	***classe per l'autenticazione oauth 1.0***
linkedinoauth.php	***parametri oauth di linkedin***
myspaceoauth.php	***parametri oauth di myspace***
socialexception.php	***classe per le eccezioni***
twitteroauth.php	***dei parametri oauth di twitter***
util.php	***classe di supporto per operazioni varie***
yahoooauth.php	***parametri oauth di yahoo***

SERVER UTENTE (MIDDLE END) :

account.php	***credenziali di accesso ad arya (nome e chiave dell'utente)***
arya-connect.php	***invia comandi e dati al back-end***

COMPUTER UTENTE (MIDDLE END) :

nota: I seguenti file sono memorizzati sul middle-end ma vengono effettivamente eseguiti e visualizzati sul front-end attraverso un browser.

arya.htm	***interfaccia grafica di arya***
arya.png	***logo del programma***
default.js	***funzioni ajax***
prototype.js	***libreria javascript***

3.5.2. Classi Rilevanti (Marco)

Classe Social (Front-end)

La classe javascript Social utilizzata nel front-end si occupa della visualizzazione delle informazioni ricevute dal back-end(attraverso il middle-end) e dell'invio di comandi da eseguire al middle-end.

È stata realizzata sfruttando la definizione delle classi offerta da prototype, una famosa libreria javascript che espande le funzionalità di questo linguaggio di programmazione e semplifica l'utilizzo di Ajax e, appunto, delle classi.
Questa classe fa ampio uso delle funzionalità ajax di *prototype Ajax.request* e *Ajax.periodicalupdater. Ajax* request permette di effettuare una richiesta asincrona ad un'altra pagina web specificando inoltre quali parametri passare, con che metodo e come comportarsi alla ricezione della risposta. Ajax periodical updater estende ancora il precedente comando permettendo di creare delle richieste periodiche di informazioni .

Il metodo *initialize()* viene invocato automaticamente alla creazione di un'istanza della classe e riceve come parametro il nome del social che andrà a gestire e visualizzare. Esso si occupa di creare le inizializzare le necessarie variabili d'appoggio e di creare lo "spazio" nell'interfaccia web nel quale verranno inserite le informazioni sui social network. Dopodichè invoca il metodo *update().*

Il metodo update attiva una richiesta periodica (ogni 60 secondi) ad *arya-connect.php* con la richiesta dello stato del social al fine di mantenere la visualizzazione aggiornata allo stato corrente del sistema. Ogni qual volta giunge la risposta per la richiesta viene eseguito il metodo *response().*

Il metodo *response()* riceve come parametro la risposta JSON ricevuta da *arya-connect.php* e si occupa della creazione e del posizionamento all'interno della pagina web delle informazioni ricevute in un formato comprensibile dall'utente. Se il social è già connesso lo visualizzerà nello spazio a lui riservato mostrando inoltre le informazioni per l'account in uso. Se il social invece non è ancora stato autorizzato visualizzerà un pulsante in un'altra posizione alla cui pressione verrà invocato il metodo *logf().*

Logf() permette infatti di inizializzare il processo di autenticazione oauth richiedendo al middle-end e al back-end attraverso esso il link a cui indirizzare l'utente per effettuare l'autorizzazione. Il back-end richiederà un *request_token* al server del social network in questione, genererà e restituirà l'indirizzo per l'autenticazione. Al ricevimento della risposta verrà invocato il metodo *loadpopup(r).*

Loadpopup(r) riceverà il link e lo visualizzerà in una finestra di popup di dimensione specifiche per ogni social network (informazioni ricevute insieme al link).
Dopodichè inizializzerà una chiamata periodica a *popwatch()* ogni 200ms.

Quando l'utente avrà autorizzato il social network nella finestra di popup, esso verrà indirizzato, sempre nella stessa finestra all'indirizzo di callback di arya, dove verrà conclusa automaticamente la procedura di oauth scambiando l'oauth_verifier ricevuto con un *access_token* definitivo. Dopodiche la pagina di callback chiuderà il popup.

Popwatch() non fa altro che monitorare lo stato della finestra di popup verificando se è ancora aperta o se è già stata chiusa. Nel caso sia verificata quest'ultima condizione smetterà la sua ripetizione ed eseguirà il metodo update() forzando così il sistema ad aggiornare lo stato del social network. Se l'autenticazione è andata a buon fine a questo punto il social dovrebbe risultare correttamente connesso e quindi dovrà essere rimosso il pulsante per effettuare la connessione e sostituito con il profilo dell'utente.

Logoutf() invece permette di richiedere al back-end, sempre attraverso il middle-end la rimozione del *token*. essa viene eseguita in seguito alla pressione del pulsante di logout disponibile nello spazio del social network insieme al profilo nel caso esso sia collegato, e previa conferma con un alert.

L'utente a questo punto può selezionare tra tutti i social visualizzati come connessi, ai quali richiedere l'invio della notizia selezionata.
La funzione *select()* imposterà il social come uno dei destinatari.

La funzione *send()* richiederà l'invio della notizia a tutti social destinatari, tramite *Ajax.request* multiple alla pagina *arya-connect.php* fornendo alla pagina l'identificatore univoco della notizia da inviare. Una volta ricevuta risposta verrà eseguito il metodo *receive(r).*

Questo metodo visualizzerà il risultato dell'inserimento indicando se esso è andato a buon fine, se vi sono stati errori risolvibili o se è stato abortito in seguito ad errori gravi.

Loadstart() e *loadstop()* ,rispettivamente, attivano e disattivano, un aiuto grafico che indica l'avanzamento di un operazione di invio e vengono invocati rispettivamente, all'inizio di *send()* e alla fine di *receive().*

```
var social=Class.create(
  {
  initialize: function(soc)

  update: function()

  response: function(r)

  logf: function()

  logoutf: function()

  loadpopup: function(r)

  popwatch: function()

  select: function(a,c,id)

  send: function()

  receive: function(r)

  loadstart: function()

  loadstop: function()

  });
```

CLASSE DB (BACK-END)

La classe astratta db, presente nel file *db.php*, definisce i metodi necessari per utilizzare il database. La connessione al database e le operazioni su di esso sono effettuate usando PDO (PHP Data Object) che permette la rapida migrazione da un server sql ad un altro (mysql, microsoft sql server, postgresql etc).

Nella funzione statica *connect* si inizializza la connessione al database con i dati di accesso presenti nel file *config.php* e la si salva nella variabile protetta $dbh.
Questo tipo di connessione è persistente (se il server sql lo permette) cioè non deve essere effettivamente reinizializzata ad ogni caricamento di pagina ma persiste un collegamento nel motore PHP che decade solo dopo diverso tempo dall'ultimo accesso.

la funzione disconnect invece dealloca la connessione al database. Anche se non viene mai invocata da nessuna classe può essere utile in fase di sviluppo o debugging.

Infine il metodo *fatal_error()* viene utilizzato per segnalare un qualche tipo di errore nell'utilizzo della connessione e per interrompere l'interpretazione da parte di php attraverso il comando die().

Qualunque classe voglia accadere a dati contenuti nel database deve estendere la classe db.

```
Class DB
  {
  // Variabili
  protected static $dbh

  // Metodi
  static function connect()
  function disconnect ()
  protected function fatal_error($msg)
  }
```

CLASSE SOCIAL (BACK-END)

La classe astratta social, contenuta nel file *social.php*, è il cuore di arya. Fornisce le indicazioni su come deve essere scritta una classe che realizzi la connessione ad un certo social network.

La variabile $utente contiene il nome dell'utilizzatore del servizio Arya, mentre $u_id contiene l'id identificativo di quell'utente nel social network in oggetto, a patto che l'autorizzazione sia già stata effettuata. In questo caso $token e $secret contengono l'*access token* e il relativo secret per effettuare le richieste autorizzate al server. La variabile $connection è un puntatore ad una pseudo co "connessione" al server sulla quale eseguire il metodo "api" , con i corretti parametri, al fine di operare sulle risorse presenti sul server.

Nei social che supportano questo servizio, la variabile $refresh, contiene il *refresh token* necessario a richiedere automaticamente al server un nuovo *access token* nel caso il precedente sia scaduto.

La variabile $url invece contiene, in seguito all'esecuzione del metodo OauthCreate(), l'indirizzo a cui reindirizzare l'utente per consentire l'autorizzazione ad arya ad operare per conto dell'utente.

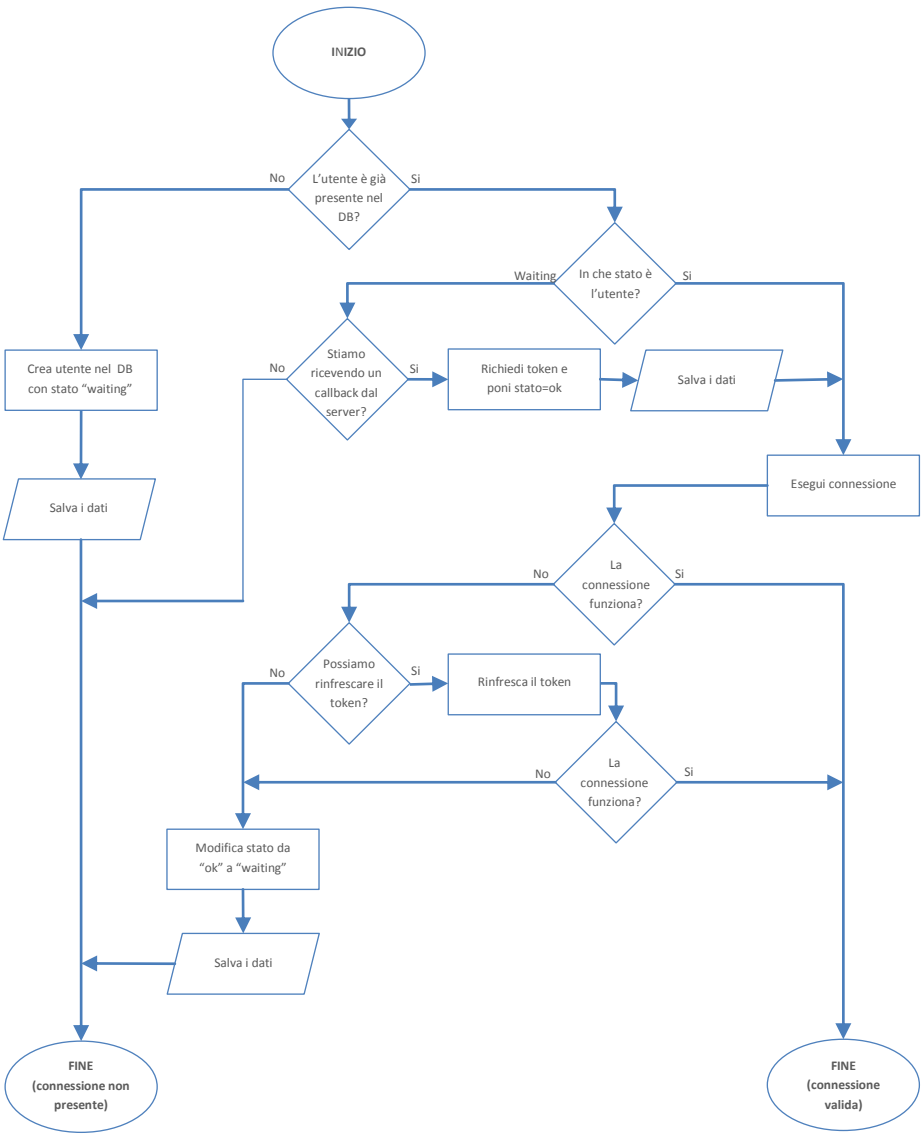
Nel solo facebook $pages contiene l'*access token* speciale necessario ad eseguire operazioni per conto di una pagina amministrata dall'utente e non per conto dell'utente stesso.

Il metodo *__construct*, ovvero il costruttore di un qualunque oggetto che implementi social, inizializza l'oggetto secondo il seguente flow-chart:

```
abstract class social extends db
  {
  //Variabili
  protected $utente='';
  protected $u_id='';
  protected $token='';
  protected $secret='';
  protected $refresh='';
  protected $connection='';
  protected $url='';
  public $status='';
  public $pages;

  //Metodi
  public function __construct($utente)
  protected function savetoken()
  public function removetoken()
  public function getloginurl()

  //metodi astratti
  abstract protected function socialname();
  abstract protected function oauthcreate();
  abstract protected function oauthcallback();
  abstract protected function oauthconnect();
  abstract protected function oauthrefresh();
  abstract protected function checkstatus();
  abstract public function getprofile();
  abstract public function sendnews($message);
  }
```



In questa maniera semplicemente costruendo l'oggetto vengono eseguite tutte le operazioni necessarie a garantirne l'operatività, sia stato esso costruito nella pagina che raccoglie i callback, sia che venga generato per poter eseguire un api.

Questo sistema serve a rendere la classe flessibile e facilmente aggiornabile. Diviene infatti molto semplice aggiungere successivamente nuove api alle classi social network, senza preoccuparsi di come debba essere instaurata la connessione o di come funzioni il sistema Oauth.

SaveToken si occupa di salvare i dati d'accesso nel database per poterli recuperare successivamente mentre, al contrario, *RemoveToken* li elimina.

GetLoginUrl, nel caso $status sia "waiting", esegue il comando *Oauthcreate()* e ritorna la variabile $url. Nel caso lo status sia "OK" , ovvero il social in questione risulta già correttamente autenticato, restituisce il valore false, ad indicare appunto che non è necessaria un ulteriore autorizzazione.

Le classi che implementeranno la classe social dovranno occuparsi di definire i metodi astratti della stessa.
SocialName() dovrà essere un metodo che restituisca il nome del social implementato.

OauthCreate() dovrà svolgere le operazioni necessarie ad inizializzare il processo di autenticazione oauth, a salvarne se necessario le informazioni, e generare l'URL a cui redirigere l'utente per la successiva conferma dell'autorizzazione.

OauthCallback() si occuperà di svolgere tutte quelle operazioni fondamentali a concludere il processo Oauth, quindi richiedere *l'access token* definitivo "in cambio" del *oauth_verifier* ricevuto nel callback.

OauthConnect() si occupa di inizializzare l'oggetto $connection preparandolo per poterne poi chiamare il metodo "api", utilizzando la "consumer key", e il relativo segreto, dell'applicazione nonché l'*access token*, e relativo segreto, nel caso in cui esso sia disponibile.

OauthConnect() dovrà nel caso sia possibile aggiornare automaticamente l'*access token* scaduto, restituendo il valore "true" nel caso l'operazione sia andata a buon fine , "false" altrimenti.
Nei social che non permetto il refresh del token questo metodo sarà un semplice "return false".

Checkstatus() dovrà eseguire la più semplice api a disposizione del social network per verificare l'effettivo funzionamento della connessione (e quindi accertarsi della validità delle credenziali in nostro possesso) e restituire il valore "true" se l'operazione ha avuto successo, "false" in caso contrario.

Le due api sicuramente disponibili in tutti i social network sono quella che permette di ottenere informazioni sul profilo dell'utente per il quale stiamo operando e quella che permette l'inserimento di un semplice messaggio di testo.
Ogni classe che implementi la classe social deve poter eseguire queste due basilari operazioni e dovra farlo, rispettivamente, nei metodi *GetProfile()* e *Sendnews($messagge*).

CLASSE SOCIALOAUTH (BACK-END)

La classe astratta socialoauth, contiene tutte le istruzioni necessarie a creare e gestire una comunicazione secondo il protocollo OAUTH.

Socialoauth si appoggia alla libreria *liboauth.php* inizialmente sviluppata da Andy Smith e liberamente reperibile all'indirizzo "http://oauth.googlecode.com/svn/code/php/".
Questa libreria fornisce le classi e le interfacce necessarie a costruire in maniera corretta le richieste secondo il protocollo oauth , preoccupandosi di tutto ciò che riguarda la firma delle richieste.
Rende disponibile le classi nelle quali inserire tutti i diversi parametri della richiesta per poi poterli estrarre in maniera coerente e correttamente firmati. Tutti i metodi utilizzati in SocialOauth si appoggiano a questa libreria per la creazione delle richieste da inviate al server.

Il costruttore __construct inizializza la "connessione" utilizzando di base le sole credenziali consumer,utilizzate da sole per le sole richieste di *request token*, se non sono presenti credenziali più "elevate". Se quest'ultime invece sono presenti vengono utilizzate in combinazione con le credenziali consumer per creare una connessione in grado di effettuare le richieste di livello superiore (request_token per ottenere l'access_token oppure access_token per eseguire le api)

getRequestToken è il metodo con il quale è possibile richiedere al server il primo componente di una corretta autenticazione oauth ovvero il request_token. Come parametri accetta $callbackUrl, l'url a cui reindirizzare l'utente dopo che egli avrà garantito l'autenticazione e $scope, ovvero l'indicazione delle funzionalità "avanzate" che l'applicazione andrà ad utilizzare e che dovranno essere confermate durante l'autenticazione dell'utente.
Questi dati dovranno essere comunicati al server in fase di richiesta di request_token e non potranno essere modificati successivamente a meno di non ripetere la richiesta.

getAuthorizeURL restituisce l'indirizzo a cui l'utente dovrà collegarsi per concedere l'autorizzazione all'applicazione.

Con *getAccessToken* è possibile richiedere al server *l'access_token*. Se la variabile $refresh ha valore true, si tratta di una richiesta tramite refresh del token e la variabile $oauth_verifier conterrà il refresh_token ricevuto insieme all'access_token ormai scaduto.
Se invece la variabile $refresh ha valore false (il valore di default), si tratta di una richiesta effettuata in seguito all'autorizzazione da parte dell'utente, e la variabile $oauth_verifier conterrà appunto l'oauth_verifier ricevuto nel callback ad arya.

ApI invece è il metodo con il quale inviando le richieste al server. Accetta come parametri il metodo della richiesta HTTP, l'URI da richiedere, i parametri da inviare con la richiesta sotto forma di vettore, ulteriori dati da inviare nel corpo della richiesta ed un'eventuale specifica della tipologia di dati inviati.

```
abstract class SocialOAuth
  {
  //VARIABILI
  private $offsite = true;
  protected $MS_API_ROOT = "";
  public $MS_DUMP_REQUESTS = '';

  // OAuth URLs
  protected abstract function accessTokenURL();
  protected abstract function authorizeURL();
  protected abstract function requestTokenURL();

  //METODI
  public function __construct($consumerKey,
                              $consumerSecret,
                              $oAuthToken = null,
                              $oAuthTokenSecret = null,
                              $isOffsite = true,
                              $authorized_verifier = '')
  public function getRequestToken($callbackUrl,$scope=null);
  public function getAuthorizeURL($token,$additionalparams=NULL);
  public function getAccessToken($oauth_verifier = FALSE, $refresh=false);
  public function Api($method=null,$REST,$params=null,$body=null, $contenttype=null);
  public function makeOAuthRequest(  $url,
                                     $qParams=array(),
                                     $method,
                                     $headers=null,
                                     $body=NULL);

  private function makeHttpRequest(  $method,
                                     $url,
                                     $qParams,
                                     $headers=array(),
                                     $bodyContent=NULL);
  private function dump($text);
  }
```

Per ogni Social Network bisognerà definire una classe del tipo <nomesocial>oauth che implementi questa classe astratta. La classe così creata dovrà semplicemente :

- Definire il contenuto della variabile $MS_API_ROOT, ovvero la parte dell'url comune a tutte le api utilizzate dal social network. Questo permetterà di semplificare la chiamata alle api indicando al sistema solo la parte finale dell'indirizzo.
- Definire il valore della variabile $MS_DUMP_REQUEST, cioè l'indirizzo locale dove salvare tutte le comunicazioni a scopo di debug. Il logging verà effettuato solamente se il debug è stato attivato ponendo a true l'apposita variabile nel file config.php.
- Definire i 3 metodi accesstokenurl(), authorizeurl() e requensttokenurl() che dovranno restituire, rispettivamente, gli indirizzi a cui effettuare le richieste per l'accesstoken, a cui indirizzare l'utente per l'utenticazioni e a cui richiedere il requesttoken.

Esempio (twitteroauth)

```
<?php
require_once('socialoauth.php');
class TwitterOAuth extends SocialOAuth
  {
  protected $MS_API_ROOT = "https://api.twitter.com/1";
  public $MS_DUMP_REQUESTS = /upload/arya/twitter.log;

  protected function accessTokenURL()
    { return 'https://api.twitter.com/oauth/access_token'; }
  protected function authorizeURL()
    { return 'https://twitter.com/oauth/authenticatÈ; }
  protected function requestTokenURL()
    { return 'https://api.twitter.com/oauth/request_token'; }
  }
?>
```

3.5.3. DATA BASE (BARBARA)

Arya utilizza 2 tabelle per memorizzare le informazioni.
La prima , di nome “user” permette la memorizzazione degli utenti e delle rispettive chiavi di cifratura.

USER

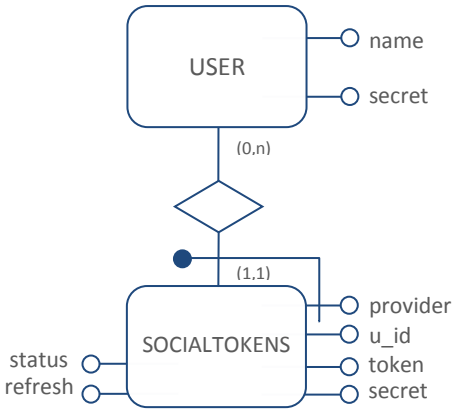
name	secret
Barbara	secretB
Marco	secretM

La seconda, di nome “socialtoken” è utilizzata per salvare i token di accesso di ogni utente per tutti i social network dei quali usufruisce.
Se l’autorizzazione è già stata concessa e verificata lo stato sarà “OK” altrimenti sarà “Waiting” che può indicare che l’utente non ha ancora permesso l’accesso a quel social network o che non ha intenzione di utilizzarlo.

SOCIALTOKEN

utente	provider	u_id	token	secret	refresh	status
Barbara	Facebook	id_B_FB	token_B_FB	secret_B_FB		OK
Marco	Facebook	id_M_FB	token_M_FB	secret_M_FB		OK
Barbara	Yahoo	id_B_YH	token_B_YH	secret_B_YH	refresh_B_YH	OK
Marco	Yahoo					Waiting
Barbara	LinkedIn		token_B_FB	secret_B_LI		Waiting
Marco	LinkedIn	id_M_LI	token_M_LI	secret_M_LI		OK

Ed ecco lo schema E/R che indica le connessioni fra le due tabelle.



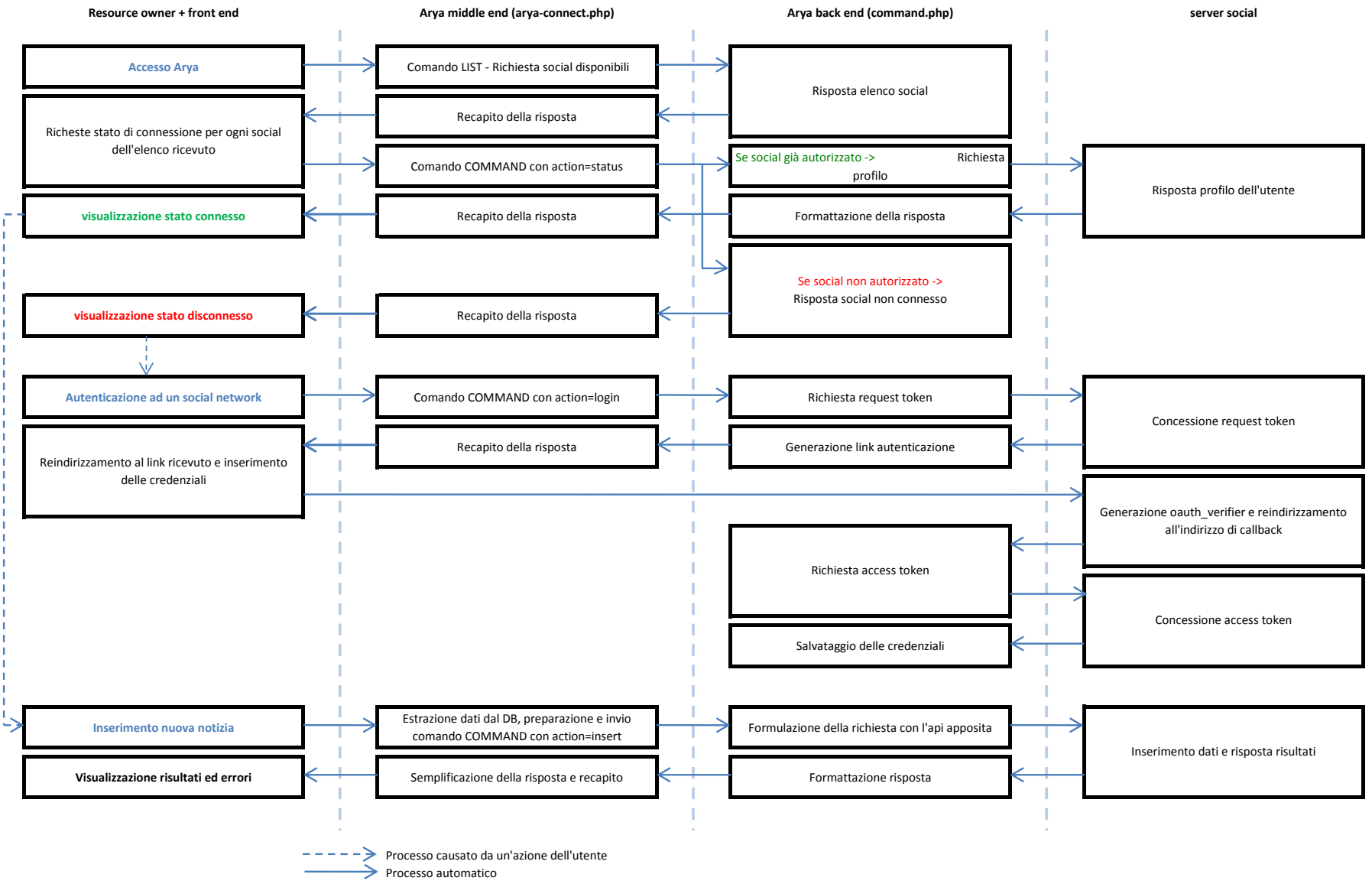
3.5.4. CRITERI DI SICUREZZA (MARCO)

Al fine di garantire l'autenticità e l'integrità delle informazione scambiate tra middle end e back end si è decisi di ricorrere alla tecnica della firma digitale (o message digest).
Questa tecnica non cifra le informazioni scambiate tra le 2 identità e quindi non assicura la segretezza della comunicazione. Si può ben capire come questo fattore sia per noi totalmente trascurabile in quanto il nostro scopo è proprio quello di PUBBLICARE informazioni su internet.
L'autenticità però deve essere garantita al fine di evitare che soggetti terzi possano pubblicare informazioni usando l'account del nostro "cliente". L'integrità, ovvero la certezza che il messaggio non sia stato modificato durante il trasferimento lungo la rete, deve essere assicurata per lo stesso motivo.

Una volta creato nel middle-end il messaggio di esso ne viene generata la firma con la tecnica HMAC (Hash-based Message Authentication Code), usando l'algoritmo SHA-1 e con una chiave simmetrica alfanumerica precondivisa, ed inserita nella stringa json nel campo 'sig'.
Nello specifico la chiave viene generata dall'amministratore di ARYA al momento della creazione di un nuovo utente, salvata automaticamente del database di ARYA, ed inserita **manualmente** in fase di configurazione del middle-end all'interno del file *account.php*.
La chiave è quindi simmetrica essendo uguale sia per chi trasmette che per chi riceve.
Per aumentarne la sicurezza la chiave può avere una lunghezza variabile tra i 160 e i 200 caratteri, e nel messaggio viene inserito il timestamp di creazione dello stesso.

Il back-end, ricevuto il messaggio, ne estrarrà la firma e la ricalcolerà con la chiave presente nel suo database. Se le due firme coincidono è **garantita** autenticità e integrità del messaggio, e viene quindi concessa l'esecuzione dell'operazione richiesta per conto dell'utente specificato.
È stato deciso in fase di creazione un tempo limite per la validità del timestamp; nel caso in cui il messaggio sia stato creato 10 minuti prima della ricezione, situazione che si può verificare in caso di errori sulla rete o in un tentativo di replay-attack da parte di un malintenzionato, il comando viene scartato in quanto obsoleto.

3.5.5. DATA FLOW (MARCO)



4. Codice

4.1. Connessioni

4.1.1. Front-End (arya.htm e default.js) (Marco)

Arya.htm si occupa solamente di definire la struttura sulla quale javascript andrà ad inserire i contenuti.
Sfruttando la possibilità di comunicazione asincrona offerta da AJAX, il front-end comunica al middle-end le istruzioni da eseguire e non appena i risultati sono pronti l'interfaccia viene aggiornata senza blocchi o rallentamenti visibili per l'utente.

L'interfaccia può richiedere al middle-end 5 tipi di operazioni:

- Elenco dei social network disponibili
- Stato di connessione di un social
- Connessione ad un social
- Disconnessione da un social
- Invio della notizia ad un social

È consentito inviare la stessa notizia a più social contemporaneamente semplicemente eseguendo più volte l'istruzione 5, una per ogni social desiderato.I diversi invii verranno così eseguiti in parallelo, visto l'utilizzo di una comunicazione asincrona tra front-end e middle-end. Questo tipo di comunicazione infatti non blocca il canale comunicativo in attesa della risposta dopo l'invio di un comando, ma permette appunto l'esecuzione di comunicazioni contemporanee sullo stesso canale, con la certezza che quando la risposta sarà disponibile essa sarà correttamente ricevuta e smistata alla richiesta corrispondente.

4.1.2. Struttura dati per comunicazione tra front e back end (Marco)

Per eseguire i 5 comandi visti sopra si utilizzano altrettante strutture dati inviate alla pagina *arya-connect.php* con il metodo POST.

<nome campo>:<tipo variabile>	***//<Significato del contenuto>***

Elenco dei social network disponibili

list:---	***//qualunque valore***

Stato di connessione di un social

status:string	***//nome del social desiderato***

Connessione ad un social

login:string	***//nome del social desiderato***

Disconnessione da un social

logout:string	***//nome del social desiderato***

Invio della notizia ad un social

insert:string	***//nome del social desiderato***
id:int	***//ID della notizia da inviare***
page:string	***//ID della pagina per cui postare(facebook)***
lingua:string	***//lingua della notizia da inviare***

4.1.3. MIDDLE-END (ARYA-CONNECT.PHP) (BARBARA)

```
<?php
//Funzione per l'invio dei dati al back-end di arya tramite la funzione PHP curl.
function send($body)
  {
  ...
  }

if (isset($_POST['list']))
  {
  $response=send(array("list"=>true)); //Richiedo l'elenco dei social network
  echo $response['body'];
  }

//Verifico che l'utente sia un amministratore del sito
//e che le credenziali per la connessione ad Arya siano presenti
else if(  isset($_SESSION["sessloggedstatus"]) &&
          $_SESSION["sessloggedstatus"] == "LOGGED" &&
          isset($_SESSION["sessuserstatus"]) &&
          $_SESSION["sessuserstatus"]=="A")
  {
  if(!defined("ARYA_NAME") || !defined("ARYA_KEY"))
    {
    header("HTTP/1.1 400 Bad Request");
    $result=array(    "error"=>"fatal",
                      "error-message"=>    "Dati di accesso non definiti. Verifica la presenza del file account.php
                                           o richiedilo a www.mediacomservice.com");
    echo json_encode($result);
    }

  //Se è tutto a posto creo il comando da inviare.
  else
    {
    $command['user']=ARYA_NAME;

    if(isset($_POST['status']) && $_POST['status'])
      {
      $command['social']=$_POST['status'];
      $command['action']="status";                    //Richiedo lo stato per il social
      $command['timestamp']=time();
      //Firmo la richiesta
      $command['sig']=hash_hmac ( "sha1" , json_encode($command) , ARYA_KEY);
      //Invio il comando
      $response=send(array("command"=>json_encode($command)));
      //Ritorno la risposta ricevuta
      echo $response['body'];
      }

    elseif(isset($_POST['logout']) && $_POST['logout'])
      {
      $command['social']=$_POST['logout'];
      $command['action']="logout";                    //Richiedo il logout dal social
      $command['timestamp']=time();
      //Firmo la richiesta
      $command['sig']=hash_hmac ( "sha1" , json_encode($command) , ARYA_KEY);
      //Invio il comando
      $response=send(array("command"=>json_encode($command)));
      //Ritorno la risposta ricevuta
      echo $response['body'];
      }

    elseif(isset($_POST['login']) && $_POST['login'])
      {
      $command['social']=$_POST['login'];
      $command['action']="login";                     //Richedo il login al social
      $command['timestamp']=time();
      //Firmo la richiesta
      $command['sig']=hash_hmac ( "sha1" , json_encode($command) , ARYA_KEY);
      //Invio il comando
      $response=send(array("command"=>json_encode($command)));
      //Ritorno la risposta ricevuta
      echo $response['body'];
      }
```

```
elseif( isset($_POST['insert']) && ($_POST['insert'])
        && isset($_POST['id']) && ($_POST['id'])
        &&isset($_POST['lingua'])&& ($_POST['lingua']))
  {
  $command['social']=$_POST['insert'];
  $command['action']="insert";          //Richiedo l'inserimento di un nuovo messaggio

  //se il campo è presente segnalo che l'invio del messaggio è per conto di una pagina
  if (isset($_POST['page'])) $command['data']['page']=$_POST['page'];


  //Query per estrarre la notizia dal database
  $q="   SELECT *
         FROM article AS a
         INNER JOIN content AS c
         ON a.id=c.id_article
         INNER JOIN lang AS l
         ON c.id_lang=l.id
         INNER JOIN tipo AS t
         ON a.id_type=t.id
         WHERE a.id='".$_POST['id'].
         "' AND l.short='".$_POST['lingua']."'";

  //Eseguo la query
  $data=mysql_query($q, $GLOBALS[conn]);
  $row = mysql_fetch_assoc($data);

  //compilo il campo data della richiesta con le informazioni appena estratte dal DB
  $command['data']['link']=ROOT_PATH.$row['namÈ].'/'.$row['permalink'];
  $command['data']['titlÈ]=($row['titlÈ]);
  $command['data']['body']=($row['abstract']);
  $command['data']['html']=($row['content']);

  //Quando necessario sostituisco i percorsi relativi dei link presenti nel corpo del
  //messaggio con i loro valori assoluti sfruttando le sostituzioni tramite
  //regular expression
  $pattern[0]='#(src|href)=(\'|")(?!(http|www))#';
  $pattern[1]='#(src|href)=(\'|")www#';
  $replace[0]="$1=$2".ROOT_PATH;
  $replace[1]="$1=$2http://";
  $command['data']['html']= preg_replace($pattern,$replace,$command['data']['html']);

  //Indico al back-end se ciò che sto inviando è un messaggio, un album o un evento
  switch($row['namÈ])
    {
    case 'images':
      $command['data']['typÈ]='album';
      break;
    case 'events':
      $command['data']['typÈ]='event';
      $command['data']['start_datÈ]=strtotime($row['date_start']);
      $command['data']['end_datÈ]=strtotime($row['date_end']);
      $command['data']['location']=$row['country'];
      break;
    case 'news' :
    default:
      $command['data']['typÈ]='messagÈ;
      break;
    }

  //Query per estrarredal DB tutte le immagni riguardanti la notizia
  $qimg="SELECT i.id AS id, i.ext AS ext, c.abstract,c.title, c.content, c.permalink
           FROM images AS i
           INNER JOIN content AS c
           ON i.id=c.id_article
           INNER JOIN lang AS l
           ON l.id=c.id_lang
           WHERE i.id_article=".$_POST['id']."
           AND l.short='".$_POST['lingua']."'
           ORDER BY id ASC";

  //Eseguo la query
  $dataimg=mysql_query($qimg, $GLOBALS[conn]);
```

```
        //Inserisco i riferimenti di tutte le immagini appena estratte, necessari al back-end
        //per il loro corretto caricamento nel social network
        while ($rowimg= mysql_fetch_assoc($dataimg))
          {
          if (isset($rowimg['titlÈ'])) $tit='['.$rowimg['titlÈ'].'] ';
          else $tit="";
          $command['data']['photos'][]=array('picture'=>ROOT_PATH.'upload/immagini/'.$rowimg['id'].'.'.$rowimg['ext'],
                                             'text'=>$tit.$rowimg['content']  );
          }

        //inserisco il timestamp corrent
        $command['timestamp']=time();

        //Firmo la richiesta
        $command['sig']=hash_hmac ( "sha1" , json_encode($command) , ARYA_KEY);

        //Invio il comando
        $response=send(array("command"=>json_encode($command)));

        //Decodifico la risposta
        $response=json_decode($response['body'],true);

        //Semplifico la risposta per l'interfaccia utente
        $result['error']=$response['error'];
        $result['error_messagÈ]=$response['error_messagÈ];
        $result['social']=$response['social'];
        $result['user']=$response['user'];
        $result['pagÈ]=$response['pagÈ];
        switch($row['namÈ])
          {
          case 'images':
            $result['link']=$response['album']['link'];
            break;
          case 'events':
            $result['link']=$response['event']['link'];
            break;
          case 'news' :
          default:
            $result['link']=$response['messagÈ]['link'];
            break;
          }

        //Rispondo alla richiesta del front-end
        echo json_encode($result);
        }
      }
    }
?>
```

Il middle-end è quella parte di Arya che si occupa di convertire i comandi dati dall'utente attraverso il front-end in ordini comprensibili ed utilizzabili dal back-end, preoccupandosi inoltre di firmarli correttamente.
Nel caso riceva il comando di pubblicare una certa notizia si incarica dell'estrazione dei dati dal database del CMS "Sapore" su cui è basato il sito dell'utente, e della loro corretta formattazione secondo lo schema visibile nel capitolo successivo.

4.1.4. STRUTTURA DATI PER COMUNICAZIONE TRA MIDDLE E BACK END (BARBARA)

Arya risponde a comandi inviati alla pagina command.php, tramite variabili in post contenti una stringa JSON.
A questi comandi arya risponde sempre con una rappresentazione JSON della risposta.

Il comando LIST, ovvero il comando che permette di richiedere ad Arya l'elenco dei social network supportati, si effettua inviando una richiesta per la pagina command contente una variabile in POST di nome list di valore qualunque. Per il comando list non è richiesta nessuna autorizzazione.


```
list:---                                    //qualunque valore
```

Per ordinare al back-end di eseguire tutte le altre operazioni è necessario inviare una variabile di nome command in POST durante la richiesta di command.php. la variabile command dovrà contenere una stringa rappresentante, in formato json, l'operazione da eseguire rispettando lo schema sotto indicato.

```
command:
  {
  *user: string                             //nome dell'utente di Arya
  *social: string                           //nome social
  *timestamp: int                           //timestamp della richiesta
  *action: (status,login,logout,insert)     //tipo di azione da eseguire
  data:
    {
    type: string(message,event,album)       //tipo di inserimento
    page: string                            //id della pagina o utente su cui postare
    link: string                            //url della notizia originaria
    title: string                           //titolo del messaggio
    body: string                            //corpo del messaggio
    html: string                            //corpo del messaggio formattato html
    public: bool(true,false)                //visibilità
    photos:                                 //array di elementi foto
      [
        {
        picture:                            //url dell'immagine
        text:                               //commento dell'immagine
        },
        ...
      ]
    start_date: int                 //timestamp di inizio dell'evento
    end_date: int                   //timestamp di inizio dell'evento
    location: string                // riassunto del luogo dell'evento
    venue :                         //dati specifici del luogo dell'evento
      {
      street:
      city:
      zip:
      country:
      latitude:
      logitude:
      }
    }
  *sig: string                              // firma
  }
```

Il campo data è obbligatorio nel caso in cui invii un comando con il valore *action* uguale ad *insert*. Il campo infatti conterrà le informazioni da inserire nel social network.

Tutte le richieste devono avere il campo sig contente la firma del messaggio con la chiave dell'utente.

4.1.5. BACK-END (COMMAND.PHP) (MARCO)


```
<?php
...
//da notarsi l'utilizzo della funzione speciale __autoload di php per il caricamento //dinamico delle classi.
function __autoload($class_name)
   {
   include "Social/" . $class_name . ".php";
   }

//Lista dei social supportati
$list=array("Twitter","Facebook","Myspace","Linkedin","Yahoo","Buzz");

//risposta al comando semplice list che restituisce la lista dei social supportati.
if (isset($_POST['list']))
   {
   $result=$list;
   }

//risposta ai comandi che necessitano di autorizzazione, inviati come stringa json nella //variabile 'command' con
il metodo POST

else if (isset($_POST['command']) && $_POST['command'])
      {

      $error="";
      //converto la stringa json in un oggetto
      $command=json_decode($_POST['command'],true);

//viene verificata la presenza e la correttezza di tutte le informazioni necessarie
//generando errore e terminando nel caso vi siano problemi
if(!isset($command['user']))
      {
      $result=array("error"=>"fatal","error_message"=>"Utente non definito.");
      break;
      }
   if(!isset($command['social']) || !in_array($command['social'],$list))
      {
      $result=array("error"=>"fatal","error_message"=>"Social non definito.");
      break;
      }
   if(!isset($command['action']))
      {
      $result=array("error"=>"fatal","error_message"=>"Nessuna azione definita.");
      break;
      }
   if(!isset($command['sig']))
      {
      $result=array("error"=>"fatal","error_message"=>"Comando non firmato.");
      break;
      }
   if(!isset($command['timestamp']))
      {
      $result=array("error"=>"fatal","error_message"=>"Timestamp non presente.");
      break;
      }
   if((time()-$command['timestamp'])>600)
      {
      $result=array("error"=>"fatal","error_message"=>"Messaggio scaduto (timestamp obsoleto).");
      break;
      }

   //Estraggo la firma ricevuta
   $oldsig=$command['sig'];
   unset($command['sig']);
   $auth= new AryaAuth();

   if(!$secret=$auth->selectAuth($command['user']))
      {
      $result=array("error"=>"fatal","error_message"=>"Utente non presente.");
      break;
      }

   //calcolo nuovamente la firma con la chiave presente nel database di arya
   $newsig=hash_hmac ( "sha1" , json_encode($command) , $secret);
```

```
//se le firme coincidono proseguo altrimenti genero un errore e termino.
if($oldsig != $newsig)
  {
  $result=array("error"=>"fatal","error_message"=>"Firma non valida.");
  break;
  }

//se è tutto corretto eseguo il commando richiesto
switch ($command['action'])
  {
  case 'status':                                          //commando status
    //creo un'istanza del social  richiesto per l'utente desiderato
    $soc=new $command['social']($command['user']);

    //se la connessione è già attiva  richiedo e ritorno lo stato dell'utente
    if ($soc->status=='OK')
      {
      $result=array(          "name" => $command['social'],
                    "logged"=>true,
                    "pages"=>$soc->getProfile(),
                    "error"=> "none",
                    "error_message" => "");
      }
    //altrimenti segnalo che il social non è connesso
    else
      {
      $result=array(          "name"    =>$command['social'],
                    "logged"=>          false,
                    "error"=> "none",
                    "error_message" => "");
      }
    break;


  case 'login':                                           //commando login

    //creo un'istanza del social  richiesto per l'utente desiderato
    $soc=new $command['social']($command['user'])

    //genero il link per l'autenticazione e lo ritorno insieme a delle variabili di supporto
    if($loginurl=$soc->getLoginUrl())
      {
      $result=array(          "name"    =>        $command['social'],
                              "link"    =>        array(    "url" => $loginurl,
                              "popupx"=> $soc->popupx,
                              "popupy"=> $soc->popupy),
                              "error"=> "none",
                              "error_message" => "");
      }
    //altrimenti ritorno un errore
    else
      {
      $result=array(          "error"=>"fatal",
                              "error_message"=>"Social già connesso.");
      }
    break;


  case 'logout':                                          //commando logout

    //creo un'istanza del social  richiesto per l'utente desiderato
    $soc=new $command['social']($command['user']);

    //rimuovo il token memorizzato
    $soc->removeToken();
    break;


  case 'insert':                                          //commando insert

    //creo un'istanza del social  richiesto per l'utente desiderato
    $soc=new $command['social']($command['user']);
```

```
        //se la connessione è già attiva  eseguo l'operazione e ritorno il risultatoricevuto
        if ($soc->status=='OK')
          {

          $function=$soc->methods[$command['data']['type']];
          $result=$soc->$function($command['data']);
          }

        //altrimenti ritorno un errore
        else
          $result=array("error"=>"fatal","error_message"=>"Social non connesso.");
        $result['social']=$command['social'];
        $result['user']=$command['user'];
        if (isset($command['data']['page']))
          $result['page']=$command['data']['page'];
        break;

      default:                                                    //commando non riconosciuto

        //ritorno un errore
        $result=array("error"=>"fatal","error_message"=>"Comando non supportato.");
      }

    }

//se il comando non è specificato ritorno un errore
else
  {
  $result=array("error"=>"fatal","error_message"=>"Nessun comando specificato.");
  }

...
//Rispondo alla richiesta del middle-end
echo json_encode($result);
?>
```

Come si può vedere la flessibilità fornita da PHP ci permette di costruire un sistema snello e decisamente modulare. Il caricamento delle librerie necessarie avviene durante l'esecuzione del codice con il criterio che se deve essere istanziata una libreria di nome, ad esempio, lib1 , PHP troverà la sua definizione nel file */Social/lib1.php* .
In questo modo non è necessario caricare preventivamente tutte le librerie lasciando al motore PHP la libertà di caricarle solo quando effettivamente necessario.

PHP ci permette inoltre di istanziare classi in runtime di cui non conosciamo il nome a priori sfruttando il comando "*new $variabile(…)*" dove $variabile conterrà il nome della classe da istanziare.

In questo modo sarà possibile aggiungere nuovi social network ad Arya semplicemente aggiungendone il nome nella la variabile $list e definendone la classe nella cartella Social.

4.2. INTERFACCIA (MARCO)

Questa è l'interfaccia grafica di Arya. Si può notare la zona a sinistra dove si trovano le icone dei social non connessi al cui click viene inizializzata la procedura di autenticazione.

Sulla Destra è visibile la zona riservata ai social già autorizzati con la possibilità di selezionare tramite checkbox a quali inviare la notizia per la pubblicazione.

Facebook mostrerà tutte anche le pagine controllate dall'utente.
La barra verde sotto il nome del social indica che è in corso un'operazione di invio per quel social. All'estrema destra appaiono, una volta terminato un invio, degli indicatore dello stato dell'operazione. Posizionandosi con il mouse sopra ognuno di esso si ottengono informazioni estese sul risultato dell'invio corrispondente.

Il pulsante invia comanda al middle-end di pubblicare la notizia corrente.

5. Conclusioni

5.1. Risultati raggiunti

Il software da noi sviluppato soddisfa tutti i requisiti richiesti nel capitolo 2.1.1 .

L'integrazione tra il lavoro dei due stagisti non ha avuto problemi vista l'attenta pianificazione iniziale e la definizione precisa dello standard di comunicazione tra le diverse componenti del client.

Pubblica le news desiderate in tutti i social analizzati sfruttando le api messe a disposizione, nella maniera migliore possibile concessa dal sito, ovvero pubblicando la notizia nella maniera più completa permessa inserendo da un semplice testo semplice in twitter ad un più completo testo formattato in html ed un album fotografico nel più accessoriato Facebook.
È molto modulare e permette l'inserimento di nuovi social e la rimozione dei vecchi in maniera trasparente per l'utente finale e, soprattutto, senza sconvolgere la struttura attuale ma attraverso l'aggiunta di poche e semplici righe di codice.

Il software Arya è stato testato su un sito di prova messo a disposizione da mediacom service, clonato, con il permesso dell'interessato, da un sito realmente esistente e funzionante e l'inserimento nella struttura è stata semplice e senza difficoltà, ben integrandosi con il resto dell'interfaccia di amministrazione.

I risultati raggiunti hanno soddisfatto appieno sia gli stagisti che, molto più importante la società.

5.2. Sviluppi futuri

Vista la solida struttura di base e l'ottima modularità il passo successivo sarà quello di aggiungere sempre più social per permettere un servizio sempre più efficiente.

In futuro, con l'avvento di Oauth 2.0 , attualmente in fase di definizione ma già utilizzato da Facebook, si andrà ad implementare nella libreria socialoauth il codice necessario per supportare i social che si avvarranno di questa nuova autenticazione.
Essendo le differenze tra le due versione non troppo accentuate, il processo non dovrebbe presentare troppe difficoltà e dovrebbe tranquillamente permettere la convivenza di social che utilizzino i due differenti standard.

6. OPERA CITATE

JSON.org. (s.d.). Tratto da JSON.org: http://www.json.org/

Wikipedia, l'enciclopedia libera. (s.d.). *Wikipedia, l'enciclopedia libera*. Tratto il giorno 03 29, 2011 da http://it.wikipedia.org