Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Triennale in
Ingegneria Informatica

SVILUPPO DI UNA LIBRERIA PER LA TRASMISSIONE E ANALISI DI DATI CON UN SERVER REMOTO ATTRAVERSO UN SISTEMA WEB SERVICE

Laureando
Riccardo Fabris
mtr. 563603

Relatore
Prof. Giorgio Satta

24 settembre 2012
Anno Accademico 2011/2012

Indice dei contenuti

Indice delle Figure

Capitolo 1

Introduzione

1.1 Azienda

Il tirocinio si è svolto presso l'azienda Horizon ADV s.r.l. (ora diventata Horizon Group s.r.l).
L'azienda nasce nel 2008 come digital agency con l'obiettivo di fornire un servizio di *web marketing*, che varia dalla creazione di siti internet, fino alla realizzazione dei formati creativi destinati alla pianificazione pubblicitaria online.
Dal 2008 ad oggi molti sono stati i progetti sviluppati, tra i quali se ne possono individuare tre di maggior rilievo (HORIZON GROUP SRL, Piattaforma di Affiliazione - Programmi di Affiliazione LeadHouse):

- Nel maggio 2010 viene lanciato Leadhouse (HORIZON GROUP SRL, Piattaforma di Affiliazione - Programmi di Affiliazione LeadHouse) «Leadhouse è una piattaforma di marketing online che gestisce campagne online a performance, con l'obiettivo di offrire ai propri clienti un'ampia gamma di servizi orientati al ROI. Attraverso il proprio sistema di delivering e tracking certificato a livello mondiale, LeadHouse è in grado di erogare qualsiasi tipologia di campagna performance-based: CPC, CPL, CPS e Co-Registrazione, nonché campagne con remunerazione a CPM, monitorate costantemente attraverso il sistema di tracking che certifica il corretto delivering dei pacchetti CPM acquistati.».

- Nel maggio 2011 viene creato Fidelityhouse (HORIZON GROUP SRL, Guadagnare online con il Social Cashback FidelityHouse), una piattaforma di *cashback*.« FidelityHouse è una community di fedeltà che permette, agli utenti iscritti, di ottenere un guadagno dalle azioni che compiono sul web. FidelityHouse nasce come risposta a due precise esigenze del mercato web: quella di valorizzare le azioni e i comportamenti dell'utente come gli acquisti, gli interessamenti a prodotti e servizi, la condivisione di contenuti, il passaparola e, più in generale, l'attività sul web, e quella di ottimizzare gli investimenti degli inserzionisti, che hanno la possibilità di pianificare campagne B2C su un target specifico.».

- Nel luglio 2012 nasce NextNews360 (HORIZON GROUP SRL, NEXTNEWS360), un network di blog di informazione.« Nextnews360 è un network di blog verticali pensato per mettere a disposizione dei lettori contenuti sempre nuovi ed aggiornati su vari argomenti di interesse. Con una grafica fresca ed accattivante e un team composto da blogger giovani e appassionati al proprio lavoro, che raccontano con

originalità le news e gli eventi più importanti, Nextnews360 si pone l'obiettivo di diventare un punto di riferimento nel panorama dei media digitali italiani, con un'offerta tematica sempre più ampia e completa.».

1.2 Contesto operativo

Lo stage si è protratto dal 31 novembre 2011 al 13 dicembre 2012 per un totale di 250 ore lavorative. Il primo periodo è stato dedicato allo studio dell'ambiente, dei linguaggi di programmazione necessari allo sviluppo del progetto e del *framework* utilizzato in azienda. Una volta acquisite le basi, è stato concordato un percorso a livelli che poi portasse alla realizzazione della libreria richiesta.

1.3 Descrizione

L'obiettivo dello stage era quello di realizzare una libreria per il *framework CodeIgniter* (si rimanda al paragrafo 2.3 per un approfondimento su *CodeIgniter*). La libreria doveva poter popolare, in primo luogo, e aggiornare, in secondo, un *database* creato *ad hoc* per poter visualizzare graficamente gli andamenti di alcuni parametri importanti per l'azienda. La libreria, inoltre, doveva permettere la consultazione di tutti i dati contenuti nel *database*, in alcuni casi mettendo a disposizione la selezione di un intervallo temporale.
In questa relazione saranno delineati gli strumenti di programmazione utilizzati per la realizzazione della libreria e dell'interfaccia grafica con cui visualizzare i dati analizzati. Saranno esaminati tutti i grafici di andamento richiesti dal progetto e verrà, inoltre, fornito il principale codice sorgente della libreria realizzata.
L'obiettivo principale di questa relazione risulta essere, quindi, la creazione di una esauriente descrizione degli strumenti messi a disposizione dalla libreria e il modo in cui sono stati implementati.

Capitolo 2

Strumenti

2.0 Strumenti

In questo capitolo verranno introdotti i principali strumenti utilizzati e le nozioni apprese che stanno alla base dello sviluppo del progetto.
In primo luogo saranno presi in considerazione i concetti di *affiliation marketing* andando ad individuare quali sono le maggiori realtà italiane che sviluppano questo tipo di *marketing*. In secondo luogo, si analizzeranno gli strumenti utilizzati.
Questa trattazione risulta essere molto utile perché individua in quale ambiente di sviluppo è stato progettata la libreria e quali sono stati gli strumenti a disposizione.

2.1 Affiliazione

Negli ultimi anni l'*online advertising* si è imposto come una delle più grandi fonti di guadagno che il World Wide Web possa generare. Ci sono svariate formule di advertising tra cui le più importanti sono:

- *Pop up ad;*
- *Web banner;*
- *E-mail spam;*
- *Serch engine marketing;*
- *Affiliation marketing.*

Di particolare interesse risulta quest'ultima formula. L'*affiliation marketing* prevede la presenza, in linea generale, di tre figure:

I. Il cliente, colui che vuole la pubblicità;
II. Il mediatore, colui che gestisce la piattaforma di affiliazione;
III. L'affiliato, colui che visualizza le pubblicità per conto del mediatore.

A livello italiano esistono due principali piattaforme di affiliazione:

- *Tradedoubler* (Tradedoubler)
- *Zanox* (Zanox)

Entrambe hanno come clienti grandi aziende internazionali o nazionali che intendono promuovere il proprio marchio, e come affiliati, aziende con realtà più piccole che utilizzano le pubblicità create per dare l'effettiva visibilità ai clienti delle piattaforme di affiliazione. Ovviamente, questo vale in linea di

principio, infatti, poi, esistono molte varianti, ma tutte comunque riconducibili a questa struttura.

Figure 1: Schema Affiliation Marketing

L'azienda in cui è stato svolto lo stage, in questo caso, ricopre il ruolo sia di affiliato, rispetto alle grandi piattaforme di affiliazione, ma anche di mediatore, essa stessa, verso degli affiliati terzi.
In particolare, l'azienda, come *web agency*, risulta essere affiliata di *Tradedoubler* e *Zanox*, quindi utilizza e rende visibile su propri siti la pubblicità di terze parti, mentre come piattaforma di affiliazione, vengono affiliati portali di una certa importanza come *Leonardo* (Triboo Editoriale S.r.l.), *Tiscali* (Tiscali Italia S.p.a), *Libero* (LIBERO), per assicurare la visibilità delle pubblicità sia per i propri clienti diretti sia per quelli indiretti derivanti da *Tradedoubler* o *Zanox*.

L'*affiliation marketing,* inoltre, presenta una grande differenza rispetto ai classici metodi di *web advertising*. Infatti, il compenso che ne deriva viene erogato solamente nel momento in cui un'azione viene eseguita sul sito affiliato, quindi con una modalità di tipo *CPA (Cost-per-Action), CPO (Cost-per-Order), CPA (Cost-per-Acquisition), CPL (Cost-per-Lead), CPD (Cost-per-Download).*

2.2 Tradedoubler

2.2.1 Presentazione

La libreria di cui l'azienda ha richiesto l'implementazione, dovrà interfacciarsi con il servizio fornito da una delle due grandi piattaforme di affiliazioni in Italia: *Tradedoubler*.

Figure 2: Logo Tradedoubler

Come si afferma nel loro sito italiano di riferimento (Tradedoubler, Affiliazioni online con Tradedoubler) «Tradedoubler è una comunità di affiliazioni online. Riuniamo inserzionisti, editori, agenzie e sviluppatori per consentire loro di condividere i contatti più redditizi e incentivare la crescita e le revenue per le loro aziende.
Il nostro obiettivo è quello di sfruttare appieno il modello delle affiliazioni Internet in cui gli inserzionisti pagano solo per le azioni di conversione, al fine di produrre valore aggiunto per i nostri clienti. Attraverso il nostro potente network di affiliazione, consentiamo ai nostri clienti di raggiungere diversi obiettivi relativi alle affiliazioni online e siamo gli unici ad aver reso disponibile l'innovativa tecnologia Tradedoubler come soluzione di affiliation marketing in-house.»

L'azienda nasce nel 1999 a Stoccolma, e, a partire dal 2001, comincia la sua ascesa. Di anno in anno le dimensioni del proprio network crescono sempre più, fino a raggiungere le dimensioni odierne che contano su oltre 140.000 editori attivi in tutto il mondo.
Nel mese di agosto 2011, *Tradedoubler* è stato votato come network di affiliazione numero 1 in Europa, nell'ambito di un sondaggio promosso da *A4U* (a4uevents) sulla qualità e affidabilità delle piattaforme di affiliazione.

2.2.2 Funzionalità utilizzate

Tra le varie funzionalità fornite da *Tradedoubler*, due sono quelle utilizzate dalla libreria sviluppata: *Programs Report e Breakdown Report.*
Il primo strumento mette a disposizione dell'affiliato un dettagliato report su tutti i programmi relativi a quel utente, mentre il secondo tutte le conversioni effettuate. Ogni conversione determina il pagamento di una percentuale del valore dell'azione eseguita, per esempio, un acquisto online, o di una somma *una tantum*, per esempio, per l'iscrizione ad un sito. I *report* vengono, poi, resi disponibili in dicersi formati tra i quali HTML, PDF, XLS, XML. Quest'ultimo risulta essere molto utile in quanto di facile utilizzo per andare a popolare prima e aggiornare poi un database contenente lo storico dei dati dei report.

2.3 CodeIgniter

Figure 3: Logo CodeIgniter

CodeIgniter è un *framework* orientato alla creazione di applicazioni web o siti internet in PHP.
Il *framework* mette a disposizione delle librerie che si occupano di quelli che sono le funzioni più comuni facilitando quindi l'implementazione dei task. L'obiettivo di *CodeIngniter* è quello di far focalizzare il programmatore sul quello che si sta realizzando, cercando di ridurre al minimo il codice. Inoltre, essendo molto compatto, permette lo sviluppo di applicazioni web particolarmente efficienti.
In principio, l'azienda in cui è stato svolto lo stage, stata indecisa sul *framework* da utilizzare. Le opzioni che sono state vagliate sono state principalmente tre:

- Simfony (Potencier);
- CodeIgniter (EllisLab, Inc, CodeIgniter - Open source PHP web application framework);
- Zend (Zend Technologies Ltd.).

Altri *framework* famosi come *Yii* e *Prado* nel periodo di valutazione non esistevano ancora.
A confronto con gli altri, *CodeIgniter* ha una buona community, dei tutorial descrittivi e video d'aiuto all'implementazione.
Il progetto è portato avanti e migliorato principalmente da un'azienda, EllisLab (EllisLab, Inc),che mantiene il progetto *opensource* con partecipanti esterni (GitHub Inc)e rilascia sul proprio sito versioni migliorate ogni sei mesi circa.

Il *framework* presenta un approccio di tipo MVC (Model-View-Controller) che permette di dividere effettivamente la logica applicative e l'interfaccia utente:

- il *model* gestisce i dati e contiene la logica implementata dall'applicazione;
- il *view* visualizza i dati gestiti dal *model* e si occupa dell'interazione con gli utenti;
- il *controller* permette che il model e il view lavorino assieme. Esso permette la trasformazione degli input provenienti delle *view* in reali modifiche del *model*.

Un altro punto fondamantale, che semplifica molto anche il primo approccio al *framework*, è che, per il suo utilizzo, non ha bisogno dell'invio di istruzioni da linea di commando. Inoltre, non richiede di imparare un apposito linguaggio per la gestione dei template, né l'utilizzo di un *template engine* esterno al *framework*.
Questo strumento risulta essere, quindi, una valida scelta sia per un primo approccio, sia per lo sviluppo di un progetto di un certo livello.

In fase di installazione non sono molte le cose da configurare. Dopo aver scaricato e decompresso il file nella *root* del *web server* è necessario solamente impostare l'url alla *root* di *CodeIgniter*. Se, per esempio, il *framework* è installato in locale e la cartella in cui è stato scompattato il pacchetto si chiama framework, l'url da inserire sarà la seguente (Mr.Webmaster):

```
$config['base_url'] = "http://localhost/framework";
```

Se, poi, ci si vuole interfacciare con il database si dovrà configurare i parametri del file

```
system/application/config/database.php
```

Si potrà agire sulle seguenti voci:

- ['hostname']: il nome di host del database server;
- ['username']: la username per la connessione al database;
- ['password']: la password per la connessione al database;
- ['database']: il nome del database che si desidera selezionare;
- ['dbdriver']: il tipo di database che si desidera utilizzare, è disponibile il supporto per MySQL, MySQLi, PostgreSQL, ODBC, MSSQL, Sqlite e Oci8;
- ['dbprefix']: il prefisso da associare opzionalmente al nome delle tabelle;
- ['pconnect']: accetta i valori TRUE o FALSE e permette di utilizzare connessioni persistenti o meno;
- ['db_debug']: accetta i valori TRUE o FALSE e permette di visualizzare gli errori prodotti dal database o meno;
- ['cache_on']: accetta i valori TRUE o FALSE e permette di abilitare o meno la cache delle query;
- ['cachedir']: consente di definire il percorso alla cache delle query;
- ['char_set']: permette di definire il set di caratteri da utilizzare per le comunicazioni al database;
- ['dbcollat']: permette di definire la *collation* di caratteri da utilizzare per le comunicazioni al database.

La procedura di installazione e configurazione è terminata. Si può, quindi, notare come questa sia semplice e veloce, e permetta di essere operativi in pochi minuti.

2.4 Goolge Charts

Uno strumento molto utilizzato per la presentazione dei dati analizzati è, sicuramente, il servizio di grafici fornito da Google.
Attraverso le API fornite, risulta essere molto facile introdurre dei grafici all'interno di pagine web.
Un esempio può essere quello del grafico a torta.
Come viene mostrato nel sito di riferimento di Google (Google), un grafico a torta del tipo mostrato in figura si può inserire con il codice seguente.

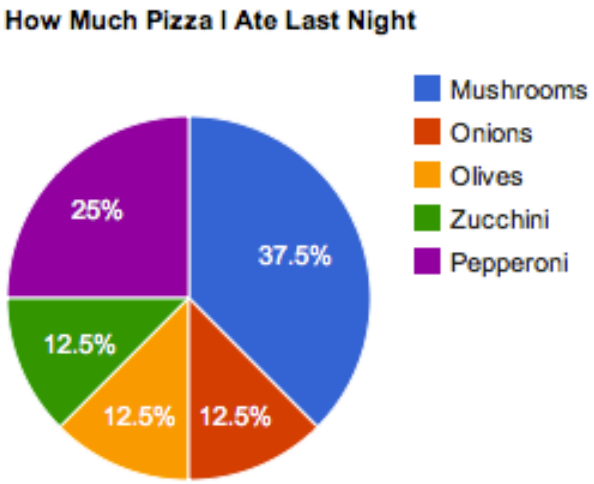

Figure 4: Pie Chart

```
<html>
  <head>
    <!--Load the AJAX API-->
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">

      // Load the Visualization API and the piechart package.
      google.load('visualization', '1.0', {'packages':['corechart']});

      // Set a callback to run when the Google Visualization API is loaded.
      google.setOnLoadCallback(drawChart);

      // Callback that creates and populates a data table,
      // instantiates the pie chart, passes in the data and
      // draws it.
      function drawChart() {

        // Create the data table.
        var data = new google.visualization.DataTable();
        data.addColumn('string', 'Topping');
        data.addColumn('number', 'Slices');
        data.addRows([
          ['Mushrooms', 3],
          ['Onions', 1],
          ['Olives', 1],
          ['Zucchini', 1],
          ['Pepperoni', 2]
        ]);

        // Set chart options
        var options = {'title':'How Much Pizza I Ate Last Night',
                       'width':400,
                       'height':300};

        // Instantiate and draw our chart, passing in some options.
        var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
        chart.draw(data, options);
      }
    </script>
  </head>

  <body>
    <!--Div that will hold the pie chart-->
    <div id="chart_div"></div>
  </body>
</html>
```

Figure 5: Codice Pie Chart

Numerosi sono i tipi di grafici ottenibili. Di seguito solo degli esempi rappresentativi.

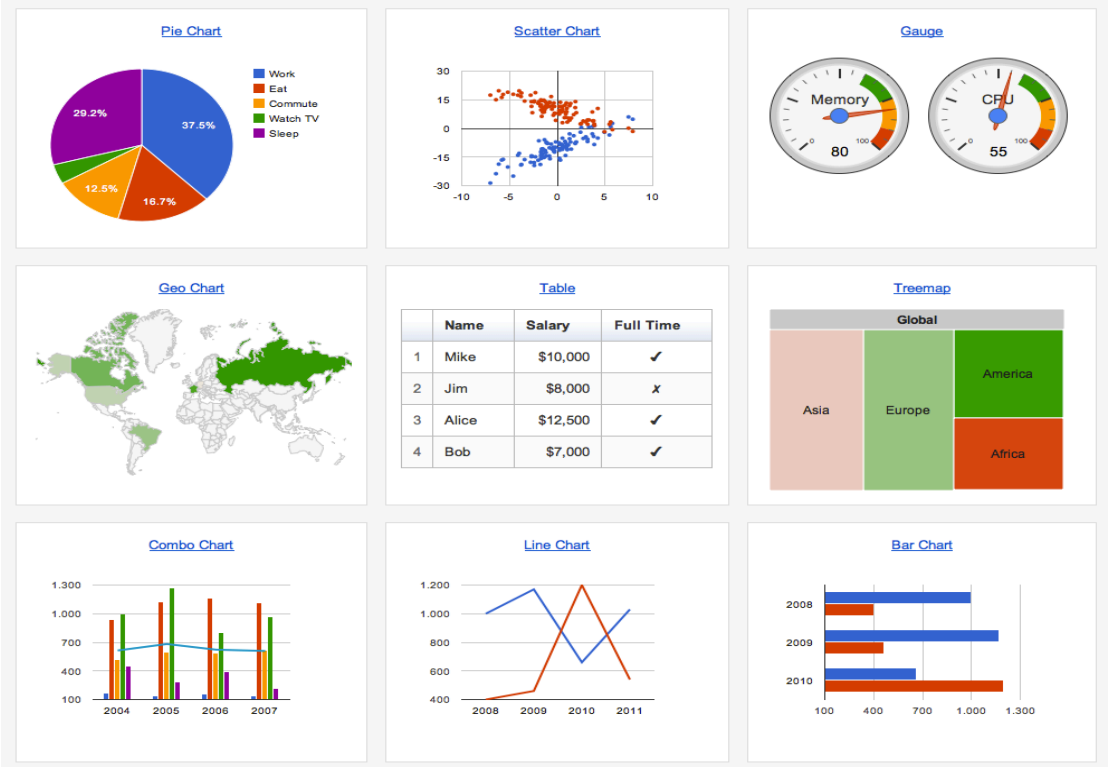

Figure 6: Esempi di grafici realizzabili

Questo tipo di servizio è stato molto utile, perchè ha permesso di ottenere in modo efficace quello che era uno degli obbiettivi della libreria da realizzare: una visualizzazione immediata degli andamenti. Infatti, attraverso questi tipi di grafici è possibile individuare come un certo parametro sta variando nel tempo e permette di averene un riscontro visivo istantaneo.

2.5 Aptana Studio

Per lo sviluppo del progetto è stato utilizzato Aptana Studio, un IDE (integrated development environment) basato su Eclipse. Uno dei punti di forza maggiore di questo software risulta essere l'esistenza di una versione per sistemi Linux, Mac OS e Windows. In azienda, coesistendo tutte e tre queste realtà, questa era una caratteristica importante.

Figure 7: Logo Aptana Studio

Come indicato nel sito di riferimento di Aptana Studio (Appcelerator, Inc) le caratteristiche fondamentali sono:

- HTML, CSS, and JavaScript Code Assist;
- Deployment Wizard;
- Integrated Debugger;
- Git Integration;
- Built-in Terminal;
- IDE Customization.

Aptana Studio è un software rilasciato con licenza open source (GNU General Public License): può essere, quindi, scaricato gratuitamente e utilizzato liberamente anche per scopi commerciali.

2.6 Cruscotto aziendale

Il cruscotto aziendale è uno strumento per la presentazione dei dati utile a prendere decisioni in ambito aziendale. Esso si occupa di sintetizzare il più possibile le informazioni di interesse e di renderle visibili in modo immediato. Il cruscotto aziendale è un indicatore di andamento, permette di avere una rappresentazione, spesso grafica, dei dati ritenuti utili dall'azienda.

L'utente di un tale strumento è sicuramente individuato nella direzione dell'azienda per la sua funzione decisionale.

Il valore che apporta un cruscotto aziendale può essere individuato in due punti:

- Aumentare il grado di consapevolezza della dirigenza;
- Aumento delle capacità di sfruttamento delle risorse.

Un buon cruscotto aziendale deve avere, inoltre, le seguenti caratteristiche:

- Comunicare con immediatezza;
- Grafica minimale che riduca le distrazioni;
- Organizzazione dei dati volta all'usabilità.

Il cruscotto aziendale risulta essere un ottima soluzione alla miriade di dati che altrimenti si dovrebbero analizzare singolarmente. È un agglomerato di informazioni facili da leggere e immediati. Ciò permette di avere una visuale a più ampio raggio e, non ultimo, risparmiare molto tempo. Questo è il punto di forza di tale strumento. Il risvolto della medaglia, d'altra parte, è l'estrema specificità. Infatti, un buon cruscotto aziendale deve rispecchiare a pieno tutte le necessità dell'azienda.
Esistono molte tipologie di cruscotti, anche se, se ne possono individuare principalmente di tre tipi (Dundas Data Visualization Inc., 2012):

I. Cruscotti strategici. Questi cruscotti non hanno bisogno di dati *real-time* e spesso vengono utilizzati per la verifica dello stato di salute di una azienda e per determinare gli obiettivi futuri.
II. Cruscotti analitici. Questi, come si può capire dal nome stesso, assistono l'utente con l'analisi di dati e premettono di capire le cause di ciò che sta accadendo in azienda.
III. Cruscotti operativi. Tali strumenti permettono di monitorare *real-time* l'andamento di qualche parametro particolare. Questi sono più orientati al controllo continuo e quindi generalmente sono caratterizzati da un'interfaccia minimale, in modo da focalizzare l'attenzione.

Esistono molti altri cruscotti e molti altri modi di classificarli, ma per una trattazione più approfondita si rimanda allo studio di testi più specifici (Chen, Manish, Rasmussen, 2009)

La libreria che si intende sviluppare, riprende alcuni degli elementi tipici di un cruscotto aziendale:

- La visualizzazione immediata dei dati attraverso grafici di andamento;
- La selettività dei dati proposti;
- La capacità di adattamento dei dati (nel caso specifico di tipo temporale);
- L'aggiornabilità dei dati.

Tale libreria potrebbe essere quindi intesa come un cruscotto analitico.
È stata vagliata la possibilità di sfruttare software di terze parti per svolgere questo compito, ma la limitatezza del progetto e la presenza delle risorse umane con le capacità di sviluppo già esistenti, ha reso i costi, necessari alla licenza di tali programmi, non sufficientemente giustificati.

Capitolo 3

Libreria

3.0 Libreria

In questo capitolo si andranno ad analizzare in primo luogo la struttura del *database*, con una attenta analisi dello schema logico relazionale. In secondo luogo si presenterà l'interfaccia con cui la libreria sarà resa visibile all'utente. Questa può essere principalmente divisa in due parti:

I. Visualizzazione dei dati del database;
II. Visualizzazione dei dati analizzati, attraverso grafici di andamento.

Questo capitolo, quindi, può essere inteso come una sorta di guida utenti, che spiega le potenzialità e i servizi messi a disposizione.

Le funzionalità richieste alla libreria sono quelle di:

- Aggiornamento del database
- Riepilogo dati del database e relativa visualizzazione
- Analisi dati e visualizzazione attraverso grafici degli andamenti

3.1 Struttura del database utilizzato

Tradedoubler fornisce una grande quantità di dati, di cui solo una parte viene utilizzata dalla libreria. Qui di seguito è presentato lo schema logico relazionale del database utilizzato.

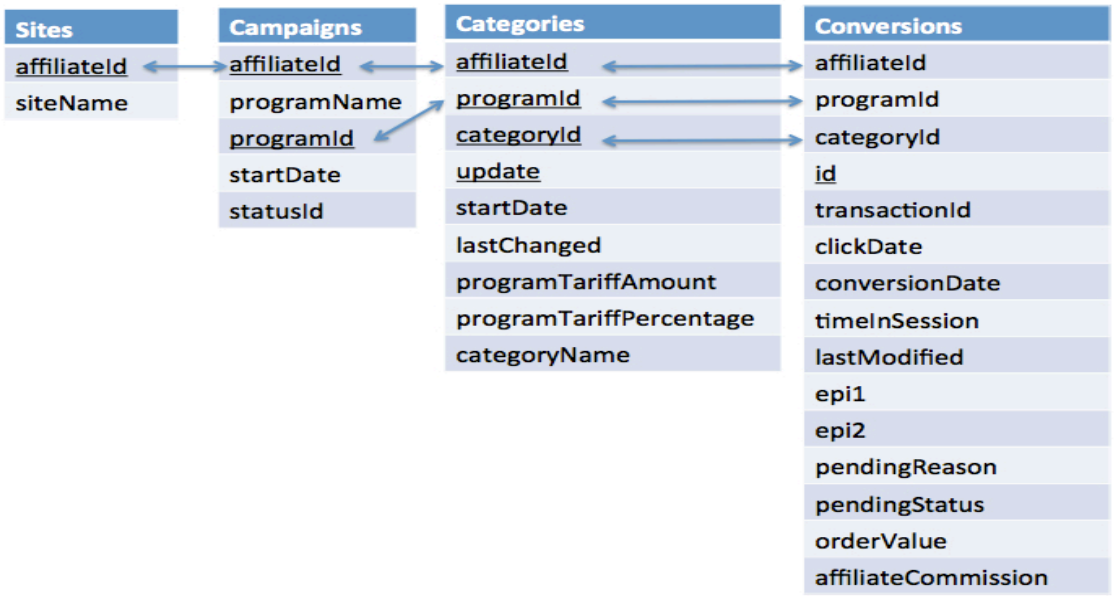

Figure 8: Schema Logico Relazionale del database

3.1.1 Analisi

Analizzando lo schema nelle sue parti più importanti, risulta che il database utilizzato è, quindi, composto da quattro tabelle.

I. Tabella *Sites*

Come si evince facilmente dallo schema relazionale, esistono dei siti cui viene associato un *affiliateId* che li identifica univocamente. Ad ogni sito, poi, corrispondono delle campagne e delle conversioni.

II. Tabella *Campaigns*

Ogni sito ha numerose campagne che possono essere:

- In attesa, se la campagna sta aspettando di essere valutata;
- In considerazione, se la campagna è sotto valutazione;
- Accettato, se la campagna risulta essere approvata;
- Respinto, se la campagna non risulta essere approvata;
- Terminato, se la campagna risulta essere conclusa.

Questi importanti *stati* delle singole campagne vengono memorizzati dall'attributo *statusId*.
Ovviamente ogni campagna viene identificata univocamente da due attributi che costituiscono la chiave dell'entità: *affiliateId* e *programId*.

III. Tabella *Categories*

Ogni campagna può essere divisa in diverse categorie, ognuna delle quali presenta degli elementi che la contraddistinguono.
Gli attributi più importanti delle singole categorie sono:

- *programTariffAmount*, che individua, se presente, la tariffa fissa che viene applicata per ogni conversione relativa a quella categoria di quella campagna;
- *programTariffPercentage,* che individua, se presente, la tariffa a percentuale che viene applicata per ogni conversione relativa a quella categoria di quella campagna;
- *update,* che memorizza la data e l'ora di eventuali modifiche relative a quella determinata categoria;

- *affiliateId, programId, categoryId, update,* che formano la chiave primaria dell'entità.

IV. Tabella *Conversions*

Questa risulta essere la tabella contenente il maggior numero di informazioni in quanto memorizza i dati di tutte le conversioni effettuate.
Gli attributi più importanti delle singole conversioni sono:

- *clickDate,* che memorizza la data e ora del click dell'utente;
- *conversionDate,* che memorizza la data e ora della conversione;
- *affiliateCommission,* che individua la commissione relativa a quella conversione;

3.1.2 Verifica normalizzazione

Il processo di normalizzazione dei dati consiste in una analisi degli schemi di relazione, basato sulle loro dipendenze funzionali e chiavi primarie. L'obiettivo che si vuole raggiungere è quello di minimizzare la ridondanza e le anomalie di aggiornamento dei dati.
Esistono diverse *forme normali* che devono essere soddisfatte per avere un database efficiente.
La *prima forma normale* (1NF) risulta essere soddisfatta in quanto ogni attributo è definito su un dominio di valori atomici ed esiste sempre una chiave primaria.
La *seconda forma normale* (2NF) è soddisfatta. Infatti, tutti gli attributi non-chiave dipendono funzionalmente dalla chiave candidata e, inoltre, è soddisfatta la 1NF.
Anche la *terza forma normale* (3NF) risulta essere soddisfatta poiché non esiste alcun attributo non-chiave che dipenda funzionalmente da un altro attributo non-chiave o da un insieme di attributi non-chiave e, inoltre, è soddisfatta la 2NF.
In ultima analisi, è verificata anche la *forma normale di Boyce Codd* (BCNF) in quanto la 3NF è soddisfatta e per ogni dipendenza funzionale non banale X -> Y, X è una superchiave per R.
Per una trattazione più dettagliata per quanto concerne la normalizzazione si rimanda a (Elmasri, Navathe, 2007)

3.2 Aggiornamento del database

L'aggiornamento del database è manuale e richiede il click su un tasto appositamente dedicato. Questa scelta nasce dal fatto che gli accessi giornalieri al servizio fornito da *Tradedoubler* sono limitati. Un aggiornamento automatico temporalizzato non rientrava quindi nelle specifiche, in quanto questa libreria non è l'unica che accede a tale servizio.

L'aggiornamento avviene contattando i server *Tradedoubler* fornendo come dati:

- un codice che identifica l'azienda cliente;
- il nome del report richiesto;
- un eventuale intervallo temporale;
- il formato con cui restituire il risultato dell'interrogazione.

Si è scelto di far restituire i dati dal server sotto forma di file XML. La libreria, una volta acquisito il file, è in grado di analizzarlo e di popolare il database locale in modo opportuno. Grazie alle possibilità di manipolare file XML fornite dal linguaggio PHP l'operazione risulta essere efficiente.
L'implementazione, per questioni di ottimizzazione, prevede un aggiornamento di tipo incrementale. L'algoritmo è in grado, quindi, di andare a scaricare dai server *Tradedoubler* solamente quei dati che hanno subito una variazione o un nuovo inserimento.

3.3 Riepilogo e visualizzazione dati del database

L'interfaccia grafica sviluppata, in generale, risulta essere molto semplice. Questa scelta progettuale nasce da una esigenza da parte della azienda di essenzialità e facilità d'uso.
La libreria permette la visualizzazione dei dati contenuti nel database dividendo, in base al report richiesto, i dati presenti. Sono disponibili i due report richiesti dalle specifiche tra quelli messi a disposizione da *Tradedoubler*: *Programs Report* e *Breakdown Report*.

I. Per quanto riguarda il *Programs Report,* è stato richiesto dall'azienda di poter selezionare direttamente lo *statusId*. Questa opzione ci permette di ottenere in modo efficace una prima divisione delle campagne a seconda del loro stato.
L'interfaccia, quindi, prevede la possibilità di visualizzare tutti i dati contenuti nel database o di selezionarne solo alcuni per ricerche più specifiche e limitate.

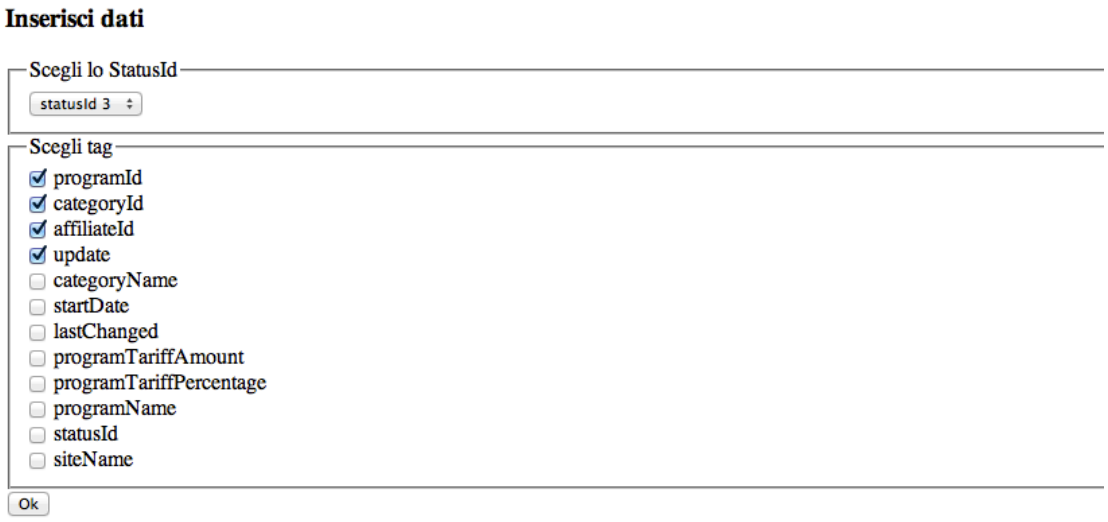

Figure 9: Scelta parametri de visualizzare (Programs Report)

Una volta fatta la scelta, la libreria si occupa di fare la ricerca sul database per visualizzare poi i dati richiesti.

programId	categoryId	affiliateId	update
4366	2	1865376	2011-12-07 19:17:25
4366	2	1932133	2011-07-18 11:06:02
4366	2	1932133	2011-12-07 19:17:24
4366	7	1865376	2011-12-07 19:17:25
4366	7	1932133	2011-07-18 11:06:02
4366	7	1932133	2011-12-07 19:17:24
4366	136718	1865376	2011-12-07 19:17:25
4366	136718	1932133	2011-04-01 00:00:00
4366	136718	1932133	2011-12-07 19:17:24
4366	136720	1865376	2011-12-07 19:17:25
4366	136720	1932133	2011-04-01 00:00:00
4366	136720	1932133	2011-12-07 19:17:24
4366	136722	1865376	2011-12-07 19:17:25
4366	136722	1932133	2011-04-01 00:00:00
4366	136722	1932133	2011-12-07 19:17:24
4366	136724	1865376	2011-12-07 19:17:25
4366	136724	1932133	2011-04-01 00:00:00
4366	136724	1932133	2011-12-07 19:17:24
4366	136726	1865376	2011-12-07 19:17:25
4366	136726	1932133	2011-04-01 00:00:00
4366	136726	1932133	2011-12-07 19:17:24
4366	136728	1865376	2011-12-07 19:17:25
4366	136728	1932133	2011-04-01 00:00:00
4366	136728	1932133	2011-12-07 19:17:24
4366	136730	1865376	2011-12-07 19:17:25

Figure 10: Visualizzazione dati scelti (Programs Report)

II. Il principio di funzionamento e sviluppo per la visualizzazione dei dati del *Breackdown Report* risulta essere in gran parte simile al precedente. Una differenza fondamentale è la possibilità di inserire un intervallo temporale sul quale fare la ricerca.

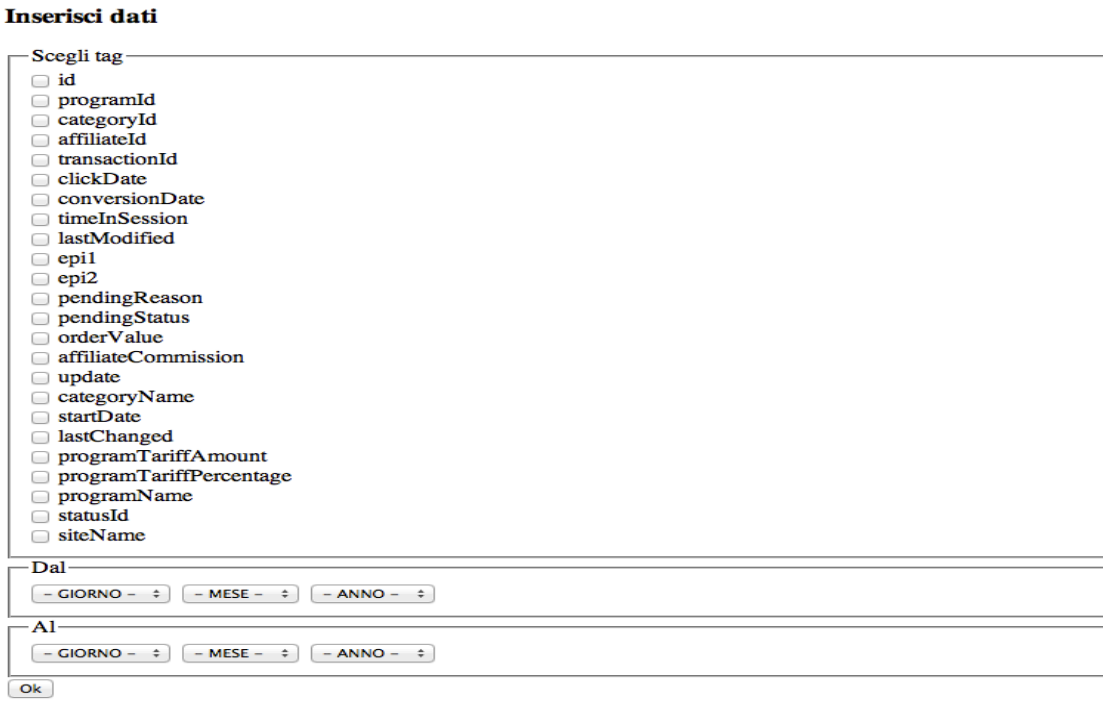

Figure 11: Scelta parametri da visualizzare (Breakdown Report)

Questa differenza è dovuta principalmente alla necessità di rendere le ricerche efficaci e funzionali alle esigenze dell'azienda. Il poter discriminare i dati in base al periodo scelto permette di fare ricerche molto più specifiche e utili soprattutto a valutare gli andamenti temporali.
La visualizzazione dei dati, poi, avviene in modo analogo al precedente.

3.4 Analisi dati e visualizzazione attraverso grafici degli andamenti

Questa risulta essere la parte più importante del progetto in quanto l'obbiettivo del progetto era, non solo la creazione di un archivio con tutto lo storico delle conversioni e delle campagne, ma soprattutto uno strumento che potesse esprimere visivamente in modo efficace gli andamenti temporali dei vari parametri.

Anche in questo caso sono stati divisi i due campi: campagne e conversioni.

3.4.1 Campagne

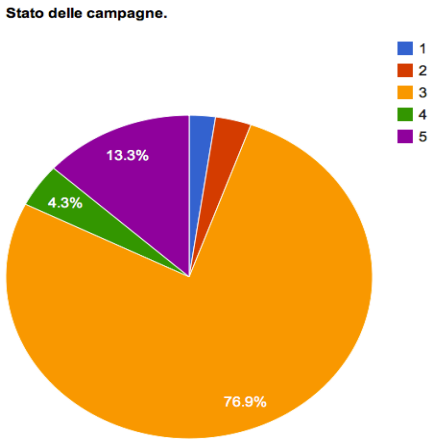

Figure 12: Grafico Stato delle Campagne

Questo è l'unico grafico relativo al *Programs Report* in quanto risulta essere utile visualizzare la percentuale di campagne che sono in ogni stato. Vengono quindi presentati i cinque *statusId* sotto forma di grafico a torta permettendo di avere una visione di insieme dello stato delle campagne.

3.4.2 Conversioni

L'interfaccia, con cui il progetto si presenta, è volutamente molto semplice ed allo stesso tempo efficace. Il menù principale è costituito da due *form*: uno per la scelta del grafico richiesto ed uno per l'inserimento dell'intervallo temporale. Inoltre ogni tipologia di grafico ha il suo corrispettivo valutato come media.

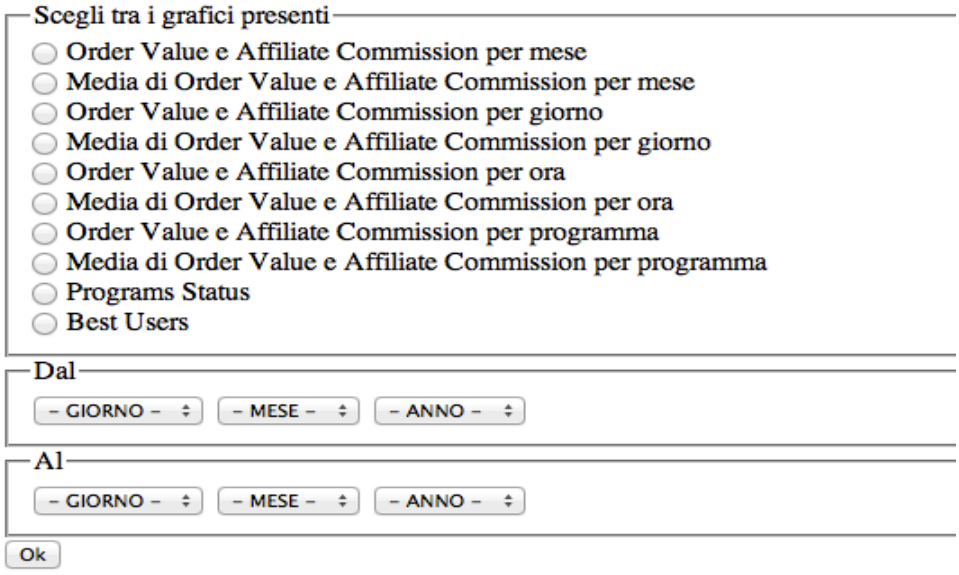

Figure 13: Scelta del grafico (Breakdown Report)

3.4.2.1 Order Value e Affiliate Commission

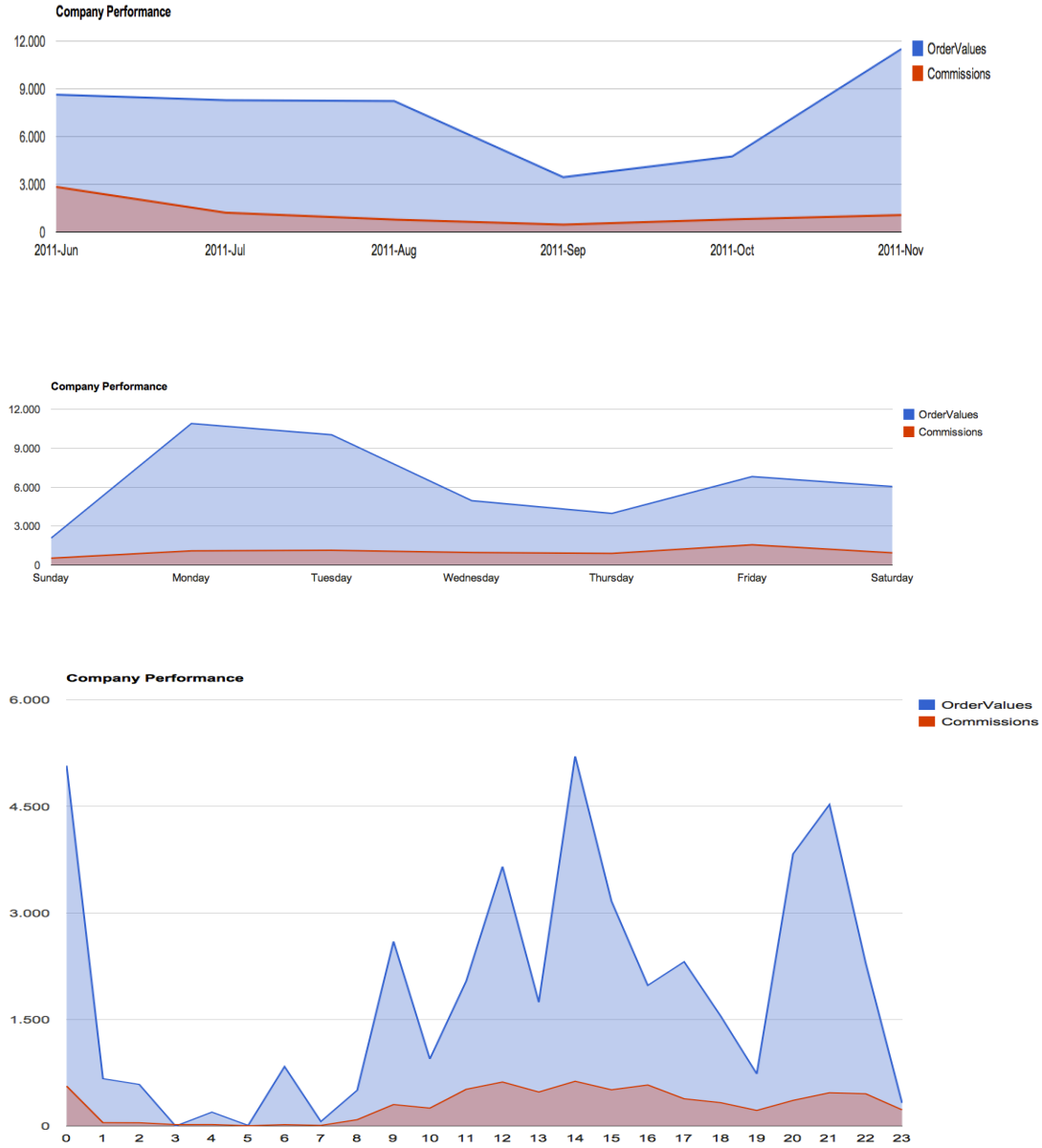

Figure 14: Grafici Order Value and Affiliate Commision

Questi grafici presentano l'andamento degli importi degli ordini fatti dagli utenti e le relative commissioni che spettano all'azienda prendendo come

intervalli temporali, in prima analisi, il mese, successivamente il giorno della settimana e, infine, l'ora del giorno.
Sull'asse delle ordinate presentano la somma in euro di tutti gli ordini fatti nell'unità di tempo di riferimento da parte degli utenti affiliati. Sull'asse delle ascisse, invece, i grafici presentano la scansione temporale.

Questo andamento poi è stato valutato anche in media. Sono state, quindi, calcolate le medie degli importi degli acquisti e le medie dei ricavi per l'azienda sotto forma di commissioni.

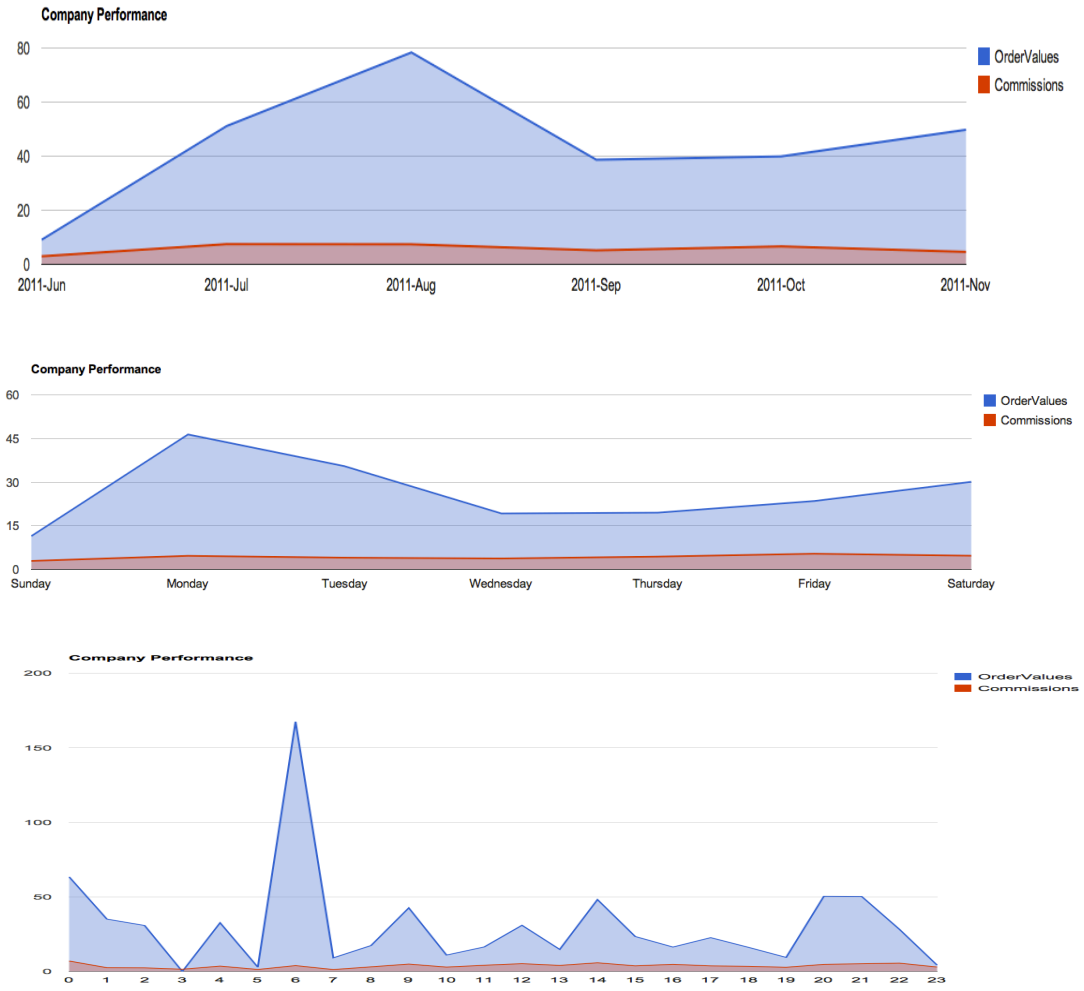

Figure 15: Grafici Order Value and Affiliate Commission in media

Come si può notare le due rappresentazioni, in media e al dettaglio, sono davvero utili perché offrono due punti di vista dello stesso dato. A titolo esemplificativo, infatti, come emerge efficacemente dai grafici, se si

confrontano i valori dei mesi di giugno e agosto, si ha che, a fronte di un valore totale di ordini simili, si hanno medie di acquisto molto diverse.
Questo tipo di rappresentazione dei dati permette di avere una visione molto più ampia sugli andamenti e quindi di poter prendere eventuali misure specifiche per innalzare le vendite.

Un ulteriore grafico si è reso necessario per andare ad identificare questo andamento basato però sul singolo programma.

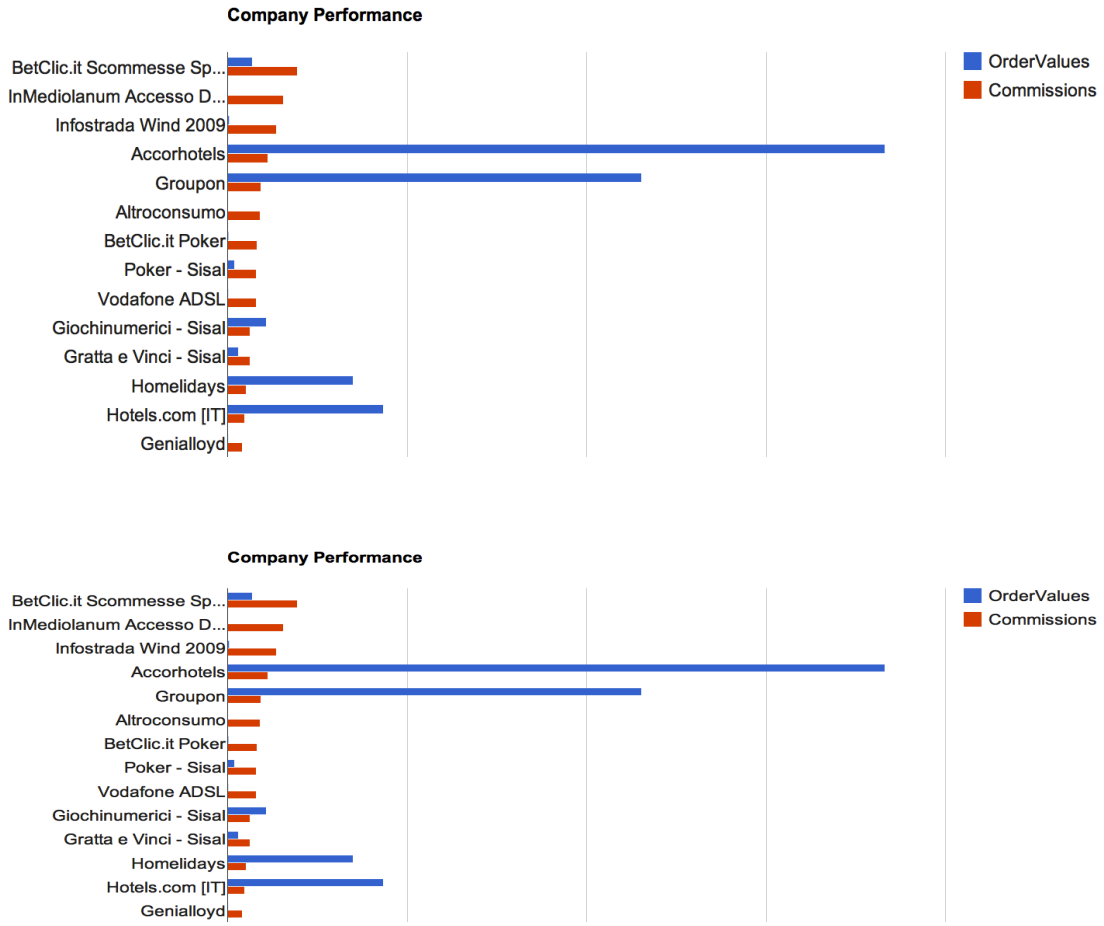

Figure 16: Grafici Order Value and Affiliate Commission per Program

Dato che l'obiettivo era quello di visualizzare i programmi che generano i maggiori introiti, si è deciso di ordinare il grafico in base alle commissioni generate.
Questo tipo di grafico permette di individuare immediatamente i programmi maggiormente remunerativi e quelli che invece non producono, oppure producono in maniera non significativa, introiti.

3.4.2.2 Program status

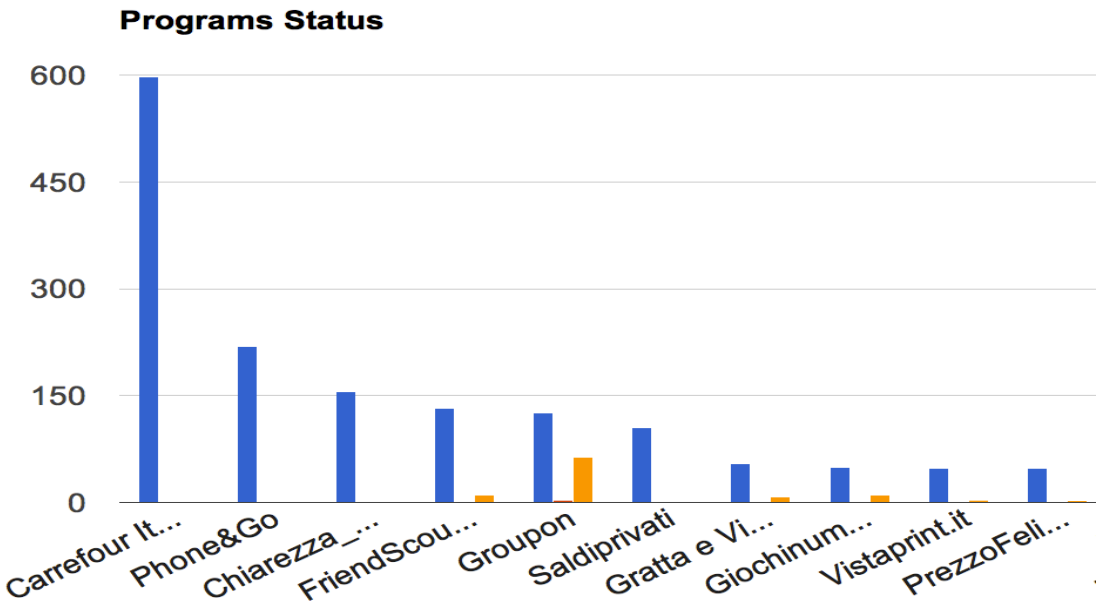

Figure 17: Grafico Program Status

Questo grafico visualizza l'andamento degli stati dei programmi, andando a considerare quanti sono stati approvati, quanti rifiutati e quanti sono in fase di considerazione. Sono ordinati per numero decrescente di programmi approvati.

3.4.2.3 Best Users

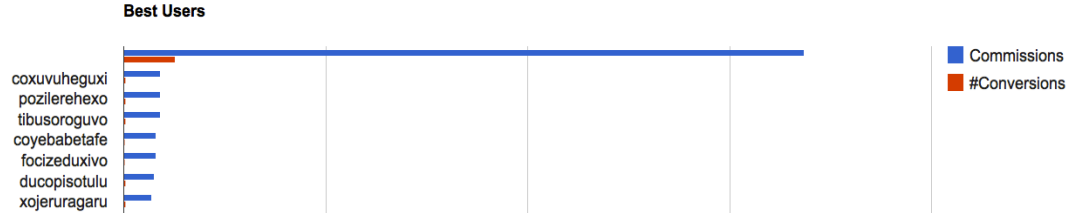

Figure 18: Grafico Best Users

Questo grafico si occupa di visualizzare quali sono gli utenti che generano il maggior numero di conversioni. Questo può servire all'aziende per eventuali

campagne volte a fidelizzare maggiormente gli utenti che effettuano numerose conversioni o incentivare coloro che ne effettuano meno.
In blu vengono visualizzati la somma totale delle commissioni generate, mentre in rosso il numero di conversioni.

Capitolo 4

Analisi del codice della libreria

4.0 Analisi del codice della libreria

Essendo lo scopo principale di questo stage lo sviluppo di una libreria, si vuole, di seguito, analizzare in modo più approfondito le scelte e le modalità di implementazione del file più importante del progetto: *GetTadeDoubler.php*. Questo file non è stato di facile realizzazione in quanto è il vero e proprio cuore della libreria. Esso si occupa di tutto quanto non concerne l'interfaccia grafica e la visualizzazione dei dati. Verranno quindi analizzate le funzioni principali rimandando all'appendice A per una eventuale consultazione dell'intero codice.

4.1 Funzione *get*

```
 9      public function get($service, $tags, $startDate, $endDate,
…  $statusId) {
10          if (strcmp($service, "breakdownReport") == 0) {
11              $myurl = STARTURLBREAKDOWN . "startDate=" . $startDate .
…  "&endDate=" . $endDate . ENDURLBREAKDOWN;
12          } else {
13              $myurl = STARTURLPROGRAMS . $statusId . ENDURLPROGRAMS;
14          }
15          $xml = new DOMDocument();
16          $xml -> load($myurl);
17          $dateLog = date("d/m/y H:i:s");
18          $result = array();
19          $column = $xml -> getElementsByTagName("columns") -> item(0);
20          $tagNodes = $column -> childNodes;
21          $flag = FALSE;
22          foreach ($tagNodes as $tag) {
23              if ($flag) {
24                  $tags[] = $tag -> nodeName;
25                  $flag = FALSE;
26              } else {
27                  $flag = TRUE;
28              }
29          }
30
31          $result['tags'] = $tags;
32          $elementi = $xml -> getElementsByTagName("row");
33          $data[][] = array();
34          foreach ($tags as $tag) {
35              foreach ($elementi as $elemento) {
36                  $nomelemento = $elemento ->
…  getElementsByTagName("$tag");
37                  $nome = $nomelemento -> item(0) -> nodeValue;
38                  $data["$tag"][] = $nome;
39              }
40          }
41          $result['data'] = $data;
42
43          return $result;
44      }
45
```

Figure 19: Codice funzione get

Questa funzione è sicuramente una delle più importanti, infatti, si occupa della comunicazione con il servizio fornito da *Tradedoubler* e della formattazione dei dati ricevuti. Essendoci la necessità di utilizzare entrambi i report per l'aggiornamento, la variabile *$service* si occupa di questo andando a specificare il report da scaricare: *breakdownReport* o *programsReport*. Un'altra variabile importante è *$statusId* perché *Tradedoubler* organizza un *programsReport* in base ad ogni *statusId* delle campagne (si rimanda al paragrafo 3.1.1 per un approfondimento su *satusId*).
Una volta scaricato il file richiesto in formato XML, attraverso la classe DOMDocument di PHP si è andati a formattare i dati rendendoli più facilmente utilizzabili. Tale classe si è rilevata particolarmente efficace in quanto attraverso dei semplici comandi è possibile manipolare file XML anche di grandi dimensioni. Andando prima ad identificare i *tags* e poi i relativi elementi si è potuto creare un array bidimensionale contenente tutti i dati necessari all'aggiornamento del *database*.

4.2 Funzione *getDataFromDb*

```
52      public function getDataFromDb($service, $tags, $startDate, $endDate,
 … $statusId) {
53          $CI = &get_instance();
54          $select = $this -> formatSelect($service, $tags);
55          $arrayStatusId = $CI -> config -> item('statusId');
56
57          if (strcmp($service, "programsReport") == 0) {
58              $CI -> db -> select($select);
59              $CI -> db -> from('categories');
60              $CI -> db -> where('statusId', $statusId);
61              $CI -> db -> join('campaigns', 'campaigns.programId =
 … categories.programId', 'campaigns.affiliateId =
 … categories.affiliateId');
62              $CI -> db -> join('sites', 'sites.affiliateId =
 … campaigns.affiliateId');
63          } else {
64              $CI -> db -> select($select);
65              $CI -> db -> from('conversions');
66              $CI -> db -> where('conversionDate >=', date("Y-m-d",
 … strtotime($startDate)));
67              $CI -> db -> where('conversionDate <=', date("Y-m-d",
 … strtotime($endDate)));
68              $CI -> db -> join('categories', 'conversions.programId =
 … categories.programId', 'conversions.categoryId=categories.categoryId',
 … 'conversions.affiliateId = categories.affiliateId');
69              $CI -> db -> join('campaigns', 'campaigns.programId =
 … categories.programId', 'campaigns.affiliateId =
 … categories.affiliateId');
70              $CI -> db -> join('sites', 'sites.affiliateId =
 … campaigns.affiliateId');
71          }
72
73          $list = $CI -> db -> get();
74          $data[][] = array();
75
76          foreach ($list->result_array() as $row) {
77              foreach ($tags as $tag) {
78                  if (strcmp($tag, "statusId") == 0) {
79                      $data[$tag][] = $arrayStatusId[$row[$tag]];
80                  } else {
81                      $data[$tag][] = $row[$tag];
82                  }
83              }
84          }
85          return $data;
86      }
```

Figure 20: Codice funzione getDataFromDb

Questa funzione si occupa di andare a recuperare i dati dal database. Basandosi sui parametri che vengono passati, crea la *query SQL* necessaria per richiamare i dati voluti. In questo caso, per le *query* al *database*, viene incontro al programmatore il *framework CodeIgniter* mettendo a disposizione degli strumenti che facilitano le interrogazioni. Di facile interpretazione è la

sintassi da utilizzare per questo scopo. Le due *query* che vengono realizzate da questa funzione sono create rispettivamente dai comandi delle linee 58-62 e 64-70.
Il risultato della *query* , che si ottiene con il comando della linea 73, deve essere successivamente elaborato per essere più facilmente utilizzabile dalle altre funzioni (linee76 -84).

4.3 Funzione *getTagsFromDb*

```
145     public function getTagsFromDb() {
146
147         $CI = &get_instance();
148         $fields = array();
149         $sql = "DESCRIBE conversions";
150         $desc = $CI -> db -> query($sql) -> result();
151         foreach ($desc as $tag) {
152             $fields["breakdownReport"][] = $tag -> Field;
153         }
154         $tables = array("categories", "campaigns", "sites");
155         $fields["programsReport"] = array();
156         foreach ($tables as $table) {   $sql = "DESCRIBE " . $table;
157             $desc = $CI -> db -> query($sql) -> result();
158
159             foreach ($desc as $tag) {
160                 if (!in_array($tag -> Field, $fields["programsReport"]))
  …  {
161                     $fields["programsReport"][] = $tag -> Field;
162                 }
163                 if (!in_array($tag -> Field,
  …  $fields["breakdownReport"])) {
164                     $fields["breakdownReport"][] = $tag -> Field;
165                 }
166             }
167         }
168         return $fields;
169     }
```

Figure 21: Codice funzione getTagsFromDb

Questa funzione si occupa di andare ad analizzare tutti gli attributi di tutte le tabelle del *database.* Questo, ovviamente, si ottiene sfruttando una delle *query* SQL più importanti, cioè il "*DESCRIBE table*". Andando ad utilizzare gli strumenti di interfacciamento con il database messi a disposizione dal *framework* si compiono una serie di interrogazioni fino ad ottenere una completa descrizione dei parametri di tutte le tabelle.

4.4 Funzioni *insertUpdateCampaigns* e *insertCampaigns*

```
241     public function insertUpdateCampaigns($data, $statusId) {
242         $tabella = "campaigns";
243         $CI = &get_instance();
244         $CI -> db -> select('affiliateId , programId');
245         $list = $CI -> db -> get($tabella);
246         $hashList = array();
247         foreach ($list->result() as $result) {
248             $hashList[$result -> affiliateId][$result -> programId] = 1;
249         }
250 
251         for ($index = 0; $index <= (sizeof($data["affiliateId"]) - 1);
  … $index++) {
252             if (empty($hashList)) {
253                 $hashList = $this -> insertCampaigns($data, $index,
  … $hashList, $statusId);
254             } else {
255                 if ((array_key_exists($data["affiliateId"][$index],
  … $hashList))) {
256                     if (!(array_key_exists($data["programId"][$index],
  … $hashList[$data["affiliateId"][$index]]))) {
257                         $hashList = $this -> insertCampaigns($data,
  … $index, $hashList, $statusId);
258                     }
259                 } else {
260                     $hashList = $this -> insertCampaigns($data, $index,
  … $hashList, $statusId);
261                 }
262             }
263         }
264     }
265 
266     public function insertCampaigns($data, $index, $hashList, $statusId)
  … {
267         $tabella = "campaigns";
268         $CI = &get_instance();
269         $insert = array('affiliateId' => $data["affiliateId"][$index],
  … 'programId' => $data["programId"][$index], 'programName' =>
  … $data["programName"][$index], 'startDate' =>
  … $data["applicationDate"][$index], 'statusId' => $statusId, );
270         $CI -> db -> insert($tabella, $insert);
271 
  … $hashList[$data["affiliateId"][$index]][$data["programId"][$index]] = 1;
272         return $hashList;
273 
274     }
```

Figure 22: Codice funzioni insertUpdateCampaigns e insertCampaigns

Queste due funzioni si occupano dell'update della tabella *Campaigns*. Si prende in esame questa tabella per facilità di spiegazione, ma anche le altre tabelle vengono aggiornate con lo stesso principio. Si rimanda all'Appendice A per il codice completo della libreria.
La funzione *insertCampaigns* ha il compito vero e proprio di inserire i dati all'interno del *database*. Anche in questo caso il *framework* ci fornisce uno

strumento veloce per interagire con il database. Il comando della linea 270 determina il reale inserimento del dato.
Prima di inserire i dati, però, bisogna verificare che non siano già presenti nel *database*. Di questo si occupa la funzione *insertUpdateCampaigns*. Il principio di funzionamento si basa, fondamentalmente, sul verificare che la chiave dell'elemento che stiamo andando ad inserire non sia già presente. Se così fosse, essendo la chiave unica per ogni elemento, vorrebbe dire che l'elemento da inserire deve essere scartato in quanto già presente nel *database*.

4.5 Funzione orderValueAndCommissionPerProgram

```
463     public function orderValueAndCommissionPerProgram($startDate,
…   $endDate, $media) {
464         $CI = &get_instance();
465         $CI -> db -> select("conversions.affiliateCommission,
…   conversions.orderValue, conversions.conversionDate,
…   conversions.programId, campaigns.programName FROM conversions");
466         $CI -> db -> where('conversions.conversionDate >=',
…   date("Y-m-d", strtotime($startDate)));
467         $CI -> db -> where('conversions.conversionDate <=',
…   date("Y-m-d", strtotime($endDate)));
468         $CI -> db -> join('categories', 'conversions.programId =
…   categories.programId AND conversions.categoryId=categories.categoryId
…   AND conversions.affiliateId = categories.affiliateId');
469         $CI -> db -> join('campaigns', 'campaigns.programId =
…   categories.programId AND campaigns.affiliateId =
…   categories.affiliateId');
470         $CI -> db -> join('sites', 'sites.affiliateId =
…   campaigns.affiliateId');
471         $query = $CI -> db -> get();
472         $data = array();
473         if ($media != 0)
474             $counter = array();
475         foreach ($query->result() as $result) {
476             $program = $result -> programName;
477             ;
478             if (isset($data[$program])) {
479                 $data[$program]["totalAffiliateCommission"] += $result
…   -> affiliateCommission;
480                 $data[$program]["totalOrderValue"] += $result ->
…   orderValue;
481                 if ($media != 0)
482                     $counter[$program]++;
483             } else {
484                 $data[$program]["totalAffiliateCommission"] = $result ->
…   affiliateCommission;
485                 $data[$program]["totalOrderValue"] = $result ->
…   orderValue;
486                 if ($media != 0)
487                     $counter[$program] = 1;
488             }
489         }
490         if ($media != 0)
491             $data = $this -> media($data, $counter);
492         return $this -> arraySort($data);
493
494     }
```

Figure 23: Codice funzione orderValueAndCommissionPerProgram

In ultima analisi viene fornita la spiegazione dell'implementazione della manipolazione operata sui dati per renderli, poi, utilizzabili attraverso il servizio fornito da *Google Charts*. Questi dati, in particolare, saranno utilizzati per visualizzare un grafico che metta in evidenza l'andamento degli ordini totali raffrontato all'andamento delle commissioni generate. Per realizzare questo, in primo luogo, viene interrogato il *database* utilizzando gli strumenti del *framework*. Attraverso un *select* si ricavano, quindi, tutti gli *orderValue* e *affiliateCommission* per ogni campagna. Il passo successivo risulta essere relativamente semplice, infatti basterà andare a sommare tutti i valori per ogni singola campagna ottenendo così i dati necessari.
La manipolazione dei dati per gli altri grafici seguono, ovviamente con le dovute modifiche specifiche, la medesima linea di sviluppo. Si rimanda, quindi, il lettore all'appendice A per la visione dell'intero listato della libreria.

Capitolo 5

Conclusioni

Conclusioni

Lo stage svolto presso Horizon ADV s.r.l., può essere diviso in due parti che corrispondono anche alla scansione temporale del progetto:

- Studio dell'ambiente di sviluppo e dei linguaggi necessari alla realizzazione del progetto;

- Implementazione vera e propria della libreria, del *database* richiesti e di una interfaccia *web based*;

Nel presente elaborato sono stati presentati sia gli strumenti utilizzati, che la logica delle decisioni prese in ambito progettuale. È stata fornita una descrizione del database realizzato e di tutti i grafici di andamento richiesti dall'azienda.
Dato il forte carattere "pratico" dello stage, volto più alla realizzazione che allo studio teorico, è stato deciso di rendere disponibile anche il codice di implementazione del file più importante del progetto: la libreria.
Un possibile sviluppo sarebbe quello di introdurre la provenienza (social network, siti internet, blog) degli utenti che generano una conversione, andando così ad individuare con maggior precisione dove si generano i migliori bacini d'utenza.
Molte sono le funzioni sviluppabili, in quanto ogni grafico in più risulterebbe uno strumento aggiuntivo a disposizione dell'azienda per meglio comprendere gli andamenti dei parametri d'interesse e per individuare, di conseguenza, eventuali azioni da intraprendere.
Gli obiettivi concordati con l'azienda risultano essere stati tutti raggiunti. La libreria, l'interfaccia grafica e il *database* sono pienamente funzionanti e utilizzabili dall'azienda.

Appendice A

Codice libreria

```php
<?php

class GetTradeDoubler {

	public function __construct() {

	}

	public function get($service, $tags, $startDate, $endDate, $statusId) {
		if (strcmp($service, "breakdownReport") == 0) {
			$myurl = STARTURLBREAKDOWN . "startDate=" . $startDate . "&endDate=" . $endDate . ENDURLBREAKDOWN;
		} else {
			$myurl = STARTURLPROGRAMS . $statusId . ENDURLPROGRAMS;
		}
		$xml = new DOMDocument();
		$xml -> load($myurl);
		$dateLog = date("d/m/y H:i:s");
		$result = array();
		$column = $xml -> getElementsByTagName("columns") -> item(0);
		$tagNodes = $column -> childNodes;
		$flag = FALSE;
		foreach ($tagNodes as $tag) {
			if ($flag) {
				$tags[] = $tag -> nodeName;
				$flag = FALSE;
			} else {
				$flag = TRUE;
			}
		}

		$result['tags'] = $tags;
		$elementi = $xml -> getElementsByTagName("row");
		$data[][] = array();
		foreach ($tags as $tag) {
			foreach ($elementi as $elemento) {
				$nomelemento = $elemento -> getElementsByTagName("$tag");
				$nome = $nomelemento -> item(0) -> nodeValue;
				$data["$tag"][] = $nome;
			}
		}
		$result['data'] = $data;

		return $result;
	}









	public function getDataFromDb($service, $tags, $startDate, $endDate, $statusId) {
		$CI = &get_instance();
		$select = $this -> formatSelect($service, $tags);
		$arrayStatusId = $CI -> config -> item('statusId');

		if (strcmp($service, "programsReport") == 0) {
			$CI -> db -> select($select);
			$CI -> db -> from('categories');
			$CI -> db -> where('statusId', $statusId);
			$CI -> db -> join('campaigns', 'campaigns.programId = categories.programId', 'campaigns.affiliateId = categories.affiliateId');
			$CI -> db -> join('sites', 'sites.affiliateId = campaigns.affiliateId');
		} else {
			$CI -> db -> select($select);
			$CI -> db -> from('conversions');
			$CI -> db -> where('conversionDate >=', date("Y-m-d", strtotime($startDate)));
			$CI -> db -> where('conversionDate <=', date("Y-m-d", strtotime($endDate)));
			$CI -> db -> join('categories', 'conversions.programId = categories.programId', 'conversions.categoryId=categories.categoryId', 'conversions.affiliateId = categories.affiliateId');
			$CI -> db -> join('campaigns', 'campaigns.programId = categories.programId', 'campaigns.affiliateId = categories.affiliateId');
			$CI -> db -> join('sites', 'sites.affiliateId = campaigns.affiliateId');
		}

		$list = $CI -> db -> get();
		$data[][] = array();

		foreach ($list->result_array() as $row) {
			foreach ($tags as $tag) {
				if (strcmp($tag, "statusId") == 0) {
					$data[$tag][] = $arrayStatusId[$row[$tag]];
				} else {
```

```
                                        $data[$tag][] = $row[$tag];
                                }
                        }
                }
                return $data;
        }

        public function formatSelect($service, $tags) {
                $CI = &get_instance();
                $CI -> load -> config('getTradeDoublerDb');
                $sites = $CI -> config -> item('sites');
                $campaigns = $CI -> config -> item('campaigns');
                $categories = $CI -> config -> item('categories');
                $conversions = $CI -> config -> item('conversions');
                $select = "";
                $flag = FALSE;
                foreach ($tags as $tag) {
                        if (strcmp($service, "breakdownReport") == 0) {
                                if (in_array($tag, $conversions)) {
                                        if (empty($select))
                                                $select = "conversions." . $tag;
                                        else
                                                $select = $select . ", conversions." . $tag;
                                } else {$flag = TRUE;
                                }
                        } else {$flag = TRUE;
                        }
                        if ($flag) {
                                if (in_array($tag, $categories)) {
                                        if (empty($select))
                                                $select = "categories." . $tag;
                                        else
                                                $select = $select . ", categories." . $tag;
                                } else {
                                        if (in_array($tag, $campaigns)) {
                                                if (empty($select))
                                                        $select = "campaigns." . $tag;
                                                else
                                                        $select = $select . ", campaigns." . $tag;
                                        } else {
                                                if (empty($select))
                                                        $select = "sites." . $tag;
                                                else
                                                        $select = $select . ", sites." . $tag;

                                        }
                                }
                        }
                        $flag = FALSE;
                }
                return $select;
        }









        public function getTagsFromDb() {

                $CI = &get_instance();
                $fields = array();
                $sql = "DESCRIBE conversions";
                $desc = $CI -> db -> query($sql) -> result();
                foreach ($desc as $tag) {
                        $fields["breakdownReport"][] = $tag -> Field;
                }
                $tables = array("categories", "campaigns", "sites");
                $fields["programsReport"] = array();
                foreach ($tables as $table) {   $sql = "DESCRIBE " . $table;
                        $desc = $CI -> db -> query($sql) -> result();

                        foreach ($desc as $tag) {
                                if (!in_array($tag -> Field, $fields["programsReport"])) {
                                        $fields["programsReport"][] = $tag -> Field;
                                }
                                if (!in_array($tag -> Field, $fields["breakdownReport"])) {
                                        $fields["breakdownReport"][] = $tag -> Field;
                                }
                        }
                }
```

```
168                     return $fields;
169             }
170
171             public function insertProgramsReport($statusId) {
172                     $data = $this -> get("programsReport", array(), 0, 0, $statusId);
173                     $data = $data["data"];
174                     $this -> insertUpdateSites($data);
175                     $this -> insertUpdateCampaigns($data, $statusId);
176                     $this -> insertUpdateCategories($data);
177             }
178
179             public function insertBrakdownReport() {
180                     $this -> insertUpdateConversions();
181             }
182
183             public function insertUpdateSites($data) {
184                     $tabella = "sites";
185                     $CI = &get_instance();
186                     $CI -> db -> select('affiliateId');
187                     $list = $CI -> db -> get($tabella);
188                     $arrayAffiliateId = array();
189                     foreach ($list->result() as $result) {
190                             $arrayAffiliateId[$result -> affiliateId] = 1;
191                     }
192
193                     for ($index = 0; $index <= (sizeof($data["affiliateId"]) - 1); $index++) {
194                             if (empty($arrayAffiliateId)) {
195                                     $this -> insertSites($data, $index);
196                                     $arrayAffiliateId[$data["affiliateId"][$index]] = 1;
197                             } else {
198                                     if (!(array_key_exists($data["affiliateId"][$index],
    $arrayAffiliateId))) {
199                                             $this -> insertSites($data, $index);
200                                             $arrayAffiliateId[$data["affiliateId"][$index]] = 1;
201                                     }
202
203                             }
204
205                     }
206             }
207
208             public function insertSites($data, $index) {
209                     $tabella = "sites";
210                     $CI = &get_instance();
211                     $insert = array('affiliateId' => $data["affiliateId"][$index], 'siteName' => $data
    ["siteName"][$index], );
212                     $CI -> db -> insert($tabella, $insert);
213             }
214
215             public function insertUpdateCampaigns($data, $statusId) {
216                     $tabella = "campaigns";
217                     $CI = &get_instance();
218                     $CI -> db -> select('affiliateId , programId');
219                     $list = $CI -> db -> get($tabella);
220                     $hashList = array();
221                     foreach ($list->result() as $result) {
222                             $hashList[$result -> affiliateId][$result -> programId] = 1;
223                     }
224
225                     for ($index = 0; $index <= (sizeof($data["affiliateId"]) - 1); $index++) {
226                             if (empty($hashList)) {
227                                     $hashList = $this -> insertCampaigns($data, $index, $hashList,
    $statusId);
228                             } else {
229                                     if ((array_key_exists($data["affiliateId"][$index], $hashList))) {
230                                             if (!(array_key_exists($data["programId"][$index],
    $hashList[$data["affiliateId"][$index]]))) {
231                                                     $hashList = $this -> insertCampaigns($data,
    $index, $hashList, $statusId);
232                                             }
233                                     } else {
234                                             $hashList = $this -> insertCampaigns($data, $index,
    $hashList, $statusId);
235                                     }
236                             }
237                     }
238             }
239
240             public function insertCampaigns($data, $index, $hashList, $statusId) {
241                     $tabella = "campaigns";
242                     $CI = &get_instance();
243                     $insert = array('affiliateId' => $data["affiliateId"][$index], 'programId' =>
    $data["programId"][$index], 'programName' => $data["programName"][$index], 'startDate' => $data
    ["applicationDate"][$index], 'statusId' => $statusId, );
244                     $CI -> db -> insert($tabella, $insert);
245                     $hashList[$data["affiliateId"][$index]][$data["programId"][$index]] = 1;
246                     return $hashList;
```

```
247
248         }
249
250         public function insertUpdateCategories($data) {
251                 echo "<pre>";
252                 $tabella = "categories";
253                 $prova = (string) date("Y-m-d");
254                 $CI = &get_instance();
255                 $CI -> db -> select('categories.affiliateId, categories.programId ,
    categories.categoryId , categories.programTariffAmount,
256                                 categories.programTariffPercentage, max(categories.update) as
    lastUpdate from categories group by categories.programId , categories.categoryId,
    categories.affiliateId');
257                 $list = $CI -> db -> get();
258                 $hashList = array();
259                 foreach ($list->result() as $result) {
260                         $hashList[$result -> affiliateId][$result -> programId][$result ->
    categoryId]["programTariffAmount"] = ((float)$result -> programTariffAmount);
261                         $hashList[$result -> affiliateId][$result -> programId][$result ->
    categoryId]["programTariffPercentage"] = ((float)$result -> programTariffPercentage);
262                 }
263                 for ($index = 0; $index <= (sizeof($data["programId"]) - 1); $index++) {
264                         if (empty($hashList)) {
265                                 $hashList = $this -> insertCategories($data, $index, $hashList);
266                         } else {
267                                 if (array_key_exists($data["affiliateId"][$index], $hashList)) {
268                                         if (array_key_exists($data["programId"][$index], $hashList
    [$data["affiliateId"][$index]])) {
269                                                 if (array_key_exists($data["eventIdView"][$index],
    $hashList[$data["affiliateId"][$index]][$data["programId"][$index]])) {
270                                                         if ((floatval($data["programTariffAmount"]
    [$index]) - $hashList[$data["affiliateId"][$index]][$data["programId"][$index]][$data
    ["eventIdView"][$index]]["programTariffAmount"] != 0) || (floatval($data
    ["programTariffPercentage"][$index]) - $hashList[$data["affiliateId"][$index]][$data["programId"]
    [$index]][$data["eventIdView"][$index]]["programTariffPercentage"] != 0)) {
271                                                                 $hashList = $this ->
    insertCategories($data, $index, $hashList);
272
273                                                         }
274                                                 } else {
275                                                         $hashList = $this -> insertCategories
    ($data, $index, $hashList);
276                                                 }
277                                         } else {
278                                                 $hashList = $this -> insertCategories($data,
    $index, $hashList);
279                                         }
280                                 } else {
281                                         $hashList = $this -> insertCategories($data, $index,
    $hashList);
282                                 }
283                         }
284                 }
285                 echo "</pre>";
286         }
287
288         public function insertCategories($data, $index, $hashList) {
289                 $tabella = "categories";
290                 $CI = &get_instance();
291                 $insert = array('programId' => $data["programId"][$index], 'categoryId' => $data
    ["eventIdView"][$index], 'affiliateId' => $data["affiliateId"][$index], 'update' => date("Y-m-d
    H:i:s"), 'categoryName' => $data["event"][$index], 'startDate' => $data["eventLastModified"]
    [$index], 'lastChanged' => $data["lastModified"][$index], 'programTariffAmount' => $data
    ["programTariffAmount"][$index], 'programTariffPercentage' => $data["programTariffPercentage"]
    [$index], );
292                 $CI -> db -> insert($tabella, $insert);
293                 $hashList[$data["affiliateId"][$index]][$data["programId"][$index]][$data
    ["eventIdView"][$index]]["programTariffAmount"] = floatval($data["programTariffAmount"][$index]);
294                 $hashList[$data["affiliateId"][$index]][$data["programId"][$index]][$data
    ["eventIdView"][$index]]["programTariffPercentage"] = floatval($data["programTariffPercentage"]
    [$index]);
295
296                 return $hashList;
297         }
298
299         public function insertUpdateConversions() {
300                 $tabella = "conversions";
301                 $CI = &get_instance();
302                 $CI -> db -> select_max('conversionDate');
303                 $query = $CI -> db -> get($tabella);
304                 $startDate = $query -> row() -> conversionDate;
305                 if (strcmp($startDate, "") == 0)
306                         $startDate = "2011-05-25";
307                 $data = $this -> get("breakdownReport", array(), date("d/m/y", strtotime
    ($startDate)), date("d/m/y"), 0);
308                 $data = $data["data"];
309                 $CI -> db -> select('programId , categoryId , transactionId, affiliateId');
```

```
310                 $list = $CI -> db -> get($tabella);
311                 $hashList = array();
312                 foreach ($list->result() as $result) {
313                         $hashList[$result -> affiliateId][$result -> programId][$result ->
    categoryId][$result -> transactionId] = 1;
314                 }
315                 for ($index = 0; $index <= (sizeof($data["programId"]) - 1); $index++) {
316                         if (empty($hashList))
317                                 $hashList = $this -> insertConversions($data, $index, $hashList);
318                         else {
319                                 if (array_key_exists($data["siteId"][$index], $hashList)) {
320                                         if (array_key_exists($data["programId"][$index], $hashList
    [$data["siteId"][$index]])) {
321                                                 if (array_key_exists($data["eventId"][$index],
    $hashList[$data["siteId"][$index]][$data["programId"][$index]])) {
322                                                         if (!(array_key_exists($data["leadNR"]
    [$index], $hashList[$data["siteId"][$index]][$data["programId"][$index]][$data["eventId"]
    [$index]])) && !(array_key_exists($data["orderNR"][$index], $hashList[$data["siteId"][$index]]
    [$data["programId"][$index]][$data["eventId"][$index]]))) {
323                                                                 $hashList = $this ->
    insertConversions($data, $index, $hashList);
324                                                         }
325                                                 } else {
326                                                         $hashList = $this -> insertConversions
    ($data, $index, $hashList);
327                                                 }
328                                         } else {
329                                                 $hashList = $this -> insertConversions($data,
    $index, $hashList);
330                                         }
331                                 } else {
332                                         $hashList = $this -> insertConversions($data, $index,
    $hashList);
333                                 }
334                         }
335                 }
336         }
337
338         public function insertConversions($data, $index, $hashList) {
339                 $tabella = "conversions";
340                 $CI = &get_instance();
341                 if (!empty($data["leadNR"][$index])) {
342                         $transactionId = $data["leadNR"][$index];
343                         $hashList[$data["siteId"][$index]][$data["programId"][$index]][$data
    ["eventId"][$index]][$data["leadNR"][$index]] = 1;
344                 } else {
345                         $transactionId = $data["orderNR"][$index];
346                         $hashList[$data["siteId"][$index]][$data["programId"][$index]][$data
    ["eventId"][$index]][$data["orderNR"][$index]] = 1;
347                 }
348                 $insert = array('programId' => $data["programId"][$index], 'categoryId' => $data
    ["eventId"][$index], 'affiliateId' => $data["siteId"][$index], 'transactionId' => $transactionId,
    'timeInSession' => $data["timeInSession"][$index], 'clickDate' => $data["timeOfVisit"][$index],
    'conversionDate' => $data["timeOfEvent"][$index], 'lastModified' => $data["lastModified"][$index],
    'epi1' => $data["epi1"][$index], 'epi2' => $data["epi2"][$index], 'pendingReason' => $data
    ["pendingReason"][$index], 'pendingstatus' => $data["pendingStatus"][$index], 'orderValue' =>
    $data["orderValue"][$index], 'affiliateCommission' => $data["affiliateCommission"][$index], );
349                 $CI -> db -> insert($tabella, $insert);
350                 return $hashList;
351         }
352
353         public function programPerSite() {
354
355                 $CI = &get_instance();
356                 $CI -> db -> select(SELECTPROGRAMPERSITE);
357                 $query = $CI -> db -> get();
358                 $data = array();
359                 foreach ($query->result() as $result) {
360                         $data[1][] = $result -> num;
361                         $data[0][] = $result -> siteName;
362                 }
363                 return $data;
364         }
365
366         public function orderValueAndCommissionChart($perDate, $perHour, $startDate, $endDate,
    $media) {
367                 $CI = &get_instance();
368                 $CI -> db -> select("conversions.affiliateCommission, conversions.orderValue,
    conversions.conversionDate  FROM conversions");
369                 $CI -> db -> where('conversions.conversionDate >=', date("Y-m-d", strtotime
    ($startDate)));
370                 $CI -> db -> where('conversions.conversionDate <=', date("Y-m-d", strtotime
    ($endDate)));
371                 $query = $CI -> db -> get();
372                 if ($media != 0)
373                         $counter = array();
374                 $data = array();
```

```
375                     if ($perDate != 0) {
376                             $data["Sunday"]["totalAffiliateCommission"] = 0;
377                             $data["Sunday"]["totalOrderValue"] = 0;
378                             $data["Monday"]["totalAffiliateCommission"] = 0;
379                             $data["Monday"]["totalOrderValue"] = 0;
380                             $data["Tuesday"]["totalAffiliateCommission"] = 0;
381                             $data["Tuesday"]["totalOrderValue"] = 0;
382                             $data["Wednesday"]["totalAffiliateCommission"] = 0;
383                             $data["Wednesday"]["totalOrderValue"] = 0;
384                             $data["Thursday"]["totalAffiliateCommission"] = 0;
385                             $data["Thursday"]["totalOrderValue"] = 0;
386                             $data["Friday"]["totalAffiliateCommission"] = 0;
387                             $data["Friday"]["totalOrderValue"] = 0;
388                             $data["Saturday"]["totalAffiliateCommission"] = 0;
389                             $data["Saturday"]["totalOrderValue"] = 0;
390                             if ($media != 0) {
391                                     $counter["Sunday"] = 0;
392                                     $counter["Monday"] = 0;
393                                     $counter["Tuesday"] = 0;
394                                     $counter["Wednesday"] = 0;
395                                     $counter["Thursday"] = 0;
396                                     $counter["Friday"] = 0;
397                                     $counter["Saturday"] = 0;
398                             }
399                     } else if ($perHour != 0) {
400                             for ($i = 0; $i < 24; $i++) {
401                                     $data[$i]["totalAffiliateCommission"] = 0;
402                                     $data[$i]["totalOrderValue"] = 0;
403                                     if ($media != 0)
404                                             $counter[$i] = 0;
405                             }
406                     }
407
408                     foreach ($query->result() as $result) {
409                             if ($perDate != 0) {
410                                     $index = date("l", strtotime($result -> conversionDate));
411                             } else if ($perHour != 0) {
412                                     $index = date("G", strtotime($result -> conversionDate));
413                             } else {
414                                     $index = date("Y-M", strtotime($result -> conversionDate));
415                             }
416
417                             if (isset($data[$index]["totalOrderValue"])) {
418                                     $data[$index]["totalOrderValue"] += $result -> orderValue;
419                                     $data[$index]["totalAffiliateCommission"] += $result ->
    affiliateCommission;
420                                     if ($media != 0)
421                                             $counter[$index]++;
422                             } else {
423                                     $data[$index]["totalOrderValue"] = $result -> orderValue;
424                                     $data[$index]["totalAffiliateCommission"] = $result ->
    affiliateCommission;
425                                     if ($media != 0)
426                                             $counter[$index] = 1;
427                             }
428                     }
429                     if ($media != 0)
430                             $data = $this -> media($data, $counter);
431                     return $data;
432
433             }
434
435             public function orderValueAndCommissionPerProgram($startDate, $endDate, $media) {
436                     $CI = &get_instance();
437                     $CI -> db -> select("conversions.affiliateCommission, conversions.orderValue,
    conversions.conversionDate, conversions.programId, campaigns.programName FROM conversions");
438                     $CI -> db -> where('conversions.conversionDate >=', date("Y-m-d", strtotime
    ($startDate)));
439                     $CI -> db -> where('conversions.conversionDate <=', date("Y-m-d", strtotime
    ($endDate)));
440                     $CI -> db -> join('categories', 'conversions.programId = categories.programId AND
    conversions.categoryId=categories.categoryId AND conversions.affiliateId =
    categories.affiliateId');
441                     $CI -> db -> join('campaigns', 'campaigns.programId = categories.programId AND
    campaigns.affiliateId = categories.affiliateId');
442                     $CI -> db -> join('sites', 'sites.affiliateId = campaigns.affiliateId');
443                     $query = $CI -> db -> get();
444                     $data = array();
445                     if ($media != 0)
446                             $counter = array();
447                     foreach ($query->result() as $result) {
448                             $program = $result -> programName;
449                             ;
450                             if (isset($data[$program])) {
451                                     $data[$program]["totalAffiliateCommission"] += $result ->
    affiliateCommission;
452                                     $data[$program]["totalOrderValue"] += $result -> orderValue;
```

```
453                                     if ($media != 0)
454                                             $counter[$program]++;
455                             } else {
456                                     $data[$program]["totalAffiliateCommission"] = $result ->
affiliateCommission;
457                                     $data[$program]["totalOrderValue"] = $result -> orderValue;
458                                     if ($media != 0)
459                                             $counter[$program] = 1;
460                             }
461                     }
462                     if ($media != 0)
463                             $data = $this -> media($data, $counter);
464                     return $this -> arraySort($data);
465
466             }
467
468             public function arraySort($array) {
469                     if (isset($array)) {
470                             array_multisort(array_values($array), SORT_DESC, array_keys($array),
$array);
471                             return $array;
472                     } else {
473                             return;
474                     }
475             }
476
477             public function statusPerProgram($startDate, $endDate) {
478                     $CI = &get_instance();
479                     $CI -> db -> select("conversions.pendingStatus, campaigns.programName from
conversions");
480                     $CI -> db -> where('conversions.conversionDate >=', date("Y-m-d", strtotime
($startDate)));
481                     $CI -> db -> where('conversions.conversionDate <=', date("Y-m-d", strtotime
($endDate)));
482                     $CI -> db -> join('categories', 'conversions.programId = categories.programId AND
conversions.categoryId=categories.categoryId AND conversions.affiliateId =
categories.affiliateId');
483                     $CI -> db -> join('campaigns', 'campaigns.programId = categories.programId AND
campaigns.affiliateId = categories.affiliateId');
484                     $query = $CI -> db -> get();
485                     $data = array();
486                     foreach ($query->result() as $result) {
487                             if (!isset($data[$result -> programName])) {
488                                     $data[$result -> programName]["A"] = 0;
489                                     $data[$result -> programName]["P"] = 0;
490                                     $data[$result -> programName]["D"] = 0;
491                                     $data[$result -> programName][$result -> pendingStatus]++;
492                             } else {
493                                     $data[$result -> programName][$result -> pendingStatus]++;
494                             }
495                     }
496                     return $this -> arraySort($data);
497             }
498
499             public function bestUsers($startDate, $endDate) {
500                     $CI = &get_instance();
501                     $CI -> db -> select("conversions.conversionDate , conversions.epi1,
conversions.affiliateCommission from conversions");
502                     $CI -> db -> where('conversions.conversionDate >=', date("Y-m-d", strtotime
($startDate)));
503                     $CI -> db -> where('conversions.conversionDate <=', date("Y-m-d", strtotime
($endDate)));
504                     $query = $CI -> db -> get();
505                     $data = array();
506                     foreach ($query->result() as $result) {
507                             if (isset($data[$result -> epi1])) {
508                                     $data[$result -> epi1]["totalAffiliateCommission"] += $result ->
affiliateCommission;
509                                     $data[$result -> epi1]["numberOfConversions"]++;
510                             } else {
511                                     $data[$result -> epi1]["totalAffiliateCommission"] = $result ->
affiliateCommission;
512                                     $data[$result -> epi1]["numberOfConversions"] = 1;
513                             }
514                     }
515                     return $this -> arraySort($data);
516             }
517
518             public function statusPerCampaigns($startDate) {
519                     $CI = &get_instance();
520                     $CI -> db -> select("campaigns.statusId from campaigns");
521                     $CI -> db -> where('campaigns.startDate >=', date("Y-m-d", strtotime
($startDate)));
522                     $query = $CI -> db -> get();
523                     $data[1] = 0;
524                     $data[2] = 0;
525                     $data[3] = 0;
```

```
526                     $data[4] = 0;
527                     $data[5] = 0;
528                     $data[6] = 0;
529                     foreach ($query->result() as $result) {
530                             $data[$result -> statusId]++;
531                     }
532                     return $data;
533             }
534
535             public function media($data, $counter) {
536                     $keys = array_keys($data);
537                     foreach ($keys as $key) {
538                             if ($counter[$key] != 0) {
539                                     $data[$key]["totalAffiliateCommission"] = $data[$key]
    ["totalAffiliateCommission"] / $counter[$key];
540                                     $data[$key]["totalOrderValue"] = $data[$key]["totalOrderValue"] /
    $counter[$key];
541                             }
542                     }
543                     return $data;
544             }
545
546             public function storico() {
547                     $CI = &get_instance();
548                     $CI -> db -> select("fi_campaigns.ptcampaignid ,
    fi_campaigns_categories.oursfiimport , fi_campaigns_categories.oursfloat ,
    fi_campaigns_categories.date , fi_campaigns_categories.eventId");
549                     $CI -> db -> where('fi_campaigns.pttypeid', 2);
550                     $CI -> db -> join('fi_campaigns', 'fi_campaigns.ptcampaignsid =
    fi_campaigns_categories.campaignsid');
551                     $query = $CI -> db -> get();
552                     foreach ($query->result() as $result) {
553                             $data["programId"] = $result -> ptcampaignid;
554                             $data["programTariffAmount"] = ($result -> oursfiimport) / 100;
555                             $data["programTariffPercentage"] = ($result -> oursfloat) * 100;
556                             $data["update"] = $result -> date;
557                             $data["categoryId"] = $result -> eventId;
558                     }
559
560             }
561
562     }
563     ?>
```

Bibliografia

a4uevents. (s.d.). *Performance Marketing Awards 15th May 2012*. Tratto il giorno 9 16, 2012 da performancemarketingawards web site: http://www.performancemarketingawards.co.uk

Appcelerator, Inc. (s.d.). *Aptana | Studio*. Tratto il giorno 9 12, 2012 da Aptana Web site: http://www.aptana.com/products/studio3

Chen, Manish, Rasmussen. (2009). *Business Dashboards: A Visual Catalog for Design and Deployment.* John Wiley & Sons.

Dundas Data Visualization Inc. (2012, 9 12). Dashboarding Articles The Dashboard Demystified.

EllisLab, Inc. (s.d.). *CodeIgniter - Open source PHP web application framework*. Tratto il giorno 9 16, 2012 da CodeIgniter Web site: http://codeigniter.com

EllisLab, Inc. (s.d.). *EllisLab - Where Ideas Hatch!* Tratto il giorno 9 18, 2012 da EllisLab, Inc Web site: http://ellislab.com

Elmasri, Navathe. (2007). *Sistemi di basi di dati: Fondamenti.* Pearson Addison Wesley.

GitHub Inc. (s.d.). *EllisLab/CodeIgniter · GitHub*. Tratto il giorno 9 12, 2012 da GitHub Web site: https://github.com/EllisLab/CodeIgniter

Google. (s.d.). *Quick Start - Google Chart Tools — Google Developers*. Tratto il giorno 9 12, 2012 da Google Developers Web site: https://developers.google.com/chart/interactive/docs/quick_start

HORIZON GROUP SRL. (s.d.). *Guadagnare online con il Social Cashback FidelityHouse*. Tratto il giorno 9 18, 2012 da Fidelity House Web site: http://www.fidelityhouse.eu

HORIZON GROUP SRL. (s.d.). *NEXTNEWS360*. Tratto da NextNews360 Web site: http://www.nextnews360.net

HORIZON GROUP SRL. (s.d.). *Piattaforma di Affiliazione - Programmi di Affiliazione LeadHouse*. Tratto il giorno 9 18, 2012 da LeadHouse Web site: https://www.leadhouse.net

HORIZON GROUP SRL. (s.d.). *Tutte le divisioni di Horizon Group: Horizon ADV, LeadHouse e FidelityHouse*. Tratto il giorno 9 18, 2012 da http://www.horizongroup.it/horizon-adv-leadhouse-fidelityhouse.html

LIBERO. (s.d.). *Libero*. Tratto il giorno 9 18, 2012 da Libero Web site: http://www.libero.it

Mr.Webmaster. (s.d.). *Installazione e configurazione del framework | Guida CodeIgniter | PHP | Mr.Webmaster*. Tratto il giorno 9 12, 2012 da mrwebmaster Web site: http://www.mrwebmaster.it/php/guide/installazione-configurazione-framework_985.html

Potencier, F. (s.d.). *symfony | Web PHP Framework*. Tratto il giorno 9 12, 2012 da Symfony Web site: http://www.symfony-project.org

Tiscali Italia S.p.a. (s.d.). *Tiscali*. Tratto il giorno 9 18, 2012 da Tiscali Web site: http://www.tiscali.it

Tradedoubler. (s.d.). *Affiliazioni online con Tradedoubler*. Tratto il giorno 9 12, 2012 da Tradedoubler Web site: http://www.tradedoubler.com/it-it/affiliazioni-online-con-tradedoubler/

Tradedoubler. (s.d.). *Connect and grow with Tradedoubler affiliate marketing*. Tratto il giorno 9 18, 2012 da Tradedoubler Web page: http://www.tradedoubler.com

Triboo Editoriale S.r.l. (s.d.). *Leonardo.it - il portale delle passioni*. Tratto il giorno 9 18, 2012 da Leonardo Web site: http://www.leonardo.it

Zanox. (s.d.). *zanox - il network europeo leader nel performance advertising*. Tratto il giorno 9 18, 2012 da Zanox Web site: http://www.zanox.com/it/?wt_mc=amc67717841&gclid=COSy6968wbICFUG-zAodGCwAdA

Zend Technologies Ltd. (s.d.). *PHP Web Application Server - PHP Development tools - PHP Training - Zend.com*. Tratto il giorno 9 12, 2012 da Zend Web site: http://www.zend.com/it/

Ringraziamenti

Colgo l'occasione per ringraziare Horizon ADV s.r.l., l'azienda presso cui ho svolto lo stage e in particolare Alessandra per avermi seguito e aiutato durante tutto lo svolgimento del tirocinio. Un doveroso ringraziamento al prof. Giorgio Satta, relatore della presente tesi, per la cortese disponibilità.
Un grazie particolare va sicuramente riservato alla mia famiglia, papà mamma, Alessandra, Luca e Maria e, ovviamente, il prossimo arrivo che, a giorni, ci renderà felici allargando la famiglia.
Grazie ai miei amici, che con consigli e, alle volte, necessarie spinte mi hanno aiutato a crescere e affrontare le difficoltà.
Grazie ai miei animati e all'animazione in generale, capaci di regalarmi tanto, tra cui, momenti davvero indimenticabili che porterò sempre con me.
Grazie a Benedetta, persona stupenda che ha incrociato la mia via, e con la quale sto vivendo una delle cose più belle della vita.