Università degli studi di Salerno

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Informatica

TESI DI LAUREA

Framework MVC per lo sviluppo di Web Application :

Zend Framework *vs* cakePHP

Relatore

Prof. Blundo Carlo

Candidato

Attanasio Ciro

056 / 101820

Anno Accademico 2010 / 2011

Sii il cambiamento che vuoi vedere
avvenire nel mondo
(Mohandas Karamchand Gandhi)

...a tutti coloro che hanno creduto in me
...alla mia paziente famiglia
...al mio dolcissimo amore
...alla mia team manager
...alla segretaria ePC
...al grande capo

Indice

Indice delle figure

Indice delle tabelle

Sommario (abstract)

Quanto più ci innalziamo, tanto più piccoli sembriamo
a quelli che non possono volare
(Friedrich Wilhelm Nietzsche)

Una “Web Application” è un software, eseguito dall’Application Server di un Web Server, che produce dinamicamente pagine HTML secondo le richieste degli utenti e degli altri sistemi della rete.

Le funzioni svolte da questo tipo di software possono andare dalla semplice ricerca e fruizione di un file in una directory a delle applicazioni estremamente sofisticate.

Anche le tecnologie che sono alle spalle di questi software sono estremamente varie: dal semplice script fino ad un complicato programma in linguaggio di alto livello che interagisce con diverse risorse dati localizzate su sistemi ed in modi diversi.

Nella produzione del software, il Framework è una struttura di supporto su cui un software può essere organizzato e progettato. Alla base di un Framework c'è sempre una serie di librerie di codice utilizzabili con uno o più linguaggi di programmazione, spesso corredate da una serie di strumenti di supporto allo sviluppo del software (es. un IDE, un debugger o altri strumenti) ideati per aumentare la velocità di sviluppo del prodotto finito.

Lo scopo di un Framework è di risparmiare allo sviluppatore la riscrittura di codice già steso in precedenza per compiti simili. Questa circostanza si è presentata sempre più spesso man mano che le interfacce utente sono diventate sempre più complesse.

In questa tesi si discutono le problematiche legate alle scelte di un Framework PHP rispetto ad un altro provando a concentrarsi sull’ottica del progettista e dello sviluppatore di software.

Nella breve introduzione si farà una panoramica sufficientemente dettagliata dell'obiettivo che si vuole perseguire e dei mezzi utilizzati.

Nel primo capitolo si presentano le caratteristiche del pattern MVC.

Nel secondo capitolo si presentano le caratteristiche del Framework Zend Framework.

Nel terzo capitolo si presentano le caratteristiche del Framework cakePHP.

Nel quarto capitolo si mettono a confronto i due Framework alla luce di quanto esposto teoricamente nel primo, secondo e terzo capitolo.

Nel quinto capitolo si espone l'Analisi e la Progettazione di una “Web Application”.

Nel sesto capitolo si mettono a confronto i due Framework alla luce di quanto progettato e sviluppato su una Web Application, che si adopera da esempio, per studiare gli accorgimenti specifici, alcuni semplici ed altri più complicati, senza influire sulla qualità e sul “time to market”.

Nell’ultimo capitolo vengono esposte alcune conclusioni a riguardo delle problematiche analizzate e delle conoscenze acquisite con questo lavoro.

Introduzione

Ho avuto un'idea grandiosa, ma non mi è piaciuta
(Samuel Goldwyn)

1. Obiettivo

Il lavoro di tesi svolto si pone come obiettivo il confronto tra due Framework MVC per lo sviluppo di Web Application: Zend Framework e cakePHP. Considerando il linguaggio comune su cui si basano, ossia PHP, tale confronto è relativo soprattutto a ciò che riguarda gli strumenti, le funzionalità e le potenzialità fornite dall'uno e dall'altro.

2. Approccio

Si parte da una descrizione del Design Pattern MVC e di tutti gli aspetti relativi al suo meccanismo di funzionamento.

Successivamente, viene dato un ampio spazio a Zend Framework descrivendone tutte le funzionalità messe a disposizione per realizzare una Web Application. Allo stesso modo è stata approfondita la struttura di cakePHP.

Raccolte le informazioni necessarie, si è passato alla realizzazione di un confronto teorico tra i due Framework evidenziando vantaggi e svantaggi dell'uno rispetto all'altro in relazione a ciascuna funzionalità offerta.

Per poter fornire una base solida al confronto teorico, è stato preso in esame un caso di studio che ha previsto la realizzazione, per l'azienda di Busto Arsizio (VA) IdeaBitmap S.r.L., di una Web Application, per la Johnson & Johnson Medical, mediante i due Framework.

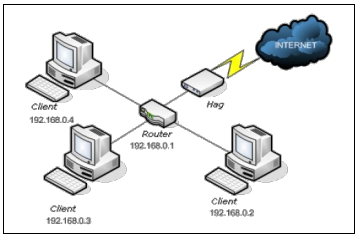
Figura 1 - Interface Life Scann Game

Fonte: http://www.ideabitmap.it

Tale software mette a disposizione degli utenti la possibilità di usufruire di un gioco via Rete Lan, ossia di un cameriere che trasporta una Chiavetta USB adagiata su di un vassoio lungo un corridoio. Registrandosi tramite il Blogger si ha la possibilità di usufruire del gioco stabilendo così dei record personali di percorrenza del corridoio per poi vederli video proiettati in sala in fase

successiva ed eventualmente poterli battere nei giorni a seguire ottenendo così un premio finale al termine della settimana. Per quanto riguarda il Blogger, egli ha a disposizione tutte le funzionalità di gestione dell'archivio dei record stabiliti durante le settimana, degli utenti (User) registrati e da registrare (Guest).

Figura 2 – Rete Lan Life Scann Game



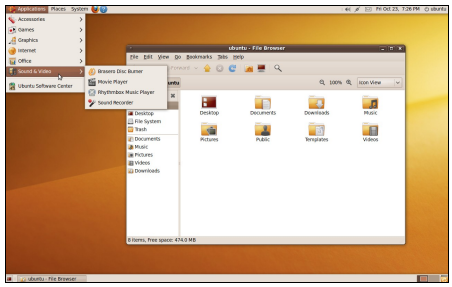
Fonte: http://blog.skydiamond.org

E' stata eseguita una preventiva fase di analisi e progettazione che ha previsto la realizzazione del documento di specifica (ed analisi) dei requisiti e di tutti i diagrammi UML (Use Case, Class Activity, Sequence e Statchart) oltre al modello concettuale e fisico della Base di Dati necessaria. A questa fase ha fatto seguito una doppia fase di implementazione che ha previsto la realizzazione della Web Application con entrambi i Framework. Infine, sulla base di quanto sviluppato sono state ripresi tutti gli aspetti che potessero rappresentare termini di confronto fra di essi e sono state evidenziate le analogie e le differenze di implementazione dell'uno e dell'altro.

3. Strumenti

Per quanto concerne gli strumenti adottati si è fatto ampio uso del software ArgoUML per lo sviluppo dei diagrammi UML, dell'ambiente IDE Eclipse 3.1 per l'implementazione ed infine di Apache 2.2.11 come ambiente di esecuzione delle due implementazioni contenuto nel pacchetto Lampp 2,0 sotto ambiente UNIX.

Figura 3 - Desktop Ubuntu 10.0



Fonte: http://www.gogeeks.tv

Il Sistema Operativo utilizzato è Ubuntu 10.0 basato sull'ambiente desktop GNOME progettato per fornire un'interfaccia semplice, intuitiva e allo stesso tempo completa e potente.

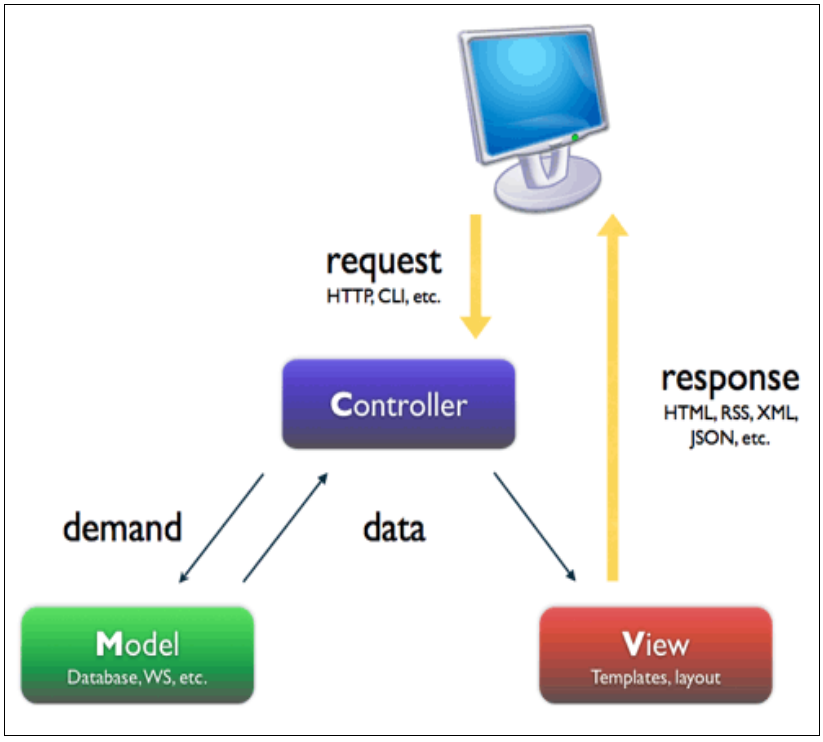
Capitolo I – Il Design Pattern MVC

Tutti sanno che una cosa è impossibile da realizzare, finché
arriva uno sprovveduto che non lo sa e la inventa
(Albert Einstein)

1. Introduzione

Model-View-Controller (MVC, talvolta tradotto in italiano Modello-Vista-Controllore) è un Design Pattern architetturale molto diffuso nello sviluppo di interfacce grafiche di sistemi software object-oriented. Originariamente impiegato dal linguaggio Smalltalk, il Design Pattern è stato esplicitamente o implicitamente sposato da numerose tecnologie moderne, come Framework basati su PHP (es. Symfony, Zend Framework e cakePHP), su Ruby (es. Ruby on Rails), su Python (es. Django), su Java (es. Swing, JSF e Struts), su Objective C o su .NET.

Figura 4 – L'Architettura Design Patterrn MVC



Fonte: http://www.symfony-project.org

A causa della crescente diffusione di tecnologie basate su MVC nel contesto di Framework o piattaforma middleware per Web Application, l'espressione Framework MVC o Sistema MVC sta

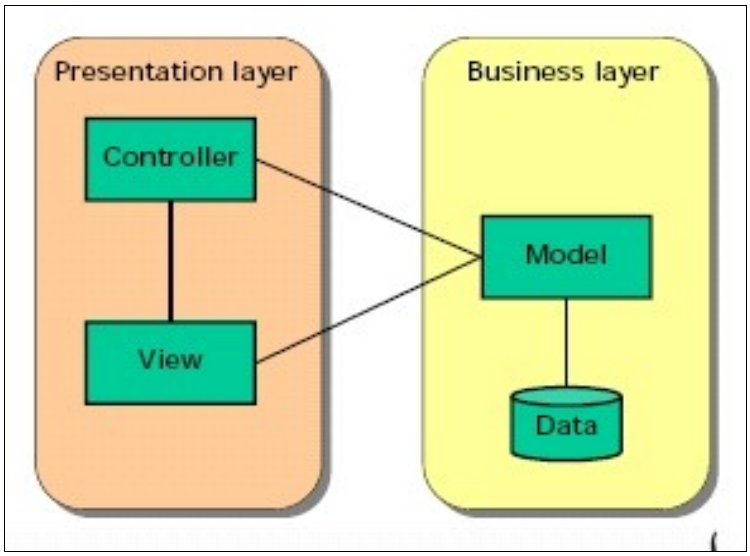
entrando nell'uso anche per indicare specificamente questa categoria di sistemi (es. Ruby on Rails, Struts, Spring, Tapestry e Catalyst).

2. Struttura

Il Design Pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- Model - Fornisce i metodi per accedere ai dati utili alla Web Application ed è detto anche domain logic/layer o business logic/layer;
- View - Visualizza i dati contenuti nel Model e si occupa dell'interazione con utenti e agenti. Tipicamente è una user interface. (es. la pagina HTML che mostra il dettaglio e il totale della spesa);
- Controller - Riceve i comandi dell'utente (in genere attraverso la View) e li attua modificando lo stato degli altri due componenti.

Figura 5 – Diagramma Design Patterrn MVC



Fonte: http://www.beansoftware.com

Questo schema di Figura 5, fra l'altro, implica anche la tradizionale separazione fra la logica applicativa (in questo contesto spesso chiamata "logica di business"), a carico del Controller e del Model, e l'interfaccia utente a carico della View.

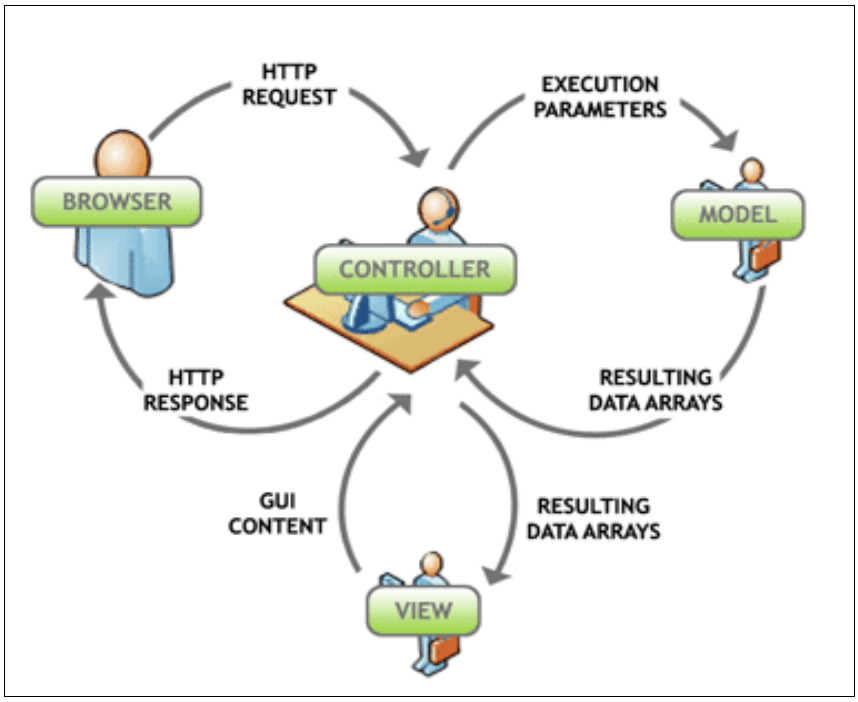
I dettagli delle interazioni fra questi tre oggetti software dipendono molto dalle tecnologie usate (es. linguaggio di programmazione, eventuali librerie, middleware e via dicendo) e dal tipo di Web Application (es. se si tratta di una Web Application o di una Web Application desktop). Quasi sempre la relazione fra View e Model è descrivibile anche come istanza del Design Pattern Observer. A volte, quando è necessario cambiare il comportamento standard della Web Application a seconda delle circostanze, il Controller implementa anche il Design Pattern Strategy.

3. Come opera

Vista per sommi capi la struttura del Design Pattern, lo schema illustrato in figura 6 mostra come opera:

- L'utente "Ranieri Antonio" interagisce con l'interfaccia (es. fa un semplice click su un link) e genera una richiesta http;
- Il Controller riceve la richiesta, inizializza il sistema, filtra i dati e individua un'azione;
- Il Model viene chiamato dal Controller e in base alla specifica richiesta ne altera lo stato;
- Viene scelta una View in base alla logica della Web Application;
- L'interfaccia utente aspetta ulteriore richieste dall'utente e inizia un nuovo ciclo.

Figura 6 – Come opera il Design Patterrn MVC



Fonte: http://www.programmazione-web.com

4. Vantaggi

I vantaggi del Design Pattern sono molteplici:

- Indipendenza delle varie componenti, che permette di lavorare separatamente in team (con competenze diverse) alle parti software astraendone al meglio il funzionamento;
- Possibilità di scrivere Wiew e Controller diversi utilizzando lo stesso Model di acceso ai dati e quindi riutilizzare parte del lavoro già fatto;

- Avere il Controller separato dal resto della Web Application che rende la sua progettazione più semplice permettendo di concentrare gli sforzi sulla logica del funzionamento;
- Obbligare gli sviluppatori a rispettare uno standard nella stesura del progetto, che ne facilita poi la comprensione e le successive implementazioni, soprattutto utilizzabile in progetti di medie/grandi dimensioni;
- Ottenere un software più flessibile, mantenibile ed aggiornabile nel tempo.

5. Svantaggi

Non potevano mancare gli svantaggi in questo Design Pattern:

- Adatto soprattutto a progetti medio/grandi;
- Architettura sostanzialmente complessa;
- Flessibilità dipendente dal Framework utilizzato.

Capitolo II – Il Framework Zend Framework

...nessun profumo vale l'odore di quel fuoco...
(Robert Baden-Powell)

1. Introduzione

Zend Framework è un Framework Web open source, scritto interamente in PHP 5 e rilasciato con licenza New BSD. Lo scopo principale è quello di offrire le funzionalità e gli strumenti necessari per rispondere a tutte le (principali) esigenze di sviluppo di una Web Application.

Offre allo sviluppatore una ricca dotazione di librerie per concentrarsi sullo sviluppo principale del progetto invece di perdere tempo, come spesso succede, nei dettagli o nel “reinventare la ruota”.

1.1 Panoramica

Ogni grande progetto (normalmente) nasce sponsorizzato, per esigenza o su commissione

La tradizione non si è interrotta neanche per la maggior parte dei Framework MVC. Basti pensare che Rails era inizialmente stato concepito da 37Signals e solo in seguito rilasciato open source, mentre Symfony è sponsorizzato da Sensiolabs.

Come facilmente intuibile dal nome, dietro c’è Zend, la società alle spalle del PHP Engine, ovvero il cuore dell’interprete PHP. Questo non solo lo colloca direttamente in una posizione privilegiata rispetto ad altri “concorrenti”, ma garantisce anche una certa qualità, affidabilità e continuità di sviluppo derivante dal chiaro interesse di Zend in questo prodotto.

Come anticipato, è completamente scritto in PHP 5. Più precisamente, l’ultima versione disponibile (ad oggi la 1.10) richiede PHP 5.1.4 o superiore.

Questo requisito non rappresenta un capriccio per incentivare l’uso di PHP 5 che, sebbene disponibile da oltre 5 anni, ancora soffre di sindrome della ridotta considerazione a causa del suo fratello minore PHP 4. In realtà il Framework sfrutta a fondo le novità di questa major release e non potrebbe coesistere con una versione precedente.

Come si analizzerà nello specifico nella sezione architettura, tutte le librerie sfruttano una programmazione object oriented. Buona parte di esse adotta funzioni o caratteristiche non retro compatibili come i metodi speciali **__call()**, **__get()** e **__set()**.

1.2 Architettura

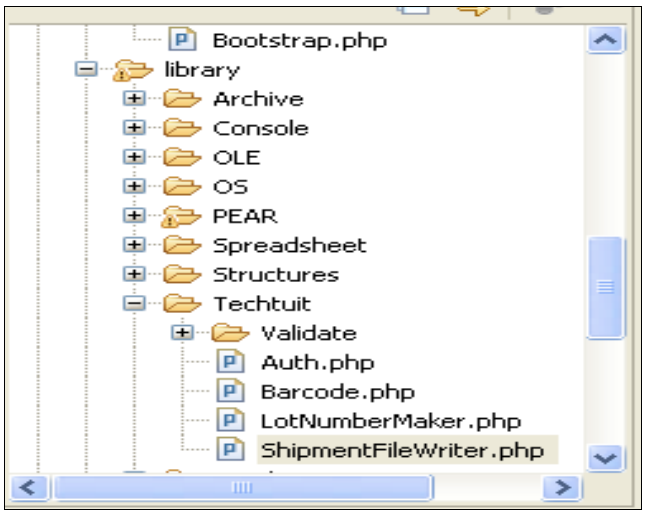
Il Framework è progettato per aderire alle caratteristiche del Design Pattern MVC. Un’altra caratteristica lampante è la sua forte modularità: rispetto ad altri, la struttura delle librerie che lo

compone non ha alcuna dipendenza da altre esterne. Inoltre, ciascuna può essere considerata come autonoma, dipendente esclusivamente ad un limitato set necessarie al suo funzionamento.

In altri termini, si può scegliere di utilizzare in un programma esistente la libreria Zend_Feed senza portare dietro l'intero pacchetto, ad eccezione di Zend_Http_Client e Zend_Uri, alle quali Zend_Feed ricorre per il fetching di feed esterni e la gestione delle URL.

Se poi in futuro si decidesse di aggiungere ulteriori elementi alla Web Application, si può sempre scegliere di integrare l'installazione con le librerie mancanti ed aggiornare il programma, come una delle più classiche architetture a Plug-In.

Figura 7 – La Directory Library Zend Framework



Fonte : http://blog.lysender.com

Questa struttura si apprezza particolarmente quando l'esigenza è di estendere un sistema esistente senza riscriverlo completamente, sia esso un semplice script o una più complessa Web Application basata a sua volta su un altro Framework.

Non è un mistero che sia spesso utilizzato in comunione con Symfony. Quest'ultimo integra, nella configurazione predefinita, un componente chiamato Zend Bridge ed un Plug-In sfZendPlugin che permettono la comunicazione tra Zend Framework e Symfony.

Altri Plug-In invece, come sfLucenePlugin, sfruttano invece specifici componenti (in questo caso Zend_Lucene) e sono la dimostrazione pratica di quanto prima affermato in merito alla sua modularità.

In sintesi, l'architettura è quella di un Framework MVC altamente modulare e completamente object oriented.

1.3 Roadmap

Ci sarà tempo per analizzare nello specifico le singole roadmap, spesso conosciute come milestone e corrispondenti a specifiche release.

Il termine roadmap, in questa circostanza, trova una migliore collocazione nel contesto di obiettivi, requisiti e traguardi alla base dello sviluppo del Framework. Ecco, di seguito, un rapido sguardo agli elementi principali.

1.3.1 Espansione del supporto ai Web Services

Uno degli aspetti che colpisce di più, quando ci si avvicina, è proprio il supporto integrato ad alcuni Web Services e la disponibilità di una serie di strumenti che rende l'integrazione con le API quasi un gioco da ragazzi.

Tra gli obiettivi rientra la continua espansione delle librerie parte di Zend_Service con il supporto a nuovi servizi e nuove API, sempre più spesso a disposizione degli sviluppatori.

1.3.2 Supporto sistemi digitali per gestione delle identità

Anche in questo campo offre, probabilmente, uno dei supporti più evoluti anche a confronto con altri linguaggi.

Da poco ha fatto la sua timida comparsa Zend_OpenId e presto il supporto includerà anche altri sistemi di autenticazione.

1.3.3 Supporto alla gestione e generazione di form

Una delle caratteristiche essenziali in una Web Application sono i form che permettono una più completa interazione tra la Web Application e l'utente. Al momento è molto carente in questo ambito. Sono già disponibili diverse proposte in fase di approvazione, è probabile che molto presto qualcosa di pratico venga messo in cantiere.

1.3.4 Supporto a YAML

Nell'epoca dei Framework MVC sembra quasi che non possa esistere un Framework che non supporti YAML e che il buon vecchio XML sia oramai troppo "verbose" per una Web Application 2.0. E' obiettivo il supporto al linguaggio con un condivisibile intento che traspare dalle pagine ufficiali: "where it makes sense".

1.3.5 Supporto a AJAX

Il supporto lato server per Ajax aiuta ad automatizzare le procedure di rilevazione e a migliorare i tempi di risposta e rende inoltre molto più facile la programmazione in JavaScript e PHP

1.4 Strumenti di sviluppo

Tra le diverse librerie non tutte hanno come obiettivo l'interazione diretta lato Web con una Web Application. Alcune costituiscono codice utile a scrivere altro codice, un pò la sottile ironia di un editor scritto in Java per scrivere codice Java.

Nell'intento di migliorare l'esperienza di sviluppo, sono in cantiere librerie utili ad automatizzare i processi, gestire la generazione automatica di attività ripetitive e minimizzare l'esigenza di "scimmie ammaestrate" per la scrittura del codice (es. Rake, Pake e Phing).

1.5 Continua espansione delle funzionalità

Come dimostrato, l'ambito non si riduce esclusivamente ad un semplice Framework MVC. Rientra in questa visione il continuo sviluppo di nuove librerie utili ad estendere le funzionalità.

1.6 Caratteristiche e finalità

Cos'è dunque Zend Framework? Si tratta di un insieme di classi che hanno lo scopo di semplificare ed accelerare la realizzazione delle applicazioni, garantendo sicurezza, scalabilità e robustezza. Vuole fornire soluzioni ai problemi ricorrenti in modo che gli sviluppatori possano concentrarsi sulle reali peculiarità del loro progetto.

Pone a se stesso degli obiettivi ambiziosi che, se raggiunti appieno, lo renderebbero unico nel panorama dei Framework per PHP:

- I suoi diversi componenti devono essere semplici da usare, in modo da rispettare la filosofia che ha decretato il successo del PHP;
- L'intero Framework non deve esser specializzato per qualche particolare compito ma deve poter essere impiegato in tutti i contesti, persino nella realizzazione di Web Applicationa da riga di comando;
- Non deve imporre scelte stringenti o pesanti vincoli allo sviluppatore ma deve lasciar ampio spazio alla personalizzazione di tutto il suo funzionamento;
- Deve essere completamente modulare, i suoi componenti devono poter essere utilizzati separatamente l'uno dall'altro ed in congiunzione con librerie esterne;
- La documentazione deve essere completa e costantemente aggiornata;
- Il codice che lo costituisce deve essere scritto in PHP 5 e deve essere compatibile con il più stringente livello di segnalazione errori E_STRICT;
- Deve essere interamente realizzato da zero e non deve avere alcuna dipendenza con librerie esterne;
- Anche se è il frutto della collaborazione dei più esperti sviluppatori PHP, deve rimanere aperto alle proposte della comunità di sviluppatori PHP.

I suddetti obiettivi non sono certo di facile realizzazione ma, all'alba dell'uscita della versione 1.11, è difficile trovarne uno che non sia stato, almeno in parte, raggiunto.

1.7 I componenti

Come accennato, si compone di una collezione di classi, quindi è realizzato secondo lo stile di programmazione ad oggetti. Questo è uno dei motivi per cui PHP 4 non viene supportato, a causa del suo ridotto supporto al paradigma OOP. Le classi sono organizzate per gruppi funzionali e

costituiscono i componenti del Framework. Lo sviluppatore però non è costretto ad usare necessariamente tutti questi componenti poiché sono indipendenti l'uno dall'altro anche se ottimizzati per interagire tra loro.

I principali componenti sono:

- Model-View-Controller (MVC) - Il motore che realizza l'architettura della Web Application e la generazione delle pagine Web. Si basa sullo schema del Deign Pattern MVC e sull'utilizzo di un Front Controller per lo smistamento delle richieste utente;
- Core Infrastructure - Componenti per la gestione della configurazione della Web Application, per la gestione dei log, della cache, per la validazione dell'input dell'utente, il caricamento automatico delle classi, ecc.;
- Databases - Classi che gestiscono l'accesso e l'utilizzo dei Database ed implementano la mappatura degli oggetti in tabelle (ORM);
- Authentication e Authorization - Gestiscono il riconoscimento dell'utente, la gestione dei diritti di accesso e le sessioni;
- Internationalization (i18n) e Localization (l10n) - Componenti che si occupano di semplificare la realizzazione di una Web Application multilingua;
- E-Mail, Formats, e Search - Permettono l'invio di E-Mail complesse, la produzione di PDF, la realizzazione di motori di ricerca e l'utilizzo di file JSON;
- Web e Web Services - consentono di realizzare client e server per i Web Services. permettono inoltre di gestire feed RSS, effettuare chiamate XMLRPC e Rest;

Inutile affrontare in dettaglio ciascuno dei suddetti componenti. Le funzionalità implementate sono numerosissime, altamente configurabili ed estendibili.

2. Funzionalità

Può essere facilmente riconducibile ad una raccolta di librerie autonome, tuttavia è possibile organizzare gli attuali componenti in categorie classificando in modo più chiaro le caratteristiche e le funzionalità offerte dal Framework.

2.1 Model View Controller (MVC)

Il tema Design Pattern MVC è senza dubbio uno dei più in voga nell'ultimo anno, anche grazie al proliferare di Framework e soluzioni di agile development implementate secondo i canoni di questo Design Pattern. Non stupisce dunque che i componenti MVC siano quelli che nell'ultimo semestre, in proporzione, abbiano assistito ad un'evoluzione maggiore.

Rientrano in questo ambito:

- Zend_Controller,Zend_Controller_Action,Zend_Controller_Dispatcher,Zend_Controller_Plugin e Zend_Controller_RewriteRouter;
- Zend_View;

- Zend_Http_Request e Zend_Http_Response.

Inganna il fatto che Zend_View conti una sola citazione rispetto ai componenti dedicati al Controller. esso infatti offre numerosissime funzionalità tipiche di un template engine, compreso il supporto a template engine esistenti.

2.2 Database

L'interazione con un Database è una delle esigenze più comuni per una Web Application. Non offre un vero e proprio ORM, piuttosto un'astrazione di livello leggermente più elevato rispetto alle API messe a disposizione dal linguaggio PHP.

Non per nulla Zend_Db e Zend_Db_Table, i due componenti che rientrano in questo ambito, sono definiti come a "lightweight solution for object-oriented programming with Databases".

Rientrano in questo ambito:

- Zend_Db;
- Zend_Db_Table.

2.3 Internazionalizzazione (i18n) e Localizzazione (L10n)

Non è difficile immaginare quanto comune sia la necessità di localizzare o internazionalizzare una Web Application soprattutto quanto il suo successo e la sua diffusione si spingono oltre i confini iniziali.

Per venire incontro a queste esigenze, supporta sia l'internazionalizzazione sia la localizzazione. Questo significa sia supporto alle classiche traduzioni, sia supporto a formati data, ora e unità di misura localizzate.

Rientrano in questo ambito:

- Zend_Date;
- Zend_Locale;
- Zend_Measure;
- Zend_Translate.

2.4 Autenticazione e gestione della Sessione

Una Web Application mediamente complessa necessita di sistemi di autenticazione adeguati per garantire l'accesso alle risorse solo agli account con i corretti privilegi. Anche in questo ambito offre diverse librerie utili per il supporto ai comuni protocolli di autenticazione, gestione delle sessioni di navigazione ed accesso a risorse basato su livelli e ruoli.

Rientrano in questo ambito:

- Zend_Acl;
- Zend_Authentication;

- Zend_Session.

2.5 E-Mail e formati di dati

Le moderne Web Application sono ricche di elementi multimediali, interfacce all'avanguardia per garantire una maggiore interazione tra l'utente ed il software. JavaScript ad oggi parte integrante di ogni sito Web dinamico, così come le E-Mail ed altri formati per lo scambio di dati.

Zend Framework offre librerie per lo scambio e la lettura di dati, la conversione di oggetti JSON in PHP e viceversa, generazione di PDF, oltre a lettura ed invio di E-Mail.

Rientrano in questo ambito:

- Zend_Json;
- Zend_Pdf;
- Zend_Mail;
- Zend_Mime.

2.6 Indicizzazione e Ricerca

Non è mistero che la Tecnologia Lucene rappresenti una delle soluzioni più avanzate per la ricerca testuale, se non la migliore. Zend Framework offre una libreria chiamata Zend_Search_Lucene che porta in PHP tutte le principali funzionalità di questo motore d'indicizzazione e ricerca scritto in Java.

Rientrano in questo ambito:

- Zend_Search_Lucene.

2.7 Web Service e Feed

Hanno sempre avuto un certo rilievo in Zend Framework così come i syndication feed. Il Framework integra un supporto per diversi popolari Web Services oltre che una libreria per leggere e scrivere feed Atom e RSS.

Inoltre, offre una serie di librerie che facilitano l'interazione con Web Services, interfacce RESTful, SOAP e XML-RPC.

Rientrano in questo ambito:

- Zend_Feed e Zend_Gdata;
- Zend_Rest_Client, Zend_Http_Client e Zend_XmlRpc_Client;
- Zend_Http_Server, Zend_Rest_Server e Zend_Server_Documentor;
- Zend_Uri;
- Zend_Server_Reflection, Zend_Soap_Server e Zend_XmlRpc_Server;
- Zend_Service.

2.8 Core

Nell'ambito Core rientrano tutte quelle librerie che fanno parte delle basi di un'architettura Web e, molto spesso, lavorano dietro alle quinte:

- Zend_Cache, Zend_Config, Zend_Console_Getopt, Zend_Log e Zend_Memory;
- Zend_Debug, Zend_Environment, Zend_Loader, Zend_Registry e Zend_Version;
- Zend_Filter e Zend_Validate.

3. Team di sviluppo

Come anticipato nell'introduzione, questo gioiellino è sponsorizzato principalmente da Zend che ne finanzia i costi tecnici per le infrastrutture, amministratori e l'organizzazione in generale. Tuttavia, questo non significa che sia esclusivamente sviluppato da Zend. Anzi, hanno collaborato al progetto, e collaborano tutt'ora, un elevato numero di programmatori individuali.

Figura 8 – Logo Zend Framework

Fonte: http://www.flickr.com/photos/ijansch/2675615660/

Ohloh classifica Zend Framework come uno tra i progetti open source con più elevato team di sviluppo nell'ultimo anno, basandosi sull'analisi dei commit eseguiti al repository SVN.
Sempre Ohloh afferma che nell'ultimo anno hanno partecipato 83 programmatori.

Come si può vedere, Zend Framework cresce anche (e soprattutto) grazie alla forte community che si è creata alla base e che ne sostiene (moralmente, tecnicamente e manualmente) lo sviluppo.

Non tutti i programmatori elencati come appartenenti a Zend Technologies sono ufficialmente (ed automaticamente) parte del ristretto team di coordinamento dello Zend Framework. Attualmente rientrano (ufficialmente) in questa cerchia le persone indicate come amministratori di Jira, l'issue tracker del progetto:

- Bill Karwin - capo progetto di Zend Framework che recentemente ha lasciato il posto;
- Wil Sinclair - project leader attuale;
- Darby Felton;
- Matthew Weier O'Phinney ;

- Laura Rooke.
- Melissa Keesling

Uno degli aspetti fondamentali di un progetto open source è la possibilità concessa a chiunque di contribuire, nei modi, tempi e competenze più adeguate.

3.1 Contribuire al progetto

La community ed il supporto da parte di utenti e sviluppatori è fondamentale per un progetto open source. Anche Zend Framework non fa eccezione e la community è una delle fonti di ispirazione, suggerimenti e debug più importante, se non forse la più importante.

Figura 9 – Homepage sito web Zend Framework

Fonte: http://www.zend.com

E' facile notare come molti sviluppatori non appartengano a Zend Technologies ma siano invece singoli o società che hanno deciso di investire parte del loro tempo a supporto del framework.

Chiunque può contribuire al progetto in diversi modi:

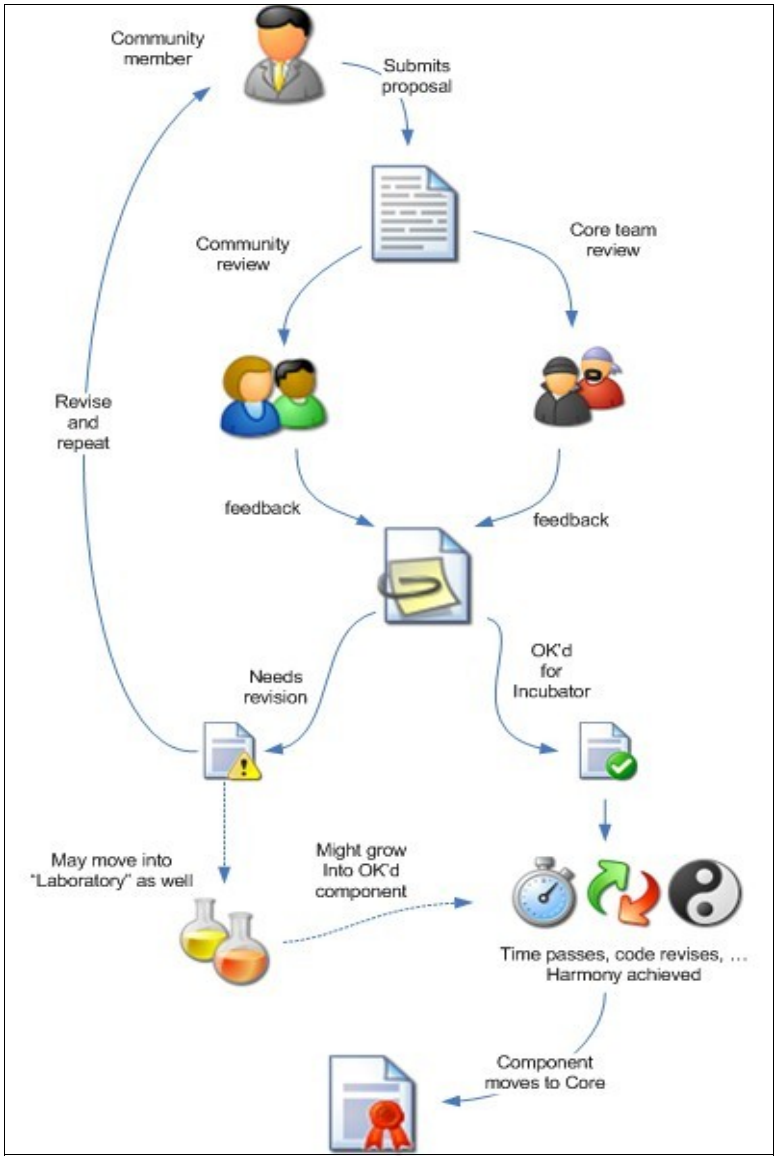
- Idee, suggerimenti e discussioni;
- Segnalazione di Errori e Bug;
- Invio di Patch e correzioni;
- Proposta di un nuovo componente;
- Parlare, scrivere e condividere le conoscenze;
- Contributor License Agreement (CLA).

3.2 Ciclo della proposta di una nuova componente

Il Proposal Lifecycle è l'iter alla quale viene sottoposto ogni nuovo componente. Il seguente

diagramma di Figura 10 illustra graficamente l'iter di una nuova proposta.

Figura 10 – Diagramma Ciclo della Proposta Zend Framework



Fonte: http://www.zend-framework.it

3.2.1 Scrittura della proposta

Il primo passo per un nuovo componente è... ...proporlo! L'autore, sia esso un componente del team di sviluppo o un utente, scrive la propria proposta seguendo il template Zend_Magic a disposizione partendo dalla pagina Zend Framework Proposals.

Una volta che la progettazione della proposta è completa, l'autore indica il componente come pronto per essere discusso.

3.2.2 Analisi della community

La creazione di un nuovo componente è soggetta a critiche degli utenti. Le critiche, i commenti ed i feedback servono all'autore per migliorare il design della proposta ed affinarla prima che venga analizzata dal Core team.

3.2.3 Analisi del Core team

Una volta che la proposta risulta matura, opportunamente disegnata e pronta per "la sentenza ufficiale", un componente del Core team di Zend analizza la proposta ed esprime il giudizio finale che può portare sostanzialmente a tre strade:

- La proposta è accettata;
- La proposta è accettata a determinate condizioni;
- La proposta è rifiutata.

Nel terzo caso, la proposta viene spostata nell'elenco delle proposte rifiutate con la motivazione che ha decretato la scelta. Negli altri due casi la proposta può essere affidata al laboratorio, all'incubatrice o finire direttamente in Core.

3.2.3.1 Laboratorio

Una proposta incompleta, il cui design non è ancora chiaro ma che ha potenzialità per essere inclusa in Zend Framework, viene assegnata al laboratorio.

Nel laboratorio il componente ha diritto ad uno spazio nel repository SVN per continuare lo sviluppo. La proposta verrà rivalutata non appena raggiunto un livello di maturità adeguato.

3.2.3.2 Incubatrice

L'incubatrice è il livello precedente all'integrazione della libreria nella distribuzione ufficiale del Framework. Nell'incubatrice una proposta, ma a questo punto è meglio chiamarlo il componente, ha diritto ad una collocazione nel repository SVN ufficiale ed un posto nell'issue tracker ufficiale.

Per abbandonare lo stadio di Incubator e passare a Core, un componente deve avere raggiunto i seguenti requisiti:

- Maturità del codice;
- Una copertura degli Unit Test di almeno dell'80% del codice;
- Adeguata documentazione.

Quando il componente ha raggiunto questi requisiti, un membro del Core team procede alla revisione finale e, in caso positivo, allo spostamento della libreria e materiale correlato in Core.

3.2.3.3 Core

Un componente parte della Core library di Zend Framework ha raggiunto il suo più alto grado

di maturità ed è disponibile in qualsiasi download ufficiale del Framework.

Sebbene si tratti di un traguardo, è bene considerare che anche i componenti presenti nel Core del Framework sono soggetti ad aggiornamenti, revisioni e cambiamenti. Un componente nel Core, inoltre, è soggetto a maggiore attenzione ma anche maggiore regolamentazione. Ad esempio, le modifiche devono sempre prestare attenzione a non rompere la compatibilità con versioni precedenti del codice, salvo rare eccezioni, e la soglia minima di accettabilità di un componente Core non deve mai essere superata anche in seguito a modifiche.

3.3 In conclusione

Il Core team di Zend Framework dedica molta cura ed attenzione all'evoluzione del Framework e delle sue librerie affinché il livello di qualità del risultato finale sia sempre all'altezza delle aspettative. Il tempo finale tra la scrittura di un componente e la sua promozione in Core può durare anche un anno, dunque non siate frettolosi!

4. Struttura di una Web Application – Hello, World!

4.1 Introduzione

Inizialmente i siti internet scritti in PHP erano composti da pagine dove il codice HTML risiedeva nello stesso file del codice PHP, e questo modo di fare sembrava tutto quello che si potesse desiderare. Con l'andare del tempo e il crescere dei progetti in dimensioni e complessità iniziarono a venir fuori diversi problemi nell'utilizzo di questa logica, questi problemi erano principalmente due: mantenibilità ed estensibilità.

Iniziarono quindi a nascere dei progetti con lo scopo di porre rimedio a questi problemi, uno di questi è proprio Zend Framework. Offre, in se, un gran numero di componenti che danno la possibilità allo sviluppatore di creare Web Application mantenibili ed estensibili a lungo termine ed implementa diversi Design Pattern tra cui l'ormai famoso Model-View-Controller per la suddivisione della logica del recupero dei dati dal Database e il successivo render su schermo.

4.2 Panoramica

Si vedrà, ora, come creare una semplice Web Application di prova sfruttando l'architettura del Design Pattern MVC di Zend Framework. Si procederà quindi alla creazione del programma di prova con lo scopo di visualizzare la stringa 'Hello, World' su schermo. Quello di cui si avrà bisogno sono i sorgenti di Zend Framework, quindi si dovrà costruire l'albero della Web Application, che sarà composto principalmente da cinque directory:

- application;
- library;
- tests;
- logs;

- public (o html).

Altre directory potranno essere create se necessario (es. “bin” per contenere file di configurazione.

Per enfatizzare la presenza del Design Pattern MVC all’interno della directory “application” è presente la directory 'default' in cui sono collocate le seguenti tre directory:

- models;
- views;
- controllers.

La directory “library” conterrà la directory “Zend” in cui saranno presenti le librerie di Zend Framework. Eventualmente, se necessarie, saranno incluse in questa directory altre librerie come il template engine Smarty o una libreria ORM come Propel.

La directory “tests” conterrà i file di UnitTest in modo da assicurare il corretto funzionamento delle varie componenti del sistema man mano che la Web Application crescerà.

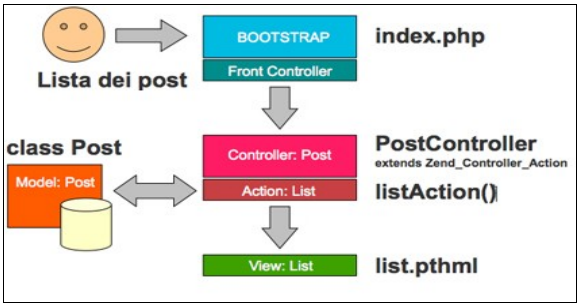
La directory “logs” conterrà semplici file di testo nei quali Apache scriverà, in generale, gli accessi e gli errori riscontrati. Per un controllo completo sul server, e magari per debuggare script o simili, è utile saper leggere questi file.

La directory “pulic” (o “html”) è l’unica directory a cui il Web Server avrà accesso. Infatti il sistema utilizzerà un file .htaccess che reindirizzerà tutte le richieste ad un file di bootstrap, che si chiamerà per convenzione index.php. Questo file risiederà all’interno della directory “pulic” (o “html”) situata nella Document Root “webzend” creata in precedenza nella directory /opt/lampp/.

4.3 Le richieste dell’utente

Per capire come si comporterà la nostra Web Application, secondo il Design Pattern MVC, alle richieste dell’utente si osservi l’immagine riportata qui sotto in Figura 11.

Figura 11 - Diagramma Richiesta dell'Utente in Zend Framework



Fonte: http://www.zend.com

1. L’utente “Angelo De Caro” richiede la visualizzazione dei post;
2. Tutte le richieste vengono inviate a un file di bootstrap (index.php) che istanzia un

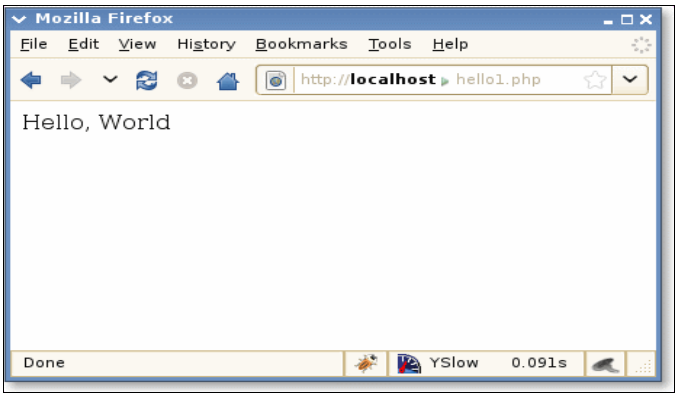
oggetto del Front Controller per elaborare le richieste;

3. Il Front Controller istanzia il Controller che si occupa della gestione dei post e chiama l'azione di visualizzazione;
4. Viene istanziato un oggetto del Model che si occupa di estrarre i post dal db. I risultati vengono restituiti dal Model al Controller;
5. I dati elaborati dal Model vengono inviati alla View per la creazione dell'HTML finale;
6. L'HTML viene creato e visualizzato all'utente "Angelo De Caro".

La richiesta per essere interpretata dal Front Controller di Zend Framework deve essere composta in questo modo:

- http://www.nomeserver.ex/

Figura 12 – La Parola "Hello, World" in Zend Framework



Fonte: http://www.zend.com

4.4 Creiamo la struttura

Per capire come si comporterà la nostra Web Application, secondo il Design Pattern MVC, alle richieste dell'utente analizziamo la figura 12 con un esempio di invocazione:

1. Creare un Virtual Host con Apache con le seguenti specifiche del server:
   - Indirizzo IP: 172.16.16.19;
   - Nome del virtualhost: http://www.webzend.ex;
   - Directory da pubblicare: /opt/lampp/webzend/public;
   - Directory dei log: /opt/lampp/webzend/logs.

   Per creare un Virtual Host con queste specifiche bisogna seguire tre operazioni:

   1. Aggiunge questa riga al file /etc/hosts:
      - 172.16.16.19 www.webzend.ex;

2. Creare il file di configurazione del Virtual Host:
    - /etc/apache2/sites-enabled/001-mioserver;

   Il file conterrà:

    1. NameVirtualHost *80
    2. <VirtualHost *80>
    3. ServerName www.mioserver.local
    4. DocumentRoot /var/www/mioserver/httpdocs
    5. CustomLog /var/www/mioserver/log/access.log combined
    6. ErrorLog /var/www/mioserver/log/error.log
    7. </VirtualHost>
3. Creare la struttura delle directory da pubblicare:
    - mkdir opt/lampp/webzend

2. Creare una cartella “application” all'interno della cartella “webzend”. Conterrà, in essa, tutti gli elementi che utilizzeremo nella Web Application: Controller, Model e View. Per motivi di sicurezza possiamo lasciare questa cartella al di fuori della Document Root del server Web in modo da non essere accessibile direttamente via Web.
3. Creare la cartella “default”, all’interno di “application”, a cui interno inserire le tre cartelle per i tre elementi citati sopra:
    - “models” conterrà le classi dei Model;
    - “views” conterrà le View separate per ogni Controller all’interno della cartella “scripts” che va creata qui dentro a “views”;
    - “controllers” conterrà tutti i Controller.
4. Creare una cartella che si può chiamare “public” (o “html” come fa Zend Studio) che conterrà il file di bootstrap e tutti gli elementi statici (es. immagini, css e javascript). La Document Root del nostro server Web dovrà puntare qui.
5. Creare un file di prova:
    - echo <h1>Hello World!</h1> > opt/lampp/Webzend/application/views/scripts/index/index.phtml
    - chmod -R www-data:www-data /var/www/mioserver/httpdocs

4.4.1 Il file .htaccess

Quando si parlava del Design Pattern MVC si è detto di dover indirizzare tutte le richieste ad un file index.php nella Document Root del server Web che era chiamata bootstrap.

Per fare in modo che tutte le richieste inviate al server vengano reindirizzate verso index.php si utilizza un modulo di apache mod_rewrite che permette di riscrivere l’indirizzo della richiesta

http in modo che utilizzando un'espressione regolare si possa reindirizzare tutte le richieste verso index.php. L'unica eccezione verrà fatta per la risorse statiche che non dovranno essere processate dal bootstrap.

Il file .htaccess permette di controllare questo comportamento:

1. Creare un nuovo file chiamato .htaccess (nb. il file inizia con un punto);
2. All'interno del file inserire le seguenti due righe:
    - RewriteEngine on
    - RewriteRule !\.(js|ico|gif|jpg|png|css)$ index.php

Perché tutto funzioni correttamente ci si deve assicurarci che il modulo mod_rewrite sia caricato e che le direttive di AllowOverride ci permettano di utilizzare i file .htaccess.

Per verificare queste impostazioni aprire il file httpd.conf all'interno della cartella conf di apache.

A questo punto se tutto ha funzionato correttamente scrivendo qualsiasi indirizzo (partendo dalla nostra Document Root) ci si dovrebbe essere reindirizzati verso index.php.

4.4.2 Il Bootstrap

Il bootstrap è il file che riceve tutte le richieste dell'utente, le elabora ed invia l'utente verso la risorsa desiderata. Questo lavoro di elaborazione e di routing viene effettuato dal Front Controller (Zend_Controller_Front), un modulo di Zend Framework.

La prima cosa che dobbiamo fare scrivendo il bootstrap è mettere nell'include path del PHP il percorso della cartella "library" di Zend Framework. La funzione di PHP da utilizzare è ovviamente set_include_path.

1. set_include_path('/path/to/library'
2. .PATH_SEPARATOR.
3. get_include_path());

Questa istruzione va a settare nell'include path il percorso verso la cartella "library" di Zend Framework (bisogna inserire il percorso reale al posto di /path/to/library) mantenendo quanto già nell'include path del PHP (che leggiamo con get_include_path). Dal momento che il separatore di path cambia da un sistema operativo all'altro lasciamo il compito di scegliere il separatore al PHP utilizzando la costante PATH_SEPARATOR.

Dopo aver incluso i file necessari creiamo un'instanza del Front Controller utilizzando il suo metodo getInstance()

1. require_once('Zend/Controller/Front.php');
2. $Controller = Zend_Controller_Front::getInstance()

Resta ancora qualche configurazione da fare al Front Controller prima che sia pronto. Per prima cosa dobbiamo settare la directory dove il Front Controller andrà a cercare i Controller:

- $Controller->setControllerDirectory('../application/Controllers');

L'ultima cosa che ci resta da fare è chiamare il metodo dispatch del Front Controller che darà il via alle operazioni di routing:

- $Controller->dispatch();

4.4.3 Il nostro primo Controller

Con Zend Framework tutti i Controller sono classi derivate da Zend_Controller_Action. Queste classi devo essere chiamate [Nome]Controller ed essere contenuti nel file [Nome]Controller.php all'interno della cartella dei Controller specificata in precedenza nel bootstrap.

[Nome] è il nome che vogliamo dare al nostro Controller con l'iniziale maiuscola. All'interno di ciuscun Controller scriveremo una o più azioni. Le azioni sono metodi del Controller che le contiene.

Questi metodi devo essere chiamati [nome]Action() dove [nome] è ovviamente il nome che vogliamo dare alla nostra azione questa volta con l'iniziale minuscola.

Fatta qualche precisazione iniziale possiamo procede con la creazione dell'IndexController. Creiamo un nuovo file PHP dove scriviamo:

1. class IndexController extends Zend_Controller_Action {
2. function indexAction() {} }

Una volta salvato il file all'interno della cartella ../application/controllers il nostro IndexController è pronto.

4.4.4 Ma indexAction() non fa niente?

Anche se indexAction è un metodo vuoto e che quindi non dovrebbe produrre alcun risultato, ci accorgeremo che Zend Framework automaticamente tenterà di visualizzare la View associata a questa azione cioè index.phtml nella cartella ../application/views/scripts/index. Questo comportamento è dato da un ActionHelper chiamato ViewRender. Una volta chiarito questo possiamo passare alla scrittura della View.

4.4.5 La nostra prima View

Per default Zend Framework associa alle viste l'estensione phtml (anche se questo comportamento può essere modificato).

Tutte le viste dovranno avere .phtml come estensione ed il nome dell'azione a cui sono associati come nome. Tutte le viste devono essere posizionate all'interno della cartella con il nome del Controller a cui si riferiscono posizionata in ../application/views/scripts/.

Nel nostro caso considerando che stiamo creando la vista per l'azione indexAction del Controller IndexController dobbiamo creare il file index.phtml e salvarlo nella cartella ../application/views/scripts/index

All'interno di index.phtml possiamo scrivere semplicemente

1.

2. <h1>Hello, World!</h1>
3.

4.5 Il risultato

Se non abbiamo fatto errori durante gli steep precedenti digitando nel browser l'indirizzo:

- http://www.webzend.ex

si dovrebbe vedere la pagina di "Hello, World!" nel browser.

Figura 13 – La Parola "Hello, World!" su http://www.webzend.ex

Fonte: http://www.webzend.ex

Capitolo III – Il Framework cakePHP

La bellezza delle cose esiste nella mente di chi le osserva
(David Hume)

1. Introduzione

cakePHP è un Framework per lo sviluppo veloce di Web Application PHP, gratuito e open-source. E' una struttura su cui i programmatori possono creare Web Application. L'obiettivo principale è permettere di lavorare in modo rapido e strutturato senza perdere in flessibilità.

1.1 Panoramica

Grazie alla sua struttura riesce ad elimina la monotonia dallo sviluppo Web mettendo a disposizione tutti gli strumenti di cui si ha bisogno per iniziare a realizzare ciò che è realmente necessario: “la logica della Web Application”. Invece di “reinventare la ruota”, ogni volta che si inizia un nuovo progetto, si scarica una copia di cakePHP e si parte direttamente dal cuore della Web Application. Ha un team di sviluppo molto attivo ed una community che fornisce un grande valore al progetto.

Oltre ad eliminare la necessità di "reinventare la ruota", garantisce che il Core della Web Application sia profondamente testato e continuamente aggiornato.

Di seguito è riportata una rapida lista delle caratteristiche che si avrà il piacere di scoprire nell'uso:

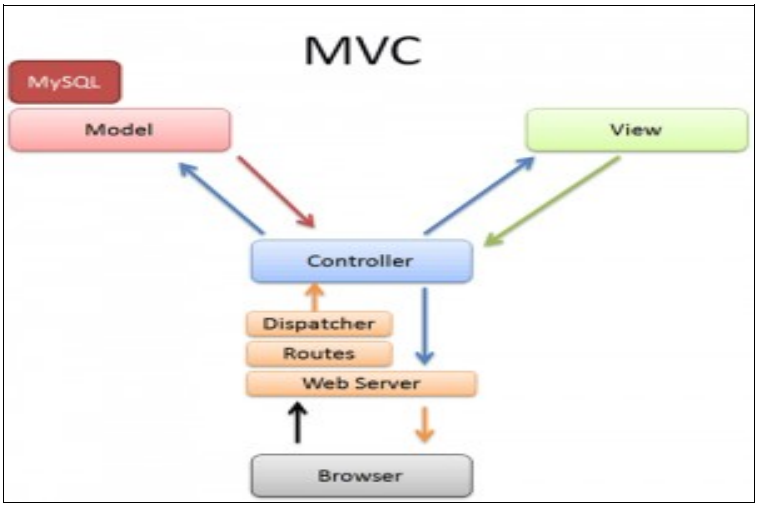
- Una community attiva ed amichevole;
- Sistema di licenza flessibile;
- Compatibile con le versioni 4 e 5 di PHP ;
- CRUD integrato per l'interazione con i Database;
- Application Scaffolding ;
- Generazione di codice;
- Architettura MVC ;
- Request Dispatcher con clean, custom URLs e routes;
- Validazione incorporata dei dati;
- Gestione dei template veloce e flessibile (es. sintassi PHP con helpers);
- View Helpers per AJAX, JavaScript, HTML Forms e più;
- Componenti per l'utilizzo di E-Mail, cookie, sicurezza, sessioni e request

- Gestione delle ACL flessibile;
- Data Sanitization;
- Flexible Caching;
- Supporto per la Localizzazione.

1.2 Comprendere il pattern MVC

cakePHP segue il Design Pattern MVC.

Figura 14 - Design Pattern MVC in cakePHP



Fonte: http://code.google.com/p/theca2/wiki/Architecture

La Figura 14 mostra una richiesta MVC di base. Per illustrarla, si assuma che un client denominato "Iovino Vincenzo" abbia effettuato un semplice click sul link "Compra subito una torta personalizzata!" dalla home page della nostra Web Application:

1. "Iovino Vincenzo" fa un semplice click su http://www.example.com/cakes/buy ed il suo browser invia una richiesta al Web Server;
2. Il dispatcher controlla l'URL di richiesta (/cakes/buy) e la affida al Controller;
3. Il Controller esegue la logica specifica della 'Web Application (es. controlla se "Iovino Vincenzo" abbia effettuato il login). Inoltre, al fine di ottenere l'accesso ai dati della 'Web Application, utilizza i Model che sono, di solito, tabelle di Database, ma possono anche rappresentare entità LDAP , feed RSS, o file sul sistema. (es. il Controller usa un Model per ritrovare gli ultimi acquisti effettuati da "Iovino Vincenzo" sul Database);
4. I dati vengono forniti ad una View, una volta che il Controller ha effettuato le proprie "magie", la quale riceve questi dati e diviene pronta a visualizzarli per il Client. Le View in cakePHP sono di solito in formato HTML, ma potrebbero essere anche un documento PDF, XML oppure un oggetto JSON a seconda delle necessità.

5. La View, una volta che ha utilizzato i dati forniti dal Controller per creare la presentazione, invia il contenuto della stessa al browser di "Iovino Vincenzo".

Quasi tutte le richieste di una Web Application possono seguire questo semplice Design Pattern. Si Aggiungeranno, in seguito, ulteriori dettagli su quali sono le specifiche di cakePHP.

1.2.1 Vantaggi

Perché usare il Design Pattern MVC? Si tratta di un Design Pattern reale e testato che trasforma una Web Application in un pacchetto manutenibile, modulare e rapidamente sviluppata. Dividere i compiti della Web Application in Controller, Model e View rendono le Web Application leggere. Nuove caratteristiche potranno essere facilmente aggiunte e costruire il nuovo sulla base del vecchio, diventerà un gioco da ragazzi. La progettazione modulare e separata permette, inoltre, di lavorare in simultanea tra sviluppatori e grafici ed include la possibilità di realizzare rapidamente prototipi. La separazione consente anche agli sviluppatori di modificare una parte della Web Application senza avere effetti sulle altre.

1.2.2 Svantaggi

Perché non usare il Design Pattern MVC? Aumento della complessità della Web Application attraverso l'introduzione, nell'architettura, di alcune classi aggiuntive dovuto alla separazione tra Model, View e Controller

2. Principi base

Il Framework cakePHP offre una robusta base per le Web Application. Infatti, è in grado di gestire ogni aspetto, dalla richiesta iniziale dell'utente, lungo tutto il percorso fino al rendering finale della pagina Web. Poiché il Framework segue il principio del Design Patter MVC è inoltre possibile personalizzare ed estendere molti aspetti della Web Application.

Offre anche una struttura organizzativa di base, dai nomi dei file ai nomi delle tabelle del Database, garantendo al contempo alla Web Application coerenza e logica. Questo concetto è semplice ma potente.

2.1 La struttura

cakePHP possiede le classi per Controller, Model e View ma possiede anche classi aggiuntive ed oggetti che rendono lo sviluppo MVC un pò più veloce e piacevole. Component, Behavior e Helper sono classi che forniscono estensibilità e riusabilità per aggiungere velocemente funzionalità alle classi MVC di base di una data Web Application.

2.1.1 Estensioni dei Controller

Un Component è una classe che aiuta nella logica del Controller. Se si possiede un codice e lo si vuole condividere tra più Controller (o applicazioni), un Component è solitamente un buon posto dove porlo (es. la classe Core E-MailComponent rende la creazione e l'invio delle E-Mail facile).

Piuttosto che scrivere un metodo del Controller per ogni singolo controllo che esegue la stessa logica, lo si può impacchettare in un componente che può essere condiviso.

Sono anche forniti di metodi di callback che sono disponibili nel caso si voglia inserire del codice tra le operazioni eseguite dal Core di cakePHP.

- beforeFilter() - Viene eseguito prima di ogni altra azione del Controller;
- beforeRender() - Viene eseguito dopo il codice del Controller ma prima che la View venga chiamata;
- afterFilter() - Viene eseguito dopo che tutta la logica del Controller è stata eseguita, inclusa, la renderizzazione della View. Non vi dovrebbero essere, quindi, differenze tra il metodo afterRender() ed afterFilter() a meno che non sia stato richiamato manualmente il metodo render() del Controller e sia stato incluso una qualche logica aggiuntiva dopo questa chiamata.

2.1.2 Estensioni delle View

Un Helper è una classe che fornisce un aiuto nella scrittura della logica di una vista. Così come un componente è utilizzato per più Controller, allo stesso modo gli Helper sono delle parti di logica di presentazione che possono essere acceduti e condivisi tra più View. Uno degli helper del Core AjaxHelper, rende la gestione delle richieste Ajax nelle View molto più semplice.

La maggior parte delle View hanno pezzi di codice che sono utilizzati in maniera ripetuta. cakePHP facilita il riuso di questo codice attraverso i layout e gli element. Per default ogni View che viene renderizzata da un Controller è inserita all'interno di un layout. Gli Element sono utilizzati, invece, quando degli snippet leggeri di contenuto vengono usati per più View.

3.3.3 Estensioni dei Model

Allo stesso modo i Behavior lavorano in maniera simile per apportare le proprie funzionalità a differenti Model (es. "Castaldi Luca" inserisce dei dati utente in una struttura ad albero. Potrebbe specificare che il Model User ha un comportamento come un albero ed ottenere, gratis le funzionalità per rimuovere, aggiungere e spostare i nodi della struttura ad albero sottostante).

I Model sono supportati anche da un'altra classe denominata DataSource. che sono un'astrazione che abilita i Model a manipolare differenti tipi di dati in maniera consistente. Mentre la maggior parte dei dati in cakePHP proviene spesso da un Database, si potrebbero scrivere dei DataSource aggiuntivi che forniscano ai Model il modo per rappresentare feed RSS, file CVS, entità LDAP o eventi iCal e permettono di associare record tra sorgenti differenti. Piuttosto che essere limitati ad un join SQL, dando la possibilità di dire al Modello LDAP che è associato a vari eventi iCal.

Così come i Controller, i Model sono costruiti con metodi di callback come segue:

- beforeFind();
- afterFind();
- beforeValidate();

- beforeSave();
- afterSave();
- beforeDelete();
- afterDelete();

2.1.4 Estensione delle Web Application

Sia i Controller, gli helper che i Model hanno una classe genitore che si può utilizzare per definire dei cambiamenti per tutta della Web Application. AppController (che è situata a /app/app_controller.php), AppHelper (che si trova su /app/app_helper.php) e AppModel (in /app/app_model.php), questi sono degli ottimi posti per aggiungere quei metodi che si vuole condividere tra tutti i Controller, gli helper od i Model della Web Application.

Anche le Route, pur non essendo una classe o un file, giocano un ruolo importante nelle richieste fatte da cakePHP. Le definizioni delle Route dicono a cakePHP come mappare nelle URL le azioni dei Controller. Il comportamento di default assume che alla URL "/controller/action/var1/var2" corrisponda controller::action($var1, $var2), ma si può usare Route per castomizzare le URL in maniera differente e come esse vengano interpretate dalla Web Application.

Qualche funzione complessa di una Web Application merita di essere posta in un apposito package. Un plug-in è appunto un package di Model, Controller e View che svolge uno specifico compito e che può essere utilizzato in molteplici Web Application (es. un sistema di gestione degli utenti o un blog semplificato).

2.1.4.1 Caricare da Plug-in

Caricare classi nei Plug-In funziona quasi allo stesso modo che caricare classi e app core a patto che si specifichi il Plug-In dal quale si sta caricando.

1. App::import('Model', 'PluginName.Comment');

2.1.4.2 Attenti al Model del Plug-in

Quando si crea un Plug-In, si può utilizzare la maggior parte delle convenzioni comuni, facendo attenzione però al Model (es. se si ha un Plug-In chiamato categories che consente di gestire le categorie strutturate con gerarchia ad albero e in questo Plug-In dichiaro il Model Category, se nella Web Application si vuole creare una relazione con il Model Article si dovrà fare attenzione ad inserire nel campo "className" il nome del plug-in anteposto al nome del Model).

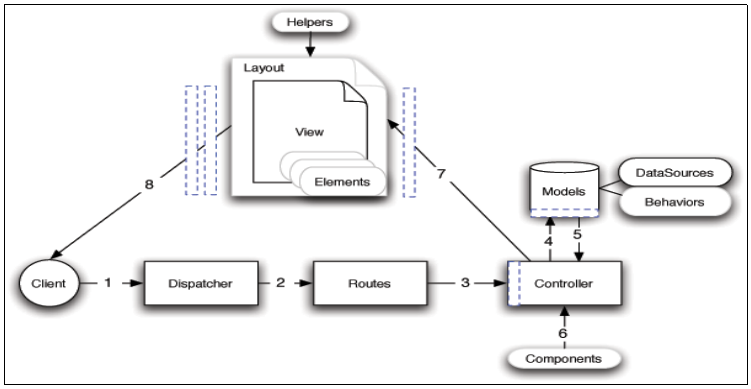
```
1. <?php
2. var $belongsTo = array(
3.
4.  'Category' => array(
5. 'className' => 'Categories.Category',
```

```
6.  'foreignKey' => 'category_id'
7.  )
8.  );
9.  ?>
```

2.2 Una tipica richiesta

Si è già visto gli “ingredienti di base” di cakePHP, adesso si vedrà come ogni oggetto lavora insieme agli altri per completare una richiesta di base. Continuando con la richiesta dell'esempio originale, immaginiamo che il nostro amico "Castaldi Luca" abbia appena fatto un semplice click sul link "Compra subito la tua torta personalizzata!" sulla pagina di destinazione di una Web Application cakePHP.

Figura 15 - Interazione tra Oggetti in cakePHP



Nero => elemento richiesto, Grigio => elemento opzionale e Blu => callback

Fonte: http://book.cakephp.org

La Figura 15 mostra una tipica Request secondo il Design Pattern MVC, alle richieste dell’utente “Castaldi Luca”.

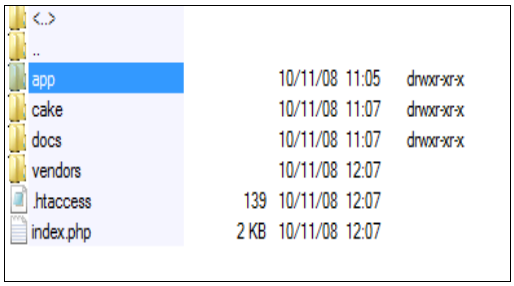
- “Castaldi Luca” fa un semplice click su http://www.example.com/cakes/buy ed il suo browser crea una richiesta sul Web Server;
- La richiesta URL, utilizzando il Router che estrae i parametri della richiesta, è connessa ad una specifica azione di un Controller (es. un metodo in una specifica classe di Controller) che in questo caso è il metodo buy() del Controller CakesController. Il callback del Controller beforeFilter() viene richiamato prima che ogni altra logica venga eseguita. Il Controller può usare più Model per avere accesso ai dati della Web Application. (es. il Controller utilizza un Model per estrarre gli ultimi acquisti di “Castaldi Luca” dal database). Qualsiasi Model callback, behavior e DataSource può essere applicato durante questa operazione. Mentre l'uso di un Model non è richiesto, tutti i Controller cakePHP inizialmente ne richiedono almeno uno;

- Dopo che il Model ha ritrovato i dati, li ritorna al Controller. Può essere applicato un Model di callback. Il Controller può usare componenti per ulteriori rifiniture dei dati o eseguire altre operazione (es. manipolazioni di sessione, autentificazione o invio di una E-Mal);
- Una volta che il Controller ha utilizzato i Model ed i componenti per preparare a sufficienza i dati, questi sono consegnati alla View utilizzando il metodo del Controller set(). Anche in questo punto possono essere usati Controller callback prima che i dati siano spediti. La logica della View viene a questo punto eseguita, ciò può includere l'utilizzo di diversi element e/o helper. Per default una View viene renderizzata insieme ad un layout. Controller callback aggiuntivi (es. afterFilter) possono essere applicati;
- Il codice della View completo e renderizzato viene inviato al browser di “Castaldi Luca”.

2.3 Struttura del file

Andiamo a dare un'occhiata a come cakePHP appare quando è stato appena installato. Già si conosce come svolge il proprio lavoro, in merito alle richieste di tipo MVC di base, ma bisogna imparare anche come i suoi file sono organizzati.

Figura 16 – La Directory app cakePHP



Fonte: http://www.ajaxlines.com/ajax

Quando si scarica cakePHP si notano quattro directory principali. La directory “app” sarà quella dove si svolge la maggior parte delle "magie": è dove i file della Web Application saranno salvati. Il folder “cake” è dove si fanno le "magie". La directory “docs” è per i file di tipo readme, changelog e per le informazioni di licenza d'uso. In ultimo, la directory “vendors” è quella dove mettere le librerie di terze parti di cui si ha bisogno per poi utilizzarle all'interno della Web Application.

Tramite il file .htaccess viene gestito l’url rewriting per tutte le chiamate. Nel caso in cui si voglia escludere una directory, dalla Web Application, delle regole impostate (es. si vuole installare un blog nella root) è sufficiente inserire una riga di codice nel file dopo la direttiva “*RewriteEngine on*“:

2.3.1 La directory App

La directory “app” è il posto dove si svolge la maggior parte del lavoro di sviluppo e contiene

le cartelle elencate qui di seguito nella tabella 1.

Tabella 1 - Directory "app" in cakePHP

Nome	Descrizione
config	Contiene i (pochi) file di configurazione che CakePHP utilizza. I dettagli di connessione col Database, bootstrapping, file di configurazione del Core e altro dovrebbero essere salvati qui.
controllers	Contiene i vostri Controller ed i loro componenti.
locale	Qui vanno posti i file per le stringe di internazionalizzazione.
models	Contiene model, behavior e datasource per la Web Application.
plugins	Contiene i pacchetti plugin utilizzati.
tmp	Questo è il posto dove CakePHP salva i file temporanei. I dati che vengono salvati dipendono da come avete configurato CakePHP, ma questa directory è di solito usata per salvare le descrizioni dei Modelli, i file di log e, talvolta, anche le informazioni di sessione.
vendors	Ogni classe o libreria di terze parti dovrebbe essere posizionata qua. Facendo questo sarà facile accedervi utilizzando la funzione vendor(). Osservatori attenti potrebbero aver notato come questo possa apparire ridondante, dato che esiste anche una directory vendors al livello principale della nostra struttura. Andremo ad osservare le differenze tra le due cartelle quando andremo a discutere della gestione di diverse Web Application e dei settaggi di sistema più complessi.
views	I file di presentazione vanno salvati qui: file per element, pagine di errore, helper, layout e View.
webroot	In un setup di produzione, questa cartella dovrebbe servire come root della vostra Web Application. Sottocartelle poste qui possono servire anche a contenere CSS stylesheets, immagini e file JavaScript.

Fonte: http://book.cakephp.org

2.3.2 App::import

app::import serve a caricare un file (es: app/venditori/example.php) partendo dalla directory

app. Per assicurare che il file venga caricato su tutti i server, si utilizza la seguente sintassi:

```
1. App::import('Vendor', 'example', array('file' => 'Example.php'));
```

A prima vista App::import sembra complesso, ma in molti casi solo due argomenti sono richiesti e caricare classi addizionali è diventato più veloce in cakePHP.

2.4 Conventions

Nel mentre si perde un pò di tempo ad imparare le convenzioni in uso su cakePHP, nel lungo periodo si risparmierà molto: “seguendo” le convenzioni, infatti, si ottiene gratuitamente l'accesso a molte funzionalità e ci si sarà liberati dalla necessità di mantenere un pacchetto di file di configurazione. L'utilizzo delle convenzioni, inoltre, rende lo sviluppo molto uniforme, permettendo ad altri sviluppatori di unirsi e di collaborare molto più facilmente.

Le convenzioni di cakePHP sono il risultato della distillazione di anni di esperienza nello sviluppo del Web e delle migliori pratiche. Nel mentre se ne suggerisce l'uso nello sviluppo con cakePHP, si può segnalare che la maggior parte di esse possono essere facilmente aggirate. Ciò è specialmente utile nella gestione di sistemi propriet.

2.4.1 Convenzioni per i file e le classi

In generale, i nomi dei file sono in minuscolo, mentre i nomi delle classi sono del tipo CamelCased (es. la classe KissesAndHugsController sarà rintracciabile nel file kisses_and_hugs_controller.php). Il nome della classe contenuta in un file può tuttavia non essere così strettamente correlato al nome dello stesso (es. la classe E-MailComponent è contenuta in un file chiamato E-Mail.php e la classe HtmlHelper in un file chiamato html.php).

2.4.2 Convenzioni di Model e Database

I nomi per le classi Model sono singolari (es. CamelCased. Person, BigPerson e ReallyBigPerson).

I nomi delle tabelle corrispondenti ai Model cakePHP sono plurali e minuscoli. Le tabelle che implementano le classi sopra menzionate dovrebbero essere denominate rispettivamente: people, big_people e really_big_people.

Foreign keys nelle relazioni hasMany, belongsTo o hasOne sono riconosciute di default come il nome (singolare) del Model associato seguite da _id. Quindi se il pasticcere “Spinelli Raffaele” si trova “hasMany torte”, la tabella torte si riferirà al pasticcere nella tabella pasticcere mediante la foreign key pasticcere_id.

I join tra le tabelle, utilizzate nella relazione hasAndBelongsToMany, dovrebbe essere denominati in accordo ai nomi delle tabelle che vanno ad unire, in ordine alfabetico (es. apples_zebras piuttosto che zebras_apples.). Tutte le tabelle con le quali cakePHP interagisce (con l'eccezione di tabelle join), richiedono una singola chiave primaria per identificare ogni riga. Se desideri utilizzare una tabella che non ha un singolo campo come chiave primaria, come le righe della post_tags tabella join, la convenzione di cakePHP è che un singolo campo di chiave primaria è aggiunto alla tabella. Non supporta chiavi primarie complesse. Ed in virtù di questo nel

momento in cui si vuole direttamente manipolare i dati join delle tabelle, bisogna utilizzare direttamente la chiamata query, o aggiungere un campo di chiave primaria per usarlo come un normale Model. E.G.:

```
1. CREATE TABLE posts_tags (
2. id INT(10) NOT NULL AUTO_INCREMENT,
3. post_id INT(10) NOT NULL,
4. tag_id INT(10) NOT NULL,
5. PRIMARY KEY(id));
```

2.4.3 Convenzioni Controller

I nomi per le classi dei Controller sono plurali (es. CamelCased) e finiscono in Controller (es. PeopleController e LatestArticlesController)

La prima funzione che si scrive per un Controller dovrebbe essere la funzione index(). Quando una richiesta specifica è Controller ma non una azione, il comportamento di default di cakePHP è quello di renderizzare la funzione index() del Controller stesso (es. una richiesta a http://www.example.com/apples/ mappa una chiamata alla funzione index() di ApplesController, mentre http://www.example.com/apples/View/ mappa una chiamata alla funzione View() di ApplesController).

Si può anche cambiare la visibilità delle funzioni in cakePHP precedendo il nome della funzione stessa con il carattere di sottolineatura. Se una funzione in un Controller è preceduta da un carattere di sottolineatura, la funzione non sarà raggiungibile via-Web, ma resterà comunque disponibile per uso interno:

```
1. <?php
2. class NewsController extends AppController {
3. function latest() {
4. $this->_findNewArticles();
5. }
6. function _findNewArticles() {
7. //Logic to find latest news articles
8. }
9. }
10. ?>


11. <?php
12. class NewsController extends AppController {
```

```
13. function latest() {
14. $this->_findNewArticles();
15. }
16. function _findNewArticles() {
17. //Logic to find latest news articles
18. }
19. }
20. ?>
```

Mentre la pagina http://www.example.com/news/latest/ sarà accessibile all'utente come di solito, tentando di accedere alla pagina http://www.example.com/news/_findNewArticles/ si riceverà un errore, perché la funzione è preceduta da un carattere di sottolineatura.

2.4.3.1 URL Considerations for Controller Names

Come si è appena visto, singole parole del Controller mappano facilmente un semplice indirizzo URL minuscolo. (es. ApplesController, definito nel file “apples_Controller.php” potrà essere visitato all'indirizzo http://example.com/apples). Parole multiple del Controller mappano un cameBacked URL tenendo la forma plurale. (es. RedApplesController e red_apples_Controller.php mappa http://example.com/redApples).

2.4.3.2 Esempio: Post in un Blog

Il controller esporrà un metodo per visualizzare i post inseriti, uno per aggiungerli ed uno per eliminarli. Come richiesto da cakePHP andrà inserito all'interno di /app/controllers con il nome di post_controller.php:

```
1. <?php
2. class PostsController extends AppController
3. {
4. var $name = 'Posts';
5.
6. unction index()
7. {
8. $this->set('posts', $this->Post->findAll());
9. }
10.
11. function view($id)
12. {
```

```
13. $this->Post->id = $id;
14. $this->set('post', $this->Post->read());
15. }
16.
17. unction add()
18. {
19. // ...
20. // ...
21. }
22.
23. }
24. ?>
```

Il Controller serve ad esporre i metodi necessari. Dato che abbiamo deciso di seguire le convenzioni sui nomi sarà possibile accedere ai modelli implementati direttamente attraverso $this->NOME_MODELLO. Il metodo index registra un array con tutti i post inseriti utilizzando il metodo set, che rende disponibile questo array alla vista che andremo successivamente a creare; il metodo View invece recupera solamente un determinato post in base al valore passato via URL dopo il path che indica la funzione da richiamare nel Controller.

2.4.4 Convenzioni per le View

I file per i template delle View vengono denominati in accordo alle funzioni del Controller che verranno mostrate nella forma undersCored. La funzione getReady() della classe PeopleController cercherà il template per la View in /app/Views/people/get_ready.ctp. Il Pattern di base è /app/Views/Controller/undersCored_function_name.ctp.

Denominando le parti della Web Application utilizzando le convenzioni di cakePHP, otterrete delle funzionalità senza il fastidio e la necessità di mantenere file di configurazione. Qui è riportato un elenco finale che tiene conto delle convenzioni:

- Database table: "people"
- Model class: "Perso"', si trova in /app/models/person.php
- Controller class: "PeopleController", si trova in /app/controllers/People_Controller.php
- View template, si trova in /app/views/people/index.ctp

Utilizzando queste convenzioni, cakePHP conosce che una richiesta a http://example.com/people/ deve essere mappata ad una chiamata alla funzione index() del Controller PeopleController, dove il Modello Person è automaticamente disponibile (ed automaticamente legato alla tabella “people” del Database) e renderizzata su un file. Nessuna di queste relazioni è stata configurata in alcun modo, a parte la creazione delle classi e dei file, che avreste dovuto creare ad ogni modo. Adesso che sono stati introdotti i fondamenti di cakePHP, ci

si potrà cimentare in seguito.

3. Componenti Core

cakePHP ha diversi componenti già built-in, inclusi nella distribuzione, che garantiscono funzionalità "out of the box" per diverse operazioni di uso comune. I componenti sono riportati in Tabella 2:

Tabella 2 - Componenti Core in cakePHP

Nome	Descrizione
Session	Il componente session permette di avere un wrapper indipendente dai meccanismi di salvataggio intorno alle sessioni PHP.
Security	Il componente security consente di impostare un livello di sicurezza più stretto e gestire l'autenticazione HTTP.
Acl	Il componente ACL fornisce una interfaccia facile da usare per delle Access Control Lists su Database o file .ini
RequestHandler	Il componente request handler consente di ispezionare ulteriormente le richieste dei visitatori e passare all'applicazione una serie di informazioni riguardo al tipo di contenuto e alle richieste.
Auth	Il componente auth fornisce un sistema di autenticazione facile da usare usando una gran varietà di processi di autenticazione, come Controller callbacks, Acl e Object callbacks.
Cooki	Il componente cookie si comporta in maniera simile al componente Session, dato che fornisce un wrapper al supporto di PHP nativo per i cookie.
E-Mail	Una interfaccia che può essere usata per inviare E-Mail usando uno dei vari mail transfer agents, incluso un SMTP o la funzione mail() di PHP.

Fonte: http://book.cakephp.org

4. Core Behaviors

Per poter aggiungere funzionalità ai Model realizzati dal programmatore. cakePHP fornisce una serie di componenti Built-In (es. Alberi).

- ACL;
- Containable,
- Translate;
- Tree.

5. Core Helpers

Gli Helpers sono le classi di componenti, per il livello di presentazione delle richieste. Contengono le logiche di presentazione che sono condivise da diverse viste, elementi o layout.

È possibile utilizzare “aiutanti” in cakePHP facendo un Controller che si occupa di esso. Ogni controllore ha una proprietà $helpers aiutanti che elenca gli helpers che devono essere messi a disposizione della vista. Per abilitare un helper nella in una vista, basta aggiungere il nome dell'helper nell'array $helpers del Controller.

```
1. <?php
2. class BakeriesController extends AppController {
3. var $helpers = array('Form', 'Html', 'Javascript', 'Time');
4. }
5. ?>
```

Puoi anche aggiungere helpers in un'azione, così che saranno disponibili per quell'azione e non nelle altre azioni del Controller. Questo salva risorse per le azioni che non usano quel Controller e aiuta inoltre a tenere i Controller meglio organizzati.

```
1. <?php
2. class BakeriesController extends AppController {
3. function bake {
4. $this->helpers[] = 'Time';
5. }
6. function mix {
7. // The Time helper is not loaded here and thus not available
8. }
9. }
10. ?>
```

Se un Core helper (o uno su Cakeforge o su Bakery) non soddisfa le tue esigenze, gli helpers sono facili da creare. Supponiamo che vogliamo creare un helper che sarà utilizzato per stampare un link formatto CSS che necessiti in diverse parti della tua Web Application. Per inserire la tua logica nella struttura di cakePHP già esistente, devi creare una nuova classe nella cartella /app/views/helpers. Chiamerò il nostro helper LinkHelper. La classe PHP risulterà qualcosa del

genere:

```
1. <?php
2. /* /app/Views/helpers/link.php */
3. class LinkHelper extends AppHelper {
4. function makeEdit($title, $url) {
5. // Logica per creare i link...
6. }
7. }
8. ?>
```

Ci sono diversi metodi inclusi negli Helper di cakePHP che potresti utilizzare: output(string $string) usa questa opzione per manipolare qualsiasi dato nelle View.

```
1. <?php
2. function makeEdit($title, $url) {
3. // Use the helper's output function to hand formatted
4. // data back to the View:
5. return $this->output(
6. "<div class=\"editOuter\">
7. <a href=\"$url\" class=\"edit\">$title</a>
8. </div>" );
9. }
10. ?>
```

6. Librerie utili

cakePHP include alcune librerie di utilità generale che possono essere richiamate ovunque nella Web Application, con Set e HttpSoket.

- Inflecto;
- String;
- Xml;
- Set;
- Security;
- Cache;
- HttpSocket.

7. Struttura di una Web Application – Hello, World!

7.1 Introduzione

cakePHP è un Framework MVC. Model, View, Controller sono i componenti principali. In generale, il Model è un oggetto e viene utilizzata per comunicare con il database per svolgere operazioni di inserimento, cancellazione e interrogazione. La View, come dal suo nome. è utilizzata per visualizzare il risultato. Di solito contiene un sacco di codice HTML, ma poco di codice PHP. Il Controller viene utilizzato per gestire le richieste HTTP e fare un pò di logica. Il Controller è centrale in un modello MVC. Si chiama il Model per ottenere i dati dal database, fare qualche lavoro, e inviare il risultato a una View da visualizzare.

7.2 Directory cakePHP

CakePHP si trova in opt/lampp/webcake. All'interno di essa si trovano le seguenti cartelle:

1. **app** - Contiene tutto il codice, i modelli e le risorse specifiche per la Web Application
    - **config** - Include tutte le configurazioni della propria Web Application tra cui database, routes e custom paths;
    - **controller** - Contiene tutti i Controller della Web Application. I controller, a loro volta, contengono la logica di business della Web Application;
    - **models** - Contiene tutti i Model, che sono responsabili di fornire dati alla tua applicazione, e salvandola dalla vostra applicazione per l'origine dati, di solito un file o database. I modelli sono responsabili per cose come la convalida dei dati. Modello di file stessi sono spesso solo una breve definizione di che tipo di dati di questo modello è.
    - **plugin** - Plug-In sono applicazioni di terze parti che si integrano con cakePHP.
    - **views** - Contiene i file di presentazione che generano ciò che l'utente arriva a vedere. Questi file contengono il reale HTML che viene inviato al browser quando un utente richiede un'azione. C'è un gruppo di punti di vista per ciascuno dei controller.
        - **layout** - Questa è una sottocartella di punti di vista e contiene i file di vista particolare, che "wrap around" la tua opinione prima di essere inviati al browser. (Pensate a come intestazione e piè di pagina in un unico file.)
1. **Webroot** - Contiene tutti i file che sono direttamente accessibili con un browser web.
    - **CSS** - Contiene tutti i vostri file CSS;
    - **files** - Contiene tutti i file da rendere disponibili per il download (es. .ZIP, ecc);
    - **img** - Contiene tutte le immagini;

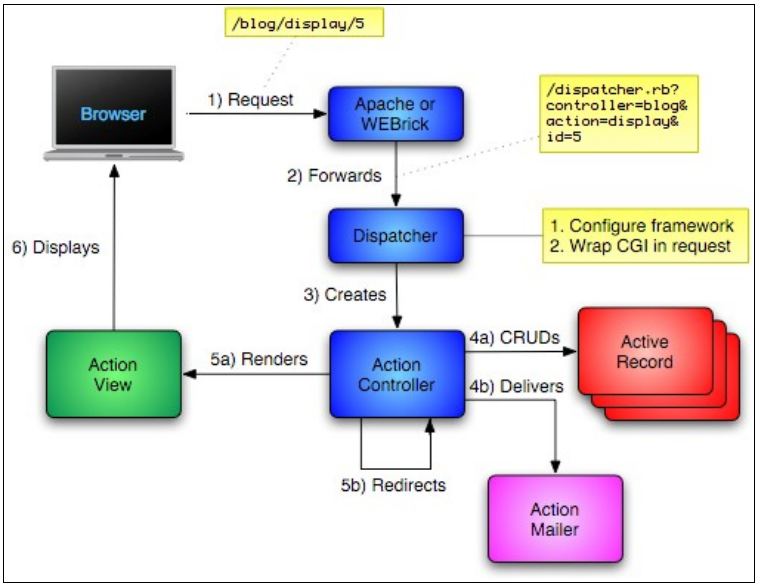
- **js** - Contiene tutti i vostri file JavaScript.

1. **cake** - Contiene le librerie cakePHP principali. Questa directory non richiede modifica.
2. **tmp** - Contiene i file temporanei e il debugging.
    - **cache** - Contiene i file memorizzati nella cache;
    - **logs** - contiene i file di log;
    - **test** - contiene i file di unit test per l'utilizzo tramite SimpleTest.
1. **vendors** - Contiene classi esterne PHP e librerie da utilizzare con cakePHP.

7.2 Le richieste dell'utente

Per capire come si comporterà la nostra Web Application, secondo il Design Pattern MVC, alle richieste dell'utente si osservi l'immagine riportata qui sotto in Figura 17.

Figura 17 - Diagramma Richiesta dell'Utente in cakePHP



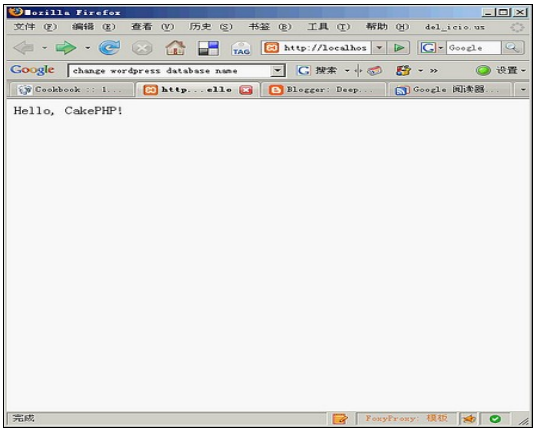
Fonte: http://www.opendcn.org/wiki/doku.php

1. l'utente "Bruno Andrea" richiede la visualizzazione dei post;
2. l **controller** di cakePHP gestisce le richieste dal server. Prende l'input dell'utente "Bruno Andrea" (es. URL e dati POST), esegue la logica applicativa, usa i Model per leggere e scrivere i dati dai database e da altre sorgenti, e infine, invia i dati in output al file di visualizzazione appropriato.
3. La visualizzazione finale delle pagine è definite tramite le **view** per la creazione dell'HTML;
4. l'HTML viene creato e visualizzato all'utente "Bruno Andrea".

La richiesta per essere interpretata dal front Controller di cakePHP deve essere composta in questo modo:

- http://www.nomeserver.ex/

Figura 18 – La Parola "Hello, World!" in cakePHP



Fonte: http://deepbluespaces.blogspot.com/2008/05/cakephp-hello-world.html

7.3 Creiamo la struttura

Per capire come si comporterà la nostra Web Application, secondo il Design Pattern MVC, alle richieste dell'utente analizziamo la figura 2,5 con un esempio di invocazione:

1. Creare un Virtual Host con Apache con le seguenti specifiche del server:
   - Indirizzo IP: 172.16.16.19;
   - Nome del virtualhost: http://www.webcake.ex;
   - Directory da pubblicare: /opt/lampp/webcake/public;
   - Directory dei log: /opt/lampp/webcake/logs.

   Per creare un Virtual Host con queste specifiche bisogna seguire tre operazioni:

   1. Aggiunge questa riga al file /etc/hosts:
      - 172.16.16.19 www.webzcake.ex;
   2. Creare il file di configurazione del Virtual Host:
      - /etc/apache2/sites-enabled/001-mioserver;

      il file conterrà:

      1. NameVirtualHost *80
      2. <VirtualHost *80>

3. ServerName www.mioserver.local
4. DocumentRoot /var/www/mioserver/httpdocs
5. CustomLog /var/www/mioserver/log/access.log combined
6. ErrorLog /var/www/mioserver/log/error.log
7. </VirtualHost>

3. Cerare la struttura delle directory da pubblicare
   - mkdir opt/lampp/webcake

2. Creare una cartella “app” all'interno della cartella “webcake”. Conterrà, in essa, tutti gli elementi che utilizzeremo nella Web Application: Controller, Model e View. Per motivi di sicurezza possiamo lasciare questa cartella al di fuori della Document Root del server Web in modo da non essere accessibile direttamente via Web.

7.3.1 Il nostro primo Model

Il nome Model sta dalla M di MVC ed è molto semplice:

```
1. <?php
2. class HelloWorld extends AppModel
3. {
4.     var $useTable = false;
5. }
6. ?>
```

Si deve inserire questo file in app/models/helloworld.php

7.3.2 Il nostro primo Controller

cakePHP utilizza CamelCode nameing per i nomi di classe e di funzione, ma nonostante ciò richiede i nomi di file in minuscolo.

Anche se abbiamo appena finito con il Model, bisogna prendere atto che non c'è ne bisogno. Infatti, i Model sono utilizzati per la connessione al database. Se non si ha intenzione di utilizzare il database basta inserire una linea di codice nel Controller

```
1. var $uses = null;
```

Tale codice dice al Controller a quale Model si può accedere.

Il nome Controller sta dalla C di MVC e deve semplicemente passare il testo "Hello, World!" per il layout e impostare il titolo della pagina.

```
1. <?php
2. class HelloWorldController extends AppController
```

```
3.  {
4.  var $name = 'HelloWorld';
5.  var $pageTitle = 'Hello, world!';
6.
7.   function index()
8.    {
9.     $this->set('data','Hello, World!');
10.   }
11. }
12. ?>
```

Si deve inserire questo file in app/controllers/helloworld_controller.php

7.3.3 La nostra prima View

Si affronta ora la parte V dall' MVC.

```
1.  <?php echo $data; ?>
```

Dovete mettere questo file in app/views/hello_world/index.thtml

Si noti che quando si esegue questo file dal browser quindi si esegue questo file come http://localhost/cakephp/hellos questo verrà eseguito normalmente la funzione di indice. Se vogliamo eseguire la funzione sayHello allora dobbiamo scrivere come http://localhost/cakephp/hellos/sayHello

Come fa un Controller ad ottenere i dati da un View? Di solito, le View contengono alcune forme per ottenere l'input degli utenti. Secondo il Design Pattern MVC, i dati di input vengono passati dal Controller.

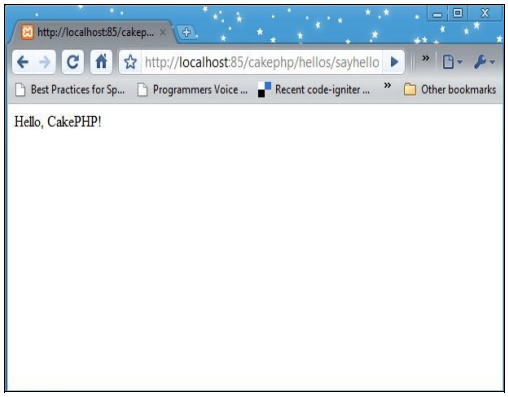
Ma c'è un altro problema. Il risultato contiene ancora un po' di cose come il grande titolo "CakePHP Rapido Sviluppo" e il logo della CakePHP in basso a destra. Perché è così? La risposta è "layout". Un layout è simile a una visualizzazione. Considerando un sito reale. Essa può contenere molte pagine. Ma tutte le pagine hanno lo stesso alcuni elementi, come il banner in alto, il testo di intestazione, CSS, ecc Non abbiamo bisogno di mettere queste cose in tutti i punti di vista. Solo bisogno di layout una volta. Layout è usato per la condivisione di elementi comuni e di solito è un file HTML compilato. Poiché non abbiamo creato il nostro layout, cakePHP userà proprio.

7.3.4 In Dettaglio

Prima di eseguirlo verificare che mod_rewrite sia acceso. Controlla il file http.conf di Apache all'interno della cartella conf di Apache., trovare la riga "LoadModule rewrite_module modules/mod_rewrite.so". Se è commentato, basta togliere i commenti (es. rimuovere "#"). Avviare o riavviare (se si è modificato http.conf), server Apache. Inserire il link http://lwww.webcake.ex nella barra degli indirizzi del browser.

Il risultato se non si verifica nessun problema:

Figura 19 – La Parola "Hello, World!" in http://www.webcake.ex



Fonte: http://learneveryday.net/php/framework/cake-php

7.4 Conclusioni

Una View di solito un file contenente codice HTML, ma molto poco di codice PHP. In cakePHP, un controller è associato con alcune View, a seconda del suo nome (es. nel controller "tester", il suo nome di classe è "TestersController" e deve essere situato in app controllers/testers_constroller. La sua vista deve essere situato in app/views /tester/.)

Ogni azione può avere la sua View di riferimento (es. la View di TesterConstroller::sayHello() è app/views/tester/say_hello.thtml).

Capitolo IV – I due Framework a confronto

Quando la strada non c'è, inventala!
(Robert Baden-Powell)

1. Introduzione

Per molti anni, cakePHP è stato il Framework più popolare al mondo ed ampiamente utilizzato per la realizzazione di Web Application mediante il linguaggio PHP. In tempi recenti, è stato introdotto un nuovo Framework, Zend Framework, che è stato definito in termini di standard e ha posto la questione di dover scegliere quale delle due tecnologie adottare ed in quali condizioni una Web Application realizzata con cakePHP dovesse utilizzare le nuove potenzialità di Zend Framework.

Le principali caratteristiche di cakePHP possono essere così riassunte:

- Un'architettura basta sul Design Pattern MVC;
- Compatibilità con PHP 4 e PHP 5;
- Liste di controllo degli accessi (ACL);
- Un approccio object-oriented;
- Active Records.

A tali caratteristiche, si aggiunge l'elevato grado di maturità che caratterizza il Framework, in quanto è stato realizzata nella sua prima versione nell'anno 2005, diventando una standard “de facto” per lo sviluppo di Web Application su PHP.

Dal lato opposto si pone Zend Framework le cui caratteristiche peculiari sono le seguenti:

- Un'architettura basata sul Design Pattern MVC, ma focalizzata soprattutto al Model fortemente agevolato;
- Capacità di gestione dei form mediante Zend_Form con il quale creare form in maniera più ‘Object Oriented’, ovvero il form come istanza dell’oggetto Zend_Form ed a loro volta, gli elementi che lo compongono, come istanze degli oggetti;
- Una collezione di librerie di supporto alla programmazione;
- Modello di gestione degli eventi e dei corrispondenti handlers;
- Ciclo di Vita di una Richiesta ben definito attraverso una serie di fasi.

La prima release di questo Framework è stata definita nel Marzo 2007, il che evidenzia un livello di maturità ancora basso rispetto a cakePHP.

In riferimento al Design Pattern MVC, su cui si basano entrambi i Framework, esso è anche noto come Front Controller, in quanto tutte le richieste che arrivano alla Web Application vengono

gestite e distribuite mediante un Controller unico della Web Application.

A questo punto, prendendo in considerazione le caratteristiche dei due Framework, si possono porre le seguenti domande:

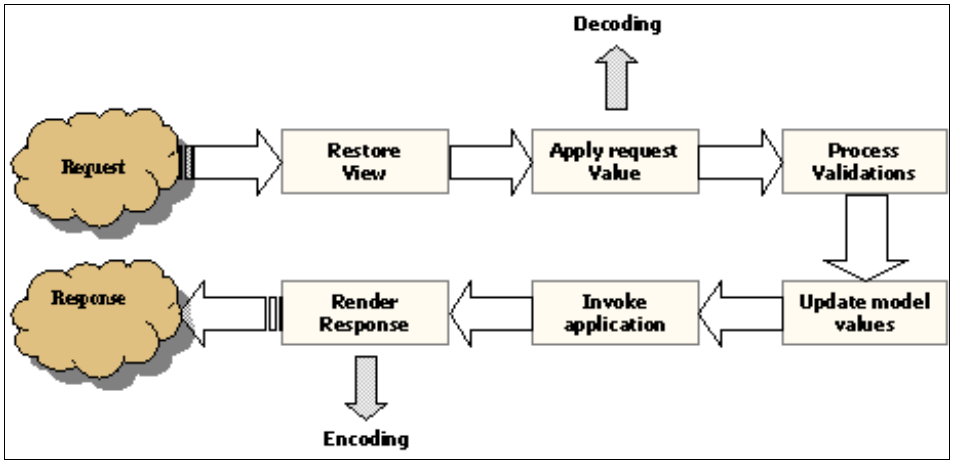
- Quale Framework adottare per lo sviluppo di una nuova Web Application?
- Disponendo di una Web Applicaion realizzata in cakePHP, conviene eseguire una migrazione da un Framework ad un altro?

Per rispondere a queste domande, si rende necessario un confronto approfondito tra queste due tecnologie, nonché bisogna tenere conto delle caratteristiche e delle dimensioni della Web Application da realizzare.

1.1 Ciclo di vita di una richiesta

Una delle principali differenze tra i due Framework risiede nel Ciclo di Vita di una Richiesta (request – lify – cycle), ossia cosa succede dal momento in cui arriva una richiesta (request) alla Web Application fino alla produzione della corrispondente risposta (response).

Figura 20 – Ciclo di Vita di una Richiesta in una Web Application



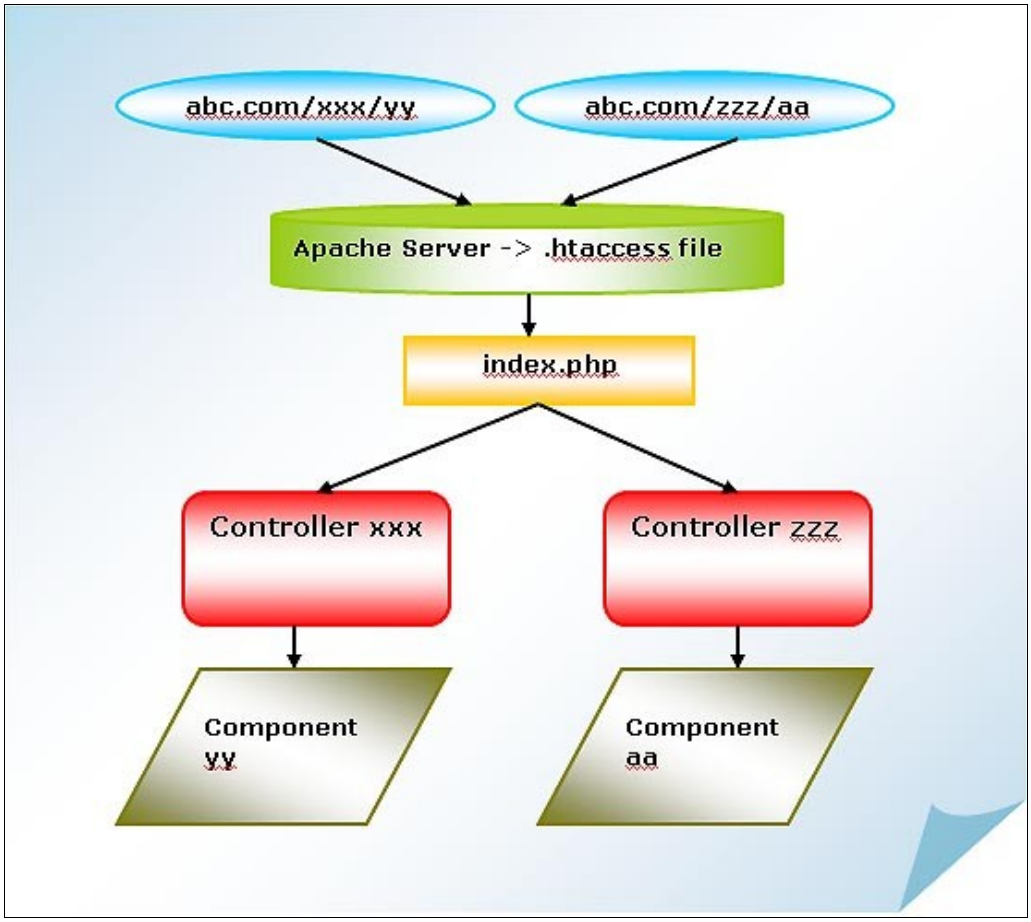
Fonte: http://www.giuseppesicari.it/

in Zend Framework, il Ciclo di Vita è relativamente semplice e può essere schematizzato nei seguenti passi principali:

- Le richieste vengono inviate, tramite file .htaccess, ad un file di bootstrap (es. index.php) che istanzia un oggetto del Front Controller per elaborare le richieste;
- Il Front Controller istanzia il Controller che si occupa della gestione dei post e chiama l'azione di visualizzazione;
- Viene istanziato un oggetto del Model che si occupa di estrarre i post dal DB. I risultati vengono restituiti dal Model al Controller;

- I dati elaborati dal Model vengono inviati alla View per la creazione dell'HTML finale.

Figura 21 – Ciclo di Vita di una Richiesta in Zend Framework



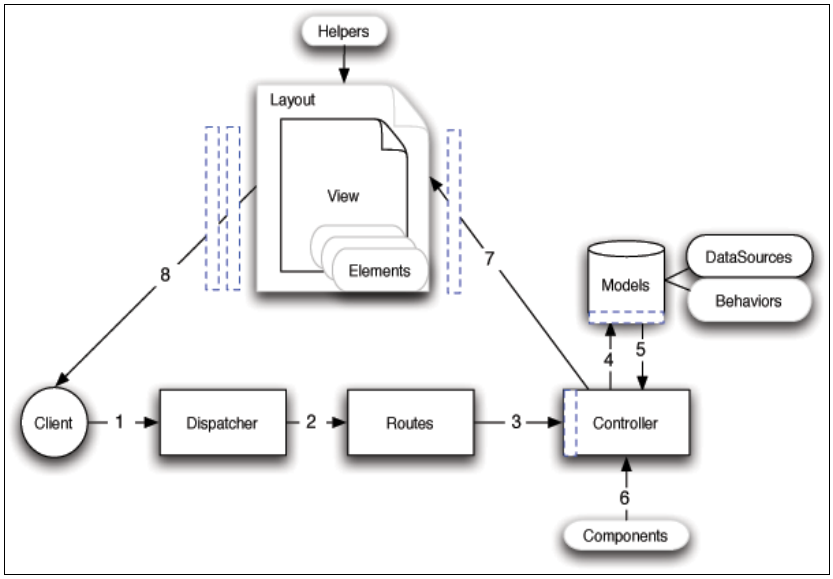
Fonte: http://segnalazionit.org/

cakePHP ha una gestione molto più complicata, caratterizzata anche dal fatto che è possibile distinguere tra più Modelli differenti richiamati dal Controller in base alla richiesta e può essere schematizzato nei seguenti passi principali:

- Il dispatcher controlla la URL di richiesta e la affida al Router;
- Il Router analizza la URL per estrarre i parametri della richiesta:
- Utilizzando le "rotte", la richiesta URL è connessa ad una specifica azione di un Controller che può usare più di un Model per avere accesso ai dati della Web Application;
- Dopo che il Model scelto dal Controller ha ritrovato i dati, li ritorna al Controller;
- Una volta che il Controller ha utilizzato i Model ed i componenti per preparare a sufficienza i dati, questi sono consegnati alla View ;

- Il codice della View completo e renderizzato viene inviato al browser.

Figura 22 – Ciclo di Vita di una Richiesta in cakePHP



Fonte: http://www.consulenza-web.com/category/cakephp/

La Figura 22 mostra una tipica Request.

1.2 Controller Tier

Entrambi i Framework prevedono un Controller realizzato attraverso un file in PHP che ha il compito di acquisire le richieste provenienti dal Client e determinare per ciascuna di esse il corrispondente handler, facente parte del Model, che dovrà gestirla. infine, sulla base dei risultati prodotti, lo stesso file si fa carico di determinare la View che rappresenterà la risposta fornita al client. l'implementazione del Controller è ovviamente differente e prevede le seguenti classi:

- cakePHP: la classe HelloWorldController;
- Zend Framework: la classe IndexController.

1.3 Validazione

In generale, tutti i sistemi software prevedono un'operazione di validazione, attraverso la quale valutare se i dati immessi dall'utente siano corretti rispetto a ciò che si attende. in questo modo, è possibile evitare errori di elaborazione, dovuti a valori non validi, in quanto tale controllo viene eseguito a priori.

Per quanto concerne tale aspetto, in riferimento al confronto tra i due Framework in questione, si può dire che cakePHP ha delle potenzialità superiori rispetto a Zend Framework, il quale però fornisce un meccanismo di estensione delle funzionalità di base, molto più semplice.

cakePHP fornisce una serie di validatori di default. Questi sono espressioni regolari che si utilizzano per validare i dati di un Model a runtime. I validatori di default a disposizione sono quattro:

- VALID_NOT_EMPTY: controlla che il valore di un attributo del Modello non sia vuoto;
- VALID_NUMBER: controlla che il valore di un attributo del Modello sia un numero valido;
- VALID_E-Mail: controlla che il valore di un attributo del Modello sia un'E-Mail valida;
- VALID_YEAR: controlla che il valore di un attributo del Modello sia un valore annuale valido;

Quando si definisce un Model si possono specificare delle espressioni regolari da utilizzare per la validazione che verranno applicate a runtime.

I validatori sono specificati nella variabile $validate, che è un array associativo che mappa ad ogni proprietà di un Modello il suo validatore. Ogni volta si tenterà di accedere in scrittura ad un elemento del Modello sarà la classe AppModel stessa che si occuperà di recuperare il validatore e, se presente, di applicarlo al valore assegnato. Se fosse necessario validare manualmente i campi è possibile utilizzare direttamente il metodo validate() del Modello.

Normalmente il processo di validazione e notifica degli errori di un form è integrato in modo trasparente nel Controller. Se fosse necessario è comunque possibile recuperare la lista dei campi che non hanno passato la validazione utilizzando il metodo invalidFields() del Modello.

Utilizzando espressioni regolari si può quindi sfruttarne tutte le potenzialità per validare i dati, mantenendo allo stesso tempo un'ottima organizzazione del codice. Il Modello definito sopra può quindi essere agganciato ad una View e ad un Controller in modo che la gestione degli errori avvenga quasi automaticamente.

1.4 Navigazione

Le Web Application. sviluppate prima dell'introduzione di Framework specifici, hanno sempre previsto la cosi detta navigazione statica nell'ambito della quale, per poter passare da una pagina all'altra, è necessario utilizzare i collegamenti ipertestuali, specificando per ciascuno di essi la pagina di destinazione.

cakePHP e Zend Framework introducono il concetto di navigazione dinamica, in cui la pagina di destinazione è determinata in seguito ad una elaborazione eseguita dalla business-logic. inoltre, in entrambi i Framework, la navigazione attraverso le pagine è specificato all'interno del file di configurazione, per cui si parla di “navigazione dichiarativa”, anche se utilizzano modalità differenti.

I due Framework prevedono, in linea di massima, tre elementi fondamentali:

- Pagina di origine;
- Outcome;
- Pagina di destinazione.

Questi vengono definiti attraverso le regole di navigazione ed i casi di navigazione. più

precisamente, ciascuna regola di navigazione è caratterizzata dalla pagina di origine e da una o più pagine di destinazione, verso le quali proseguire la navigazione sulla base dell'outcome che viene manipolato. quest'ultimo, in entrambi in Framework, può essere specificato direttamente all'interno di un tag di una pagina, relativamente ad un link oppure ad un bottone, oppure determinato in base all'esecuzione di un metodo richiamato dal Controler

1.5 Eccezioni

Uno degli aspetti che rappresenta un netto vantaggio di Zend Framework rispetto a cakePHP, è la gestione delle eccezioni. infatti, il Framework Zend Framework permette di dichiarare, all'interno di un file di configurazione, le eccezioni che potranno essere sollevate dalla Web Application ed i corrispondenti gestori.

in particolare, sono previste le due seguenti classi:

- cakePHP: la classe ErrorController;
- Zend Framework: la classe ErrorController.

In pratica, all'interno del file ErrorController.php può essere dichiarata un'eccezione che tipicamente è del tipo ModelException oppure una classe che la estende. Inoltre, ad essa viene associato un gestore che può essere quello di default oppure una classe che estende ErrorController. Nel momento in cui, durante l'esecuzione della Web Application, viene sollevata un'eccezione di questo tipo, essa viene intercettata e gestita in maniera completamente automatica, senza la necessità di blocchi di try-catch-finally all'interno del codice.

Questa potenzialità non è assolutamente prevista in cakePHP, nell'ambito del quale le eccezioni devono essere completamente gestite dallo sviluppatore, secondo le funzionalità offerte dal linguaggio PHP.

1.6 Sicurezza

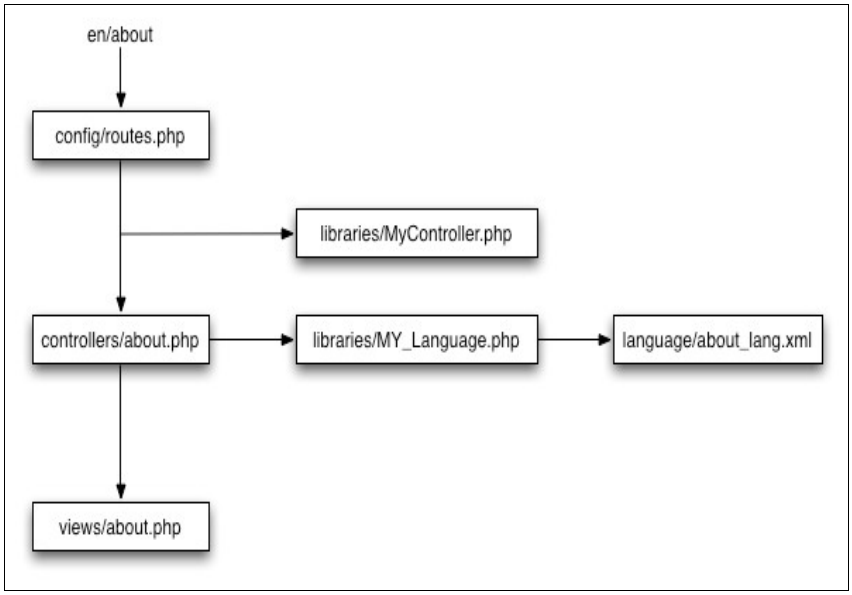
Una delle più importanti garanzie che deve fornire una Web Application ai propri visitatori è la sicurezza nella comunicazione e nello scambio dei dati. in moltissimi casi, la Web Application prevede il trasferimento di informazioni sensibili (es. il numero di carta di credito) per la quale si rende necessaria un'operazione di critografia. Mediante quest'ultima, è possibile rendere i dati leggibili a chiunque li intercetti durante il trasferimento, a meno del destinatario che ha a propria disposizione la chiave per decritografarne il contenuto. il meccanismo che permette di garantire la sicurezza nella comunicazione in rete prevede l'utilizzo del protocollo SSL (Secure Sochet Layer) ed in particolare del protocollo HTTPS (HTTP basato su SSL). Per quanto riguarda questo aspetto, di due Framework a confronto sono perfettamente uguali, in quanto non prevedono approcci differenti per garantire la sicurezza delle pagine delle Web Application realizzate.

1.7 Internazionalizzazione (i18n)

i18n è un processo di adattamento di un prodotto, pensato e progettato per un mercato o un ambiente definito, ad altri mercati o ambienti, in modo particolare altre nazioni e culture.

I prodotti che possono essere l'oggetto di tali processi sono i più svariati: dalla pubblicità (es. televisiva, editoriale, ecc.) ai software (es. sistemi operativi, applicazioni, programmi, ecc.), dai siti web ai manuali d'uso, dalle pubblicazioni mediche e scientifiche, alle etichette dei prodotti venduti sul mercato internazionale.

Figura 23 – Library for CodeIgniter



Fonte: http://maestric.com/doc/php/codeigniter_i18n_2008_04_24

L'internazionalizzazione è caratteristica comune di cakePHP e Zend Framework che viene praticamente gestita allo stesso modo. Infatti, entrambi i Framework offrono la possibilità di definire uno o più Resource Bundle (Message Resource), all'interno dei quali ci sono i messaggi di testo che devono essere visualizzati nelle pagine della Web Application, in lingue diverse. In questo modo, sulla base della localizzazione geografica dell'utente, la Web Application riesce ad utilizzare il Resource Bundle corretto, contenenti i messaggi nella lingua adatta.

1.8 Configurazione

Per configurare il Framework cakePHP, in modo che sia in grado di accedere correttamente ai dati, bisogna rinominare il file /app/config/database.php.default in /app/config/database.php e successivamente modificare l'assegnazione alla variabile $default in modo che i dati specificati aderiscano alle impostazioni del nostro server. Come driver è possibile utilizzare MySql, e come valore di connect mysql_pconnect, in modo da mantenere persistente la connessione tra varie sessioni.

Compiuti questi semplici ma necessari accorgimenti è possibile abilitare la riscrittura degli URL grazie al modulo mod_rewrite, in modo che questi siano più leggibili. Come accennato negli articoli precedenti è buona norma adottare questa soluzione, soprattutto se si desidera una

completa indicizzazione del proprio sito ed un meccanismo di URL più semplice da ricordare.

Lo Zend Framework è una completa collezione di classi da impiegare nella realizzazione di applicazioni PHP di media e grande complessità. Ci sono classi per tutte le esigenze quindi è inutile provare a farne un elenco dettagliato e descriverne il funzionamento.

Per installarlo bisogna:

- Scaricare la libreria direttamente da: http://framework.zend.com/download/current/ (Libreria completa). Nel caso in cui si sia solo interessati alle funzionalità delle librerie GData per interagire con i servizi Google si può scaricare solo la libreria: http://framework.zend.com/download/WebServices (Solo la libreria Gdata);
- Scompattare il file compresso, e copiare la cartella library sotto la cartella specificata nel php.ini in include_path.

File php.ini

```
1. ;;;;;;;;;;;;;;;;;;;;;;;;;
2. ; Paths and Directories ;
3. ;;;;;;;;;;;;;;;;;;;;;;;;;
4. ; UNIX: "/path1:/path2"
5. ;include_path = ".:/php/includes"
6. ;
7. ; Windows: "\path1;\path2"
8. ;include_path = ".;c:\php\includes"
9. include_path = ".;D:\0USER\LAMP\php526\extras\library"
```

Dopo aver riavviato il Web server con phpinfo() verificare la variabile include_path:

1.9 Database

il termine Database, banca dati, base di dati (soprattutto in testi accademici) o anche base dati, indica un archivio strutturato in modo tale da consentire la gestione dei dati stessi (l'inserimento, la ricerca, la cancellazione ed il loro aggiornamento) da parte di applicazioni software. Il Database è un insieme di informazioni, di dati che vengono suddivisi per argomenti in ordine logico (tabelle) e poi tali argomenti vengono suddivisi per categorie (campi).

Informalmente e impropriamente, la parola "Database" viene spesso usata come abbreviazione dell'espressione Database Management System (DBMS), che si riferisce a una vasta categoria di sistemi software che consentono la creazione e la manipolazione efficiente di Database.

PHP ha un buon supporto per i Database, ma ha il difetto di fornire troppe soluzioni alternative per attuare l'interfacciamento con i server di dati. In mezzo a questa quantità veramente enorme di classi e funzioni mi pare una buona scelta optare per una via univoca da perseguire in tutti i propri applicativi. La soluzione migliore è sicuramente quella di affidarsi ad un layer di astrazione

potente e funzionale, che permetta l'interfaccia con sistemi diversi ed architetture differenti; PHP offre un paio di buone soluzioni a questo proposito, che sono PDO ed SDO.

La seconda soluzione, di cui si parlerà in modo più approfondito in un'altra sede, offre anche funzionalità per accedere univocamente ad altre fonti di dato, quale l'XML. La prima invece si focalizza solamente sui Database e, anche se con qualche bug di troppo non ancora risolto, offre una buona serie di funzionalità. La soluzione proposta dallo Zend Framework si basa appunto su questa libreria, che cerca di migliorare con l'aggiunta di interessanti routine di utilità.

Il modulo Zend_Db rappresenta il layer di astrazione basato su PDO utilizzato dal framework per accedere ai Database. L'utilizzo di PDO come libreria nativa permette al layer di astrazione di fornire l'accesso a tutti i sistemi di Database supportati dalla libreria stessa. Tra questi ricordiamo MySQL, PostgreSQL, Microsoft SQL Server, SQLite ed altri. La creazione di un oggetto per interfacciarsi ad un Database è eseguita tramite l'utilizzo del pattern factory:

Connessione ad un DB tipo MySQL

```
1.  <?php
2.  /* Presuppongo che il Framework Zend sia nel vostro include_path */
3.  require_once 'Zend/Db.php';
4.  // Opzioni di connessione
5.  $options = array ('host' => '127.0.0.1',
6.  'username' => 'utente',
7.  'password' => 'password',
8.   'dbname'  => 'test');
9.  $db = Zend_Db::factory('pdoMysql', $options);
10. ?>
```

La funzione statica factory della classe Zend_Db accetta come parametri una stringa rappresentante il driver di connessione che si desidera utilizzare ed un array di opzioni specifiche per il driver e necessarie per connettersi correttamente alla fonte di dati richiesta. Se avessimo voluto connetterci ad un Database SQLite, che non richiede l'autenticazione e l'utilizzo di un server, avremmo utilizzato il codice seguente:

Connessione ad un db tipo SQLite

```
1.  <?php
2.  require_once 'Zend/Db.php';
3.  $options = array ('dbname' => 'test');
4.
5.  $db = Zend_Db::factory('pdoSqlite', $options);
6.  ?>
```

Il risultato di queste operazioni è un'implementazione specifica del driver utilizzato

dell'interfaccia Zend_Db_Adapter_Abstract. Tramite questa interfaccia è possibile accedere alle funzionalità aggiuntive fornite dal Framework e ad alcune delle funzionalità esposte da PDO.

cakePHP cerca i dati di configurazione del Database nel file app/config/database.php. Si Può trovare un esempio di configurazione nel file app/config/database.php.default. Una configurazione corretta assomiglia a questa:

```
1. var $default = array('driver'
2. 'persistent'  => false,
3. 'host'        => 'localhost',
4. 'login'       => 'cakephpuser',
5. 'password'    => 'c4k3roxx!',
6. 'Database'    => 'my_cakephp_project',
7. 'prefix'      => '');
```

È possibile, con cakePHP, usare più di un Database? Questo è un problema assai comune, e ovviamente si può fare. Basta definire nel config/database.php più variabili, ognuna delle quali definisce la connessione al DB nel solito modo. Ad esempio:

```
1. class Database_CONFIG
2. {
3. var $default = array('driver' => 'mysql',
4. 'connect' => 'mysql_connect',
5. 'host' => 'localhost',
6. 'login' => '****',
7. 'password' => '****',
8. 'Database' => 'default_db',
9. 'encoding' => 'utf8',
10. 'prefix' => '');
11.
12. var $alternative = array('driver' => 'adodb',
13. 'connect' => 'mssql',
14. 'host' => 'mssql_server',
15. 'login' => '*****',
16. 'password' => '*****',
17. 'Database' => 'alternative_db',
18. 'prefix' => '');
19. }
```

Per fare in modo che un Model sia riferito a questo secondo DB invece che a quello di default si deve definire:

```
1. class MyModel extends AppModel()
2. {
3. var $useDbConfig = 'alternative';
4.
5. // ...
6. }
```

1.10 ACL

Una lista di controllo degli accessi, spesso chiamata col nome inglese di Access Control List (ACL), è un meccanismo generalmente usato in informatica per esprimere regole complesse. Tra le sue applicazioni principali, la configurazione di Firewall e dei diritti di accesso a file e directory.

Una ACL è una lista ordinata di regole che viene usata per prendere una decisione (es. se far passare o meno un pacchetto o se permettere o meno ad un certo utente l'accesso ad un file).

Ciascuna regola, detta Access Control Entry (ACE), esprime una o più proprietà dell'oggetto da valutare (es. l'indirizzo sorgente di un pacchetto IP), e se queste proprietà sono verificate indica quale decisione prendere (es. far passare il pacchetto oppure rifiutarlo).

La valutazione inizia dalla prima regola e continua fino a quando le condizioni di una regola non sono verificate. Se le condizioni sono verificate, la valutazione finisce e viene applicata la decisione presa. Altrimenti, la valutazione prosegue alla regola successiva. Se nessuna regola viene soddisfatta, viene applicata una decisione di default, chiamata policy dell'ACL.

Le ACL sono utilizzate anche per la determinazione dei permessi riguardanti file e cartelle memorizzati sui dischi. Alcuni dei sistemi operativi che ne fanno uso sono Microsoft Windows, OpenVMS, Linux e Mac OS X. In quest'ultimo sono state introdotte nella versione 10.4 (Tiger) e rese attive per default nella versione 10.5 (Leopard) e sono gestibili via interfaccia grafica con il Finder e da riga di comando tramite chmod.

In cakePHP, ACL viene utilizzato per specificare agli utenti l'accesso a un Controller, o un'azione specifica di un Controller. Prima di arrivare alla modalità di sezione, ci sono alcuni termini che dobbiamo prima capire.

- Access Object Request (ARO) è definito che vogliono un oggetto di accesso o richiesta. In termini semplici, si tratterà di un utente o gruppo. Questo elenco di utente o gruppo è memorizzato nella tabella chiamata 'AROS'.
- Object Access Control (ACO) è oggetto definito, che è protetto da un utente o un gruppo. In altre parole ACO collegato con il Controller o l'azione di Controller che si desidera proteggere. ACO è memorizzato nella tabella chiamata 'acos'.

'acos' e tabella 'AROS' viene creato automaticamente quando si inizializza le tabelle Acl DB (ci arriveremo in questo più avanti in quanto a parte). Insieme a quelli da tavolo, c'è tabella chiamata

'aros_acos' che precisare il legame tra ARO e ACO. In inglese si significa che l'utente ha il permesso di un controllore o azione di controller.

Inizializzare ACL con questo comando.

- cake schema run create DbAcl

Questo comando creerà 'acos' tabella 'AROS' e 'aros_acos'. Otterrete qualcosa di simile:

```
1. Welcome to CakePHP v1.2.1.8004 Console
2. ---------------------------------------------------------------
3. App : app
4. Path: /var/www/labs/acl/app
5. ---------------------------------------------------------------
6. Cake Schema Shell
7. ---------------------------------------------------------------
8.
9. The following table(s) will be dropped.
10. acos
11. aros
12. aros_acos
13.
14. Are you sure you want to drop the table(s)? (y/n)
15. [n] > y
16. Dropping table(s).
17. acos updated.
18. aros updated.
19. aros_acos updated.
20.
21. The following table(s) will be created.
22. acos
23. aros
24. aros_acos
25.
26. Are you sure you want to create the table(s)? (y/n)
27. [y] > y
28. Creating table(s).
```

29. acos updated.
30. aros updated.
31. aros_acos updated.
32. End create.

Zend_Acl fornire le implementazioni per la gestione dei privilegi usando i ruoli e risorse.

- Risorse
  - Si tratta di un oggetto a cui accesso è controllato.
  - Zend fornire modo molto semplice per creare una risorsa. Basta implementare
  - Zend_Acl_Resource_Interface che forniscono getResourceId singolo metodo (). È necessario eseguire l'override di questo metodo nella classe.
  - Puoi aggiungere più risorse per Zend_Acl. Tali risorse verranno aggiunti come struttura ad albero. Treee struttura consente di organizzare le risorse dal generale allo specifico.
  - La risorsa sulla parte superiore della struttura ad albero avrà i privilegi di carattere generale, mentre i nodi dalla struttura ad albero avrà privilegi più specifici.
- Ruoli
  - Il ruolo è un oggetto che richiedere l'accesso a una risorsa.
  - Creazione di ruolo è anche semplice in Zend_Acl. Implementare Zend_Acl_Role_Interface e ignorare lo getRoleId () metodo. Questo è l'unico metodo Zend_Acl_Role fornito.
  - In Zend_Acl, ruolo può essere ereditata costituire uno o più ruoli.
  - Per creare risorse e ruoli, è necessario creare prima istanza come Zend_Acl $acl = new Zend_Acl();

1.12 Cache

Il Web caching è la caching di documenti Web (es. pagine HTML, immagini, etc.) che permette di ridurre l'uso della banda e il tempo di accesso ad un Sito Web. Una Web cache memorizza copie di documenti richiesti dagli utenti, successive le richieste possono essere soddisfatte dalla cache se si presentano certe condizioni. Le Web cache di solito raggiungono picchi d'efficienza nell'ordine del 30%-50%, e migliorano la loro efficienza al crescere del numero di utenti.

HTTP ha un insieme di funzionalità che gli user agent e i server originari possono usare o meno per controllare che i documenti siano memorizzati in una cache e per sapere quando la copia deve essere riutilizzata. Alcuni siti permettono l'utilizzo di una cache, altri no. Le Web cache si differenziano in: lato client e lato server. Le cache lato client, anche chiamate forward cache, vengono utilizzate per servire un gruppo di utenti locale. Sono spesso utilizzate da Internet Service Provider, scuole, e aziende per i loro utenti. Le cache lato server, anche conosciute come

reverse-caches e Web accelerator, sono poste davanti ai server per ridurre il loro carico di lavoro. Inoltre esistono servizi detti Content Delivery Network, costituiti da una rete di server dislocati in punti strategici di internet, che erogano contenuti di siti molto frequentati. Il più famoso è Akamai.

Tutti i maggiori siti che solitamente ricevono milioni di accessi al giorno hanno bisogno di qualche forma di Web caching. Se molteplici server cache sono utilizzati insieme, possono essere coordinati da protocolli quali Internet Cache Protocol e HTCP.

I Web browser moderni includono Web cache interne. Delle tipiche Web cache esterne sono:

- squid cache;
- Il Server Web Apache può essere usato anche come Web cache, utilizzando un apposito modulo;
- Microsoft ISA Server;
- memcached;

Le Web cache possono svolgere anche funzioni come il controllo d'accesso, l'autenticazione degli utenti e il filtraggio dei contenuti.

Alcune persone sono preoccupate che il Web chaching possa essere un atto di violazione del copyright. Nel 1998 la DMCA aggiunse regole all'United States Code (es. 17 Sec. 512) che tutelano ampiamente gli operatori da responsabilità sul copyright per lo scopo del caching.

In Zend Framework c'è Zend Data Cache che è un sistema di cache per PHP (es. simile ad APC). APC (Alternative PHP Cache) è un'estensione nativa per PHP che svolge principalmente il compito di precompilare, ottimizzare e mantenere in memoria il codice intermedio associato agli script PHP in modo che venga bypassato questo passaggio dopo la prima richiesta effettuata ad un file PHP.

cakePHP utilizza un sistema interno di caching degli oggetti per velocizzare le operazioni più comuni. In automatico si occupa del caching di modelli, filepath e traduzioni (es. quelle dei files .po). In aggiunta è possibile configurare il caching per altri tipi di oggetto, tra cui le Views, i CSS e files JS.

1.13 Helper

Un helper contiene metodi a supporto di una View o, più raramente, di un Controller. cakePHP e Zend Framework offrono una ricca dotazione di helper per creare form, tag e link html,

È possibile utilizzare aiutanti in cakePHP realizzando un Controller che si occupa di esso. Ogni controllore ha una proprietà $helpers aiutanti che elenca gli helpers che devono essere messi a disposizione della View. Per abilitare un helper nella View in esame, si deve aggiungere il nome dell'helper nell'array $helpers del Controller.

```
1. <?php
2. class BakeriesController extends AppController {
3. var $helpers = array('Form', 'Html', 'Javascript', 'Time');
4. }
```

```
5. ?>
```

Si può anche aggiungere helpers in un'azione, così che saranno disponibili per quella azione e non nelle altre azioni del Controller. Questo salva risorse per le azioni che non usano quel Controller e aiuta inoltre a tenere i Controller meglio organizzati.

```
1. <?php
2. class BakeriesController extends AppController {
3. function bake {
4. $this->helpers[] = 'Time';
5. }
6. function mix {
7. // The Time helper is not loaded here and thus not available
8. }
9. }
```

Per quanto concerne Zend Framework si consideri una prima cosetta semplice e utile: gli helper, ovvero dei piccoli pezzetti di codice che possono essere richiamati dalla View più volte e che possono mantenere un loro stato.

Ricetta semplice per creare l'helper “showErrors” che in pratica incrementa una variabile e restituisce una stringa: Creare il file showErrors.php nella directory /application/views/helpers/ con il seguente contenuto

```
1. <?php
2. class Zend_View_Helper_ShowErrors
3. {
4. protected $_count = 0;
5. public function showErrors() {
6. $this->_count++;
7. $output = "I have seen 'The Jerk' {$this->_count} time(s).";
8. return htmlspecialchars($output);
9. }
10. }
```

Richiamare l'helper dalla View nella View basta inserire

```
1. <?php echo $this->showErrors(); ?>
```

Convenzioni da seguire:

- Il file che contiene la classe dell'helper si deve chiamare come l'helper con la lettera iniziale minuscola (es. showErrors.php);

- La classe dell'helper deve iniziare con Zend_View_Helper_ seguita dal nome dell'helper con la lettera iniziale maiuscola (es. class Zend_View_Helper_ShowErrors);
- Deve esistere nella classe almeno un metodo che si chiama come l'helper con la lettera minuscola iniziale (es. public function showErrors());
- Se l'helper è salvato nella directory views/helpers/ viene caricato il tutto in modo automatico.

Visto che helper deve mostrare i messaggi di errore, si deve accedere alla variabile $messages che si trova nella View. Normalmente nella View si accede alla variabile semplicemente con $this->messages. Nell'helper si deve fare un passo in più: recuperare l'oggetto View: Ricetta per gli impazienti: Si Aggiungere un metodo nell'helper che si chiama setView che riceve l'oggetto View e lo salva in una variabile e poi è possibile utilizzarlo in tutti i metodi.

```
1.  <?php
2.  class Zend_View_Helper_ShowErrors
3.  {
4.  protected $_count = 0;
5.  public $View;
6.  public function setView(Zend_View_Interface $View) {
7.  $this->View = $View;
8.  }
9.  public function showErrors() {
10. if (isset($this->View->messages)) {
11. $this->_count++;
12. $output = "I have seen 'The Jerk' {$this->_count} time(s).";
13. return htmlspecialchars($output);
14. }
15. }
16. }
```

In questo modo si può accedere ad ogni elemento della View semplicemente con $this->View (es. alle variabili impostate dal Controller come in questo caso)

1.14 Scaffolding

Il termine scaffolding, introdotto in psicologia da Jerome Bruner e altri nel 1976, significa letteralmente "impalcatura". Indica quelle strategie di sostegno e quella guida ai processi di apprendimento che consentono di svolgere un compito sebbene non si abbiano ancora le competenze per farlo in autonomia, riuscendovi grazie all'aiuto di un esperto, di un adulto o di un

pari più preparato che fornisce indicazioni e suggerimenti, nell'attesa che si riesca a maturare una piena autonomia nello svolgimento del compito.

Le componenti generali dello scaffolding sono:

- Reclutare il bambino al compito;
- Mantenere la direzione dell'attività verso il problema da risolvere;
- Semplificare le componenti del compito;
- Mostrare le possibili soluzioni;
- Ridurre i gradi di libertà della situazione.

Le caratteristiche principali dello scaffolding sono da rintracciarsi innanzitutto nell'adeguamento continuo di tale supporto alla zona di sviluppo prossimale (teorizzata da Lev Vygotskij) che il bambino mostra di possedere in un certo momento dello sviluppo, e nella sua progressiva riduzione, fino alla sua scomparsa, quando il bambino è in grado di mettere in atto autonomanente l'abilità o la conoscenza appresa Spesso le tabelle dei Database cambiano e di conseguenza il programmatore deve occuparsi di modificare gli oggetti che interagiscono con queste ultime ed i form che si occupano di inserire i dati all'interno delle tabelle. Per ovviare a questo inconveniente, e rendere più semplice e produttivo il processo di sviluppo, è stato introdotto lo scaffolding che analizza il Database generando i Model e le View necessarie per lavorare con queste ultime. Anche dopo che i dati sono stati modificati lo scaffolding permetti di aggiornare il codice autogenerato aumentando notevolmente la produttività ed eliminando quelle operazioni ripetitive che tendono a rendere tediose le operazioni di modifica successive agli aggiornamenti alla struttura del Database.

calePHP, a differenza di Zend Framework, è dotato di questa caratteristica che permette di avere un prototipo di Web Application perfettamente funzionante senza doversi preoccupare di impazzire con HTML o xHTML, lasciando quindi la fase strutturale per ultima e potendosi concentrare sulla logica della Web Application.

1.15 Skeleton

Seconda la guida di ProE uno skeleton è: "A skeleton part is a special part Model created in the context of an assembly to develop design criteria without having to create components and assemble them together. The skeleton part is a 3-D layout of an assembly that is used as the framework to build the assembly”.

Bakeme, in cakePHP, è una applicazione skeleton con gli script personalizzabili, che permette di creare applicazioni avanzate quali siti web data-driven.

SkeletonBased, in Zend Framework, è sulle migliori consigli, Gli obiettivi sono fornire una semplice e pronta all'uso struttura delle directory, per iniziare ad imparare le basi con le opzioni di scalabilità e prendersi cura delle migliori pratiche per avviare le applicazioni in via di sviluppo.

1.16 Plug-in

Il plugin (o plug-in, o addin o add-in o addon o add-on), in campo informatico è un programma

non autonomo che interagisce con un altro programma per ampliarne le funzioni (es. un plugin per un software di grafica permette l'utilizzo di nuove funzioni non presenti nel software principale.

La capacità di un software di supportare i plug-in è generalmente un'ottima caratteristica, perché rende possibile l'ampliamento e la personalizzazione delle sue funzioni in maniera semplice e veloce.

In alcuni casi, il plug-in viene anche denominato estensione o extension (es. l'architettura a estensioni di Mozilla Firefox o di OpenOffice).

Entrambi i Framework supportano queste possibilità di usufruire sempre di nuovi con l'unica variante che Zend Framework offre uno stimolo ii più alla realizzazione da parte degli utenti stessi del Framework.

1.17 CLI

Call Level Interface è uno standard ideato da The Open Group alla base dell'ODB che rappresenta una ottima soluzione per rendere la programmazione indipendente dal DBMS.

Le sue Caratteristiche principali sono

- La comunicazione Client/Server basata su una API standardizzata (e non proprietaria);
- Il codice SQL non immerso nel codice sorgente, ma inviato al DBMS sotto forma di stringhe;
- I Client che dispongono di un opportuno modulo, detto “driver” che implementa l’API e gestisce la comunicazione con il DBMS.

È uno standard come gli altri con i suoi "pro" ed i suoi "contro". I principali vantaggi e svantaggi sono elencati di seguito.

I vantaggi rispetto a SQL immerso sono:

- Indipendenza della Web Application rispetto al DBMS (API standard);
- Possibilità di accedere a più di una sorgente di dati nella stessa Web Application.

Gli svantaggi rispetto ad SQL immerso sono:

- Essere meno efficiente se il driver non è ottimizzato;
- Architettura più complessa (driver manager).

CLI rappresenta il Bake (il cuore) di cakePHP mentre Zend Framework non ne possiede uno.

1.18 Web Service

Secondo la definizione data dal World Wide Web Consortium (W3C) un Web Service (servizio Web) è un sistema software progettato per supportare l'interoperabilità tra diversi elaboratori su di una medesima rete; caratteristica fondamentale di un Web Service è quella di offrire un'interfaccia software (descritta in un formato automaticamente elaborabile quale, ad esempio, il Web Services Description Language) utilizzando la quale altri sistemi possono interagire con il

Web Service stesso attivando le operazioni descritte nell'interfaccia tramite appositi "messaggi" inclusi in una "busta" (es. la più famosa è SOAP): tali messaggi sono, solitamente, trasportati tramite il protocollo HTTP e formattati secondo lo standard XML.

Proprio grazie all'utilizzo di standard basati su XML, tramite un'architettura basata sui Web Service (chiamata, con terminologia inglese, Service oriented Architecture - SOA) applicazioni software scritte in diversi linguaggi di programmazione e implementate su diverse piattaforme hardware possono quindi essere utilizzate, tramite le interfacce che queste "espongono" pubblicamente e mediante l'utilizzo delle funzioni che sono in grado di effettuare (es. i "servizi" che mettono a disposizione) per lo scambio di informazioni e l'effettuazione di operazioni complesse (quali, ad esempio, la realizzazione di processi di business che coinvolgono più aree di una medesima azienda) sia su reti aziendali come anche su Internet: la possibilità dell'interoperabilità fra diversi software (es. tra Java e Python) e diverse piattaforme hardware (es. Windows e Linux) è resa possibile dall'uso di standard "aperti" (per un concetto assimilabile cfr. la voce "open source").

Il consorzio OASIS (Organization for the Advancement of Structured Information Standards) ed il W3C sono i principali responsabili dell'architettura e della standardizzazione dei Web Service; per migliorare l'interoperabilità tra le diverse implementazioni dei Web Service l'organizzazione WS-I sta inoltre sviluppando una serie di "profili" per meglio definire gli standard coinvolti. I comportamento Web Service consente di effettuare facilmente le richieste di servizi web e gli indirizzi a distanza, tramite GET e POST XMLRPC.

Mentre la base di cakePHP rende fornendo Web Services facile. Collegamento a Web Services è ancora un compito. La prima volta a costruire il mio sito personale ho cercato una soluzione già costruito e trovato una soluzione parziale. Tuttavia, ho voluto rimuovere la sua dipendenza da cURL come l'estensione non è sempre disponibile. Volevo anche trasformarlo in un comportamento, che la rende facile da riutilizzare e più conformi agli standard cakePHP. La trasformazione in un comportamento è stato un processo facile. Rimozione curl e scrivendo manualmente tutto il codice Socket non era una prospettiva allettante. Fortunatamente, cakePHP 1.2 è la nuova classe CakeSocket che facilita la creazione di connessioni socket. Il mio risultato finale è un comportamento che non ha estensione e dipendenze è un comportamento per facilitarne il riutilizzo. Questa è una classe PHP5, quindi se siete in PHP4 che dovete incidere tutte le parole chiave visibilità. Utilizzando il comportamento WebService è abbastanza facile. Basta aggiungere al actsAs array $ per il modello

Zend Framework prevede:

- Client per Web Service – Set di Client pronti all'uso per interagire con i principali Web Service quali Yahoo, Amazon, Thecnorati e Gdata;
- Componenti per fruttare i Web Service – Librerie e strumenti per interagire con dei Web Service esistenti;
- Componenti per sviluppare i Web Service – Librerie e strumenti per sviluppare Web Service;
- Componenti al supporto – Librerie aggiuntive utilizzabili a supporto per lo scambio e la validazione dei dati, come URI e Data,

1.19 Feed

Una Il feed Web è un'unità di informazioni formattata secondo specifiche (es. di genesi XML) stabilite precedentemente. Ciò per rendere interoperabile ed interscambiabile il contenuto fra le diverse applicazioni o piattaforme. Un feed è usato per fornire agli utilizzatori una serie di contenuti aggiornati di frequente. I distributori del contenuto rendono disponibile il feed e consentono agli utenti di iscriversi. L'aggregazione consiste in un insieme di feeds accessibili simultaneamente, ed è eseguita da un aggregatore Internet. L'uso principale dei feed RSS (detti anche flussi RSS) attualmente è legato alla possibilità di creare informazioni di qualunque tipo che un utente potrà vedere molto comodamente, con l'aiuto di un lettore apposito, nella stessa pagina, nella stessa finestra, senza dover andare ogni volta nel sito principale. Questo è dovuto al fatto che il formato XML è un formato dinamico.

Il Web feed presenta alcuni vantaggi, se paragonato al ricevere contenuti postati frequentemente tramite E-Mail:

- Nell'iscrizione ad un feed, gli utenti non rivelano il loro indirizzo di posta elettronica. In questo modo non si espongono alle minacce tipiche dell'E-Mail: lo spam, i virus, il phishing ed il furto di identità.
- Se gli utenti vogliono interrompere la ricezione di notizie, non devono inviare richieste del tipo "annulla la sottoscrizione". Basta che rimuovano il feed dal loro aggregatore.

cakePHP è dotato di un Plug-In mentre Zend Framework si serve di una libreria chiamata zend_feed che consente agli sviluppatori di recuperare i feed con facilità. Se si conosce l'URI di un feed è sufficiente utilizzare il metodo Zend_Feed::import().

1.20 Form

Un form (letteralmente "modulo") è un termine usato per indicare l'interfaccia di una Web Application che consente all'utente di inviare uno o più dati liberamente inseriti dallo stesso; per descriverlo, può essere utile la metafora della "scheda da compilare" per l'inserimento di dati. Nella maggior parte dei casi il termine è riferito a form contenute in una pagina Web: ad esempio le caselle di testo e i menu a tendina di una pagina di registrazione costituiscono un form.

Più in particolare, l'elemento HTML <FORM> è usato per realizzare form in una pagina Web. In html si intendono per parte del form anche i pulsanti ripristina o cancella e invia. Di solito i form vengono utilizzati per inviare dati ad un Database oppure per inviare E-Mail. Il termine è usato, in senso più esteso, nell'ambiente di sviluppo Visual Basic per indicare una finestra principale della Web Application, in cui possono essere inseriti elementi visuali come pulsanti o caselle di testo.

Zend_Form semplifica la creazione e la gestione in forma l'applicazione web. Esso svolge i seguenti compiti:

- Element input filtering e validation;
- Element ordering;
- Element e Form rendering, including escaping;
- Element e form grouping;

- Element e form-level configuration.

Il FormHelper è una nuova aggiunta a cakePHP. La maggior parte del lavoro nella creazione di form è ora fatto usando questa nuova classe, piuttosto che (ora deprecati) i metodi nel HtmlHelper. Il FormHelper si concentra sulla creazione di moduli in modo rapido, in modo tale da snellire la convalida, ripopolamento e il layout. Il FormHelper è anche flessibile.

2. Migrazione da cakePHP a Zend Framework

2.1 Servizi di Migrazione

2.1.1 Panoramica

Ogni fase di migrazione presenta rischi e costi, soprattutto se manca una sufficiente conoscenza delle tecnologie di supporto sottostanti. Affidarsi ad una buona progettualità per gestire il piano di migrazione da cakePHP a Zend Framework (da PHP 4 a PHP 5) può ridurre i costi e limitare i rischi. In passato, molti sviluppatori hanno scelto cakePHP per la realizzazione di Web Application, sulla base delle notevoli funzionalità. Ad oggi, in virtù dell'introduzione del Framework Zend Framework, ogni team di sviluppo deve prendere in considerazione la possibilità di migrare verso questa nuova tecnologia, in riferimento ad una Web Application realizzata con cakePHP. La migrazione è consigliata sopratutto nel caso in cui, la Web Application debba essere ulteriormente potenziata, nel qual caso si pùò fare uso degli strumenti innovativi di Zend Framework. Nel caso in cui, debba essere previsto esclusivamente un processo di manutenzione, si può pensare di mantenere l'architettura attuale realizzata in cakePHP.

2.1.2 Strategie

Una **strategia** è la descrizione di un piano d'azione di lungo termine usato per impostare e successivamente coordinare le azioni tese a raggiungere uno scopo predeterminato. La strategia si applica a tutti i campi in cui per raggiungere l'obiettivo sono necessarie una serie di operazioni separate, la cui scelta non è unica e/o il cui esito è incerto. La parola strategia deriva dal termine greco στρατηγός (*strateghós*), ossia "generale".

L'architettura fortemente espandibile di Zend Framework metta a disposizione diversi approcci per poter essere adottata in Web Application presistenti. L'approccio maggiormente conservativo prevede semplicemente di utilizzare i componenti di Zend Framework, senza alcun altra caratteristica del Framework. L'altra possibilità è quella di eseguire una migrazione incrementale oppure completa.

2.2 I vantaggi del passaggio a Zend Framework

2.2.1 Panoramica

Portare una Web Application da cakePHP a Zend Framework presenta molti vantaggi. Nello

specifico, Zend Framework offre un'ampia varietà di nuove caratteristiche che amplieranno le possibilità della Web Application, faciliteranno lo sviluppo, miglioreranno l'esperienza degli utenti e consentiranno di ottenere un risparmio. Poiché la community PHP ha annunciato che interromperà il supporto di PHP 4 (che è alla base di cakePHP) dopo l'agosto 2008, è importante impostare il piano di transizione ora, in base alle proprie condizioni.

2.2.2 Migliori performance

PHP 5 è più veloce di PHP 4. Il miglioramento nelle performance dipende da diversi fattori, ma nella maggior parte dei casi si ottiene un miglioramento di 2-3 volte semplicemente assicurandosi che la Web Application sia compatibile con PHP 5.

2.2.3 Manutenzione più semplice

Il nuovo object Model e le sue funzionalità di supporto semplificano la manutenzione delle applicazioni Web facilitando la Web Application di un'architettura software che consente di leggere e scrivere codice di qualità più elevata.

2.2.4 Integrazione Web 2.0

I **Web 2.0** è un termine utilizzato per indicare uno stato dell' evoluzione di Internet (e in particolare del World Wide Web), rispetto alla condizione precedente. Si tende a indicare come Web 2.0 l'insieme di tutte quelle applicazioni online che permettono uno spiccato livello di interazione tra il sito e l'utente (es. blog, forum, chat, sistemi quali Wikipedia, Youtube, Facebook, Myspace, Twitter, Gmail, Wordpress, Trip advisor ecc.).

Per le applicazioni Web 2.0, spesso vengono usate tecnologie di programmazione particolari, come AJAX (Gmail usa largamente questa tecnica) o Adobe Flex.

Le applicazioni tendono a non essere più indipendenti, dentro e fuori l'impresa. È sempre più importanti che possano interagire, condividere dati, integrarsi e combinarsi in mash-up, una "miscela" che offre agli utenti un'esperienza più ricca. PHP 5 fornisce le funzionalità che consentono di sfruttare le più recenti opzioni di integrazione.

2.3 Un ausilio nella transizione

2.3.1 Panoramica

La migrazione di una Web Application a una nuova versione può essere un processo pesante. Avere una visione ben chiara e in anticipo di tutto ciò che sarà richiesto consente di gestire tale fase in modo più rapido e con meno problemi, risparmiando tempo e denaro. Pianifica il passaggio da cakePHP a Zend Framework definendone tu stesso le condizioni.

I servizi di migrazione di Zend Framework possono aiutarti ad avviare la transizione, a prescindere dalla necessità o meno di rivolgersi a esperti PHP per analizzare la Web Application al fine di individuare incompatibilità con PHP 5. Con i servizi di migrazione di Zend Framework, la

transizione a PHP 5 sarà più facile in ogni caso.

2.3.2 Meno rischi, meno costi

Ogni processo di migrazione presenta rischi e costi, soprattutto per chi ha competenze in campi diversi da quello delle tecnologie della Web Application interessata. Sfruttando i servizi di migrazione di Zend Framework per identificare, pianificare e implementare una migrazione efficace delle applicazioni PHP 4 a PHP 5, si possono ridurre contemporaneamente i costi e i rischi associati. Così potrai mantenere le tue risorse concentrate sul fornire valore ai tuoi clienti, e ti assicurerai che la migrazione sia effettuata correttamente al primo colpo:

- Predisponi un piano di transizione che soddisfi le esigenze della tua attività;
- Mantieni le tue risorse focalizzate sull'attività quotidiana;
- Assicurati che la migrazione sia realizzata correttamente al primo colpo.

I servizi di migrazione di Zend Framework sono una serie di servizi separati pensati per aiutarti a identificare, pianificare e implementare la migrazione delle tue applicazioni da PHP 4 a PHP 5.

3. Conclusioni

Zend Framework, sviluppato da Zend Technologies è un open-source, orientato agli oggetti, realizzato in PHP 5. E 'noto come ZF ed è stato sviluppato con lo scopo di rendere le cose più facile per gli sviluppatori PHP.

cakePHP è un open-source orientato agli oggetti ed è utilizzato per la creazione di applicazioni Web. Scritto in PHP, è sviluppato da Cake Software Foundation, Inc e si basa sul Modello di Ruby on Rails.

Zend Framework è dotato di facile metodi di licenza grazie alla nuova licenza BSD e di una rapida e ben testato il codice di base che l'azienda può contare su. Si fa uso di API disponibili comunemente noto da fornitori come Google, Amazon, Yahoo!, Flickr e fornitori di API e catalogatori come StrikeIron Web e programmabili.

cakePHP è un quadro di sviluppo rapido per PHP che offre una architettura flessibile per lo sviluppo, la manutenzione e la realizzazione di applicazioni. Si utilizza il solito Design Pattern MVC e come ORMA entro la convenzione oltre configurazione paradigma che aiuta inferiore spese di sviluppo e non è necessario agli sviluppatori per scrivere un sacco di codice. CakePHP vanta clienti come Mozilla Addons, Gratta e Vinci da MIT, Yale Daily News, Cipolla e MapMe Store per citarne alcuni.

ZF offre una semplice libreria di componenti di fornire il 80 per cento delle funzionalità necessarie per la maggior parte degli sviluppatori. Il resto del 20 per cento può essere modificata secondo le vostre esigenze di business. Questo può essere fatto dal momento che ha un architettura flessibile e non è necessario alcun file di configurazione per iniziare a utilizzare. A causa di ciò, i costi di formazione sono diminuiti e il prodotto raggiunge il mercato prima di tale data. Inoltre, Zend Framework offre inoltre:

- Sostegno attraverso JSON AJAX, che offre facilità d'uso del quadro;

- La versione di PHP del motore di ricerca Lucene;
- Semplice accesso ai dati necessari in formati di applicazioni Web 2.0;
- La destinazione ideale per utilizzare e pubblicare i servizi Web;
- Di alta qualità, orientata agli oggetti PHP 5 libreria di classi.

Proprio come Zend Framework, cakePHP non richiede configurazione. È molto semplice da utilizzare. L'azienda dispone di un utente amichevole comunità chiamato #cakephp su IRC che aiuta i nuovi utenti di iniziare. E 'distribuito sotto la licenza MIT e promette migliori pratiche, come la sicurezza, la sessione di autenticazione e di gestione. Ha un approccio object-oriented per tenervi a proprio agio.

cakePHP e Zend Framework sono i quadri di sviluppo rapido per PHP che fornisce una architettura estensibile per lo sviluppo, il mantenimento, e distribuzione di applicazioni. Uso comunemente noto come Design Pattern MVC e ORM in seno alla Convenzione oltre configurazione paradigma, sia di ridurre i costi di sviluppo e aiuta gli sviluppatori scrivere meno codice.

cakePHP rende più facile per l'utente di interfaccia con il Database con i record attivi. Essa incoraggia inoltre l'uso del Modello-View-Controller Modello architettonico.

- Compatibile con PHP4 e PHP5;
- Aiuta gli sviluppatori professionisti o per ridurre il codice pentimento nella creazione del sito Web;
- Fornisce potenti funzioni di convalida dei dati flessibili;
- Sito Web directory indipendente:
- Liste di controllo degli accessi (ACL);
- Dati sanificazione;
- Sicurezza, sessione, e richiesta di movimentazione componenti;

Caratteristiche di Zend Framework :

- Tutti i componenti sono completamente object-oriented di PHP 5 e sono conformi E_STRICT ;
- Usa-at-architettura si scioltamente accoppiato con i componenti e le interdipendenze minimo;
- Sostenere l'attuazione Extensible MVC layout e PHP basato su template di default;
- Supporto per più sistemi di Database e venditori, tra cui MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite, e Informix Dynamic Server ;
- E-Mail composizione e la consegna, il recupero attraverso mbox, Maildir, POP3 e IMAP4 ;
- Flessibile caching sub-sistema con il supporto per diversi tipi di backend, come la memoria o un file system.

Nel complesso, questi due quadri sono buoni da utilizzare per le vostre esigenze. cakePHP permette uno sviluppo più rapido delle applicazioni, riducendo le righe di codice necessarie alla

scrittura degli applicativi. Si riducono i costi di sviluppo e, grazie alla sua un architettura estendibile, si semplificano la scrittura e la manutenzione del codice. ZF fa della semplicità nello sviluppo della programmazione orientata agli oggetti il suo punto di forza. Permette di realizzare applicazioni più sicure, efficienti e in pieno stile Web 2.0. Entrambe le strutture sono in via di sviluppo.

Capitolo V – Case Study : Analisi e Progettazione

Ciò che dobbiamo imparare lo impariamo facendo
(Aristotele)

1. Introduzione

La presentazione dei due principali Framework, quali Zend Framework e cakePHP, utilizzati per la realizzazione della Web Application ha come obbiettivo principale la possibilità di poter effettuare un confronto. Una volta evidenziate le potenzialità dell'uno e dell'altro, si rende necessario lo sviluppo di una Web Application con ciascuno di essi, in modo da individuare pregi e difetti di entrambi e vantaggi e svantaggi dell'uno rispetto all'altro. Tale confronto è relativo a tutti gli aspetti che riguardano la realizzazione di una Web Application, con strumenti che si basano sul pattern MVC, quali appunto i Framework presi in considerazione.

In questa sede, è stata realizzata una Web Application che permette agli utilizzatori di usufruire di un gioco, un cameriere che corre lungo un corridoio di un mezzo di trasporto con in mano un vassoio, mettendo a disposizione degli User, dei Blogger e degli Admin del sistema una serie di funzionalità definite dalle specifiche assegnate.

La Web Application è stata realizzata con l'utilizzo di entrambi i Framework, usufruendo, li dove possibile, di tutte le potenzialità di ciascuno di essi.

Le fasi di sviluppo hanno seguito il tipico Ciclo di Vita di un Software secondo il “modello a cascata”, *Waterfall Model*, e possono essere riassunte di seguito:

- Studio di fattibilità;
- Analisi e specifiche dei requisiti;
- Progettazione;
- Codifica ed implementazione;
- Testing;
- Messa in esecuzione;
- Manutenzione.

Lo studio di fattibilità, in questo caso, è stato immediatamente superato, in quanto non c'erano da fare considerazioni riguardo le possibili soluzioni da adottare, le risorse finanziare ed umane a disposizione,, tenendo conto che sono stati fissati a priori gli strumenti da utilizzare.

L'analisi e la specifica dei requisiti ha previsto la realizzazione del Documento di Specifica dei Requisiti Software (SRS – Software Requierements Specification), elencando tutte le funzionalità fornite dal sistema software sia all'utente che all'amministratore e considerando, per ciascuno di essi, una breve descrizione, gli input previsti, le elaborazioni eseguite ed i possibili output

prodotti.

Nella fase di progettazione, sono stati realizzati i principali diagrammi UML (Unifed Modelling Language) previsti per lo sviluppo, quali:

- Use Case Diagram;
- Class Diagram;
- Sequence Diagram;
- Activity Diagram;
- Statechart Diagram.

Ad essi, vanno poi aggiunti il Conceptual Data Model ed il Physical Data Model, per la descrizione concettuale e fisica della Base di Dati utilizzata dalla Web Application.

Lo sviluppo ha poi previsto due distinte fasi di codifica, realizzando la Web Application con entrambi i Framework. Facendo riferimento alla struttura del Model prevista dal Patter MVC, il sottolivello Data Access è stato realizzato separatamente dagli altri e le classi che ne sono scaturite, sono state utilizzate per entrambe le implementazioni. Per quanto concerne i sottolivelli superiori, Business-Logic e External Interface, si è preferita una modalità di “fusione”, più legata a Zend Framework che non a cakePHP.

La fase di testing ha permesso la correzioni di errori e problemi rilevati durante il funzionamento del software.

Infine, sono state previste delle vere e proprie fasi di messa in esercizio e manutenzione attraverso le varie ferie a cui il software era destinato.

Lo strumento software di tipo CASE, che è stato utilizzato per la realizzazione di tutti i diagrammi della fase di progettazione, è il noto ArgoUML.

2. Requisiti

Il case study (caso di studio) preso in esame, prevede lo sviluppo di una Web Application che permette all'utenza in generale, di usufruire di un gioco, quale obbiettivo finale è tenere in piedi un cameriere per enne metri lungo un corridoio al più a lungo possibile.

Gli utilizzatori possono avere permessi di accesso diversi e quindi funzionalità differenti, sulla base delle distinzioni dei seguenti ruoli:

- User;
- Blogger;
- Admin.

Per ciascuno di essi, le funzionalità accessibili nella Web Application sono descritte di seguito.

- User:
  - Accesso all'archivio punteggio, con la possibilità di visionare le informazioni, per potersi migliorare nella posizione;

- Accesso al gioco, con la possibilità di giocare più volte;
- Operazione di login e logout.

- Blogger:
  - Registrazione “base degli User” per giocare e gestione degli utenti registrati, con la possibilità di visionare i dati ed eseguire le operazioni di cancellazione e sblocco, qualora un utente voglia rigiocare;
  - Operazione di login e logout.
- Admin:
  - Gestione completa dell'archivio punteggio, con la possibilità di inserire, cancellare e modificare i punteggi, usufruendo della funzionalità automatica del sistema di garantire coerenza tra i tag SCORE degli stessi e le informazioni immesse nella Base di Dati. Possibilità di visionare i dati degli User ed eseguire le operazioni di recupero Login e Password, qualora un utente che voglia rigiocare li abbia dimenticati;
  - Operazione di login e logout.

3. Progettazione

La fase di progettazione ha previsto la realizzazione dei principali diagrammi UML, in relazione sopratutto all'analisi del software e non tanto alla sua corrispondente implementazione.

Essi costituiscono un'astrazione di alto livello del sistema e sono completamente indipendenti da quelli che possono essere gli strumenti utilizzati nella codifica.

3.1 Use Case Diagram

I diagrammi dei casi d'uso (*Use Case Diagram*) costituiscono uno strumento utile per catturare il comportamento esterno del sistema da sviluppare, senza dover specificare come tale comportamento debba essere realizzato: Il sistema è visto come una scatola nera (*black-box*).

Essi forniscono una descrizione dei “modi” in cui il sistema potrà essere utilizzato, ossia ciò che l'utente può fare su di esso e come quest'ultimo risponde alle sollecitazioni. La realizzazione dei diagrammi ha previsto un approccio *top-down*, partendo dallo scenario di carattere generale, in cui le modalità d'uso logicamente correlate sono accorpate, fino ad una decomposizione in cui sono descritti i casi di utilizzo non ulteriormente scomponibili.

Un *caso d'uso* è rappresentato graficamente come un'ellisse contenente il nome del caso d'uso. Formalmente, lo *Use Case* è un classifier dotato di comportamento: lo si potrebbe intendere come una classe di comportamenti correlati. Praticamente, uno *use case* rappresenta una funzione o servizio offerto dal sistema a uno o più attori. La funzione deve essere *completa* e *significativa* dal punto di vista degli attori che vi partecipano.

L'associazione fondamentale negli *Use Case Diagram* è quella che congiunge gli attori con i casi d'uso a cui essi *partecipano*. Un attore può essere associato a un qualsiasi numero di casi

d'uso, e viceversa. Pur non richiedendo ulteriori informazioni, l'associazione fra gli use case e gli actor si considera implicitamente dotata di una semantica più specifica rispetto all'associazione generica UML; essa infatti implica uno scambio messaggi fra attori e use case associati.

Il sistema nel suo complesso è rappresentato come un rettangolo vuoto. Questo simbolo viene messo in relazione con gli altri nel senso che i *model element* che rappresentano caratteristiche del sistema verranno posizionati *all'interno* del rettangolo, mentre quelli che rappresentano entità esterne (appartenenti al dominio o al contesto del sistema) sono posizionati *all'esterno*. In molti diagrammi UML il simbolo per il sistema viene omesso in quanto la distinzione fra concetti relativi al sistema e concetti relativi al suo contesto si può in genere considerare implicita.

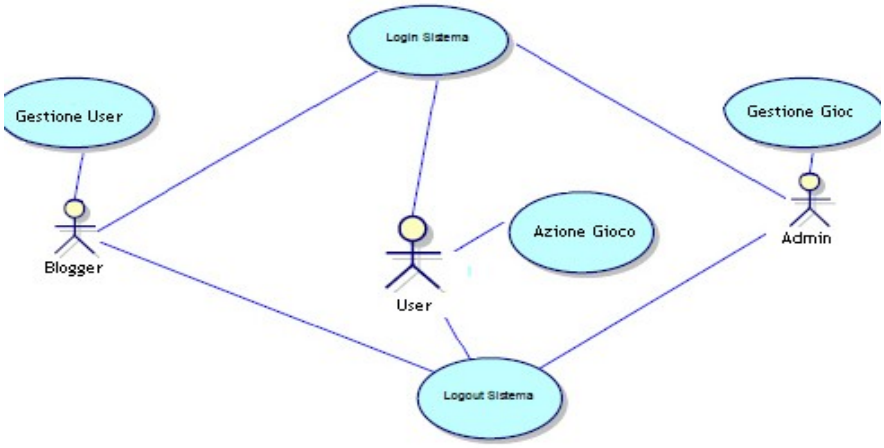
3.1.1 Generale

Gli *attori* sono rappresentati graficamente nel diagramma da un'icona che rappresenta un uomo stilizzato (*stickman*). Formalmente, un attore è una classe con stereotipo «actor». Praticamente, un attore rappresenta un *ruolo* coperto da un certo insieme di entità interagenti col sistema (inclusi utenti umani, altri sistemi software, dispositivi hardware e così via). Un ruolo corrisponde a una certa famiglia di interazioni correlate che l'attore intraprende col sistema.

Facendo riferimento ai requisiti assegnati, è stato definito un *Use Case Diagram Generale*, che si pone al livello di astrazione più elevato del sistema, mettendo in evidenza tutte quelle che sono le modalità di utilizzo dello stesso, da parte dell'User, del Blogger e dell'Admin.

Questi elementi rappresentano gli attori che possono accedere alle funzionalità offerte dal sistema, di cui alcune sono condivise ossia accessibili da entrambe, mentre altre prerogativa esclusiva di uno dei tre ruoli.

Figura 24 - Use Case Diagram Generale



Fonte: Attanasio Ciro tramite ArgoUML

A partire da questo diagramma, è possibile effettuare una decomposizione scendendo ad un livello di astrazione minore, in cui si prendono in considerazione nel dettaglio le modalità di

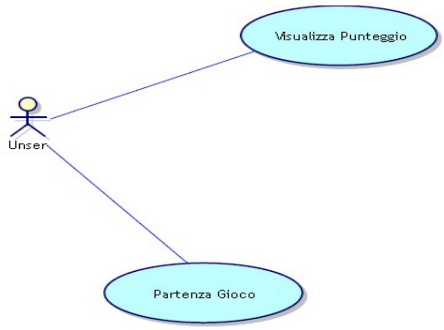
utilizzo del sistema.

3.1.1.1 User

Le modalità accessibili esclusivamente all'User sono le seguenti, derivanti da “Azione Gioco, sono:

- Visualizza Punteggio – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dall'User per accedere e usufruire del servizio della Web Application;
- Partenza Gioco - La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dall'User per accedere e usufruire del servizio della Web Application.

Figura 25 - Use Case Diagram Azioni sul Gioco



Fonte: Attanasio Ciro tramite ArgoUML

3.1.1.2 Blogger

Considerando nuovamente il digramma dei Casi D'Uso Generale, si evince che c'è la modalità “Gestione User” che è accessibile solo ed esclusivamente dal Blogger.

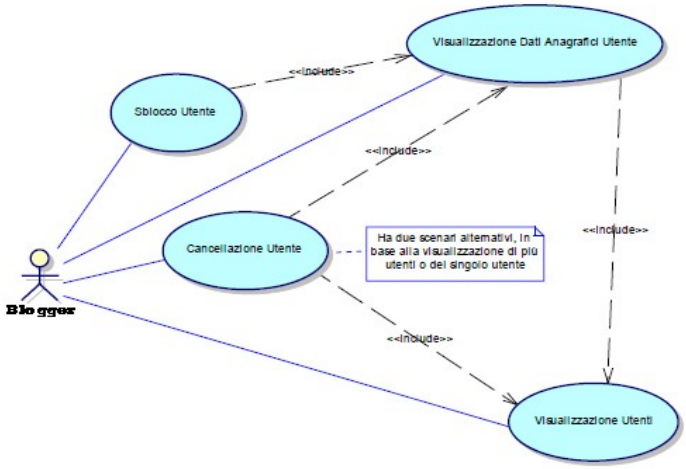
Essa comprende alcune azioni basilari che posso essere eseguite sugli User, in particolare:

- Visualizzazione User – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dal Blogger per accedere e usufruire del servizio della Web Application;
- Cancellazione User – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dal Blogger per accedere e usufruire del servizio della Web Application;
- Visualizzazione Dati Anagrafici User – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dal Blogger per accedere e usufruire del servizio della Web Application;
- Registrazione User – La modalità può essere considerata atomica e non ulteriormente

decomponibile, facendo riferimento alla procedura eseguita dal Blogger per accedere e usufruire del servizio della Web Application;

- Sblocco User, qualora abbia commesso due tentativi errati consecutivi di accesso al sistema e sia stato bloccato – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dal Blogger per accedere e usufruire del servizio della Web Application;

Figura 26 - Use Case Diagram Gestione User



Fonte: Attanasio Ciro tramite ArgoUML

3.1.1.3 Admin

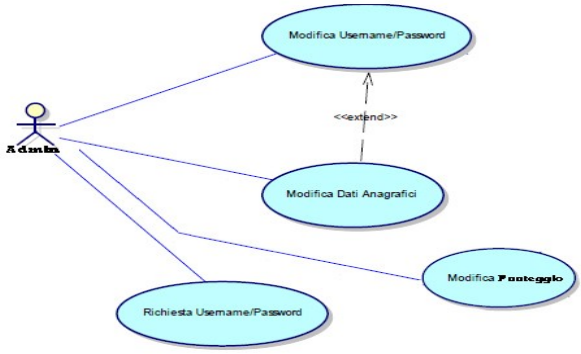
In fine, le modalità accessibili esclusivamente all'Admin, riguardano la “Gestione Gioco”che comprende le operazioni di:

- Richiesta Username – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dall'Admin per accedere e usufruire del servizio della Web Application;;
- Modifica Username e Password – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dall'Admin per accedere e usufruire del servizio della Web Application;;
- Modifica Dati Anagrafici – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dall'Admin per accedere e usufruire del servizio della Web Application;;
- Modifica Punteggio – La modalità può essere considerata atomica e non ulteriormente decomponibile, facendo riferimento alla procedura eseguita dall'Admin per accedere e usufruire del servizio della Web Application.

In particolare, l'Admin ha la possibilità di modificare e richiedere, esclusivamente, la Password

e l'Username degli User oltre che poter modificare, esclusivamente, Dati Anagrafici e Punteggio.

Figura 27 - Use Case Diagram Gestione Gioco



Fonte: Attanasio Ciro tramite ArgoUML

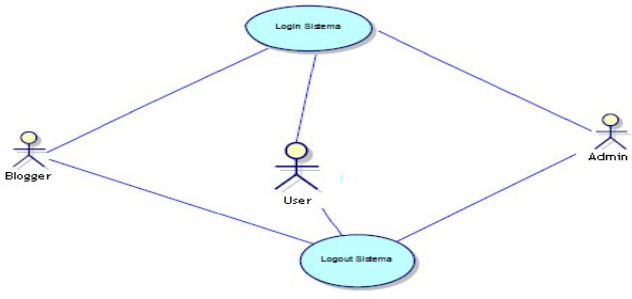
3.1.2 Use Case comuni

Infine, le modalità di utilizzo del sistema che sono comuni all'User, al Blogger e all'Admin riguarda il:

- Login Sistema – La modalità può essere considerata atomica e non ulteriormente decomponibile;
- Logout Sistema – La modalità può essere considerata atomica e non ulteriormente decomponibile.

Il tutto è riportato in Figura 28.

Figura 28 - Use Case Diagram Comuni : Login e Logout



Fonte: Attanasio Ciro tramite ArgoUML

3.2 Class Diagram

Uno degli assunti fondamentali del paradigma a oggetti è che il concetto di classe, e concetti

correlati come l'ereditarietà o il polimorfismo, si prestino a rappresentare in modo diretto e intuitivo la realtà, in qualsiasi ambito (per usare le parole di Grady Booch, "un oggetto è qualsiasi cosa a cui si possa pensare"). I diagrammi delle classi UML sono basati su versioni astratte di tali concetti, e possono essere utilizzati per descrivere sostanzialmente qualsiasi contesto a qualsiasi livello di astrazione (enfatizzandone, però, solo alcuni aspetti). Di conseguenza, UML prevede un loro impiego a livello di analisi e in particolare analisi del dominio (ovvero la descrizione del contesto in cui un sistema software deve operare), ma anche a livello di progettazione (nella descrizione della struttura interna del sistema, dei suoi componenti e delle loro relazioni).

Mediante il *Class Diagram* è possibile definire tutte quelle che sono le entità caratteristiche del sistema software e le relazioni che ci sono tra di esse. Ciascuna entità è ovviamente modellabile attraverso il concetto di classe, che ne definisce le caratteristiche (es. i dati) ed il comportamento (es. le operazioni). Ovviamente, anche il *Class Diagram* può essere realizzato a diversi livelli di astrazione, passando da un livello elevato, definito di seguito, ad un livello più basso e strettamente legato alla fase di implementazione e quindi relazionato al Framework ed al linguaggio utilizzato. Attraverso questo diagramma è possibile modellare la vita statica del sistema, tenendo anche conto di quello che sono le funzionalità offerte da quest'ultimo e rese disponibili attraverso le entità che lo compongono.

L'elemento di modello principale dei diagrammi delle classi è la classe. Una classe rappresenta una categoria di entità (istanze), nel caso particolare dette oggetti. Il nome della classe indica la categoria di entità descritta dalla classe. Ogni classe è corredata da un insieme di attributi (che descrivono le caratteristiche o lo stato degli oggetti della classe) e operazioni (che descrivono il comportamento della classe). Il simbolo grafico che rappresenta le classi UML è un rettangolo suddiviso in tre scomparti, rispettivamente dedicati al nome della classe, agli attributi e alle operazioni.

Le relazioni tra classi sono: associazione, generalizzazione (derivazione), dipendenza, raffinamento e aggregazione.

Sulla base delle specifiche definite per l'applicazione, il sistema prevede le seguenti entità e le relative classi che lo modellano:

- Ruolo – Definisce un generico utilizzatore del sistema, da distinguere tra User, Blogger ed Admin;
- User – Generico utente registrato che accede al sistema;
- Admin – Amministratore del sistema;
- Blogger – Amministratore delle registrazioni;
- Accesso – Classe stratta che contiene le informazioni relative all'accesso al sistema da parte di un utilizzatore qualsiasi;
- AccessoUser – Informazioni di accesso di un User;
- AccessoAdmin – Informazioni di accesso dell'Admin;
- AccessoBlogger – Informazioni di accesso del Blogger;
- Punteggio – Definisce la lista dei punteggi ottenuta dagli User;
- Gioca – Generica operazione per avviare il gioco.

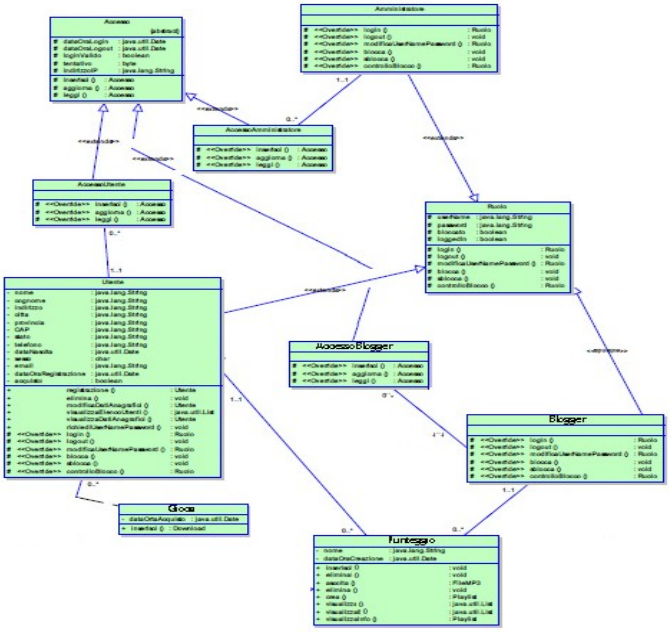
Il diagramma definito di seguito si pone ad un livello di astrazione intermedio, nell'ambito del quale i membri di ciascuna classe (campi e metodi), fanno riferimento al linguaggio utilizzato nell'implementazione, ossia PHP. Ovviamente, astraendosi da quest'ultimo, è facile osservare che la struttura del sistema che ne scaturisce è completamente indipendente dall'implementazione.

In riferimento alle dipendenze che ci sono tra le viarie classi, si possono fare le seguenti osservazioni:

- le classi *AccessoUser, AccessoBlogger* e *AccessoAdmin* costituiscono una specializzazione della classe Accesso, più da un punto di vista concettuale che pratico:
- la classe *Ruolo* generalizza la classi *User*, *Blogger* ed *Admin*, in quanto queste ultime definiscono un particolare tipo di utilizzatore del sistema, l'uno diverso dall'altro ma con alcune caratteristiche comuni.

Le altre tipologie di relazione previste sono autoesplicative, in quanto esprimono il legame tra un'entità e le altre, sulla base delle specifiche e delle funzionalità del sistema. Infine, alle classi suddette sono associate, per ciascuna di esse, le corrispondenti classi per l'accesso alla base di dati per garantire la permanenza delle informazioni.

Figura 29 - Class Diagram



Fonte: Attanasio Ciro tramite ArgoUML

3.3 Activity Diagram – Sequenz Diagram

L'**Activity Diagram** è un diagramma definito all'interno dello Unified Modeling Language

(UML) che definisce le attività da svolgere per realizzare una data funzionalità. Può essere utilizzato durante la progettazione del software per dettagliare un determinato algoritmo. Più in dettaglio, un *activity diagram* definisce una serie di attività o flusso, anche in termini di relazioni tra le attività, i responsabili per le singole attività ed i punti di decisione. L' *activity diagram* è spesso usato come modello complementare allo *Use Case Diagram*, per descrivere le dinamiche con cui si sviluppano i diversi use case.

Activity rappresenta una specifica attività che deve essere svolta all'interno della funzione. È rappresentata da un rettangolo smussato con una descrizione dell'attività.

Facendo riferimento a quelle che sono le funzionalità offerte dal sistema e le modalità di utilizzo di quest'ultimo, attraverso gli Activity Diagrams è possibile modellare le azioni necessarie per compiere un'attività ed il flusso di controllo tra di esse. Ciascuna “activity” rappresenta un'azione che viene eseguita dal sistema oppure dall'utilizzatore e puo essere atomica oppure scorporata in più “action” elementari. Nella gestione del flusso di controllo, è possibile definire esecuzioni parallele di operazioni e le loro sincronizzazioni, nonché esecuzioni condizionali. Inoltre, attraverso lo strumento dell'”object flow” è possibile specificare quali sono gli oggetti del sistema che entrano in gioco per ciascuna azione eseguita e quale sia il loro stato.

Un **Sequence Diagram** è un diagramma previsto dall'UML utilizzato per descrivere uno scenario. Uno scenario è una determinata sequenza di azioni in cui tutte le scelte sono state già effettuate; in pratica nel diagramma non compaiono scelte, né flussi alternativi. Normalmente da ogni Activity Diagram sono derivati uno o più *Sequence Diagram*; se caso l'Activity Diagram descrive due flussi di azioni alternativi, se ne potrebbero ricavare due scenari, e quindi due *Sequence Diagram* alternativi. Dalla versione 2 dell'UML è stata introdotta la possibilità di indicare nello stesso diagramma anche delle sequenze alternative. Il *Sequence Diagram* descrive le relazioni che intercorrono, in termini di messaggi, tra Attori, Oggetti di business, Oggetti od Entità del sistema che si sta rappresentando.

Attraverso i *Sequence Diagram* è possibile descrivere la vista dinamica del sistema, enfatizzando le interazioni dovute allo scambio di messaggi tra gli oggetti e il loro ordinamento temporale. Ciascun diagramma evidenzia il modo in cui uno stesso scenario, ossina uno specifico percorso in un caso d'uso, viene risolto dalla collaborazione tra una insiemi di oggetti. Generalmente, però, al posto di realizzare un diagramma per ciascuno scenario, si preferisce definire un unico diagramma all'interno del quale vengono descritte le situazioni alternative, facendo uso dei numeri di sequenza sui messaggi.

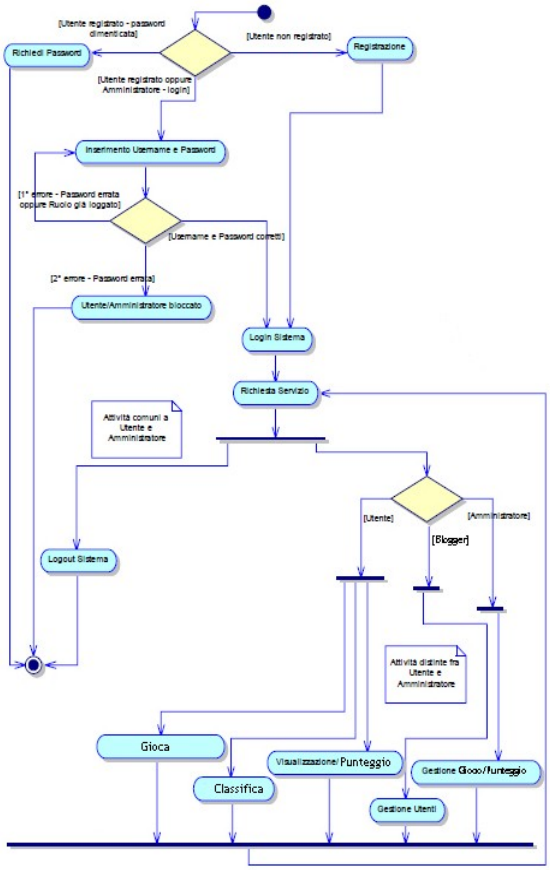
Un messaggio è un'informazione che viene scambiata tra due entità. Solitamente chi invia il messaggio, la parte attiva, è l'attore. Il messaggio è sincrono, se l'emittente rimane in attesa di una risposta, o asincrono, nel caso l'emittente non aspetti la risposta e questa può arrivare in un secondo momento. Il messaggio che viene generato in risposta ad un precedente messaggio, al quale si riferisce anche come contenuto informativo, è detto messaggio di risposta. Un messaggio, in cui il ricevente è nello stesso tempo l'emittente, è detto ricorsivo.

Da osservare che nell'ambito dei Sequence Diagram sono previsti degli oggetti di tipo “interfaccia”, che definiscono le parti del sistema che interagiscono con l'utente, ossia la cosi detta UI (User Interface). In questa fase, non viene specificata la tipologia di interfaccia utente, in modo da poter sfruttare i diagrammi realizzati anche con implementazioni diverse. È ovvio che, nel caso di una Web Application, l'interfaccia utente sia caratterizzata da una pagina Web e dai

relativi form che le compongono.

È stato definito un Activity Diagram di carattere generale, che si pone ad un elevato livello di astrazione e che descrive tutte le possibili azioni ed il relativo flusso, che possono essere eseguite da un User, da un Blogger e ovviamente dall'Admin.

Figura 30 - Activity Diagram Generale



Fonte: Attanasio Ciro tramite ArgoUML

A partire da esso, è possibile scendere ad un livello di dettaglio maggiore, specificando la sequenza delle azioni per ogni funzionalità del sistema.

Gli Activity Diagram possono specificare una procedura come sequenza di azioni che si differenziano a seconda delle scelte dell'utente o della tipologia dell'utente stesso, come si può vedere nel seguente diagramma che spiega la differente procedura. L'inserimento di uno stato iniziale ed uno stato finale consentono di disaccoppiare completamente l'attività *Consegna* rispetto al diagramma complessivo

Essi chiariscono l'esatta sequenza delle azioni compiute dall'utente o dal DBMS o dal Sistema/webServer per il corretto svolgimento delle funzionalità descritte. Una visualizzazione chiamata a Swim lane, perché ricorda le corsie di una piscina, può essere utilizzata per chiarire quali operazioni devono essere eseguite e da quale entità . In ogni corsia viene inserito il

nominativo di un attore e sotto di esso le azioni che deve compiere per far progredire il sistema.

3.3.1 Login

La procedura di Login, visualizzata proprio secondo il metodo delle Swim lane, presenta una struttura standard: inserimento di Username e Password, verifica da parte del DBMS e caricamento delle pagine da parte del sistema/Web server.

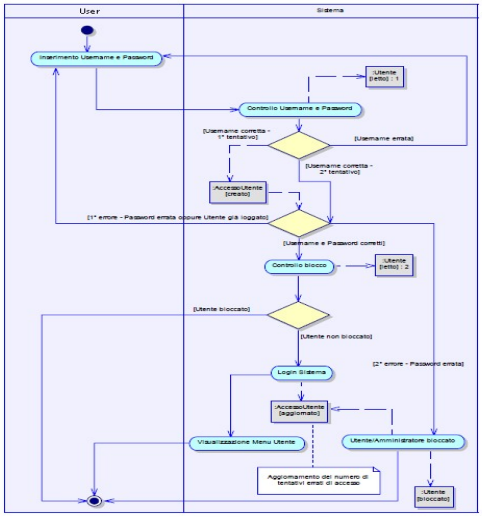
Viene creato un semplice form per gestire il login. Il form oltre a richiedere l'inserimento di username e password, consente anche di spuntare la casella 'Salva LogIn' per memorizzare i dati in un apposito cookie.

Una volta recuperato l'identificativo del corso, da un'opportuna variabile di sessione, il processing dei risultati sarà passato allo stesso script che redireziona poi l'User alla relativa pagina.

Per quanto riguarda in particolare il Login dell'User, dell'Admin e del Blogger, le azioni da eseguire sono praticamente le stesse a meno degli oggetti che entrano in gioco.

Attraverso questo Activity Diagram viene mostrato il comportamento del sistema durante la fase di login che riguarda l'User.

Figura 31 - Activity Diagram Login User



Fonte: Attanasio Ciro tramite ArgoUML

Una volta inseriti Username e Password, il sistema ne effettua il controllo della correttezza, valutando in primo luogo se lo Username esiste e successivamente verificando la validità della Password. Ovviamente, a colui che tenta di accedere, viene sempre dato un messaggio generico di errore, senza mai specificare quale dei due parametri sia errato.

Per evitare tentativi di accesso continuamente errati, magari eseguiti non effettivamente dalla persona titolare dell'account, il sistema blocca l'utilizzatore dopo due tentativi di accesso

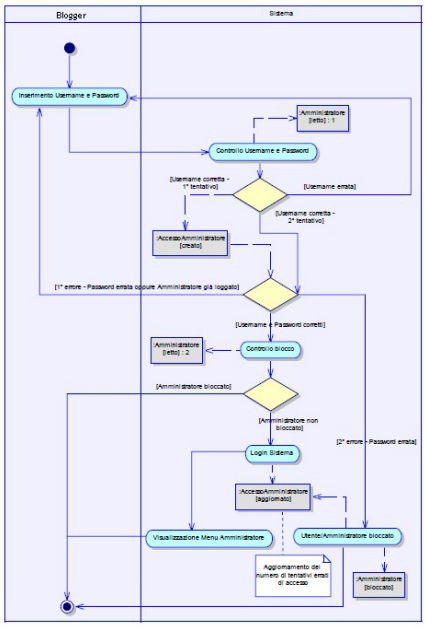
consecutivi errati. Tale blocco può essere rimosso esclusivamente contattando ilo Blogger, che ha i diritti di accesso alla gestione degli User.

Ovviamente, nel caso in cui Username e Password siano corretti, si valuta comunque se è attivo un blocco causato da tentativi errato precedenti: nel caso di Login consentito, si esegue l'aggiornamento delle informazioni di accesso.

La monitorizzazione e registrazione degli accessi, validi o non validi, relativi agli utilizzatori della Web Application, può essere considerato uno strumento mediante il quale sia possibile eseguire delle statistiche o comunque ricercare eventuali frodi e tentativi di accesso non legittimi al sistema.

Attraverso questo Activity Diagram viene mostrato il comportamento del sistema durante la fase di login che riguarda il Blogger

Figura 32 - Activity Diagram Login Blogger



Fonte: Attanasio Ciro tramite ArgoUML

Il sistema o l'attore responsabile di una determinata attività è rappresentato tramite una descrizione a cui afferisce una porzione dello schema, individuata da una linea verticale (swimlane).

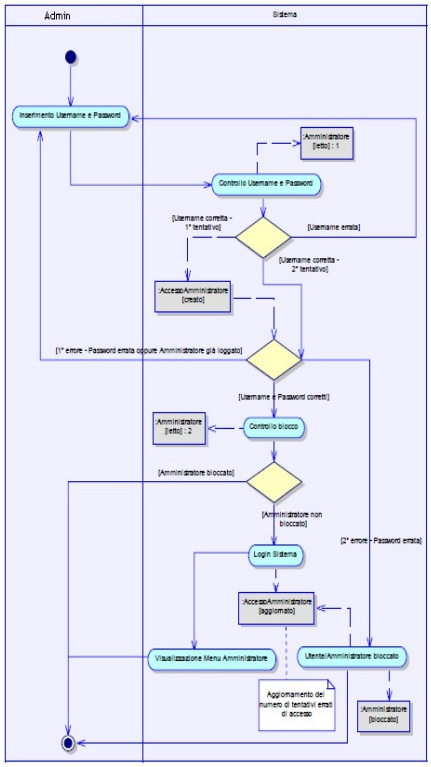
Ovviamente, sono previste tutte le operazioni di validazione dei dati, necessarie a garantire che i dati immessi dal Blogger rispettino i formati attesi dalla Web Application.

Attraverso il sequence diagram di UML si può modellare il funzionamento temporale della applicazione. In particolare può essere utile descrivere cosa avviene durante la fase di login di un utente nell'area di amministrazione del sito web.

Attraverso questo Activity Diagram viene mostrato il comportamento del sistema durante la

fase di login che riguarda l'Admin.

Figura 33 - Activity Diagram Login Admin



Fonte: Attanasio Ciro tramite ArgoUML

Attraverso il sequence diagram di UML si può modellare il funzionamento temporale della applicazione. In particolare può essere utile descrivere cosa avviene durante la fase di login di un utente nell'area di amministrazione del sito web.

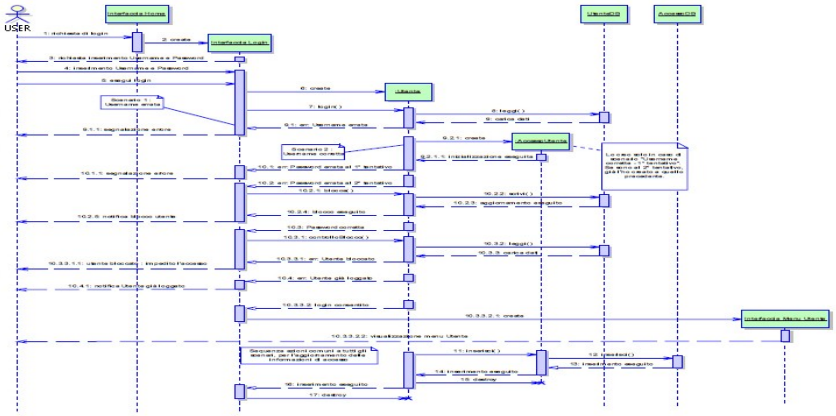
Nell'ambito del Sequence Diagram, è possibile osservare lo scenario di "Username Errato" e quello di "Username Corretto", nell'ambito del quale si possono verificare le seguenti situazioni distinte:

- Password errata al primo tentativo;
- Password errata al secondo tentativo che comporta il blocco dell'utilizzatore;
- Password corretta ma l'utilizzatore è bloccato;
- Utilizzatore già loggato;
- Login consentito.

L'utente fornisce in una apposita pagina l'username e la password., la cui validità viene controllata attraverso la classe User. Se non viene trovato un match, il login fallisce e viene lanciata un eccezione.

Per quanto riguarda il Sequence Diagram Login User, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

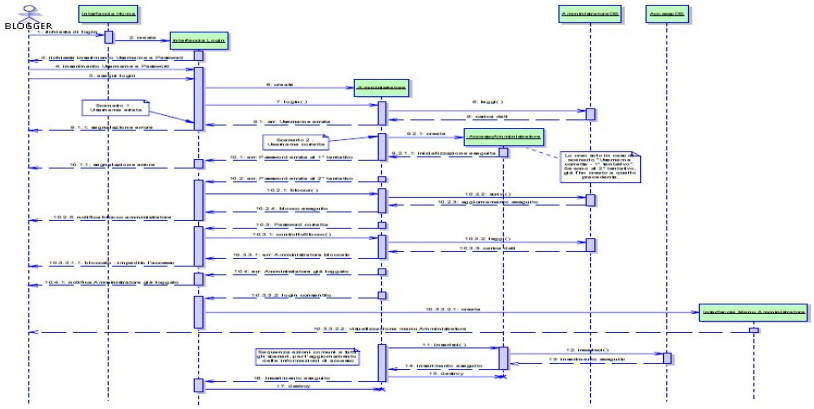
Figura 34 - Sequence Diagram Login User



Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il Sequence Diagram Login Blogger, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

Figura 35 - Sequence Diagram Login Blogger



Fonte: Attanasio Ciro tramite ArgoUML

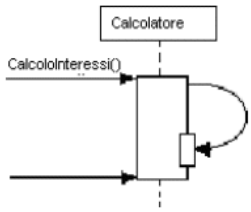
I sequence diagrams, come si nota, forniscono a possibilità di rappresentare delle istruzioni condizionali (if) e dei cicli while in cui ogni condizione descritta da un if (oppure da un while) viene racchiusa tra parentesi quadrate (nel caso del while la parentesi quadrata sinistra viene preceduta

da un asterisco). Si veda l'esempio più avanti per maggiore chiarezza.

Esiste la possibilità, come si nota, di **creare un oggetto**. Quando ciò avviene, l'oggetto creato viene rappresentato nel modo visto finora e cioè con un rettangolo all'interno del quale viene descritto il nome dell'oggetto.

La differenza, in questo caso, consiste nel fatto che l'oggetto non viene posizionato alla sommità del sequence diagram come si farebbe con gli altri oggetti ma, piuttosto, lungo la dimensione verticale che indica il tempo in cui tale oggetto viene creato. Può accadere, come si nota, che un oggetto esegua un'operazione che invochi lo stesso oggetto che l'ha provocata. In tal caso si parla di **ricorsione**

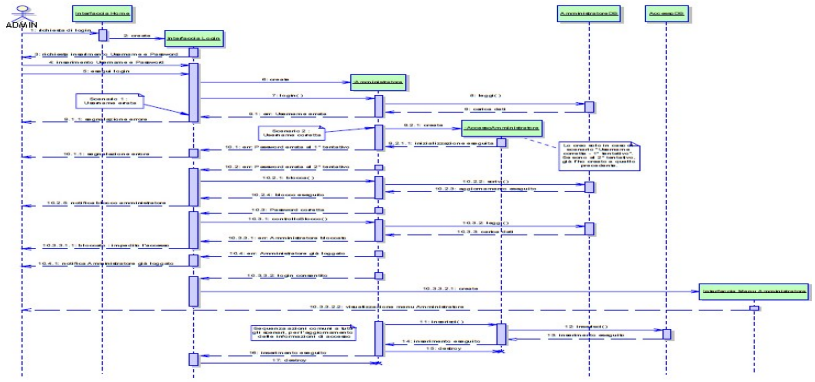
Figura 36 - Sequence Diagram: Esempio di Ricorsione



Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il Sequence Diagram Login Admin, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

Figura 37 - Sequence Diagram Login Admin



Fonte: Attanasio Ciro tramite ArgoUML

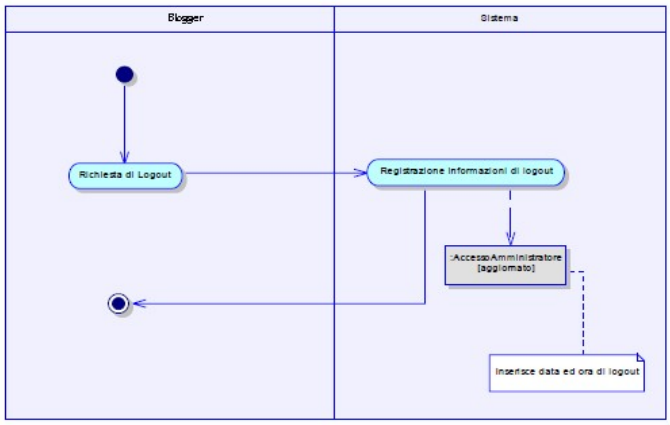
3.3.2 Logout

Anche le funzionalità di Logout prevede le medesime azioni per User, Blogger ed Admin, a

meno degli oggetti che vengono utilizzati per le operazioni necessarie.

Per quanto riguarda il *Activity Diagram* Logout Blogger, illustra tutti o buona parte degli stati che sono stati di azione e nel quale tutte o buona parte delle transizioni sono innescate dal completamento di azioni negli stati sorgenti.

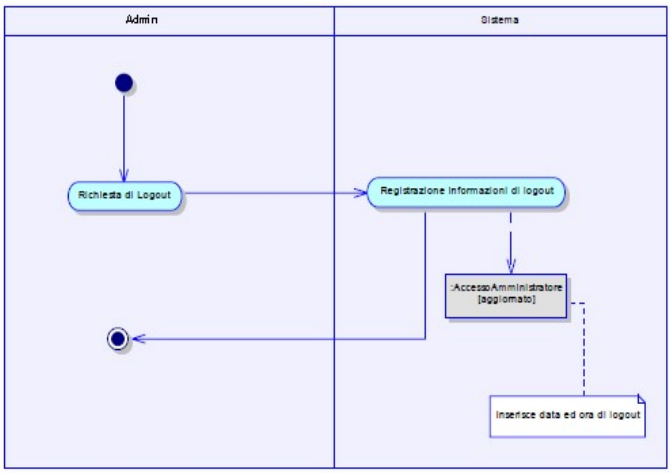
Figura 38 - Activity Diagram Logout Blogger



Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il *Activity Diagram* Logout Admin, illustra tutti o buona parte degli stati che sono stati di azione e nel quale tutte o buona parte delle transizioni sono innescate dal completamento di azioni negli stati sorgenti.

Figura 39 - Activity Diagram Logout Admin

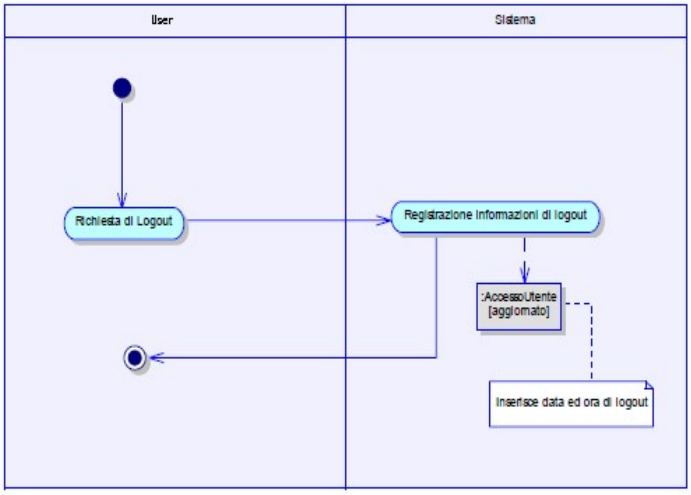


Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il *Activity Diagram* Logout User, illustra tutti o buona parte degli stati che

sono stati di azione e nel quale tutte o buona parte delle transizioni sono innescate dal completamento di azioni negli stati sorgenti.

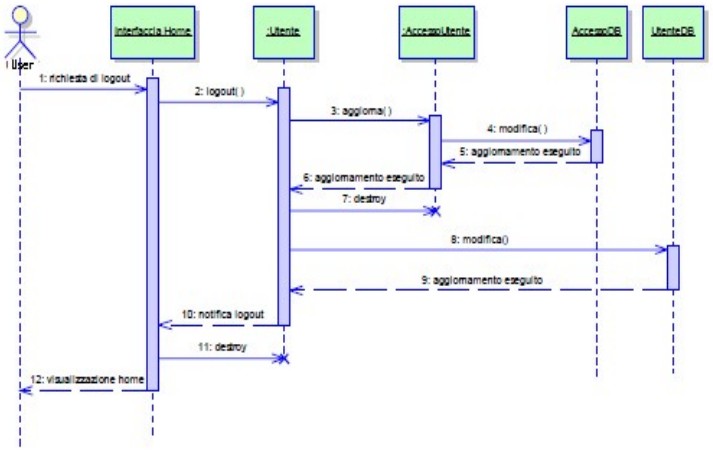
Figura 40 - Activity Diagram Logout User



Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il Sequence Diagram Logout User, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

Figura 41 - Sequence Diagram Logout User

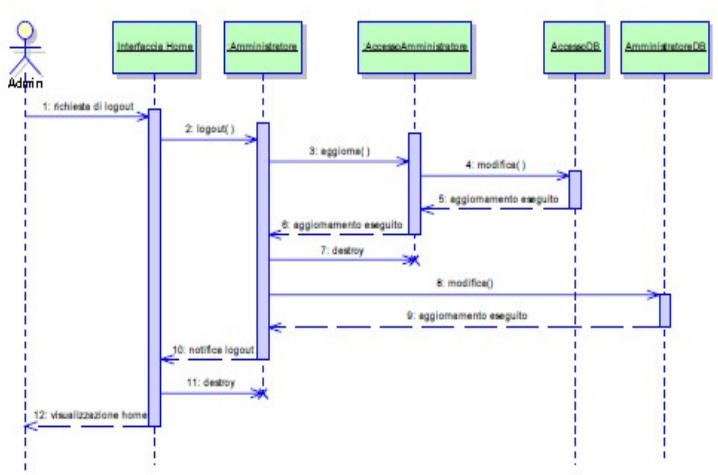


Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il Sequence Diagram Logout Admin, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla

interazione e la sequenza dei messaggi scambiati.

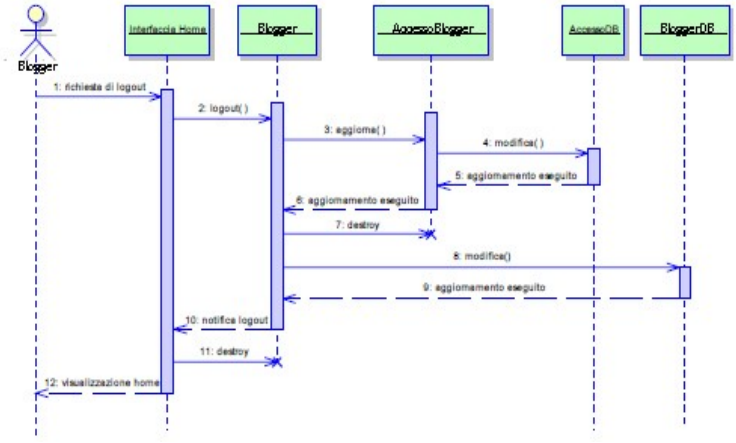
Figura 42 - Sequence Diagram Logout Admin



Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il Sequence Diagram Logout Blogger, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

Figura 43 - Sequence Diagram Logout Blogger



Fonte: Attanasio Ciro tramite ArgoUML

3.3.3 Registrazione User

Per quanto riguarda la Registrazione dell'User, per poter usufruire dei servizi della Web Application, ne sono previste due tipologie: la versione “Base” ed “Estesa”, che differiscono dal

fatto che quest'ultima fornisce in più la possibilità di cancellare da soli il punteggio realizzato.

La registrazione di base prevede l'immissione da parte del Blogger a nome dell'User, soltanto dei dati di accesso, Username e Password, oltre che dell'indirizzo E-Mail. In questo modo, si ha la possibilità di usufruire solo di alcuni servizi.

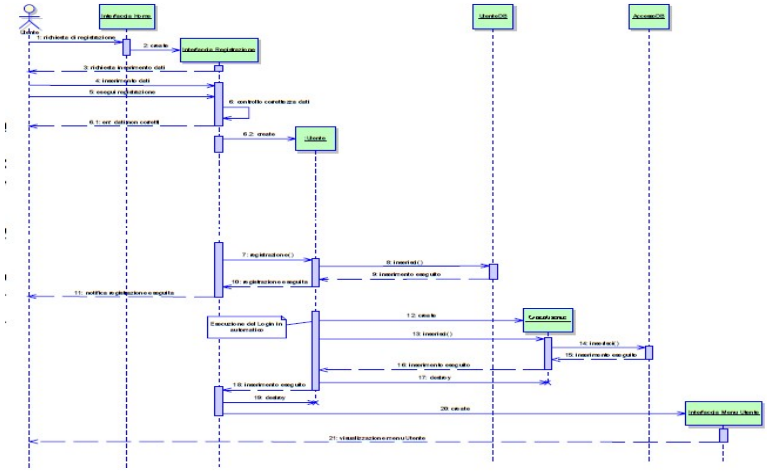
L'User ha poi la possibilità di decidere se vuole usufruire di altri servizi, nel qual caso deve far immettere al Blogger anche i dati anagrafici.

Ovviamente, sono previste tutte le operazioni di validazione dei dati, necessarie a garantire che i dati immessi dal Blogger su dettatura dell'User rispettino i formati attesi dalla Web Application.

Al termine della registrazione, l'User potrà eseguire il login al sistema, visualizzando all'User stesso le voci di menù relative alle funzionalità abilitate.

Per quanto riguarda il Sequence Diagram Logout Registrazione, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

Figura 44 - Sequence Diagram Registrazione



Fonte: Attanasio Ciro tramite ArgoUML

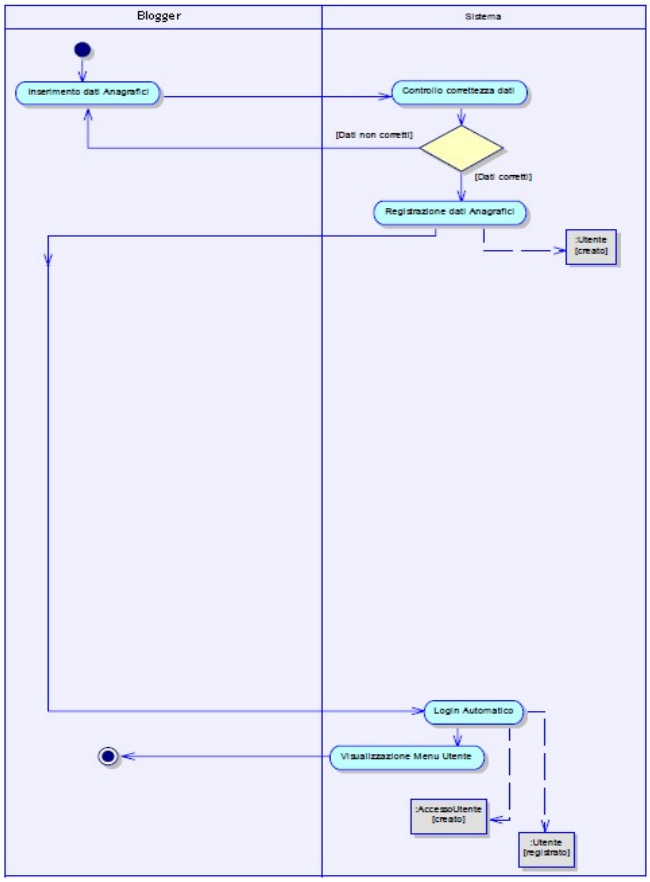
In maniera uguale agli **use case diagrams** in cui è possibile mostrare o un'istanza (uno scenario) singola di uno use case o, più genericamente, un insieme degli scenari di un use case, così i **sequence diagrams**, come si nota, possono essere definiti su una singola istanza oppure definire un modello più generico.

Registrazione “base degli User” per giocare e gestione degli utenti registrati, con la possibilità di visionare i dati ed eseguire le operazioni di cancellazione e sblocco, qualora un utente voglia rigiocare.

Per quanto riguarda il *Activity Diagram* Logout Registrazione, illustra tutti o buona parte degli stati che sono stati di azione e nel quale tutte o buona parte delle transizioni sono innescate dal

completamento di azioni negli stati sorgenti.

Figura 45 - Activity Diagram Registrazione



Fonte: Attanasio Ciro tramite ArgoUML

Gli User non hanno la possibilità di modificare i propri dati di accesso e le informazioni anagrafiche ma si deve rivolgere all'Admin.

Il sistema visualizza i dati memorizzati nel Database permettendone l'aggiornamento.

Nel caso i dati venissero modificati, il sistema esegue controlli formali sui dati inseriti ed in particolare viene verificato che la nuova password inserita e la conferma coincidano.

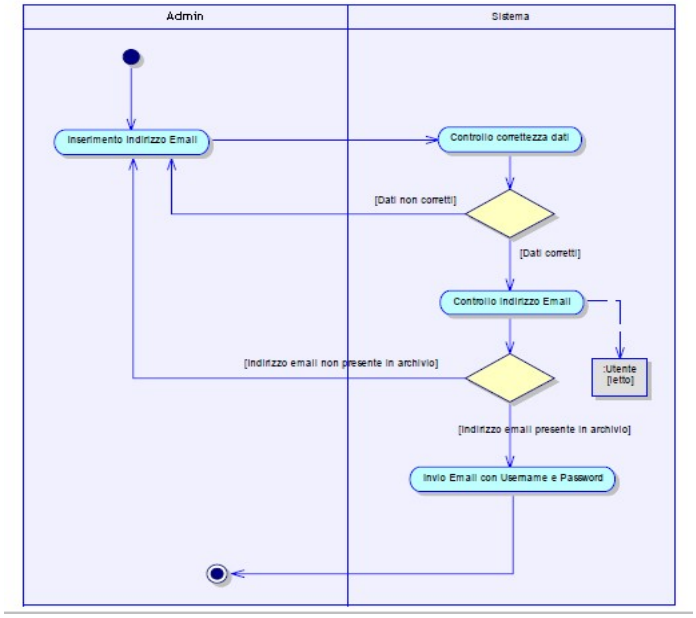
3.3.4 Richiesta Username / Password

Nel caso in cui un User abbia perso oppure dimenticato le informazioni relative all'accesso, è possibile utilizzare la funzionalità del sistema, rivolgendosi all'Admin, che permette di far comparire a schermo tali informazioni.

Per quanto riguarda il *Activity Diagram* Logout User, illustra tutti o buona parte degli stati che sono stati di azione e nel quale tutte o buona parte delle transizioni sono innescate dal

completamento di azioni negli stati sorgenti.

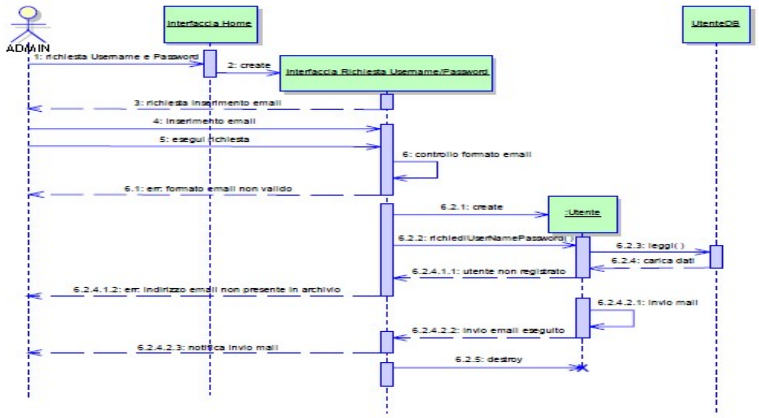
Figura 46 - Activity Diagram Richiesta Username / Password



Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il Sequence Diagram Richiesta Username / Password, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

Figura 47 - Sequence Diagram Richiesta Username / Password



Fonte: Attanasio Ciro tramite ArgoUML

L'User deve semplicemente fornire al sistema l'Username con cui ha effettuato la

registrazione, dopodiché il sistema stesso esegue il controllo di correttezza del formato e verifica che quest'ultimo sia correttamente registrato. In caso di esito positivo, L'User riceverà una stampa dal Blogger con tutte le informazioni richieste.

3.3.5 Gestione User

Il Blogger ha accesso ad una particolare sezione della Web Application, mediante la quale può effettuare semplici operazioni relative alla gestione degli User.

La prima disponibile è quella che permette la visualizzazione dell'elenco degli User, con la possibilità di selezionare uno o più User e di eseguirne la cancellazione dal sistema.

A partire da tale elenco, è inoltre possibile selezionare uno degli utenti per poter visualizzare le informazioni di registrazione, ovviamente esclusa la password di accesso al sistema.

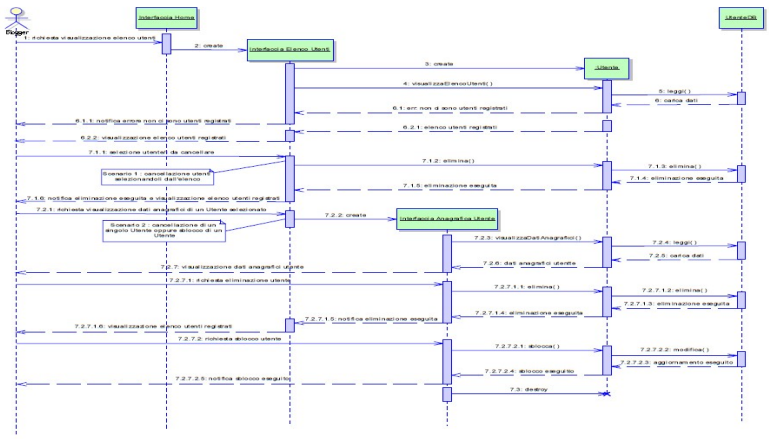
Le informazioni visualizzate dipendono dal tipo di registrazione che ha eseguito il Blogger su richiesta dell'User. In virtù di questa distinzione, saranno visualizzate solo le informazioni di accesso oppure anche i dati anagrafici.

In corrispondenza dell'interfaccia di visualizzazione dei dati del singolo User, il Blogger può eseguire due semplici operazioni:

- Eliminare l'User;
- Sbloccare l'User, qualora quest'ultimo risulti bloccato in virtù del fatto che sono stati eseguiti due accessi consecutivi al sistema.

Per quanto riguarda il Sequence Diagram Gestione User, illustra le interazioni tra gli oggetti disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

Figura 48 - Sequence Diagram Gestione User

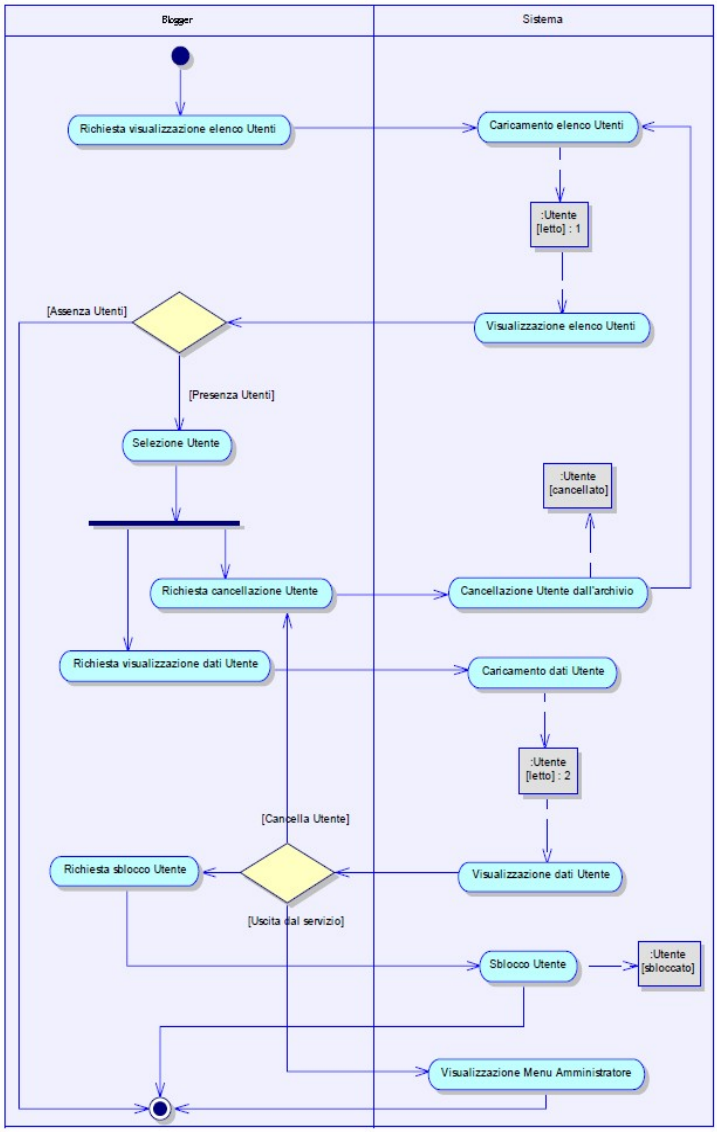


Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il *Activity Diagram* Logout User, illustra tutti o buona parte degli stati che

sono stati di azione e nel quale tutte o buona parte delle transizioni sono innescate dal completamento di azioni negli stati sorgenti.

Figura 49 - Activity Diagram Gestione User



Fonte: Attanasio Ciro tramite ArgoUML

3.3.6 Gestione Gioco

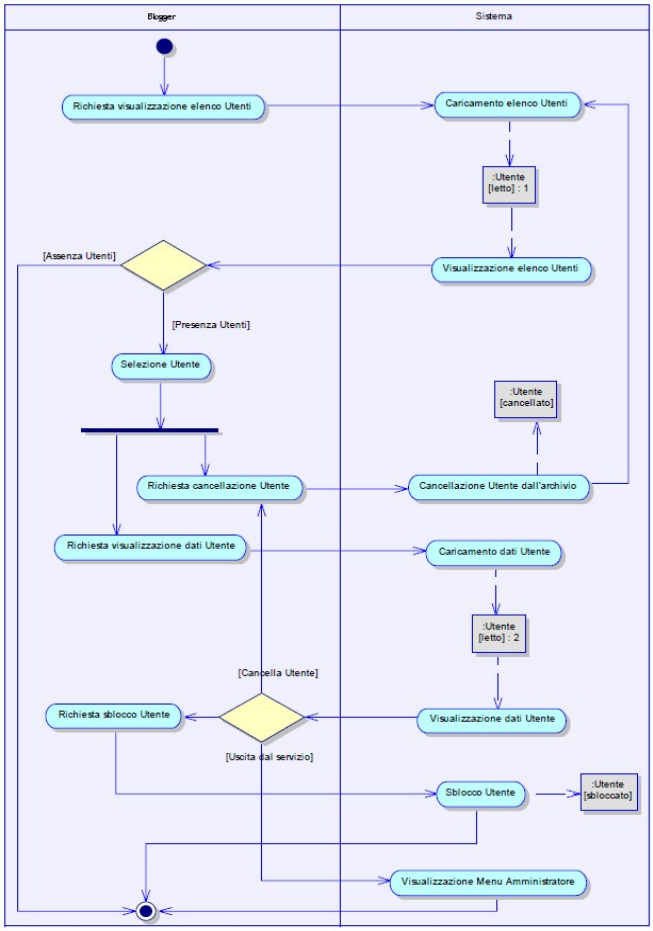
La Gestione Gioco rappresenta il servizio principale offerto dalla Web Application per l'Admin. Attraverso questo servizio è possibile eliminare un punteggio e non solo.

L'Admin può inserire, modificare o eliminare un punteggio. Questo permette di gestire correttamente i nuovi punteggi dei singoli User e di modificare i punteggi dei singoli User già presenti.

L'operazione di di un punteggio è relativamente semplice, in quanto l'unica informazione associata al punteggio è il nome con cui User è associato a quest'ultimo.

Per quanto riguarda il *Activity Diagram* Logout User, illustra tutti o buona parte degli stati che sono stati di azione e nel quale tutte o buona parte delle transizioni sono innescate dal completamento di azioni negli stati sorgenti.

Figura 50 - Activity Diagram Gestione Gioco



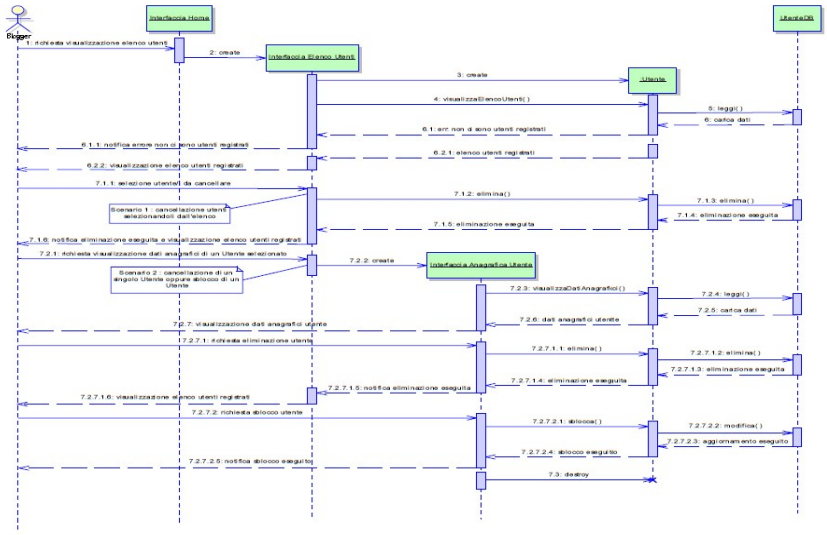
Fonte: Attanasio Ciro tramite ArgoUML

Il Sequence Diaagram evidenzia i due scenari possibili per la cancellazione di uno o più User, nonché la funzionalità di sblocco.

Per quanto riguarda il Sequence Diagram Logout Admin, illustra le interazioni tra gli oggetti

disponendole lungo una sequenza temporale. In particolare mostra gli oggetti che partecipano alla interazione e la sequenza dei messaggi scambiati.

Figura 51 - Sequence Diagram Gestione Gioco



Fonte: Attanasio Ciro tramite ArgoUML

3.4 Statechart Diagram

Gli Statechart Diagram costituiscono uno strumento mediante il quale è possibile descrivere una state machine, enfatizzando il flusso di controllo tra gli state.

Una state machine definisce un comportamento che specifica i vari state che un oggetto può assumere durante la sua vita in risposta a certi eventi. Ogni state rappresenta una condizione di un oggetto, in cui sono soddisfatti certi requisiti, mentre un evento è la specificazione di un accadimento significativo, posizionabile nel tempo e nello spazio, che determina una transazione di stata da parte dell'oggetto. Quest'ultima esprime il fatto che, un oggetto a partire da un certo stato ed al verificarsi di un determinato evento passa in uno stato differente, poiché sono cambiate le condizioni in cui si trova. Sulla base di quanto detto, un diagramma di questo tipo può rappresentare un'estensione della descrizione di un automa a stati finiti.

Lo *State Chart Diagram* è un diagramma previsto dall'UML per descrivere il comportamento di entità o di classi in termini di stato (macchina a stati). Il diagramma mostra gli stati che sono assunti dall'entità o dalla classe in risposta ad eventi esterni. Il concetto di stato è spesso posto in relazione a ciclo di vita; l'insieme completo di stati che un'entità o una classe può assumere, dallo stato iniziale a quello finale, ne rappresenta il ciclo di vita. Gli elementi rappresentati da uno *State Chart Diagram* sono lo stato - distinguendo tra iniziale, intermedi e stato finale - l'evento, l'azione e la guardia. Lo stato descrive una qualità dell'entità o classe che si sta rappresentando (pratica aperta, in lavorazione, sospesa, chiusa); l'evento è la descrizione dell'azione che comporta il cambiamento di stato, l'azione è l'evento che ne consegue, la guardia è l'eventuale condizione

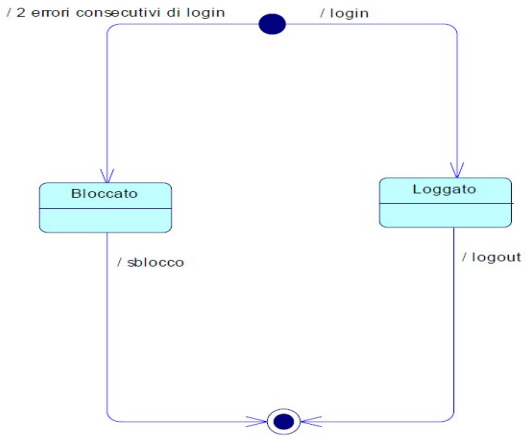
che si deve verificare perché si possa compiere l'azione.

Considerando il case study sviluppato, sono stati definiti gli Statechart Diagrams per quelli oggetti che possono compiere delle transazioni significative durante la vita della Web Application. In particolare sono:

- Admin;
- Blogger;
- User;
- Punteggio,

Per quanto riguarda L'Admin, quest'ultimo può trovarsi fondamentalmente in due possibili stati: Loggato e bloccato. Ovviamente, bisogna tener conto che la condizione di Admin non loggato rappresenta il punto di inizio del diagramma. Lo stato “loggato” si raggiunge dopo aver eseguito correttamente il login, mentre lo stato “bloccato” dopo aver commesso due errori consecutivi di accesso.

Figura 52 - Statechart Admin



Fonte: Attanasio Ciro tramite ArgoUML

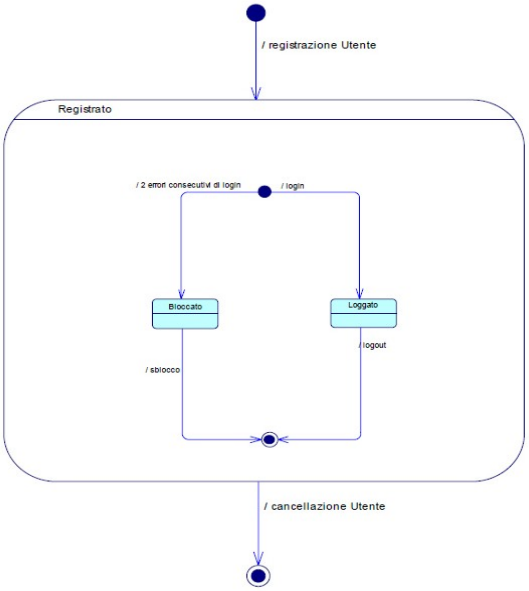
Gestione completa dell'archivio punteggio, con la possibilità di inserire, cancellare e modificare i punteggi, usufruendo della funzionalità automatica del sistema di garantire coerenza tra i tag SCORE degli stessi e le informazioni immesse nella Base di Dati. Possibilità di visionare i dati degli User ed eseguire le operazioni di recupero Login e Password, qualora un utente che voglia rigiocare li abbia dimenticati; Operazione di login e logout.

L'User prevede una situazione leggermente complicata, in quanto si parte da uno stato iniziale in cui l'User non è registrato. Una volta eseguita la registrazione a mezzo Blogger, l'User passo allo stato “registrato” ed all'interno di esso può evolvere tra gli stessi stati dell'Admin e del Blogger, ossia “loggato” e “bloccato”, sulla base dei medesimi eventi.

Lo Statechart Diagram prevede una struttura innestata, nell'ambito della quale all'interno dello stato “registrato” è previsto un ulteriore diagramma che descrive le possibili transazioni di stato

dell'Blogger che ha eseguito la registrazione.

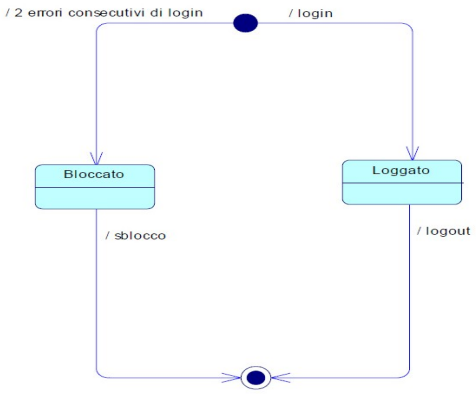
Figura 53 - Statechart Users



Fonte: Attanasio Ciro tramite ArgoUML

Per quanto riguarda il Blogger, quest'ultimo può trovarsi fondamentalmente nella stessa identica situazione dell'Admin già descritta in precedenza. Questo comporta la seguente situazione descritta un Figura 53.

Figura 54 - Statechart Blogger

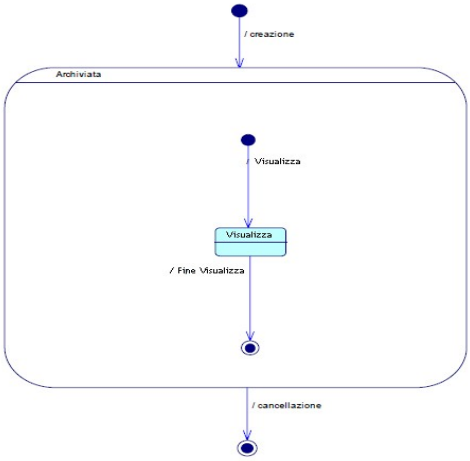


Fonte: Attanasio Ciro tramite ArgoUML

Il Punteggio non può essere paragonata a nessuna operazione precedente avendo una complessità inferiore. Il Punteggio passa nello stato “archiviato”, nell'ambito del quale può

trovarsi o meno nello stato di “ascolto”..

Figura 55 - Statechart Punteggio



Fonte: Attanasio Ciro tramite ArgoUML

3.5 Conceptual Data Model

Identifica ad un più alto livello le relazioni tra le diverse entità. Caratteristiche del modello concettuale dei dati sono:

- Le entità e le relazioni tra di loro;
- Nessun attributo specificato;
- Nessuna chiave primaria specificata.

La definizione del modello concettuale rappresenta il primo passo verso la progettazione della Base di Dati. Esso permette di descrivere tutte quelle che sono le entità del mondo reale e le relazioni che ci sono fra essi. Inoltre, per ciascuna entità o relazione che sia, è possibile definire gli attributi caratteristici. In un certo senso, lo stesso Class Diagram può essere considerato in moltissimi casi il modello concettualmente di una database, poiché le entità e le relazioni rappresentate sono praticamente le stesse, a meno di un formalismo differente.

Facendo riferimento alla Web Application in esame, le entità del modello sono le seguenti:

- AccessoUser, AccessoBlogger ed AccessoAdmin – rappresentano le informazioni di accesso dell'User, del Blogger e dell'Admin. Esse rappresentano una specializzazione dell'entità Accesso, alla quale sono legate con una relazione del tipo Generalizzazione/Specializzazione;
- User, Blogger ed Admin – definiscono le informazioni di un User, del Blogger e dell'Admin. Sono una Specializzazione dell'entità Ruolo;
- Punteggio – rappresenta il generico punteggio realizzato dall'User.

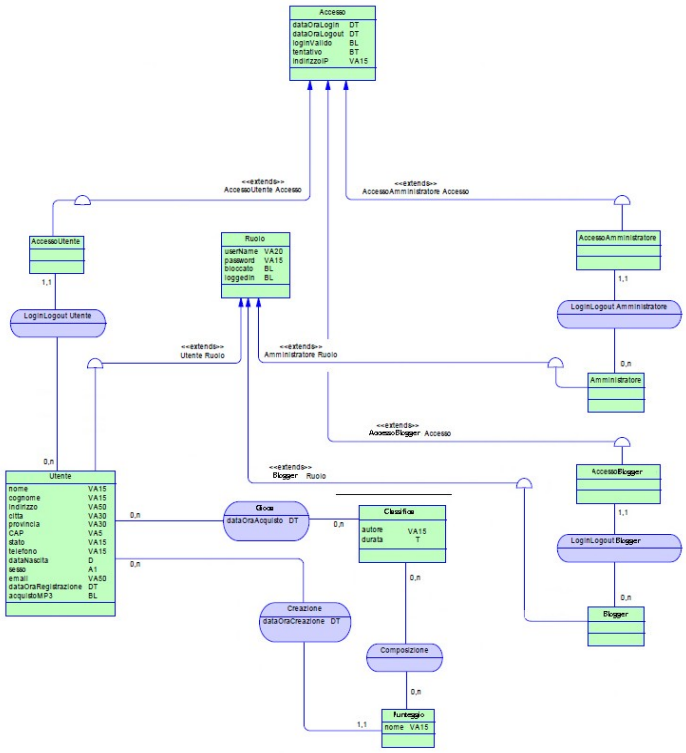
È da osservare che, rispetto al Class Diagram non è presente l'entità E_E, in virtù del fatto che l'applicazione non memorizza in maniera permanente le sue informazioni all'interno della Base di Dati, ma utilizza sempre un oggetto in memoria che al termine dei controlli necessari viene distrutto.

Inoltre, le relazioni che legano queste entità sono:

- LoginLogout User;
- LoginLogoutBlogger;
- LoginLogout Admin.

Infine, per ciascuna entità e per ciascuna relazione sono specificate gli attributi che le caratterizzano con il corrispondente tipo di dato. È da osservare che alcune relazioni non hanno attributi, in quanto servono ad esprimere esclusivamente un associazione tra due entità ma non hanno informazioni proprie. Nel modello logico, alcune di queste sono destinate a scomparire, ossia LoginLogout User, LoginLogout Blogger e LoginLogout Admin.

Figura 56 - Conceptual Data Model



Fonte: Attanasio Ciro tramite ArgoUML

3.6 Physical Data Model

Per completare la progettazione della base di Dati, è necessario effettuare la trasformazione

del modello concettuale nel modello logico, anche detto modello fisico. Quest'ultimo è costituito da una serie di tabelle e le relative colonne che prendono il posto delle entità e delle relazioni con i relativi attributi, presenti nel modello concettuale. La trasformazione viene eseguita sulla base di una serie di regole, che permettono di adattare il modello reale, rappresentato dal modello concettuale, all'implementazione fisica che verrà successivamente adottata per uno specifico DBMS (Data Base Management System).

Il livello fisico nasce dall'estensione del livello medio aggiungendo le necessarie chiavi e le caratteristiche fisiche al modello. Le caratteristiche fisiche del modello dei dati dipendono in gran parte dal DBMS che intendiamo usare. Una volta sviluppato esso sarà rappresentato da una serie di tabelle.

In questo modello bisogna anche tener conto delle performance che si vogliono ottenere e di conseguenza decidere il livello di granularità dei dati ed il loro partizionamento. Le considerazioni si fanno soprattutto tenendo conto dell'uso delle risorse di I/O.

L'obiettivo da perseguire è quello di ottenere il maggior numero di record ad ogni operazione di I/O: ossia, di trasferire in memoria un gruppo di record che ha un'alta probabilità di essere quello richiesto dall'end-user in modo tale da ridurre il più possibile il numero di I/O.

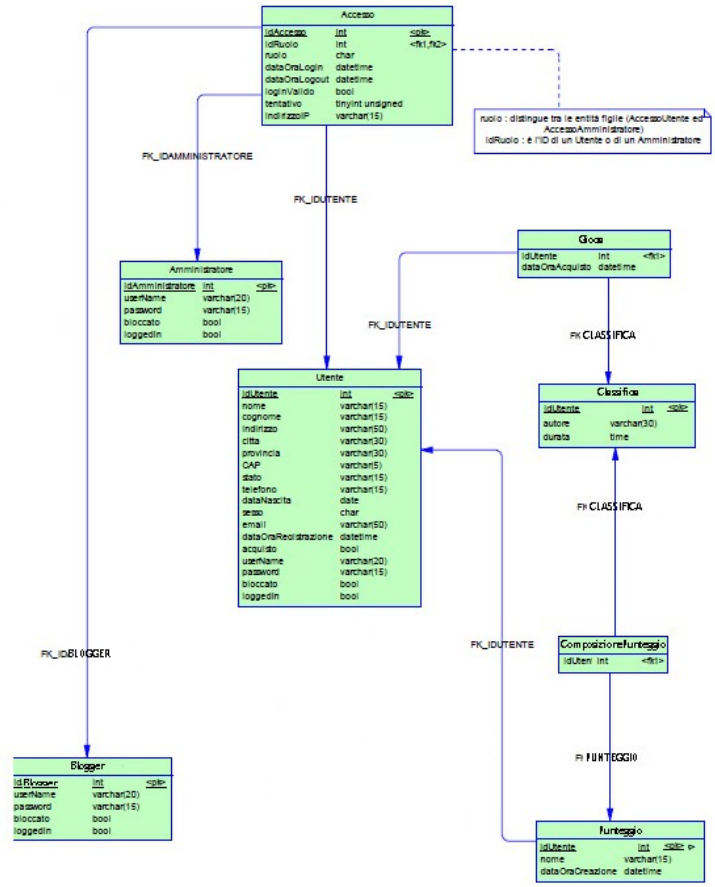
Il fatto che inoltre l'end-user non possa modificare i dati del data warehouse libera lo sviluppatore da alcuni fattori di cui altrimenti dovrebbe tener conto (es. la concorrenza che si potrebbe instaurare in un processo di scrittura o la scelta da parte dell'utente di dove andare a salvare dei dati).

Le principali operazioni di trasformazione sono le seguenti:

- Eliminare tutte le relazioni di tipo Generazionale/Specializzazione, sulla base di tre possibili alternative:
    1. Accorpamento delle entità figlie nel padre, con l'introduzuine di un attributo che le possa distinguere;
    2. Accorpamento del padre nelle entità figlie;
    3. Sostituzione delle generalizzazioni con semplici associazioni.
- Scegliere uno o più attributi di ciascuna entità che possono costituire una chiave primaria, ossia tale da individuare ogni istanza dell'entità stessa in maniera univoca. Nel caso in cui l'entità non sia caratterizzata da attributi con questo tipo di funzionalità, è possibile introdurre un nuovo attributo che abbia il solo ruolo di chiave primaria;
- Ogni entità va rappresentata con una tabella, le cui colonne sono esattamente gli attributi dell'entità stessa;
- Una relazione uno-a-molti tra due entità non viene rappresentata con una tabella ma viene assorbita dall'entità lato “uno”;
- Una relazione molti-a-molti viene rappresentata attraverso una tabella che sarà legata alle due tabelle relative alle entità in gioco, sulla base delle loro chiavi primarie;
- Una relazione uno-a-uno non viene rappresentata attraverso una tabella ma è

assorbita indifferentemente da una delle due entità legate.

Figura 57 - Physical Data Model



Fonte: Attanasio Ciro tramite ArgoUML

Sulla base delle regole suddette, il modello logico risultante ha le seguenti caratteristiche:

- In tutte le tabelle è stato introdotto un attributo (colonna) che ne rappresenti la chiave primaria, con un nome del tipo “idNomeTabella” per esprimere che si tratta di un identificativo univoco;
- La relazione Generalizzazione/Specializzazione avente come padre l'entità Accesso e come figlie le entità AcessoUser, AcessoBlogger e AccessoAdmin è stata eliminata accorpando queste ultime nel padre. Per poter distinguere le occorrenze dell'una e dell'altra, è stato introdotto l'attributo “ruolo” nella tabella Accesso definitiva. La scelta della soluzione da adottare è basata sul fatto che le due entità figlie sono caratterizzate da occorrenze concettualmente diverse ma che hanno la medesime informazioni;
- La relazione Generalizzazione/Specializzazione avente come padre l'entità Ruolo e

come figlie le entità User, Blogger ed Admin è stata eliminata accorpando allinterno di queste ultime, l'entità padre. Tutto ciò determina la presenza di tre tabelle distinte per User, Blogger ed Admin ed è da sottolinerare che la scelta fatta è motivata dalla presenza di informazioni completamente diverse chew caraterizzano i tre utilizzatori;

- Le relazioni uno-a-molti da User, Blogger ed Admin verso Accesso, sono state risolte introducendo all'interno di qest'ultima tabella uan chiave esterna “idRuolo” che corrisponde alle chiavi primarie “idUser”, “idBlogger” e “idAdmin”. È da osservare che l'identificatore univoco di un accesso, appartenete ad un User, al Blogger oppure all'Admin, è individuato dalla coppia “idRuolo-ruolo”.

Oltre a queste trasformazioni, sono stati introdotti dei particolari vincoli (costraints) di “casade” sugli eventi 2onDelete”, in modo da non lasciare informazioni inutili che rendono il database “sporco”. Qui di seguito sono elencate le tabelle interessate, le chiavi esterne e le motivazioni dei vincoli:

- Le due chiavi esterne della tabella Visualizza, in modo che se si cancella un User si perdono anche le tracce dei punteggi realizzati;
- Le chiavi esterne della tabella Gioca, in modo che cancellando un User siano eliminati anche i sui dati anagrafici.

Attraverso i legami che ci sono tra le tabelle ed i vincoli di “casade” definiti sugli eventi “onDelete”, si determinano delle operazioni di cancellazione a catena che permettono di lasciare nel database solo ed esclusivamente i dati necessari (es. cancellando un User vengono eliminati i punteggi ad essi associati)

Capitolo VI – Case Study : Implementazione a confronto

La scienza è sempre imperfetta. Ogni volta che risolve
un problema ne crea almeno dieci nuovi
(George Bernard Shaw)

1. Introduzione

Il confronto tra due tecnologie, che nello caso specifico specifico sono i due Framework di sviluppo cakePHP e Zend Framework, può essere considerato effettivamente valido soltanto nel momento in cui viene supportato da una prova sul campo. Infatti, non basta descrivere in maniera puramente discorsiva le caratteristiche dell'uno e dell'altro Framework, ma è necessario applicare le due tecnologie al medesimo caso di studio, per poter dare maggiore risalto al confronto tecnico.

Per questo motivo, la Web Application precedentemente progettata è stata implementata sia con cakePHP che con Zend Framework, in modo da valutare con quali modalità i due Framework permettono di realizzare le medesime funzionalità. Ovviamente, facendo seguito alla fase di progettazione, il sistema software risultante ha un'architettura diversa per ciascuna tecnologia, sulla base delle potenzialità che differenziano l'una dall'altro.

I termini di paragone possono riguardare:

- La semplicità con cui un Framework permetta di realizzare una funzionalità rispetto all'altro, attraverso gli strumenti che fanno parte della propria architettura;
- La necessità da parte di un Framework di utilizzare strumenti esterni alla propria implementazione;
- La possibilità di estensione e la flessibilità di un Framework rispetto all'altro, in modo da poter definire degli strumenti di sviluppo personalizzati.

Sulla base di questi aspetti può essere impostato un confronto pratico, che non faccia altro che confermare quanto descritto nel Capitolo IV.

Per quanto riguarda il Caso di Studio preso in esame, gli strumenti software utilizzati per lo sviluppo e gli ambienti di sviluppo sono i seguenti:

- Piattaforma linux *LAMP 2.0*;
- Ambiente IDE *Eclipse 3.1* con Plug-In *UML Omondo* per la realizzazione dei Class Diagramm delle Classi implementate;
- Web Container/Server *Apache 2.2.11.*

LAMP è l'acronimo di Linux Apache Mysql Php e indica una piattaforma per lo sviluppo di applicazioni web che prende il nome dalle iniziali dei componenti software con cui è realizzata.

La piattaforma è una delle più utilizzate a livello mondiale (es. Wikipedia e Facebook). Ognuna

delle applicazioni dalle quali è composta si predispone per un eccellente funzionamento in concomitanza con altre oltre a fornire tutte le classi principali e le librerie del linguaggio PHP. Per quanto concerne le librerie Zend Framework e cakePHP basta configurare il file php.ini aggiungendo la stringa include_path = "/opt/lampp/php526/extras/library".

Per l'ambiente di sviluppo IDE (Integrated Development Environment), la scelta è caduta sul progetto Open Source Eclipse 3.1, dopo un rapido confronto con l'antagonista NetBeans. Ha nettamente prevalso il primo, considerando il supporto fornito per lo sviluppo di Web Application basate appunto su Zend Framework e cakePHP, grazie al plug-in free di PHP Development Tools Project come nel caso di Struts o JSF (es. plug-in free di Exadel). Il secondo invece, fino alla versione 4.1 non ha ancora fornito tale supporto, introdotto a partire dalla versione 5 in fase di Beta Testing. Prendendo in esame tale versione, la semplicità ed il supporto offerti da Eclipse 3.1 sono comunque nettamente superiori.

Un integrated development environment (IDE), in italiano ambiente integrato di sviluppo, (conosciuto anche come *integrated design environment* o *integrated debugging environment*, rispettivamente *ambiente integrato di progettazione* e *ambiente integrato di debugging*) è un software che aiuta i programmatori nello sviluppo del software. Normalmente consiste in un editor di codice sorgente, un compilatore e/o un interprete, un tool di building automatico, e (solitamente) un debugger. A volte è integrato con un sistema di controllo di versione e con uno o più tool per semplificare la costruzione di una GUI. Alcuni IDE, rivolti allo sviluppo di software orientato agli oggetti, comprendono anche un navigatore di classi, un *analizzatore di oggetti* e un *diagramma della gerarchia delle classi*. Un IDE (Integrated Development Environment) è un software che permette la programmazione classica in un determinato linguaggio aiutandoci però con un'interfaccia grafica. Per chi è abituato a programmare in Java, C++, .NET può sembrare più comprensibile se si pensa agli ambienti di sviluppo per tali linguaggi come Eclipse, Dev-C++ o il Visual Studio.

L'ambiente è stato ulteriormente ampliato utilizzando il plug-in free Omondo per la realizzazione dei diagrammi UML, a partire dalle classi implementate. Tale strumento è stato utilizzato solo ed esclusivamente per generare i Class Diagramm a livello di implementazione.

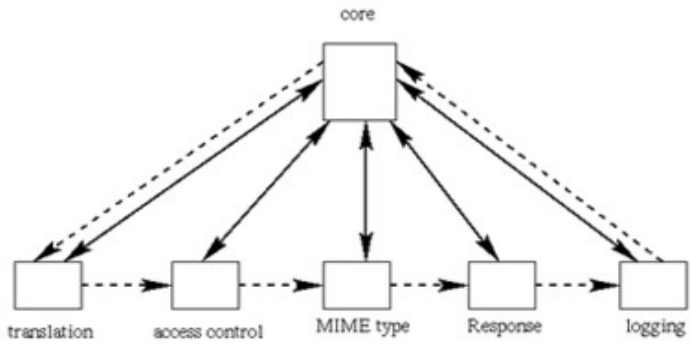
Per quanto riguardano il Web Container, la scelta è ovviamente caduta su progetto Open Source *Apache 2.2.11*, sicuramente tra i migliori per l'esecuzione di Web Application realizzate in ambiente PHP.

È altresì necessario specificare che, per quanto riguarda Zend Framework, è stata utilizzata la versione 1.1 mentre per cakePHP è stata scelta l'implementazione 1.0. in questo modo, è possibile confrontare due tecnologie realizzate da sviluppatori diversi, in quanto Zend Framework rientra tra i progetti dei sviluppatori di PHP mentre cakePHP no.

Apache HTTP Server, o più comunemente Apache (IPA pronuncia: /ə'pætʃi/), è il nome dato alla piattaforma server Web modulare più diffusa (ma anche al gruppo di lavoro *open source* che ha creato, sviluppato e aggiornato il software server), in grado di operare da sistemi operativi UNIX-Linux e Microsoft. Apache è un software che realizza le funzioni di trasporto delle informazioni, di internetwork e di collegamento, ha il vantaggio di offrire anche funzioni di controllo per la sicurezza come quelli che compie il proxy. Il Web Server apache presenta un'architettura modulare, quindi ad ogni richiesta del client vengono svolte funzioni specifiche da ogni modulo di cui è composto, come unità indipendenti. Ciascun modulo si occupa di una funzionalità, ed il

controllo è gestito dal core.

Figura 58 - Architettura Apache HTTP Server



Fonte: http://it.wikipedia.org/wiki/Apache_HTTP_Server

I moduli:

- Core - programma principale composto da un ciclo sequenziale di chiamate ai moduli.
- Translation - traduce la richiesta del client
- Acces Control - controlla eventuali richieste dannose
- MIME Type - verifica il tipo di contenuto
- Response - invia la risposta al client e attiva eventuali procedure
- Logging - tiene traccia di tutto ciò che è stato fatto

Il core suddivide la richiesta ai vari moduli in modo sequenziale, usando i parametri di uscita di un modulo come parametri di accesso per l'altro, creando così l'illusione di una comunicazione orizzontale fra i moduli (Pipeline software). Sopra il ciclo del core c'è un ulteriore ciclo di polling svolto da un demone che interroga continuamente le linee logiche da cui possono pervenire messaggi di richiesta.

2. Struttura della Web Application

Il primo aspetto presso in considerazione riguarda la struttura della Web Application, in termini di eventuali moduli che la compongono, oltre che delle pagine, delle immagini, degli eventuali fogli di stile (CSS) e di tutti gli altri elementi che ne fanno parte.

Per quanto riguarda l'implementazione con Zend Framework, l'architettura del sistema prevede un unico modulo al quale è ovviamente associato un file di configurazione (es. application.ini). I file componenti il progetto riguardano l'Applet Flash che implementa il player per

giocare e visionare il punteggio, i Templates che definiscono il layout delle pagine, le pagine ed i relativi contenuti, le immagini, il foglio di stile (es. style.css) per la formattazione della grafica. Infine, oltre alle librerie che contengono le classi costituenti l'architettura del Framework, sono utilizzati i seguenti gruppi di ulteriori librerie:

- Service – classi per interfacciare Flash;
- Db – contiene le classi relative ai driver per l'accesso alla Base di Dati con il DBMS MySql;
- Acl – contiene classi per la gestione delle sessioni e delle autenticazioni;

Per quanto riguarda l'implementazione con cakePHP, l'applicazione ha un'architettura che prevede quattro moduli distinti:

- Default – modulo principale che fornisce le funzionalità generiche di login e visualizzazione del punteggio;
- User – modulo contenente tutte le funzionalità disponibili ad un utente registrato;
- Blogger - modulo che fornisce le funzionalità per il moderatore.
- Admin – modulo che fornisce le funzionalità per l'amministratore.

Ciascuno di questi moduli ha il proprio file di configurazione all'interno del quale descrive il mapping delle action, i forwand, le pagine e le eccezioni.

Il modello “Default” viene avviato al momento della prima richiesta di un client. Nel momento in cui l'utilizzatore effettua un'operazione di login, in base al fatto che si tratti di un User, dell'Admin oppure del Blogger, viene eseguito lo “switch” verso il modulo corrispondente. Il passaggio dai moduli “User”, “Blogger” e “Admin” verso il modulo “Default” viene effettuata in seguito ad un'azione di logout.

Attraverso la scomposizione in moduli, non soltanto si ha il notevole vantaggio di raggruppare concettualmente le funzionalità legate ad un particolare utilizzatore ma anche quello di dover gestire tre file di configurazione differenti di dimensione ridotta, in luogo di un unico file di grandi dimensioni.

La struttura in termini di file prevede, similmente a Zend Framework, l'Applet Flash che implementa il gioco, i Templates per il layout delle pagine e il relativo file delle definizioni, le pagine ed i relativi contenuti, le immagini, il foglio di stile (es. style.css) per la formattazione della grafica. Tale file sono opportunamente separati in corrispondenza dei tre moduli.

Oltre alle librerie proprie di cakePHP, che includono il Plug-In Validatore, sono utilizzati anche i seguenti componenti core:

- Session – l componente session permette di avere un wrapper indipendente dai meccanismi di salvataggio intorno alle sessioni PHP;
- Acl – Il componente ACL fornisce una interfaccia facile da usare per delle access control lists su Database o file .ini;
- Auth - Il componente auth fornisce un sistema di autenticazione facile da usare usando una gran varietà di processi di autenticazione, come Controller callbacks, Acl e Object callbacks.

Rispetto all'implementazione in Zend Framework, si evince che il Tiles è incluso nella distribuzione di cakePHP ed è perfettamente integrato con quest'ultimo.

Per quanto riguarda il modulo “default”, si osserva che dalla homepage sia possibile accedere alla funzionalità di login, seguita dalla visualizzazione del punteggio. Si osservi che sono definiti due forword, associati ad un SwitchAction, che permettono di passare al modulo “user”, “blogger” oppure “admin”. Infine, è definita un'eccezione globale nel caso si verifichi un errore di accesso alla base di dati.

La struttura “User” prevede l'accesso alle seguenti funzionalità:

- Gioca;
- Visualizza punteggio;
- Operazione di logout, associata ad una SwitchAction, che permette di tornare al modulo “Default”.

Anche in questo caso è definita un'eccezione globale nel caso si verificchi un errore di accesso alla base di dati.

La struttura “Blogger” prevede l'accesso alle seguenti funzionalità:

- Gestione utente;
- Operazione di logout, associata ad una SwitchAction, che permette di tornare al modulo “Default”.

Anche in questo caso è definita un'eccezione globale nel caso si verifichi un errore di accesso alla base di dati.

Infine, l'architettura del modulo “admin”, fornisce i seguenti servizi:

- Gestione gioco;
- Operazione di logout, associata ad una SwitchAction, che permette di tornare al modulo “Default”.

Analogamente agli altri tre moduli, è definita un'eccezione globale che viene sollevata qualora si verifichi un errore di accesso alla base di dati.

Per completare l'analisi della struttura complessiva della Web Application, bisogna descrivere quelli che sono i package contenenti le classi ed il loro ruolo. Le due implementazioni hanno dei package comuni, in base al fatto che le funzionalità sono sostanzialmente le stesse. Ovviamente, le classi contenute all'interno di esse possono presentare delle differenze.

Un'ulteriore considerazione riguarda l'internazionalizzazione (i18n), supportata dalla Web Application attraverso la definizione di due Resoure Brudle. In particolare, ne è prevista una per la localizzazione di default, ossia l'Italia, è l'altro per un'ulteriore localizzazione supportata, in lingua Inglese. I file che contengono i messaggi sono perfettamente identici nelle due implementazioni, a meno di alcuni messaggi relativi agli errori di validazione, che hanno chiavi diverse in relazione al Framework utilizzato.

Infine, ci sono da motivare le scelte relative alla realizzazione del Model sostanzialmente diverso tra le due implementazioni.

In entrambi i casi, il sotto livello Data Access è completamente indipendente dagli altri ma le classi che lo implementano sono diverse tra le due implementazioni. Questo njon ne favorisce la portabilità, qualora si prenda si prenda in considerazione l'ipotesi di realizzare la Web Application con un ulteriore Framework.

Per quanto riguarda i sottolivelli di Business Logic ed External Interface, si è fatta una scelta diversa per evidenziare le due possibili modalità dio approccio che meglio si adattino ai due Framework.

3. Controller

Ciascuno dei due Framework fornisce le proprie classi di base che implementano la struttura del Controller di default. Seppur in maniera completamente diversa, è possibile intervenire in cakePHP e Zend Framework per poter personalizzare tali classi, in modo da aggiungere particolari funzionalità che vanno applicate durante la fase di comunicazione tra Model e View. Nel caso di studio in esame, si è reso necessario l'intervento sul Controller, per poter gestire due situazioni particolarmente delicate:

- La gestione delle sessioni non concluse correttamente dall'utilizzatore;
- La possibilità di accesso a pagine protette senza necessaria operazione di login.

La prima problematica è stata risolta allo stesso modo nelle due implementazioni, in quanto è più legata alla gestione delle ACL che non alle classi di uno specifico Framework. Il secondo problema è stato affrontato in maniera completamente diversa, considerando le differenze architetturali che assume il Controller in cakePHP e Zend Framework.

3.1 Gestione delle Sessioni

Uno dei principali problemi da cui sono affette tutte le Web Application è la gestione delle sessioni che non vengono concluse correttamente da un utente. Nel caso specifico, quando l'utilizzatore, User o Blogger che sia, effettua il login al sistema, vengono memorizzate all'interno della base di dati informazioni relative all'accesso (es. la data e l'ora di ingresso, l'indirizzo IP e il numero di tentativi effettuati). Inoltre, viene settato un flag nel database che indica che l'utente è online, in modo che un tentativo fraudolento di accesso da parte di un secondo utente, che disponga dei dati del primo, venga ovviamente bloccato. Nell'effettuare correttamente la fase di logout, le informazioni di accesso vengono aggiornate settando opportunamente la data e l'ora di uscita dal sistema, nonché, viene azzerato il flag nel database per segnalare che l'utente è offline. Ci sono moltissime situazioni in cui l'utente chiude direttamente il browser, senza completare correttamente il logout. In questo caso è necessario intervenire per ristabilire la coerenza delle informazioni contenute nel database rispetto alla realtà. L'evento di chiusura del browser non può essere intercettato dal server, sul quale è in esecuzione la Web Application, ma ovviamente solo il client. Il linguaggio di scripting lato client, che permette di eseguire elaborazioni su quest'ultimo è ovviamente Java Script, il quale però non mette a disposizione la possibilità di intercettare un semplice evento come la chiusura di una finestra del browser, a meno che questa operazione non venga eseguita sempre mediante codice. Per questo ed altri motivi, è stato introdotto il concetto di sessione, mediante la quale si stabilisce un legame un legame tra l'utente che utilizza

l'applicazione ed il server, su cui quest'ultima è in esecuzione. Alla sessione è associato un concetto di timeout, per cui se non si rivela attiva da parte dell'utente fino allo scattare di quest'ultimo, ne viene eseguita l'invalidazione. È in corrispondenza di questo evento, che si può intervenire effettuando le operazioni che garantiscono la coerenza delle informazioni nel sistema. Da qui, la necessità di realizzare un listener, che allo scadere del timeout di sessione, intervenga per poter permettere allo sviluppatore di effettuare delle elaborazioni necessarie.

Per entrambe le implementazioni è stata realizzata la classe InactivityListenr, nell'ambito della quale il metodo sessionCreated() ha il compito di settare il timeout di sessione e sessionDestroyed() ha l'onere di cambiare lo stato del flag nel database, per segnalare che l'utente è offline. Quest'ultimo metodo viene appunto invocato nel momento in cui scatta il timeout, in quanto l'utente non sta più generando attività sul server e presumibilmente ha chiuso il browser senza eseguire correttamente il logout.

Questo aspetto non può essere considerato un termine di confronto tra i due Framework, in quanto le classi in gioco non sono specifiche di cakePHP e Zend Framework, ma rientrano nell'architettura in generale.

3.2 Protezione Pagine

La seconda problematica in cui è necessario l'intervento del Controller, riguarda la possibilità che un utilizzatore voglia accedere alle pagine protette dell'applicazione, senza aver eseguito correttamente l'operazione di login. Ovviamente, il sistema deve intercettare un'azione di questo tipo, redirezionando l'utente alla homepage, nella quale è costretto ad eseguire l'acceso. Dall'analisi del problema, si evince che l'intervento deve essere eseguito per ciascuna richiesta che arrivi al client, in modo da sincerarsi che l'utilizzatore sia correttamente loggato, qualunque sia la pagina a cui vuole accedere, escluse le pagine relative alla visualizzazione del punteggio.

I due Framework prevedono una risoluzione della problematica completamente differente, data la diversa struttura del Controller, i punti in cui è possibile estendere quest'ultimo ed il differente Ciclo di Vita a cui è sottoposta una generica richiesta.

Per quando riguarda Zend Framework, è noto che la gestione di una richiesta del client viene eseguita attraverso sei file distinti. In corrispondenza di ciascuno di essi, può essere utilizzato un listner che permette di eseguire delle operazioni prima e dopo l'esecuzione della fase stessa. La soluzione adottata precede la realizzazione della classe LoginbChek che implementa l'interfaccia ed esegue i controlli necessari prima della fase di Render Response. Essa prevede che il metodo afterPbase() sia vuoto, mentre all'interno del metodo beforePbase() ci sia il controllo che l'utilizzatore, User o Blogger che sia, risulti correttamente loggato oppure la richiesta si riferisca a pagine non protette da login. Nel caso di esito positivo, la navigazione procede normalmente, altrimenti viene ricavato un riferimento al NavigationHandler, il quale permette di spostare il flusso dell'applicazione verso la homepage. Le informazioni relative all'evento verificatosi, ossia l'esecuzione della fase di Render Response, sono contenute nella classe PbaseEvent.

La soluzione con cakePHP è completamente diversa, in quanto il ciclo di vita di una richiesta è certamente più semplice e prevede esclusivamente l'intervento della classe RequestProccessor. Per questo motivo, dovendo eseguire delle operazioni di controllo prima che la richiesta venga elaborata, la soluzione prevede di realizzare la classe CustonnRequestProcessor che estende la

classe base RequestProcessor. In realtà, osservando il codice, si evince che la classe estesa è TilesRequestProcessor che comunque deriva da RequestProcessor. È necessaria la derivazione della prima, in quanto l'utilizzo di Tiles per il layout prevede l'intervento del proprio Controller. Il metodo che contiene il codice delle operazioni da eseguire è processPreporcess(), il quale viene appunto eseguito prima di iniziare la gestione della richiesta. Se l'utilizzatore risulta correttamente loggato, la navigazione prosegue normalmente altrimenti si viene redirezionati alla homepage.

È da osservare che la classe CustonRequestPreocessor è registrata nei moduli “Admin” e “Blogger” e non nel modulo “User”, in quanto quest'ultimo non contiene pagine protette, per cui può utilizzare il RequestProcessor di base.

4. View

La realizzazione dell'interfaccia utente (UI) è sicuramente uno degli aspetti nei quali si evidenzia la netta differenza tra Zend Framework e cakePHP, in quanto il primo è strettamente orientato allo sviluppo del Presentation Layer dell'applicazione. Per quanto riguarda la produzione delle pagine PHTML, l'implementazione in cakePHP ha previsto l'utilizzo delle librerie di tag, Core e Formatting, oltre alla libreria cakePHP HTML. La libreria Formatting viene utilizzata sopratutto per la visualizzazione dei messaggi all'utente, prelevandoli dal Resource Bundle. Infine, la libreria cakePHP HTML contiene tutti i tag necessari alla costruzione dei form per l'immissione dei dati, cosi comi i tag per la visualizzazione dei messaggi di errore. La formattazione delle pagine ha previsto prevalentemente l'uso dei tag HTML, in particolar modo l'utilizzo delle tabelle.

L'implementazione in Zend Framework ha previsto l'utilizzo delle due librerie proprie del framework, ossia Core e d HTML, la prima per quanto riguarda aspetti avanzati come l'uso dei validatori mentre la seconda per la costruzione dei form, la visualizzazione dei messaggi e la formattazione del contenu8to delle pagine.

Con entrambi i Framework non è stato previsto l'utilizzo di Tiles essendo incluso solo nella distribuzione di cakePHP.

Inoltre, sfruttando l'enorme potenzialità di Zend Framework, sono stati realizzati dei Costom Componets ed il relativo Tag Library Desciptor, in modo da utilizzare questi componenti all'interno delle pagine con un corrispondente tag.

Infine, l'implementazione mediante cakePHP ha previsto l'utilizzo dei formbeans per l'acquisizione dei dati dai form, mentre con Zend Framework è bastato utilizzare i backing beans e le relative proprietà.

4.1 Le Librerie Standard

Per quanto riguarda le librerie di tag standard che i due framework mettono a disposizione, si può affermare che nessuno dei due prevalga sull'altro. Sia Zend Framework che cakePHP hanno i propri vantaggi e svantaggi, per quanto concerne i tag che possono essere utilizzati per definire gli elementi costruttivi di ciascuna pagina PHTML.

È da osservare, però, che mentre Zend Framework utilizza solo ed esclusivamente le proprie librerie, Core ed HTML, invece con cakePHP si preferisce utilizzare altre librerie.

Inoltre, c'è una differenza in termini di configurazione delle librerie da adottare, in quanto cakePHP non prevede alcun tipo di modifica nel Deployment Descriptor, mentre Zend Framework prevede l'elenco delle librerie che verranno utilizzate nell'applicazione.

4.1.1 I form: contenuto e layout

Per quanto riguarda la costruzione dei form, entrambi i Framework dispongono di tag equivalenti, ovviamente con sintassi diversa. Un limite di Zend Framework è, però, rappresentato dalla mancanza di un tag che permetta di selezionare un file all'interno del File System locale, per poi trasferirlo sul server. Tale aspetto rappresenta un vantaggio per cakePHP.

Da un punto di vista puramente grafico, si evince che non sussiste alcuna differenza tra le due implementazioni, anche se il procedimento adottato per usufruire di questa funzionalità è ben diverso tra i due Framework.

4.1.2 Costruzione condizionale dei contenuti

Un'ulteriore notevole differenza riguarda la costruzione condizionale di una pagina, che consiste nel fare in modo che un certo contenuto sia visualizzato o meno a seconda del verificarsi di opportune condizioni.

Per quando riguarda cakePHP, l'unica possibilità è quella di utilizzare l'attributo di cui è dotato ciascun tag della libreria HTML. Tale attributo può assumere valore booleano che indica appunto se il tag ed il relativo contenuto debba essere reindirizzato o meno.

Il Framework cakePHP, invece, dispone di una libreria all'interno della quale ci sono una serie di tag che permettono di definire dei costrutti condizionali e ciclici.

4.1.3 DataTable

Nell'ambito della Web Application realizzata, si è più volte posto il problema di visualizzare un elenco di elementi in forma tabellare, come nel caso della visualizzazione del punteggio e degli utenti registrati.

L'implementazione cakePHP ha previsto l'utilizzo di potentissimi tag, i quali permettono di specificare la lista degli elementi da visualizzare ed eventualmente le informazioni di formattazione. Inoltre, ciascuna colonna viene definita con dei tag, all'interno del quale ne viene specificato un'eventuale intestazione e il relativo contenuto.

Nel caso di Zend Framework, si è reso necessario l'utilizzo di un tag che permette di realizzare un vero e proprio ciclo for all'interno di uan pagina. Esso permette di specificare la lista degli elementi da scorrere attraverso l'attributo items ed un eventuale variabile che individua l'oggetto corrente.

4.2 Custon Components

Il Framework Zend Framework mette a disposizione una potenzialità notevole che consiste nel poter estendere le classi dei componenti della UI, per implementare componenti personalizzabili,

molto più potenti, da utilizzare nella propria interfaccia utente.

Nella Web Application in questione, non sono stati realizzati i seguenti componenti :

- DataSelectComponent – componente che visualizza tre drop-down list contenente giorni, mesi ed anni, per la selezione di una data (es. immissione data di nascita). Inoltre permette di visualizzare un eventuale messaggio di errore;;
- DDListComponent – componente che visualizza semplicemente una drop-down list ma con la possibilità di specificare un'etichetta e visualizzare un eventuale messaggio di errore;
- TimeSelectComponent – componente che permette la selezione di una durata attraverso due drop-down list che contengono minuti e secondi. Inoltre, permette di visualizzare un'etichetta al fianco delle drop-down list.

il Framework cakePHP non mette assolutamente a disposizione un meccanismo che permetta di implementare dei componenti personalizzati come in Zend Framework. La realizzazione dell'interfaccia utente può essere svolta utilizzando tutti quelli che sono i tag della libreria HTML, senza possibilità di estensione. Tale limite si è evidenziato sopratutto per alcuni campi necessari nei form (es. la data di nascita).

Con Zend Framework, queste necessità particolari sono state risolte attraverso la realizzazione di corrispondenti Custom Componets, mentre nel caso di cakePHP, si è preferito utilizzare dei semplici campi di testo. Si sarebbe potuto pensare di servirsi di tag relativi alle drop-down list, ma in quel caso sarebbe stato necessario far gestire alla action ricevente, l'operazione di comporre valori separatamente ricavati dalla request per ottenere un unico valore corrispondente. L'utilizzo di un semplice campo di testo ha permesso di avvalersi di una particolare regola di validazione, prevista nel Plu-In Validator, relative alle date. In questo modo, si può comunque effettuare un opportuno controllo per verificare che l'utilizzatore abbia immesso una data valida.

4.3 I formbean

In riferimento all'acquisizione dei dati contenuti nel form ed inviati all'utente, il Framework cakePHP introduce il concetto di formbean, il quale altro non è che una particolare classe le cui proprietà sono destinate a contenere i valori dei campi form stesso. Nel momento in cui viene inviata una richiesta, è compito del RequestProcessor popolare il formbean ed effettuare la validazione dei dati in esso contenuti, per poi passarlo alla action che dovrà occuparsi di gestire la richiesta.

In generale, un forbean può essere di due tipi:

- Statico – realizzato attraverso una classe che estende la classe ActionForm oppure una delle sue derivate e poi dichiarato nel file di configurazione;
- Dinamico – dichiarato direttamente all'interno del file di configurazione, senz ala necessità di implementare una classe corrispondente.

Tipicamente, si preferisce utilizzare i formbean dinamici, anche se sussistono alcune situazioni in cui si rende necessaria l'implementazione delle classi per quelli statici.

Anche nel caso della Web Application realizzata, si è fatto uso di entrambe le tipologie di

formbean, dinamici li dove possibile e statici li dove fossero necessari per un certo motivo.

Rispetto a cakePHP, il Framework Zend Framework ha una gestione completamente diversa dei form. Dal punto di vista puramente grafico, questi ultimi sono sostanzialmente identici a meno della possibilità di utilizzare i Custon Components. Per quanto le operazioni di acquisizione dei dati, quest'ultimo non prevede assolutamente il concetto di formbean. I campi di ciascun form sono associati alle proprietà di backing beans, così come la pressione di un bottone per l'invio dei dati non prevede l'esecuzione di un action esterna, ma bensì di uno dei metodi di backing. Tutto ciò ha il vantaggio di non dover utilizzare un oggetto intermedio, il formbean appunto, che abbia il compito di trasferimento dei dati dall'utilizzatore alla elaborazione da eseguire.

4.4 Gioco Flash

Il player adottato per giocare è implemntato in Flash. Attraverso il file di inizializzazione jlgui.ini, esso mette a disposizione la possibilità di modificare una serie di opzioni.

Dal punto di vista dell'interfaccia User, viene aperta una finestra all'interno della quale viene caricato l'applet. Da qui, l'User ha la possibilità di interagire con quest'ultimo inserendo il proprio nome e scegliendo l'ambiente. È da osservare che, con questo approccio, l'utilizzatore ha comunque la possibilità di proseguire la navigazione all'interno del sistema, proseguendo dopo la fine del gioco stesso.

L'unico limite del player è che bisogna aver finito di giocare per uscire dal sistema.

4.5 Internazionalizzazione (i18n)

Attraverso il meccanismo dell'Internazionalizzazione (I18N), è possibile visualizzare il testo presente nella Web Application in lingue diverse, in relazione alla localizzazione geografica da cui si collega l'utente oppure sulla base delle impostazione della lingua del browser. Nel caso dell'applicazione realizzata, si è previsto il supporto per la lingua italiana, considerata di default, e per la lingua inglese. In questo modo, senza la necessità di dover sviluppare due volte la medesima Web Application, è possibile comunque garantire l'utilizzo di quest'ultima a persone di lingua differente. Per quanto riguarda questo aspetto, i framework cakePHP e Zend Framework sono perfettamente identici, in quanto forniscono tale supporto allo stesso modo, anche in virtù del fatto che questo concetto è proprio del linguaggio PHP e non solo del linguaggio Java.

In entrambe le implementazioni sono stati realizzati due Resource Bundle contenenti i messaggi da visualizzare nelle due lingue. Ciascuna di esse è costituita da una serie di coppie “key-message”, attraverso le quali è possibile visualizzare un certo messaggio semplicemente specificando la chiave ad essa associata.

In Zend Framework La struttura della vostra applicazione IT dovrebbe assomigliare alla seguente:

1. /application/
2. /languages/it/
3. /library/

Invece, sempre in Zend Framework, la struttura della vostra applicazione EN dovrebbe assomigliare alla seguente:

4. /application/
5. /languages/en/
6. /library/

In cakePHP la struttura della vostra applicazione IT / EN dovrebbe assomigliare alla seguente:

1. locale/eng/LC_MESSAGES/default.po (English)
2. locale/fre/LC_MESSAGES/default.po (French)

Ora che si sa dove saranno inseriti i file, di entrambi i Framework, si deve procedere a realizzare una nuova funzione _init* all'interno della nostra classe Bootstrap.

Entrambi i Framework hanno previsto la Resource Bundle, con una leggera differenza. Nel caso di Zend Framework è stato necessario specificare la localizzazione (locale) di default e quella supportata, oltre la nome del file contenete i messaggi.

```
1.  protected function _initLanguage()
2.  {
3.  $localeValue ='it';
4.
5.  $locale = new Zend_Locale($localeValue);
6.  Zend_Registry::set('Zend_Locale',$locale);
7.  $translationPath=dirname(APPLICATION_PATH).
8.  DIRECTORY_SEPARATOR.'languages'.DIRECTORY_SEPARATOR.$localeValue;
9.
10. $translate = new Zend_Translate('array',$translationPath,$localeValue);
11.
12. Zend_Registry::set('Zend_Translate',$translate);
13. Zend_Validate_Abstract::setDefaultTranslator($translate);
14. Zend_Form::setDefaultTranslator($translate);
15. }
```

Invece, cakePHP ha previsto semplicemente la dichiarazione del Resource Bundle.

```
16. public function setLang($language = 'en') {
17. $this->L10n = new L10n();
18. $this->L10n->get($language);
19. //Configure::write('Config.language', $language);
```

```
20. $_SESSION['Config']['language'] = $language;
21. }
```

Come si può osservare, in entrambi i file le chiavi sono perfettamente identiche mentre i messaggi sono in lingua diversa. Inoltre, i file stessi hanno praticamente il medesimo nome, ossia AR_IT e AR_EN, a meno del file in lingua inglese che ha il suffisso “en”. Tipicamente, il file che non ha alcun suffisso è quello relativo alla lingua di default mentre i file associati alle altre lingue devono avere lo stesso nome del primo ma seguito da un suffisso che identifica la lingua.

Per quanto riguarda l'utilizzo dell'internazionalizzazione all'interno delle pagine dell'applicazione, le librerie di entrambi i Framework mettono a disposizione un tag che permette di specificare quale sia il Bundle del quale prelevare il messaggio.

Un limite legato all'internazionalizzazione riguarda il fatto che, nel caso in cui la navigazione tra le pagine non sia definita attraverso collegamenti ipertestuali ma attraverso elementi grafici, non è possibile visualizzare un elemento al posto di un altro in base alla differente lingua di accesso.

5. Validazione

La validazione dei dati immessi nei form da parte dell'User è uno degli aspetti su cui si differenziano maggiormente i Framework a confronto. La differenza sostanziale risiede nel fatto che Zend Framework mette a disposizione un proprio meccanismo per la validazione e per la sua estensione mentre cakePHP, nella maggior parte dei casi, si affida al plug-in Validator anche se è previsto un metodo di validazione interno al Framework. Le scelte fatte nelle due implementazioni sono fondate sulla riusabilità e sulla rapidità di sviluppo, sfruttando li dove possibile tutti gli strumenti a disposizione.

5.1 Zend Framework Custon Validator

Zend Framework mette a disposizione soltanto tre validatori standard che permettono di valutare se un certo valore ha una certa lunghezza entro un minimo ed un massimo prefissati oppure se il medesimo valore rientra in un certo range. Per quanto riguarda l'obbligatorietà di un campo, quest'ultima viene specificata mediante l'attributo required del tag associato al componente in questione.

Considerando il numero limitato di controlli standard che possono essere eseguiti, il Framework mette a disposizione due meccanismi per la realizzazione di ulteriori validatori.

L'implementazione di Zend Framework ha previsto lo sviluppo di tre validatori:

- EmailValidator – permette di valutare se il formato di un indirizzo e-mail sia o meno coretto (es. vincenzoiovino@antonioranieri.it)
- SeqCifreValidator – per valutare se un certo valore è rappresentato da una sequenza di cifre (es. CAP);
- TelefonoValidator – per verificare se un numero di telefono abbia un formato valido (es. prefisso / numero).

Nel caso di esito negativo della validazione, il messaggio di errore viene prelevato dal Resource Bundle.

5.2 Defautl Validator cakePHP

Per poter effettuare la validazione dei contenuti di un form, il Framework cakePHP mette a disposizione un proprio meccanismo che è strettamente legato ai formbean associati a questi ultimi.

Per quanto riguarda le regole di validazione, queste ultime non sono state assolutamente modificate e non ne sono state create delle ulteriori, in quanto il cakePHP mette a disposizione le funzionalità di validazione più comuni, tra cui anche il controllo di correttezza di un indirizzo e-mail. Le principali validation-ruls utilizzate sono le seguenti:

- VALID_NOT_EMPTY – controlla che un campo non sia vuoto, in quanto è obbligatoria l'immisione di un valore;
- VALID_E-Mail – verifica che un valore abbia il formato corretto di un indirizzo e-mail (es. andreabruno@angelodecaro.it)
- VALID_NUMBER – valuta se il valore di un campo rispetti una lunghezza minima prefissata;
- VALID_YEAR – verifica che un campo contenga una data corretta, secondo un certo formato (es. gg / mm / aaaa).

La disponibilità di queste regole non ha reso necessaria nessuna estensione, realizzando delle funzioni che implementano ulteriori regole di validazione. Da questo punto di vista cakePHP si può considerare superiore a Zend Framework, tenendo conto che i validatori standard di quest'ultimo sono soltanto tre.

6. Model

Il Model può essere considerato come il cuore dell'applicazione realizzata, in quanto è costituito da tutte quelle classi che definiscono la business-logic. Così come anticipato in precedenza, l'implementazione in Zend Framework ha previsto lo sviluppo dei Backing Beans e dei relativi metodi che realizzano le funzionalità del sistema.

Con cakePHP si è resa necessaria la realizzazione delle action, per cui le classi della Backing Beans risultano notevolmente più “leggere”. Ovviamente, la logica che è alla base delle funzionalità è praticamente la stessa, con la differenza che in Zend Framework è completamente incapsulata nei metodi dei Backing Beans mentre in cakePHP è distribuita fra gli oggetti cella Business-Logic e le action. Queste ultime prevedono tutto ciò che riguarda l'accesso agli elementi del framework, mentre i primi eseguono le operazioni previste dall'applicazione. Infine, sono implementate alcune classi comuni tra le due implementazioni e che hanno i seguenti ruoli:

- Una serie di eccezioni che possono essere sollevate da particolari elaborazioni;
- Una serie di classi che gestiscono le operazioni di accesso alla Base di Dati.

6.1 Classi Exception

L'applicazione prevede alcune funzionalità nell'ambito delle quali si possono verificare delle condizioni di errore che vanno opportunamente gestite. Uno dei metodi “eleganti” è l'utilizzo del meccanismo delle eccezioni messo a disposizione del linguaggio PHP. Per questo motivo, sono state realizzate le seguenti classi che estendono la classe base Exception:

- PasswordErrata – eccezione che si verifica nel caso in cui un utilizzatore abbia tentato un accesso al sistema sbagliando la password;
- UsernameErrato – eccezione sollevata nel momento in cui un utilizzatore tente di effettuare il login con un username non registrato;
- UsernameEsistente – eccezione che si verifica in fase di registrazione dell'User, quando il Blogger ha specificato un username già presente nella Base di Dati;
- RuoloBloccato – eccezione sollevata quando un utilizzatore tenta di eseguire il login con username e password corretti, ma il suo account risulta bloccato a causa di tentativi di accesso errati in precedenza;
- RuoloLoggato – eccezione che viene sollevata nel momento in cui l'utilizzatore tenta di accedere al sistema ma risulta già loggato.

Queste classi sono perfettamente identiche tra le due implementazioni ed inoltre sono praticamente tutte una semplice estensione della classe Exception alla quale non aggiungo alcuna funzionalità. Lo scopo della loro realizzazione è sopratutto concettuale, per poter distinguere all'interno del codice le eccezioni di tipo diverso.

Soltanto la classe PasswordErrata è leggermente diversa dalle altre, in quanto ha la proprietà idRuolo che permette di individuare l'utilizzatore che ha immesso lo username corretto ma la password sbagliata, per cui bisogna tenere conto dei tentativi di accesso errati e bloccando nel caso in cui sia necessario.

6.2 Classi di interfacciamento con la Base di Dati

Nella realizzazione di un sistema software che debba funzionare in locale, ci sono tipicamente degli oggetti per i quali va garantita la persistenza attraverso l'utilizzo delle Base di Dati. Per poter gestire le operazioni di lettura e scrittura nel database, sono previste due possibili soluzioni:

- Ogni oggetto gestisce la propria persistenza in maniera autonoma, prevedendo dei metodi che accedano alla Base di Dati;
- Ogni oggetto delega la gestione della propria persistenza ad un'altra classe che mette a disposizione un serie di metodi di accesso al Database.

Con lo scopo di disaccoppiare le funzionalità della Business-Logic delle operazioni di accesso alla Base di Dati, generalmente si preferisce adottare la seconda soluzione. Ciò è stato fatto anche nella Web Application realizzata, implementando il sottolivello Data Access del Model attraverso un'ulteriore insieme di classi, una per ciascuna corrispondente classe della Business-Logic.

Ogni classe utilizzata prevede una serie di proprietà private che rappresentano la connessione (conn), una generica query (stnt), un resultSt (rs) ed infine la stringa di interrogazione (sQuery).

Per garantire l'utilizzo dell'applicazione con qualsiasi tipo di DBMS (es. MySQL, MS SQL Server, MS Acces) e facilitarne il passaggio dall'uno all'altro, è stata realizzata la classe astratta InfoDBMS che non ha metodi ma esclusivamente le seguenti proprietà:

- driver;
- comeString;
- user;
- userPassword:

la classe che si interfaccia con il Database utilizza le proprietà della classe suddetta, per impostare i parametri di accesso con i quali stabilire la connessione alla Base di Dati. In questo modo, è possibile modificare i campi statici all'interno della classe InfoDBMS, senza alcun intervento all'interno delle altre classi. Un ulteriore vantaggio è dato dal fatto che la configurazione è completamente centralizzata in un'unica classe.

Per quanto riguarda le informazioni di accesso alla Web Application è stato realizza la classe AcessoDB, la quale dispone dei metodi per inserire, aggiornare e leggere un record nelle tabelle degli accessi.

Le informazioni che riguardano l'Admin sono gestite attraverso la classe AdminDB che mette a disposizione i metodi per eseguire l'operazione di login, per leggere ed aggiornare un record. In particolare, il metodo di login() prevede in ingresso le username e la password dell'Admin e ne verifica la validità, con la possibilità di sollevare una delle eccezioni introdotte in precedenza, se si verifica una particolare condizione di errore. Ovviamente, questo metodo è invocato dalla funzionalità della business-Logic che permette di effettuare il login e il suo scopo è solo ed esclusivamente di accesso allal Base di Dati. I metodi login() e blocca() prevedono in ingresso l'identificativo dell'Admin di cui leggere le informazioni oppure di bloccare, mentre aggiorna() riceve un oggetto Admin.

In maniera del tutto analoga è definita la classe UserDB che permette di garantire la persistenza delle informazioni dall'User. Esso mette a disposizione i metodi tipici per la lettura, l'aggiornamento, l'inserimento e l'eliminazione di un record. I metodi leggi() ed elimina() prevedono in ingresso l'identificativo dell'User di cui leggere le informazioni oppure eliminare, mentre i metodi aggiorna() ed inserisci() prevedono un oggetto User. È da osservare che l'operazione di inserimento è piuttosto generica, in quanto essa viene utilizzata nelle funzionalità di registrazione della Business-Logic, certamente più complessa. Il metodo login() opera allo stesso modo come per la classe AdminDB. La gestione della permanenza di tutto ciò che è strettamente legato alla Gioco in Flash viene eseguita mediante le tre seguenti classi.

6.3 Business-Logic e funzionalità di sistema

Considerato le due implementazioni con Zend Framework e cakePHP, le classi che costituiscono le Business-Ligic sono caratterizzate dalle seguenti differenze:

- Con Zend Framework a tutti gli effetti i Backing Beans, per cui la maggior parte dei metodi restituiscono gli outcome per la navigazione essendo invocati direttamente dalle pagine PHTML, con cakePHP vengono utilizzate all'interno delle action per cui

non devono assolutamente occuparsi della navigazione;

- Con Zend Framework i metodi accedono agli oggetti del Framework, così come alle variabili di sessione e di ciascuna richiesta, con cakePHP tali operazioni vengono eseguite dalle action in modo che gli oggetti dalla Business-Logic siano disaccoppiati dal Framework stesso.

Facendo riferimento alle elaborazioni eseguite per ciascuna funzionalità, basta osservare che nel caso di Zend Framework sono concentrate nei metodi delle classi stesse, mentre nel caso di cakePHP sono distribuite tra questi ultimi e le action. Inoltre, è da sottolineare che dal punto di vista concettuale sono sostanzialmente identiche, tenendo comunque conto che la Web Application è la medesima.

Una precisazione è doverosa per quanto riguarda l'inserimento e la lettura degli oggetti del sistema all'interno delle variabili di sessione. Queste ultime sono ampiamente utilizzate per memorizzare alcuni oggetti della Business-Logic che contengono le informazioni associate all'utilizzatore corrente, di cui bisogna mantenere lo stato durante la navigazione. Per poter gestire le operazioni di lettura e scrittura in sessione, le classi sono state dotate di alcuni metodi statici che ovviamente necessitano dell'accesso all'istanza della classe HttpSession. Questo tipo di operazione non comporta nulla sull'implementazione con Zend Framework, mentre sembrerebbe introdurre una contraddizione nell'implementazione con cakePHP, perchè in questo modo si introduce un legame con il Framework. La scelta è stata fatta per rendere più “snello” il codice all'interno delle action ma ciò non toglie che, a meno dei metodi statici suddetti, le classi della Business-Logic dell'implementazione in cakePHP sono assolutamente indipendenti dal Framework e riutilizzabili con Framework diversi.

6.4 I Backing Beans di Zend Framework

Nell'implementazione realizzata utilizzando Zend Framework, tutte le classi che costituiscono il Model sono dichiarate all'interno di un file di configurazione come backing beans. Questo tipo di approccio garantisce due vantaggi fondamentali:

- Ciascun componente dell'interfaccia utilizzatore può essere associato ad una proprietà di un backing bean attraverso il value-biding, in modo tale che il form a cui appartiene il componente viene trasmesso, il valore che assume il componente viene trasferito nella corrispondente proprietà a cui è legato;
- Utilizzando gli eventi, action e value-change, è possibile invocare direttamente d aun componente un metodo di backing bean, attraverso il method-binding, in modo da poter effettuare delle elaborazioni sui dati contenuti nel backing bean stesso.

In base a quanto detto, è evidente la netta differenza rispetto al Framework cakePHP, per quanto riguarda l'acquisizione dei dati dai form e l'esecuzione delle elaborazioni su di essi. Infatti, cakePHP utilizza formbean le cui proprietà sono associate ai campi di un formo in modo che, uan volta che quest'ultimo sua stato trasmesso, i valori dei campi stessi vengono copiati nel formbean corrispondente. Sarà compito della action che riceve i formbeans copiare tali dati nelle corrispondenti proprietà di un oggetto del Model, per poterne eseguire una qualsiasi elaborazione. Zend Framework evita questo passaggio intermedio facendo in modo che i valori dei campi dei form vengano copiati direttamente nelle corrispondenti proprietà di un backing bean.

Per quanto riguarda l'avvio delle elaborazioni da una pagina PHTML, cakePHP prevede il concetto di action che viene invocato alla pressione di un bottone oppure al click di un link. La action riceve un formbean con i dati da elaborare, li trasferisce nelle corrispondenti proprietà di un oggetto del Model e su di esso invoca un metodo per eseguire l'elaborazione richiesta. Zend Framework permette di avviare direttamente da una pagina PHTML un metodo di backing bean, ossia di un oggetto Model, nel quale ci saranno automaticamente i dati da elaborare. Anche in questo caso, si evita un passaggio intermedio che riguarda l'utilizzo delle action di cakePHP

7. Plug-In

Il Framework cakePHP prevede un particolare meccanismo per poter estendere le sue funzionalità legato al concetto di Plug-in. Ovviamente, lo sviluppatore ha la possibilità di realizzare uno proprio mediante una classe che implementi l'interfaccia Plug-in. Il loro uso può risultare utile quando ch'è la necessità di eseguire una certa operazione una solo volta all'avvio dell'applicazione. Proprio questa modalità di utilizzo è stata adottata nell'implementazione con cakePHP, realizzando il Plug-in StarupManager che, una volta avviato, crea le liste di item che verranno usate in tutte le drop-down list dell'applicazione (es. quella relativa alle province, agli statti e così via).

Nell'implementazione con cakePHP, non essendo disponibile uno strumento come il Plug-In di Zend Framework, è stata realizzata la classe per la quale ne è stata dichiarata un backing bean.

8. Navigazione

La navigazione all'interno dell'applicazione è certamente uno degli aspetti che differenzia maggiormente i due Framework, in quanto essi fruttano due meccanismi diversi che comunque possono essere paragonati l'uno all'altro.

Zend Framework si basa sul concetto della navigation-rules, all'interno di ciascuna delle quali va definita la pagina di partenza della specifica “regola” e vanno definiti uno o più navication-case che specificano, sulla base dell'outcome rinvenuto dal NavigationHandler, quale sia la pagina verso la quale proseguire la navigazione.

In base a quanto detto, ogni regola prevede:

- La pagina che rappresenta il punto di partenza di riferimento;
- Uno o più pagine destinazione che è possibile raggiungere dalla suddetta pagina, ciascuna delle quali è distinta da un differente outcome.

Ciò vuol dire che, nel momento in cui viene invocato un metodo di backing bean da una certa pagina PHTML, quest'ultimo dovrà restituire come parametri di uscita una stringa che presenti l'outcome di navigazione. Sulla base di quest'ultimo, il NavigationHandler esegue una ricerca nel file di configurazione e determina verso quale pagina deve proseguire la navigazione. Nel caso della Web Application implementata si è evidenziata la non perfetta integrazione tra Zend Framework e Tiles. Infatti,, se si utilizza questo framework per la definizione del layout, non è possibile utilizzare le regole di navigazione come spiegato in precedenza, ma è necessario definire un'unica regola, senza alcuna pagina associata, all'interno della quale ci sono i navigation-case.

Il motivo di questo limite è dato dal fatto che nel Deployment Decrictor web.xml dell'applicazione realizzata con Zend Framwork, l'utilizzo di Tiles è specificato attraverso la dichiarazione di un'ulteriore servlet, TilesServlet, che viene avvia immediatamente dopo la FacesServlet e che intercetta per prima le richieste verso le pagine, interferendo con il NavigationHandler.

Se in Zend Framework la definizione della navigazione viene dichiarata in termini generali, con cakePHP è descritta in maniera più specifica e localizzata, all'interno delle action mediante il concetto di forward. Infatti, ciascuna action prevista dall'applicazione può avere un unico metodo execute() oppure più metodi, nel caso di un Dispath Action, che comunque restituiscono un oggetto del tipo ActionForword. Ciascun forword specifica la pagina verso la quale proseguire la navigazione ed è dichiarato all'interno del file di configurazione, secondo due modalità:

- Locale – è relativo ad una specifica action che è unica che permette la navigazione verso la pagina ad esso associata;
- Globale – è comune a tutte le action, in modo tale che ciascuna di esse possa redirezionare il flusso dell'applicazione verso la pagina ad esso associata.

In un certo senso, il nome forward restituito può essere paragonato all'outcome di Zend Framework e ciascun forword può essere equiparato ad un navigation-case come dichiarazione della action ad un navigation-rule.

9. Eccezioni

La gestione delle eccezioni dichiarative è una potenzialità messa a disposizione solo ed esclusivamente da Zend Framework ma non da cakePHP. Infatti, con Zend Framework, è possibile dichiarare un'eccezione nel file di configurazione in modo tale che, nel caso in cui venga sollevata all'interno del codice, entri in gioco in maniera automatica l'ExeptionHandler cher si fa carico di gestirla. Inoltre, all'eccezione stessa è possibile associare una pagina verso la quale redirizionare il flusso dell'applicazione. Nel caso di studio in esame, è stato previsto l'utilizzo delle eccezioni dichiarative per poter visualizzare all'utente una pagina di errore, qualora si verificasse qualche problema di accesso alla base di dati. Moltissime applicazioni, in questi casi, prevedono semplicemente la visualizzazione di errori di accesso assolutamente incomprensibili all'utente. Per evitare una situazione di questo tipo, si è fatto in modo che, se durante l'accesso al database si verificasse un errore, viene sollevata una ModuleException la quale è intercettata dall'ExceptionHandler che la ricerca nel file di configurazione e fa in modo che all'utente venga visualizzata la pagina ad essa associata.

cakePHP non fornisce questa potenzialità per cui in questo caso si è ricorso alla semplice funzionalità di navigazione. Più precisamente, nel caso in cui si verifichi un errore di accesso alla base di dati, viene sollevata un'eccezione che, intercettata all'interno del codice stesso, fa in modo che il metodo del backing bean in questione restituisca l'outcome verso la pagina di errore.

10. Sicurezza

La Web Application realizzata prevede una funzionalità particolare, nell'ambito della quale

sono trasmessi dei dati fortemente sensibili, quali l'indirizzo e il numero di telefono, per inviare eventuali riviste. In una situazione di questo tipo, si rende necessario l'utilizzo di un meccanismo di sicurezza mediante il quale sia possibile crittografare i dati inviati attraverso la rete, in modo da renderli incomprensibili a chiunque riesca ad intercettarli in maniera fraudolenta.

Gli elementi principali che permettono un approccio di questo tipo sono fondamentalmente due:

- L'acquisizione di un certificato digitale;
- Uso del protocollo HTTPS, HTTP basato su SSL (Secure Socket Layer)

Nel caso di studio in esame, per garantire la sicurezza,, si renderebbero necessarie le seguenti operazioni:

- Creazione di un certificato digitale di esempio;
- Abilitazione del Connector SSL nel Web Container Apache;
- Configurazione delle pagine protette nel Deployment Descriptor dell'applicazione.

La creazione non è stata necessaria in quanto la Web Application era rilegata ad una rete LAN locale non connessa ad Internet.

Conclusioni

Quanto manca alla vetta?
Tu sali e non pensarci!
(Friedrich Wilhelm Nietzsche)

Fino a poco tempo fa il framework PHP era considerata “il figlio del servo”, perché non c’era ancora la percezione del valore e della sua utilità.

PHP, acronimo ricorsivo di Hypertext Preprocessor, attualmente è uno dei linguaggi Web più diffusi su internet che permette di creare da semplici script fino ad interi portali dinamici. Un linguaggio interpretato nato da un progetto della Apache Software Foundation, l’interprete che ne costituisce il cuore è proprietà della Zend inc (http://www.zend.com), che ne distribuisce i sorgenti gratuitamente. Consente di realizzare velocemente pagine a contenuti dinamici, grazie al supporto nativo con i singoli rdbms ed anche alla numerosità delle funzioni built-in e delle librerie reperibili in modo gratuito

A partire dal primo Congresso Web tenutosi a Ginevra nel 1994 è iniziata ad affermarsi l’esigenza di rendere il Web dinamico. L’HTML di per se è statico, permette al massimo di inserire in una pagina un’immagine, una colonna sonora oppure uno spezzone video. L’HTML non è un linguaggio di programmazione ma solo di formattazione dei contenuti, permettendo di controllare come un testo comparirà nella finestra del browser.

Volendo riassumere questo lavoro è possibile dire che i Framework aumentano la produttività, presentano soluzioni già pronte ai problemi comuni, sono scalabili e rendono semplice la manutenzione del software.

Abbiamo visto come l’applicazione non possa essere considerata un blocco nel quale entrano ed escono dati, ma piuttosto debba essere considerata un insieme di sistemi ‘vivi’ ed in evoluzione che cooperano fra loro, alle volte.

Secondo tali considerazione, non esiste un Framework più adeguato ma esiste un Framework migliore o peggiore dipendente dal problema da affrontare. Il fatto stesso che la funzione htlmflash(arguno,argdue) di Zend Framework non sia utilizzabile con tutti i browser mettono cakePHP, nonostante la poca praticità delle sue ACL, in vantaggio rispetto a Zend Framework nella realizzazione di Web Application che si interfacciano con Flash pur essendo in possesso Zend Framework stesso, su base teorica, di ottimi mezzi tra i quali il Service.

Sebbene, attraverso questo lavoro di tesi, si noti che il PHP 5.0 abbia fornito grosse novità nell'ambito della programmazione orientata agli oggetti e dei passi fatti in questo campo, non possiamo ancora affermare che questo linguaggio sia completamente OOP (es. Java o C++), anche se, abbiamo letto che gli stessi sviluppatori del PHP 5.0 stanno già lavorando su PHP 6.0.

concludiamo osservando che la programmazione web, anche se stretta “parente” della programmazione software, al momento non potrà fornire in modo concreto tutti i vantaggi della OOP, in quanto il contesto e' molto diverso.

Appendice – SRS Life Scann Game

Quando sei a corto di idee, risparmia il cervello e serviti delle orecchie...
(Robert Baden-Powell)

1. Introduzione

In lingua italiana, la parola **appendice** indica genericamente una cosa aggiunta ad un'altra e che da essa dipende. Altri significati sono:

- Letterature - Parte di un libro, giornale o rivista;
- Medicina - Appendice della cute;
- Altro - Elemento strutturale di una nave.

1.1 Propositi

Questo documento ha lo scopo di definire i requisiti e le specifiche del prodotto software "", al fine di facilitarne la realizzazione e la validazione.

È destinato sia al committente del prodotto software che allo sviluppatore, al fine di definire una base di riferimento per la validazione del prodotto e creare le premesse per la produzione di un software di alta qualità.

1.2 Obiettivi

Il prodotto software "**Life Scann Game**" ha come obiettivo la realizzazione di una Web Application che permetta agli utilizzatori di usufruire di un gioco, quale un cameriere che corre lungo un corridoi con una penna USB sul vassoio. Il sistema permette l'autorizzazione delle seguenti operazioni, per quanto riguarda l'interazione con L'User:

1. Identificazione dell'User tramite l'Username e la Password;
2. Visualizzazione del proprio punteggio;
3. Accesso al gioco;
4. Uscita dal sistema.

il Blogger dispone di funzionalità avanzate, mediante le quali saranno consentite le seguenti operazioni:

1. Identificazione del Blogger tramite l'Username e la Password;
2. Gestione degli utenti registrati;
3. Uscita dal sistema.

L'Admin dispone di funzionalità avanzate, mediante le quali saranno consentite le seguenti operazioni:

1. Identificazione dell'Admin tramite l'Username e la Password;
2. Gestione del gioco;
3. Uscita dal sistema.

1.3 Definizioni, Acronimi e Abbreviazioni

L'**acronimo** (dal greco ἄκρον, *akron*, "estremità" + ὄνομα, *onoma*, "nome") è un nome formato con le lettere o le sillabe iniziali o finali di determinate parole di una frase o di una definizione, leggibili come se fossero un'unica parola.

L'**abbreviazione** (detta anche **abbreviatura**) è la riduzione, mediante simboli convenzionali, di una parola o di una frase nella scrittura o nella pronuncia. Non va confusa coll'apocope o troncamento.

In questo caso di studio:

- User – Persona che risulta abilitata all'utilizzo dei servizi resi disponibili;
- Blogger - Persona addetta alla gestione degli archivi degli User oltre che ha effettuare la registrazione (es. “Base” oppure “Estesa”) degli User;
- Admin – Persona addetta alla gestione degli archivi degli User e dei Blogger oltre alla gestione dei servizi offerti dal sistema.
- Utilizzatore / Ruolo / Guest – Persona non ancora identificata tra User, Blogger o Admin che non ha effettuato l'accesso al sistema;
- Servizio – Ciascuna delle attività accessibile e consentite ad un qualunque tipo di User, Blogger o Admin;
- Registrazione – Contratto stipulato dall'User, per mezzo del Blogger e necessario per poter usufruire dei servizi offerti dal sistema. Puo essere di tre tipi:
  - Base - permette di accedere a tutti i servizi a meno della registrazione dell'User;
  - Estesa – Offre i medesimi servizi della registrazione “Base” ed in più permette la registrazione degli User e la gestione degli archivi;
  - Illimitata – accesso a tutti i servizi;
- Username – Identificativo univoco dell'User, del Blogger e dell'Admin;
- Password – Chiave di Accesso al sistema dell'User, del Blogger e dell'Admin;
- Applet Flash - Il termine **applet** (neologismo degli anni '90, composto aplologico di *application* e *let*, usato in italiano al maschile) indica un programma che viene eseguito come "ospite" nel contesto di un altro programma, detto per questo *container*, su un computer Client. In altre parole, un Applet è un programma progettato per essere eseguito all'interno di un programma-container; ne consegue che l'Applet non può essere eseguito indipendentemente da un altro programma. Il

termine è stato introdotto sin dal 1993 con gli Applescript, anche se, oggigiorno, è comunemente riferito alle applet Java, ospitate ("fatte girare" nel gergo informatico) da un browser. Esempi di applet sono Adobe Flash e Windows Media Player. Gli applet presentano in genere qualche forma d'interfaccia utente. Questa è la caratteristica che le distingue da programmi scritti con linguaggi di *scripting* (ad esempio JavaScript) i quali, anche se eseguiti nel contesto di un programma ospite sul Client, non vengono generalmente considerati Applet. Il programma "ospite" eseguito nel contesto di un altro programma su di un computer server è invece definito *Servlet*.

1.4 Riferimenti

In programmazione, la parola **riferimento** (o ***reference*** in inglese) indica, in generale, un valore che identifica univocamente (e permette di accedere a) un dato in memoria. Uno standard più attinente al sistema nel suo complesso (e che concerne sia il kernel che le librerie) è lo standard POSIX. Esso prende origine dallo standard ANSI C, che contiene come sottoinsieme, prevedendo ulteriori capacità per le funzioni in esso definite, ed aggiungendone di nuove.

1.5 Generalità

L'intento di questo documento è quello di descrivere le funzionalità che il software deve soddisfare, le quali saranno specificate nel paragrafo successivo in modo chiaro e coinciso.

Il resto del documento contiene l'elenco delle funzioni che il prodotto software deve avere insieme ai vincoli che deve soddisfare. Più avanti nel testo sono presentate l'interfaccia (sia verso l'User che verso il Blogger e l'Admin) dei servizi messi a disposizione dal prodotto e l'interfaccia verso l'hardware. Al presente documento ne sono allegati altri contenenti i diagrammi realizzati secondo lo Standard UML, necessari per la miglior comprensione del dominio applicativo.

2. Descrizione Generale

Essa è la rappresentazione verbale di un fatto, una cosa o di una persona (SIN. **esposizione**, **narrazione**: *d. di una festa, di un viaggio).*

2.1 Prospettive del Prodotto

Questo prodotto non si integra con nessun altro sistema software ed indipendente ed autonomo per quanto riguarda l'elaborazione (stand-alone).

2.2 Funzionalità

Il prodotto software “**Life Scann Game**” dovrà:

- Nei confronti dell'Admin fornire le seguenti funzionalità:
  - Accesso al sistema;

- Registrazione dell'accesso;
- Gestione del gioco che prevede:
  - Eliminazione di un punteggio;
  - Modifica User;
  - Richiesta dei dati di accesso.
- Uscita dal sistema;
- Registrazione dell'uscita.

- Nei confronti del Blogger fornire le seguenti funzionalità:
  - Accesso al sistema;
  - Registrazione dell'accesso;
  - Gestione degli utenti registrati, che prevede:
    - Sblocco di un User bloccato;
    - Cancellazione User;
    - Registrazione User.
  - Registrazione dell'uscita.
  - Uscita dal sistema;
- Nei confronti dell'User fornire le seguenti funzionalità:
  - Accesso al sistema;
  - Registrazione dell'accesso;
  - Accesso al gioco;
  - Accesso alla visualizzazione del punteggio;
  - Registrazione dell'uscita.
  - Uscita dal sistema;

2.3 Caratteristiche Utente

Il prodotto software è destinato ad utenti che abbiano una conoscenza di base nell'utilizzo del Persona Computer e relativi software per la produttività personale, mentre i Blogger e gli Admin, che usano il software di gestione, hanno bisogno di una conoscenza informatica di base.

2.4 Vincoli Generali

I vincoli generali sono i seguenti:

- L'User, che richiede l'accesso ai servizi del sistema, deve digitare esclusivamente il proprio Username e la relativa Password;

- L'User non può inserire più di due volte una password errata;
- L'Admin non può inserire più di due volte una password errata;
- L'Admin che richiede l'accesso ai servizi del sistema deve digitare esclusivamente il proprio Username e la relativa Password;
- Il Blogger non può inserire più di due volte una password errata;
- il Blogger che richiede l'accesso ai servizi del sistema deve digitare esclusivamente il proprio Username e la relativa Password;
- La Password di accesso dell'User, del Blogger o dell'Admin può essere composta da almeno cinque caratteri alfanumerici.

2.5 Assunzioni e Dipendenze

Il sistema software dovrà essere utilizzato su una macchina dotata di una distribuzione Linux, con 128 Mb di memoria di RAM, Hard Disk da 10 GB ed una connessione Intranet.

3. Requisiti Specifici

Sono le caratteristiche che utente e committente desiderano che siano presenti in un prodotto software da realizzare:

- L’obiettivo di un progetto di sviluppo software, quando si realizza un nuovo sistema o si apportano modifiche ad un sistema già esistente, è sempre realizzare un prodotto che soddisfi i requisiti;
- Utente e committente valuteranno il risultato sulla base del fatto che soddisfi o meno i requisiti.

3.1 Requisiti di interfaccia esterna

L’interfaccia del sistema deve essere realizzata in modo da essere visualizzata all’interno del portale preesistente nell’organizzazione, ovvero Plumtree Corporate Portal 4.5, ciò implica che l’interfaccia utente deve essere realizzata in HTML rispettando lo stile del portale e l’interfaccia per interagire il portale deve essere quella fornita dal produttore Plumtree, ovvero il Plumtree Gadget Developer Studio. la tecnologia di interazione adottata è di uso comune (video, tastiera e mouse). L'uso della tastiera è richiesto solo per inserire dati testuali e non per invocare l'esecuzione di comandi. Presso l’utente non è richiesta l’installazione di software particolare, se non un comune browser Internet per accedere al portale, pertanto non esistono particolari richieste hardware presso l’utente.

L’interazione con il sistema CRM deve avvenire nel modo più semplice possibile, senza apportare modifiche ad esso, preferibilmente accedendo ai suoi dati. Inoltre, data la natura prototipale del sistema CRM, progettare il sistema d’accesso ai dati in modo da essere il più indipendente possibile da esso, sia in termini di assunzione della tecnologia usata per la base di dati, sia in termini di dati trattati dal sistema CRM, i quali possono subire sia modifiche sia

incrementi nel corso del tempo. Altra caratteristica da tenere in considerazione è la possibilità futura di uno spostamento fisico del sistema CRM in una locazione differente dall'attuale e con metodi di protezione che possono non consentire l'accesso utilizzando determinate tecnologie. Si devono utilizzare quindi tecnologie di integrazione che tengano conto di questi fattori (es. Web Service).

3.1.1 Interfaccia Utente

L'interazione tra il sistema e l'utente avviene mediante un'interfaccia Web-Based. Il client sarà un comune browser installato sul computer dell'utente del portale. Tutte le maschere saranno quindi realizzate in HTML seguendo lo stile del portale che ospiterà il gadget, in particolare per quanto riguarda colori e formattazione del testo. È richiesta un'interfaccia utente visuale, con finestre, bottoni e form di immissione dati.

3.1.2 Interfaccia Hardware

Il sistema richiede le risorse presenti nel laboratorio SERlab, in particolare i computer denominati Serlab10, Serlab23 e Serlab26. Il sistema software non si interfaccia con alcun particolare sistema hardware.

3.1.3 Interfaccia Software

Il sistema richiede le seguenti risorse software:

- Ubuntu 10.0.
- Zend Framework;
- cakePHP;
- MySql DB;
- Apache.

Il sistema software non si integra e non comunica con nessun altro sistema software.

3.1.4 Interfaccia di Comunicazione

La comunicazione fra il sistema software "lato client" e quello "late server", si svolge utilizzando la rete LAN e si basa fondamentalmente sull'accesso di un database condiviso localizzato sul server stesso.

3.2 Requisiti Funzionali

Requisiti su ciò che il sistema deve fare. Specificano quali funzioni il sistema deve fornire per soddisfare i bisogni degli stakeholder. Possono essere a livello generale o di dettaglio. Possono venire scoperti grazie alla descrizione dei casi d'uso ed all'analisi degli scenari – oppure con altre tecniche, per sistemi le cui funzionalità non sono adatte ad una descrizione in termini di casi d'uso.

3.2.1 Admin

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e serve a migliorare la comprensione dei requisiti del caso d'uso Login Admin.

Tabella 3 - Login Admin

Nome	Descrizione
Introduzione	Questa funzionalità permette all'Admin di accedere al sistema.
Input	Username (obbligatorio). Paasword (obbligatorio).
Elaborazione	Visualizzazione del form per l'input dei dati, sul quale il sistema esegue le seguenti operazioni: • Controlla che Username e Password siano corrette e cioè corrispondenti a quelle in memorizzate in archivio; • Controlla che l'Admin non sia già logato.
Output	Menù di accesso alle aree di gestione. Dati relativi all'accesso registrati in archivio: • Data e Ora di login; • Indirizzo IP; • Validità login; • Tentativi. Messaggi: • “Username e/o Password errati” • Tentativi di accesso esauriti: sistema bloccato” • “Già loggato”

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Cancella Punteggio

Tabella 4 - Cancellazione Punteggio

Nome	Descrizione
Introduzione	Questa funzionalità permette all'Admin di cancellare uno o più punteggi presenti in archivio.
Input	Identificativo punteggio (obbligatorio).
Elaborazione	Visualizzazione di un form, contenente l'elenco dei punteggi, con possibilità di selezionare i punteggi da cancellare. Inoltre, il sistema effettua le seguenti operazioni: • Controlla che sia stato selezionato almeno un punteggio; • Visualizzazione dell'elenco dei punteggi.
Output	Elenco dei Brani. Messaggi: • "Eliminazione effettuata con successo" • "Non è stato selezionato nessun punteggio"

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Richiesta Username / Password.

Tabella 5 - Richiesta Username / Password

Nome	Descrizione
Introduzione	Questa funzionalità permette all'Utente chi chiedere all'Admin l'invio di una E-Mail contenente l'Username e la Password qualora se ne fosse dimenticato.
Input	E-Mail (obbligatorio).
Elaborazione	Visualizzazione di un form per l'inserimento dei dati, sui quali il sistema effettua le seguenti operazioni: • Controlla che il campo relativo all'E-mail non sia vuoto e sia valido; • Caricamento dei dati personali in archivio; • Invio dell'E-Mail contenete Username e Password.
Output	Messaggi: • “E-Mail inviata con successo” • “E-Mail non registrata in archivio” • “L'E-Mail ha un formato non valido” • “Il campo relativo all'E-Mail non può essere vuoto”

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Modifica Dati Anagrafici.

Tabella 6 - Modifica Dati Anagrafici

Nome	Descrizione
Introduzione	Questa funzionalità permette all'Admin di modificare i dati personali degli User.
Input	Nome (obbligatorio). Cognome (obbligatorio). Indirizzo (obbligatorio). CAP (obbligatorio). Provincia (obbligatorio). Città (obbligatorio). Telefono (obbligatorio). Sesso (obbligatorio). Data di Nascita (obbligatorio).
Elaborazione	Visualizzazione del form per l'input dei dati, sul quale il sistema esegue le seguenti operazioni: • Controlla che i campi non siano vuoti; • Registrazione dei nuovi dati di input in archivio.
Output	Tutti i dati letti in input registrati in archivio. Messaggi: • "Aggiornamento effettuato con successo" • "E-Mail non valido". • "Il campo relativo al cognome non può essere vuoto". • "Il campo relativo al nome non può essere vuoto". • "Il campo relativo al CAP non può essere vuoto". • "Il campo relativo al telefono non può essere vuoto". • "Il campo relativo alla provincia non può essere vuoto". • "Il campo relativo alla città non può essere vuoto".

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Logout Admin

Tabella 7 - Logout Admin

Nome	Descrizione
Introduzione	Questa funzionalità permette all'Admin di eseguire il logout dal sistema.
Input	Identificativo Admin (obbligatorio).
Elaborazione	Registrazione dei dati in archivio che tengono traccia del logout
Output	Dati relativi all'uscita dal sistema registrati in archivio: • Data logout; • Ora Logout. Messaggi: • “Logout effettuato con successo” • “Sessione scaduta”

Fonte: Attanasio Ciro tramite ArgoUML

3.2.2 User

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Login User.

Tabella 8 - Login User

Nome	Descrizione
Introduzione	Questa funzionalità permette all'User di accedere al sistema.
Input	Username (obbligatorio). Paasword (obbligatorio).
Elaborazione	Visualizzazione del form per l'input dei dati, sul quale il sistema esegue le seguenti operazioni: • Controlla che Username e Password siano corrette e cioè corrispondenti a quelle in memorizzate in archivio; • Controlla che l'Admin non sia già loggato.
Output	Menù di accesso alle aree di gestione. Dati relativi all'accesso registrati in archivio: • Data e Ora di login; • Indirizzo IP; • Validità login; • Tentativi. Messaggi: • “Username e/o Password errati”; • Tentativi di accesso esauriti: sistema bloccato” • “Già loggato”.

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Gioca.

Tabella 9 - Gioca

Nome	Descrizione
Introduzione	Questa funzionalità permette all'User di visualizzare il gioco.
Input	Identificativo User (obbligatorio).
Elaborazione	Visualizzazione del form dove inserire il nome.
Output	Messaggi: • "Nessun Punteggio presente"; • "Operazione non valida".

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Visualizza Punteggio.

Tabella 10 - Visualizza Punteggio

Nome	Descrizione
Introduzione	Questa funzionalità permette all'User di visualizzare l'elenco dei punteggi presenti in archivio.
Input	Punteggio (obbligatorio).
Elaborazione	Visualizzazione del form con l'elenco completo dei punteggi.
Output	Messaggi: • “Nessun Punteggio presente”; • “Operazione non valida”.

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Logout User.

Tabella 11 - Logout User

Nome	Descrizione
Introduzione	Questa funzionalità permette all'User di eseguire il logout dal sistema.
Input	Identificativo User (obbligatorio).
Elaborazione	Registrazione dei dati in archivio che tengono traccia del logout
Output	Dati relativi all'uscita dal sistema registrati in archivio: • Data logout; • Ora Logout. Messaggi: • “Logout effettuato con successo”; • “Sessione scaduta”,

Fonte: Attanasio Ciro tramite ArgoUML

3.2.3 Blogger

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Login Blogger.

Tabella 12 - Login Blogger

Nome	Descrizione
Introduzione	Questa funzionalità permette al Blogger di accedere al sistema.
Input	Username (obbligatorio). Paasword (obbligatorio).
Elaborazione	Visualizzazione del form per l'input dei dati, sul quale il sistema esegue le seguenti operazioni: • Controlla che Username e Password siano corrette e cioè corrispondenti a quelle in memorizzate in archivio; • Controlla che il Blogger non sia già loggato.
Output	Menù di accesso alle aree di gestione. Dati relativi all'accesso registrati in archivio: • Data e Ora di login; • Indirizzo IP; • Validità login; • Tentativi; Messaggi: • “Username e/o Password errati” • Tentativi di accesso esauriti: sistema bloccato” • “Già loggato”

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Cancellazione di un User.

Tabella 13 - Cancellazione di un User

Nome	Descrizione
Introduzione	Questa funzionalità permette al Blogger di cancellare uno o più User presenti in archivio.
Input	Identificativo User (obbligatorio).
Elaborazione	Visualizzazione del form contenente l'elenco degli user, con possibilità di selezionare gli user da cancellare. Inoltre, il sistema effettua le seguenti operazioni: • Controlla che sia stato selezionato almeno un User; • Eliminazione dei dati presenti in archivio relativi agli User selezionati; • Visualizzazione dell'elenco degli Users.
Output	Elenco degli Users. Messaggi: • “Eliminazione effettuata con successo”; • “Nessun User è stato selezionato”.

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Sblocco di un User.

Tabella 14 - Sblocco di un User

Nome	Descrizione
Introduzione	Questa funzionalità permette al Blogger di sbloccare un User che ne abbia fatto richiesta ufficiale in maniera verbale.
Input	Identificativo User (obbligatorio).
Elaborazione	Registrazione dei dati in archivio.
Output	Sblocco User.

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Visualizzazione Dati Anagrafici.

Tabella 15 - Visualizzazione Dati Anagrafici di un User

Nome	Descrizione
Introduzione	Questa funzionalità permette al Blogger di visualizzare i dati anagrafici di un user presente in archivio.
Input	Identificativo User (obbligatorio). Dati caricati dal sistema provenienti dall'archivio.
Elaborazione	Caricamento dei dati presenti in archivio. Visualizzazione dei dati anagrafici dell'User.
Output	Dati anagrafici dell'User selezionato

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Registrazione User.

Tabella 16 - Registrazione User

Nome	Descrizione
Introduzione	Questa funzionalità permette al Blogger di effettuare la registrazione degli User, per permettere agli User di usufruire dei servizi offerti dalla Web Application
Input	Dati dell'User: • E-Mail (obbligatorio); • Username (obbligatorio); • Password (obbligatorio).
Elaborazione	Visualizzazione di un form per l'input dei dati, sui quali il sistema esegue le seguenti operazioni: • Controlla che i campi relativi all'E-Mail, Username e Password non siano vuoti o non validi. In particolare, l'E-Mail abbia una struttura corretta e la Password abbia la lunghezza di cinque caratteri; • Registrazione dei dati di input in archivio.
Output	Tutti i dati letti in input registrati in archivio. Altri dati registrati: • Data registrazione; • Ora registrazione. Messaggi: • “Registrazione effettuata con successo” • “E-Mail non valida” • “Il campo relativo all'E-Mail non può essere vuoto” • “Il campo relativo all'Username non può essere vuoto” • “Il campo relativo alla Password non può essere vuoto” • “La password deve essere alemn0o di cinque caratteri”

Fonte: Attanasio Ciro tramite ArgoUML

La documentazione testuale descrive la sequenza di eventi di un attore che utilizza il sistema e

serve a migliorare la comprensione dei requisiti del caso d'uso Logout Blogger.

Tabella 17 - Logout Blogger

Nome	Descrizione
Introduzione	Questa funzionalità permette al Blogger di eseguire il logout dal sistema.
Input	Identificativo Blogger (obbligatorio).
Elaborazione	Registrazione dei dati in archivio che tengono traccia del logout
Output	Dati relativi all'uscita dal sistema registrati in archivio: • Data logout; • Ora Logout. Messaggi: • “Logout effettuato con successo”; • “Sessione scaduta”.

Fonte: Attanasio Ciro tramite ArgoUML

3.3 Requisiti di Prestazione

La pianificazione delle prestazioni e della capacità è il processo di associazione della struttura della soluzione a un formato e all'insieme di componenti hardware che dovranno supportare gli obiettivi aziendali prefissati. Questa sezione contiene informazioni sulla pianificazione delle prestazioni e della capacità.

Esistono due modi per individuare i requisiti prestazionali sono:

- definire in modo chiaro l'obiettivo del vostro progetto in modo tale da non perdere tempo su requisiti che non sono necessari;
- formulare domande sulla base dei vostri modelli di analisi.

I modelli di analisi devono essere ideati in collaborazione con esperti delle varie aree di business. Questi modelli aiutano a scoprire e a omologare le esigenze degli utenti dando anche

una base per ottenere informazioni circa i requisiti.

Il sistema sarà eseguito all'interno di un portale. Data la natura multiutente di quest'ultimo si suppone che al sistema possa accedere più di una persona contemporaneamente. Il sistema dovrà quindi essere progettato in modo da sopportare un carico di utenti maggiore di uno. Data la natura sperimentale del sistema, tale requisito non è momentaneamente essenziale.

3.4 Vincoli di Progetto

Tutti i progetti hanno dei vincoli (constraints) che li contraddistinguono. Un vincolo è "una restrizione o una limitazione, sia interna che esterna al progetto, che influisce sulle prestazioni del progetto" (es. un vincolo di schedulazione, di budget, sulle risorse del progetto, ecc.).

I project manager parlano spesso di "triplice vincolo", vale a dire ambito (es. il lavoro da svolgere per fornire i deliverable con le caratteristiche e le funzioni specificate), tempi e costi del progetto.

La relazione tra questi tre driver è tale per cui alla variazione anche di uno solo dei tre, almeno un altro ne risulta influenzato: ad esempio, se ad un project manager viene chiesto, in corso di esecuzione del progetto, di ridurre i tempi mantenendo inalterato l'ambito del progetto, è chiaro che occorrerà aumentare i costi.

Il continuo sforzo per individuare il giusto equilibrio tra queste tre variabili ha, inoltre, un sensibile impatto sulla qualità del progetto, sui rischi e sul "clima" delle relazioni tra i membri del project team. I progetti di successo rilasciano i deliverable richiesti nell'ambito stabilito, in conformità ai requisiti del cliente, entro il tempo fissato e restando entro i limiti del budget definito.

Il sistema deve tener conto delle varie figure che possono avere accesso ad esso, in particolare in termini di quali statistiche possono essere visualizzate da ogni figura. Il sistema deve adattarsi automaticamente all'utente, presentandogli solo ed esclusivamente le statistiche abilitate ad essere visualizzate da esso.

Il sistema deve essere realizzato in modo da permettere l'aggiunta futura di nuove statistiche eseguibili con il minor sforzo possibile e demandando quanto più l'elaborazione di tali statistiche ad applicazioni preesistenti nell'organizzazione. Tali applicazioni andranno quindi integrate nell'intero sistema.

3.5 Attributi

L'**attributo** indica, nell'analisi sintattica, la funzione attributiva di un aggettivo o di un sostantivo che non possiede una funzione autonoma ma dipende da un costituente nominale al quale aggiunge qualificazioni o determinazioni. In altre parole è una apposizione aggettivale. Se l'attributo è dato da un aggettivo concorda con il nome dal quale dipende nel numero e nel caso (es. *i capelli neri*, *la ragazza alta e l'uomo gentile)*. Se invece è costituito da un sostantivo, facente la funzione logica di apposizione, ovviamente non si declinerà ma sarà legato al sostantivo di riferimento tramite semplice apposizione o tuttalpiù attraverso particelle e locuzioni come "da", "in qualità" e "come" (es. Catone il censore, in qualità di madre ecc.)

Essendo l'attributo privo di una propria funzione sintattica assume quella del costituente che lo regge. Si distingue così tra attributo del soggetto, del complemento oggetto, come pure di qualsiasi complemento indiretto tranne che nel predicato verbale, dal momento che la funzione dell'aggettivo (attributo) o di un altro elemento nominale non assume più la funzione attributiva ma predicativa.

In informatica:

- Attributo – file e sistemi operativi;
- Attributo – database;
- Attributo – sicurezza;
- Attributo - in un linguaggio di programmazione;
- Attributo - grammatiche ad attributi nei linguaggi di programmazione.

3.5.1 Sicurezza

Come strumento di protezione è previsto l'utilizzo di una Password al fine di evitare accessi al sistema da parte di perone non autorizzate.

3.5.2 Database

Il sistema software prevede l'utilizzo di un database relazionare per l'archiviazione dei dati.

3.6 Altri requisiti

Esistono molti diversi sistemi di classificazione dei requisiti, nel mondo del software. E finora non si è imposto alcuno standard. Qui di seguito, in sintesi, alcuni tra i più diffusi tra gli altri sistemi classificatori.

- ISO 9126 - non nasce per classificare i requisiti, è relativa agli aspetti di qualità dei prodotti sw. Ma è comunque molto usata nella classificazione dei requisiti;
- Incose - International Council on Systems Engineering;
- Volere - un template per la specifica dei requisiti di James e Suzanne Robertson.

Ringraziamenti

Quanto più ci innalziamo, tanto più piccoli sembriamo
a quelli che non possono volare
(Friedrich Wilhelm Nietzsche)

Quando nella vita si raggiungono grandi traguardi e si realizzano i propri sogni, ci si rende conto che, senza l'appoggio delle persone che ci vivono accanto, gli sforzi e l'impegno personale non avrebbero mai consentito da soli tali risultati. Ho sempre considerato lo studio e la conoscenza tra gli elementi fondamentali della vita di una persona e per questo, per raggiungere il mio obbiettivo, non ho mai lesinato nei miei confronti l'impegno, aggiungendovi costanza, instancabilità, caparbietà e tenacia che comunque sarebbero state vane senza le persone che qui ho il piacere ed il dovere di ringraziare.

I miei ringraziamenti vanno innanzitutto al Professore *Carlo Blundo*, il quale ha accettato di seguirmi in questo mio lavoro che ho svolto con grande entusiasmo. Grazie al suo si, ho avuto la possibilità di affrontare ed approfondire nuovi temi legati allo sviluppo del Software per Internet, per me di estremo interesse.

Gli anni di studio, che mi hanno portato a questo grande traguardo, sono stati caratterizzati da momenti di gioia cosi come da periodi di sacrifici ed amarezze, che non avrei saputo superare senza coloro che mi hanno preso per mano lungo questo cammino.

Un immenso grazie va innanzitutto alla mia famiglia (*Mamma*, P*apà* ed i miei *Fratelli*/*Sorelle*) che mi hanno dato la possibilità di realizzare questo sogno, senza mai ostacolarmi nelle decisioni ma dandomi sempre il massimo appoggio. In questi anni, hanno dovuto sopportare i miei comprensibili momenti di tensione e nervosismo, senza mai rimproverarmi alcun comportamento. A loro devo chiedere scusa, per non aver dedicato il giusto tempo che merita una famiglia che ama il proprio figlio e il proprio fratello/sorella.

Grazie alla mia *Team Manager*, che rappresenta per me il modello da seguire, umanamente e professionalmente, la quale non ha mai risparmiato consigli ed elogi nei miei confronti. Spero di poter crescere nel tempo ed acquisire le notevoli capacità che gli riconosco.

Grazie a tutti i miei *Amici* più sinceri, con i quali ho trascorso delle ore spensierate, lontano dagli affanni dello studio, e che mi hanno fatto sentire una persona importante per loro.

Un grazie particolare va alla mia *Ragazza*, che ho avuto l'immensa fortuna di conoscere circa sei anni fa, diventando l'amore della mia vita. Si è ritrovata “catapultata” nel mio mondo, fatto di moltissime ore di studio, di sacrifici e di preoccupazioni ma dandomi sempre la forza di rialzarmi nei momenti di rabbia e di conforto. In questo ultimo anno, il suo amore nei miei confronti è stato una grande fonte di energia, dalla quale spero di attingere per tutto il resto della mia vita.

Grazie di cuore a tutti.

Bibliografia

Non ho particolari talenti, sono solo appassionatamente curioso
(Albert Einstein)

[1] Matt Zandstra
"*PHP 5 Objects, Patterns and Practice*"
Apress, 2005.

[2] Gamma, Helm, Johnson e Vlissides (GoF)
"*Design Patterns: Element of Reusable Object-Oriented Software*"
Addison-Wesley, 1995.

[3] Martin Fowler
"*Patterns of Enterprise Application Architecture*"
Addison-Wesley, 2003.

[4] P. Atzeni et al
"*Basi di dati: concetti, linguaggi e architetture*"
McGraw-Hill, seconda edizione 1999.

[5] Chris Shiflett
"*PHP Security O'Reilly Open Source Convention Portland*"
Oregon, USA 26 Jul 2004.

[6] Raghu Ramakrishnan e Johannes Gehrke
"*Database Management Systems, 2nd Edition*"
McGraw Hill, 2000 (ISBN 0-07-232206-3).

[7] Quentin Zervaas
"*Sviluppare applicazioni Web 2.0 con PHP*"

Apogeo, 2008.

[8] Allen Rob

"Zend Framework in Action"

Manning, 2009.

[9] Mariano Iglesias

"CakePHP 1.3 Application Development Cookbook (RAW)"

Manning, 2009.

[10] Kai Chan and John Omokore

"Practical CakePHP Projects"

Published Dec 2008.

Riferimenti

La grandezza dell'uomo si misura in base a quel che cerca
e all'insistenza con cui egli resta alla ricerca
(Martin Heidegger)

[1] http://www.phpframeworks.com/

[2] http://cakephp.org/

[3] http://book.cakephp.org/

[4] http://framework.zend.com/

[5] http://framework.zend.com/docs/overview

[6] http://www.zend.com/products/studio/

[7] http://www.zend.com/zend/art/art-oertli.php

[8] http://www.php.net/manual/en/security.php

[9] http://www.w3.org/Protocols/rfc2616/rfc2616.html

[10] http://www.php.net/