Università degli Studi di Padova

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Triennale in Informatica

Tesi di Laurea:

"Servizi di Hosting per Ruby on Rails"

Candidato
Luca Bagante
465215

Relatrice
Dott.ssa K. Brent Venable

A Claudio e Patrizia.
A Ruggero.
A Sara.

Indice

Rigraziamenti 73

Nota

i riferimenti bibliografici e i riferimenti sul web sono riconoscibili nel testo perché segnalati da un numero o da una lettera minuscola racchiusi tra parentesi quadre. Es. *[1]* è il primo riferimento bibliografico, *[a]* è il primo riferimento sul web.

Introduzione

Obiettivo principale dello stage, è stato la realizzazione di un'applicazione web in grado di semplificare e automatizzare la configurazione di hosting su piattaforma Linux, con lo scopo di fornire supporto per il linguaggio di scripting Ruby, il framework Rails e il database relazionale Mysql. Tale applicazione, scritta essa stessa in Ruby On Rails e basata su Mysql, fornisce un sistema agile per la configurazione di siti web ospitati in hosting presso la struttura dell'azienda.

Scopo del documento

Il presente documento descrive il periodo di stage aziendale svolto presso Windnet s.r.l., Internet Service Provider di Padova. Tale stage ha avuto l'obiettivo principale di concludere l'attività formativa prevista nel Corso di Laurea in Informatica, fornendo allo studente un contatto diretto con il mondo del lavoro e con le problematiche operative ad esso connesse, cogliendo l'opportunità, in questo modo, di applicare ad una vera realtà sul mercato i concetti teorizzati nei corsi di studio. Sarà pertanto fornita una

descrizione dettagliata di tutte le fasi del progetto di sviluppo realizzato durante lo stage.
Nel primo capitolo viene eseguita una panoramica sulle tecnologie utilizzate nei processi dello stage, cercando di coglierne i tratti essenziali, senza pertanto entrare in un dettaglio che renderebbe necessario uno studio dedicato ad ognuna di esse.
Nel secondo capitolo sono affrontate le questioni relative alla pianificazione dei processi e alle questioni temporali ad esso connesse.
Il terzo capitolo espone l'elenco dei requisiti individuati durante l'analisi condotta in collaborazione con l'azienda, e costituisce una traccia per gli obiettivi da raggiungere attraverso lo stage.
Nel quarto capitolo sono descritti i dettagli di progettazione, le scelte architetturali e le conseguenti specifiche tecniche.
Il quinto capitolo analizza il processo di codifica e di messa in opera dell'applicazione progettata.
Il sesto capitolo affronta le problematiche relative alla qualità di processo e di prodotto, soffermandosi sulle attività svolte nell'ambito della verifica e della validazione.
Nel settimo capitolo si vuole esprimere una valutazione conclusiva sul periodo di stage, sia sul versante specifico del progetto, sia su quello più generale, relativo all'esperienza di professionalizzazione.

L'azienda e le esigenze

Windnet s.r.l. è un Internet Service Provider, che opera a Padova dal 1995. Dal 1997 è Maintainer presso IT-NIC (Registration Authority Italiana) e dal 1998 è Local Internet Registry per l'Italia presso il RIPE (Registry europeo per

l'allocazione di risorse IP). Si occupa prevalentemente di fornire servizi di Hosting e Housing e soluzioni di Networking per piccole medie aziende, Enti pubblici e privati.
Ben conoscendo lo scenario di mercato che ha profondamente rivoluzionato le logiche di costo dei propri prodotti, ha attuato negli ultimi anni un processo di innovazione teso a ottimizzare le risorse disponibili abbattendo, da un lato i costi di gestione (grazie a una sempre crescente automazione dei processi) e ampliando l'offerta su nuovi fronti tecnologici dall'altro.
É in questo panorama che si inserisce l'esigenza, percepita dalla direzione, di sviluppare un nuovo sistema in grado di offrire al mercato soluzioni di Hosting per Ruby On Rails, architettura applicativa ormai diffusa ed ampiamente apprezzata da numerosi sviluppatori di software web based.
Tale esigenza si inserisce in un progetto di più ampio respiro, connesso con l'azione di ricerca e sviluppo portata avanti da Windnet nell'ultimo periodo, che toccherà tra le varie cose, anche l'ambito della virtualizzazione.

Capitolo I – Le tecnologie coinvolte

In questo capitolo sono descritte le tecnologie selezionate in fase di studio di fattibilità. Esse sono state utilizzate, in gran parte, per la progettazione architetturale dell'applicazione sviluppata.

Il linguaggio Ruby e il framework Rails

Ruby è un linguaggio di scripting interpretato. É stato creato da Yukihiro Matsumoto nel 1995. La sua natura deriva dall'amore e dall'insoddisfazione dell'autore per suoi predecessori come Perl, Python, Smalltalk, Eiffel, Ada e Lisp da cui eredita numerose caratteristiche, sia per sintassi, sia per architettura. É un linguaggio ad oggetti che si pone come obiettivo la sintesi tra un linguaggio funzionale ed uno imperativo, oltre che una sintassi fortemente ispirata al linguaggio naturale. *[1]*

```
5.times { print "We *love* Ruby -- it's outrageous!" }
```

Ruby è distribuito con una licenza libera.

Tra le varie caratteristiche principali, possiamo elencare le seguenti:

1. gestione delle eccezioni;
2. garbage Collector basato sull'algoritmo "mark-and-sweep";
3. possibilità di scrivere estensioni C in Ruby grazie a specifiche API;
4. possibilità di caricare dinamicamente estensioni;
5. gestione del multithreading indipendente dal Sistema Operativo;
6. portabilità su tutti i principali Sistemi Operativi (GNU/Linux, UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2, etc.).

[a]

Il successo che Ruby ha riscosso negli ultimi anni è stato determinato sicuramente dal framework Rails, creato nel 2003 da David Heinemeier Hansson.

Ruby On Rails (d'ora in avanti anche "RoR" o semplicemente "Rails"), fornisce un set di librerie e codice organizzato secondo un pattern MVC. Questo permette di comprimere sensibilmente i tempi di sviluppo di applicazioni web, grazie all'efficace disaccoppiamento dell'architettura e al notevole supporto garantito dalla comunità che si è condensata attorno al progetto Open Source. *[b]*

Inoltre, Rails è considerato un framework full-stack, ovvero fornisce allo sviluppatore tutte le funzionalità necessarie alla creazione di una web application completa, senza cioè specializzarsi su un aspetto specifico e trascurarne altri. Gli strumenti che fornisce spaziano dalla realizzazione dei sistemi di caching per i dati in persistenza fino all'integrazione di componenti JavaScript. *[2]*

Ruby Enterprise Edition e Phusion Passenger

La navigazione di siti web scritti in Ruby On Rails determina l'esecuzione di più istanze di applicazioni Rails. É stato osservato che sia il codice relativo al

framework, sia il sorgente dell'applicazione, dedicano la maggior parte della memoria agli "AST nodes". Ottenere una riduzione significativa dell'utilizzo di memoria, diventa quindi essenziale. Bisognerebbe trovare il modo per far condividere ad applicazioni Rails differenti la memoria usata dagli "AST nodes", un po' come avviene durante l'esecuzione di programmi C/C++ che condividono la memoria delle librerie.

Questo scenario è reso possibile, nei sistemi operativi di tipo UNIX, dalle API POSIX che, grazie alla chiamata di sistema `fork()`, consentono la creazione di un processo figlio facendo sì che esso condivida la maggior parte della memoria con il proprio padre. Se viene creato un processo figlio tramite `fork()`, solo quando la scrittura su pagine di memoria avviene da parte del padre o del figlio, tale memoria viene copiata, impedendo all'uno di scrivere nella zona di memoria riservata all'altro e viceversa. Questa tecnica si chiama "copy-on-write" e funziona molto bene perché consente di evitare sprechi di memoria, conservando i vantaggi di sicurezza della separazione degli spazi di allocazione.

Purtroppo, l'interprete Ruby non dispone di un Garbage Collector compatibile con la tecnica "copy-on-write". L'implementazione dell'algoritmo "mark-and-sweep" registra l'informazione in un bit direttamente all'interno di ogni oggetto. Questo significa che al primo ciclo eseguito dal Garbage Collector all'interno di un processo figlio, esso accede a tutte le pagine di memoria cui si riferisce, determinandone l'immediata copiatura. Questa architettura causa la completa inutilità del "copy-on-write".

Per sfruttare in modo vantaggioso le API POSIX ed ottenere il risparmio di memoria auspicato, un'azienda olandese di nome "Phusion", ha avviato un progetto chiamato Ruby Enterprise Edition, con l'obiettivo di reimplementare il Garbage Collector di Ruby mantenendo l'elenco delle pagine riferite in una lista esterna, registrata in una zona diversa della memoria. *[g]*

Grazie a questa riprogettazione è possibile osservare un risparmio medio del 33% in termini di memoria virtuale e per questo motivo Ruby Enterprise Edition è stato selezionato per il progetto, nella versione 1.8.6. *[f]*

Phusion ha poi realizzato un altro software molto interessante e che possiamo considerare fondamentale nel dominio del progetto. Si chiama "Passenger" e consiste in un set di applicazioni che hanno lo scopo di eseguire software scritto in Ruby On Rails attraverso il web server Apache (per questo motivo viene anche chiamato "mod_rails"). Assumendo la forma di un "modulo" per il web server, elimina la necessità di utilizzare altri componenti software per l'esecuzione di applicazioni in Ruby On Rails, permettendo direttamente ad Apache di occuparsi degli aspetti di configurazione dell'ambiente.
Un approccio di questo tipo consente ai gestori delle infrastrutture dedicate (come Windnet) di fornire servizi in Ruby On Rails con la stessa agilità con cui questo è fattibile per altri linguaggi più diffusi (PHP, Perl o Python per esempio) e che ne ha determinato in gran parte il successo.
La versione utilizzata nel progetto è la 2.0.3. *[e]*

Il web server Apache

L'httpd server di Apache Software Foundation (chiamato anche semplicemente Apache), costituisce il software più diffuso nel proprio ambito. Lo sviluppo, iniziato nel 1994, ha trasformato un progetto embrionale, nato presso l'NCSA, nel miglior web server oggi a disposizione.
Alcune delle caratteristiche principali sono:

1. funzionamento in una moltitudine di sistemi operativi;
2. tecnologia libera, che permette la contribuzione della comunità allo sviluppo e al bugfixing;

3. approccio modulare alle estensioni, che consente il supporto a numerosissimi linguaggi di programmazione (nel nostro caso mod_rails consente di servire pagine scritte in Ruby);
4. altissima configurabilità, sia delle funzioni di base, sia dei moduli. Inoltre è possibile personalizzare i messaggi di errore o la verbosità e la struttura dei log files.

Detenendo il ruolo di "leader de facto" nel mercato dei web server, e nei sistemi operativi Unix-like in particolare, è logico che Apache sia stato selezionato come tecnologia attivamente coinvolta in questo progetto.
La versione adottata è la 2.2. *[c]*

L'RDBMS Mysql

Mysql è il database relazionale opensource più diffuso. otto tra i primi dieci siti web più visitati al mondo (fonte: alexa.com), tra cui Google, utilizzano Mysql.
Alcune tra le sue caratteristiche principali, che ne hanno decretato la massima diffusione, sono:

1. portabilità;
2. scalabilità garantita dall'uso di thread e dal supporto di più processori;
3. semplicità d'uso;
4. licenza libera.

La versione adottata nel progetto è la 5.0. *[d]*

L'FTP Server VSFTPD

I servizi FTP sono da sempre oggetto di numerose polemiche da parte degli amministratori di sistema, costituendo in molte circostanze la causa principale di falle nella sicurezza. Si è cercato pertanto di selezionare un'applicazione ampiamente diffusa e considerata sicura.
Dopo una fase di software selection, il miglior candidato per semplicità d'uso e per caratteristiche di sicurezza è stato VSFTPD (Very Secure FTP Daemon).
Le principali funzionalità sono:

1. configurazione che supporta IP virtuali;
2. configurazione che supporta utenti virtuali;
3. esecuzione autonoma o basata su superserver inetd;
4. configurabilità granulare per utente;
5. assegnazione granulare della banda;
6. configurazione basata sugli IP di provenienza dei client;
7. supporto IPV6;
8. supporto alla cifratura attraverso l'integrazione di SSL.

La versione utilizzata è la 2.0.6. *[i]*

Il tool Monit e il controllo delle risorse

Monit è un software per il controllo delle risorse. Si occupa di monitorare processi, files, directory e in generale il filesystem in sistemi operativi di tipo Unix. In particolare è in grado di:

1. avviare un processo, se rileva che esso non è in esecuzione;
2. riavviare un processo, se si trova in uno stato di stallo o non risponde;
3. arrestare un processo, se eccede l'utilizzo di risorse assegnate;

4. controllare le modifiche di files, directory e più in generale del filesystem utilizzando il timestamp, la dimensione o il checksum;
5. informare l'amministratore di sistema, attraverso messaggi e-mail, in caso di superamento delle soglie di carico configurate.

Monit va quindi considerato come parte "a sé stante" del progetto, in quanto non è coinvolto come componente dell'applicazione e il suo ruolo è indipendente dallo sviluppo.

In questa fase è stato configurato soltanto per la notifica del superamento dei limiti di consumo delle risorse ma nelle future iterazioni del ciclo di vita dello sviluppo avrà un ruolo centrale per la gestione dei processi di Phusion Passenger, Apache, Mysql e FTP.

La versione utilizzata nel progetto è la 4.8.1. *[j]*

Capitolo II – Piano di progetto

Il progetto ha coinvolto una sola risorsa. La pianificazione dei processi ha quindi avuto principalmente lo scopo di coordinare il lavoro con l'azienda e di calcolare lo sforzo quotidiano necessario per raggiungere gli obiettivi evidenziati in analisi entro la fine dello stage. In questo capitolo seguono alcune considerazioni in merito.

Ciclo di vita

Il ciclo di vita più adatto ad un progetto di questo tipo è quello incrementale-iterativo. È conveniente cioè eseguire ciclicamente le fasi tradizionali di sviluppo (analisi, progettazione, codifica, verifica, deployment), fino all'ottenimento del prodotto voluto, passando per versioni comunque funzionanti. Questo modello di ciclo di vita è utile perchÈ consente di selezionare un sottoinsieme dei requisiti generali individuabili e assegnarlo ad una specifica iterazione, separando il carico lavoro sia in termini di risorse coinvolte, sia di fasce temporali. Il progetto di stage riguarda la prima

iterazione dello sviluppo, assegnata unicamente ad una persona e che pertanto si dipana, sulla linea del tempo, con un andamento sequenziale.

Pianificazione e schedulazione

Lo stage si è svolto in un periodo complessivo di 304 ore, ripartite in sette settimane e tre giorni lavorativi. Descriveremo ora le cinque fasi principali in cui si raggruppano le attività svolte.

Fase 1: Studio di fattibilità e analisi preliminare

Durata: **40 ore**

Descrizione:
Analisi delle esigenze dell'azienda e prima stima sugli strumenti necessari

Dettaglio:
32 ore per analizzare le esigenze dell'azienda in affiancamento con il tutor interno e 8 ore per selezionare gli strumenti necessari (IDE, formato documentazione, abbozzo di piano di progetto).

Fase 2: Analisi dei requisiti

Durata: **56 ore**

Descrizione:
Analisi dei requisiti del sistema, con conseguente produzione del documento

relativo.

Dettaglio:
Individuazione del ciclo di vita del sistema e realizzazione di alcuni diagrammi usecase, lista definitiva dei requisiti.

Fase 3: Progettazione

Durata: **88 ore**

Descrizione:
Progettazione del sistema e specifica tecnica.

Dettaglio:
Scelta definitiva degli strumenti e progettazione dettagliata di ogni parte del sistema.

Fase 4: Codifica

Durata: **48 ore**

Descrizione:
Codifica e realizzazione di alcuni prototipi applicativi.

Dettaglio:
Produzione del codice e della relativa documentazione.

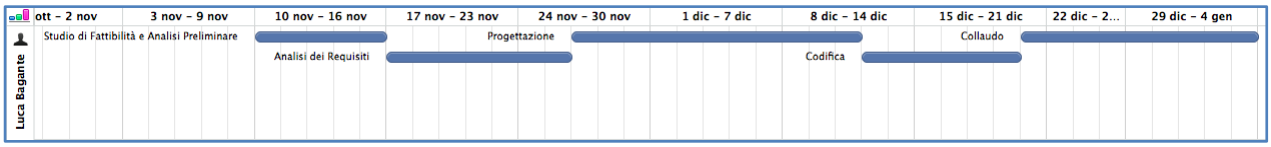
Fase 5: Collaudo

Durata: **72 ore**

Descrizione:
Verica e validazione dell'applicazione sviluppata e relativo collaudo.

Dettaglio:
Redazione manuale utente e attuazione test per verifica e validazione del sistema.

Diagramma di Gantt:



Il diagramma mostra l'andamento temporale delle attività assegnate all'unica risorsa umana coinvolta.
Durante la primissima fase di studio di fattibilità si sono svolti alcuni incontri con il tutor interno che hanno permesso di focalizzare maggiormente gli obiettivi da raggiungere. Il progetto è poi proseguito in modo totalmente autonomo.
L'assenza di ulteriori attori coinvolti, ha permesso di abbattere considerevolmente i tempi di coordinamento del team (inesistente). Questo ha avuto come effetto uno scostamento trascurabile fra la schedulazione preventiva e quella consuntiva.

Capitolo III – Analisi dei requisiti

La fase di analisi è stata caratterizzata da numerosi briefing con l'azienda. In queste riunioni si sono evidenziate le esigenze principali, cercando di organizzarle secondo criteri di priorità. In questo capitolo sono descritti requisiti del sistema sviluppato.

Richieste dell'azienda

Durante la prima fase di studio di fattibilità e analisi preliminare, l'azienda ha espresso la necessità di usufruire di uno strumento per gestire facilmente l'accounting di pacchetti hosting basati su Ruby On Rails. In particolare è stata espressa la necessità di un'applicazione web-based, possibilmente anch'essa sviluppata in Ruby On Rails, in grado di occuparsi della configurazione di Apache, Mysql e FTP Server, della gestione dei processi di Apache collegati alle varie applicazioni assegnate e di unificare i vari utenti in modo da poter gestire attraverso un'unica interfaccia le problematiche di help-desk. Inoltre è stata richiesta una particolare attenzione alle

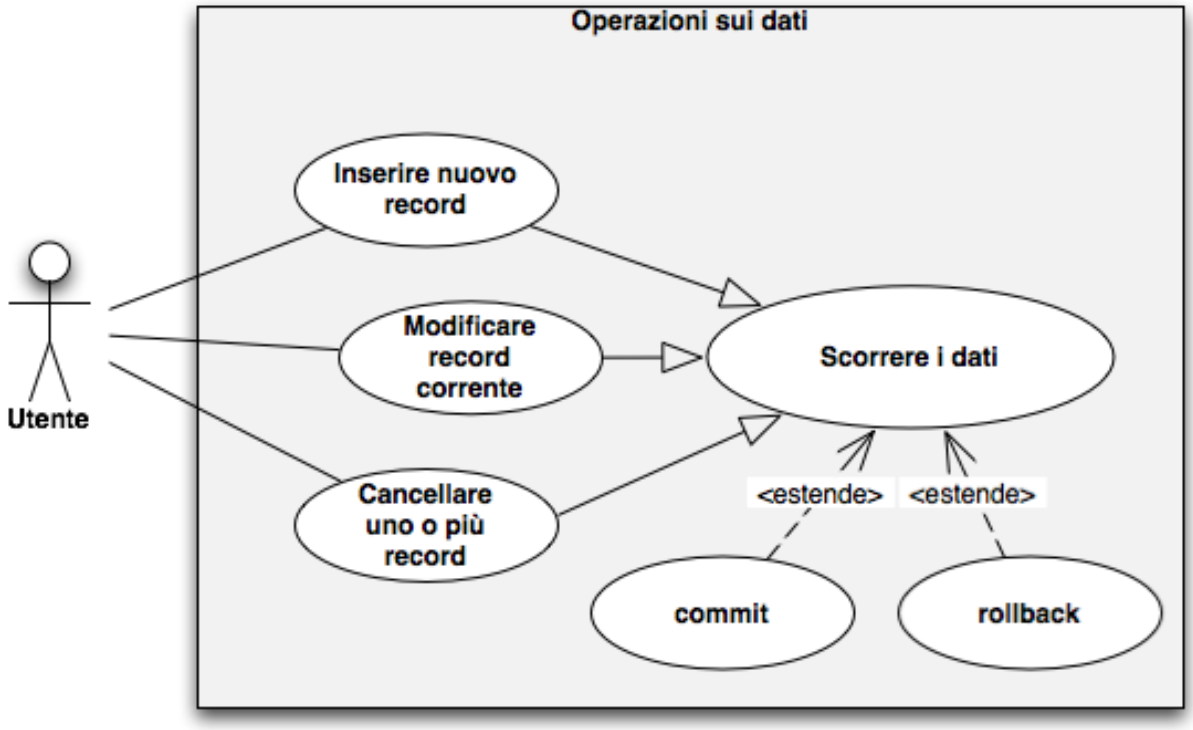
problematiche inerenti il consumo e l'utilizzo delle risorse, per cui è stato coinvolto nel progetto il tool Monit.

Diagrammi d'uso

Nel processo di definizione dei requisiti del prodotto, ci si avvale di alcuni diagrammi d'uso (usecase), come definito nello standard UML 2.1.2, con l'obiettivo di chiarire meglio il comportamento atteso del software, specialmente nell'ambito dell'interazione con l'utente. Questo permette di definire in modo più intuitivo l'elenco dei requisiti funzionali e consente di progettare i test, in fase di verifica e validazione, riferendosi meglio alle esigenze individuate durante l'analisi.
I diagrammi d'uso hanno una struttura che comprende uno o più attori colti nell'atto di interagire con il sistema o con parte di esso. Si vuole evidenziare il tipo di attività che l'attore sta eseguendo e le azioni che esso scatena. Tale struttura permette un approccio più vicino alle logiche "bottom-up", facilitando il lavoro dell'analista che riesce così a focalizzare le reali problematiche che il software è tenuto ad risolvere.
Una volta definiti i vari usecase è possibile stilare più agevolmente la lista dei requisiti.

1. Operazioni sui dati

Diagramma



Descrizione

Attori coinvolti

Utente.

Scopo e descrizione sintetica

Il sistema permette all'utente di scorrere, modificare, cancellare, inserire nuovi record nella struttura dati selezionata. La struttura dati può essere un'anagrafica cliente o una configurazione di virtualhost.

Flusso base degli eventi

È possiblile che:

- l'utente scorra i dati e scelga un'azione tra quelle esposte;

- l'utente chieda di fare commit delle modifiche confermando l'azione;
- l'utente chieda di fare rollback delle modifiche annullando l'azione.

Precondizioni

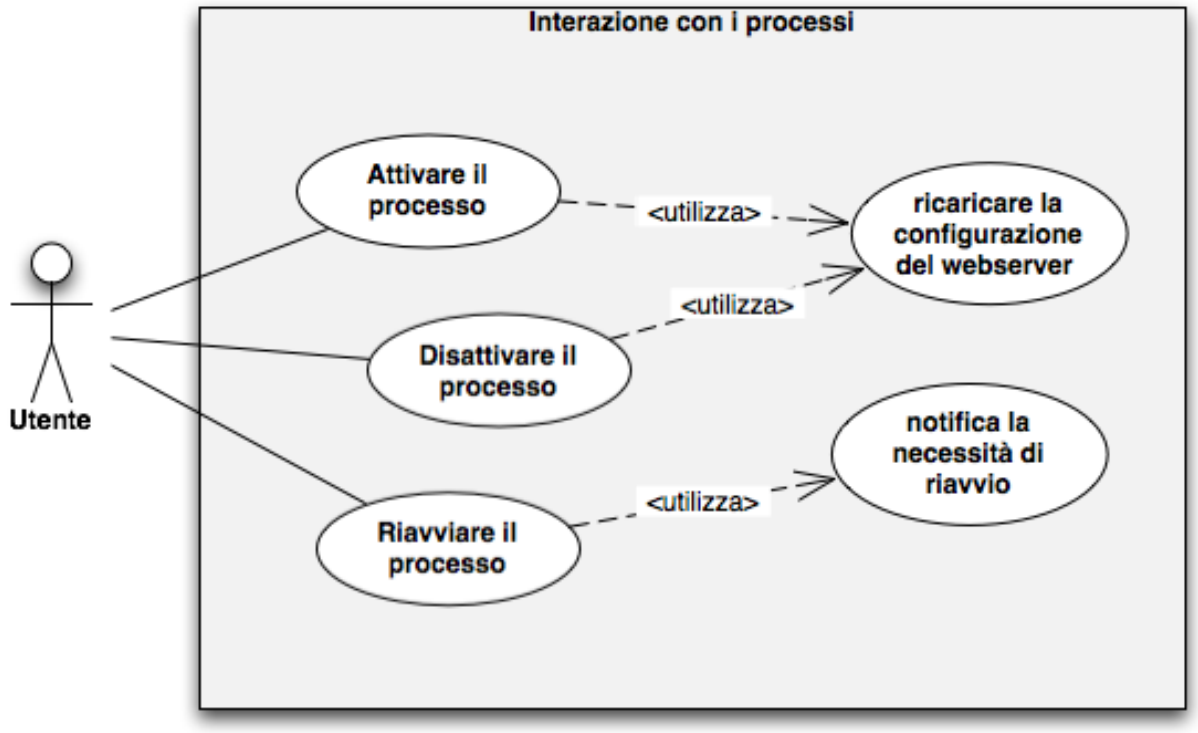
L'utente è autenticato e ha scelto l'azione da svolgere.

Postcondizioni

La base di dati rimane sempre consistente.

2. Interazione con i processi

Diagramma



Descrizione

Attori coinvolti

Utente.

Scopo e descrizione sintetica
L'utente ha la facoltà di avviare un processo legato ad una configurazione (attivare un sito), di disattivarlo, e di riavviarlo.

Flusso base degli eventi
L'utente esegue una delle operazioni esposte su di un processo.

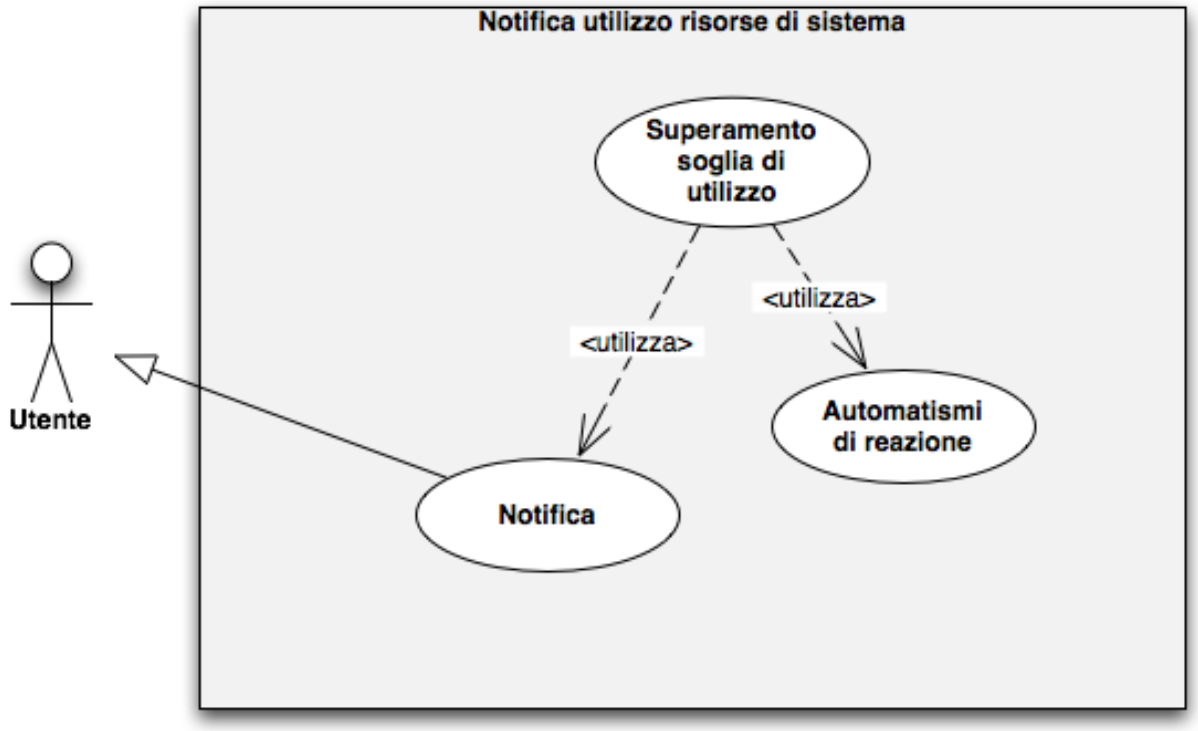
Flusso alternativo degli eventi
L'azione scelta genera un errore.

Precondizioni
L'utente è autenticato e ha scelto l'azione da svolgere, la configurazione che si vuole attivare è completa.

Postcondizioni
Il processo ha cambiato il suo stato.

3. Notifica utilizzo risorse di sistema

Diagramma



Descrizione

Attori coinvolti

Utente.

Scopo e descrizione sintetica

Se il sistema supera un certo carico, l'utente viene notificato e scattano degli automatismi come il riavvio di alcuni servizi.

Flusso base degli eventi

L'utente riceve una notifica di avvenuto superamento di una soglia di carico.

Lista dei requisiti

Segue la lista dei requisiti individuati:

Requisiti funzionali

cod	*descrizione*
RF-1	L'utente è in grado di inserire un nuovo cliente
RF-2	L'utente è in grado di modificare i dati di un cliente
RF-3	L'utente è in grado di eliminare un cliente
RF-4	L'utente è in grado di configurare un virtualhost per un cliente
RF-5	L'utente è in grado di modificare una configurazione virtualhost
RF-6	L'utente è in grado di eliminare una configurazione virtualhost
RF-7	L'utente è in grado di attivare una configurazione virtualhost
RF-8	L'utente è in grado di disattivare una configurazione virtualhost
RF-9	L'utente è in grado di ricaricare una configurazione virtualhost
RF-10	L'utente è in grado di interrogare una statistica relativa al consumo di risorse per uno specifico virtualhost
RF-11	L'utente è in grado di collegare un database Mysql ad un virtualhost
RF-12	L'utente è in grado di rimuovere un database Mysql collegato ad un virtualhost
RF-13	L'utente è in grado di collegare uno spazio FTP ad un virtualhost
RF-14	L'utente è in grado di rimuovere uno spazio FTP collegato ad un virtualhost
RF-15	L'utente è in grado di modificare le opzioni globali relative all'applicazione
RF-16	L'utente è in grado di modificare le opzioni globali relative al modulo phusion-passenger (mod_rails)
RF-17	L'applicazione è in grado di notificare l'utente in caso di

	superamento dei limiti di utilizzo delle risorse

Requisiti di qualità

RQ-1	Semplicità delle interfacce, che permetta un utilizzo anche da parte di utenti poco alfabetizzati
RQ-2	Estensibilità delle funzioni, per iterazioni future nel ciclo di vita
RQ-3	Robustezza e stabilità, che consenta un immediato impiego in ambienti di produzione

Requisiti di interfacciamento

RI-1	Il sistema deve basarsi su interfacce web
RI-2	Il sistema deve essere accessibile con Microsoft Internet Explorer 7 e con Mozilla Firefox 3

Capitolo IV – Progettazione

In questo capitolo ci addentriamo negli aspetti progettuali dell'applicazione. Essendo fortemente condizionata dal framework su cui si basa, si è ritenuto doveroso trattare a parte gli aspetti architetturali ad esso connessi.

Background: Architettura di Ruby On Rails

L'architettura dell'applicazione è fortemente influenzata dalla struttura del framework Ruby On Rails. Si tratta infatti di un pattern MVC (Model View Controller), i cui componenti possono essere schematizzati dal *diagramma 4.1*.
Alcuni benefici noti del paradigma MVC sono:

- separazione tra la Business Logic (logica applicativa) e l'interfaccia utente;
- facilità di manutenzione del codice che è strutturato in modo non ripetitivo e riutilizzabile (DRY: Don't Repeat Yourself);
- facilità di manutenzione del codice organizzato nel file system secondo il layer di riferimento.

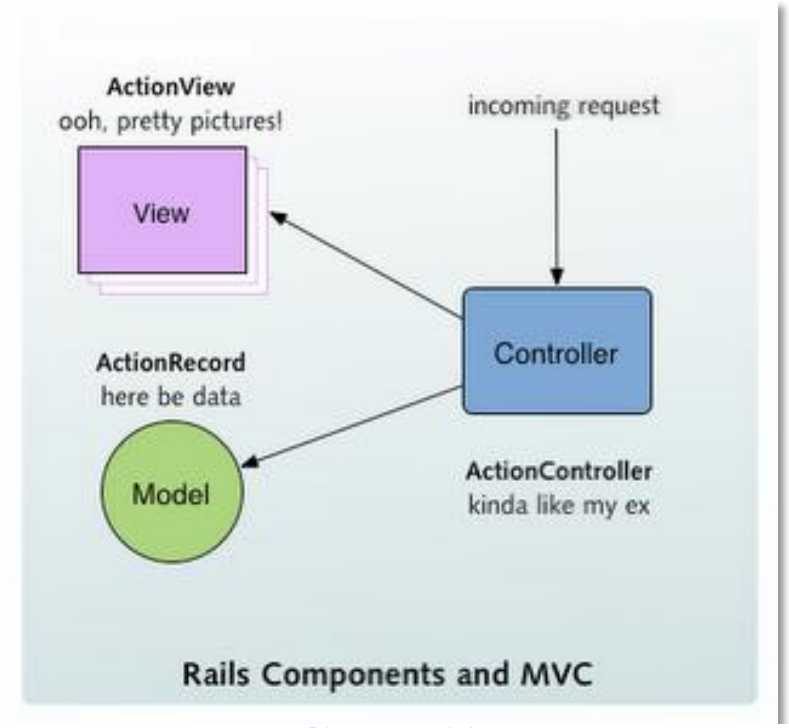


Diagramma 4.1

Model

Un "model" rappresenta l'informazione (dato) dell'applicazione e le regole per manipolarlo. Nel caso di Rails i model sono utilizzati per gestire le regole di interazione con una corrispondente tabella del database. Nella maggior parte dei casi (e nello specifico è vero per questo progetto), ad ogni tabella nel database corrisponde un model.

I models genericamente contengono il cuore della Business Logic dell'applicazione. Essi sono slegati dalla specifica versione del database sottostante. La connessione con l'RDBMS è gestita interamente dal framework per consentirne la sostituzione in maniera completamente trasparente agli altri componenti. Questa caratteristica è cruciale sotto l'aspetto della scalabilità.

View

Le "views" rappresentano le interfacce utente dell'applicazione. In Rails le views sono file HTML che racchiudono codice Ruby incaricato di effettuare

operazioni collegate solamente alla presentazione dei dati. Non vi è alcuna istruzione relativa alla logica.
Ai file HTML sono associati anche altri file, CSS e JavaScript e anch'essi includono codice Ruby per un utilizzo parametrico.
Rails supporta anche la possibilità di definire un "layout" standard da applicare a tutte le views, ottenendo buoni risultati sul piano del riutilizzo del codice.

Controller

I "controllers" forniscono una sorta di collante fra view e model. In Rails essi sono responsabili di processare le richieste provenienti dai client (browser), interrogando i models per avere i dati e trasferendoli alle views per la visualizzazione.
Possiamo dire che il controller è quel componente che si occupa di gestire il flusso di utilizzo dell'applicazione.

In realtà il pattern MVC non è un concetto così nuovo. Da circa vent'anni i progettisti software lo utilizzano con successo per la realizzazione di applicazioni con interfaccia grafica. Pian piano però si è affacciato anche fra gli sviluppatori di applicazioni web, determinando una nuova evoluzione che favorisce il disaccoppiamento e il riutilizzo del codice.
Ruby On Rails implementa il pattern MVC attraverso i propri componenti e ne favorisce l'interazione grazie a un approccio basato soprattutto sulla convenzione, piuttosto che sulla configurazione (tipica invece di Java).
I componenti coinvolti nell'MVC di Rails sono i seguenti:

ActiveRecord

È il componente che si occupa di gestire i models e che fornisce gli strumenti per collegarsi ai database. I suoi connettori supportano diversi database relazionali (Oracle, PostgreSQL, Mysql, SQLite, ecc.) . Il ruolo cruciale di ActiveRecord consiste nella sua capacità di mappare un record di una tabella

del database con un'istanza di un oggetto Ruby e i vari campi con gli attributi dell'oggetto. In altre parole, associa una tabella nel db ad una classe nell'applicazione. Questo è ciò che viene chiamato ORM (acronimo di Object-relational mapping) e consente di utilizzare il database come semplice storage per i dati, lasciando ad una struttura ad oggetti la responsabilità di implementare la logica.
Le associazioni tra le varie tabelle del database, che costituiscono la logica relazionale dell'applicazione, sono implementate in ActiveRecord con delle convenzioni. Ad esempio la dichiarazione di chiavi esterne avviene nella forma `tabellariferita_id`.
La struttura di ActiveRecord fornisce allo sviluppatore la possibilità di eseguire agilmente operazioni di tipo CRUD (Create – Read – Update – Delete), senza bisogno di appoggiarsi ad espliciti costrutti SQL.

ActionView

Costituisce la libreria che si occupa della gestione delle viste (presentation layer). Grazie ad avanzati meccanismi di riutilizzo del codice, permette l'inclusione di viste parziali, l'utilizzo di layout application-wide, e metodi cosiddetti "helper", utilizzati per generare XHTML più complesso o per fornire elementi AJAX.
Ad ogni ActionView corrisponde uno o più file, con estensione `html.erb`, che vengono interpretati e richiedono i dati al controller invocando un'azione specifica. Il forte disaccoppiamento dei componenti consente, per esempio, di utilizzare metodi del controller in viste diverse, sfruttando l'efficacia del riuso.

ActionController

Gestisce le azioni eseguibili dall'utente. Costituisce il componente che si occupa di implementare i controllers. Ogni istanza di questo oggetto (e delle classi da esso derivate, ovviamente) contengono metodi pubblici, che

corrispondono alle azioni specifiche che l'utente può compiere. Ogni azione corrisponde ad una vista. La formattazione REST degli URL di Rails permette di invocare azioni del controller direttamente attraverso le chiamate HTTP, senza ulteriore logica. Ad esempio invocando tramite metodo GET l'URL:

```
http://localhost.localdomain/railsapp/apples/show/43
```

si ottiene una chiamata al metodo `show` del controller `Apple` (classe che eredita da ActionController) cui viene passato il parametro `'43'`, id dell'elemento da recuperare.

L'architettura dei componenti di Ruby On Rails è schematizzabile come nel *diagramma 4.2*.

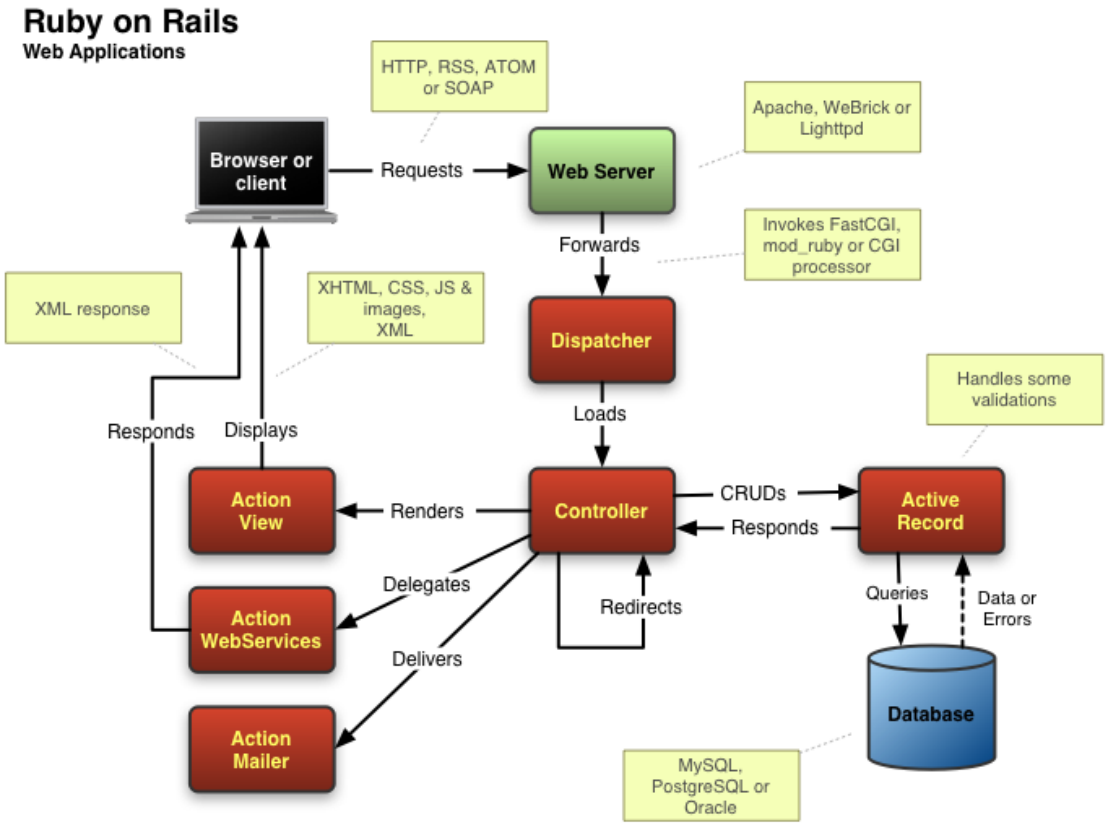


Diagramma 4.2

A questa struttura bisogna aggiungere un nutrito numero di plugins, elementi fondamentali di un'applicazione scritta in Ruby On Rails. Il framework, infatti, è estendibile nelle funzionalità principali, grazie al sistema dei plugins, pacchetti organizzati sottoforma di "gemme", ovvero archivi compressi che contengono porzioni codice che viene incluso ad ogni avvio dell'applicazione.

Diagramma delle classi

Segue un diagramma UML delle classi che, nel caso di un'applicazione Rails, fornisce la rappresentazione della struttura dei models e quindi coinvolge la maggior parte della progettazione. Per questo motivo è stata leggermente modificata la struttura del *diagramma 4.3* per adattarlo meglio a rappresentare un progetto Rails.

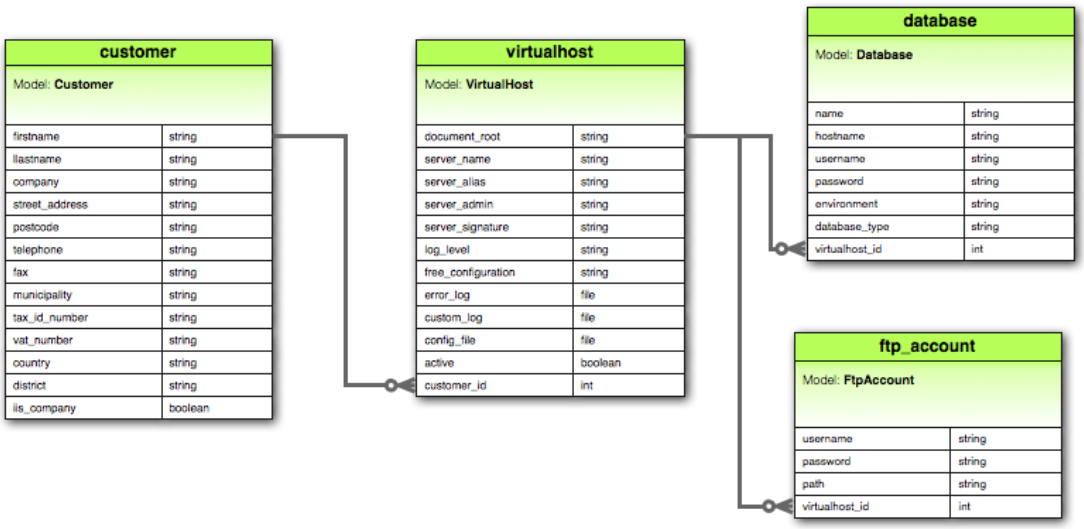


Diagramma 4.3

Descrizione dei componenti

Prima di addentrarci nel dettaglio di ogni componente, segue una lista dei plugins di Ruby On Rails che sono stati inclusi nel progetto, con una breve descrizione della loro utilità.

Plugins

Come già detto, i plugins costituiscono parti di applicazioni Rails e, talvolta, vere e proprie applicazioni indipendenti, che estendono le funzionalità del framework. Nel progetto sono stati inclusi i seguenti:

1. ActiveScaffold

Costituisce uno dei più importanti contributi della comunità al framework. Tutte le intefacce del progetto sono completamente basate su di esso. Tale plugin, genera automaticamente le views a runtime, sulla base di configurazioni presenti nei controllers. In particolare mette a disposizione:

- un'interfaccia tabellare basata su AJAX per creare, modificare e cancellare gli oggetti (operazioni CRUD);
- gestione automatica delle associazioni di ActiveRecord;
- ordinamento, ricerca e paginazione automatica dei dati;
- retrocompatibilità con i vecchi browser grazie alla cosiddetta "Graceful JavaScript degradation";
- supporto alle API RESTful (XML/YAML/JSON) per accesso ai dati tramite web service.

[h]

2. ActiveScaffoldExport

È un'estensione di ActiveScaffold, che permette l'esportazione in formato CSV dei dati rappresentati nelle tabelle. Richiede pochissima configurazione e permette di usufruire dei dati anche in altre applicazioni. Il formato CSV è il più diffuso per creare interoperabilità fra diversi software e per eseguire calcoli statistici (per esempio con elaborazioni tramite foglio di calcolo).

3. ActsAsParanoid

Implementa l'override di alcuni metodi di ActiveRecord in modo che, invece di eliminare un dato, setta un parametro `deleted_at`.
La cancellazione diviene quindi "simulata" e consente di recuperare dati altrimenti persi per sempre. Per basi di dati la cui dimensione e crescita sono prevedibili, è un'ottima soluzione. Grazie al disaccoppiamento efficace tra Models e il resto dell'applicazione, questi override rendono del tutto trasparente l'effetto sul database agli altri componenti. Infatti le query al db vengono modificate, in modo da servire i risultati che non sono marcati come eliminati.

4. ExceptionNotifier

Si tratta di un plugin delegato a inviare una mail di notifica nel caso in cui venga lanciata un'eccezione durante l'esecuzione del codice. La gravità delle eccezioni notificate è configurabile. Fondamentale in fase di sviluppo, permette di comprimere i tempi di debugging e di verifica, e costituisce un'ottima fonte di informazioni per progettare le iterazioni future nel ciclo di vita.

5. AuthLogic

È un plugin che gestisce l'autenticazione degli utenti. L'approccio usato è di mantenere il più semplice possibile la business logic e di distribuire in modo coerente al pattern MVC i vari componenti delegati alle operazioni di autenticazione.

6. Settings

Settings è un plugin che fornisce la gestione delle configurazioni globali dell'applicazione. Si può pensare ad esso come ad una tabella hash, residente nel database a cui si accede tramite model di ActiveRecord. Questo consente di non cablare all'interno del codice gli elementi di configurazione globali che possono cambiare a seconda delle varie installazioni dell'applicazione. Settings è utilizzato per registrare le configurazioni globali di Apache e di mod_rails.

Componenti

La descrizione dei singoli componenti è organizzata secondo il raggruppamento previsto dal pattern MVC.

Models

Customer:

Questo model contiene gli attibuti e i metodi relativi alla gestione dei clienti. Ogni record registra i dati anagrafici necessari per il customer care e per la gestione contabile.

Campi:

- `firstname`: stringa che contiene il nome del riferimento;
- `lastname`: stringa che contiene il cognome del riferimento;
- `company`: stringa che contiene la ragione sociale;
- `street_address`: stringa che contiene l'indirizzo;
- `postcode`: stringa che contiene il codice di avviamento postale;
- `telephone`: stringa che contiene il numero di telefono;
- `fax`: stringa che contiene il numero di fax;
- `municipality`: stringa che contiene il nome della località;
- `tax_id_number`: stringa che contiene il codice fiscale;
- `vat_number`: stringa che contiene il numero di partita iva;

- `district`: stringa che contiene la provincia;
- `country`: stringa che contiene la nazione;
- `is_company`: booleano che identifica se si tratta di un'azienda o di un privato.

Metodi:
Non sono stati sviluppati metodi specifici oltre a quelli messi a disposizione dal framework per la gestione del model.
Si rimanda alla documentazione ufficiale di Rails per l'elenco completo dei metodi messi a disposizione.

Virtualhost:
gestisce i dati relativi alla configurazione di un virtualhost di Apache. Rappresenta anche la configurazione di un'applicazione Rails ad esso associata e permette l'attivazione di un sito web. Ogni record contiene le direttive tipiche di una configurazione di Apache.
Campi:

- `document_root`: stringa che contiene il percorso sul filesystem in cui sono salvati i file dell'applicazione;
- `server_name`: stringa che contiene l'FQDN (Fully Qualified Domain Name) con cui raggiungere l'applicazione;
- `server_alias`: stringa che contiene FQDN alternativi con cui raggiungere l'applicazione;
- `server_admin`: stringa che contiene l'e-mail del webmaster del sito attivato;
- `server_signature`: stringa che contiene la configurazione della verbosità della firma del server, che compare in caso di errore;
- `log_level`: stringa che contiene la configurazione della verbosità dei log;

- `free_configuration`: stringa che contiene tutte le configurazioni "libere" relative al virtualhost. In questo campo è possibile salvare configurazioni non gestite direttamente dall'applicazione;
- `error_log`: stringa che contiene il percorso sul filesystem dove salvare il file di log degli errori generati dall'applicazione;
- `custom_log`: stringa che contiene il percorso sul filesystem dove salvare il file di log degli accessi all'applicazione;
- `active`: booleano che indica se il virtualhost è attivo o meno;
- `customer_id`: intero che rappresenta la chiave esterna verso la tabella customer.

Metodi:

- `prepare_def_values`: assegna dei valori di default ai vari campi, sulla base di informazioni recuperate dai settings globali e prepara le directory sul filesystem se ancora non esistono;
- `write_vhost_files`: scrive sul filesystem i file di configurazione di apache appena salvati sul database;
  Il file di configurazione di Apache che viene generato ha la seguente struttura:

```
# Default VirtualHost configuration
<VirtualHost :80>
#
      ServerName         www.windnet.it
      ServerAlias        windnet.it
      ServerAdmin        webmaster@windnet.it
      ServerSignature    EMail
#
      DocumentRoot       /srv/www/windnet.it/www/public
#
# Optional free configuration here
#
```

```
      LogLevel    warn
      ErrorLog    /var/log/apache2/windnet.it/www/error.log
      CustomLog   /var/log/apache2/windnet.it/www/access.log
combined

</VirtualHost>
```

- `activate`: aggiorna lo stato del virtualhost e ricarica apache per rendere attiva l'applicazione;
- `deactivate`: aggiorna lo stato del virtualhost e ricarica apache per disattivare l'applicazione;
- `reload`: ricarica apache per applicare le modifiche alla configurazione;
- `restart`: effettua un riavvio dell'applicazione Rails collegata al virtualhost (necessario in caso di modifiche al codice). Tale riavvio è descritto nel *diagramma 4.4*.

  Il riavvio consiste nella creazione di un file di testo (anche vuoto) chiamato `restart.txt` la cui presenza nella directory /tmp relativa all'applicazione, notifica a phusion passenger la necessità di riavvio uccidendo tutti i processi attivi e ricaricando il codice. Per essere sicuri che il riavvio sia avvenuto, viene eseguito un confronto tra due pid prima e dopo le operazioni di riavvio.

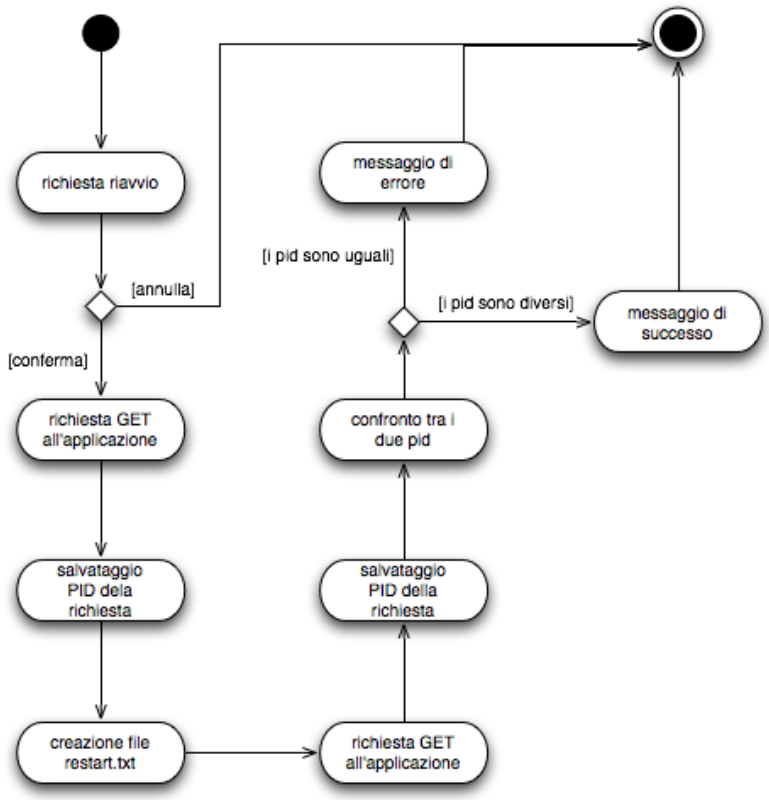


Diagramma 4.4

Database:

La classe database rappresenta un db di Mysql assegnato ad un virtualhost.

Campi:

- `username`: stringa che contiene il nome utente per l'accesso al db;
- `password`: stringa che contiene la password per l'accesso al db;
- `name`: stringa che contiene il nome del database;
- `hostname`: stringa che contiene l'host di connessione al database;
- `environment`: stringa che contiene l'environment Rails associato a quel database (Testing, Development, Production);
- `database_type`: stringa che contiene il tipo di database a cui l'applicazione si deve collegare. Per ora è disponibile solo Mysql;

- `virtualhost_id`: intero che rappresenta la chiave esterna verso la tabella virtualhost.

Metodi:

- `create_database`: metodo che si collega a Mysql, crea il nuovo database e applica le politiche di sicurezza sulla base delle credenziali specificate. Questo metodo è chiamato ogni volta che viene creato un nuovo database associato ad un virtualhost;
- `drop_database`: metodo che elimina il database da Mysql. Questo metodo è chiamato ogni volta che viene eliminato un database associato ad un virtualhost;
- `update_credentials`: metodo che aggiorna i campi username e password collegati al database e i relativi GRANT;
- `test_connection`: verifica la corretta connessione al database con le credenziali fornite;
- `get_dump`: esegue un dump del database e lo consegna in download.

Ftp_account:

Questa classe rappresenta un account FTP collegato ad un virtualhost.

Campi:

- `username`: stringa che contiene il nome utente per l'accesso FTP;
- `password`: stringa che contiene la password per l'accesso FTP;
- `path`: stringa che contiene il percorso su filesystem collegato all'account FTP. Tipicamente si assegna a questo campo il path della radice dell'applicazione Rails;
- `virtualhost_id`: intero che rappresenta la chiave esterna verso la tabella virtualhost.

Metodi:

- `create_ftp_account`: metodo che crea effettivamente l'account FTP di sistema e aggiorna la configurazione con le credenziali

specificate. Questo metodo è chiamato ogni volta che viene creato un FTP account associato ad un virtualhost;

- `delete_ftp_account`: metodo che elimina l'account FTP. Questo metodo è chiamato ogni volta che viene eliminato un FTP account associato ad un virtualhost;
- `update_credentials`: metodo che aggiorna i campi username e password collegati all'account FTP;
- `test_connection`: verifica la corretta connessione al server FTP con le credenziali fornite.

Views

Ogni view dell'applicazione è gestita dal plugin ActiveScaffold ed è creata a runtime utilizzando soltanto il file di layout. Esso costituisce un componente centralizzato cui si applicano stili CSS ed effetti JavaScript. È stato quindi molto semplice progettare l'infrastruttura delegata alla presentazione dei dati. Per gestire le voci del menu dell'applicazione, è stato creato un metodo che esegue il parsing di un file yml dov'è configurato l'albero delle varie voci. Questa struttura consente di utilizzare un file di configurazione centralizzato per gestire velocemente i menu. YAML (YAML Ain't Markup Language) è uno standard per creare file in un formato di serializzazione dei dati, ed è utilizzato per numerosi scopi dal framework Rails, come per esempio la configurazione delle connessioni ai database.
ActiveScaffold fornisce un'ottima funzionalità a livello view, ovvero la creazione automatica di tabelle "innestate" chiamate appunto nested che permettono di modificare i dati relativi alle relazioni esterne di ogni record. Nel caso dell'applicazione sviluppata, per esempio, ogni riga della tabella VirtualHost contiene un link chiamato "databases" che apre, attraverso AJAX, una sottotabella innestata contenente tutti i record della tabella Database che hanno chiave esterna uguale all'id del record corrente. Grazie ad

ActiveRecord è poi possibile visitare le relazioni nel senso inverso e quindi, attraverso tabelle nested di ActiveScaffold, accedere al record di VirtualHost cui si riferisce una riga della pagina che elenca tutti i Database.

Controllers

Ad ogni model corrisponde un controller che ne gestisce le azioni e il flusso di navigazione relativo. Come già detto, i controllers si occupano di interagire con i dati gestiti dai models per poi fornire allo strato view il risultato.
Anche in questo caso è stato utilizzato esclusivamente il plugin ActiveScaffold. Nei controllers infatti sono state inserite le configurazioni di ogni componente attraverso una chiamata al metodo statico `active_scaffold`, della classe ActiveScaffold. In questo modo è possibile definire nel controller le seguenti opzioni:

- i campi del model visualizzare nella view e in che ordine;
- le azioni CRUD abilitate;
- i criteri di ordinamento dei record;
- le relazioni con gli altri models e le conseguenti tabelle nested da attivare nella visualizzazione;
- i criteri di filtro dei record;
- l'accesso ad eventuali altre azioni aggiuntive rispetto a quelle predefinite in ActiveScaffold a cui deve corrispondere uno specifico metodo del controller.

Customer:

Sono stati abilitati tutti i campi in visualizzazione e in modifica.

VirtualHost:

Sono stati abilitati tutti i campi in visualizzazione e in modifica, e alcuni valori sono stati pre-impostati attraverso la lettura delle variabili gestite dal plugin Settings.

Alcune action aggiuntive sono state implementate nei seguenti metodi del controller:

- `activate`: chiama il metodo `activate` del model;
- `deactivate`: chiama il metodo `deactivate` del model;
- `reload`: chiama il metodo `reload` del model;
- `restart`: chiama il metodo `restart` del model.

In sostanza, quindi, i metodi aggiuntivi del controller possono essere considerati dei semplici "wrapper" che danno accesso tramite ActiveScaffold alle relative funzionalità del model.

Database:

Sono stati abilitati tutti i campi in visualizzazione e in modifica, e alcuni valori sono stati pre-impostati attraverso la lettura delle variabili gestite dal plugin Settings.

Anche in questo caso abbiamo alcuni metodi che replicano nel controller le funzionalità messe a disposizione dal model:

- `test_connection`: chiama il metodo `test_connection` del model;
- `get_dump`: chiama il metodo `get_dump` del model.

Ftp_account:

Sono stati abilitati tutti i campi in visualizzazione e in modifica.

I metodi del model riportati sono:

- `test_connection`: chiama il metodo `test_connection` del model.

Capitolo V – Codifica e deployment

In questo capitolo sono descritte le attività e gli strumenti coinvolti nella fase di codifica. Il prodotto, una volta realizzato e testato è stato installato in un ambiente di produzione dove è attualmente a disposizione del personale aziendale.

Strumenti per la codifica

La codifica è avvenuta utilizzando diversi strumenti di sviluppo, raggruppabili in due macrocategorie: gli strumenti primari, come l'IDE, i sistemi di versionamento e i software per eseguire il debugging, e quelli secondari come la documentazione delle librerie di Ruby, del framework Ruby On Rails e dei vari plugins selezionati.

Strumenti primari

La selezione degli strumenti primari è stata svolta durante la fase iniziale di studio di fattibilità, consultando altri sviluppatori di Ruby On Rails e basandosi sull'esperienza maturata negli anni.

1. Aptana Studio

   Aptana è un progetto libero che si basa sull'IDE Eclipse e fornisce numerose funzionalità per lo sviluppo. In particolare propone una collezione di plugins specializzati per la codifica di applicazioni web. Le features utilizzate nel progetto sono:

   - supporto per AJAX, JavaScript, HTML, CSS;
   - supporto per Ruby e Ruby On Rails;
   - supporto per Subversion;

   Il supporto per Rails in particolare mette a disposizione un server per eseguire in locale le applicazioni senza bisogno di installare un vero e proprio environment di sistema. Il server in questione è Mongrel, uno dei software più utilizzati per servire applicazioni Rails.

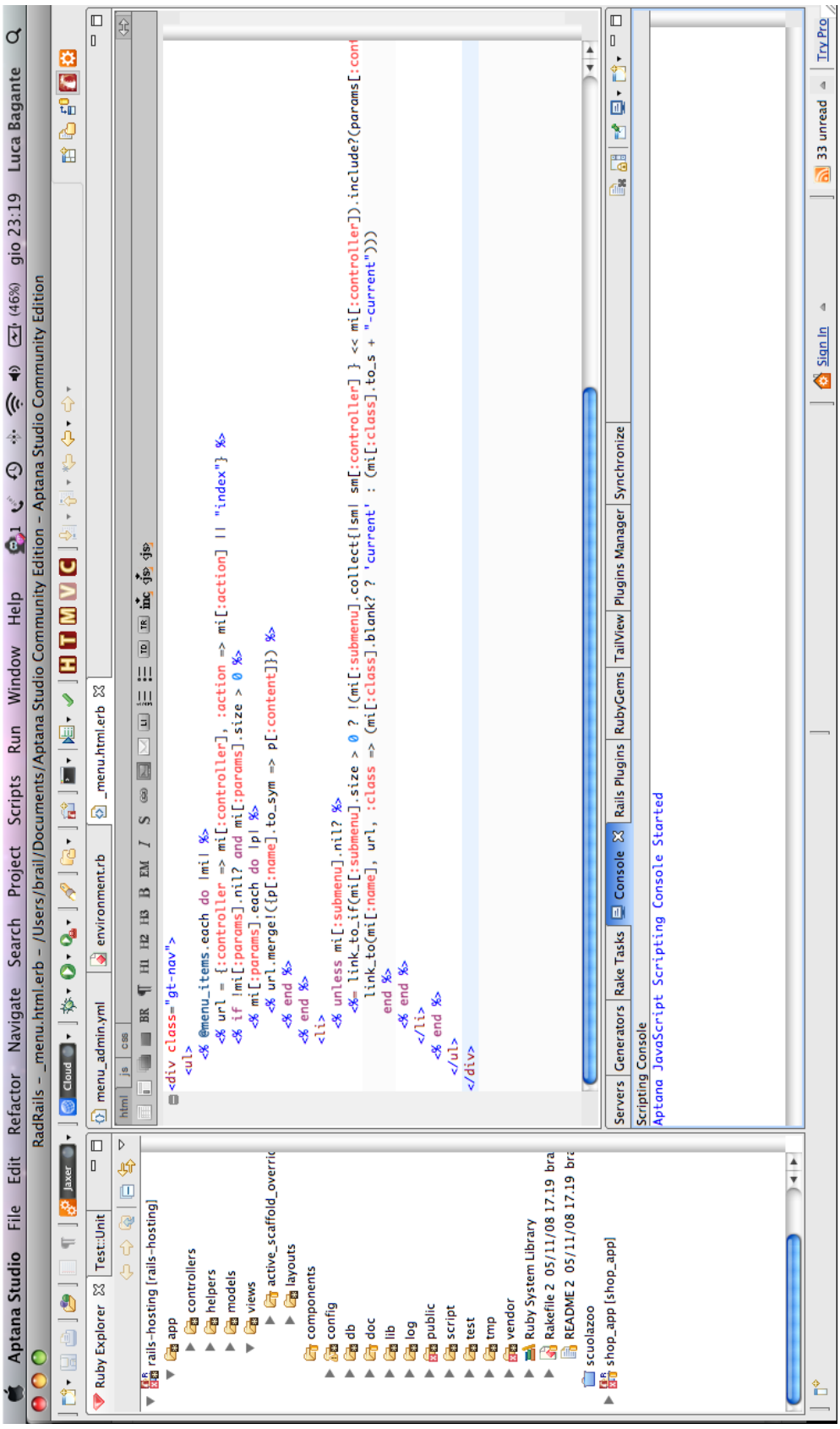
   In *figura 5.1* è riportato uno screenshot dell'IDE.

   Tra le varie caratteristiche, una che sicuramente merita di essere menzionata è la possibilità di passare da un model al relativo controller, helper o view in maniera rapidissima, grazie ad un menu dedicato. Grande enfasi è anche data al supporto per l'autocompletamento dei metodi e a quello per la generazione dell'infrastruttura dei files. Sfruttando infatti specifici script di Rails chiamati "generator", permette di creare rapidamente models e controllers a partire da skeleton predefiniti.

   Infine va menzionata l'utilissima console di output, efficace per l'immediata individuazione dei bug senza la necessità di analizzare i log. *[l]*

2. Subversion

   Si tratta di un diffusissimo sistema per il versionamento del codice. Nonostante il progetto abbia interessato un'unica risorsa umana, si è preferito versionare il codice nell'ottica di possibili ampliamenti del


Aptana Studio File Edit Refactor Navigate Search Project Scripts Run Window Help
RadRails - _menu.html.erb - /Users/brail/Documents/Aptana Studio Community Edition - Aptana Studio Community Edition
Luca Bagante
Ruby Explorer
Test::Unit
rails-hosting [rails-hosting]
app
controllers
helpers
models
views
components
config
db
doc
lib
log
public
script
test
tmp
vendor
Ruby System Library
Rakefile 2 05/11/08 17.19
README 2 05/11/08 17.19
scuolazoo
shop_app [shop_app]
menu_admin.yml
environment.rb
_menu.html.erb

<% @menu_items.each do |mi| %>
<% url = {:controller => mi[:controller], :action => mi[:action] || "index"} %>
<% if !mi[:params].nil? and mi[:params].size > 0 %>
<% mi[:params].each do |p| %>
<% url.merge!({p[:name].to_sym => p[:content]}) %>
<% end %>
<% end %>

<% unless mi[:submenu].nil? %>
<%= link_to_if(mi[:submenu].size > 0 ? !(mi[:submenu].collect{|sm| sm[:controller] } << mi[:controller]).include?(params[:cont
link_to(mi[:name], url, :class => (mi[:class].blank? ? 'current' : (mi[:class].to_s + "-current")))
end %>
<% end %>

<% end %>

Servers
Generators
Rake Tasks
Console
Rails Plugins
RubyGems
TailView
Plugins Manager
Synchronize
Scripting Console
Aptana JavaScript Scripting Console Started
Sign In
33 unread
Try Pro

team di sviluppo, oltre che per sfruttare il sistema come log delle modifiche fatte nelle varie sessioni di sviluppo.
Subversion (spesso abbreviato come SVN) è un software opensource che si pone come alternativa al suo antenato CVS mantenendone i requisiti fondamentali e che, in più, consente di:

- versionare le directory;
- eseguire commits atomici per prevenire versioni inconsistenti del repository;
- versionare anche dei metadati, dove sono registrati particolari attributi assegnati dallo sviluppatore ai files e alle directory.

3. Firebug

   Firebug è un'estensione per il browser Mozilla Firefox. Supporta lo sviluppatore nelle operazioni di analisi e debugging delle pagine web, in modo da poter evidenziare rapidamente gli effetti della codifica soprattutto dei componenti dello strato view. Le principali funzionalità sono:

   - ispezione e modifica del codice HTML;
   - analisi e modifica del codice CSS;
   - controllo del traffico di rete generato;
   - debugging Debugging con possibilità di fermare l'esecuzione;
   - evidenziazione degli errori HTML, CSS, Debugging e XML.

   [m]

Strumenti secondari

Gli strumenti che si possono definire secondari sono quelli che non costituiscono applicazioni o software direttamente coinvolti nella codifica, ma che assumono comunque un ruolo importante per supportare il processo di sviluppo. In sostanza, coincidono con i principali riferimenti informativi e normativi.

1. API di Ruby, importanti per la consultazione della core library del linguaggio;
2. API di Ruby On Rails, fondamentali per avere a portata di mano i metodi messi a disposizione dai vari componenti del framework;
3. API di ActiveScaffold;
4. progetto e documentazione del corso di Ingegneria del Software.

Ambiente di deployment

Lo sviluppo è avvenuto utilizzando degli strumenti e un ambiente di test personali. Grazie alle funzionalità complete messe a disposizione dall'IDE, le risorse si sono rivelate sufficienti. Una volta completato il ciclo del progetto, si è passati alla fase di deployment, con il trasferimento del codice e delle applicazioni nell'ambiente di produzione, all'interno dell'infrastruttura di Windnet. L'applicazione è stata installata in un server DELL dedicato, le cui caratteristiche sono le seguenti:

Hardware

- CPU Quad Core Intel® Xeon® E5405, 2X6MB Cache, 2.0GHz, 1333MHz FSB;
- 16GB Memory, 667MHz;
- 2 Hard Disk 300GB, SAS, 3.5-inch, 15,000 rpm (hot-plug).

Software

- Sistema Operativo Ubuntu Linux 8.04 LTS (Long Term Support) Server Edition; *[k]*
- Apache httpd 2.2.8;
- Mysql server 5.0.51;
- Vsftpd 2.0.6;

- Monit 4.8.1.

Capitolo VI – Verifica

La fase di verifica del processo di sviluppo è stata eseguita "a vista". Trattandosi di un progetto con una sola risorsa è stato infatti molto semplice controllare l'aderenza al piano di progetto.
La validazione del codice, invece, ha richiesto maggiore sforzo. Un ottimo supporto è stato dato dallo stesso linguaggio Ruby e dal framework Rails, che forniscono numerosi strumenti a supporto.

Test e debugging

Ruby utilizza un framework per il testing conosciuto come `Test::Unit`, simile a xUnit di altri linguaggi di programmazione, per eseguire le applicazioni per il test. Troviamo implementati quattro importanti concetti:

1. le asserzioni, ovvero linee di codice che valutano ogni espressione e gestiscono il risultato sulla base di un valore;
2. i test, cioè metodi il cui nome inizia con il prefisso `test_` e contiene un certo numero di asserzioni;

3. i test case, sottoclassi di `Test::Unit::TestCase`, che contengono un insieme di metodi di test progettati per testare aree funzionali delle applicazioni;
4. una test suite, costituita da un insieme di test case.

In Rails, ci sono tre tipi differenti di test:

1. unit test, che è utile per testare i models;
2. functional test, utilizzato per testare i controllers;
3. integration test, che testa l'interazione ad alto livello tra i controllers.

Normalmente gli sviluppatori di software distinguono tra codice in esecuzione in ambiente di sviluppo da quello eseguito in un ambiente produttivo. Nel primo caso sono implementate attività di debugging, logging, che caricano il programma di overhead non necessari.

I test realizzati per il progetto si inseriscono in questo scenario. Grazie anche alla semplicità architetturale dell'applicazione sviluppata, è stato possibile realizzare con rapidità i test considerati necessari per superare i controlli di qualità.

La fase di debugging che è derivata dall'esecuzione dei test è rappresentabile con il *diagramma 6.1*.

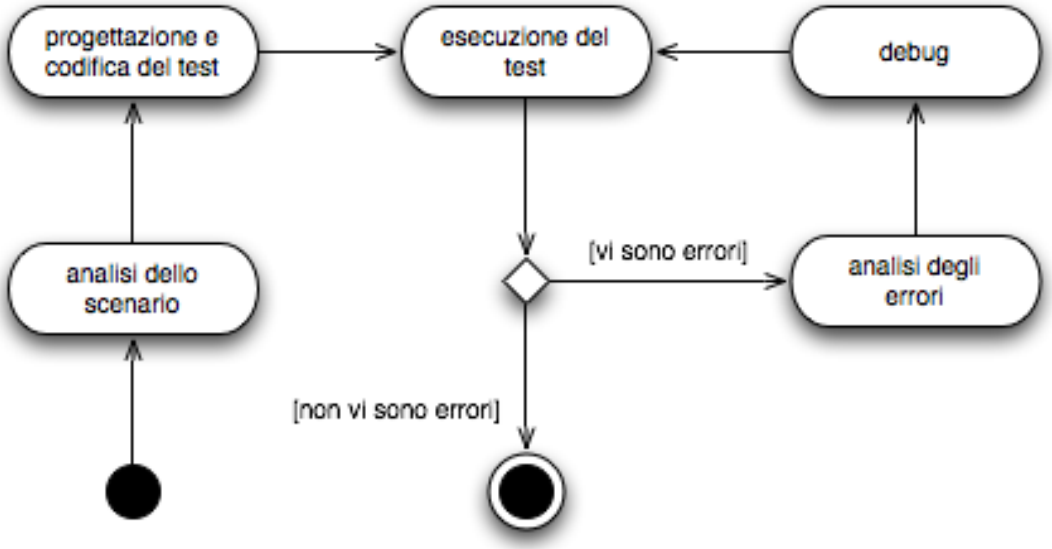


Diagramma 6.1

Capitolo VII – Conclusioni

Esperienza di stage

Il progetto di stage presso Windnet s.r.l. si è svolto secondo le aspettative. Il buon rapporto umano, già presente prima dell'inizio dello stage, ha permesso di entrare da subito in sintonia con l'azienda. I briefing con il sig. Stievano, tutor interno e CTO di Windnet, hanno permesso di mettere a fuoco in brevissimo tempo gli obiettivi gestionali cui il progetto tendeva, sia sul piano delle esigenze tecniche, sia su quello più legato al customer care.

Il progetto ha richiesto competenze che già possedevo sia riguardo la tecnologia di Ruby On Rails, sia riguardo le problematiche di amministrazione di sistema. Questo ha permesso di selezionare la strada più breve per arrivare ad un prodotto utilizzabile, seppur alla prima iterazione del suo ciclo di vita.

Ritengo che il progetto si sia dimostrato un ottimo esempio di applicazione al mondo del lavoro delle conoscenze maturate in ambito accademico. L'approccio pragmatico alle problematiche di ingegneria del software hanno consentito uno svolgimento fluido delle attività, mentre le competenze acquisite nei corsi di reti e sistemi operativi, oltre all'esperienza diretta di cui già disponevo, hanno fornito il valore di cui il prodotto necessitava. Nonostante ciò, lo stage si inserisce in un contesto più ambizioso, teso ad

automatizzare numerosi processi aziendali di Windnet e pertanto saranno necessari in futuro ulteriori interventi di estensione e miglioramento delle funzionalità, anche rispetto all'applicazione sviluppata in questo progetto.
Concludendo, l'esperienza di stage è stata sicuramente stimolante sul piano professionale, in quanto mi ha permesso di entrare ancora più in profondità nelle logiche organizzative di un'azienda che opera nel campo dell'IT.
La necessità che l'azienda ha espresso in merito all'espansione dei servizi offerti anche in ambito Ruby On Rails, fa emergere in modo evidente come la domanda per il supporto di queste nuove tecnologie di sviluppo stia condizionando sempre di più il mercato e il comparto degli Internet Service Providers. In quest'ottica credo che sia interessante, per il mondo accademico, l'opportunità di inserire maggiore materiale didattico nell'offerta formativa proposta. Ciò potrebbe sicuramente interessare numerosi corsi, in modo trasversale. Primo fra tutti, il corso di Ingegneria del Software potrebbe integrare valutazioni e analisi sulle nuove metodologie di sviluppo di cui il framework Rails è pioniere, il cosiddetto approccio "Agile". La comunità di sviluppatori tende sempre di più a questo modello in quanto è estremamente efficace per definire in modo esaustivo le esigenze della committenza, sconvolgendo le logiche tradizionali che caratterizzano i processi di analisi e progettazione del software attualmente studiati. Altro corso interessato potrebbe essere Tecnologie Web, in cui l'inserimento di un'analisi approfondita di un pattern MVC, come quello proposto da Rails, fornirebbe agli studenti la possibilità di vedere una concreta applicazione delle tecnologie coinvolte. Per finire, il corso di Sistemi Operativi, potrebbe includere una parte progettuale, in affiancamento alle lezioni frontali, dove mostrare agli studenti alcuni esempi circa l'infrastruttura sistemistica alla base dei servizi web. Su quest'ultima considerazione va detto che, nella mia esperienza professionale, ho trovato enormi vantaggi nell'approccio pragmatico di questi corsi, che mi hanno fornito un inestimabile bagaglio culturale che ora sono in grado di applicare nelle più disparate situazioni

lavorative. Ciò è stato possibile, tuttavia, grazie al fatto che ho sempre affiancato all'attività di studio quella lavorativa, traendone immediatamente i benefici professionalizzanti, applicando l'indomani ciò che era stato studiato oggi. Non credo che questo sia il caso di tanti altri studenti che, pertanto, troverebbero motivazione e giovamento nell'applicazione più concreta dei concetti studiati.

Glossario

- **Agile:**
  Agile è un processo di sviluppo che sostituisce il classico waterfall lifecycle nella produzione del software. In estrema sintesi Agile spezzetta in più cicli quello che di solito viene fatto in un unico ciclo. Quindi le fasi di Analisi, Progettazione, Programmazione e QA Testing e Rilascio vengono svolte solo per un piccolo gruppo di funzionalità e ripetute per i cicli che seguono, su altri piccoli gruppi di funzionalità. Ognuna di queste fasi, quasi sempre di carattere incrementale, è, nella terminologia Agile (metodo SCRUM), chiamato sprint.
- **AJAX:**
  Asynchronous JavaScript and XML, è una tecnica di sviluppo web per creare applicazioni web interattive. L'intento di tale tecnica è quello di ottenere pagine web che rispondono in maniera più rapida, grazie allo scambio in background di piccoli pacchetti di dati con il server, cosicché l'intera pagina web non debba essere ricaricata ogni volta che l'utente effettua una modifica. Questa tecnica riesce, quindi, a migliorare l'interattività, la velocità e l'usabilità di una pagina web.

- **AST Nodes:**
  Abstract Syntax Tree Nodes, sono i nodi di un Albero Sintattico Astratto, ovvero una rappresentazione (semplificata e con struttura ad albero) di un codice sorgente scritto in uno specifico linguaggio di programmazione. *[n]*
- **Business Logic:**
  con il termine Business Logic ci si riferisce a tutta quella logica applicativa che rende operativa un'applicazione. Il business logic racchiude in sÈ regole cosiddette di "business", piuttosto che regole ed elementi legati alla visualizzazione delle informazioni (Vista o interfaccia grafica) o alla memorizzazione dei dati (es. database, ecc.). *[n]*
- **CRUD:**
  Create, Read, Update, Delete, sono le quattro operazioni fondamentali che vengono eseguite nei confronti di dati persistenti. In particolare, nei database relazionali, alla *Create* corrisponde una query di INSERT, alla *Read* una query di SELECT, alla *Update* una query di UPDATE e alla *Delete* una query di DELETE. *[n]*
- **Deployment:**
  consegna al cliente, con relativa installazione e messa in funzione, di un'applicazione o di un sistema software. Lo si può considerare come una fase del ciclo di vita del software che conclude lo sviluppo e dà inizio alla manutenzione. *[n]*
- **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. *[n]*

- **Framework Full-stack:**
  è un Framework in cui i componenti che fanno parte dello stack sono integrati in modo tale che i collegamenti fra di essi non devono essere impostati manualmente.
- **Hosting:**
  servizio che consiste nell'allocare su un server web le pagine di un sito web, rendendolo cosí accessibile dalla rete Internet.
- **Housing:**
  concessione in locazione ad un utente di un intero server connesso a internet. Normalmente i server vengono ospitati in webfarm in cui si garantisce un'attenta gestione degli aspetti hardware, software ed infrastrutturali, anche la corrente è gestita e garantita grazie a gruppi di continuità così anche in caso di blackout il server continua a funzionare. *[n]*
- **Internet Service Provider:**
  struttura commerciale o organizzazione che offre agli utenti (residenziali o imprese) accesso a Internet con i relativi servizi. Per estensione si usa il termine Internet Service Provider anche per fornitori di servizi Internet diversi dall'accesso.
- **Maintainer:**
  uno dei servizi principali è la registrazione di nomi a dominio DNS. In Italia, gli ISP che forniscono questo tipo di servizio si chiamano anche Maintainers o MNT. Questa terminologia è stata introdotta dal NIC, il centro che gestisce il ccTLD .it
- **Plugin:**
  programma non autonomo che interagisce con un altro programma per ampliarne le funzioni. *[n]*

- **RDBMS:**
  Relational Database Management System. Indica un database management system basato sul modello relazionale, ed è stato introdotto da Edgar F. Codd. Un database management system è un sistema software progettato per consentire la creazione e manipolazione efficiente di database (ovvero di collezioni di dati strutturati) solitamente da parte di più utenti. *[n]*
- **REST:**
  Representational State Transfer. È un tipo di architettura software per i sistemi di ipertesto distribuiti come il World Wide Web. REST si riferisce ad un insieme di principi di architetture di rete, i quali delineano come le risorse sono definite e indirizzate. Il termine è spesso usato nel senso di descrivere ogni semplice interfaccia che trasmette dati su HTTP senza un livello opzionale. *[n]*
- **Superserver inetd:**
  è un demone presente su numerosi sistemi UNIX che controlla i servizi Internet.
- **VirtualHost:**
  pratica di mantenere più di un server su una singola macchina, differenziati dal loro host name. Infatti è spesso desiderabile per l'azienda che condividono un server Web di avere il proprio dominio, che sia accessibile da un nome di dominio del tipo www.azienda1.com o www.azienda2.it, senza richiedere all'utente di conoscere altre informazioni a riguardo. *[c]*
- **Virtualizzazione:**
  creazione di una versione virtuale di una risorsa normalmente fornita fisicamente. Qualunque risorsa hardware o software può essere virtualizzata: sistemi operativi, server, memoria, spazio disco,

sottosistemi. Un tipico esempio di virtualizzazione è la divisione di un disco fisso in partizioni logiche. *[n]*

- **Wrapper:**
  porzione di codice che intercetta le chiamate ad una libreria e le redireziona verso un'altra.

Riferimenti

Riferimenti Bibliografici

[1] An Introduction to Ruby by Colin Steele – O'Reilly Media - 25/10/2001

[2] Agile Web Development with Rails, 3rd Edition by Sam Ruby, Dave Thomas, David Hansson - The Pragmatic Programmers LLC. - 07/01/2009

Riferimenti sul Web

[a] http://www.ruby-lang.org

[b] http://www.rubyonrails.org

[c] http://www.apache.org

[d] http://www.mysql.com

[e] http://www.modrails.org

[f] http://www.rubyenterpriseedition.com/

[g] http://blog.beaver.net/2005/03/ruby_gc_and_copyonwrite.html

[h] http://www.activescaffold.com

[i] http://vsftpd.beasts.org/

[j] http://mmonit.com/monit/

[k] http://www.ubuntu.com

[l] http://www.aptana.com

[m] http://www.getfirebug.com

[n] http://it.wikipedia.org/

Rigraziamenti

Ringrazio in primo luogo i miei professori e in particolare la dott.ssa Brent Venable, per avermi dato preziosi strumenti da applicare ogni giorno nel mondo del lavoro.
Grazie anche a Windnet s.r.l. e, in particolare, a Roberto Stievano, per avermi dato la possibilità di concludere con questo progetto il mio percorso formativo.
Un pensiero speciale è per i miei compagni di studi, a cui va molto del merito dei miei risultati. Saper creare un ambiente di studio produttivo è ardua impresa.
Un grazie di cuore ai miei soci, che per amicizia si sono fatti carico anche delle mie responsabilità per consentirmi di concludere il mio percorso universitario.
Per finire ringrazio tutti coloro che mi hanno sostenuto in questi anni. Riuscire a coniugare attività di studio e lavoro non è facile, ma tutto sommato le soddisfazioni hanno pagato il sacrificio. In particolare ringrazio i miei genitori e tutta la mia famiglia. Hanno sempre saputo assecondarmi (loro malgrado) e motivarmi nei momenti di sconforto.
Un grazie particolare a Sara, per aver saputo aspettare così a lungo.