Progetto SIGEOL

Specifica Tecnica

v0.1.0

Redazione:

21 gennaio 2009

quixoft.sol@gmail.com

Verifica:	Freo Matteo
Approvazione:	Scortegagna Carlo
Stato:	Formale
Uso:	Esterno
Distribuzione:	QuiXoft
	Rossi Francesca
	Vardanega Tullio
	Conte Renato

Sommario

Documento contenente la specifica tecnica per il progetto "SIGEOL" commissionato dalla prof. Rossi Francesca.

Indice

1 Introduzione

1.1 Scopo del documento

Il presente documento denominato Specifica tecnica ha lo scopo di mostrare la struttura del progetto *SIGEOL* e descrivere i componenti che fanno parte.

1.2 Scopo del prodotto

Il progetto sotto analisi, denominato *SIGEOL*, si prefigge di automatizzare la generazione, la gestione, l'ottimizzazione e la consultazione degli orari di lezione.

1.3 Glossario

Le definizioni dei termini specialistici usati nella stesura di questo e di tutti gli altri documenti possono essere trovate nel documento Glossario al fine di eliminare ogni ambiguità e di facilitare la comprensione dei temi trattati. Ogni termine la cui definizione è disponibile all'interno del Glossario verrà marcato con una sottolineatura.

2 Definizione del prodotto

2.1 Metodo e formalismo di specifica

Lo strumento principale nel redarre la specifica tecnica sarà il linguaggio UML, che permetterà di realizzare i diagrammi delle classi, di sequenza, di collaborazione e di attivita'.

La decomposizione architetturale utilizzata sara' di tipo Top-down. E' prevista una descrizione generale dell'architettura del sistema, alla quale seguiranno le specifiche dettagliate dei suoi componenti.

Per semplificare la progettazione, si utilizzaranno i seguenti pattern:

MVC Il pattern MVC (Model View Controller)si basa sulla separazione tra i componenti software del sistema, che gestiscono il modo in cui presentare i dati e i componenti che gestiscono i dati stessi.

Facade Permette attraverso un'interfaccia più semplice, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro, nonché a blocchi di codice complessi.

Rest Representational state transfer (REST) 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.

Convention Over Configuration Convention Over Configuration è un paradigma di programmazione che prevede configurazione minima (o addirittura assente) per il programmatore che utilizza un framework che lo rispetti, obbligandolo a configurare solo gli aspetti che si differenziano dalle implementazioni standard o che non rispettano particolari convenzioni di denominazione o simili.

DRY Le definizioni devono essere poste una volta soltanto

View Helper Questo pattern disaccoppia il Business Logic dallo strato View, il che facilita la manutenibilit'a. Aiuta a separare, in fase di sviluppo, la responsabilt'a del web designer e dello sviluppatore.

Il sistema verrà implementato utilizzando Ruby on rails, un framework la cui architettura è fortemente ispirata al paradigma Model-View-Controller. Oltre a veicolare lo sviluppo di applicazioni secondo il pattern MVC, il RESTful Rails impone un'ulteriore disciplina nella codifica che garantisce maggiore compattezza, migliore leggibilità, "pretty-urling" e semplicità nella costruzione di API.

2.2 Presentazione dell'architettura generale del sistema

Per presentare l'architettura generale del sistema *Sigeol* si utilizzerà il pattern Model-View-Controller(MVC). In questo modello i ruoli di presentazione, controllo ed accesso ai dati vengono affidati a componenti diversi e sono tra di loro disaccoppiati.

View

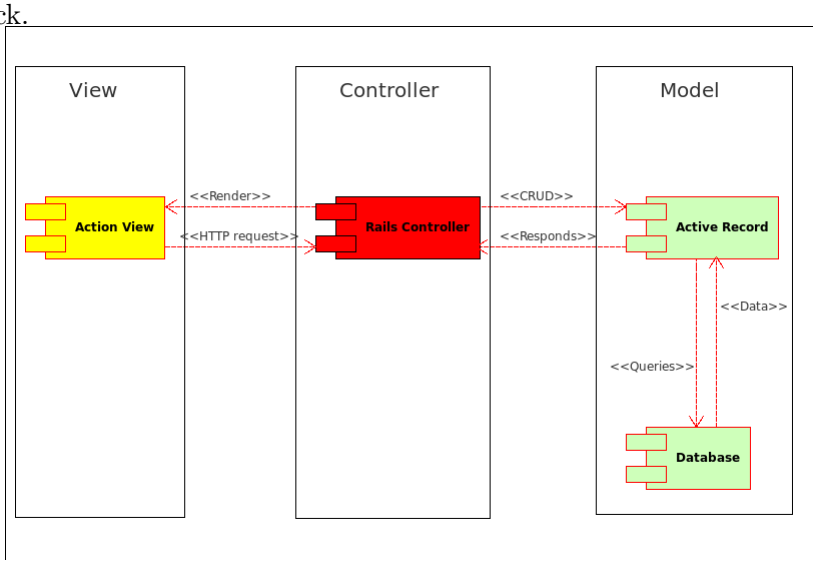
E' il primo livello che si incontra e contiene i componenti che costituiscono l'interfaccia grafica, tramite la quale l'utente interagisce con il sistema *Sigeol*. La View delega al Controller l'esecuzione dei processi richiesti dall'utente dopo averne catturato gli input e la scelta delle eventuali schermate da presentare. ActionView gestisce l'aspetto delle pagine da restituire al client. Sono per lo piu' file .rhtml che contengono delle direttive scritte in Ruby immerse nel codice XHTML.

Model

Definisce i dati e le operazioni che possono essere eseguite su questi. Quindi definisce le regole di business per l'interazione con i dati, esponendo alla View ed al Controller rispettivamente le funzionalità per l'accesso e l'aggiornamento. Viene gestito da una libreria chiamata ActiveRecord, che si occupa di tutta la logica di business e che permette l'accesso a numerosi DataBase bastati su SQL. Con questa libreria si stabilisce un collegamento tra le classi scritte in Ruby e le tabelle del DB.

Controller

Questo componente ha la responsabilità di trasformare le interazioni dell'utente della View in azioni eseguite dal Model. Ma il Controller non rappresenta un semplice ponte tra View e Model. Realizzando la mappatura tra input dell'utente e processi eseguiti dal Model e selezionando le schermate della View richieste, il Controller implementa la logica di controllo dell'applicazione. Gestisce le richieste del browser e facilita la comunicazione fra Model e View. Tutti i controller creati dall'utente ereditano da ActionController. L'ApplicationController raccoglie funzionalita' condivise nell'intera applicazione. In particolare verranno inserite le azioni CRUD (Create-Read-Update-Delete) per l'oggetto e le varie funzioni che possono servirci per la nostra applicazione. I Controller sono gestiti dal componente ActionController che fa parte del pacchetto di moduli rails chiamato Action Pack.



3 Descrizione dei singoli componenti

3.0.1 Parte Controller

Action Controller

Il modulo che gestisce la parte controller è contenuto all'interno del framework Ruby on rails ed è chiamato Action Controller. Questo, assieme all'Action View(vedi Parte view) è integrato all'interno di un pacchetto chiamato Action Pack.

Ogni controller è una normale classe, ed ogni metodo pubblico definito in questa classe corrisponde ad un'azione specifica. Ad ogni azione definita corrisponde una vista.

Ruby on Rails riconosce il tipo di richiesta pervenuta codificando le informazioni all'interno di un URL e si serve di un componente chiamato Rounting, per determinare l'azione cui deve essere sottoposta la richiesta. Il componente **Routing** traccia una mappatura che permette a Rails di collegare gli URL esterni e l'azione contenuta in una determinata classe Controller.

Ad esempio, Dato un URL nel formato *nomecontroller/azione/id*, viene identificato il Controller *nomecontroller* e viene istanziato. A questo punto l'oggetto richiama il metodo con nome *azione* e con parametro *id*. Il Controller infine cercherà di visualizzare un template con lo stesso nome dell'azione.

3.0.2 Parte View

Action View

Il modulo Action View contenuto nel framework Ruby on Rails, offre meccanismi avanzati per il riutilizzo del codice, tramite l'uso di viste e di metodi helper pensati per generare ad esempio pagine XHTML. I metodi helper sono semplici metodi pubblici di una classe Controller.

3.1 Relazioni d'uso di altre componenti

3.2 Interfacce con e relazioni di uso da altre componenti

3.3 Attività svolte e dati trattati

4 Stime di fattibilità e di bisogno di risorse

5 Tracciamento della relazione componenti-requisiti

Diario delle modifiche

Data	Versione	Modifica
20 gennaio 2009	0.1.0	Creazione dell'indice