



UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN
INGEGNERIA INFORMATICA

Studio di Yii PHP Framework
per lo sviluppo di applicazioni web

RELATORE: PROF. GIORGIO MARIA DI NUNZIO

LAUREANDO: LIVIO BROCCA

Anno Accademico 2013/2014

INDICE

1. INTRODUZIONE	1
2. Yii PHP Framework.....	3
2.1 Il Progetto	3
2.2 Il Framework	4
2.2.1 PRADO Framework	5
2.2.2 jQuery Framework	5
2.3 PHP - Hypertext Preprocessor	6
2.3.1 Cenni Storici	7
2.4 Architettura MVC – Model View Controller	8
2.5 Supporto al Database	10
2.5.1 SQL - Structured Query Language.....	11
3. Sviluppo di un’applicazione con Yii.....	13
3.1 Progettazione del Database.....	13
3.1.1 Requisiti Strutturati	13
3.1.2 Modello Concettuale - Schema E-R	14
3.1.3 Modello Logico: Relazionale	15
3.2 Generazione dell’applicazione	15
3.3 Connessione al Database.....	16
3.4 Implementazione delle operazioni CRUD	17
3.4.1 Abilitazione Gii - Generatore di Codice ad Interfaccia Web.....	17
3.4.2 Generazione del Model User	18
3.4.3 Generazione del codice CRUD	21
3.4.4 Accesso alle pagine CRUD	25
4. CONCLUSIONI	27
APPENDICE	29
BIBLIOGRAFIA	31

ELENCO DELLE FIGURE

Figura 2.1 : Struttura statica di un'applicazione Yii	9
Figura 2.2 : Workflow tipico di un'applicazione Yii.....	9
Figura 3.1 : Schema E-R.....	14
Figura 3.2 : Schema Logico Relazionale.....	15
Figura 3.3 : Model Generator	18
Figura 3.4 : CRUD Generator	21
Figura 3.5 : List Comuni.....	25
Figura 3.6 : Manage Comuni	26
Figura 4.1 : PHP Framework Perfomance Comparison	27

ELENCO DELLE STRUTTURE

Struttura 3.1 : Lista ad Albero dei File di una Yii Web Application Base..... 15

ELENCO DEI LISTATI DI CODICE

Codice 3.1 : dbConnection main.php.....	16
Codice 3.2 : GiiModule main.php.....	17
Codice 3.3 : Model persone.php.....	18
Codice 3.4 : Controller PersoneController.php.....	21
Codice 3.5 : View Persone\View.php.....	24
Codice A.1 : Codice SQL struttura Database.....	29

1. INTRODUZIONE

Lo sviluppo di applicazioni web risulta molto facilitata se si utilizza un framework. La scelta del framework però deve essere considerata un processo delicato, poiché oltre a trattarsi di confrontare funzionalità e caratteristiche, significa scegliere quello che più si adatta alle proprie conoscenze o quello che si ritiene migliore per il tipo di sito o servizio che si desidera progettare.

Il framework per applicazioni web (web application framework) è un framework software progettato per supportare lo sviluppo di siti web dinamici, applicazioni e servizi web. Lo scopo del framework è quello di alleggerire il lavoro associato allo sviluppo delle attività più comuni di un'applicazione web. Molti framework forniscono ad esempio delle librerie per l'accesso alle basi di dati, per la creazione di template HTML o per gestire la sessione dell'utente. Uno dei principi fondamentali è riassunto dall'acronimo DRY (Don't Repeat Yourself), nel senso che viene fortemente consigliata l'adozione di tecniche di riuso di codice.

Sinteticamente si tratta di software che comprendono in un unico pacchetto l'intero set di librerie e strumenti utili alla progettazione di un sito, un'applicazione web o qualsiasi altro servizio online sia necessario creare con PHP o con un altro linguaggio di programmazione.

La tesina si focalizza sullo studio di un framework relativamente recente, Yii PHP framework ¹, nato come soluzione ad alcuni inconvenienti del framework PRADO, quando sul mercato esistevano già numerose valide alternative come CodeIgniter, CakePHP, Zend, e tanti altri che consentivano di programmare ad oggetti (secondo il modello MVC) scrivendo poco codice e sfruttando le funzionalità già pronte per l'utilizzo.

Yii, proposto come un framework open source ad alte prestazioni scritto per versioni di PHP uguali o superiori alla versione 5.1, ottimo per sviluppare applicazioni per il Web 2.0, sta divenendo velocemente uno dei framework PHP più utilizzati.

¹ <http://www.yiiframework.it/blog/>

2. Yii PHP Framework

2.1 Il Progetto

Il Progetto Yii ², pronunciato come “Yee” o [ji:] ed acronimo di “Yes It Is!”, nasce il 1° gennaio 2008, dall’ideatore Qiang Xue, con l’esigenza di creare un Framework, basato sull’architettura MVC, capace di risolvere alcuni problemi riscontrati nel Progetto PRADO, e dopo un primo sviluppo durato dieci mesi che produsse il rilascio in versione beta trovò successivamente l’ufficialità con la prima versione, rilasciata sotto New BSD License, il giorno 3 dicembre 2008.

Le principali caratteristiche e funzionalità di Yii PHP Framework, che lo hanno reso così rapidamente competitivo, sono le seguenti:

- *Model-View-Controller (MVC) design pattern* adotta una collaudata architettura MVC che consente una reale e netta separazione tra models, views e controllers;
- *Database Access Objects (DAO), Query Builder, Active Record, DB Migration* Consente agli sviluppatori di gestire il modello dei dati del database (SQLite, MySQL, PostgreSQL e Oracle) come oggetti evitando di scrivere ripetitivamente istruzioni SQL;
- *Form input and validation* rende la creazione di form estremamente semplice e sicura. Viene fornito con una serie di validatori, numerosi metodi di supporto e widgets per semplificare l’inserimento dei dati e la loro convalida;
- *AJAX-enabled widget* ovvero integrazione con jQuery. Yii viene fornito con una serie di widget AJAX (come ad esempio l’autocomplete, treeview e data grid), che permettono di realizzare un’interfaccia utente altamente efficiente e versatile in modo estremamente semplice;
- *Autenticazione ed autorizzazione* poiché è dotato di un supporto per l’autorizzazione e l’autenticazione tramite hierarchical role-based access control (RBAC);
- *Temi e skinning* dato che implementa un meccanismo tematizzazione e di skinning che consente di cambiare in modo rapido e veloce il layout;
- *Web services* considerato che supporta la generazione automatica di complesse specifiche di WSDL service e la gestione delle richieste a Web service;
- *Internationalization (I18N) and Localization (L10N)* rilevato che supporta la traduzione di messaggi, formattazione di data e ora, la formattazione dei numeri, e la localizzazione dell’interfaccia;
- *Layered caching scheme* poiché supporta il caching dei dati, caching delle pagine, il caching di porzioni di contenuto e contenuti dinamici. Il supporto di memorizzazione nella cache può essere modificato facilmente senza toccare il codice dell’applicazione, ma agendo sul file di configurazione;

² <http://www.yiiframework.com/doc/guide/1.1/it/>

- *Error handling and logging* ovvero gli errori vengono gestiti e presentati in maniera gradevole. I messaggi di log possono essere categorizzati, filtrati e instradati verso diverse destinazioni;
- *Sicurezza* dato che Yii è dotato di molte misure di sicurezza per aiutare a prevenire attacchi come SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), e cookie tampering;
- *Generazione automatica del codice* fornita da una serie di strumenti intuitivi che aiutano a generare rapidamente il codice necessario per funzionalità come la creazione di form, CRUD;
- *Conformità a XHTML* infatti il codice HTML prodotto dai componenti Yii è conforme allo standard XHTML;
- *Object-oriented* considerato che Yii framework utilizza rigorosamente il paradigma OOP. Non definisce alcuna funzione o una variabile globale. La gerarchia di classi che lo definisce permette la massima riutilizzabilità e la personalizzazione;
- *Aperto a codice di terze parti* perché progettato per funzionare al meglio anche con codice di terze parti. Ad esempio, è possibile utilizzare codice da PEAR o Zend Framework;
- *Documentazione dettagliata* per ogni singolo metodo o proprietà. Sono disponibili un tour e screencast per iniziare, un libro e vari tutorials completi, oltre ad una class reference;
- *Extension library* dato che fornisce una libreria di estensione realizzata grazie al contributo degli sviluppatori.

2.2 Il Framework

In informatica, e specificatamente nello sviluppo software, un framework è una struttura logica di supporto su cui un software può essere progettato e realizzato, spesso facilitandone lo sviluppo. 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, come ad esempio un IDE, un debugger, o altri strumenti ideati per aumentare la velocità di sviluppo del prodotto finito. L'utilizzo di un framework impone dunque al programmatore una precisa metodologia di sviluppo del software.

Lo scopo di un framework è di risparmiare allo sviluppatore la riscrittura di codice già scritto in precedenza per compiti simili. Questa circostanza si è presentata sempre più spesso man mano che le interfacce utente sono diventate sempre più complesse, o, più in generale, man mano che è aumentata la quantità di software con funzionalità secondarie simili.

Ad esempio, il tipo di interazione con l'utente offerto da un menu a tendina sarà sempre la stessa indipendentemente dall'applicazione cui il menu appartiene (o almeno questo è ciò che l'utente si aspetta); in casi come questo un framework, che permette di aggiungere la funzionalità di una finestra con un menu a tendina con poche righe di codice sorgente a carico del programmatore, o magari permettendogli di disegnare comodamente il tutto in un ambiente di sviluppo, permetterà al programmatore di concentrarsi sulle vere funzionalità dell'applicazione, senza doversi far carico di scrivere codice "di contorno". Il termine inglese framework quindi può essere tradotto come intelaiatura o

struttura, che è appunto la sua funzione, a sottolineare che al programmatore rimane solo da creare il contenuto vero e proprio dell' applicazione.

Un framework è definito da un insieme di classi astratte e dalle relazioni tra esse. Istanziare un framework significa fornire un'implementazione delle classi astratte. L'insieme delle classi concrete, definite ereditando il framework, eredita le relazioni tra le classi; si ottiene in questo modo un insieme di classi concrete con un insieme di relazioni tra classi.

2.2.1 PRADO Framework

Il Progetto PRADO ³, acronimo di “PHP Rapid Application Development Object-oriented” ed avviato anch'esso da Qiang Xue, è un framework open source per applicazioni basate su componenti PHP web, orientato agli oggetti. Ispirato da Apache Tapestry con spunti di Borland Delphi e del framework ASP.NET di Microsoft esce come prima release nel giugno 2004, ma poiché stato scritto utilizzando il modello ad oggetti di PHP 4, molto limitato e ormai superato, ha riscontrato molti problemi portando Qiang a riscrivere il Framework per il nuovo modello ad oggetti di PHP 5.

Sulla base dell'esperienza ottenuta con il Progetto PRADO Framework, Qiang ha creato il Framework Yii, con l'obbiettivo di avviare una riprogettazione teorica di PRADO, al fine di superare le problematiche riscontrate nelle sue prime versioni nel maneggiare le pagine complesse con elevate prestazioni o scenari ad alto traffico.

2.2.2 jQuery Framework

Il Framework Yii, basato sull'architettura Model View Controller, è integrato con un framework javascript, noto anche come jQuery, con il quale è possibile creare potenti applicazioni AJAX (Asynchronous JavaScript and XML), ovvero una tecnica di sviluppo software per la realizzazione di applicazioni web interattive (Rich Internet Application).

jQuery ⁴ è una libreria di funzioni (framework) Javascript, cross-browser per le applicazioni web, che si propone come obiettivo quello di semplificare la programmazione lato client delle pagine HTML. Pubblicato, per la prima volta il 22 agosto 2005 da John Resig, ha raggiunto la versione 1 (stabile) il 26 agosto dell'anno successivo.

Tramite l'uso della libreria jQuery è possibile, con poche righe di codice, effettuare svariate operazioni, come ad esempio ottenere l'altezza di un elemento, o farlo scomparire con effetto dissolvenza. Anche la gestione degli eventi è completamente standardizzata, automatica; stessa cosa per quanto riguarda l'utilizzo di AJAX, in quanto sono presenti alcune funzioni molto utili e veloci che si occupano di istanziare i giusti oggetti ed effettuare la connessione e l'invio dei dati.

Lo sviluppo di applicazioni HTML con AJAX si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente. AJAX è asincrono nel senso che i dati extra sono richiesti al server e caricati in background senza interferire con il comportamento della pagina esistente.

³ <http://www.pradosoft.com/>

⁴ <http://jquery.com/>

L'integrazione di Yii con AJAX permette inoltre di generare automaticamente il codice JavaScript necessario per lavorare.

2.3 PHP - Hypertext Preprocessor

Il PHP ⁵, acronimo ricorsivo di "PHP: Hypertext Preprocessor" (preprocessore di ipertesti) originariamente acronimo di "Personal Home Page", è un linguaggio di programmazione interpretato, originariamente concepito per la programmazione Web ovvero per la realizzazione di pagine web dinamiche. L'interprete ha una licenza open source e libera (ma incompatibile con la GPL). Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server, ma può essere usato anche per scrivere script a riga di comando o applicazioni stand-alone con interfaccia grafica.

PHP riprende per molti versi la sintassi del C, come peraltro fanno molti linguaggi moderni, e del Perl. È un linguaggio a tipizzazione debole e dalla versione 5 migliora il supporto al paradigma di programmazione ad oggetti. Certi costrutti derivati dal C, come gli operatori fra bit e la gestione di stringhe come array, permettono in alcuni casi di agire a basso livello; tuttavia è fondamentalmente un linguaggio di alto livello, caratteristica questa rafforzata dall'esistenza delle sue moltissime API, oltre 3.000 funzioni del nucleo base.

PHP è in grado di interfacciarsi a innumerevoli database tra cui MySQL, PostgreSQL, MariaDB, Oracle, Firebird, IBM DB2, Microsoft SQL Server, solo per citarne alcuni, e supporta numerose tecnologie, come XML, SOAP, IMAP, FTP, CORBA. Si integra anche con altri linguaggi/piattaforme quali Java e .NET e si può dire che esista un wrapper per ogni libreria esistente, come CURL, GD, Gettext, GMP, Ming, OpenSSL ed altro.

Fornisce un'API specifica per interagire con Apache, nonostante funzioni naturalmente con numerosi altri server web. È anche ottimamente integrato con il database MySQL, per il quale possiede più di una API. Per questo motivo esiste un'enorme quantità di script e librerie in PHP, disponibili liberamente su Internet. La versione 5, comunque, integra al suo interno un piccolo database embedded, SQLite.

Dispone di un archivio chiamato PEAR che mette a disposizione un framework di librerie riusabili per lo sviluppo di applicazioni PHP e di PECL che raccoglie tutte le estensioni conosciute scritte in C.

PHP non ha ancora un supporto nativo per le stringhe Unicode o multibyte; il supporto Unicode è in fase di sviluppo per una futura versione di PHP, e consentirà di usare caratteri non ASCII in stringhe e nomi di funzioni, classi e metodi.

Il PHP permette il passaggio di parametri da una pagina all'altra attraverso tre array di variabili: \$_GET, \$_POST e \$_SESSION. Il primo tipo di parametro viene passato tramite la stringa che compare nella barra dell'indirizzo del browser; il secondo viene passato in background, mentre il terzo rimane persistente durante la sessione.

⁵ <http://www.php.net/>

2.3.1 Cenni Storici

Nato nel 1994 ad opera del danese Rasmus Lerdorf, PHP era in origine una raccolta di script CGI che permettevano una facile gestione delle pagine personali. Il significato originario dell'acronimo era Personal Home Page (secondo l'annuncio originale di PHP 1.0 da parte dell'autore sul newsgroup comp.infosystems.www.authoring.cgi).

Il pacchetto originario venne in seguito esteso e riscritto dallo stesso Lerdorf in C, aggiungendo funzionalità quali il supporto al database mSQL, o miniSQL, e prese a chiamarsi PHP/FI, dove FI sta per Form Interpreter (interprete di form), prevedendo la possibilità di integrare il codice PHP nel codice HTML in modo da semplificare la realizzazione di pagine dinamiche. In quel periodo, 50.000 domini Internet annunciavano di aver installato PHP.

A questo punto il linguaggio cominciò a godere di una certa popolarità tra i progetti open source del web, e venne così notato da due giovani programmatori: Zeev Suraski e Andi Gutmans. I due collaborarono nel 1998 con Lerdorf allo sviluppo della terza versione di PHP (il cui acronimo assunse il significato attuale) riscrivendone il motore che fu battezzato Zend da una contrazione dei loro nomi. Le caratteristiche chiave della versione PHP 3.0, frutto del loro lavoro, erano la straordinaria estensibilità, la connettività ai database e il supporto iniziale per il paradigma a oggetti. Verso la fine del 1998 PHP 3.0 era installato su circa il 10% dei server web presenti su Internet.

PHP diventò a questo punto talmente maturo da competere con ASP, linguaggio lato server analogo a PHP sviluppato da Microsoft, e cominciò ad essere usato su larga scala. La versione 4 di PHP venne rilasciata nel 2000 e prevedeva notevoli migliorie. Attualmente siamo alla quinta versione, sviluppata da un team di programmatori, che comprende ancora Lerdorf, oltre a Suraski e Gutmans.

La popolarità del linguaggio PHP è in costante crescita grazie alla sua flessibilità: nel giugno 2001, ha superato il milione di siti che lo utilizzano. Nell'ottobre 2002, più del 45% dei server Apache usavano PHP.

Nel gennaio 2005 è stato insignito del titolo di "Programming Language of 2004" dal TIOBE Programming Community Index, classifica che valuta la popolarità dei linguaggi di programmazione sulla base di informazioni raccolte dai motori di ricerca.

Nel 2005 la configurazione LAMP (Linux, Apache, MySQL, PHP) supera il 50% del totale dei server sulla rete mondiale.

Nel 2008 PHP 5 è diventata l'unica versione stabile in fase di sviluppo. A partire da PHP 5.3.0, PHP implementa una funzione chiamata "late static binding" che può essere utilizzata per fare riferimento alla classe chiamata in un contesto di eredità statica.

A partire dal 5 febbraio 2008, a causa dell'iniziativa GoPHP5, sostenuta da una serie di sviluppatori PHP, molti dei progetti open-source di alto profilo cessano di supportare PHP 4 nel nuovo codice e promuovono il passaggio da PHP 4 a PHP 5.

2.4 Architettura MVC – Model View Controller

Il Model View Controller (MVC) è un pattern architetturale per la programmazione di sistemi software orientata agli oggetti in grado di separare la logica di presentazione dei dati dalla logica applicativa. Anche se originariamente sviluppato per i personal computer, Model View Controller è stato ampiamente adottato come architettura per le applicazioni Web in tutti i principali linguaggi di programmazione. I diversi Framework variano nelle loro interpretazioni, soprattutto nel modo in cui sono divise le componenti MVC tra il client e il server.

I primi Framework MVC adottavano un approccio *thin client* che gestivano il modello interamente da server. In questo approccio, il client inviando le richieste al server riceve direttamente una pagina web completa e aggiornata.

MVC è stato di fatto una delle prime intuizioni nel campo delle interfacce grafiche, nonché una delle prime opere utili per descrivere e applicare costrutti software in termini di responsabilità.

Trygve Reenskaug introdusse per la prima volta MVC in Smalltalk-76 durante una visita a Xerox Parc, nel 1970, successivamente, nel 1980, Jim Althoff insieme ad altri programmatori attuò una versione di MVC per la libreria di classi Smalltalk-80. Solamente in seguito, in un articolo del 1988, MVC venne espresso come concetto generale.

Il 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 all'applicazione;
- *View* visualizza i dati contenuti nel *Model* e si occupa dell'interazione con utenti e agenti;
- *Controller* riceve i comandi dell'utente (in genere attraverso il *View*) e li attua modificando lo stato degli altri due componenti.

Il *Controller* ovvero "il modulo che si occupa dell'input" (in modo simile a come *View* si occupa dell'output), in applicazioni moderne degli anni 2000, è un modulo, o una sezione intermedia di codice, che media la comunicazione tra *Model* e *View*, ed unifica la convalida utilizzando anche le chiamate dirette per separare il *Model* dal *View* nel modello attivo .

Il Framework Yii implementa lo schema di progettazione MVC separando la logica applicativa (spesso chiamata "logica di business") a carico del *Controller* e del *Model*, e l'interfaccia utente a carico del *View*. Nel pattern MVC di Yii il *Model* rappresenta le informazioni (i dati) e la logica di business; la *View* contiene elementi dell'interfaccia utente come testi, form di inserimento dati; e il *Controller* gestisce le comunicazioni tra *Model* e *View*.

Il progetto Yii, oltre ad implementare il Model View Controller, introduce un front-controller, definito *Application* che incapsula il contesto di esecuzione per il processo di una richiesta, raccogliendo alcune informazioni sulla richiesta dell'utente e per poi smistarle al controller appropriato per ulteriori manipolazioni.

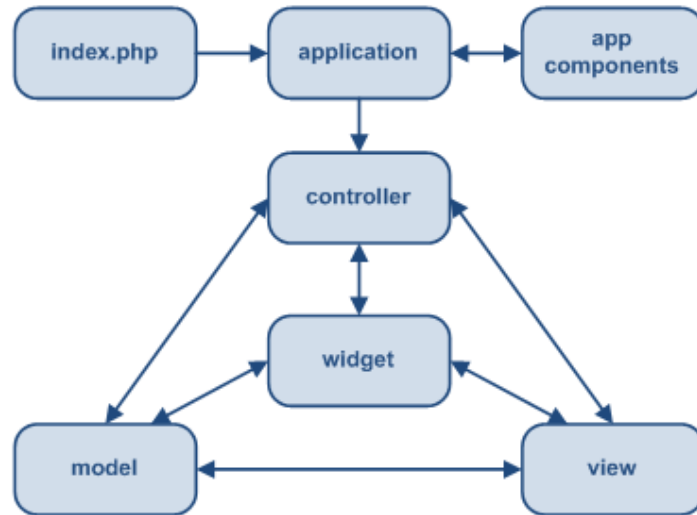


Figura 2.1 : Struttura statica di un'applicazione Yii

Tipicamente, quando deve esser gestita una richiesta utente, un'applicazione Yii segue un diagramma di flusso di lavoro come a seguire:

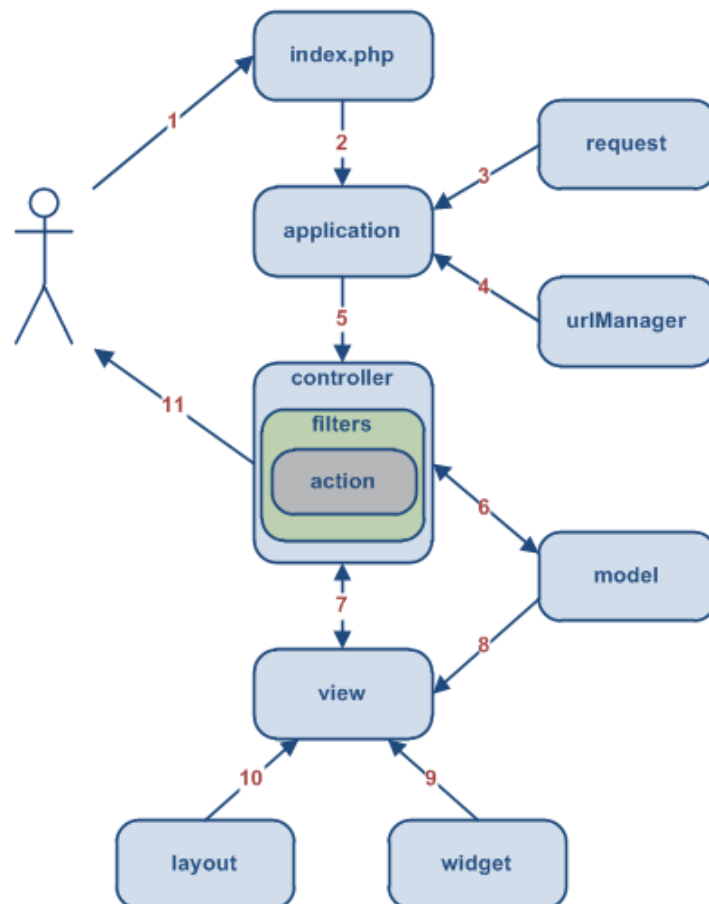


Figura 2.2 : Workflow tipico di un'applicazione Yii

- L'utente avanza una richiesta all'URL (Uniform Resource Locator) `http://www.example.com/index.php?r=post/show&id=1`, quindi il web server gestisce la richiesta eseguendo lo script di avvio `index.php`;
- Lo script di avvio crea un'istanza dell'Applicazione e la esegue;
- L'*Application* riceve le informazioni dettagliate della richiesta dell'utente da un component dell'applicazione che si chiama `request`;
- L'*Application* determina sia il controller che l'action richiesti grazie all'aiuto dell'*application component* che si chiama `urlManager`. In questo esempio, il controller è `post`, il quale si riferisce alla classe `PostController`; e l'action è `show`, il cui significato attuale è determinato dal controller;
- L'applicazione crea una istanza del controller richiesto per gestire ulteriormente la richiesta utente. Il controller determina che l'action `show` si riferisce al metodo, all'interno della classe del controller, che si chiama `actionShow`. Il metodo crea ed esegue i filtri (es. controllo accessi, benchmark) associati con questa action. L'action viene eseguita se ciò è permesso dai filtri;
- L'action, tramite il model, legge dal database il Post il cui ID è 1;
- L'action produce una view chiamata `show` con i prodotti dal model `Post`;
- La view legge e visualizza gli attributi del model `Post`;
- La view esegue alcuni widget;
- La produzione della view viene incorporata in un layout;
- L'action completa la produzione della view e ne visualizza il risultato all'utente;

2.5 Supporto al Database

Poiché la maggior parte delle web application si appoggiano ad un database, il Framework Yii opera fornendo un supporto potente per la programmazione con gli stessi. Costruito sull'estensione del PHP Data Objects (PDO), il Data Access Object (DAO) di Yii consente l'accesso a diversi sistemi di gestione database (DBMS) con un'unica interfaccia uniforme. Le applicazioni sviluppate utilizzando Yii DAO possono essere facilmente modificate per utilizzare un diverso DBMS senza dover modificare la parte di codice inerente l'accesso al database.

Il Yii Query Builder (costruttore di query di Yii) offre un metodo object-oriented per la costruzione di query SQL, che aiuta a ridurre i rischi di attacchi per SQL injection.

L'Active Record di Yii (AR), implementato con l'approccio Object-Relational Mapping (ORM) ormai ampiamente adottato, semplifica ulteriormente la programmazione con i database. Con la rappresentazione della tabella come se fosse una classe ed un record come se fosse una sua istanza, l'AR di Yii elimina il compito ripetitivo di scrivere quelle istruzioni SQL riguardanti prevalentemente le operazioni CRUD (create, read, update and delete - creare, leggere, aggiornare e cancellare).

Anche se le funzionalità incluse in Yii sono in grado di soddisfare quasi tutti i compiti inerenti il database, è comunque possibile continuare ad utilizzare le proprie librerie per database nella propria applicazione Yii. Dato di fatto, il framework Yii è stato progettato accuratamente per essere utilizzato insieme ad altre librerie esterne.

2.5.1 SQL - Structured Query Language

SQL ⁶ (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per:

- creare e modificare schemi di database (DDL - Data Definition Language);
- inserire, modificare e gestire dati memorizzati (DML - Data Manipulation Language);
- interrogare i dati memorizzati (DQL - Data Query Language);
- creare e gestire strumenti di controllo ed accesso ai dati (DCL - Data Control Language).

Nonostante il nome, non si tratta dunque solo di un semplice linguaggio di interrogazione, ma alcuni suoi sottoinsiemi si occupano della creazione, della gestione e dell'amministrazione del database.

SQL è un linguaggio per interrogare e gestire basi di dati mediante l'utilizzo di costrutti di programmazione denominati query. Con SQL si leggono, modificano, cancellano dati e si esercitano funzioni gestionali ed amministrative sul sistema dei database. La maggior parte delle implementazioni dispongono di interfaccia alla riga di comando per l'esecuzione diretta di comandi, in alternativa alla sola interfaccia grafica GUI.

Originariamente progettato come linguaggio di tipo dichiarativo, si è successivamente evoluto con l'introduzione di costrutti procedurali, istruzioni per il controllo di flusso, tipi di dati definiti dall'utente e varie altre estensioni del linguaggio. A partire dalla definizione dello standard SQL:1999 molte di queste estensioni sono state formalmente adottate come parte integrante di SQL nella sezione SQL/PSM dello standard.

Alcune delle critiche più frequenti rivolte ad SQL riguardano la mancanza di portabilità del codice fra *vendor* diversi, il modo inappropriato con cui vengono trattati i dati mancanti (Null), e la semantica a volte inutilmente complicata.

⁶ <http://www.postgresql.org/>

3. Sviluppo di un'applicazione con Yii

Preso atto delle potenzialità del Framework Yii PHP si vuole testare lo stesso cercando di connetterlo ad un database esistente. La base di dati scelta riguarda un progetto di una Associazione Onlus destinato al recupero e alla redistribuzione gratuita di prodotti alimentari ai fini di solidarietà sociale utilizzato per gestire i prodotti in modo tale da poterli rendere rintracciabili in ogni movimento partendo dalla provenienza, e quindi i fornitori, fino alla destinazione, ovvero le famiglie inserite nel progetto, passando ovviamente per la catena di redistribuzione gestita dai volontari.

3.1 Progettazione del Database

Al fine di apprendere al meglio le funzionalità del Framework viene riportato brevemente a seguire la teoria di progettazione del database utilizzato per lo sviluppo dell'applicazione.

3.1.1 Requisiti Strutturati

Fraasi per Comune: per ogni comune, identificato univocamente dal Codice Catastale, sede dei fornitori o luogo di residenza delle persone intese come operatori volontari o famiglie inserite nel progetto, si rappresentano i dati CAP, nome del Comune e Provincia.

Fraasi per Fornitore: per ogni fornitore, identificato univocamente dall'Id_Fornitore, ovvero un codice numerico ad uso interno possibilmente progressivo, rappresentiamo la relativa partita IVA, il nome della società, l'indirizzo, il relativo codice catastale comunale, un numero di telefono e delle note aggiuntive utili per il nome e cognome dell'attuale referente. Un fornitore può avere più sedi territoriali e pertanto, utilizzando sempre la stessa partita IVA, non è possibile utilizzare tale dato come chiave primaria.

Fraasi per Fornitura: per ogni fornitura, identificata univocamente dall'Id_Fornitura, rappresentiamo la data di consegna, il fornitore ed il prodotto consegnato nonché la quantità consegnata, il numero della bolla di accompagnamento, se prevista, ed infine l'operatore incaricato al ritiro della stessa. La bolla di accompagnamento non viene prevista come chiave primaria poiché non è obbligatoriamente prevista.

Fraasi per Prodotto: per ogni prodotto, identificato univocamente dal codice a barre, rappresentiamo un nome (o etichetta/descrizione) dello stesso, è inoltre richiesto che si possa sempre conoscere la quantità attualmente disponibile in magazzino per l'associazione.

Frazi per Ordine: per ogni ordine, identificato univocamente dall'Id_Ordine, rappresentiamo la data di composizione/distribuzione, il destinatario ed il prodotto, nonché la quantità prevista, una variabile di controllo della distribuzione (Evaso), ed infine l'operatore incaricato al trasporto dell'ordine.

Frazi per Persona Fisica: per ogni Persona, intesa come operatore volontario o famiglia inserita nel progetto, identificata univocamente tramite il proprio codice fiscale (da intendersi univoco), rappresentiamo i dati anagrafici quali cognome, nome, indirizzo, il relativo codice catastale comunale, un numero di telefono, un numero di cellulare, una eMail e quindi una variabile di distinzione per gli operatori rispetto ai destinatari.

3.1.2 Modello Concettuale - Schema E-R

A seguire viene riportato lo schema concettuale prodotto per la rappresentanza della realtà di interesse.

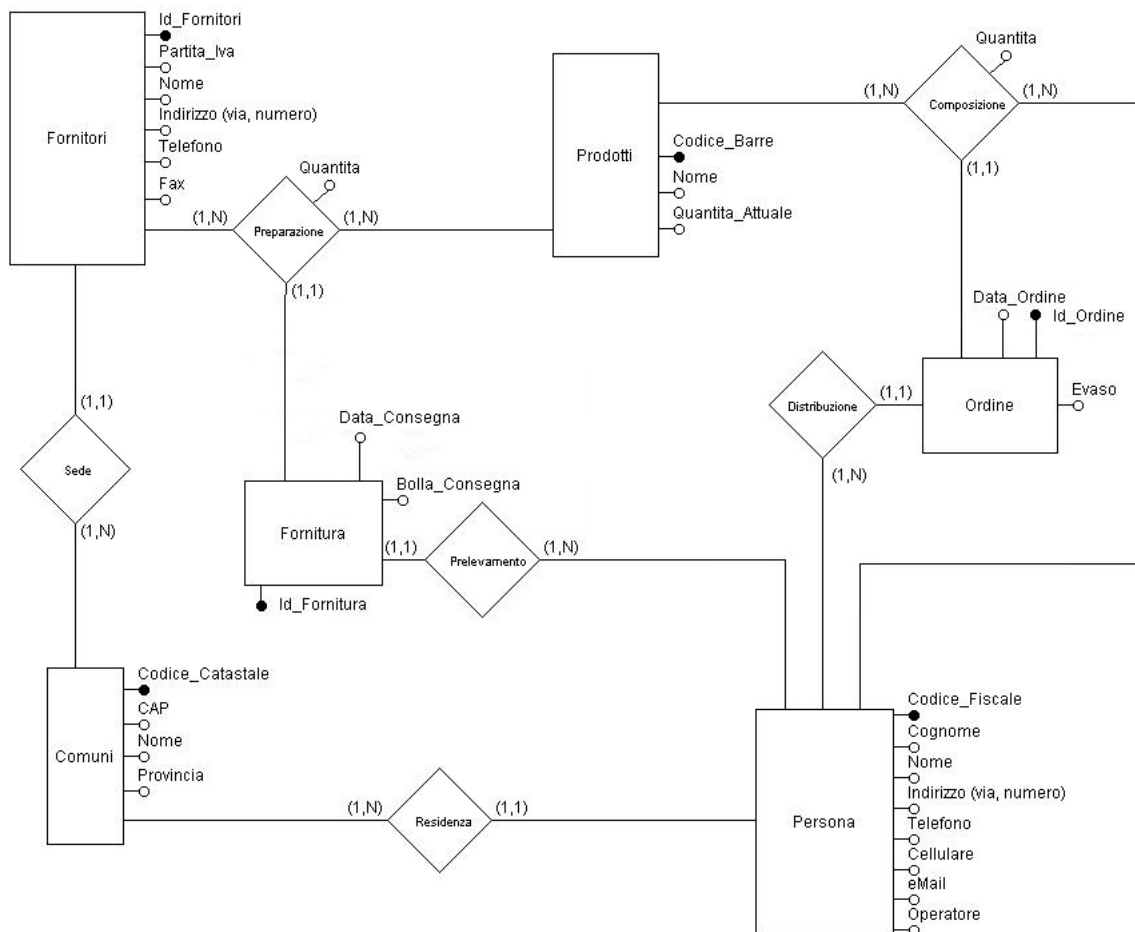


Figura 3.1 : Schema E-R

3.1.3 Modello Logico: Relazionale

A seguire viene riportato lo schema logico relazionale prodotto per la rappresentanza della realtà di interesse.

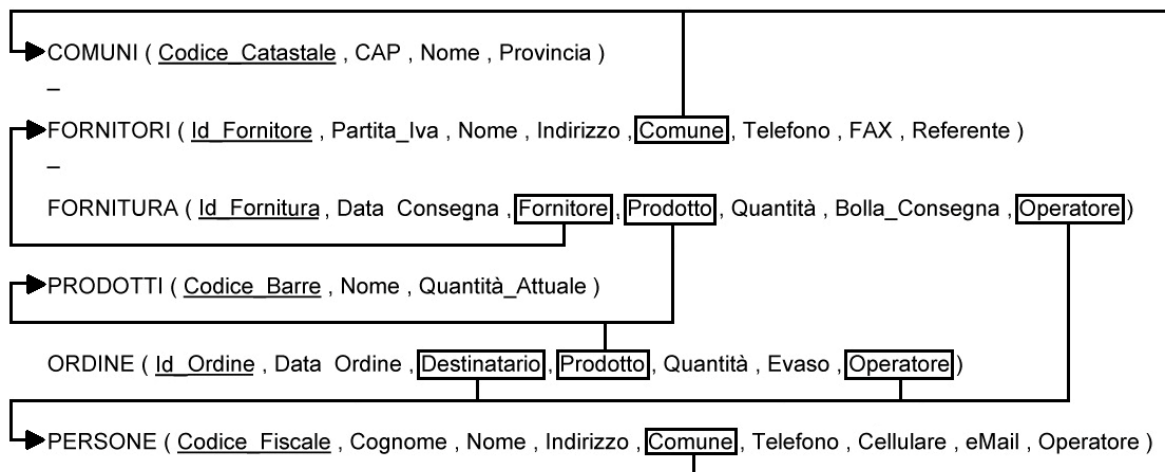


Figura 3.2 : Schema Logico Relazionale

3.2 Generazione dell'applicazione

A seguito di una precedente installazione del software necessario è possibile generare la prima web application sviluppata con Yii. Si procede aprendo una shell, posizionandoci all'interno della cartella nella quale desideriamo generare la nostra web application e lanciando il seguente comando *yii webapp xlavita*, dove "xlavita" è il nome che viene dato all'applicazione, ed infine confermando l'intenzione di voler creare una nuova web application all'interno della cartella */var/www/xlavita* per sistemi Linux, oppure in alternativa *C:\xampp\htdocs\xlavita* per sistemi Windows. Automaticamente Yii creerà una nuova web application base (Home, About, Contact e Login) funzionante all'indirizzo *http://hostname/xlavita/index.php*, completa di uno skeleton di base con i relativi file minimi necessari alla gestione della stessa.

Struttura 3.1 : Lista ad Albero dei File di una Yii Web Application Base

xlavita/	
index.php	script di entrata della web application
index-test.php	script di entrata per i test funzionali
assets/	contenitore dei file di risorse pubblicate
css/	contenitore dei file CSS
images/	contenitore dei file immagine
themes/	contenitore dei temi dell'applicazione
protected/	contenitore dei file protetti dell'applicazione
yii	script di riga di comando per Unix/Linux
yii.bat	script di riga di comando per Windows
yii.php	script di riga di comando in PHP
commands/	contenitore di comandi 'yii' personalizzati
shell/	contenitore di comandi 'yii shell' personalizzati
components/	contenitore di componenti utente riutilizzabili
Controller.php	classe di base per tutte le classi Controller
UserIdentity.php	la classe 'UserIdentity' usata per l'autenticazione
config/	contenitore dei file di configurazione

console.php	configurazione della console dell'applicazione
main.php	configurazione della web application
test.php	configurazione dei test funzionali
controllers/	contenitore dei file delle classi Controller
SiteController.php	classe Controller di default
data/	contenitore di database di esempio
schema.mysql.sql	schema di esempio database MySQL
schema.sqlite.sql	schema di esempio database SQLite
testdrive.db	file di esempio database SQLite
extensions/	contenitore estensioni di terze parti
messages/	contenitore delle traduzioni dei messaggi
models/	contenitore dei file delle classi Model
LoginForm.php	il Model del form di login per l'Action di 'login'
ContactForm.php	il Model del form di contatto per l'Action 'contact'
runtime/	contenitore dei file generati temporaneamente
tests/	contenitore degli script di test
views/	contenitore dei file di View
layouts/	contenitore dei file di View dei layout
main.php	layout di base condiviso da tutte le pagine
column1.php	layout per le pagine con singola colonna
column2.php	layout per le pagine che utilizzano due colonne
site/	contenitore dei file di View del Controller 'site'
pages/	contenitore delle pagine "static" (statiche)
about.php	la View della pagina "about"
contact.php	la View della action 'contact'
error.php	la View della action 'error'
index.php	la View della action 'index'
login.php	la View della action 'login'

3.3 Connessione al Database

Lo step successivo riguarda per l'appunto la configurazione della web application appena creata con la base di dati. Si deve procedere accedendo al file di configurazione dell'applicazione *xlavita/protected/config/main.php*.

Codice 3.1 : dbConnection main.php

```
return array(
    .....
    'components'=>array(
        .....
        'db'=>array(
            'connectionString'=>'sqlite:protected/data/testdrive.db',
        ),
    ),
    .....
);
```

Poiché il database SQLite risulta incluso nella struttura dell'applicazione appena generata, come descritto dal codice sopra riportato, si deve procedere all'implementazione dello stesso modificandolo a seconda della base di dati che si sta utilizzando. Pertanto andremo a modificare il contenuto della stringa *connectionString*, per esempio, con una delle seguenti righe:

- MySQL: *mysql:host= hostname;dbname=xlavita*
- PostgreSQL: *pgsql:host= hostname;port=5432;dbname=xlavita*
- SQL Server: *mssql:host= hostname;dbname=xlavita*
- Oracle: *oci:dbname=// hostname:1521/xlavita*

Seguita dal codice riguardante le credenziali di accesso allo stesso (*'username'=>'username', 'password'=>'password',*)

3.4 Implementazione delle operazioni CRUD

Una volta collegati al database non rimane altro che implementare il codice di gestione dello stesso, ovvero implementare delle operazioni CRUD (create, read, update and delete – crea, leggi, aggiorna e cancella) per le tabelle del database. Con l'impiego del Framework Yii, anziché scrivere il codice necessario, è possibile utilizzare un generatore di codice ad interfaccia web denominato Gii.

3.4.1 Abilitazione Gii - Generatore di Codice ad Interfaccia Web

Gii è stato reso disponibile a partire dalla versione 1.1.2 di Yii PHP Framework e per poterlo utilizzare basta aggiornare nuovamente il file di configurazione dell'applicazione *xlavita/protected/config/main.php* verificando dapprima che il codice riportato sia abilitato, modificando poi la password di accesso al generatore stesso aggiornandola, ed infine accedendo all'URL *http://hostname/xlavita/index.php?r=gii*.

Codice 3.2 : GiiModule main.php

```
return array(
    .....
    'import'=>array(
        'application.models.*',
        'application.components.*',
    ),

    'modules'=>array(
        'gii'=>array(
            'class'=>'system.gii.GiiModule',
            'password'=>'inserire qui una password',
        ),
    ),
);
```

3.4.2 Generazione del Model User

Il passo successivo all'accesso a Gii è quello di generare i Model cliccando sul link *Model Generator*.

The screenshot shows the 'Model Generator' interface in the Yii Code Generator. The sidebar on the left has a 'Generators' menu with 'Model Generator' selected. The main area contains the following configuration options:

- Database Connection ***: db
- Table Prefix**: [empty]
- Table Name ***: persone
- Model Class ***: Persone
- Base Class ***: CActiveRecord
- Model Path ***: application.models
- Build Relations**:
- Use Column Comments as Attribute Labels**:
- Code Template ***: default (C:\xampp\htdocs\YiiPath\framework\yii\generators\model\templates\default)

A 'Preview' button is located at the bottom of the configuration area.

Figura 3.3 : Model Generator

Nel campo *Table Name* si potrà inserire il nome della tabella del database alla quale si vuol accedere, automaticamente il campo *Model Class* si chiamerà in ugual modo e proseguendo verrà generato un nuovo file di codice all'interno della cartella *protected/models* completa di tutti i campi della nostra tabella.

Questo passaggio può esser ripetuto per tutte le tabelle del nostro database per le quali abbiamo la necessità di creare una classe del Model.

Codice 3.3 : Model persone.php

```
<?php

/**
 * This is the model class for table "persone".
 *
 * The followings are the available columns in table 'persone':
 * @property string $Codice_Fiscale
 * @property string $Cognome_Persona
 * @property string $Nome_Persona
 * @property string $Indirizzo_Persona
 * @property string $Comuni_Codice_Catastale
 * @property string $Telefono_Persona
 * @property string $Cellulare
 * @property string $eMail
 * @property integer $Operatore
 *
 * The followings are the available model relations:
```

```

* @property Fornitura[] $fornituras
* @property Ordine[] $ordines
* @property Ordine[] $ordines1
* @property Comuni $comuniCodiceCatastale
*/
class Persone extends CActiveRecord
{
    /**
     * @return string the associated database table name
     */
    public function tableName()
    {
        return 'persone';
    }

    /**
     * @return array validation rules for model attributes.
     */
    public function rules()
    {
        // NOTE: you should only define rules for those attributes that
        // will receive user inputs.
        return array(
            array('Codice_Fiscale, Cognome_Persona, Nome_Persona, Indirizzo_Persona,
Comuni_Codice_Catastale', 'required'),
            array('Operatore', 'numerical', 'integerOnly'=>true),
            array('Codice_Fiscale', 'length', 'max'=>16),
            array('Cognome_Persona, Nome_Persona, Indirizzo_Persona, eMail', 'length', 'max'=>45),
            array('Comuni_Codice_Catastale', 'length', 'max'=>4),
            array('Telefono_Persona, Cellulare', 'length', 'max'=>15),
            // The following rule is used by search().
            // @todo Please remove those attributes that should not be searched.
            array('Codice_Fiscale, Cognome_Persona, Nome_Persona, Indirizzo_Persona,
Comuni_Codice_Catastale, Telefono_Persona, Cellulare, eMail, Operatore', 'safe', 'on'=>'search'),
        );
    }

    /**
     * @return array relational rules.
     */
    public function relations()
    {
        // NOTE: you may need to adjust the relation name and the related
        // class name for the relations automatically generated below.
        return array(
            'fornituras' => array(self::HAS_MANY, 'Fornitura', 'Operatore_Codice_Fiscale'),
            'ordines' => array(self::HAS_MANY, 'Ordine', 'Destinatario_Codice_Fiscale'),
            'ordines1' => array(self::HAS_MANY, 'Ordine', 'Operatore_Codice_Fiscale'),
            'comuniCodiceCatastale' => array(self::BELONGS_TO, 'Comuni',
'Comuni_Codice_Catastale'),
        );
    }

    /**
     * @return array customized attribute labels (name=>label)
     */
    public function attributeLabels()
    {
        return array(
            'Codice_Fiscale' => 'Codice Fiscale',
            'Cognome_Persona' => 'Cognome Persona',
            'Nome_Persona' => 'Nome Persona',
            'Indirizzo_Persona' => 'Indirizzo Persona',
            'Comuni_Codice_Catastale' => 'Comuni Codice Catastale',
            'Telefono_Persona' => 'Telefono Persona',
            'Cellulare' => 'Cellulare',
            'eMail' => 'E Mail',
            'Operatore' => 'Operatore',
        );
    }
}

```

```

        );
    }

/**
 * Retrieves a list of models based on the current search/filter conditions.
 *
 * Typical usecase:
 * - Initialize the model fields with values from filter form.
 * - Execute this method to get CActiveDataProvider instance which will filter
 * models according to data in model fields.
 * - Pass data provider to CGridView, CListView or any similar widget.
 *
 * @return CActiveDataProvider the data provider that can return the models
 * based on the search/filter conditions.
 */
public function search()
{
    // @todo Please modify the following code to remove attributes that should not be searched.

    $criteria=new CDbCriteria;

    $criteria->compare('Codice_Fiscale',$this->Codice_Fiscale,true);
    $criteria->compare('Cognome_Persona',$this->Cognome_Persona,true);
    $criteria->compare('Nome_Persona',$this->Nome_Persona,true);
    $criteria->compare('Indirizzo_Persona',$this->Indirizzo_Persona,true);
    $criteria->compare('Comuni_Codice_Catastale',$this->Comuni_Codice_Catastale,true);
    $criteria->compare('Telefono_Persona',$this->Telefono_Persona,true);
    $criteria->compare('Cellulare',$this->Cellulare,true);
    $criteria->compare('eMail',$this->eMail,true);
    $criteria->compare('Operatore',$this->Operatore);

    return new CActiveDataProvider($this, array(
        'criteria'=>$criteria,
    ));
}

/**
 * Returns the static model of the specified AR class.
 * Please note that you should have this exact method in all your CActiveRecord descendants!
 * @param string $className active record class name.
 * @return Persone the static model class
 */
public static function model($className=__CLASS__)
{
    return parent::model($className);
}
}

```

3.4.3 Generazione del codice CRUD

Dopo aver creato i file necessari con il *Model Generator*, verrà generato il codice dei View e dei Controller, implementando le operazioni CRUD sui dati utente, selezionando il link *Crud Generator* di Gii.

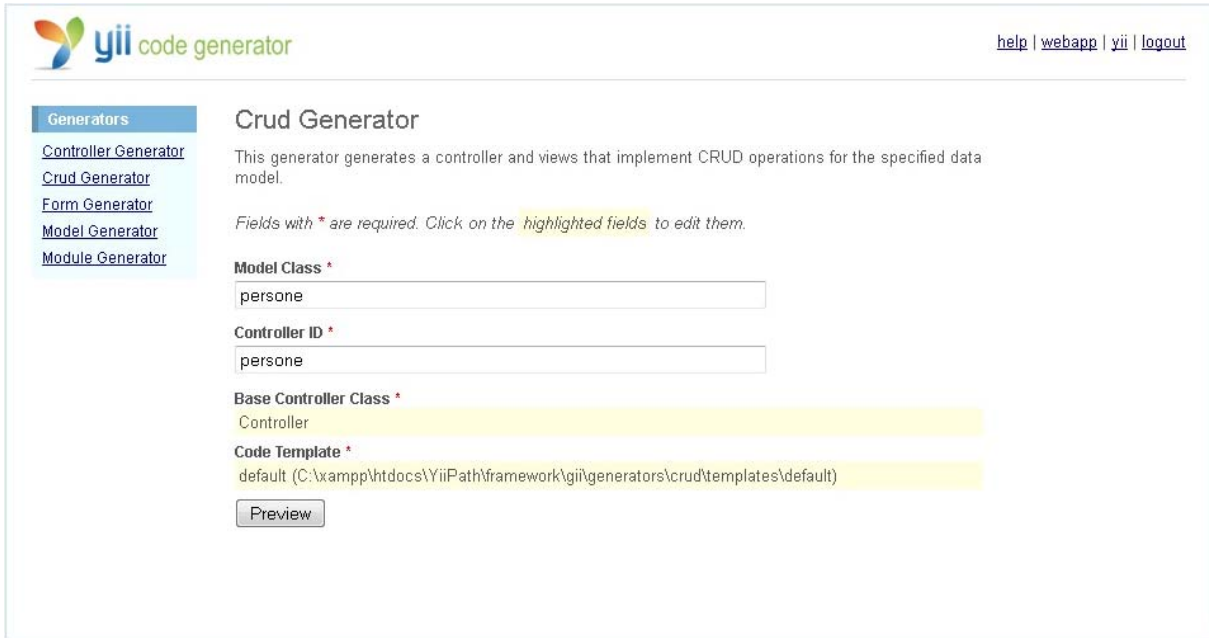


Figura 3.4 : CRUD Generator

Nel campo *Model Class* digitiamo il nome del Modello precedentemente creato che vogliamo utilizzare, automaticamente il campo *Controller ID* si chiamerà in ugual modo e proseguendo verrà generato un nuovo file di codice all'interno della cartella *protected/controllers* seguito da una serie di file all'interno della cartella *protected/views* utili per le funzionalità CRUD.

Codice 3.4 : Controller *PersoneController.php*

```
<?php
class PersoneController extends Controller
{
    /**
     * @var string the default layout for the views. Defaults to '//layouts/column2', meaning
     * using two-column layout. See 'protected/views/layouts/column2.php'.
     */
    public $layout='//layouts/column2';

    /**
     * @return array action filters
     */
    public function filters()
    {
        return array(
            'accessControl', // perform access control for CRUD operations
            'postOnly + delete', // we only allow deletion via POST request
        );
    }

    /**
     * Specifies the access control rules.

```

```

* This method is used by the 'accessControl' filter.
* @return array access control rules
*/
public function accessRules()
{
    return array(
        array('allow', // allow all users to perform 'index' and 'view' actions
            'actions'=>array('index','view'),
            'users'=>array('*'),
        ),
        array('allow', // allow authenticated user to perform 'create' and 'update' actions
            'actions'=>array('create','update'),
            'users'=>array('@'),
        ),
        array('allow', // allow admin user to perform 'admin' and 'delete' actions
            'actions'=>array('admin','delete'),
            'users'=>array('admin'),
        ),
        array('deny', // deny all users
            'users'=>array('*'),
        ),
    );
}

/**
 * Displays a particular model.
 * @param integer $id the ID of the model to be displayed
 */
public function actionView($id)
{
    $this->render('view',array(
        'model'=>$this->loadModel($id),
    ));
}

/**
 * Creates a new model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 */
public function actionCreate()
{
    $model=new Persone;

    // Uncomment the following line if AJAX validation is needed
    // $this->performAjaxValidation($model);

    if(isset($_POST['Persone']))
    {
        $model->attributes=$_POST['Persone'];
        if($model->save())
            $this->redirect(array('view','id'=>$model->Codice_Fiscale));
    }

    $this->render('create',array(
        'model'=>$model,
    ));
}

/**
 * Updates a particular model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $id the ID of the model to be updated
 */
public function actionUpdate($id)
{
    $model=$this->loadModel($id);

    // Uncomment the following line if AJAX validation is needed

```

```

// $this->performAjaxValidation($model);

if(isset($_POST['Persone']))
{
    $model->attributes=$_POST['Persone'];
    if($model->save())
        $this->redirect(array('view','id'=>$model->Codice_Fiscale));
}

$this->render('update',array(
    'model'=>$model,
));
}

/**
 * Deletes a particular model.
 * If deletion is successful, the browser will be redirected to the 'admin' page.
 * @param integer $id the ID of the model to be deleted
 */
public function actionDelete($id)
{
    $this->loadModel($id)->delete();

    // if AJAX request (triggered by deletion via admin grid view), we should not redirect the browser
    if(!isset($_GET['ajax']))
        $this->redirect(isset($_POST['returnUrl']) ? $_POST['returnUrl'] : array('admin'));
}

/**
 * Lists all models.
 */
public function actionIndex()
{
    $dataProvider=new CActiveDataProvider('Persone');
    $this->render('index',array(
        'dataProvider'=>$dataProvider,
    ));
}

/**
 * Manages all models.
 */
public function actionAdmin()
{
    $model=new Persone('search');
    $model->unsetAttributes(); // clear any default values
    if(isset($_GET['Persone']))
        $model->attributes=$_GET['Persone'];

    $this->render('admin',array(
        'model'=>$model,
    ));
}

/**
 * Returns the data model based on the primary key given in the GET variable.
 * If the data model is not found, an HTTP exception will be raised.
 * @param integer $id the ID of the model to be loaded
 * @return Persone the loaded model
 * @throws CHttpException
 */
public function loadModel($id)
{
    $model=Persone::model()->findByPk($id);
    if($model===null)
        throw new CHttpException(404,'The requested page does not exist.');
```

```

/**
 * Performs the AJAX validation.
 * @param Persone $model the model to be validated
 */
protected function performAjaxValidation($model)
{
    if(isset($_POST['ajax']) && $_POST['ajax']==='persone-form')
    {
        echo CActiveForm::validate($model);
        Yii::app()->end();
    }
}
}

```

Codice 3.5 : View Persone\View.php

```

<?php
/* @var $this PersoneController */
/* @var $model Persone */

$this->breadcrumbs=array(
    'Persone'=>array('index'),
    $model->Codice_Fiscale,
);

$this->menu=array(
    array('label'=>'List Persone', 'url'=>array('index')),
    array('label'=>'Create Persone', 'url'=>array('create')),
    array('label'=>'Update Persone', 'url'=>array('update', 'id'=>$model->Codice_Fiscale)),
    array('label'=>'Delete Persone', 'url'=> '#', 'linkOptions'=>array('submit'=>array('delete', 'id'=>$model->Codice_Fiscale), 'confirm'=>'Are you sure you want to delete this item?')),
    array('label'=>'Manage Persone', 'url'=>array('admin')),
);
?>

<h1>View Persone #<?php echo $model->Codice_Fiscale: ?></h1>

<?php $this->widget('zii.widgets.CDetailView', array(
    'data'=>$model,
    'attributes'=>array(
        'Codice_Fiscale',
        'Cognome_Persona',
        'Nome_Persona',
        'Indirizzo_Persona',
        'Comuni_Codice_Catastale',
        'Telefono_Persona',
        'Cellulare',
        'eMail',
        'Operatore',
    ),
)); ?>

```

3.4.4 Accesso alle pagine CRUD

Infine non rimane altro che accedere ad una delle pagine di visualizzazione (view) precedentemente generate, per esempio per caricare la pagina di comuni, visitiamo il seguente indirizzo `http://hostname/xlavita/index.php?r=comuni`, dove “comuni” identifica la tabella del database alla quale vogliamo accedere, per visualizzare l'elenco delle voci della tabella indicata.

The screenshot shows a web application interface for managing municipalities. At the top, there is a navigation bar with links for Home, Comuni, Fornitori, Prodotti, Persone, Fornitura, Ordine, About, Contact, and Login. Below the navigation bar, the breadcrumb trail shows 'Home » Comuni'. The main heading is 'Comuni'. To the right of the heading, it says 'Displaying 1-3 of 3 results.' On the right side, there is a sidebar with a blue header 'Operations' and two buttons: 'Create Comuni' and 'Manage Comuni'. The main content area contains three entries, each in a white box with a light blue border. Each entry displays the following information: 'Codice Catastale' (with a link), 'Cap', 'Nome Comune', and 'Provincia'. The entries are: 1. Codice Catastale: [G224](#), Cap: 35100, Nome Comune: Padova, Provincia: PD. 2. Codice Catastale: [I595](#), Cap: 35030, Nome Comune: Selvazzano Dentro, Provincia: PD. 3. Codice Catastale: [L710](#), Cap: 35030, Nome Comune: Veggiano, Provincia: PD. At the bottom of the page, there is a footer with the text: 'Copyright © 2013 by My Company. All Rights Reserved. Powered by [Yii Framework](#).'

Figura 3.5 : List Comuni

Si può notare fin da subito il menù caricato a destra della pagina, visibile a tutti, ma accessibile solamente attraverso autenticazione (*Login*). Cliccando il link *Create Comuni* verrà visualizzato un form che consente di aggiungere un nuovo comune, per esempio, compilando i campi del modulo e cliccando sul bottone *Create*. Qualora ci sia un errore di input, verrà visualizzato un messaggio di errore che ci impedirà di salvare i dati inseriti, altrimenti tornando alla pagina di elenco degli utenti dovremmo vedere apparire nella lista il nuovo inserimento.

My Web Application

Home Comuni Fornitori Prodotti Persone Fornitura Ordine About Contact Logout (admin)

Home » Comuni » Manage

Manage Comuni

You may optionally enter a comparison operator (<, <=, >, >=, <> or =) at the beginning of each of your search values to specify how the comparison should be done.

[Advanced Search](#)

Codice Catastale

Cap







Nome Comune

Provincia

Operations

- List Comuni
- Create Comuni

Displaying 1-3 of 3 results.

Codice Catastale	Cap	Nome Comune	Provincia	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
G224	35100	Padova	PD	 
I595	35030	Selvazzano Dentro	PD	 
L710	35030	Veggiano	PD	 

Copyright © 2013 by My Company.
All Rights Reserved.
Powered by [Yii Framework](#).

Figura 3.6 : Manage Comuni

Il secondo link *Manage Comuni* invece, che diversamente carica la seguente URL `http://hostname/xlavita/index.php?r=comuni/admin` permette di operare nel database avanzando query specifiche nella tabella che stiamo visualizzando, ma anche di aggiornare o addirittura cancellare le corrispondenti righe di dati dei singoli Comuni. Tutte queste caratteristiche interessanti sono disponibili senza che ci sia la necessità di scrivere una sola riga di codice PHP!

4. CONCLUSIONI

Da questo studio di Yii PHP Framework si osserva come lo stesso presenti la minima complessità, e quindi un'alta intuitività di impiego, nello sviluppo di applicazioni web finalizzate alla gestione di database. Yii, infatti, si presenta come uno strumento che in pochi step ci rende operativi, dimostrando come sia possibile creare, con pochi passaggi, le pagine necessarie alla gestione delle tabelle del nostro database. Il Progetto Yii risulta esser un framework cui prestare attenzione, perché ottimamente integrato con il pattern MVC. I model consentono di effettuare tutte le operazioni di base sulle tabelle senza nemmeno scrivere una riga di codice poiché l'aggiunta del modulo Gii permette l'autogenerazione di model e tutti i file necessari per le operazioni CRUD.

Il Framework Yii si dimostra capace di interagire con i più diffusi database fornendo all'utente sempre la stessa interfaccia per svolgere le operazioni di manipolazione dei dati.

Infine la scelta di questo strumento, rispetto ad altri Framework concorrenti, è dovuta non solo all'ottima documentazione offerta dal sito ufficiale, ma al fatto che Yii è un framework ad alte prestazioni ⁷, il grafico a seguire riporta l'efficienza di Yii rispetto ad altri framework PHP popolari. Nel grafico, RPS sta per "richiesta per secondo", che descrive la quantità di richieste di un'applicazione scritta in un contesto in grado di elaborare al secondo. Più alto è il numero, più efficiente è un Framework. Si nota che Yii supera tutti gli altri Framework in questo confronto. Il vantaggio delle prestazioni Yii è particolarmente significativo quando l'estensione APC (Alternative PHP Cache) è abilitata.

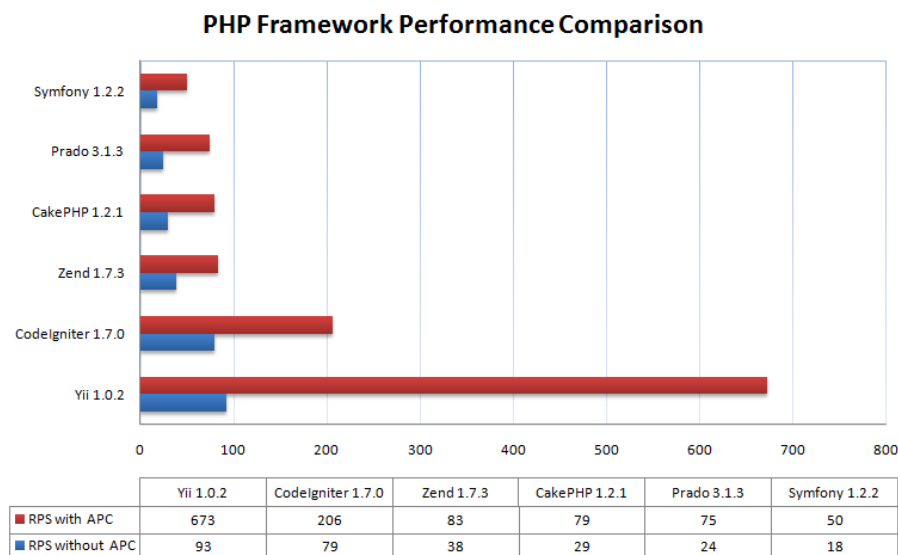


Figura 4.1 : PHP Framework Performance Comparison

⁷ <http://www.yiiframework.com/performance/>

APPENDICE

Codice A.1 : Codice SQL struttura Database

```
CREATE TABLE Comuni (  
  Codice_Catastale VARCHAR(4) NOT NULL DEFAULT 'G224' ,  
  CAP VARCHAR(5) NULL DEFAULT '35100' ,  
  Nome_Comune VARCHAR(50) NULL DEFAULT 'Padova' ,  
  Provincia VARCHAR(2) NULL DEFAULT 'PD' ,  
  PRIMARY KEY (Codice_Catastale) );
```

```
CREATE TABLE Fornitori (  
  Id_Fornitori INT NOT NULL ,  
  Partita_Iva VARCHAR(11) NULL DEFAULT NULL ,  
  Nome_Fornitore VARCHAR(45) NOT NULL ,  
  Indirizzo_Fornitore VARCHAR(45) NULL DEFAULT NULL ,  
  Comuni_Codice_Catastale VARCHAR(4) NOT NULL ,  
  Telefono VARCHAR(15) NOT NULL DEFAULT '+ 39' ,  
  FAX VARCHAR(15) NULL DEFAULT '+ 39' ,  
  Referente VARCHAR(45) NULL DEFAULT NULL ,  
  PRIMARY KEY (Id_Fornitori) ,  
  FOREIGN KEY (Comuni_Codice_Catastale )  
    REFERENCES Comuni (Codice_Catastale )  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE);
```

```
CREATE TABLE Prodotti (  
  Codice_Barre VARCHAR(13) NOT NULL ,  
  Nome_Prodotto VARCHAR(45) NOT NULL ,  
  Quantita_Attuale INT NULL DEFAULT 0 ,  
  PRIMARY KEY (Codice_Barre) );
```

```
CREATE TABLE Persone (  
  Codice_Fiscale VARCHAR(16) NOT NULL ,  
  Cognome_Persona VARCHAR(45) NOT NULL ,  
  Nome_Persona VARCHAR(45) NOT NULL ,  
  Indirizzo_Persona VARCHAR(45) NOT NULL ,  
  Comuni_Codice_Catastale VARCHAR(4) NOT NULL ,  
  Telefono_Persona VARCHAR(15) NULL DEFAULT '+ 39' ,  
  Cellulare VARCHAR(15) NOT NULL DEFAULT '+ 39' ,  
  eMail VARCHAR(45) NULL DEFAULT 'a@b.cd' ,  
  Operatore BOOLEAN DEFAULT FALSE ,  
  PRIMARY KEY (Codice_Fiscale) ,  
  FOREIGN KEY (Comuni_Codice_Catastale )  
    REFERENCES Comuni (Codice_Catastale )  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE);
```

```

CREATE TABLE Fornitura (
  Data_Consegna DATE NOT NULL ,
  Fornitori_Id_Fornitori INT NOT NULL ,
  Prodotti_Codice_Barre VARCHAR(13) NOT NULL ,
  Quantita_Consegna INT NOT NULL DEFAULT 1 ,
  Bolla_Consegna VARCHAR(45) NULL DEFAULT NULL ,
  Operatore_Codice_Fiscale VARCHAR(16) NOT NULL ,
  PRIMARY KEY (Data_Consegna, Fornitori_Id_Fornitori, Prodotti_Codice_Barre) ,
  FOREIGN KEY (Fornitori_Id_Fornitori )
  REFERENCES Fornitori (Id_Fornitori )
  ON DELETE RESTRICT
  ON UPDATE CASCADE,
  FOREIGN KEY (Prodotti_Codice_Barre )
  REFERENCES Prodotti (Codice_Barre )
  ON DELETE RESTRICT
  ON UPDATE CASCADE,
  FOREIGN KEY (Operatore_Codice_Fiscale )
  REFERENCES Persone (Codice_Fiscale )
  ON DELETE RESTRICT
  ON UPDATE CASCADE);

```

```

CREATE TABLE Ordine (
  Data_Ordine DATE NOT NULL ,
  Destinatario_Codice_Fiscale VARCHAR(16) NOT NULL ,
  Prodotti_Codice_Barre VARCHAR(13) NOT NULL ,
  Quantita_Ordine INT NOT NULL DEFAULT 0 ,
  Evaso BOOLEAN DEFAULT FALSE ,
  Operatore_Codice_Fiscale VARCHAR(16) NOT NULL ,
  PRIMARY KEY (Data_Ordine, Destinatario_Codice_Fiscale, Prodotti_Codice_Barre) ,
  FOREIGN KEY (Prodotti_Codice_Barre )
  REFERENCES Prodotti (Codice_Barre )
  ON DELETE RESTRICT
  ON UPDATE CASCADE,
  FOREIGN KEY (Destinatario_Codice_Fiscale )
  REFERENCES Persone (Codice_Fiscale )
  ON DELETE RESTRICT
  ON UPDATE CASCADE,
  FOREIGN KEY (Operatore_Codice_Fiscale )
  REFERENCES Persone (Codice_Fiscale )
  ON DELETE RESTRICT
  ON UPDATE CASCADE);

```

BIBLIOGRAFIA

- [1] Cardinale Andrea “Yii: il miglior framework PHP” (2012)
<<http://www.andrea-cardinale.it/php/yii-il-miglior-framework-php.html>>
- [2] Marotta Ciro “Framework PHP a Confronto” (2011)
<<http://www.html.it/articoli/framework-php-una-comparazione-1/>>
- [3] Marotta Ciro “Guida Yii Framework” (2012)
<<http://www.html.it/guide/guida-yii-framework/>>
- [4] Ramez A. Elmasri, Shamkant B. Navathe, S. Castano, L. Pretto
“Sistemi di Basi di Dati. Fondamenti” (2004) Pearson Addison Wesley
- [5] Yii Software LLC - Fabio Ingala “La guida definitiva a Yii” (2008-2010)
<<http://www.yiiframework.com/doc/guide/1.1/it/index>>
- [6] Wikipedia “AJAX” (2013)
http://it.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML>
- [7] Wikipedia “Framework per applicazioni web” (2013)
<http://it.wikipedia.org/wiki/Framework_per_applicazioni_web>
- [8] Wikipedia “Framework” (2013)
<<http://it.wikipedia.org/wiki/Framework>>
- [9] Wikipedia “jQuery” (2013)
<<http://it.wikipedia.org/wiki/JQuery>>
- [10] Wikipedia “Model–view–controller” (2013)
<<http://en.wikipedia.org/wiki/Model-view-controller>>
- [11] Wikipedia “PHP” (2013)
<<http://it.wikipedia.org/wiki/Php>>
- [12] Wikipedia “PRADO Framework” (2013)
<http://en.wikipedia.org/wiki/PRADO_Framework>
- [13] Wikipedia “SQL” (2013)
<<http://it.wikipedia.org/wiki/SQL>>