Francesca Santucci  
Matricola: 278531

Danilo Argentiero  
Matricola: 274892

|  |
| --- |
| TOP ON CONCERT |
| toponconcert.com |
|  |
| Università degli studi di Perugia  Dipartimento di Matematica e Informatica  Progetto di Sistemi Aperti e Distribuiti |
|  |

TOP ON CONCERT

toponconcert.com

# INTERVISTA AL COMMITTENTE

• • •

# Intervista al committente

Ci viene richiesto di realizzare un Web Service che ci consenta di gestire la prenotazione (vendita e acquisto) di biglietti per dei concerti.

Partendo quindi da un’analisi dei requisiti andiamo a distinguere tre tipi principali di utenti del servizio:

* **Cliente:** è l’utente finale, colui che è interessato a visualizzare i concerti ed acquistare il relativo biglietto.  
  La sua interfaccia sarà costituita da una tabella all’interno della quale saranno presenti tutte le informazioni relative ai concerti disponibili nonché le funzionalità che gli consento di acquistare i biglietti degli stessi.
* **Organizzatore:** è l’organizzatore del concerto.

All’organizzatore vengono messe a disposizione una serie di funzionalità che gli consento di gestire (creare, modificare, eliminare) i vari concerti e il relativo numero di biglietti disponibili.

* **Utente non registrato:** è l’utente che naviga il sito senza aver effettuato l’accesso. La sua interfaccia sarà costituita da strumenti per la registrazione ed avrà a disposizione la lista dei concerti ma non sarà in grado di operare con essi (acquisto biglietto, aggiungi, elimina, modifica concerto).

Ogni Cliente e Organizzatore prima di poter usufruire delle varie funzionalità deve registrarsi al Web Service e successivamente autenticarsi.

È importante che i vari utenti abbiano interfacce distinte ed opportunamente sicure.

Ci viene richiesto inoltre che il Web Service sia disponibile solo in lingua italiana.

# ANALISI DEL TEAM DI SVILUPPO

• • •

TOP ON CONCERT

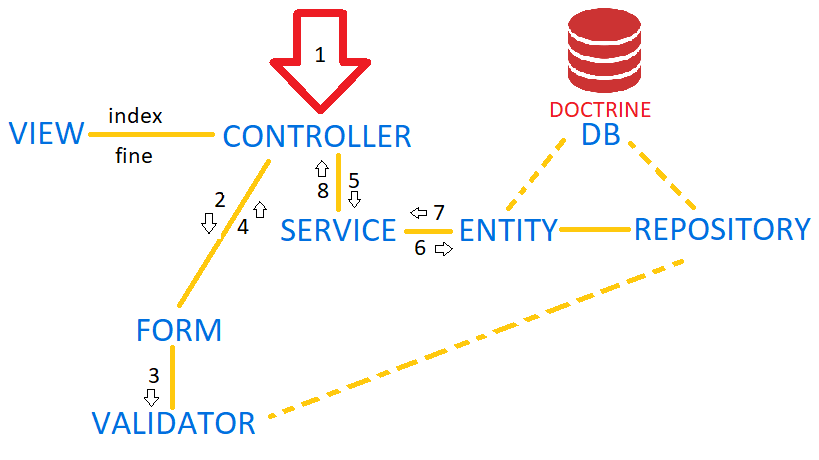
toponconcert.com

# Analisi di sviluppo: back-end

Per realizzare il back-end del Web Service richiesto dal committente abbiamo deciso di utilizzare Zend Framework 3, un’architettura logica di supporto per lo sviluppo di applicazioni web scritta in linguaggi PHP.

Zend Framework si basa su una serie di design-pattern tra cui troviamo MVC (Model-View-Controller)tanto che la struttura del nostro progetto rifletterà proprio quest’ultimo.

Di seguito mostriamo come sono interconnesse tutte le varie componenti del nostro progetto e a seguire ne diamo una breve descrizione:



TOP ON CONCERT

toponconcert.com

# ANALISI DEL TEAM DI SVILUPPO

• • •

1. View: visualizza i dati forniti dal Controller e si occupa dell’interazione con gli utenti.
2. Controller : riceve i comandi dell'utente (in genere attraverso la view o le REST API) e li esegue richiamando (ed eventualmente modificando) le altre componenti.
3. Form: usato per collezionare i dati inseriti dall’utente.
4. Validator: sono usati per verificare la correttezza dei dati inseriti.
5. Service: incapsulano alcune funzionalità logiche della nostra applicazione.
6. Entity: è una delle componenti fondamentali sfruttate da Doctrine.

È destinata alla memorizzazione di alcuni dati.

Nel nostro progetto le entità saranno: customer, organizer, concert e ticket.

1. Repository: è una componente di Doctrine. Responsabile del salvataggio e recupero delle entità.
2. Database: è l’archivio dei nostri dati.

**Caso d’uso:** supponiamo che l’utente debba registrarsi.  
Attraverso la View, che contiene una serie di elementi interattivi, l’utente richiamerà il Controller che gli mostrerà il Form da compilare per la registrazione. Una volta fatto, l’utente invierà tutti i dati inseriti di nuovo al Controller che, tramite la configurazione del Form e dei vari Validator, verificherà la correttezza dei campi inseriti. Se questi risultano essere validi allora il Controller li assegnerà allo userManager (Service) che genererà un’Entity che verrà salvata all’interno del Dababase.

Nel caso in cui invece l’utente ha bisogno di visualizzare la lista dei biglietti da lui acquistati entrerà in gioco anche il Repository che ci consente di recuperare una o più entità dal Database.

È importante notare che, nel nostro caso, abbiamo deciso di usare Mysql come DBMS e usufruiremo inoltre di Doctrine che ci semplificherà di molto l’esecuzione delle varie query alle tabelle del Database.

TOP ON CONCERT

toponconcert.com

# ANALISI DEL TEAM DI SVILUPPO

• • •

# Analisi di sviluppo: front-end

Per quanto riguarda il front-end abbiamo deciso di svilupparlo sfruttando Ionic Framework.   
Ionic Framework è uno strumento Open Source per lo sviluppo di applicazioni mobile ibride basato su Cordova e Angular.

Le comunicazioni e lo scambio delle informazioni tra l’applicazione mobile e l’applicazione server avviene tramite chiamate RESTful API.

REST è un insieme di linee guida per la progettazione e realizzazione di un Web Service (o più in generale di un’architettura di sistema).

Tali linee guida consistono in:

* **Identificazione delle risorse**
* **Utilizzo esplicito dei metodi http**
* **Risorse autodescrittive**
* **Collegamenti tra risorse**
* **Comunicazione senza stato**

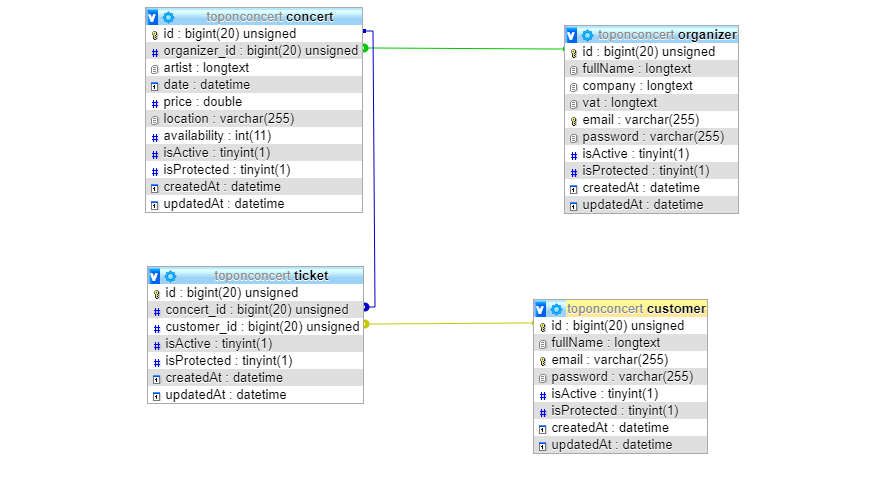
# STRUTTURA DEL DATABASE

• • •

TOP ON CONCERT

toponconcert.com

# Struttura del database:



Le relazioni sono le seguenti:

* **Customer Ticket 1 : n**
* **Concert Ticket 1 : n**
* **Organizer Concert 1 : n**

TOP ON CONCERT

toponconcert.com

# REST

• • •

# Chiamata REST:

In TOP ON CONCERT implementiamo delle chiamate REST per ottenere i concerti disponibili sia nell’interfaccia realizzata con Zend Framework sia in quella realizzata con Ionic.

Nel primo caso (cioè con Zend Framework) aggiungeremo al progetto un nuovo modulo che chiameremo API e nel suo file di configurazione specificheremo l’URL tramite la quale sarà possibile accedere a tale servizio.

Affinché la chiamata REST vada a buon fine occorre specificare che il modulo API dipende dal modulo Application all’interno del Module.php di API tramite le seguenti istruzioni:

public function **getModuleDependencies**() {

return [

'Application'

];

}

Una volta fatto, occorrerà solo definire un nuovo Controller che gestirà l’intera chiamata REST attraverso vari metodi:

* **getOptions():** restituisce la lista dei metodi HTTP consentiti;
* **getResetData():** unisce i dati ricevuti dall’utente tramite Json, Post o file;
* **onDispach():** verifica se le credenziali di autenticazione sono corrette prima di gestire il relativo controller.

TOP ON CONCERT

toponconcert.com

# Chiamata REST - continuo:

Nel secondo caso (Ionic) analogamente al primo, realizzeremo la chiamata RESTful all’interno del controller che gestisce la stampa dei vari concerti nel seguente modo (esempio in JavaScript):

.controller('ConcertsCtrl', function ($rootScope, $scope) {

$scope.concerts = [];

$.getJSON('http://toponconcert.com/api/V1/concert', function(data){

$scope.concerts = data.concerts;

$rootScope.$applyAsync();

});

})