# *Progetto di Sistemi Aperti e Distribuiti*

**Progettazione di un Web Service in grado di offrire servizi per la produzione e prenotazione di eventi musicali.**

Andrea Castellani Perelli

-

Francesco Gradi

# Descrizione

L'applicazione è divisa in **client** e **server** ed è strutturata in modo tale da poter consentire due tipi di accesso diversi: l'accesso come *cliente* e l'accesso come *organizzatore*.

Sia l'organizzatore che il cliente possono registrarsi nel database inserendo le seguenti informazioni: nome, cognome, e-mail, password, compagnia organizzativa (solo per l'organizzatore).

Dopo aver effettuato il login, l'*organizzatore* ha la possibilità di:

- vedere la lista degli eventi attualmente prenotabili.
- crearne di nuovi.
- eliminare unicamente eventi organizzati dalla sua compagnia.

Il *cliente* ha invece la possibilità di:

- vedere la lista degli eventi attualmente prenotabili.
- acquistarne a piacimento.

Frameworks utilizzati:

- *Sails.js* per la parte Server
- *Ionic* per la parte Client

Sia Web Service che client sono stati scritti in **Javascript**. Il Web Service è stato caricato online su Heroku (piattaforma cloud per la distribuzione di applicazioni online) in modo che il client possa connettersi ad esso ed effettuare chiamate specifiche di tipo REST.

# Le chiamate REST

Le chiamate REST possibili (che rispettano la notazione addottata in Sails.js) sono:

```
GET /events -> EventsController.find
GET /events/:id -> EventsController.findOne
POST /events -> EventsController.create
PUT /events/:id -> EventsController.update
DELETE /events/:id -> EventsController.destroy
```

di cui alcune effettuate dal client:

```
$http.post(link, {email : this.email, password : this.password}).then(function (res){
    $scope.response = res.data.message;
    organizzatore = res.data.org
    orgCompany.addCompany(organizzatore.company);
    if(res.data.message === "Logged In Successfully"){
        window.location.href = "#/page7";
    }
    else {
        var myPopup = $ionicPopup.show({
            title: 'ERROR',
            subTitle: res.data.message,
            buttons: [{text: 'OK', type: 'button-positive'}]
        });
    }
});
```

```
$http.delete(link).then(function (res){
    var myPopup = $ionicPopup.show({
        title: 'Evento eliminato',
        buttons: [{text: 'OK', type: 'button button-positive button-small'}]
    });
```

Il database utilizzato è quello che offre sails stesso (*sails-disk-db*) ed è costituito da 3 tabelle:

- *User* – per i clienti. Contiene nome, cognome, email, password, id.
- *Organizzatore* – per gli organizzatori. Contiene gli stessi campi di User con l'aggiunta del campo compagnia.
- *Events* – per gli eventi. Contiene i campi artista, luogo, data, prezzo, compagnia dell'organizzatore che crea l'evento.

Per ragioni di sicurezza è stato definito un file specifico ("AuthController.js") che, oltre ad assicurarsi che il login vada a buon fine, impedisce l'accesso alla lista degli eventi da parte di utenti che non hanno effettuato il login.

Per evitare inoltre che un organizzatore qualsiasi possa cancellare eventi di altre compagnie diverse dalla sua, è stato necessario impostare una politica di recupero della compagnia organizzativa al momento del login dell'organizzatore stesso per poi riempire automaticamente il campo "compagnia" della tabella eventi al momento della creazione di un nuovo evento. In questo modo quando un organizzatore cerca di cancellare un evento specifico, il sistema controlla se il campo compagnia dell'evento corrisponde al campo compagnia dell'organizzatore in questione. In caso negativo, impedisce a quell'organizzatore di cancellare quell'evento.