Progetto di Sistemi Aperti e Distribuiti

Progettazione di un Web Service in grado di offrire servizi per la registrazione e prenotazione di ristoranti.

Lorenzo Gigli

Raffaele Raggi

A.A. 2017/2018

Descrizione

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

- ***Cliente***
- **R*istoratore***

Sia il Ristoratore che il Cliente possono registrarsi nel database inserendo le seguenti informazioni: nome, cognome, e-mail, password e compagnia (solo per il ristoratore).

Il campo "compagnia" serve come termine di controllo per identificare in modo univoco un ristoratore.

Dopo aver effettuato il login, il **Ristoratore** ha la possibilità di:

- vedere la lista dei ristoranti attualmente prenotabili.
- crearne di nuovi.
- eliminare unicamente *i suoi* ristoranti.

Il ***Cliente*** ha invece la possibilità di:

- vedere la lista dei ristoranti attualmente prenotabili.
- prenotare a piacimento.

<u>Frameworks</u> 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 /rests --> RistoratoriController.find
GET /rests:id --> RistoratoriController.findOne
POST /rests --> RistoratoriController.create
PUT  /rests:id --> RistoratoriController.update
DELETE /rests:id --> RistoratoriController.destroy
```

esempi di GET e POST utilizziati:

```
$http.get(link).then(function (res){
    rests = res.data;
    $scope.rests = rests;
```

```
$http.post(link, {email : this.email, password : this.password}).then(function (res){
    $scope.response = res.data.message;
    ristoratore = res.data.org
    orgCompany.addCompany(ristoratore.company);
    console.log(res.data);
    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-assertive'}]
        });
    }
});
```

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.
- ***Ristoratore*** – per i Ristoratori. Contiene gli stessi campi di User con l'aggiunta del campo compagnia.
- ***Ristoranti*** – per i Ristoranti. Contiene i campi *nome*, *indirizzo*, *orario apertura*, *orario chiusura*, *turno*, *email*, *telefono*, *posti*, *compagnia* del ristoratore che aggiunge il ristorante.

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 dei ristoranti da parte di utenti che non hanno effettuato il login.

Per evitare che un ristoratore qualsiasi possa cancellare ristoranti non suoi, è stato necessario impostare una politica di recupero del ristorante al momento del login del ristoratore stesso, per poi riempire automaticamente il campo "compagnia" della tabella ristoranti al momento della creazione di un nuovo ristorante.

In questo modo quando un ristoratore cerca di cancellare un ristorante specifico, il sistema controlla se il campo "compagnia" del ristorante corrisponde al campo "compagnia" del ristoratore in questione. In caso negativo, impedisce a quel ristoratore di cancellare quel ristorante.