MUSE

Aplicación para los que buscan inspiración

Indice

Presentación La idea inicial

Muse surgió de la idea de que los artistas no siempre encuentran su musa y una ayudita nunca viene mal. Por ello he creado esta aplicación para Android usando Ionic como tecnología en la que artistas de todos los niveles puedan encontrar referencias o subir las suyas propias. De ahí el lema “Busca la inspiración o ayuda a otros a que la encuentren”.

◄Esta pequeña es Muse, la mascota de la aplicación y musa de los dibujantes.

Requisitos

Para el correcto funcionamiento de la aplicación es necesario que esta cuente con un menú sencillo y claro que facilite al usuario acceder a todas las secciones. Además, deberán existir las secciones de subida de fotos, búsqueda y, por supuesto, el “catálogo” con las categorías y las imágenes subidas. Como extra, la opción de subir imágenes no sólo desde la cámara si no a través de url.

Creando la aplicación paso a paso

Para empezar a crear la aplicación ha sido necesaria la instalación de diferentes programas en nuestro equipo.

- Instalamos node.js y npm desde http://nodejs.org/
- Nos ponemos a instalar Bower, Gulp y Yeoman:
  - Bower npm install -g bower
  - Gulp npm install -g gulp
  - Yeoman npm install -g yo
- Instalamos las últimas versiones de cordova e ionic desde la consola de comandos a través del comando npm install -g cordova ionic
- Nos aseguramos de tener instalados jdk, sdk de android y ant (en mi caso usaré winant):
  - Para el jdk http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - Para el sdk de android http://developer.android.com/intl/es/sdk/index.html , descargamos el sdk manager y descargamos los sdk que queremos en nuestro caso las de android 4.4 y

android 5.

- Para winant https://code.google.com/p/winant/ (Windows installer for Apache Ant)

- Añadiremos los path de java, android y winant a las variables de entorno de la siguiente manera:
  - Abrimos la información del sistema, hacemos click en configuración avanzada, en la pestaña de opciones avanzadas hacemos click en variables de entorno. Una vez en esa ventana añadimos los path en las variables del sistema haciendo click en nueva y creando cada una de la siguiente forma:
  - JAVA_HOME C:\Program Files (x86)\Java\jdk1.8.0
  - ANDROID_HOME C:\Users\Nezumi\AppData\Local\Android\android-sdk
  - ANT_HOME C:\Users\Nezumi\WinAnt
  - Y modificamos path añadiendo las nuevas variables que le hemos añadido anteriormente:
  - path c:\windows\system32;c:\windows;C:\Program Files\nodejs; %ANT_HOME%\bin; %JAVA_HOME%\bin;%ANDROID_HOME%\tools;%ANDROID_HOME%\platform-tools;

Nota de la autora: Actualmente todo esto de los path da problemas pero se ha podido solucionar gran parte de los errores que daba el pc modificando los path.

- Tenemos que contar además con un editor de texto apropiado, en este caso usamos notepad++ pero también podían funcionar otros como sublime-text.

Iniciando el proyecto

Primero elegimos el tipo de proyecto inicial que queremos crear. Ionic pone a nuestra disposición tres plantillas con las cuales poder empezar nuestro proyecto. En nuestro caso hemos optado por la del sidemenu y para crearlo metemos por comandos desde la carpeta que va a contener los archivos la siguiente linea `ionic start “nombre de la aplicación” sidemenu`

Nota de la autora: como curiosidad diré que, en un primer intento de crear la aplicación, dejé el nombre por defecto pero la segunda vez le llamé seccondApp antes de que se me ocurriera el nombre de Muse y no supe como cambiarle el nombre una vez creada.

El comando start crea la base de la aplicación en la carpeta elegida que contiene varios archivos para facilitar la programación del mismo.

Añadiendo el contenido

Una vez creado el proyecto, contamos en nuestro escritorio con una carpeta con las bases para empezar a trabajar. En esta fase creamos el contenido de nuestra aplicación y le damos algo de forma.

- Dentro de la carpeta “www” se encuentran el índice de la aplicación, las plantillas, estilos y archivos .js que usamos.

Index.html

Este archivo es el que va a contener los elementos comunes de la aplicación. Lo hemos modificado ligeramente para añadir un pie de página. Desde este archivo llamamos a todas las hojas de estilo y los archivos javascript.

- Dentro de la carpeta “templates” se encuentran las demás páginas que componen la aplicación.

Menu.html

Este es el archivo donde se encuentran las secciones y hacia donde redirecciona cada una de ella.

El menú aparece al hacer click en el icono de las tres rallitas de la esquina superior izquierda.

He añadido un icono a cada sección para ilustrar algo mejor lo que representa cada una.

Home.html

Esta es la página principal, lo primero que se muestra al iniciar la aplicación.

Login.html

Esta sección despliega un modal que permite al usuario acceder a su cuenta.

Search.html

Sección de búsqueda para encontrar cosas más concretas más rápido. Contiene un input de tipo búsqueda desde el cual recoger la información y un botón para enviarla.

Browse.html

Desde esta página se puede tanto crear categorías como subir imágenes tanto por cámara como por url. Contiene un pequeño formulario por si queremos añadir categorías, una sección para subir por url y otra para subir por captura de cámara.

Nota de la autora: Durante la creación de esta página siempre ha habido el problema de que el botón para tomar las fotos quedaba tapado por el footer. Para solucionarlo, y a pesar de ser un poco cutre, insertamos un div en blanco.

Playlists.html

Página principal que muestra todas las categorías.

Playlist.html

Página que muestra las sub-categorías.

Imagenes.html

En este archivo se especifica el formato de las imágenes, en este caso en forma de card list.

Detalles de los archivos javascript

- En la carpeta de js se encuentran los archivos javascript. A continuación explicamos cada archivo y comentamos su contenido:

Angular-resource.js

Archivo incluído donde se encuentran las fuentes de angular.

App.js

```
// Ionic Starter App

angular.module('starter', ['ionic', 'starter.controllers', 'starter.categories', 'starter.config', 'ngCordova', 'ngResource'])

.run(function($ionicPlatform) {
```

```
  $ionicPlatform.ready(function() {

    if (window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if (window.StatusBar) {
      // org.apache.cordova.statusbar required
      StatusBar.styleDefault();
    }
  });
})
//Desde aquí se controla que el menú vaya a cada sección correspondiente
.config(function($stateProvider, $urlRouterProvider) {
  $stateProvider

  .state('app', {
    url: "/app",
    abstract: true,
    templateUrl: "templates/menu.html",
    controller: 'AppCtrl'
  })

  .state('app.home', {
    url: "/home",
    views: {
      'menuContent': {
        templateUrl: "templates/home.html"
      }
    }
  })

  .state('app.search', {
    url: "/search",
    views: {
      'menuContent': {
        templateUrl: "templates/search.html"
      }
    }
  })
```

```
  .state('app.browse', {
    url: "/browse",
    views: {
      'menuContent': {
        templateUrl: "templates/browse.html"
      }
    }
  })


    .state('app.playlists', {
      url: "/playlists",
      views: {
        'menuContent': {
          templateUrl: "templates/playlists.html",
          controller: 'CategoryListCtrl'
        }
      }
    })

  .state('app.single', {
    url: "/playlist/:category_id",
    views: {
      'menuContent': {
        templateUrl: "templates/playlist.html",
        controller: 'ItemDetailCtrl'
      }
    }
  })


  .state('app.images', {
    url: "/images/:item_id",
    views: {
      'menuContent': {
        templateUrl: "templates/images.html",
        controller: 'ImageDetailCtrl'
      }
    }
  });
  // if none of the above states are matched, use this as the fallback
```

nota de la autora: haciendo pruebas con los controladores de las sub-categorías me di cuenta de que algo iba mal por que me redirigía a la página de inicio.

```
  $urlRouterProvider.otherwise('/app/home');
});
```

Categories.controller.js

```
//Este archivo conecta con la parte de REST de los países usada para hacer pruebas

'use strict';
var countryController = angular.module('app.data.controllers', []);

countryController.controller('CountryListCtrl', function ($location, $scope, Country) {
 Country.query(function (data) {
 $scope.countries = data;
 });
 $scope.insert = function (currentCountry) {
 console.log("llega ok." + currentCountry.code);
 Country.add({}, currentCountry);
 $location.path('/countries');
 };
 $scope.remove = function (currentCountry) {
 Country.remove({id: id}, {}, function (data) {
 $location.path('/');
 });
 };

});
countryController.controller('CountryDetailCtrl', ['$location', '$scope', '$routeParams', 'Country'
 , function ($location, $scope, $routeParams, Country) {
 $scope.country = Country.get({id: $routeParams.id}, function (country) {
 console.log("carga ok");
 });
 $scope.update = function (currentCountry) {
 Country.update({id: $scope.country.code}, currentCountry, function (data) {
 $location.path('/');
 });
 };

 }]);
```

Categories.js

//Este archivo es el que permite la relación de las categorías con sus correspondientes en el REST además de la subida de categorías e imágenes.

Cuando un usuario accede a la sección de categorías este código activa el controlador asignado haciendo que realice una consulta en el servidor REST con la que obtiene las categorías guardadas en la base de datos y las pasa a la página html. En esta página, usando el ng-repeat, mostramos todo el contenido de las categorías. En el config.js se tiene que configurar el acceso a la base de datos asignando a una variable categories los datos para acceder a REST. Una vez hecho esto, los datos se usarán en el factory de categories donde se realiza la consulta por id a la base de datos que usará el controlador.

```
angular.module('starter.categories',[])
.factory("Category", function (RESOURCES, $resource) {
   console.log("factory: Category");
   return $resource(RESOURCES.CATEGORIES + "/:id", null,
      {
         query: {method: 'GET', isArray: true},
         get: {method: 'GET', isArray: true},
         add: {method: 'POST'},
         delete: {method: 'DELETE'},
         update: {method: 'PUT'} /*,params:{id:'@code'}*/
      });
})
.factory("Item", function (RESOURCES, $resource) {
   console.log("factory: Item");
   return $resource(RESOURCES.ITEMS + "/:id", null,
      {
         query: {method: 'GET', isArray: true},
```

```
            get: {method: 'GET', isArray: true},
            add: {method: 'POST'},
            delete: {method: 'DELETE'},
            update: {method: 'PUT'} /*,params:{id:'@code'}*/
        });
})

.factory("Images", function (RESOURCES, $resource) {
    console.log("factory: Images");
    return $resource(RESOURCES.IMAGES + "/:id", null,
        {
            query: {method: 'GET', isArray: true},
            get: {method: 'GET', isArray: true},
            add: {method: 'POST'},
            delete: {method: 'DELETE'},
            update: {method: 'PUT'} /*,params:{id:'@code'}*/
        });
})

.factory('Camera', ['$q', function($q) {
  return {
    getPicture: function(options) {
      var q = $q.defer();
      navigator.camera.getPicture(function(result) {
        q.resolve(result);
      }, function(err) {
        q.reject(err);
      }, options);
```

```
      return q.promise;
    }
  }
}]);
```

Config.js

Como ya hemos dicho en la descripción del archivo categories.js, en este archivo se configura el acceso a la base de datos asignando a una variable categories los datos para acceder a REST.

```
var myApp = angular.module("starter.config",[]);
myApp.constant('RESOURCES', (function() {
    var resource = 'http://www.planificaciondeportiva.es/bmoll-app';
    var activeApi = '/api/web/v1';
    return {
        USERS_DOMAIN: resource,
        USERS_API: resource + activeApi,
        COUNTRIES: resource + activeApi+ '/countries',
        CATEGORIES: resource + activeApi+ '/categories',
        ITEMS: resource + activeApi+ '/items',
        IMAGES: resource + activeApi+ '/images'
    }
})());
/* Definimos soporte de formularios para objetos JSON
 http://cacodaemon.de/index.php?id=44 */
angular.module('starter').config(function ($httpProvider) {
    $httpProvider.defaults.transformRequest = function (data) {
        var str = [];
        for (var p in data) {
            data[p] !== undefined && str.push(encodeURIComponent(p) + '=' +
encodeURIComponent(data[p]));
```

```
        }
        return str.join('&');
    };
    $httpProvider.defaults.headers.put['Content-Type'] =
$httpProvider.defaults.headers.post['Content-Type'] =
        'application/x-www-form-urlencoded; charset=UTF-8';
});
```

Controllers.js

//Controladores del modal de login y la generación de las secciones de categorías según lo obtenido de REST. También está el controlador de la cámara

```
angular.module('starter.controllers', [])
//Comienza los controladores del modal de login de apertura, cierre o simulación de retraso al logear.
.controller('AppCtrl', function($scope, $ionicModal, $timeout) {
  $scope.loginData = {};

  $ionicModal.fromTemplateUrl('templates/login.html', {
    scope: $scope
  }).then(function(modal) {
    $scope.modal = modal;
  });

  $scope.closeLogin = function() {
    $scope.modal.hide();
  };

  $scope.login = function() {
    $scope.modal.show();
  };

  $scope.doLogin = function() {
    console.log('Doing login', $scope.loginData);

    $timeout(function() {
      $scope.closeLogin();
    }, 1000);
  };
```

```
})

//Supuesto controlador que debería haber metido unas categorías por defecto en vez de las de REST

.controller('PlaylistsCtrl', function($scope) {
  $scope.playlists = [
   { title: 'Poses', id: 1 },
    { title: 'Animals', id: 2 },
    { title: 'Clothes', id: 3 },
    { title: 'Scene', id: 4 },
    { title: 'Misc', id: 5 },
  ];
})

//Controladores que supervisan el comportamiento de las listas de categorías.

.controller('PlaylistCtrl', function($scope, $stateParams) {
});

.controller('CategoryListCtrl', function ($location, $scope, Category) {
    Category.query(function (data) {
        $scope.categories = data;
    });
    $scope.insert = function (currentCategory) {
        console.log("llega ok." + currentCategory.code);
        Category.add({}, currentCategory);
        $location.path('/categories');
    };
    $scope.remove = function (currentCategory) {
        Category.remove({id: id}, {}, function (data) {
            $location.path('/');
        });
    };

})
.controller('CategoryDetailCtrl', ['$location', '$scope', '$stateParams', 'Category'
    , function ($location, $scope, $stateParams, Category) {
        $scope.Category = Category.get({id: $stateParams.id}, function (Category) {
            //$scope.mainImageUrl = phone.images[0];
```

```
            console.log($scope.Category);
        });
        $scope.update = function (currentCategory) {
            Category.update({id: $scope.Category.code}, currentCategory, function (data) {
                $location.path('/');
            });
        };

    }])

.controller('ItemListCtrl', function ($location, $scope, Item) {
  console.log(Item);
    Item.query(function (data) {
        $scope.items = data;
    });
    $scope.insert = function (currentItem) {
        console.log("llega ok." + currentItem.code);
        Item.add({}, currentItem);
        $location.path('/items');
    };
    $scope.remove = function (currentItem) {
        Item.remove({id: id}, {}, function (data) {
            $location.path('/');
        });
    };

})


.controller('ItemDetailCtrl', function($location, $scope, $stateParams, Item) {
   var items = Item.query({category_id: $stateParams.category_id}, function (data) {
          console.log(items);
          $scope.Category = $stateParams;
          $scope.Item = [];
         for(x=0; x<items.length; x++) {
              if (items[x].category_id == $stateParams.category_id ){
                $scope.Item.push(items[x]);
              }
         }
           });
```

```
        $scope.update = function (currentItem) {
            Item.update({id: $scope.Item.code}, currentItem, function (data) {
                $location.path('/');
            });
        };

    })

.controller('ImagesListCtrl', function ($location, $scope, Images) {
    Images.query(function (data) {
        $scope.images = data;
    });
    $scope.insert = function (currentImages) {
        Images.add({}, currentImages);
        $location.path('/images');
    };
    $scope.remove = function (currentImages) {
        Images.remove({id: id}, {}, function (data) {
            $location.path('/');
        });
    };

})

// Este controlador se encarga de recoger la información de la cámara. La idea surgió de aquí
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=AccesoCamaraPhoneGap


.controller("CallCam", function($scope, $cordovaCamera) {

    $scope.takePicture = function() {
        var options = {
            quality : 75,
            destinationType : Camera.DestinationType.DATA_URL,
            sourceType : Camera.PictureSourceType.CAMERA,
            allowEdit : true,
            encodingType: Camera.EncodingType.JPEG,
            targetWidth: 300,
            targetHeight: 300,
```

```
            popoverOptions: CameraPopoverOptions,
            saveToPhotoAlbum: true
        };

        $cordovaCamera.getPicture(options).then(function(imageData) {
            $scope.imgURI = "" + imageData;
        }, function(err) {
            // An error occured. Show a message to the user
        });
    }

})

.controller('ImageDetailCtrl', function($location, $scope, $stateParams, Images) {
        var images = Images.query({item_id: $stateParams.item_id}, function (data) {
          $scope.Images = [];
          for(x=0; x<images.length; x++) {
              if (images[x].item_id == $stateParams.item_id ){
                $scope.Images.push(images[x]);
              }
          }
        });
        $scope.update = function (currentItem) {
            Item.update({id: $scope.Item.code}, currentItem, function (data) {
                $location.path('/');
            });
        };
    });

;
```

Ng-cordova-min.js

Este archivo contiene un repositorio de angular para cordova y es tan extenso que no querría tener que explicarlo.

Diseño responsive y grid system

Esta aplicación tiene, entre otros, el propósito de ser responsive y para ello usamos las guías

proporcionadas por ionic para el diseño que podemos encontrar aquí http://ionicframework.com/docs/components/

Mediante el uso de las clases proporcionadas por ionic conseguimos que la aplicación se vea bien tanto en navegador como en un dispositivo móvil.

Otra de las fuentes usadas en esta aplicación es esta página http://ionicons.com/ desde la que hemos elegido los iconos que querían que aparecieran.

Dinamizando el contenido

Instalamos POSTMAN en Chrome como extensión y seguimos la siguiente guía: http://www.planificaciondeportiva.es/learntic/media_diw/ConsumoServiciosREST.pdf

nota de la autora: para que funcione el uso de datos de REST hace falta tener en Chrome CORS activado.

https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfghkpbjhddihlkkiljbi?utm_source=chrome-app-launcher-info-dialog

Intenta conectarte a http://www.planificaciondeportiva.es/bmoll-app/api/web/v1/

- Importa los contenidos de este archivo file en las colecciones de Postman
- Usa las siguientes direcciones

- categories http://www.planificaciondeportiva.es/bmoll-app/api/web/v1/categories
- items http://www.planificaciondeportiva.es/bmoll-app/api/web/v1/items
- images http://www.planificaciondeportiva.es/bmoll-app/api/web/v1/images

Viendo el progreso de nuestro proyecto

Durante la creación del proyecto lo lógico es ir mirando si el proyecto avanza correctamente o ha dejado de funcionar. Para ello usamos el comando ionic serve desde la carpeta contenedora para que se muestre en el navegador. Además, Chrome permite la opción de ver nuestra aplicación no sólo como si fuera para escritorio si no también para móvil desde el inspeccionador de elementos. No obstante, también existe la opción de usar el comando ionic serve -lab para mostrar como se vería en un dispositivo móvil android o iphone.

Creación del splash y el icon

Desde la consola de comandos, en la carpeta del proyecto escribimos:

ionic resources –icon

ionic resources –splash

Esto debería crear los iconos y las imágenes necesarias para nuestra aplicación.

Compilando la apk

Antes de compilar la aplicación debemos añadir una plataforma a nuestro proyecto. En este caso hemos optado por android como ya hemos dicho anteriormente.

Para añadir la plataforma usamos el comando:

ionic platform add android

Luego, desde la consola de comandos, en la carpeta del proyecto escribimos:

cordova build android

Y esto ha creado la apk.

Enfoque técnico

Además de todo lo creado por ionic desde la creación de la aplicación, podemos decir que en realidad ésta consta de un documento html llamado index.html que es la base y en el cual se inicializan todos los ficheros javascript que permiten el correcto uso de la aplicación y su conexión a REST. Por otra parte se encuentran las ocho plantillas también de tipo html.

Casos de uso

Por parte de la aplicación

La aplicación permite que las fotos subidas por el usuario y las nuevas categorías se guarden en el servicio REST y devuelve los datos que se han pedido del servicio REST.

Por parte del usuario

Un usuario puede realizar fotos y subirlas, crear nuevas categorías, realizar búsquedas y consultar tanto las categorías como las imágenes que pueden contener.

Nota de la autora: una de las opciones que no está implementada es la de mostrar un mensaje en la sección en el caso de que no exista ninguna imagen.

Uso

Para facilitar el uso de la aplicación a continuación detallaré las diferentes secciones de las que cuenta:

Home

La sección de Home muestra la página principal con una pequeña introducción a la finalidad de la aplicación y la mascota de la aplicación, la musa, que nos da la bienvenida.

Login

Al pulsar en el login se te permite acceder con tu cuenta al contenido que has subido.

Busca

La sección de búsqueda facilita aún más el encontrar aquello concreto que se busca. Se implementará la opción de búsqueda en siguientes versiones.

Sube

La sección de subida permite al usuario añadir nuevas categorías y subir imágenes tanto por cámara como por url.

Categorías

Esta sección es la que muestra todas las categorías que se encuentran en el servicio REST.

Nota de la autora: al menú se puede acceder desde cualquiera de las secciones.