Aplicaciones móviles

PID_00254187

Àlex Bartrolí Muñoz

Universitat Oberta de Catalunya

Índice

1. Introducción

Existen tres formas principales para desarrollar aplicaciones móviles:

- Aplicaciones nativas: aquellas que están íntegramente programadas en un lenguaje y entorno de programación desarrollado específicamente para cada sistema operativo (Java para Android, Objective-C para iOS y C# para Windows Phone).

- Aplicaciones móviles con estándares web: completamente desarrolladas en HTML5.

- Aplicaciones híbridas: aplicaciones creadas en parte con el entorno de desarrollo nativo y en parte con lenguaje web (HTML5).

Hasta día de hoy las aplicaciones nativas han aprovechado mejor el rendimiento de los dispositivos. Para desarrollar proyectos que requieran un uso intensivo de los recursos del dispositivo es aconsejable desarrollar la aplicación en lenguaje nativo, aunque estas tienen el inconveniente de que son más costosas de desarrollar y mantener.

En los siguientes enlaces tenéis una comparativa entre las diferentes formas de crear una aplicación móvil:

- 150 Mobile platforms

- Aplicaciones web nativas híbridas

2. Desarrollo de aplicaciones híbridas con PhoneGap

En este tutorial veremos cómo crear aplicaciones híbridas con PhoneGap.

Las fases para desarrollar las aplicaciones móviles con PhoneGap son:

1) Creación de una aplicación web.

2) Utilización de un emulador para testear la aplicación.

3) Creación de la aplicación y descarga de la aplicación en nuestro teléfono móvil.

3. Creación de aplicaciones móviles con estándares Web

Cada página de una aplicación móvil tiene la siguiente estructura:

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" type="text/css" href="css/index.css" />
        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.0/jquery.mobile-
            1.3.0.min.css" />
        <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
        <script src="http://code.jquery.com/mobile/1.3.0/jquery.mobile- 1.3.0.min.js"></script>
        <title>Hello World</title>
</head>
```

3.1. Estructura del header

Con el tag viewport indicamos al dispositivo que la anchura del dispositivo debe ser la anchura total de la pantalla del dispositivo.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

La otra parte importante del header es la declaración de las bibliotecas jQuery y jQuery mobile.

En este ejemplo tomamos las bibliotecas del repositorio de jQuery, pero si queremos que el usuario pueda utilizar la aplicación sin tener que estar conectado a internet, tendremos que descargar las bibliotecas en local y enlazarlas con su camino relativo.

3.2. Estructura del body

Para que jQuery mobile reconozca cada parte del documento, es importante definir los data-role de cada puerto del body.

Con `<div id="index" data-role="page">` indicamos que este contenedor es de tipo «pagina» y que esta página se llama index.

Con `<div data-role="header">` indicamos que se trata de la cabecera de la aplicación.

Con `<div data-role="content">` indicamos que es el contenido de la página.

Y con `<div data-role="footer">` indicamos que es el footer de la página.

La estructura básica de una página HTML5 para crear una aplicación móvil será:

```
<body>
        <div id="index" data-role="page">
            <div data-role="header" data-position="fixed">
                   <h1>Hello Word Header</h1>
            </div>
            <div data-role="content">
                  Hello world Content!
            </div>
            <div data-role="footer">
                  <h2>Hello Word Footer</h2>
            </div>
        </div>
    </body>
</html>
```

4. Uso de un emulador para testear la aplicación

Phonegap dispone de un emulador en línea para testear nuestras aplicaciones.

Para utilizarlo primero tendremos que instalarnos una extensión de Google Chrome llamada Ripple Emulator1 y después ir al emulador de aplicaciones de PhoneGap utilizando el navegador Google Chrome: http://emulate.phonegap.com.

Para emular la aplicación solo hace falta introducir la dirección de nuestra aplicación web, que en la mayoría de los casos estará en nuestro servidor en local.

Antes de publicar la aplicación, tendremos que asegurarnos de que solo tenemos caminos relativos y que aquellos caminos que tengan que ser absolutos deberán pertenecer a una URL accesible (es decir, no podrán formar parte de nuestro localhost).

Podéis descargar la extensión Ripple Chrome en:

https://chrome.google.com/webstore/detail/ripple-emulator-beta/geelfhphabnejjhdalkjhgipohgpdnoc?hl=en

5. Creación de la aplicación y descarga en nuestro teléfono móvil

Una vez testeada la aplicación en el emulador, podemos publicarla e instalarla en dispositivos móviles.

Antes de crear la aplicación con PhoneGap, tenemos que realizar dos tareas:

- Comprimir en un .zip todos los archivos de nuestra aplicación.
- Instalar un lector de códigos QR.

Cuando tengamos los archivos comprimidos, nos dirigimos a: https://build.phonegap.com/.

Si es la primera vez que nos conectamos, nos pedirá registrarnos en GitHub o en Adobe.

Cuando nos hayamos dado de alta en alguno de estos dos sistemas, nos identificamos en build.phonegap.com, nos pedirá subir el archivo .zip que acabamos de crear y creará la aplicación.

Una vez creada, para instalar la aplicación en nuestro dispositivo móvil solo tendremos que leer la imagen QR que se nos mostrará en pantalla y la aplicación se instalará automáticamente en nuestro móvil.

6. Interacción de la aplicación con el dispositivo móvil

Hasta el momento hemos visto cómo realizar aplicaciones con HTML, jQuery y CSS, pero no hemos visto cómo interactuar con los componentes del dispositivo móvil.

Con la API de PhoneGap tenemos acceso a los siguientes componentes móviles.

Acelerómetro

Captura el movimiento del dispositivo en la dirección *x*, *y* y *z*.

Cámara

El objeto Cámara proporciona acceso a la aplicación de cámara por defecto del dispositivo.

Captura

Proporciona acceso al audio, a la imagen y a las capacidades de captura de vídeo del dispositivo.

Brújula

Obtiene la dirección a la que apunta el dispositivo.

Conexión

El objeto Connection, expuesto a través de navigator.connection, proporciona información sobre la conexión wifi del dispositivo.

Contactos

El objecto Contacts proporciona acceso a la base de dados de contactos del dispositivo.

Dispositivo

El objeto Device describe el hardware y software del dispositivo.

Archivo

Una API para leer, escribir y navegar por las jerarquías de sistema de archivos, basadas en la API de archivos del W3C.

Geolocalización

El objeto Geolocation proporciona acceso a los datos de localización basados en el sensor GPS del dispositivo o deducidos a partir de las señales de la red.

Globalización

Obtiene información y realiza las operaciones específicas de la configuración regional del usuario y su zona horaria.

7. Aplicación web con varias páginas, recuperación dinámica de datos y Google Maps

A continuación mostramos el código de una aplicación web que contiene dos páginas con información sobre pizzerías.

La primera página de la aplicación muestra un listado de pizzerías después de recuperar los datos desde un servidor público con una llamada JSON.

En esta página, cada restaurante tiene un enlace, que lleva a una página nueva con los detalles del restaurante.

La página de detalle del restaurante recupera los datos del restaurante, actualiza el contenido de las etiquetas HTML de la página y muestra un mapa de Google con la situación del restaurante.

Código HTML

```
<!DOCTYPE html>
<html>
```

Cabecera del fichero HTML con enlaces al código JavaScript y a las bibliotecas de Google Maps, jQuery y jQuery mobile:

```
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" type="text/css" href="css/index.css" />
        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.0/jquery.mobile-
            1.3.0.min.css" />
        <link rel="stylesheet" type="text/css" href="css/index.css" />
        <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
        <script src="http://code.jquery.com/mobile/1.3.0/jquery.mobile- 1.3.0.min.js">
           </script>
       <!-- Librerías para google maps -->
       <script type="text/javascript" src="http://maps.googleapis.com/maps/api/js? sensor=false">
          </script>
       <link href="http://code.google.com/apis/maps/documentation/javascript/examples/default.css"
           rel="stylesheet" type="text/css" />
        <title>Pizza World</title>
    </head>
```

Código HTML del body de la página

Es muy similar al ejemplo de «hello world», pero en este caso el elemento `<div data-role="content">` contiene un elemento `<UL>` para añadir la lista de restaurantes.

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.0/jquery.mobile-
        1.3.0.min.css" />
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
    <script src="http://code.jquery.com/mobile/1.3.0/jquery.mobile- 1.3.0.min.js"></script>
    <!-- Librerías para google maps -->
    <script type="text/javascript" src="http://maps.googleapis.com/maps/api/js? sensor=false">
       </script>
    <link href="http://code.google.com/apis/maps/documentation/javascript/examples/default.css"
        rel="stylesheet" type="text/css" />
    <title>Pizza World</title>
</head>
```

Código JavaScript para la página index.html

Declaramos el evento para cargar los datos de la página de inicio.

Fijaos en que unimos el evento pageinit al contenedor central de la página declarada en el código HTML con `<div id="index" data-role="page">`.

```
$('#index').bind('pageinit', function(event) {
loadRestaurants();
});
```

Esta función recupera la lista de restaurantes de una web mediante datos en formato JSON y muestra en la página index.html la lista de restaurantes. Podéis ver el contenido de la lista de restaurantes en formato JSON en la URL:

http://comoras.uoc.edu/~abartroli/apps/pizza/src/restaurants.php

```
function loadRestaurants(){
$.getJSON("http://comoras.uoc.edu/~abartroli/apps/pizza/src/restaurants.php")
  .done(function (data) {
    var i, rest;
    $.each(data.restaurants, function (i, rest) {
          $("#restaurant-list").append("<li><a href='restaurant.html? id_rest="+rest.id+"&aa=4'>"+
                  "<h3>" + rest.name + "</h3>" +
                  "<div> <img src='"+rest.img+"' width=120 height=120 /></div>" +
                  "</a>"+
```

```
                    "<p> Tlf:"+rest.phone+"</p> <p> Av. Diagonal 621</p></li>");
            });
    });
$('#restaurant-list').listview('refresh'); // refrescamos la lista de la página index.html
}
```

Página detalle restaurante

El header de esta página es idéntico al de la página index, en cambio el contenido del body es diferente y tiene un marcado HTML más definido.

```
<body>
     <div id="restoDetail" data-role="page" data-add-back-btn="true">
          <div data-role="header">
                <h1>Restaurant details</h1>
                <button id="saveBtn" data-icon="star" class="ui-btn- right">Favoritos</button>
          </div>
         <div data-role="content">
               <div id="restoDetails">
                      <h3 id="restoName"></h3>
                     <img id="restoImg" src="" width="100%"/>
                     <h5>Telephon</h5>
                     <p id="telephon"></p>
                     <h5>Description</h5>
                     <p id="description"></p>
                     <p id="forks"></p>
               </div>
               <div id="map_canvas" style="width:device-width; height:150px; "></div>
         </div>
</div>
<script type="text/javascript" src="js/index.js"></script>
</body>
```

Código JavaScript para la página restaurante.html

El código JavaScript para esta página es un poco más elaborado porque en ella se muestra el mapa de Google.

Las funciones en este fichero son:

- Captura del evento para mostrar el código cuando se carga la página.
- Función para recuperar los parámetros de la página con el identificador del restaurante.
- Función para recuperar los datos del restaurante.

- Función para definir el evento de mostrar el mapa de Google cuando el dispositivo esté preparado.

- Funciones para mostrar el mapa de Google.

Declaramos el evento para cargar los datos de la página de detalle del restaurante:

```
$(document).delegate("#restoDetail", "pageshow", function() {
        var id = getUrlVars().id_rest;
        loadRestoDetail(id);
});
```

Esta función recupera los datos de un restaurante en concreto mediante una llamada JSON:

```
function loadRestoDetail( id ){
     $.getJSON("http://comoras.uoc.edu/~abartroli/apps/pizza/src/resto_detail.php?
       id_rest="+id).done(function(data) {
     var rest = data;
     $('#restoName').html("<p>" + rest.name + "</p>");
     $('#restoImg').attr('src', rest.img );
     $('#telephon').text(rest.phone);
     $('#description').text(rest.descr);
```

La llamada típica que veréis en tutoriales es document.addEventListener("deviceready", onDeviceReady, false); pero si queréis pasar argumentos a la función unida a un addEventListener, la forma para hacerlo es sustituyendo el nombre de la función por:

```
document.addEventListener("deviceready", function() { onDeviceReady( lat, lon );}, false);
```

El evento deviceready se utiliza para dispositivos móviles.

Si queremos hacer funcionar el dispositivo en un navegador web, tendremos que descomentar la línea siguiente, por la llamada a la función «onDeviceReady»:

```
    onDeviceReady( rest.lat, rest.lon );
     });
}
```

Función para recuperar los parámetros de la URL:

```
function getUrlVars() {
    var vars = [], hash;
    var hashes = window.location.href.slice(window.location.href.indexOf('?') + 1).split('&');
```

```
    for(var i = 0; i < hashes.length; i++)
    {
        hash = hashes[i].split('=');
        vars.push(hash[0]);
        vars[hash[0]] = hash[1];
    }
    return vars;
}
```

Funciones para Google Maps:

```
function GoogleMap(){
```

Inicializamos el mapa de Google:

```
     this.initialize = function( lat, lon ){
     var map = showMap( lat, lon  );
}
```

Definimos los parámetros del mapa y los mostramos:

```
var showMap = function( lat, lon ){
     var mapOptions = {
          zoom: 16,
          center: new google.maps.LatLng(lat, lon),
          mapTypeId: google.maps.MapTypeId.ROADMAP
     }
     var map = new google.maps.Map(document.getElementById("map_canvas"), mapOptions);
     return map;
     }
}
function onDeviceReady( lat, lon ){
     var map = new GoogleMap();
     map.initialize( lat, lon );
}
```