CREANDO APPS

aprende a programar aplicaciones móviles

PARTE 6

PERSISTENCIA Y RED

Una función imprescindible en todas nuestras aplicaciones es la de poder grabar información, ya sea la puntuación de un juego, fotos o documentos que creamos, pero ya no basta con tenerlos disponibles en nuestro dispositivo. Con el auge de internet en el móvil ahora tenemos la posibilidad de guardar nuestros datos en la nube y tenerlos disponibles cuando queramos.

6.1 Persistencia y red / MOCKUP

Partimos de una aplicación vacía y creamos un prototipo del diseño de la aplicación. Este diseño no tiene ningún comportamiento. El html tiene la definición de los componentes visuales que queremos y en el css les damos un poco de color y separación.

Mockup prototipo de la aplicación

JSON Y COMPORTAMIENTO

Ahora le daremos un poco de vida al proyecto. Como siempre, añadimos la librería fastclick y haremos que algunos `divs` respondan a eventos.

También empezaremos a guardar datos aunque de momento no los persistiremos. Para ello usaremos un objeto JSON[1], llamado `model`, en el que guardaremos nuestros datos añadiendo nuevos objetos al array `notas` usando el método `push`.

```
construirNota: function() {
  var notas = app.model.notas;
  notas.push({
    "titulo": app.extraerTitulo() ,
    "contenido": app.extraerComentario()
  });
},
```

La extracción de los datos de nuestro interfaz de usuario la hacemos accediendo al DOM.

```
extraerTitulo: function() {
  return document.getElementById('titulo').value;
},

extraerComentario: function() {
  return document.getElementById('comentario').value;
},
```

(1) Descripción JSON
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/JSON

6.2 Persistencia y red / JSON Y COMPORTAMIENTO

Persistencia capacidad de que el estado de la aplicación pueda ser recuperado tras salir de la misma

Storage (Almacenamiento) - se refiere al lugar físico de persistencia, memoria del teléfono o servicio en la nube

Nube (Cloud Computing) - servicios ofertados por terceros que se pueden usar sin que el usuario tenga que conocer aspectos avanzados de cómo gestionarlos

DOM interfaz que permite acceder y manipular los componentes de una página HTML

6.3 Persistencia y red /
ALMACENAMIENTO LOCAL

Hora de empezar a persistir la información en el teléfono. Vamos a usar el plugin `cordova-plugin-file`[2]. Para obtener los `readers` y `writers` de la aplicación debemos encadenar llamadas en la fuciones de éxito de `resolveLocalFileSystemURL`, `getFile` y `createWriter` o `createReader`.

Todas ellas siempre tiene la misma estructura una función que hace algo, y un callback de éxito y otro de error.

La primera de ellas se encarga de pedir al dispositivo si se puede puede acceder a la Sistema de Ficheros Local, en el directorio que le solicitamos en el parámetro.

Si se puede acceder llamamos a la función de exito y si no, se llama a la función de error.

(2) File plugin
`http://cordova.apache.org/docs/en/latest/reference/cordova-plugin-file/index.html`

```
grabarDatos: function() {
  window.resolveLocalFileSystemURL(
    cordova.file.externalApplicationStorageDirectory,
    this.gotFS, this.fail
  );
},
```

Cuando las tenemos, en el writer solo tenemos que pasar nuestro objeto JSON a texto (`stringify`). En esta ocasión lo que tenemos que hacer es registrar en el *writer* que hacer cuando se termina de escribir con éxito. La estrategia que hay que hacer para lograrlo no es mediante callbacks como en las anteriores funciones, sino registrando funciones en los métodos que se ejecutan al ocurrir eventos. Esta vez registramos una función que escribe un mensaje en la consola en el evento de `onwriteend`.

6.3 Persistencia y red / ALMACENAMIENTO LOCAL

```
gotFileWriter: function(writer) {
  writer.onwriteend = function(evt) {
    console.log("datos grabados en
externalApplicationStorageDirectory");
  };
  writer.write(JSON.stringify(app.model));
},
```

Para leerlas, la estrategia es exactamente igual que en el paso anterior. En el reader tenemos que leer el fichero y registrar una función en el evento onloadend que hace que, al finalizar de cargar el fichero, pasar el texto a un objeto JSON (`parse`)

```
leerFile: function(file) {
  var reader = new FileReader();
  reader.onloadend = function(evt) {
    var data = evt.target.result;
    app.model = JSON.parse(data);
    app.inicio();
  };
  reader.readAsText(file);
},
```

JSON (JavaScript Object Notation)	formato de texto para intercambio de datos
Array	agrupación de datos del mismo tipo que puede ser iterado (recorrido)

ALMACENAMIENTO REMOTO Y WIFI

Finalmente vamos a guardar nuestra información en la nube. Añadimos la librería de firebase para el almacenamiento remoto y el plugin cordova-plugin-network-information[3] para saber si tenemos wifi disponible.

Para persistir el fichero en remoto, añadimos la configuración de firebase[4] al iniciar la app.

```
firebaseConfig: {
  apiKey: “XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”,
  authDomain: “mooc-notes.firebaseapp.com”,
  databaseURL: “https://mooc-notes.firebaseio.com”,
  storageBucket: “mooc-notes.appspot.com”,
  messagingSenderId: “XXXXXXXXXXXXXXXXX”
},
iniciaFirebase: function() {
  firebase.initializeApp(this.firebaseConfig);
},
```

(3) Network Information plugin
https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-network-information

(4) Firebase
Descripción - https://firebase.google.com/docs/storage/
Setup - https://firebase.google.com/docs/storage/web/start
Ejemplo - https://firebase.google.com/docs/storage/web/create-reference

6.4 Persistencia y red / ALMACENAMIENTO REMOTO Y WIFI

Posteriormente solo tenemos que añadir las funciones de tipo de conexión(`navigator.connection.type`)

```
hayWifi: function() {
  return navigator.connection.type===’wifi’;
},
```

y guardar en el storage de firebase(`firebase.storage().ref()` y `putString`), cuando terminamos de grabar los datos en local.

```
salvarFirebase: function() {
  var ref = firebase.storage().ref(‘model.json’);
  ref.putString(JSON.stringify(app.model));
},
```

¡Con esto ya tenemos la aplicación lista!

Offline sin conexión a internet