Alumno: Fran Adrvoer Vallejo

Centro: IES Francesc de Borja Moll

Modulo: DIW

Docente: Alberto Soto

IonicAp

Índice de contenido

Presentación y propósito

En nuestro día a día el mundo de los gadchets y su contenido están cada vez mas en auge, redes sociales, apps informativas referentes a cualquier temática, contenidos multimedia o servicios en la nube son solo unos ejemplos del abanico que nos ofrece este mercado.

En el presente proyecto hemos decido desarrollar una aplicación que nos permita inmortalizar cualquier momento en cualquier lugar, pudiendo posteriormente compartir nuestra experiencia con el resto del mundo.

Se trata de una aplicación móvil que nos permita realizar fotografías para poder posteriormente subirlas a nuestro servidor, además nos ofrecerá la posibilidad de elegir entre opciones de visualización, de edición, de creación o de supresión tanto de categorías como de todos los items creados y subidos por el resto de usuarios.

Requerimientos

Nuestra aplicación cuenta con la característica de poder realizar fotografías, mediante un botón que encontraremos en una de las opciones de menú, cuyo cometido será acceder a la cámara del dispositivo en cuestión, ofrecer la opción al usuario de poder realizar fotografías y posteriormente subirlas al servidor, habiendo etiquetado la imagen en cuestión previamente.

Situado en la misma opción de menú el usuario dispondrá también de un botón que le permitirá la posibilidad de poder subir fotografías almacenadas en le carrete del dispositivo, pudiendo al igual que en la opción anterior, etiquetar dicha imagen.

Además la aplicación constará de diversas opciones de menú adicionales, las cuales brindarán al usuario la posibilidad de poder filtrar por categorías o por items, pudiendo de esta manera visualizar contenidos según el criterio de búsqueda, otra opción de edición en donde podremos, crear, editar y eliminar, tanto categorías como items, y nombrar también una opción de menú que nos llevará al home de la aplicación.

Caso de Uso

El usuario deberá ejecutar la aplicación, logearse, si previamente no lo ha hecho, y a continuación la aplicación nos llevara al home. Una vez dentro, las opciones a realizar son varias, el usuario dispone de un menú lateral que le ofrecerá diversas tareas;

1. HOME: Dicho botón nos devuelve a la página principal de nuestra app.
2. CATEGORIES: Nos muestra todas las categorías almacenadas en el servidor, pudiendo de esta manera filtrar nuestra selección en función de categoría.
3. ITEMS: Del mismo modo que la opción anterior, se nos mostrarán todos los items almacenados en el servidor, sin contemplar categoría en cuestión.
4. EDIT: Esta opción nos permite tanto la edición, la creación o la supresión tanto de categorías como items.
5. UPLOAD: Dispondremos de dos botones, uno, cuyo cometido es activar la cámara del dispositivo, realizar la fotografía deseada, etiquetarla y subirla al servidor, y otro botón, el cual nos brinda la posibilidad de elegir una imagen desde la biblioteca del dispositivo, etiquetar y subir posteriormente.

Entorno de desarrollo

Ionic es un framework desarrollado bajo tecnologías Javascript que permite desarrollar aplicaciones celulares híbridas. A grandes rasgos, cuando hablamos de aplicaciones celulares híbridas nos estamos refiriendo a que son aplicaciones que no son del tipo nativas, es decir, que no son desarrolladas con la tecnología nativa con la que ejecuta en el sistema operativo del móvil.

Particularmente este framework posee grandes características que lo diferencian de muchos de otros frameworks similares. Entre estas ventajas enumeramos las siguientes:

- Posee un numero importante de componentes propios de aplicaciones móviles nativas desarrolladas en **HTML5**.

Esto nos permite no solo obtener una interfaz que emula casi a la perfección una aplicación nativa sino que también, nos permite ver la aplicación desde un navegador web y los mismos componentes no perderán calidad ni se deformarán sin importar que resolución tengamos. Punto importante en la portabilidad.

- Implementa un patrón MVC para dividir las capas de la aplicación haciendo uso de las buenas prácticas.

El patrón de diseño MVC utilizado en el framework es aportado por otro framework muy importante, que es **AngularJS** desarrollado por la gente de Google, aplicando la buena práctica de modularización de las vistas sin necesidad de embeber el template en el mismo index que es como lo van a ver en muchos de los ejemplos existentes en internet y que no son del todo prolijos ya que no utilizan toda la fuerza que brinda **AngularJS**.

- Utiliza **Apache Cordova** como wrapper o capa de abstracción del OS del celular nativo en el cual se quiere desarrollar.

Apache Cordova es una plataforma que se utiliza para la construcción de aplicaciones nativas de celulares o dispositivos móviles en general utilizando HTML, CSS y Javascript.

- Permite customizar el front-end de forma fácil, rápida y muy sencilla al permitir el diseño de la aplicación con CSS y HTML.
- Trabaja en conjunto con la tecnología conocida como Bower que funciona como repositorio de modulos javascript.

Este framework de dependencias javascript nos permite instalar módulos o plugins más facilmente para ser utilizados por **AngularJS**

Instalación:

Para la instalación de este framework basta solamente con tener instalado **NodeJS** y el **NPM** *(Node Packaged Modules).*

sudo apt-get install nodejs nodejs-legacy npm

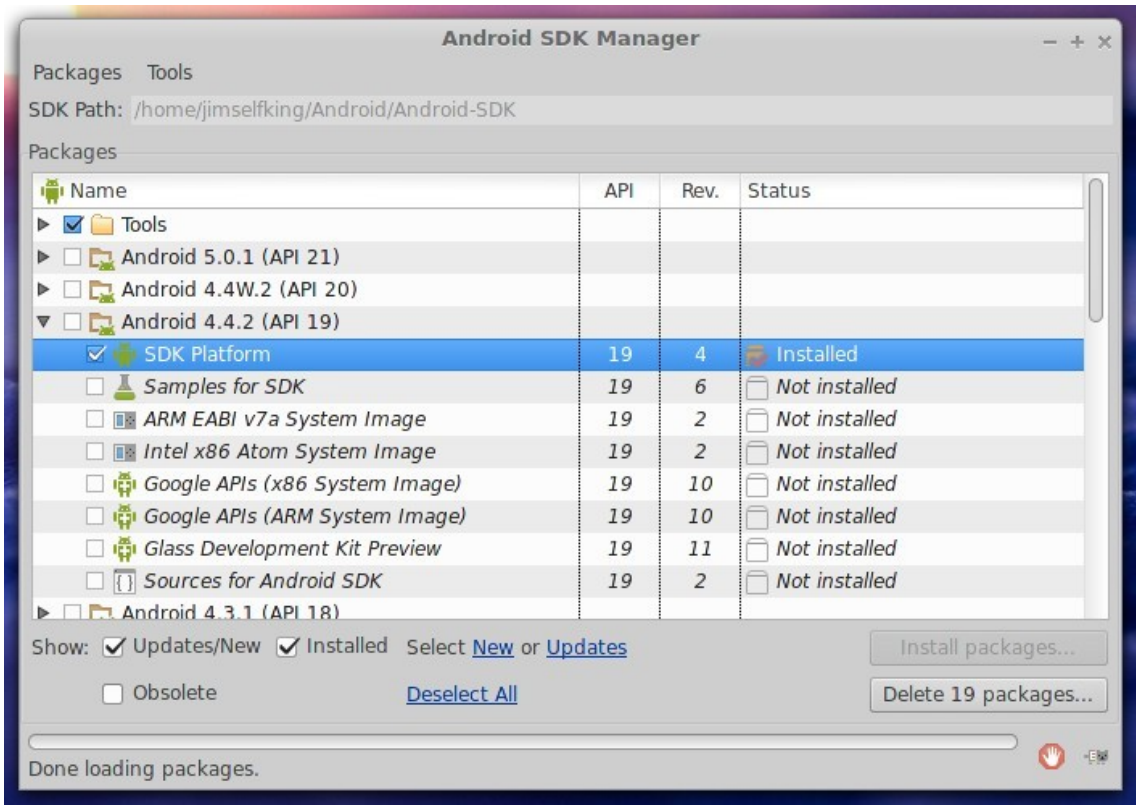
Luego de ejecutada la misma, ya tenemos instalado tanto NodeJS como su manejador de paquetes necesario para la instalación del framework ionic.

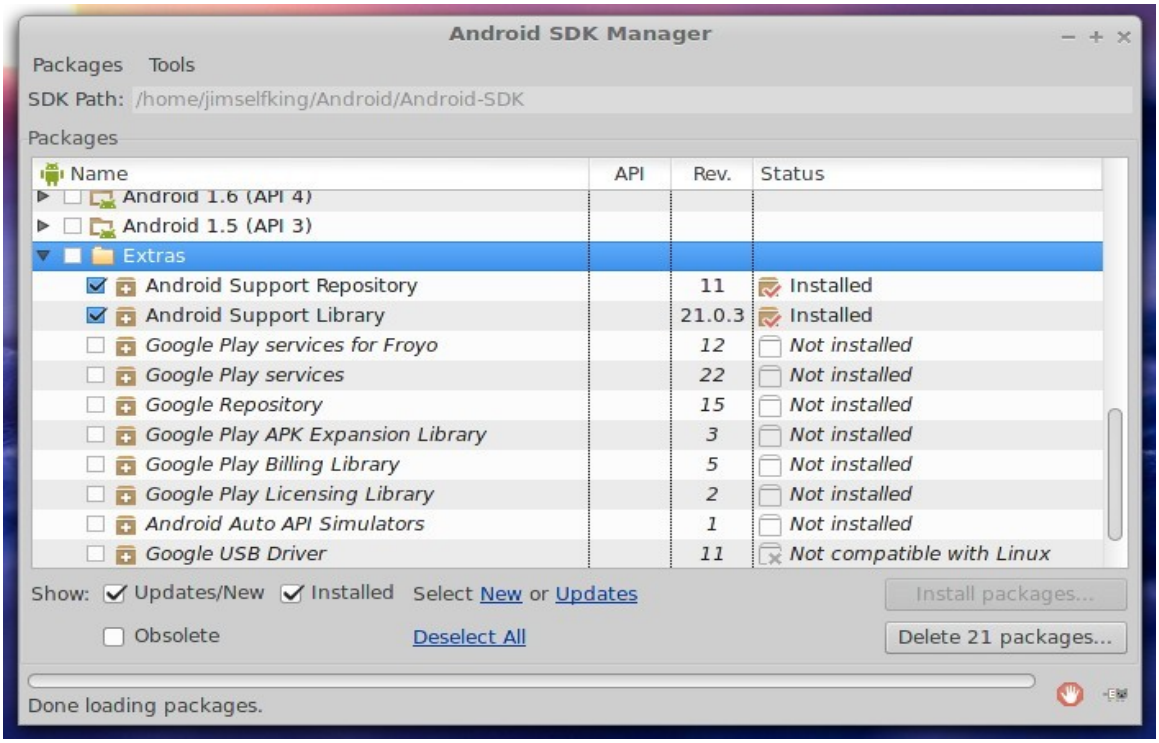
También deberemos instalar Yeoman, Bower y Grunt usando npm:

npm install -g yo bower grunt-cli gulp

Debemos descargar también la ultima versión del SDK (Software Development Kit) de Android . Lo descomprimiremos y emplazaremos en la carpeta Documentos de nuestro Home (el emplazamiento es opcional). Dentro de la carpeta que acabamos de descomprimir deberemos acceder a Tools y ejecutar en un terminal el archivo “Android”.
$ cd /home/$USER/Documentos/android-sdk_versionDescargada/tools/ && ./Android

La ventana que nos abrirá el comando anterior es el administrador del SDK de Android. En él deberemos descargarnos los paquetes marcados en las siguientes imágenes


Android SDK Manager
Packages Tools
SDK Path: /home/jimselfking/Android/Android-SDK
Packages
Name API Rev. Status
Tools
Android 5.0.1 (API 21)
Android 4.4W.2 (API 20)
Android 4.4.2 (API 19)
SDK Platform 19 4 Installed
Samples for SDK 19 6 Not installed
ARM EABI v7a System Image 19 2 Not installed
Intel x86 Atom System Image 19 2 Not installed
Google APIs (x86 System Image) 19 10 Not installed
Google APIs (ARM System Image) 19 10 Not installed
Glass Development Kit Preview 19 11 Not installed
Sources for Android SDK 19 2 Not installed
Android 4.3.1 (API 18)
Show: Updates/New Installed Select New or Updates
Install packages...
Obsolete
Deselect All
Delete 19 packages...
Done loading packages.


Android SDK Manager
Packages Tools
SDK Path: /home/jimselfking/Android/Android-SDK
Packages
Name API Rev. Status
Android 1.6 (API 4)
Android 1.5 (API 3)
Extras
Android Support Repository 11 Installed
Android Support Library 21.0.3 Installed
Google Play services for Froyo 12 Not installed
Google Play services 22 Not installed
Google Repository 15 Not installed
Google Play APK Expansion Library 3 Not installed
Google Play Billing Library 5 Not installed
Google Play Licensing Library 2 Not installed
Android Auto API Simulators 1 Not installed
Google USB Driver 11 Not compatible with Linux
Show: Updates/New Installed Select New or Updates
Install packages...
Obsolete
Deselect All
Delete 21 packages...
Done loading packages.

A continuación instalaremos la ultima versión del JDK que tengamos en los repositorios. El siguiente paso es configurar nuestro bashrc para que cargue al inicio el path al SDK de linux. Para ello introducimos en una terminal el siguiente comando:

gedit /home/$USER/.bashrc

Introduciremos al final de todo el documento las siguientes lineas:

export HOME="/home/$USER"
export ANDROID_HOME="$HOME/Documentos/android-sdk-linux"
export PATH="$HOME/Documentos/android-sdk-linux/tools:
$ANDROID_HOME/plataform_tools:$PATH”

Estas lineas permitirán a Ionic comunicarse con el sdk de android. Reiniciamos el sistema y cuando accedamos de nuevo, en una terminal escribiremos android, si nos abre el administrador del SDK habremos realizado correctamente este paso.

El siguiente paso sera instalar cordova, ionic y gulp. Para ello introducimos en la Terminal:

Sudo npm install -g cordova && sudo npm install -g gulp ionic

Para poder compilar el proyecto en un apk (paquete de instalación de android) deberemos instalar ademas apache ant.

Inicio de un proyecto nuevo

Desde terminal accedemos a la carpeta donde queramos que se cree el proyecto y tecleamos el siguiente comando

Ionic start NombreDelProyecto blank

Existen dos plantillas mas de inicio, para usarlas deberemos sustituir en el comando anterior “blank” por “tabs” o “sidemenu”.

Añadir una plataforma nueva a nuestro proyecto

Desde terminal, y situados en la carpeta donde tengamos el proyecto, escribimos el siguiente comando:

Ionic platform add android

Para poder trabajar sobre plataformas IOS, necesitaremos contar con un ordenador de marca Apple.

Ver los resultados de nuestros avances en el navegador

En la terminal, accedemos a la carpeta de nuestro proyecto e introducimos el siguiente comando:

Ionic serve --lab

Compilando el proyecto

Para compilar el proyecto en un apk instalable en un terminal android, abriremos un terminal, nos situaremos en la carpeta de nuestro proyecto y teclearemos lo siguiente:

Ionic build android

Añadir un SplashScreen y un icono a nuestra aplicación

Para añadir un splashScreen y un icono deberemos instalar en primer lugar el plugin de cordova para Splash:

cordova plugin add org.apache.cordova.splashscreen.

A continuación escribir en el terminal el comando

ionic resources.

El siguiente paso será emplazar una imagen en formato png o psd en la carpeta resources con el nombre splash.png, y la otra con nombre icon.png. Solo queda repetir el comando anterior para que cree los diferentes tamaños adaptados a diversos tamaños de pantalla.

Enfoque técnico y descripción

El proyecto está desarrollado en lenguaje HTML, que actuará a modo de GUI para el usuario final. Existe una página inicial llamada index.html que es donde cargamos todos los archivos JavaScript encargados de dar funcionalidad a la aplicación. En el flujo de datos entre nuestra aplicación y el servidor, REST, en este caso, corre a cargo de objetos JSON, que son los que nos permiten ingresar y recuperar los datos que serán mostrados al usuario como resultado final.

Estructura del proyecto

Carpetas propias de Cordova:

hooks: Almacena los scripts para realizar tareas personalizadas como la carga automática de plugins.

Plugins: Almacena el código fuente de los plugins que vayamos a utilizar.

www: Almacena el código fuente que va a ser ejecutado dentro de cada proyecto híbrido. En el caso de este generador, es aquí donde va a almacenar la salida de las tareas que ejecute.

Resources: Carpeta destinada a parámetros de Splash Screen

config.xml: Es el fichero que almacena información de relevancia para poder configurar los proyectos dependiendo de la plataforma.

Carpetas propias de la aplicación web:

css: Almacena todos los archivos correspondientes a los estilos

img: Almacena las imágenes del proyecto

js: Almacena los archivos javascript

> **app.js:** Archivo de suma importancia dado que es el primer archivo a ejecutar tras index.html. Cuando inicia la aplicación este archivo hace de motor de arranque, contiene configurada la vista o state que se mostrará por defecto al ser iniciada la app, asi como todos los modulos de angular empleados por la app. Consta de todos los states de los que la aplicación hace uso, dichos states están asociados a diversos parámetros configurados de manera eficaz para que el flujo de la aplicación sea el correcto. Parámetros como una url asociada, un template que será el archivo html a mostrar, un controlador en caso de que la vista configurada lo requiera, asi como varias opciones que no hemos necesitado para nuestra aplicación. Un state puede ser configurado como abstracto, lo que significa que el template asociado a este ejercerá como template principal de nuestra aplicación, el resto de states (dependientes del abstracto) contendrán un parámetro el cual hará referencia a la vista, y que serán mostrados en función de donde se produzca la

llamada a estos, en nuestro caso la pagina que contiene el menú, que consta de una directiva propia de Ionic en donde serán cargados nuestros templates.

Controllers.js: Es la capa que sirve de enlace entre la vista y el modelo. Envía comandos al modelo para actualizar su estado, y a la vista correspondiente para cambiar su presentación. Los controladores contienen todas las funciones que son ejecutadas en la aplicación. En los controladores en donde se desata toda la actividad de la app, y son los que hacen posible que la vista tenga un comportamiento dinámico. Destaca la presencia del scope que es el parámetro que en comunicación con los templates nos brindará el flujo de la aplicación, también suelen trabajar con los posibles servicios que tengamos en uso en nuestra app, haciendo posible la interacción de datos con el servidor. A continuación un detalle de los controladores que contiene el archivo.

Controlador: AppCtrl

Función: Este controlador es que hace posible la aparición y posterior desaparición del modal de login, también contiene una función que será la que nos permita navegar hasta la página de inicio tras pulsar el botón al logearnos

Controlador: HomeCtrl

Función: Este controlador contiene dos funciones que entran en juego al pulsar en dos iconos alojados en el footer de la app, uno nos dirigirá al home, y el otro a la página de login.

Controlador: CategoryListCtrl

Función*:* Realiza la consulta de las categorías al servidor REST que serán mostradas en la opción de menú dedicada a visualizar categorías. Contiene también dos funciones, una es ejecutada al pulsar un botón para añadir categorías, lo que hace esta función es mediante el parámetro state.go llevarnos al state configurado en cuestión. La otra función hace referencia al botón dedicado a

eliminar categorías, que al pulsarlo le pasará como parámetro la categoría deseada por el usuario. En dicha función se realizará una consulta para extraer el numero de items que contiene la categoría a eliminar, si dicha categoría contiene items en su interior se mostrará un ventana emergente gracias a la directiva de Ionic $ionicPopup, avisando al usuario que previamente debe realizar la eliminación de items para poder realizar la operación,

Controlador: ItemDetailCtrl

Función: Nos mostrará la lista de items en función de la categoría en la que hayamos entrado, contiene también dos funciones, una que nos llevará al state configurado al pulsar un botón dedicado a añadir nuevos items, y otra cuyo cometido es el de eliminar items de la misma forma que el controlador anterior eliminaba categorías.

Controlador: ImageDetailCtrl

Función: Al igual que el controlador anterior, este nos mostrara contenido en forma de imágenes en función del item seleccionado. También contiene una función encargada de mostrarnos las imágenes a pantalla completa, en dicha función se cargará el template dedicado a ello gracias a la directiva de Ionic, $ionicModal.fromTemplateUrl. Hay también tres funciones mas para el manejo del fullScreen, dos referentes a los botones de avance o retroceso y una mas para cerrar el modal, y una ultima función que sera la encargada de eliminar imágenes una vez pulsado el botón referente a la eliminación de imágenes en la vista.

Controlador: SelectTabCtrl

Función: Este controlador esta compuesto por cuatro funciones, cuya tarea es la redirigirnos a un state o a otro, en función de si el usuario desea editar categorías, items, realizar una captura desde la cámara, o elegir una imagen desde la librería para su posterior subida al servidor.

Controlador: AddCategoryCtrl

Función*:* Dicho controlador nos permitirá a través de la función addCategory añadir nuevas categorías, controlará que los campos a rellenar cumplan con los criterios fijados y si todo es correcto, nos mostrará un popup emergente con el aviso de que la categoría se ha ingresado con éxito. Después de esto, nos redirigirá de nuevo al state inicial de la presente opción de menú.

Controlador*:* UpdateCategoryCtrl

Función: La tarea de este controlador es la de mediante el parámetro $stateParams y el $scope, cargar atributos como, nombre, descripción etc.., (de la categoría a actualizar que habremos seleccionado previamente en la vista que nos carga dichos atributos), en la vista en forma de formulario. Una vez modificados los campos es de nuevo el $scope el encargado de devolver los atributos al controlador para que este pueda realizar la actualización de los nuevos datos en el servidor. Al hacer click en el botón encargado de ejecutar la función de actualización también ejecutará un popup para una clara visualización del usuario.

Controlador: AddItemCtrl

Función: La tarea de este controlador es exactamente la misma que la del controlador encargado de añadir categorías, con el agravante de que este controlador dispone de una consulta al servidor para listar las categorías alojadas en su base de datos, puesto que al añadir un item debemos seleccionar la categoría deseada, opción que nos brindará la vista en forma de lista desplegable.

Controlador: UpdateItemCtrl

Función: Este controlador contiene la función updateItem, función que realiza la misma tarea que el controlador de actualización de categorías, recoge parámetros desde el state del item a actualizar por medio del $scope, recogiendo en este caso

a diferencia del controlador actualizador de categorías, el objeto JSON referente a la categoría en cuestión, y carga los nuevos campos introducidos en el formulario de envío para su posterior en subida al servidor.

Controlador: CameraCtrl

Función: Es el controlador que nos permitirá poder acceder a la cámara del dispositivo, contiene configuraciones como calidad de imagen, tipo de destino, tipo de origen, tipo de codificación o tamaño del marco que alojará la fotografía captada por la cámara.

Controlador: LoadFileCtrl

Función: Recoge la imagen cargada desde la librería del dispositivo mediante la etiqueta input y la directiva de Angular file-model en la vista. Si existe, es decir, si hemos cargado alguna imagen, carga los datos de la imagen en un array y previamente le asigna el parámetro $rootScope para poder luego acceder a dicho array desde un controlador externo.

Controlador: UploadCtrl

Función: Este controlador tiene varias funciones que hacen posible la subida de imágenes al servidor. La primera controlara que hayamos introducido parámetros descriptivos de la imagen como el nombre o descripción, en caso de que no cumpla con los criterios de subida la función nos dirigirá a otra para que sea mostrada una alerta en modo de popup avisando al usuario del error, en caso contrario nos dirigirá a la función encargada de subir el formulario con los con los campos rellenados correctamente, esta función nos mostrara también un popup con el mensaje de que la carga se ha realizado y ha continuación se produce una llamada a la función upload que se encargará de pasarle al servicio programado para la subida de la imagen, la imagen en si, la url donde alojarla y los atributos con sus respectivos valores.

config.js: Este archivo nos permite la conexión al la base de datos, contiene una constante con la configuración adecuada para atacar al servidor REST en cuestión, y poder de esta manera disponer de la información deseada. Esta compuesto también por servicios o factorías, que juntamente con la constante nos permitirán poder realizar diversas consultas al servidor, como una selección, para así visualizar contenido, hacer una actualización, una posible supresión de datos o incluso inserción de estos, es decir, es en dichas factorías donde se llevan a cabo la conversión de comandos propios del protocolo HTTP al lenguaje JavaScript.

lib: Contiene todas las librerías JavaScript necesarias para un correcto funcionamiento de la aplicación

ng-cordova-min.js: Archivo que hace que sea posible la interacción entre la aplicación y el hardware del dispositivo.

angular-resource.js: Necesario incluirlo en el proyecto para poder hacer uso del modulo de Angular, ngResource, el cual usaremos para interactuar con el api del servidor REST.

angular-ui-router.js: Este archivo es necesario si queremos hacer uso de una correcta navegación a través de nuestra aplicación mediante el state, deberemos incluir su módulo junto a los demás en la llamada a estos en el archivo de inicio

jquery-2.1.3.min.js: Archivo necesario en caso de uso de librerías Jquery.

templates: directorio donde tenemos almacenados todos los archivos html.

Index.html: Archivo de suma importancia. Es en él donde se cargarán y mostrarán todas las vistas de la app. El resto de templates son cargados en dicho fichero, concretamente en la directiva Angular llamada <ion-nav-view>. El archivo index.html hace de contenedor al resto de plantillas html, las cuales únicamente contienen el código explicito en función de su cometido en la aplicación. Hay otro detalle a destacar, su nombre es, ng-app, un atributo de la etiqueta <body>, el valor asignado a este atributo sera el nombre identificador de nuestra app, el cual es

pasado como parámetro a la llamada de módulos de Angular en el archivo de inicio app.js, asignando así todos los módulos al nombre identificador de la app. En este archivo, como suele ser normal, también es donde se realizan las cargas del resto de archivos necesarios para el flujo de la aplicación, estilos, librerías, archivos JavaScript etc...

homelogin.html: Contiene una imagen de fondo y un botón que es el encargado de mostrarnos el modal del login.

home.html: En este archivo podremos visualizar otra imagen de fondo y el icono de la aplicación junto al nombre de esta

menu.html: Consta de dos partes, la contenida entre las directivas <ionl-side-menu-content>, que se trata de el botón que nos deplegará el menú, y la parte donde se van a cargar todos los templates, es decir las diversas vistas que tengamos en la app y lo harán en el interior de la directiva de ionic <ion-nav-view> que en este caso depende de un atributo, llamado name, que cuyo valor es el mismo identificador que tienen asignadas las vistas en los states. La segunda parte esta contenida entre directivas llamadas <ion-side-menu> y mostrará todas las opciones de menú desplegado en forma de lista. Al pulsar la opción deseada del menú, seremos trasladados al state al que apunte dicha opción, gracias al parámetro ui-sref de Angular, el cual hace uso del state para la navegación y de la url para el paso de parámetros al controlador.

categories.html: Archivo encargado de mostrar todas las categorías almacenadas en el servidor. Contiene directivas esenciales para la posible visualización de la vista, como por ejemplo, <ion-view> o <ion-content>, hacen de pequeños contenedores de contenido. Las categoría se mostrarán en forma de lista, pero esta será creada dinámicamente, gracias a la directiva de Ionic <ion-item> y su atributo Angular ng-repeat, que nos mostrará todas las categorías alojadas en el REST, cada categoría nos hará de enlace para navegar hasta los items contenidos a la categoría actual. Volvemos a encontrarnos aquí con la orden ui-sref, a la que ahora le pasaremos parámetros que serán de uso para mostrar el template de items correctamente, para ello haremos uso del $stateProvider y del url para la recepción de parámetros en el state, los cuales serán pasados por medio del

$stateParams al controlador asignado al state en donde nos encontramos.

items.html: Se trata del mismo caso que el archivo anterior, hace uso de las mismas directivas de Ionic, y ordenes como ui-sref y ng-repeat o el paso de parámetros mediante $stateProvider o $StateParams. Mostrará la lista de items en modo thumbnail, con una imagen y la referencia de cada item.

images.html: Este archivo nos mostrará las imágenes contenidas dentro del item seleccionado. Lo hará de nuevo haciendo uso de la directiva ng-repeat contenida dentro de otra directiva Ionic llamada ion-scroll, que nos permitirá la visualización de las imágenes en forma de slider. Este archivo contiene también un botón dedicado a la supresión de imágenes que será cargado en cada imagen asi como también sera cargada para cada una de ellas la llamada a una función encargada de la visualización de las imágenes a pantalla completa por medio de nuevo de la directiva ng-click.

imagesFullscreen.html: Se trata de un modal a modo de pantalla completa que nos mostrara la imagen junto a dos botones para el avance o el retroceso a la siguiente imagen.

item_list.html: Muestra una lista de todos los items sin concretar ninguna categoría.

edit.html: Esta página únicamente contiene un div de clase tab con dos enlaces en forma de botón que nos redirigirán o a la edición de categorías o de items, según la intención del usuario. Dicha redirección es gracias al atributo de Angular ng-click, el cual recibe como parámetro un nombre equivalente a una función programada en el controlador asociado al state actual.

editCategoriesIndex.html: Lo primero que nos muestra esta página es un botón que nos llevará a la página encargada de crear categorías, todo ello mediante el atributo de Angular ng-click de la misma forma que en el ejemplo explicado anteriormente. Lo siguiente se trata de una lista bajo la directiva de Ionic <ion-list> y de nuevo Angular con ng-repeat para mostrarnos todas las categorías del REST, destacar que en cada linea de categoría encontraremos dos botones, uno para editar y el otro para eliminar dicha categoría.

addCategoy.html: Consta de un formulario para la recogida de datos al crear una nueva categoría, como un nombre y una descripción, estos datos son pasados al controlador gracias a la directiva de Angular ng-model al hacer click en el botón que llama a la función addCategory la cual transferirá el valor introducido por el usuario al controlador que será el encargado crear la nueva categoría.

updateCategory.html: Este archivo es igual que el anterior, contiene un formulario de envío que pasará parámetros al controlador encargado de la actualización de la categoría gracias al paso de estos mediante el botón al final del archivo que llama a la función con dicho cometido.

editItemsIndex.html: Archivo exacto al editCategoiresIndex.html. Contiene un botón que nos llevará a la página de crear nuevos items y a continuación una lista mostrando cada item alojado en le servidor con un botón de editado y supresión.

addItem.html: Vista en modo de formulario al cual le introduciremos los valores deseados al nuevo item referentes al nombre, categoría y un descripción. Estos valores son pasados al controlador por medio de la llamada a la función addItem al hacer click en el botón de envio.

updateItem.html: Este archivo opera bajo los mismos patrones y nos proporciona la misma vista que el de la actualización de categorías añadiendo únicamente un desplegable con todas las categorías.

upload.html: Contiene además del logo de la aplicación junto al nombre un div de clase tab que contiene dos botones, uno nos redirigirá a la subida de imágenes por medio del la cámara y el otro accediendo a la librería de imágenes del dispositivo.

camera.html: Este archivo esta asignado a un controlador llamado CameraCtrl gracias a la directiva de Angular ng-controller. Se trata de un archivo en modo de formulario de envío, un botón para el acceso a la cámara que al ser pulsado llama a la función takePicture por medio de la directiva ng-click, y un segundo botón que es el encargado de llamar a la función donde se iniciará todo el proceso de subida de imágenes.

l**ibrary.html**: De nuevo se trata de un formulario de envío en el cual introduciremos los campos deseados para la imagen a subir, en la parte inferior encontramos dos

botones situados en el mismo eje, uno es el encargado de abrir la biblioteca de imágenes del dispositivo y una vez elegida la imagen el otro hará la función de pasarla al controlador y cargarla en un array que será posteriormente tratado, dicho trato se llevará a cabo al pulsar un tercer botón, situado por debajo de los anteriores, que es el encargado de llamar a la función donde se inicia la validación de los campos y la subida al servidor.

Flujo y desarrollo de la aplicación

Al ejecutar la aplicación el usuario se encuentra frente a un botón que le ofrecerá el acceso a la misma, dicho botón al ser pulsado ejecuta una función llamada login(), el acceso a dicha función lo obtendremos mediante el controlador “AppCtrl” que es iniciado al cargar la página inicial establecida en el parámetro “$urlRouterProvider”, la pagina que contiene el botón, “homelogin.html”. La acción a realizar por esta función (login()), es la de mostrarnos el modal, el cual nos permitirá ingresar nuestras credenciales de logeo e ingresar al home de la aplicación activando la función destinada a ello al pulsar el botón de submit del modal, la función que nos hace dicho ingreso posible (closeLogin()), es caragada por el mismo controlador asociado al archivo “homelogin.html”.

Una vez en el home, dispondremos de un menú que contiene cinco opciones, una opción para volver al home, otra para poder visualizar y seleccionar las posibles categorías disponibles, otra la cual nos mostrará todos los items alojados en le servidor sin contemplar la categoría a la que pertenecen, una opción donde se nos permitirá la edición tanto de categorías como de items y una quinta opción de menú cuya funcionalidad es la de realización o selección de fotografías, según la elección del usuario, con una previa edición y su posterior envío al servidor. Cabe resaltar que el template del menú lo cargamos inicialmente como aplicación, es decir, es un template el cual esta presente en todo el flujo de la aplicación.

Si pulsamos la opción “Home” simplemente accedemos a nuestro home a través de la directiva de angular ui-sref, que hace uso del modulo $stateProvider para una óptima navegación.

Al seleccionar la opción “Categorias”, se nos direccionará al archivo “categories.html” el cual inicializa el controlador “CategoryListCtrl”, cuya función es la de cargarnos en el $scope todas las categorías contenidas en nuestra base de datos actualizando de esta manera la lista desplegable de categorías por medio de la directiva de Angular ng-repeat. Al seleccionar una categoría cualquiera lo que estamos haciendo a parte de navegar hasta el archivo responsable de mostrarnos los items de dicha categoría, “items.html”, es cargar el id de la categoría seleccionada, gracias al parámetro establecido en “url” del state de dicho archivo. Este state nos carga un controlador

llamado “ItemDetailCtrl”, cuya función es la de recoger todos los ids de los items alojados en el servidor y mediante el $stateParams para poder de esta manera filtrar los items de la categoría seleccionada previamente, el resultado final será una lista de items referentes a la categoría deseada. Situados en la lista de items, es decir el archivo “items.html”, el siguiente paso es hacer click sobre algún item para poder acceder el avatar entero junto a datos descriptivos al respecto, cuya acción nos llevará al archivo “images.html”, que es el encargado de mostrar dicho contenido. El state de este archivo contiene la carga de un id, el nombre y la descripción del item para poder ser mostrados junto a la imagen correspondiente, contiene también la llamada a un controlador llamado “ImageDetailCtrl”, la función de dicho controlador es la misma que el anterior, obtener los id de las imágenes almacenadas en el servidor, acción que es posible gracias a la configuración establecida en el archivo “config.js” y a la manera en la que interactuan entre ellos, y filtrar las imágenes correspondientes gracias de nuevo a el parámetro “$stateParams”, que es el que contiene el id obtenido al realizar la selección previa. Situado en la vista de imágenes estas serán mostradas en modo de slider, gracias a la directiva ion-scroll, que nos mostrara un slider es posición horizontal, además dispondremos de un botón que nos hará posible la visualización de las imágenes a pantalla completa en forma de modal de Ionic, el cual además de la imagen a gran escala contendrá dos botones, uno para para avanzar a la imagen siguiente y el otro hacia la anterior. Tendremos también un botón que nos permitirá la eliminación de dicha imagen, gracias a la llamada de una función llamada deleteImage() alojada en el controlador correspondiente a dicha vista.

La siguiente opción de menú nos ofrecerá una vista en modo de lista de todos los items almacenados en el servidor REST, lo hará sin tener en cuenta la categoría a la que pertenecen, una vez dentro de la vista las opciones son las mismas que las nombradas anteriormente, visualizar imágenes y si el usuario lo desea, eliminarla.

Una cuarta opción será la encargada de ofrecernos la oportunidad de editar tanto categorías como items, es decir, crear nuevas entradas, actualizar las existentes o poder eliminarlas. Al hacer click en dicha opción seremos trasladados a un vista, “edit.html”, la cual contiene dos botones en forma de enlace que nos proporcionarán la navegación en función de si queremos editar categorías o items. En el caso de que la elección recaiga sobre las categorías la navegación nos conducirá a la vista designada a ello, “editCategoriesIndex.html”, en la cual se nos mostrará un botón para el ingreso de nuevas

categorías y una lista de todas la categorías, junto a cada categoría aparecerán también dos botones mas, uno para una posible actualización el cual nos trasladará al archivo de actualización por medio de la directiva ui-sref y pasándole parámetros como el id, nombre o descripción, y otro para poder eliminarla, gracias a la llamada a la función dedicada a ello.

Y la quinta y última opción de menú nos llevara al template llamado “upload.html”. Dicho template es similar al nombrado anteriormente, “edit.html”, contiene dos botones direccionables, cuya función es llevada a cabo gracias a directivas como ng-click o $state.go. Uno de ellos nos llevará al archivo “camera.html” que es el encargado de abrir la cámara del dispositivo mediante la llamada a la función takePicture(), alojada en el controlador CameraCtrl, realizar una fotografía, editarla gracias a un formulario de envío situado en su interior y subirla al servidor, y con el segundo botón navegaremos hasta el template “library.html”, que de nuevo se trata de un formulario de envío que será el encargado gracias al $scope de volcar los atributos con sus respectivos valores en el controlador correspondiente, “UploadCtrl”. Este template tiene asignado también un segundo controlador mediante la directiva ng-controller de Angular llamado “LoadFileCtrl” que entra en acción después de haber seleccionado alguna imagen desde la librería, acción que se lleva a cabo con un botón adherido al botón encargado de llamar a la función loadImage() alojada en dicho controlador con la tarea de comprobar si hemos seleccionado alguna imagen desde la librería del dispositivo, si es así se cargarán los atributos de dicha imagen en un array que posteriormente será trasladado mediante el $rootScope al controlador encargado de comprobar dichos campos y realizar la subida al servidor, este controlador entra en juego tras la llamada de la función checkAtributes() contenida en su interior y cuya llamada se hace desde un tercer botón bajo la directiva ng-click.

Otras funcionalidades que tendrá el usuario disponibles son unos iconos situados en el footer de la pagina. Se trata de un icono para regresar al Home, otro de configuración y un tercero para salir a la página de logeo. Esto es posible debido a un controlador llamado “homeCtrl” que esta situado en el state de carga de inicio inicializándose de esta manera al ser cargada la raíz de la aplicación, dicho state nos cargara el menu y el footer para así permanecer en toda la aplicación. El contenido de dicho controlador simplemente son dos funciones que mediante el parámetro “state.go”

mas el state propio en la aplicación nos redirigirá al lugar establecido.