**JMFCameraIOS**

Práctica examen asignatura iOS avanzado

Master Programación Smartphones y Tables

[www.u-tad.com](http://www.u-tad.com)

Jorge Marcos Fernández

**Pantalla inicial**

Al arrancar la App aparece una splash screen con el titulo de la aplicación.



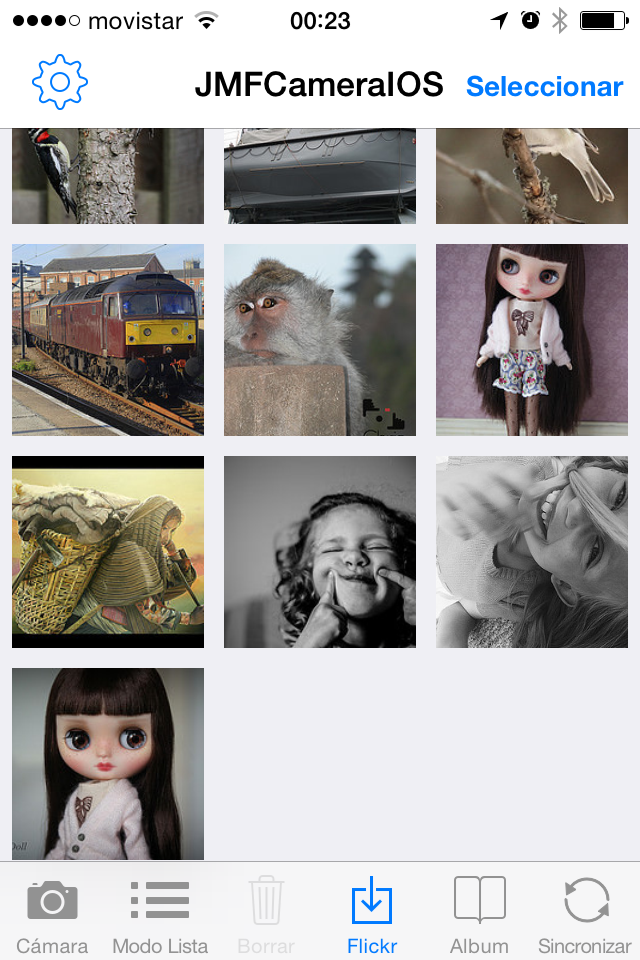
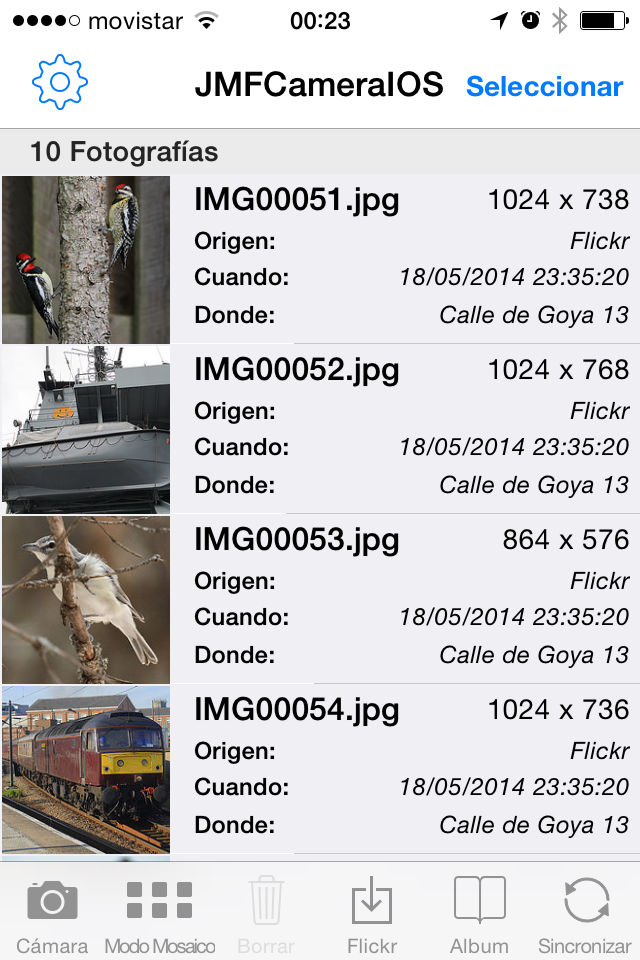
Cuando no existen fotografías en el álbum de la aplicación aparece la pantalla vacía que indica al usuario que añada fotografías al álbum.



Estas fotografías pueden adquirirse pulsando el botón  de la cámara donde podrá elegir entre tomar una nueva fotografía a seleccionarla de la galería del dispositivo.

También puede adquirirla desde Flickr pulsando el botón  donde podrá introducir un término de búsqueda y la aplicación intentará bajar de Flickr un máximo de 10 fotografías.

La pantalla inicial presenta las fotografías en dos modos posibles, Modo Mosaico y Modo Lista que se alternan pulsado los botones  y  .

En modo Mosaico las fotografías se muestran en una UICollectionView y en modo Lista se presentan en una UITableView con celdas personalizadas donde se indica información adicional sobre la fotografía.

Pulsando sobre el botón  el usuario puede visualizar las fotografías una tras otra dentro de UIPageControl.

El botón “**Seleccionar**” situado a la derecha en la Navigation Bar permite entrar en modo selección múltiple y el usuario podrá seleccionar mas de una fotografía. Pulsando el botón  podrá eliminarlas de una sola vez. Se deberá pulsar el botón “**Cancelar**” para salir del modo de selección múltiple.

El botón  permite sincronizar las fotografías de la aplicación con la cuenta de Flickr del usuario.

Previamente la sincronización debe activarse en la pantalla de preferencias (ver apartado “Preferencias”). Esta sincronización se realiza en segundo plano utilizando una background sesión de NSURLSession por lo que el usuario puede seguir utilizando la aplicación o bien pasarla a segundo plano y la sincronización seguirá realizándose .

En la imagen de la fotografía que este sincronizándose aparecerá una UIProgressView en el lado inferior con un color rojo con el progreso correspondiente. Una vez se haya sincronizado se marcará con un icono en la parte inferior izquierda de color verde.

Si el usuario está en selección múltiple no se permite la selección de la fotografía que se está subiendo a Flickr.

NOTA: por motivos de tiempo el borrado de las fotografías no se sincroniza con Flickr ya que Flickr ofrece un api diferente para el borrado que el de upload con autenticación diferente.

Al pulsar sobre el botón  el usuario podrá editar las preferencias de la aplicación.

**Pantalla de Preferencias**

****

La pantalla de preferencias ofrece al usuario la posibilidad de sincronizar o no las fotografías con su cuenta de Flickr, establecer el intervalo de subida entre una fotografía y otra y realizar un borrado de la base de datos.

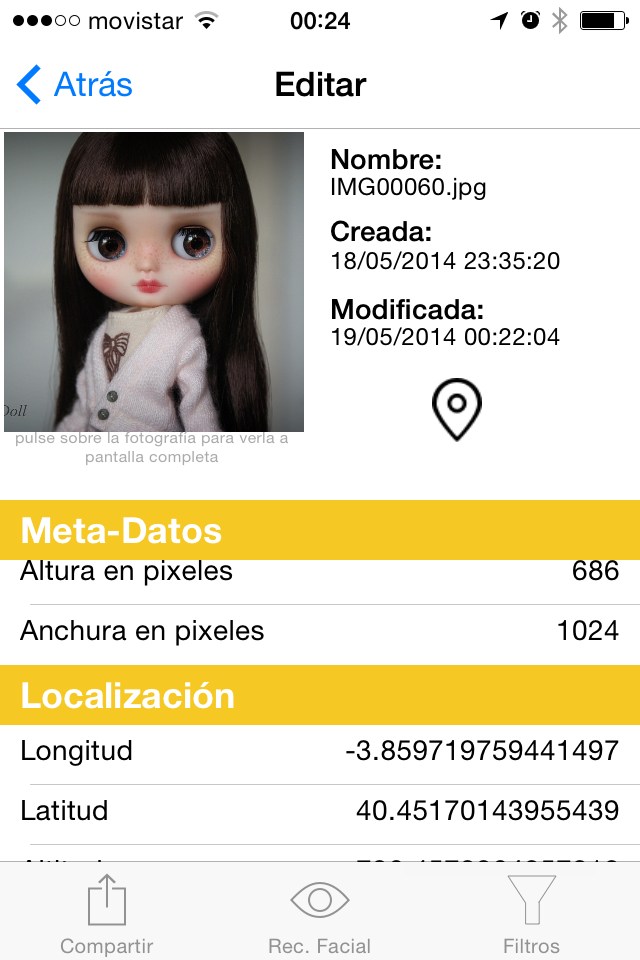
La primera vez que el usuario activa la sincronización con Flickr la aplicación realiza la autenticación OAuth del usuario y, si la autenticación es correcta, almacena las credenciales en las NSUserDefaults de la aplicación.

Pulsando el botón **OK** volverá a la pantalla inicial.

**Pantalla de Edición**

Cuando se pulsa sobre una fotografía en selección multiple se entra en la pantalla de Edición.

Esta pantalla ofrece un thumbnail de la fotografía y a la derecha los datos básicos: nombre y fechas de creación y actualización.



Al pulsar sobre la imagen se puede ver a pantalla completa y realizar zoom.

Al pulsar sobre el icono  la aplicación muestra una mapa con la ubicación del dispositivo en el momento de originarse la fotografía.

En la parte central se muestra una UITableView con diferentes secciones para los Meta-Datos, la Localización, los Rostros y los Filtros de la fotografía.

Además del los metados de la UIImage se incluyen los datos de origen y fecha de sincronización con Flickr, si existiera.

La sección de Localización ofrece los datos de CLLocation y la geo-codificación inversa que se presenta en una celda personalizada.

Los sección Rostros indican las coordenadas del rectángulo de los rostros detectados.

La sección Filtros muestra los filtros aplicados a la fotografía.

La secciones de Rostros y Filtros aparecen sin información cuando la fotografía no se ha editado y ambas permiten entrar en la opción de borrado realizando un gesto swipe hacia la izquierda sobre la celda.

La parte inferior muestra una UITabBar con tres botones que permiten Compartir, Reconocimiento Facial y selección de Filtros.

Al pulsar el botón  el usuario puede compartir la fotografía, dependiendo de las aplicaciones que tenga instaladas, con todos los tipos de actividad que admite el UIActivityViewController excepto Weibo, Assign to Contact, Add to Reading List y Vimeo.

Al pulsar el botón  entrará en la pantalla de reconocimiento facial y al pulsar sobre el botónel usuario podrá aplicar diferentes filtros a la fotografía.

**Pantalla de Reconocimiento Facial**

La pantalla de reconocimiento facial ofrece al usuario la posibilidad de detectar todos los rostros de una fotografía.



Para reconocer los rostros deberá pulsarse el botón . Todos los rostros detectados serán marcados en la fotografía por un cuadrado de color verde. Cada ojo, si existe será marcado por un circulo relleno de un color morado y la cada boca, si existe, por otro circulo de color verde claro.

Pulsando el botón  podrá eliminar todos los rostros detectados de la fotografía.

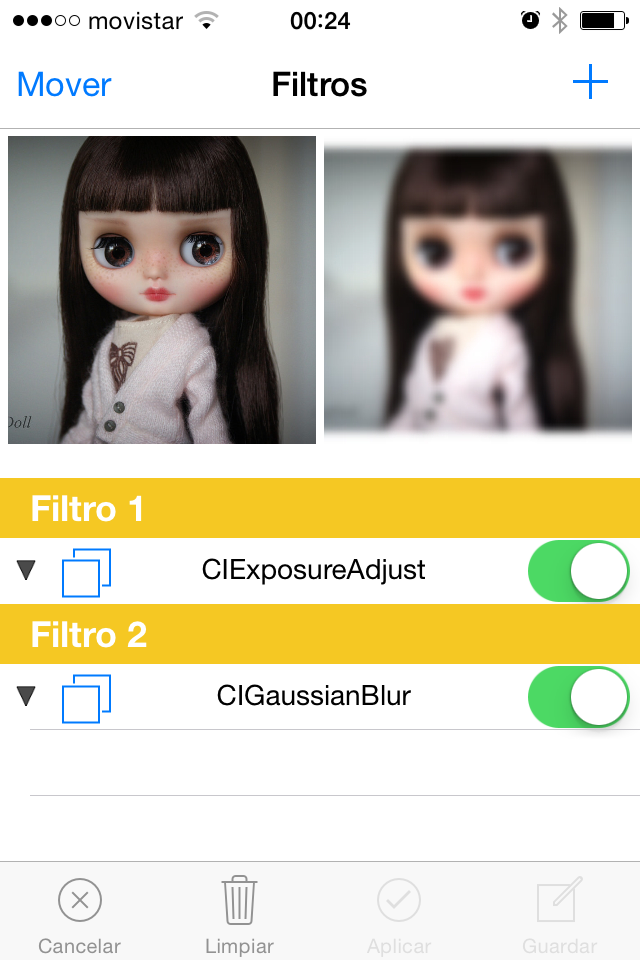
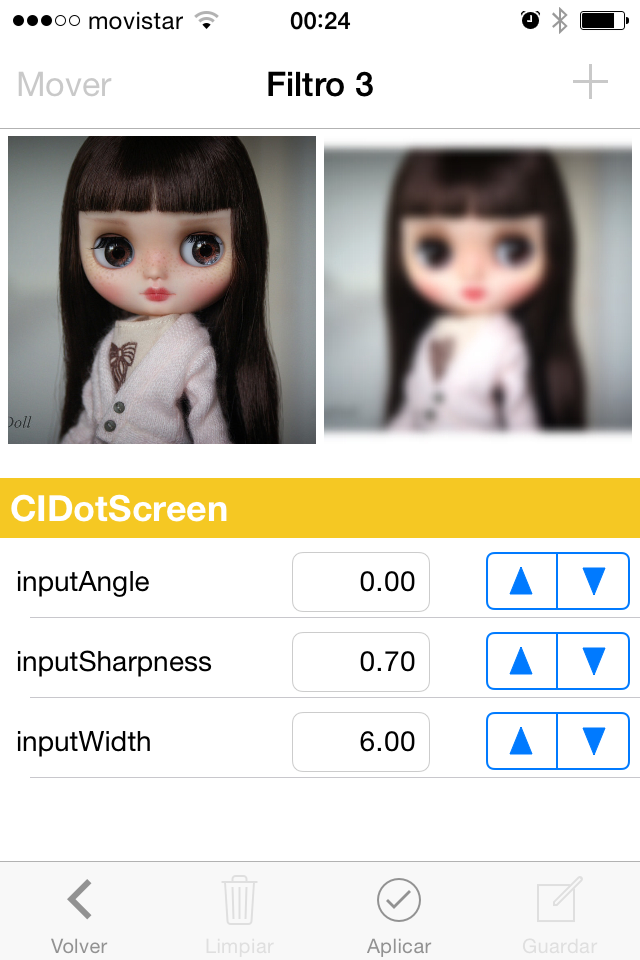
Esta opción está gobernada por NSUndoManager de CoreData por lo que para guardar los cambios deberá pulsarse el botón  o bien el botón  para desecharlos.

Una vez guardados, aparecerán en la sección Rostros de la pantalla de edición.

**Pantalla de Filtros**

La opción de filtros ofrece al usuario dos thumbnails de la fotografía. La original y la imagen con los filtros aplicados.

En la parte central aparece una UITableView con los filtros seleccionados.

Para añadir un nuevo filtro se deberá pulsar el botón + situado en la parte derecha de la Navigation Bar.

Cuando se añade un nuevo filtro se crea una nueva sección y su celda correspondiente con el nombre a CIFilterNone y el botón de propiedades y el switch de activación deshabilitados.

Pulsando sobre la celda se añade un UIPickerView debajo de la celda seleccionada para elegir el filtro a aplicar. Una vez elegido el filtro, se debe pulsar otra vez sobre la celda para ocultar el UIPickerView.

El UISwitch de la parte derecha de la celda permite activar o desactivar cada filtro.

El botón  permite editar alguna de las propiedades de cada filtro. Los valores de cada propiedad se puede editar con el UIStepper o bien pulsando sobre el valor y teclear el valor desde el teclado.

Esta última opción realiza el scroll correspondiente para visualizar y ocultar el teclado.

Para volver a la lista de Filtros debe pulsarse el botón <.

NOTA: por motivos de tiempo solo se han incluido los filtros que permiten una lista de propiedades cuyo tipo son NSNumber. Por este motivo es posible que algunos filtros no produzcan ningún efecto ya que les faltarían algunas propiedades que no son NSNumber.

El botón  permite aplicar los filtros a la imagen tanto en el modo lista de filtros como el modo edición de propiedades de un filtro.

La imagen de la derecha reflejará el resultado de aplicar los filtros que estén activos.

Pulsando sobre las imágenes podrán visualizarse a pantalla completa y realizar zoom sobre ellas.

Cuando existen dos filtros o más se habilita el botón izquierdo de la Navigation Bar que permite mover de posición los filtros y así cambia el orden de aplicación.

Pulsando el botón  podrá eliminar todos los filtros.

Esta opción está gobernada por NSUndoManager de CoreData por lo que para guardar los cambios deberá pulsarse el botón  o bien el botón  para desecharlos.

Una vez guardados, aparecerán en la sección Filtros de la pantalla de edición.

**Requisitos de la Práctica**

* *Crear un App para iPhone o iPad (según el dispositivo que tengas) de cámara con 5 filtros, geoposicionamiento, reconocimiento facial e integración con redes sociales y flickr.*

- funciona correctamente en iPhone 4S y iPad mini. Lo he probado en iPad 2 y 3 pero las dimensiones no son las correctas y habría que ajustarla.

- permite añadir los filtros que se deseen

- en el momento de añadir la fotografía incluye la localización actual del dispositivo.

- reconoce los rostros de las caras que haya en la fotografía utilizando CIDetector.

- permite compartir las fotografías con las actividades que soporta UIActivityViewController excepto las menos populares. Depende de que el usuario tenga instalada la aplicación nativa.

* *Crea un modelo de datos que almacene las fotos, los metadatos disponibles (consulta la documentación), la latitud y longitud y los rectángulos donde se hayan detectado caras. Para almacenar un CGRect, transfórmalo en una cadena mediante NSStringFromCGRect().*
* el modelo se compone de 4 entidades principales, Foto, Rostro, Filtro y Propiedad de Filtro.
* La clase JMFPhoto almacena los metadatos de CGImage, la localización, la geolocalización inversa y los rostros utilizando NSStringFromCGRect(), NSStringFromCGPoint() y sus contrapartidas.
* *Debes de mostrar las imágenes ya obtenidas en una UICollectionView, cada imagen individual (junto a los demás datos) en una table view con celdas personalizadas, además de sacar nuevas fotos.*
* La pantalla principal muestra el álbum de fotos en modo mosaico utilizando un UICollectionView y en modo lista utilizando una UITableView con celdas personalizadas.
* Las nuevas fotos pueden provenir de la cámara, de la galería del dispositivo y desde Flickr.
* *Cuando se saque una foto, debes de buscar las caras (mira la documentación de Core Image). Hazlo en segundo plano. Cuando lo tengas, guárdalo en el modelo y muéstralo en pantalla de alguna manera.*
* en la pantalla de edición se realiza reconocimiento facial utilizando CIDetector en segundo plano utilizando GCD.
* Se muestran en la imagen tanto el rectángulo de la cara como los ojos y boca si los hubiera
* *Tambien debes de obtener la geolocalización de la imagen.*
* Se detecta la localización y la geolocalización inversa en el momento de adquirir la fotografía y se almacena en la entidad JMFPhoto.
* *El usuario debe de poder aplicar al menos 5 filtros. Elige los que más te gusten y asegúrate de que pueda deshacer mediante el undo manager de Core Data.*
* Pueden aplicarse mas de 5 filtros seleccionando de una lista y además se puede mopdificar los valores de sus propiedades.
* *El usuario debe de poder compartir las fotos (mira UIActivityViewController) con al menos una red social.*
* se puede compartir con todas la que soporta UIActivityViewController excepto las menos populares.
* *Un diseño que no haga que Erik se corte las venas.*
* Creo que el diseño es bastante aceptable. A ver que dice Erik.
* *Opcional (sin esto no se puede obtener un 10): sincronizar, en segundo plano, las imagenes del dispositivo con la cuenta flickr del usuario.*
* Realiza la sincronización de las nuevas fotos autenticando mediante OAuth y se realiza background utilizando NSURLSession y NSURLSessionUploadTask.

NOTA: Flickr proprociona otro api diferente con autenticación diferente para el borrado de fotos por lo que la sincronización de las fotos borradas en el dispositivo no la he hecho por motivos de tiempo.

* *Opcional (sin esto es imposible obtener una cerveza la próxima vez que nos veamos): Cuando la tengas hecha, hazte una selfie (vestido, por favor) y compártela en twitter con @frr149.*
* Te envío mejor la splash screen por twitter.

**Otros**

* La App soporta multi-idioma. De momento Español e Inglés.

**Carpetas y ficheros**

Utility

JMFUtility – Métodos estáticos de ayuda.

Flickr

JMFFlickr - Objeto Flickr principal

JMFFlickrOAuth - Objeto de autenticación

JMFFlickrUpload - Objeto de Upload

JMFFlickrUploadResponse - Objeto Response

JMFFlickrSync - Planificador de Uploads

CoreData

Clases \_Entity generados por mogenerator.

JMFCoreDataStack - Objeto principal stack Core Data

JMFArrayViewController - CoreData -> Array Model View Controller

JMFDataTableViewController - CoreData -> Table View Model View Controller

JMFDataCollectionViewController- CoreData -> Collection View Model View Controller

JMFDataViewController - DataTableViewController + DataCollectionViewController

Categories

UIActionSheet+CompletionBlock - UIActionSheet con bloque de finalización

UIAlertView+CompletionBlock - UIAlertView con cloque de finalización

UIImageView+GeometryConversion- UIImageView mas geometría para dibujar los rostros

NSString+URLEncode - NSString mas encoding de OAuth.

Views

Clases de celdas personalizadas

Controllers

Classes de los ViewControllers de cada pantalla

Model

Clases del modelo de Datos generados por mogenerator

Resources

Imágenes, Iconos y recursos strings de multi-idioma