Componentes para iPad y aplicaciones universales - Ejercicios

Índice

1. Programando una aplicación para iPad con Split View (1)

a) Vamos a crear un nuevo proyecto de tipo *Single View Application*. El nombre del proyecto será `BibliotecaMusical` y nuestra organización `es.ua.jtech`. Utilizaremos como prefijo para todas las clases `UA` y será sólo para **iPad**. También desmarcaremos todas las casillas adicionales, ya que no vamos a utilizar por el momento ninguna de esas características (Core Data, ARC, pruebas de unidad, storyboards).

b) Ahora añadimos un controlador de tipo `Split View Controller` a la vista `UAViewController.xib`. ¿Qué dos vistas forman la vista *Split View*?.

c) Creamos la vista de la izquierda (`UITableViewController`). Creamos una nueva clase que llamaremos `ListadoDiscosViewController` la cual será de tipo `UIViewController subclass` y heredará de (subclass of) `UITableViewController`. Recuerda marcar las casillas de *Targeted for iPad* y *With XIB for user interface*

d) Creamos la vista de la derecha, para ello añadiremos un nuevo archivo a nuestro proyecto que se llamará `DetalleDiscoViewController` la cual será de tipo `UIViewController subclass` y heredará de `UIViewController`. Como en el paso anterior dejaremos marcadas las opciones de *Targeted for iPad* y de *With XIB for user interface*.

e) Añadimos el código necesario dentro de la clase `didFinishLaunchingWithOptions` del fichero `UAAppDelegate` que declare un `SplitViewController` y asigne las clases del listado de discos y del detalle a este.

f) Arrancamos la aplicación y comprobamos que funciona bien.¿Qué vista corresponde a la "Master" y cual al "Detail"? ¿Qué pasaría si no asignamos la vista detalle al Split View? ¿Y si no asignamos la vista Master?

g) Modificamos la clase `ListadoDiscosViewController` para que muestre 4 títulos de discos. Deberemos modificar los métodos necesarios de la clase `UITableViewController`.

h) Modificamos la clase `DetalleDiscoViewController` para que muestre una label para el título del álbum seleccionado, otra label para el grupo/autor, otra para el número de temas y por último otra para la carátulal títulos de discos. Deberemos modificar la vista de la clase `DetalleDiscoViewController`, añadir los Outlets necesarios y enlazarlos correctamente.

2. Programando una aplicación para iPad con Split View (2)

Continuamos con el ejercicio anterior, ahora debemos enlazar la vista de la izquierda con la de la derecha. Cuando seleccionemos un álbum, la vista de detalle (derecha) debe de mostrar todos los datos de dicho álbum en sus correspondientes labels (creadas en el

ejercicio anterior).

3. Programando una aplicación para iPad con Split View usando XCode 4.2

En este ejercicio vamos a diseñar la misma aplicación que hemos programado anteriormente pero usando las últimas opciones presentadas en la última versión de XCode. Para ello simplemente deberemos crear un nuevo proyecto al que llamaremos `BibliotecaMusical 2` usando la plantilla `Master-Detail View Controller`. Arranca la aplicación. ¿Qué diferencias encuentras respecto al proyecto de `BibliotecaMusical` que hemos creado anteriormente?

4. Programando una aplicación universal

a) Vamos a crear ahora una aplicación universal desde cero. Creamos un nuevo proyecto en XCode que llamaremos `UnaPeliculaCualquiera`. El proyecto va a ser de tipo *Single View Application*, organización `es.ua.jtech` y prefijo `UA`. Seleccionamos `iPhone` como "target" y desmarcaremos todas las casillas adicionales, ya que no vamos a utilizar por el momento ninguna de esas características (Core Data, ARC, pruebas de unidad, storyboards).

b) Modificamos la vista `UAViewController.xib` añadiendo una label para el título de la pelicula, otro para la sinopsis y una imagen para la carátula (`UIImageView`). Creamos las propiedades y Outlets correspondientes en la clase `UAViewController`.

c) Creamos ahora la vista para iPad que llamaremos `UAViewController_iPad` y añadimos las mismas etiquetas que en la vista de iPhone pero organizadas de distinta manera. Modificamos la propiedad del *Identity Inspector* necesaria para que la vista cargue los Outlets de la controladora y asignamos los Outlets necesarios para que la clase sea *key-value compilant*.

d) Añadimos el código necesario dentro del método `didFinishLaunchingWithOptions` que cargue una vista u otra dependiendo del tipo de dispositivo que esté ejecutando la aplicación.

e) ¿Qué propiedad del proyecto deberemos de modificar para que la aplicación se considere como "Universal"? Cambia dicha propiedad.

f) Arrancamos la aplicación en ambos dispositivos en el simulador y comprobamos que funciona todo bien. ¿Y si queremos añadir una vista de tabla? ¿Tendremos que hacerlo en dos vistas por separado o no hace falta?

5. Programando una aplicación para iPad con un Popover

a) En este ejercicio vamos a mostrar los detalles básicos de una película cualquiera dentro de un popover. Comenzamos creando un nuevo proyecto en XCode que llamaremos `PeliculaEnPopover`. El proyecto va a ser de tipo *Single View Application*, organización `es.ua.jtech` y prefijo `UA`. Seleccionamos `iPad` como "target" y desmarcaremos todas las casillas adicionales, ya que no vamos a utilizar por el momento ninguna de esas características (Core Data, ARC, pruebas de unidad, storyboards).

b) Editamos la vista `UAViewController.xib` y añadimos al menos dos carátulas de películas dentro de botones (`UIButton`).

c) Creamos una clase nueva que será la que muestre el popover. Esta clase la llamaremos `DatosPelicula` y será de tipo `UIViewController`.

d) Modificamos la vista de la clase `DatosPelicula` como queramos, añadiendo al menos dos labels. Añadimos las propiedades y Outlets necesarios a la clase.

e) Implementamos las acciones de los botones de la clase `UAViewController` y las asignamos en la vista.

f) Dentro de las acciones de los botones crearemos un popover y lo mostraremos de modo que la flecha apunte al botón. Los objetos de tipo `UIPopoverController` deben de crearse como propiedades dentro de la clase `UAViewController` y tendremos que realizar las comprobaciones necesarias para que no existan dos popover iguales al mismo tiempo en pantalla.