

Ejercicios de transiciones y storyboards

Índice

1 Pestañas.....	2
2 Aplicación basada en storyboards.....	3

1. Pestañas

Cambiaremos el punto de entrada de la aplicación, para que ahora el controlador principal sea un controlador basado en pestañas. Tendremos tres pestañas: *Películas*, *Búsqueda* y *Configuración*. Se pide:

a) Utilizaremos el controlador de navegación inicial (UAMasterViewController) como contenido para la pestaña *Películas*, y el controlador de búsqueda creado en la sesión anterior (UABusquedaViewController) para la pestaña *Búsqueda*. Vamos a añadir un controlador adicional UAConfiguracionViewController, que será de tipo UIViewController, para la pestaña *Configuración*.

b) Vamos a crear un UITabBarController de forma programática en el *app delegate*. Añadiremos a este controlador las tres pestañas indicadas anteriormente, cada una de ellas instanciando el controlador correspondiente.

Ayuda

Esto deberá realizarse en el método `application:didFinishLaunchingWithOptions:` de `UAppDelegate`. En lugar de poner como controlador raíz `UAMasterViewController`, crearemos un nuevo controlador de tipo `UITabBarController` que hará de controlador raíz, y contendrá los tres controladores anteriores como pestañas.

c) Modificar el inicializador designado de cada controlador para incluir la información del título y el icono de la pestaña. Puedes descargar las plantillas de la sesión para obtener los iconos necesarios e incluirlos en el proyecto (los iconos de las pestañas se encuentran en una carpeta `iconos_filmoteca`).

d) De forma alternativa, podemos implementar todo lo anterior de forma visual en un fichero NIB. En este caso deberemos:

- Crear un NIB con el `UITabBarController` como elemento principal.
- Añadir tres controladores como pestañas, arrastrándolos desde la librería de objetos. Uno de ellos debe ser de tipo `UINavigationController` y los otros de tipo `UIViewController`.
- Modificar el atributo *Class* del inspector de identidad de los controladores `UIViewController` anteriores para establecer su tipo (a `UABusquedaViewController` y a `UAConfiguracionViewController`).
- Establecer el tipo del controlador raíz del controlador de navegación como `UAMasterViewController`.
- Indicar para cada pestaña su título y su icono en los campos del inspector de atributos.
- En el *app delegate* carga el contenido del NIB anterior y asigna el `UITabBarController` raíz a `window.rootViewController`. Crea el *outlet* necesario para tener acceso a dicho controlador.

Importante

Recuerda que en este caso los controladores que se cargan del NIB no se inicializarán mediante sus inicializadores, sino que se recuperará el objeto ya creado directamente del NIB. Por lo tanto, toda la inicialización necesaria debería realizarse en el método `awakeFromNib`, o en `viewDidLoad`, no en su inicializador designado.

2. Aplicación basada en storyboards

Vamos a crear una nueva aplicación utilizando storyboards. Se pide:

- a) Crear un nuevo proyecto basado en pestañas (*Tabbed Application*) llamado `Tareas`, que utilice *storyboards* y ARC.
- b) Crear 3 pestañas (*tabs*). La primera (*Tareas*) deberá contener un controlador de navegación, la segunda (*Configuración*) un controlador tipo tabla estática, y la tercera (*Acerca de*) un controlador básico.
- c) Asignar un icono a cada pestaña. Para ello importaremos en el proyecto el conjunto de iconos que se encuentran en el directorio `iconos_tareas` de las plantillas de la sesión.
- d) En el controlador de navegación (*Tareas*), crear como pantalla raíz una tabla dinámica (lista de tareas), y como pantalla secundaria un controlador de tabla estática (datos de la tarea).
- e) En el controlador de la pestaña *Acerca de*, pon una imagen (por ejemplo el logo del curso) y el nombre del autor mediante una etiqueta de texto.
- f) En la tabla estática de la pantalla de configuración crear dos secciones, con 3 filas la primera de ellas, y 2 filas la segunda. En la primera sección tendremos:
 - Fila de tipo *custom*, con un `UITextField` donde introducir un mensaje, que tendrá como *placeholder* el texto *Ej. Mensaje de alerta*.
 - Fila de tipo *custom*, con un `UISwitch` que nos permite activar o desactivar el sonido de las alertas.
 - Fila de tipo *custom*, con un `UISwitch` que nos permite activar o desactivar la aparición alertas.

En la segunda sección, las filas serán:

- Fila de tipo *right detail* con la versión de la aplicación.
- Fila de tipo *right detail* con el número de *build* de la aplicación.

El aspecto de esta pantalla deberá ser el siguiente:



Pantalla de configuración

Sólo implementaremos el aspecto de esta interfaz. Por simplicidad no implementaremos las funcionalidades asociadas a estas opciones.

g) Dentro de las pantallas de la sección *Tareas*:

- En la pantalla principal (lista de tareas) crearemos una celda prototipo de tipo *Basic* en la tabla dinámica.
- La celda prototipo tendrá un identificador de reutilización *TareaCell*.
- Conectaremos la celda con la pantalla de datos de la tarea mediante un *segue*, que tendrá como identificador *TareaSeleccionada* y será de tipo *push*.
- La pantalla con la lista tendrá como título *Tareas* en la barra de navegación, y un

botón para añadir nuevos *items* a la lista (utilizar el botón del sistema de tipo *Add*, atributo *Identifier*).

- La pantalla con los datos de la tarea tendrá como título *Tarea* en la barra de navegación, y un botón para guardar los cambios realizados en la tarea.
- La tabla estática de la pantalla de datos tendrá una única fila, de tipo *custom*, que contendrá un campo de texto editable con el nombre de la tarea seleccionada. Tendrá como *placeholder* el texto *Ej. Entregar trabajo*, y durante la edición deberá aparecer el botón para vaciar el texto (*Clear Button*)

h) Crear un controlador para la pantalla con la lista de tareas, al que llamaremos *UATareasViewController*. Será de tipo *UITableViewController*, y lo asociaremos a la pantalla de la lista de tareas en el *storyboard* mediante el atributo *Class* del inspector de identidades. En dicho controlador implementaremos lo siguiente:

- Crearemos un *NSMutableArray* como propiedad de la clase, que contendrá la lista de tareas almacenadas. Para cada tarea almacenaremos en la lista únicamente un *NSString* con su nombre (no es necesario crear una nueva clase para las tareas).
- Implementar los métodos de *data source* (*numberOfSectionsInTableView:*, *tableView:numberOfRowsInSection:*, y *tableView:cellForRowAtIndexPath:*) para poblar la tabla de datos a partir de la lista anterior.
- Añadiremos una acción para el botón de la barra de navegación para agregar una nueva tarea. En dicha acción insertaremos una nueva tarea de nombre @"Nueva tarea", y la insertaremos en la última posición de la tabla mediante una animación.
- Añadiremos el botón de edición proporcionado por el sistema como botón izquierdo de la barra de tareas. Esto lo haremos de forma programática en el método *viewDidLoad* del controlador.
- Implementaremos el método *tableView:commitEditingStyle:forRowAtIndexPath:* del delegado para permitir el borrado de tareas.
- Implementaremos el método *tableView:moveRowAtIndexPath:toIndexPath:* del delegado para permitir la reordenación de tareas.



Pantalla de lista de tareas

i) Crear un controlador para la pantalla con los datos de una tarea, al que llamaremos `UATareaViewController`, también de tipo `UITableViewController`. En él implementaremos lo siguiente:

- Una propiedad `tarea` de tipo `NSString` que recogerá el nombre de la tarea que vamos a editar.
- Una propiedad `indexPath` de tipo `NSIndexPath` que almacenará la posición de la tarea que estamos editando.
- Una acción `guardarTarea`, vinculada al botón *Guardar* de la barra de navegación (implementaremos su código en el siguiente apartado)
- Adoptaremos el protocolo `UITextFieldDelegate`, e implementaremos el método

`textFieldShouldReturn:`, en el que llamaremos a `guardarTarea` y devolveremos `YES` para que se cierre el teclado tras editar.

- Al tratarse de una tabla estática, no implementaremos ningún método de *data source* (y tampoco del delegado de la tabla).
- Crearemos un *outlet* `textFieldTarea`, conectado con el campo de texto que tenemos en la tabla estática.
- En `viewDidLoad` mostraremos en el campo de texto anterior el texto que hayamos recibido en la propiedad `tarea`.



Pantalla de datos de la tarea

j) Establecer la comunicación entre la lista de tareas y los datos de la tarea. Para ello

deberemos:

- Para que la comunicación sea bidireccional (es decir, que la lista de tareas puede recibir el aviso de que una tarea ha sido editada), vamos a seguir el patrón delegado. Crearemos en `UATareaViewController.h` un protocolo para el delegado de la pantalla de edición de tareas:

```
@class UATareaViewController;

@protocol UATareaViewControllerDelegate <NSObject>

- (void)tareaViewController:(UATareaViewController *)controller
    didChangeTarea:(NSString *)tarea
    atIndexPath:(NSIndexPath *)indexPath;

@end
```

- Creamos una propiedad delegate en la clase `UATareaViewController`:

```
@property (nonatomic, unsafe_unretained)
    id<UATareaViewControllerDelegate> delegate;
```

- Cuando pulsemos sobre el botón para guardar una tarea, cogeremos el valor introducido en el campo de texto y se lo notificaremos al delegado. Tras esto, volveremos a la pantalla anterior:

```
[self.delegate tareaViewController:self
    didChangeTarea:self.textFieldTarea.text
    atIndexPath:self.indexPath];
[self.navigationController popViewControllerAnimated:YES];
```

- Hacer que `UATareasViewController` adopte el protocolo anterior, implementando el método correspondiente:

```
- (void)tareaViewController:(UATareaViewController *)controller
    didChangeTarea:(NSString *)tarea
    atIndexPath:(NSIndexPath *)indexPath {
    [self.tareas replaceObjectAtIndex:indexPath.row withObject:tarea];
    [self.tableView
        reloadRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
        withRowAnimation:UITableViewRowAnimationFade];
}
```

- Implementar en `UATareasViewController` el método `prepareForSegue:sender:` para configurar el controlador destino en el momento en el que se realiza la transición "TareaSeleccionada":

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if([segue.identifier isEqualToString:@"TareaSeleccionada"]) {
        UATareaViewController *controller =
            segue.destinationViewController;
        controller.delegate = self;
        controller.indexPath = self.tableView.indexPathForSelectedRow;
        controller.tarea = [self.tareas objectAtIndex:
            self.tableView.indexPathForSelectedRow.row];
    }
}
```

k) Guarda la lista de tareas en disco cuando la aplicación se mueva a segundo plano. En este caso, ¿podemos utilizar directamente los métodos que nos proporciona la clase

NSArray? Utiliza el método más sencillo que puedas para realizar el almacenamiento. En el inicializador de `UATareasViewController` se deberá intentar cargar la lista de tareas del disco.

Ayuda

Al trabajar con *storyboards* no tenemos una relación directa entre la clase `UAppDelegate` (que recibe el evento que nos notifica que la aplicación se ha movido a segundo plano), y la clase `UATareasViewController` (que es la que contiene la lista de tareas, y por lo tanto la encargada de guardarla en disco). Sin embargo, hemos visto que existe una forma para comunicar clases que no están directamente relacionadas en el diagrama de clases. Busca entre la documentación `UIApplicationDidEnterBackgroundNotification`.

