Desarrollo de Apps para iOS
Presentación Modal

IWEB 2018-2019

Santiago Pavón

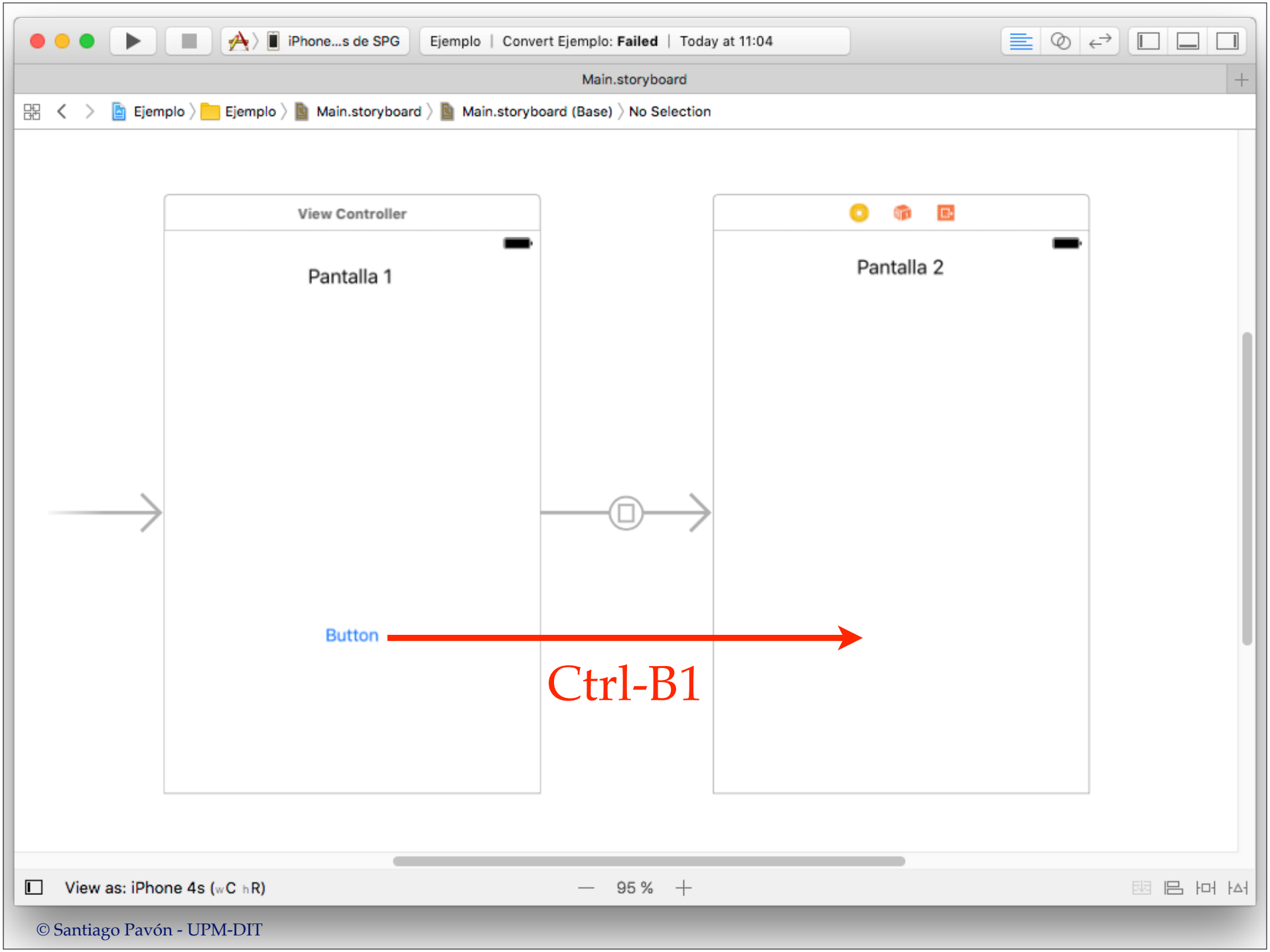
ver: 2018-10-15

Presentación Modales

- Presentar un VC de forma modal consiste en mostrarlo delante del VC actual,
  - y hasta que no se elimina el VC modal, no se puede volver a interactuar con el VC inicial.
- La forma de presentar el VC modal se adapta dependiendo del tamaño disponible.
  - La programación es la misma, pero dependiendo del tipo de terminal o del espacio disponible el VC modal se presentará de la forma más adecuada.
  - Si el terminal es pequeño (ej: iPhone) el VC modal se presentará ocupando toda la pantalla.
  - Si el terminal es grande hay más opciones para presentar el VC modal.
    - presentar el VC modal sustituyendo solo a una subview del VC inicial.
    - presentar el VC modal ocupando una parte de la pantalla.
- Al presentar un VC puede configurarse:
  - el estilo de la **transición** entre los VC.
  - el estilo de **presentación** del VC modal.
- La presentación de un VC suele hacerse diseñando sobre el fichero storyboard y usando segues.
  - Pero también puede hacerse programáticamente.

Storyboard y Segues Modales

- Editando un Storybard con Interface Builder:
  - Para presentar modalmente un VC usar un segue **Present Modally**:
    - Crear un segue de tipo **Present Modally** que conecte el VC origen con el VC destino.
      - El **origen** del segue será normalmente un UIControl o un reconocedor de gestos perteneciente al VC origen.
        - También podría ser el propio objeto VC origen si el segue se va a disparar programáticamente.
      - El **destino** del segue es el VC a mostrar de forma modal.
  - Para volver hacia atrás puede usarse un segue de tipo **unwind**:
    - Crear un segue en el VC modal.
      - El **origen** del segue será un UIControl, un reconocedor de gestos, el propio VC, ...
      - El **destino** del segue es el icono **Exit** y seleccionando un método de vuelta perteneciente a un VC anterior.
- Consultar el tema sobre **Segues** para profundizar en los detalles de su uso.



Programáticamente

- Un VC inicial presenta un segundo VC modalmente ejecutando las siguientes sentencias:

```
// Crear el VC a presentar cargándolo desde el storyboard:
//    La propiedad storyboard apunta al objeto storyboard
//    desde el que se cargo este VC.
//    SecondViewController es la clase del VC a presentar.
//    Configurar el Storyboard ID del segundo VC como "El Segundo VC".
if let vc2 = storyboard?.instantiateViewController(withIdentifier(
                                "El Segundo VC") as? SecondViewController {
    // Configurar los parámetros de entrada del VC a mostrar:
    vc2.msg = "Hola Mundo"

    // Presentar el segundo VC:
    present(vc2, animated: true)
}
```

- Para eliminar el segundo VC ejecutar:

```
dismiss(animated:true)
```

Métodos

- Método para hacer que un VC presente a otro VC:

```
func present(_ viewControllerToPresent: UIViewController,
             animated flag: Bool,
             completion: (@escaping () -> Swift.Void)? = nil)
```

  - Parámetros:
    - El VC a presentar.
    - Un booleano para indicar si se desea presentar de forma animada.
    - Un closure que se llama después de ejecutar el método **viewDidAppear** del VC presentado.
      - **nil** si no se usa.
  - Este método lo llama el VC que desea presentar el nuevo VC.

- Método para quitar del interface el VC presentado antes:

```
func dismiss(animated flag: Bool,
             completion: (@escaping () -> Swift.Void)? = nil)
```

  - Parámetros:
    - Un boolean para indicar si se desea quitar de forma animada o no.
    - Un closure que se ejecutará después del método **viewDidDisappear** del VC presentado.
      - **nil** si no se usa.
  - Este método se puede ejecutar sobre el VC presentado, o sobre el VC que presentó a este VC, o sobre el VC que se ocultó para mostrar a este VC.

Propiedades

- Un VC puede acceder al VC que lo presentó modalmente accediendo a su propiedad:

```
var presentingViewController: UIViewController? { get }
```

  - Apunta al VC que lo presentó modalmente, pero si el VC no fue presentado modalmente, esta propiedad apunta al VC que presento modalmente a algún antepasado. O puede ser nil.

- Un VC puede acceder al VC que ha presentado modalmente accediendo a su propiedad:

```
var presentedViewController: UIViewController? { get }
```

  - Puede apuntar al VC presentado modalmente o a un antepasado en la jerarquía de View Controllers. O puede ser nil.
  - Un VC guarda en esta propiedad la referencia del VC que ha pedido que se presente modalmente.
    - pero el valor de esta propiedad también se asigna en el VC que se ha ocultado para presentar el VC modal. Normalmente son el mismo.

Estilo de la Transición

- El VC modal a mostrar puede aparecer/desaparecer de forma animada.
- El estilo de la animación se indica usando la propiedad:

```
var modalTransitionStyle: UIModalTransitionStyle
```

- Valores:
  - **`UIModalTransitionStyle.coverVertical`**
    - Aparece y desaparece deslizándose desde la parte inferior de la pantalla
  - **`UIModalTransitionStyle.flipHorizontal`**
    - La pantalla rota sobre su eje vertical central.
  - **`UIModalTransitionStyle.crossDissolve`**
    - La pantalla se desvanece cuando aparece o desaparece el VC modal.
  - **`UIModalTransitionStyle.partialCurl`**
    - Una esquina de la pantalla simula el efecto de pasar página. Al tocar el borde doblado se deshace la presentación modal.

Estilo de la Presentación

- Por defecto, el VC presentado modalmente ocupa toda la pantalla.
  - pero podemos configurar otros estilos de presentación.
- El estilo de presentación se indica usando la propiedad:

```
var modalPresentationStyle: UIModalPresentationStyle
```

- Valores:

```
.fullScreen          // ocupa toda la pantalla.
.overFullScreen      // sin quitar el presenting VC.
.pageSheet           // no ocupa todo el ancho de la pantalla.
.formSheet           // ocupa solo el centro de la pantalla.
.currentContext      // lo que ocupe la view del presenting VC.
.overCurrentContext  // sin quitar la presenting VC.
.popover             // presentar como un popover.
...
```

- También pueden usarse las propiedades **`definesPresentationContext`**, **`currentContext`**, **`overCurrentContext`** para indicar que VC por encima en la jerarquía proporciona el contexto (es tapado) para presentar el VC modal.

Size Class y Presentación Adaptativa

- En función del Size Class se puede:
  - indicar que estilo de presentación se desea tener.
  - o presentar modalmente un VC distinto.
- La presentación adaptativa se realiza usando un objeto **UIPresentationController** y el delegado **UIAdaptivePresentationControllerDelegate**.

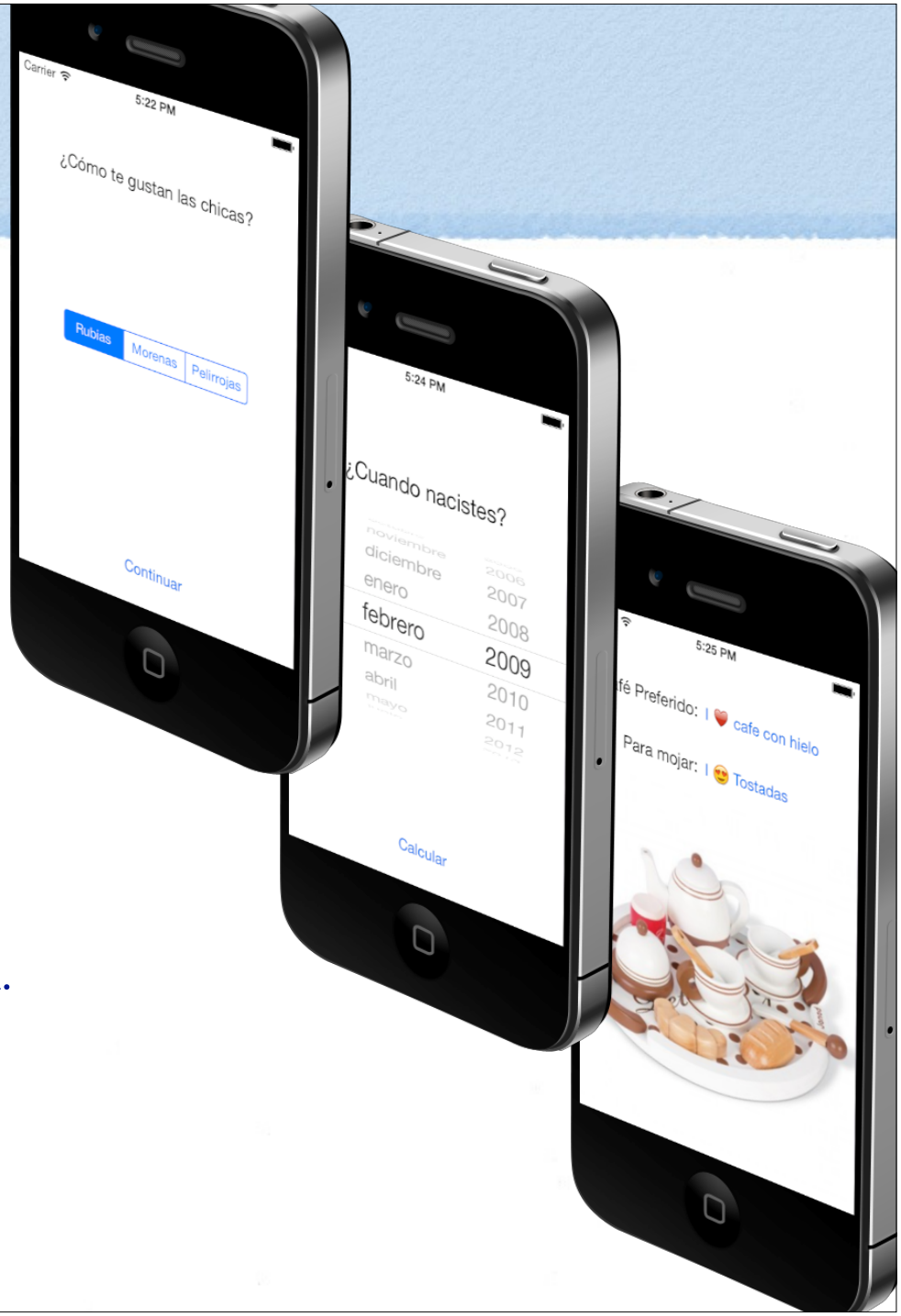
Flujo de Datos

- Normalmente los view controllers por los que se navega se pasan datos entre ellos.
  - Ejemplo: Desde el view controller origen se quiere pasar un dato a un parámetro del siguiente view controller a mostrar.
- Buen diseño: poca cohesión entre los VC:
  - Evitar efectos laterales, facilitar la reutilización, VC independientes, ...
    - No usar variables globales, objetos singletons, el delegado de la aplicación, etc. para pasar datos entre los VC.
- Para pasar datos al siguiente view controller:
  - ej: usar parámetros de entrada, protocolo data source.
- Para devolver datos al view controller anterior.
  - ej: usar delegación, unwind segues, closures.

Demos

Demos

- 1- Color del pelo:
  - Sin segues.
  - Con segues modales y unwind.
- 2 - Días de vida:
  - Con segues modales y unwind.
- 3 - Desayuno
  - Con segues modales y delegación.
  - Con segues modales y unwind.
  - Con segues modales y closures.



Notas

Detalles de la demo: Color del Pelo sin Segues (Programático)

La primera pantalla es ViewController.

- Añadir un Segmented Control para seleccionar el color del pelo. Crear un IBOutlet a este control.
- Añadir el botón Continuar. Crean una IBAction que se ejecutará al pulsar el botón.

Añadir un segundo UIViewController a Main.storyboard para que sea la segunda pantalla.

- Crear el fichero InfoViewController.swift para crear la clase InfoViewController.
- Cambiar la clase del UIViewController añadido al storyboard para que sea InfoViewController.
- Poner el identificador de Storyboard "Info VC" a InfoViewController. (Seleccionarlo en el storyboard y poner ese valor en el campo "Storyboard ID" del inspector de identidad.
- Añadir a la clase InfoViewController la propiedad msg donde ViewController pondrá el string a mostrar.
- Añadir una UILabel y crear un IBOutlet a ella.
- Añadir el botón Volver. Crean una IBAction que se ejecutará al pulsar el botón.
- En viewDidLoad se poner el valor de msg como texto de la label.

Las acciones:

- La acción ShowInfo de ViewController lo que hace es crear una instancia de InfoViewController, asignar un valor a su propiedad msg, y presentar el VC creado.
- La acción volver de InfoViewController quita (se destruye) la instancia de InfoViewController que estamos viendo modalmente para volver a ViewController.

Notas

Detalles de la demo: Color del Pelo con Segues Modales y Unwind

Cambios respecto de la version sin segues:

- No se crean las acciones para los botones Continuar y Volver.

Lo nuevo es:

- Crear un segue desde el botón Continuar hasta InfoViewController. Con el inspector de atributos poner "Show Info VC" como identificador del segue.
- Crear el método "prepare(for segue:sender:)" en ViewController.
  - Este método comprueba el identificador del segue, obtiene una referencia a la instancia de InfoViewController que se ha creado automáticamente, y pone el valor de su propiedad msg. No hay que llamar a present, la instancia de InfoViewController se presenta sola automáticamente.
- Para crear el segue unwind para volver a la primera pantalla:
  - Añadir el método "@IBAction func goHome(_ segue: UIStoryboardSegue)" en ViewController. Marca el punto donde saltará el segue unwind.
  - En el Interface Builder: Crear el segue unwind arrastrando con Ctrl-B1 desde el botón Volver hasta el menú Exit en la cabecera de InfoViewController, y seleccionar goHome en el menú que se despliega.

Detalles de la demo: Dias de Vida con Segues Modales y Unwind

Esta demo es prácticamente idéntica a Color del Pelo con segues modales y unwind, pero en vez de jugar con un Segmented Control y un String, se usar un Date Picker y una fecha.

Notas

Detalles de las demos de los Desayunos

En estas demos hay tres pantallas:

- **ViewController**: Tiene dos strings para guardar el tipo de cafe y galletas que nos gustan. Estos valores se muestran como el titulo de dos botones. Estos botones se usan para saltar a las otras dos pantallas de edición.
- **CoffeeSelectorViewController**: Para editar el String con el tipo de cafe que nos gusta. Esta pantalla tiene una propiedad (currentCoffee) donde ViewController copia el valor a editar.
- **CookieSelectorViewController**: Para editar el String con el tipo de galletas que nos gusta. Esta pantalla tiene una propiedad (currentCookie) donde ViewController copia el valor a editar.

Las tres demos se diferencian en la forma en la se actualiza la primera pantalla con los valores editados por la segunda y tercera pantalla:

- **Delegación**: Las pantallas donde editamos el valor del cafe y galletas definen un protocolo delegado con los mensajes (métodos) que hay que enviar a la primera pantalla con los nuevos valores o para informar de que se ha cancelado la edición. Se crea una propiedad "delegate" en estas pantallas. En el método "prepare(for segue:sender:)" de ViewController se asigna a si mismo (instancia de ViewController) como delegado de las pantallas de edición. La clase ViewController debe adoptar entonces los protocolos delegados e implementar todos los métodos declarados en ellos.
- **Unwind**: Al hacer el unwind a la primera pantalla se devuelven los valores. Para ello se combina el uso de dos métodos:
  - El método "prepare(for segue, sender:)" de CoffeeeSelectorViewController. En este método se actualiza la propiedad currentCoffee con el valor editado en el Text Field (excepto si cancelo la operación).
  - El método "coffeeSelected(_:)" de ViewController donde se recupera el valor actual de la propiedad currentCoffee de CoffeeeSelectorViewController.
  - Lo mismo para CookieSelectorViewController.
- **Closures**: en este caso ViewController le pasa a las pantallas de edición un closure que contiene las sentencias que hay que ejecutar para actualizarse a los nuevos valores. Cada pantalla de edición tienen una propiedad para guardar la closure, y solo tiene que ejecutarlo cuando se termina la edición.

Continua . . .

Notas

Más detalles de CookieSelectorViewController y CoffeeSelectorViewController:

- Para editar los Strings se usa un control UITextField, del que crea un IBOutlet.
- En viewDidLoad se asigna a self como delegado del UITextField. Esto se hace porque queremos que se atiendan las pulsaciones del botón Return del teclado para provocar que se termine la edición.
  - Estas clase deben adoptar el protocolo UITextFieldDelegate.
  - Hay que implementar el método textFieldShouldReturn(_:) para que oculte el teclado y que devuelva true.

- Se ha modificado la clase de view (self.view) para que sea una instancia de UIControl.
  - Se hace desde el inspector de identidades de Interface Builder.
  - Usando el modo asistente, se ha creado una IBAction llamada hideKeyboard que dispara con el evento "Touch Up Inside". Se usa para terminar la edición al tocar sobre el fondo de la pantalla.

- Dependiendo de la versión (delegados, closures o unwind), los botones ok y Cancel pueden apuntar a acciones o disparar segues unwind, o pueden necesitar o no llamar a dismiss para quitar la pantalla modal y volver a ViewController.