Minitutorial 1.

Implementar un Tab Bar

Cuando se seleccione una comida de la tabla, en lugar de mostrarla en un sola vista, tal y como aparece ahora, se debe mostrar un tab bar con tres botones, uno para que muestre sólo el nombre de la comida, otro para que muestre sólo la imagen y otro para que muestre sólo la puntuación. Se debe poder mover por cada uno de estos tres componentes de la tab bar y guardar los datos de todos una vez que se pulse el botón “Save”. Grado de dificultad: 4

Desarrollo de Software para Dispositivos Móviles
Rafael Martínez Rubio

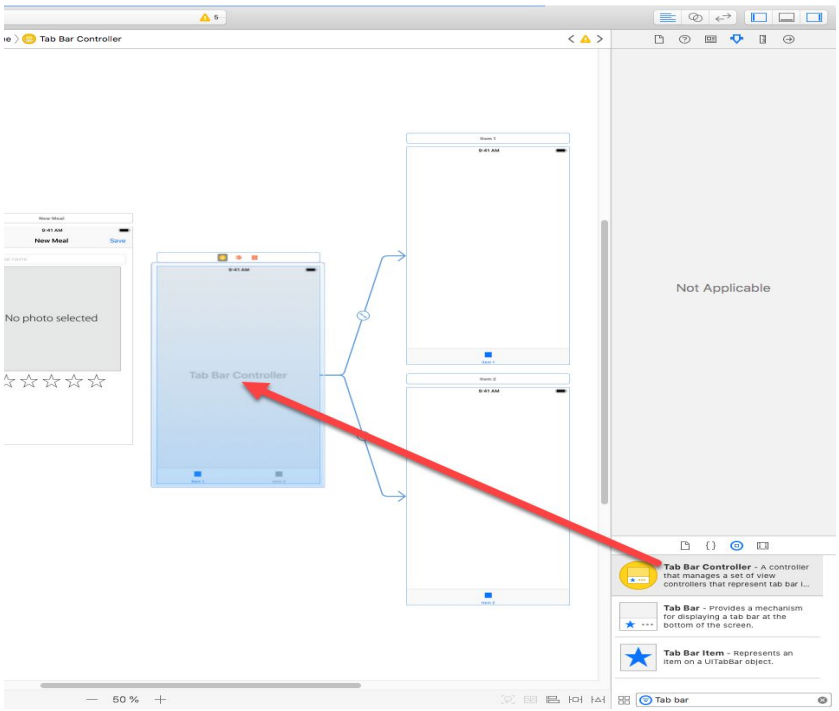
Minitutorial 1. Implementar un Tab Bar

Prerrequisitos

Dado que este tutorial pertenece a la sección de “Basados en FoodTracker”, lo primero será descargarnos el proyecto base , y abrirlo con Xcode.

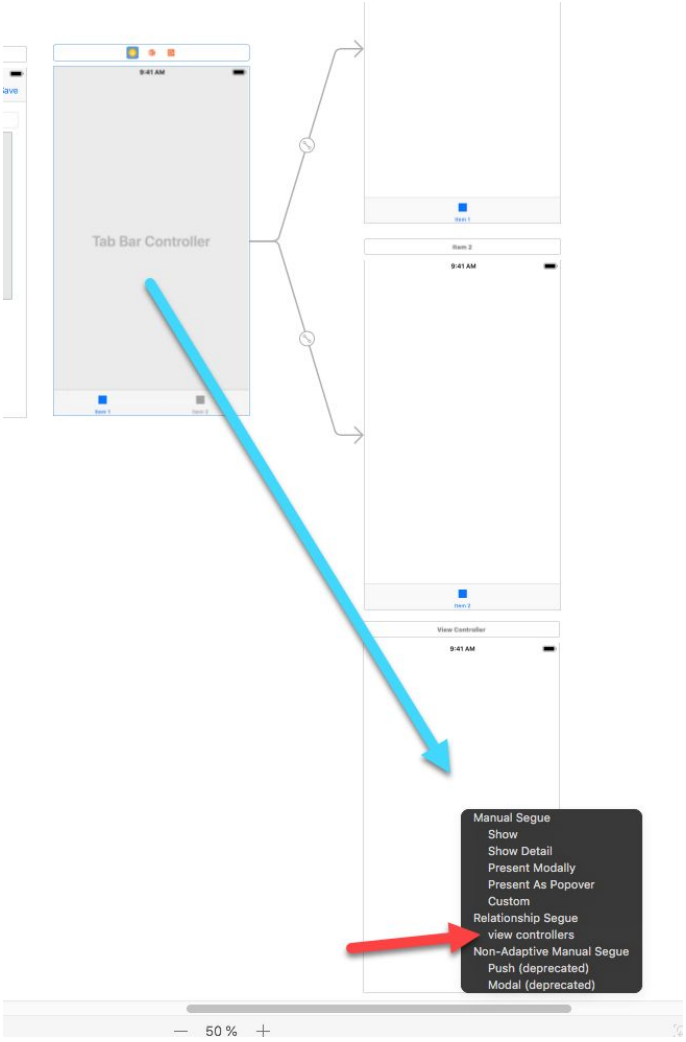
Primeros pasos

Puesto que este minitutorial va sobre Tab bar, qué mejor forma de empezarlo así: desde el área de utilidades, dentro de la librería de objetos, buscamos Tab Bar controller y lo arrastramos hasta nuestro Main.storyboard.



Minitutorial 1. Implementar un Tab Bar

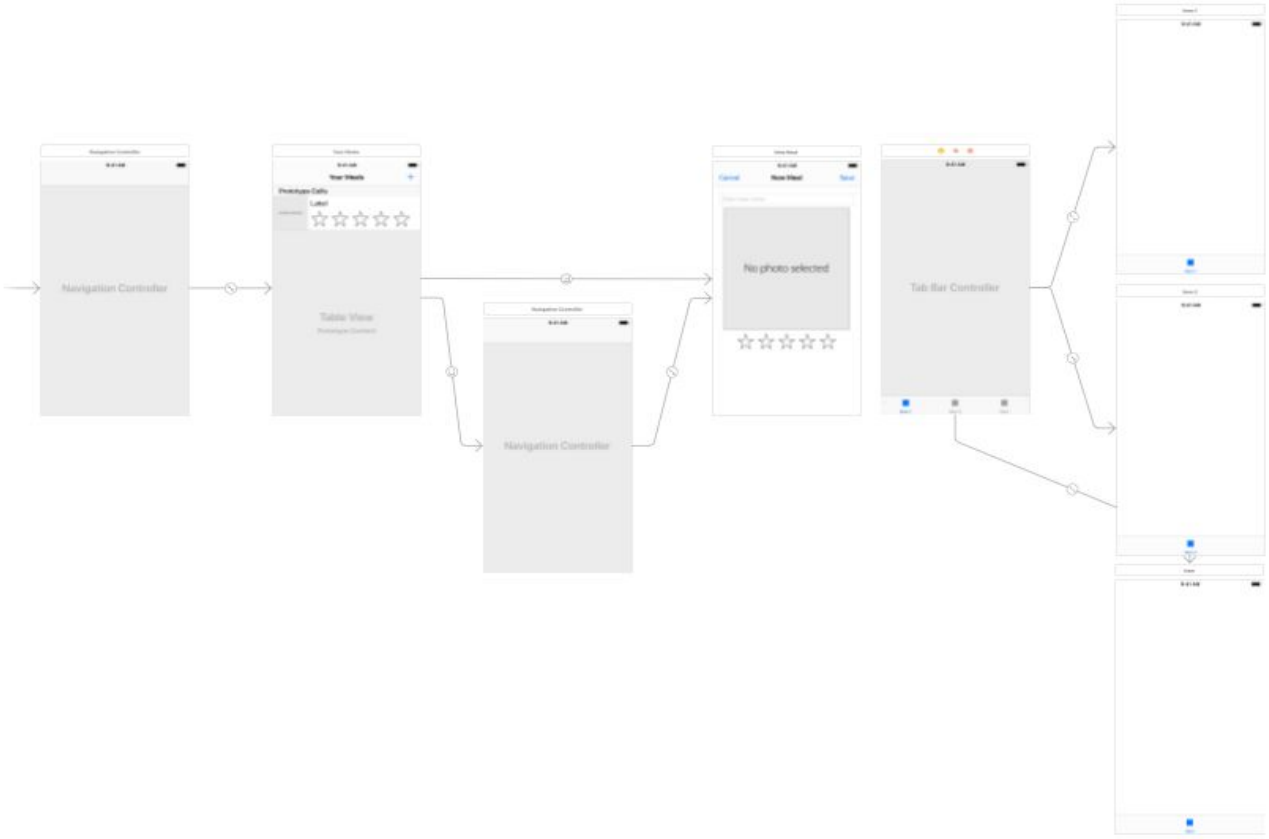
Tal y como pide el enunciado del ejercicio, necesitaremos 3 vistas ya que en el tab bar, se adjuntan 2, así que vamos otra vez a la librería de objetos y añadimos un View Controller.



Tras añadirlo al Main.storyboard, lo incorporamos al Tab Bar de la siguiente forma: pulsando CTRL + click izquierdo del ratón, desde el Tab Bar Controller hasta el View Controller que acabamos de añadir. Ahora sólo seleccionamos la opción “view controllers” para añadirlo.

Checkpoint

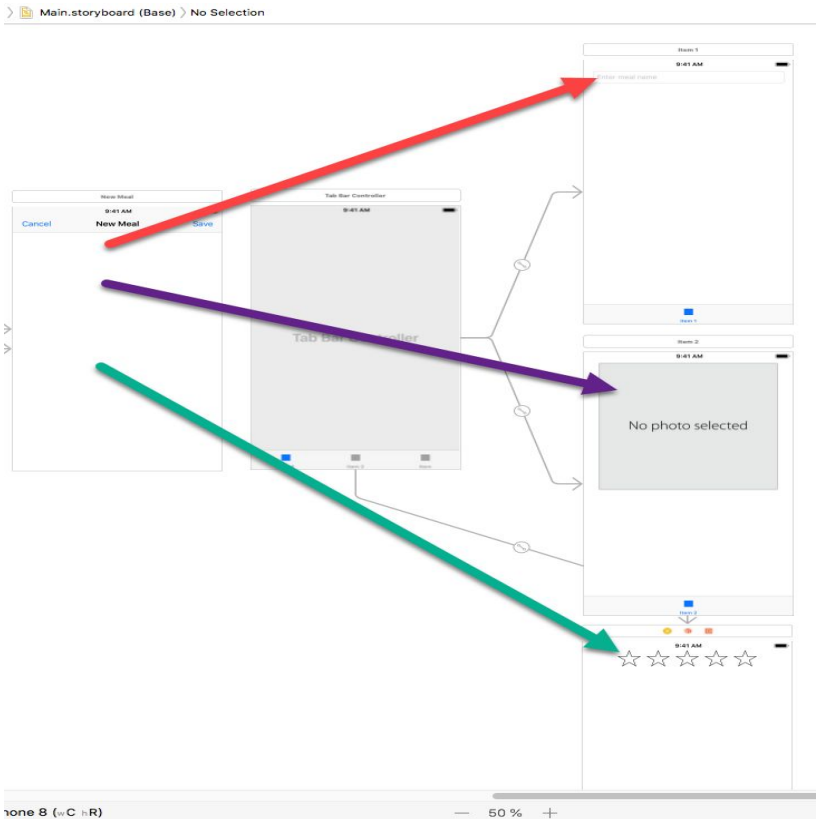
Tu storyboard debería ser algo parecido a la siguiente imagen:



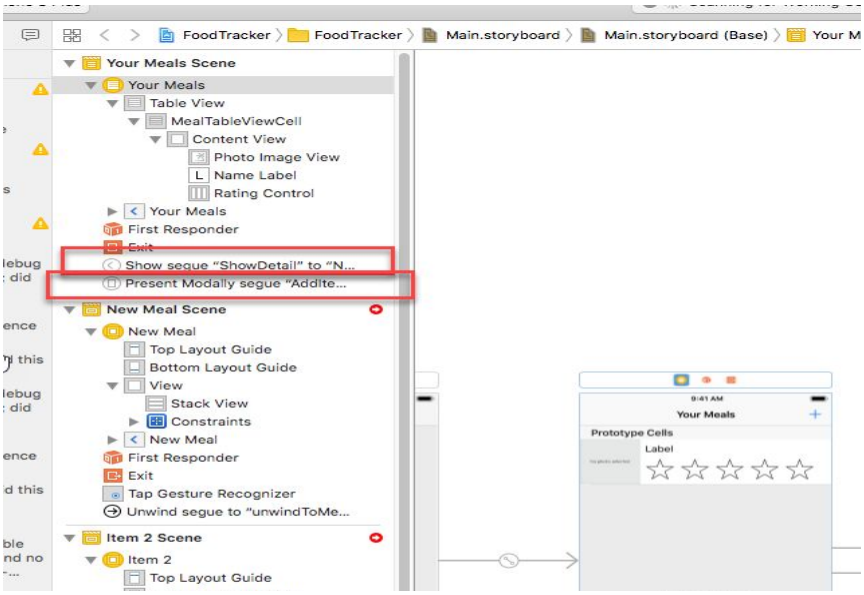
Continuando con el tutorial...

Para continuar, separaremos las 3 partes del MealViewController en las 3 View Controller de nuestro Tab Bar. En la primera pondremos el TextField, en la segunda el ImageView y en la tercera el RatingControl. Para hacerlo más fácil, simplemente arrastraremos del MealViewController hasta cada uno de los View Controller.

Minitutorial 1. Implementar un Tab Bar

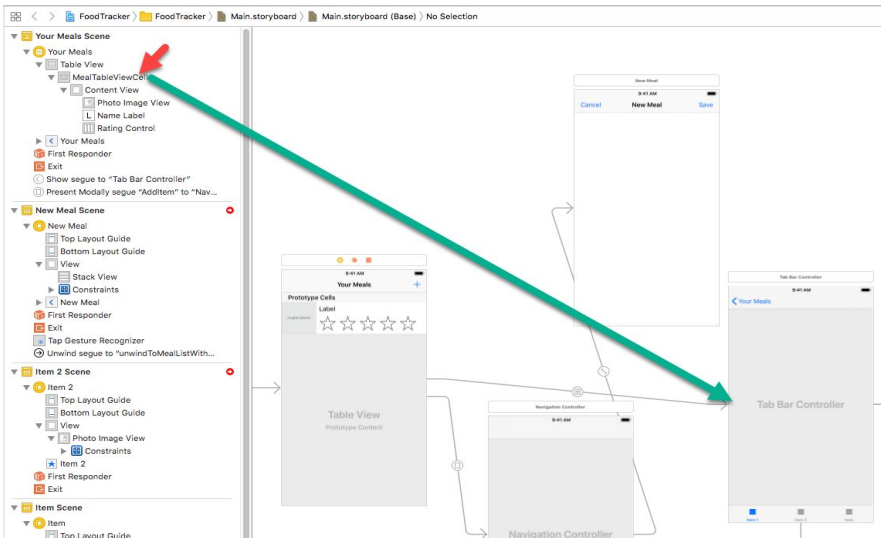


Lo siguiente será sustituir nuestra nueva Tab Bar por el MealViewController con el propósito de generar el flujo correcto ya sea añadiendo una nueva comida o viendo en detalle una ya existente. Para esto, pinchamos sobre el MealViewController y en el lado izquierdo del área de edición vemos que tiene 2 segues.

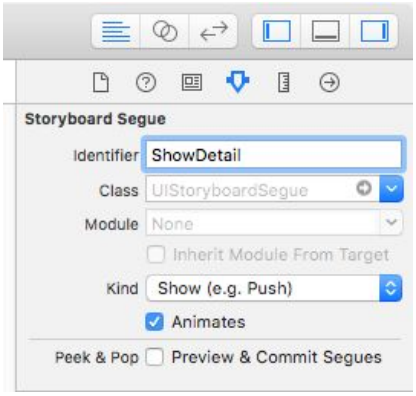


Pinchando en cada uno de los segue vemos cómo el show segue tiene un identificador llamado ShowDetail, que es el segue que irá a nuestro Tab Bar. Y el otro, AddItem, va previamente a una navigation Controller.

Eliminamos el primer segue pinchando en él y en el botón suprimir. Ahora lo volvemos a crear pinchando en su apartado de MealTableViewCell y con el CTRL pulsado, arrastramos el click izquierdo hasta nuestro nuevo Tab Bar Controller, eligiendo la opción : *Selection Segue, show*.



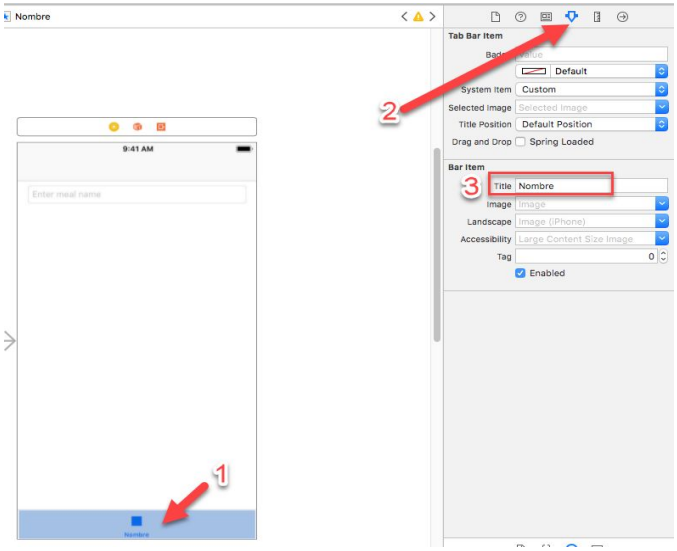
Además, al pinchar en el segue, debemos de añadir el mismo identificador que ya tenía, pinchando en el inspector de atributos y escribiendo **ShowDetail**. **Es necesario** que se llame así, **puesto que es como aparece en el case dentro del MealTableViewControlle**r.



El segue restante se trata de un modal segue desde un Navigation Controller, como sólo puede ir a un solo controlador a la vez, en vez de borrarlo podemos simplemente pinchar en este Navigation Controller y crear un segue hasta el Tab Bar Controller igual que hemos hecho antes.

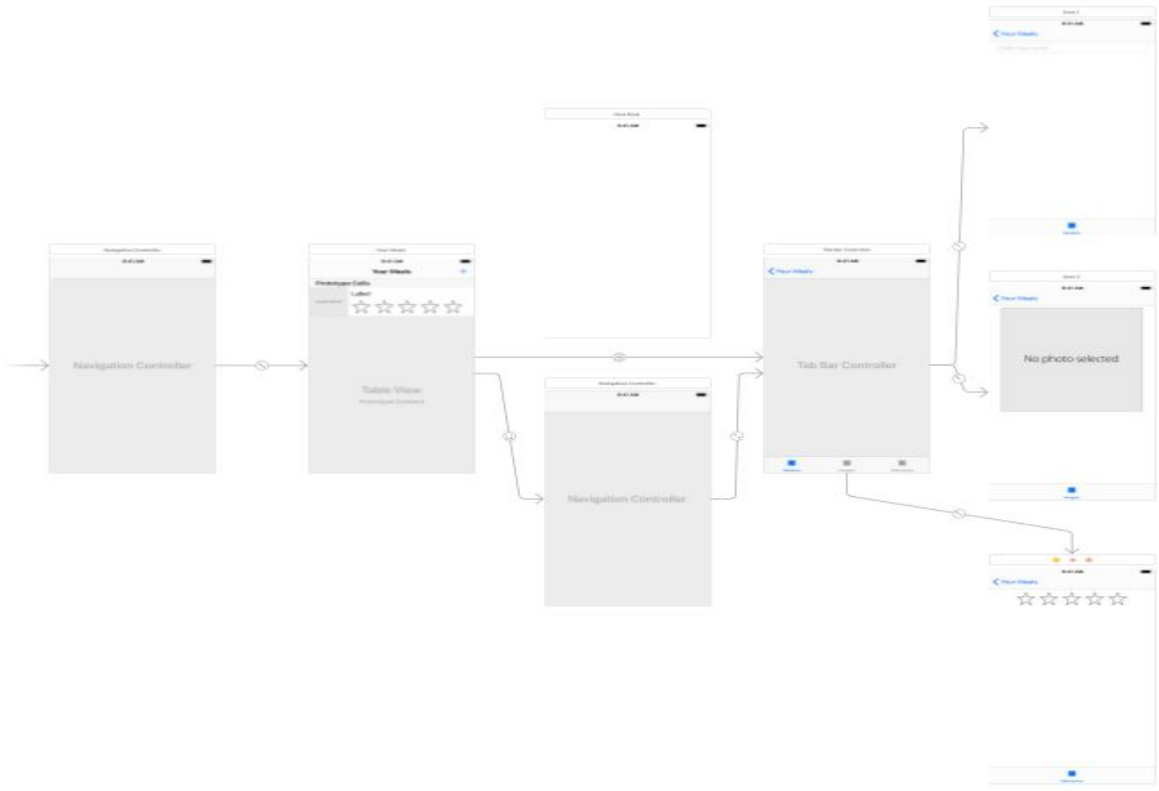
Con la excepción de seleccionar el tipo de segue: root view controller.

Para dejarlo más bonito, pinchamos en cada una de las vistas del tab bar, sobre la opción marcada que por defecto será Item 1, Item 2 e Item 3 respectivamente. Con el inspector de atributos adecuamos sus nombres cambiando el atributo *Title* donde escribiremos Nombre, Imagen y Valoración, en cada una de las vistas.



Checkpoint

Tu storyboard debería ser algo parecido a la siguiente imagen:



Pasamos al código

Nos vamos al navegador del proyecto, donde pinchamos con el botón derecho del ratón sobre la principal carpeta del proyecto (FoodTracker) seleccionamos New File, elegimos de tipo Swift y lo nombraremos **MealTabBarController**.

Ahora iniciaremos nuestro controlador con lo básico, importamos la librería *UIkit* y creamos una clase con el mismo nombre del controlador, la cual heredará de la clase *UITabBarController* . Además sobrecargamos la función viewDidLoad()

```
10  import UIKit
11
12  class MealTabBarController: UITabBarController {
13
14      override func viewDidLoad() {
15          super.viewDidLoad();
16      }
```

Tomando como referencia el *MealViewController* que estamos sustituyendo, cogeremos el código donde crea un nuevo objeto de tipo *Meal* que será manejado por nuestro controlador.

```
18      /*
19       This value is either passed by `MealTableViewController` in `prepare(for:sender:)`
20       or constructed as part of adding a new meal.
21       */
22      var meal: Meal?;
```

Y como ya habrás podido imaginar **necesitaremos 3 controladores distintos** para cada una de las vistas que componen nuestra Tab Bar así que crearemos 3 archivos mas de tipo Swift:

- ***MealNameViewController*** para la vista del nombre.
- ***MealPhotoViewController*** para gestionar la imagen de la comida
- ***MealRatingViewController*** para la valoración.

A continuación, **desde el Main.storyboard establecemos a cada vista del Tab Bar su respectiva Custom Class que acabamos de crear.** Para eso simplemente pinchamos en cada vista, luego en el inspector de identidad y escribimos el nombre de la clase que va a controlar cada una de estas 3 vistas.

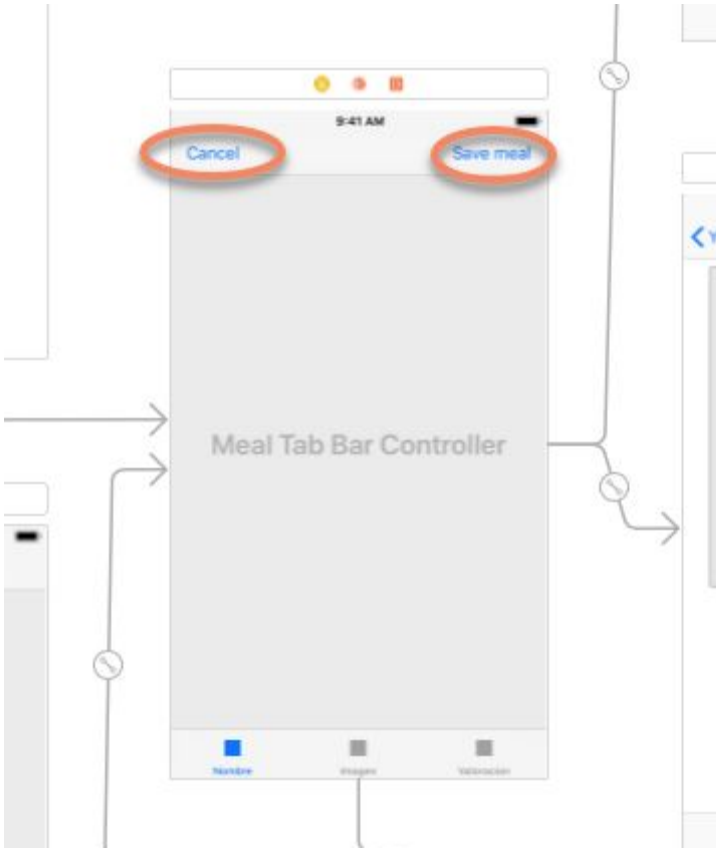
En el controlador que estamos sustituyendo, el *MealViewController* **tenía 3 parámetros para manejar el nombre, la foto y la valoración de la comida**, nosotros necesitaremos esos mismos tres parámetros sólo que ahora **serán las 3 clases que acabamos de crear** por lo que transformaremos esto:

```
14      //MARK: Properties
        @IBOutlet weak var nameTextField: UITextField!
        @IBOutlet weak var photoImageView: UIImageView!
        @IBOutlet weak var ratingControl: RatingControl!
```

En esto:

```
15      weak var nameVC: MealNameViewController?
16      weak var photoVC: MealPhotoViewController?
17      weak var ratingVC: MealRatingViewController?
```

Y terminando ya con esta clase, sólo nos faltará **añadir el botón de Cancelar y el de Guardar la comida**. Para hacer esto, volvemos al Main.storyboard y añadimos los dos botones desde la librería de objetos, buscaremos Bar button Item y tras ponerlos y renombrarlos tendremos algo así:



El **botón de Save meal lo guardemos como Outlet** para poder activar y desactivarlo cuando exista un nombre y necesitaremos **una action con el botón Cancel** puesto que debemos distinguir si se ha creado **una vista modal** o un **push en la pila de presentaciones**.

Ambas se crean con el CTRL pulsado y arrastrando hasta el controlador MealTabBarcontroller, por supuesto.

El botón de Save meal lo guardaremos como saveButton:

```
9  import Foundation
10 import UIKit
11
12 class MealTabBarController: UITabBarController {
13
       @IBOutlet weak var saveButton: UIBarButtonItem!
```

Y la action de cancelar del botón de Cancelar quedaría así:

```
    @IBAction func cancel(_ sender: UIBarButtonItem) {
31
32
33  }
```

Y podemos usar la función con el mismo nombre, perteneciente a la clase MealViewController, así que la copiaremos y pegaremos en nuestro método.

```
    @IBAction func cancel(_ sender: UIBarButtonItem) {
31      // Depending on style of presentation (modal or push presentation), this view controller needs to be
            dismissed in two different ways.
32      let isPresentingInAddMealMode = presentingViewController is UINavigationController
33
34      if isPresentingInAddMealMode {
35          dismiss(animated: true, completion: nil)
36      }
37      else if let owningNavigationController = navigationController{
38          owningNavigationController.popViewController(animated: true)
39      }
40      else {
41          fatalError("The MealViewController is not inside a navigation controller.")
42      }
43  }
```

Este método no hará otra cosa más que, en el caso de venir de estar añadiendo una nueva comida, se ha creado un modal, en ese caso lo borra con la animación propia de iOS y listo. Por otro lado, si hemos seleccionado una comida ya existente, se ha realizado un Show que implica añadir otra vista a la pila de vistas por lo que quitaría esta última.

Para terminar, también usaremos la función ya existente en la clase *MealViewController* *updateSaveButtonState* para que no se use el botón de guardado si no hay nombre para la comida.

```
138     //MARK: Private Methods
139
140     private func updateSaveButtonState() {
141         // Disable the Save button if the text field is empty.
142         let text = nameTextField.text ?? ""
143         saveButton.isEnabled = !text.isEmpty
144     }
```

Pero la cambiaremos para que reciba un texto que vendrá del controlador del nombre y **no sea privada**.

```
46      func updateSaveButtonState(withText text: String) {
47          // Disable the Save button if the text field is empty.
48          saveButton.isEnabled = !text.isEmpty
49          navigationItem.title = text
50      }
```

De esta forma la ventana tendrá el nombre que se le pase, que será el nombre de la comida.

Checkpoint MealTabBarController

```
9  import Foundation
10 import UIKit
11
12 class MealTabBarController: UITabBarController {
13
14     @IBOutlet weak var saveButton: UIBarButtonItem!
15
16     weak var nameVC: MealNameViewController?
17     weak var photoVC: MealPhotoViewcontroller?
18     weak var ratingVC: MealRatingViewcontroller?
19
20     override func viewDidLoad() {
21         super.viewDidLoad();
22     }
23
24     /*
25      This value is either passed by `MealTableViewController` in `prepare(for:sender:)`
26      or constructed as part of adding a new meal.
27      */
28     var meal: Meal?
29
30     @IBAction func cancel(_ sender: UIBarButtonItem) {
31         // Depending on style of presentation (modal or push presentation), this view controller needs to be dismissed in two different ways.
32         let isPresentingInAddMealMode = presentingViewController is UINavigationController
33
34         if isPresentingInAddMealMode {
35             dismiss(animated: true, completion: nil)
36         }
37         else if let owningNavigationController = navigationController{
38             owningNavigationController.popViewController(animated: true)
39         }
40         else {
41             fatalError("The MealViewController is not inside a navigation controller.")
42         }
43     }
44     //MARK: Private Methods
45
46     private func updateSaveButtonState(withText text: String) {
47         // Disable the Save button if the text field is empty.
48         saveButton.isEnabled = !text.isEmpty
49         navigationItem.title = text
50     }
51
52 }
```

MealNameViewController

Lo primero será **abrir el archivo Swift,** que creamos hace muy poquito, **importar la librería *UIKit*** y **heredar de las clases** *UIViewController*, *UITextFieldDelegate*, *UINavigationControllerDelegate*

Volviendo a tomar ejemplo del ya existente MealViewController, podemos usar los **3 métodos dedicados al TextField** que encontraremos bajo la notación *//MARK: UITextFieldDelegate*

```
44      //MARK: UITextFieldDelegate
45
46      func textFieldDidBeginEditing(_ textField: UITextField) {
47          // Disable the Save button while editing.
48          saveButton.isEnabled = false
49      }
50
51      func textFieldShouldReturn(_ textField: UITextField) -> Bool {
52          // Hide the keyboard.
53          textField.resignFirstResponder()
54          return true
55      }
56
57      func textFieldDidEndEditing(_ textField: UITextField) {
58          updateSaveButtonState()
59          navigationItem.title = textField.text
60      }
```

Además de usar el delegate de la entrada de texto:

```
32          // Handle the text field's user input through delegate callbacks.
33          nameTextField.delegate = self
```

Una vez establecida la clase controladora, arrastramos el LabelText ya existente hasta la clase que estamos definiendo.

Será una conexión Outlet, de nombre *nameTextField* , tipo UITextField y Weak. Outlet porque no necesitamos una acción, necesitamos una objeto que manejar, el tipo porque es el tipo correcto que usaremos, y de guardado weak.

Para terminar con esta clase, debemos coger la comida de la vista de la que venimos por lo que haremos lo siguiente:

```
20      override func viewDidLoad() {
21          // Handle the text field's user input through delegate callbacks.
22          nameTextField.delegate = self
23
24          if let tbc = tabBarController as? MealTabBarController{
25              tbc.nameVC = self
26              if let meal = tbc.meal{
27                  nameTextField.text = meal.name
28              }
29              tbc.updateSaveButtonState(withText: nameTextField.text ?? "")
30          }
31      }
```

La línea 22 (que hemos reutilizado de MealViewcontroller) sirve para que usemos esta clase como delegate que se encargará del Text field.

Con el resto, lo que hacemos es obtener de la vista anterior, el nombre de la comida el cual usaremos para ponerlo (si existe) en el textField, además de título de la ventana en el caso de que ya haya nombre de la comida.

Y para terminar con esta clase, cambiamos un poco los métodos para su correcto funcionamiento.

```
35      func textFieldDidBeginEditing(_ textField: UITextField) {
36          // Disable the Save button while editing.
37          if let tbc = tabBarController as? MealTabBarController{
38              tbc.saveButton.isEnabled = false
39          }
40      }
```

En esto sólo interactuamos con el botón de la vista anterior en el caso en que se esté editando el campo de texto.

```
48      func textFieldDidEndEditing(_ textField: UITextField) {
49          if let tbc = tabBarController as? MealTabBarController{
50          tbc.updateSaveButtonState(withText: nameTextField.text ?? "")
51          }
52      }
53  }
```

Y aquí pasaremos el nombre de la comida para que se cambie correctamente el título de la ventana.

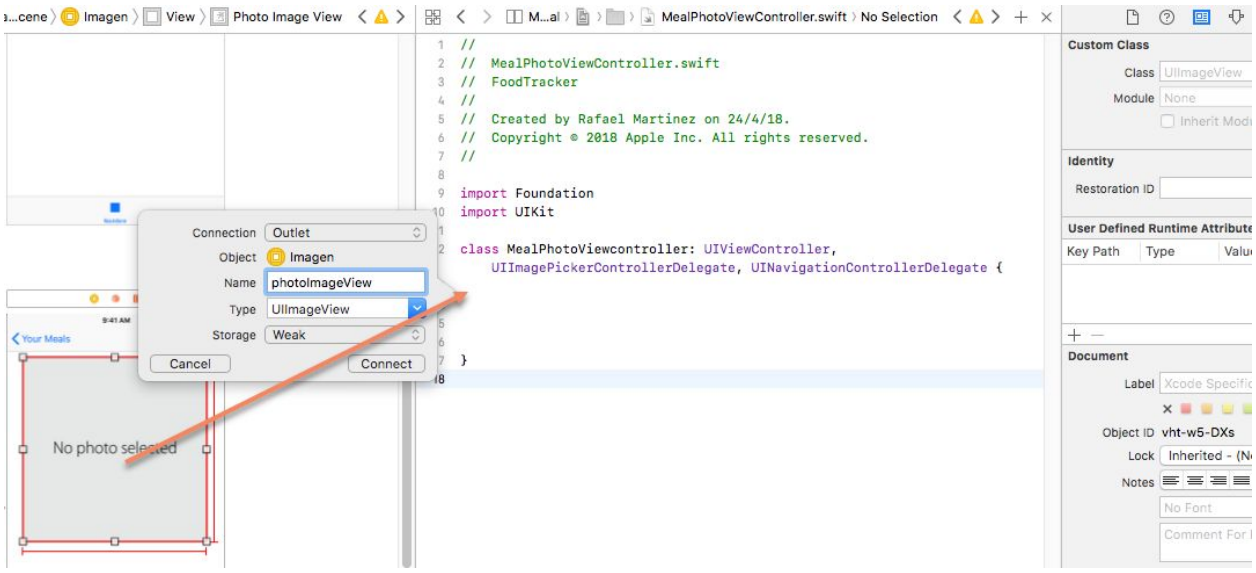
Hemos terminado este controlador por completo, deberías ver algo así en él.

```
import Foundation
import UIKit

class MealNameViewController : UIViewController, UITextFieldDelegate, UINavigationControllerDelegate{

    @IBOutlet weak var nameTextField: UITextField!

    override func awakeFromNib() {
        super.awakeFromNib()
    }

    override func viewDidLoad() {
        // Handle the text field's user input through delegate callbacks.
        nameTextField.delegate = self

        if let tbc = tabBarController as? MealTabBarController{
            tbc.nameVC = self
            if let meal = tbc.meal{
                nameTextField.text = meal.name
            }
            tbc.updateSaveButtonState(withText: nameTextField.text ?? "")
        }
    }

    //MARK: UITextFieldDelegate

    func textFieldDidBeginEditing(_ textField: UITextField) {
        // Disable the Save button while editing.
        if let tbc = tabBarController as? MealTabBarController{
            tbc.saveButton.isEnabled = false
        }
    }

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        // Hide the keyboard.
        textField.resignFirstResponder()
        return true
    }

    func textFieldDidEndEditing(_ textField: UITextField) {
        if let tbc = tabBarController as? MealTabBarController{
        tbc.updateSaveButtonState(withText: nameTextField.text ?? "")
        }
    }
}
```

MealPhotoViewController

Lo primero será **abrir el archivo Swift,** que creamos hace muy poquito, **importar la librería *UIKit*** y **heredar de las clases:** *UIViewController*, *UIImagePickerControllerDelegate* y *UINavigationControllerDelegate*

Después, crearemos el Outlet perteneciente a la imagen de la comida. Lo llamaremos *photoImageView*, por ejemplo



Creamos la sobrecarga de awakeFromNib y viewdidLoad como siempre, pero en esta última funciona, al igual que hicimos con el nombre. Cargamos la foto de la vista anterior usando self, lo cual quedaría así:

```
class MealPhotoViewcontroller: UIViewController, UIImagePickerControllerDelegate, UINavigationControllerDelegate {

    @IBOutlet weak var photoImageView: UIImageView!

    override func awakeFromNib() {
        super.awakeFromNib()
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        if let tbc = tabBarController as? MealTabBarController{
            tbc.photoVC = self
            if let meal = tbc.meal{
                photoImageView.image = meal.photo
            }
        }
    }
```

Ahora sólo nos quedará crear las funciones del delegate de la imagen, que para mayor simplicidad podemos copiar las ya existentes en el archivo MealViewController y que se explicaran a continuación.

```
31      //MARK: UIImagePickerControllerDelegate
32      func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
33          // Dismiss the picker if the user canceled.
34          dismiss(animated: true, completion: nil)
35      }
36
37      func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {
38
39          // The info dictionary may contain multiple representations of the image. You want to use the original.
40          guard let selectedImage = info[UIImagePickerControllerOriginalImage] as? UIImage else {
41              fatalError("Expected a dictionary containing an image, but was provided the following: \(info)")
42          }
43
44          // Set photoImageView to display the selected image.
45          photoImageView.image = selectedImage
46
47          // Dismiss the picker.
48          dismiss(animated: true, completion: nil)
49      }
```

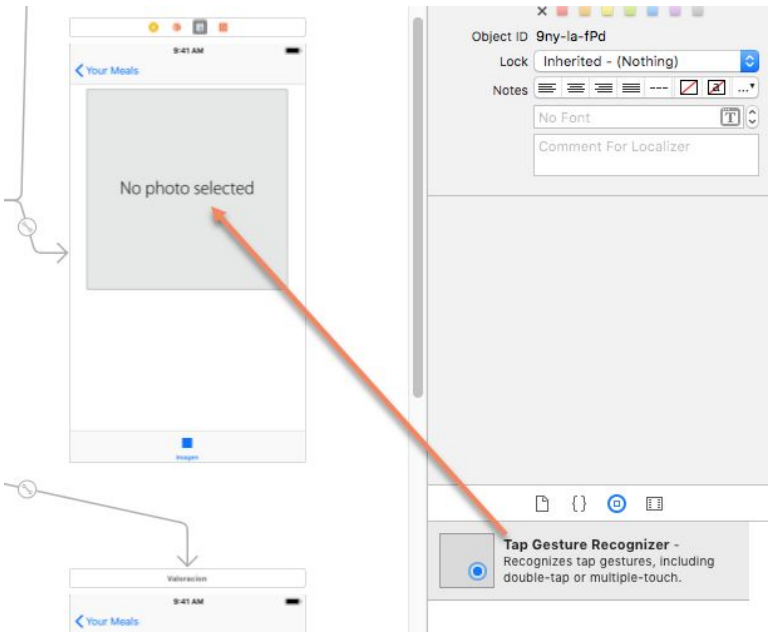
imagePickerControllerDidCancel, actuará cuando el usuario esté eligiendo una foto para la comida pero decida cancelar, en ese momento se quitará la ventana de selección de imágenes.

imagePickerController actuará tras la interacción del usuario con el objeto de imagen en sí; maneja posibles errores, cambia la imagen seleccionada o retrocede en caso de que así lo quiera el usuario. Más información en [la documentación oficial]

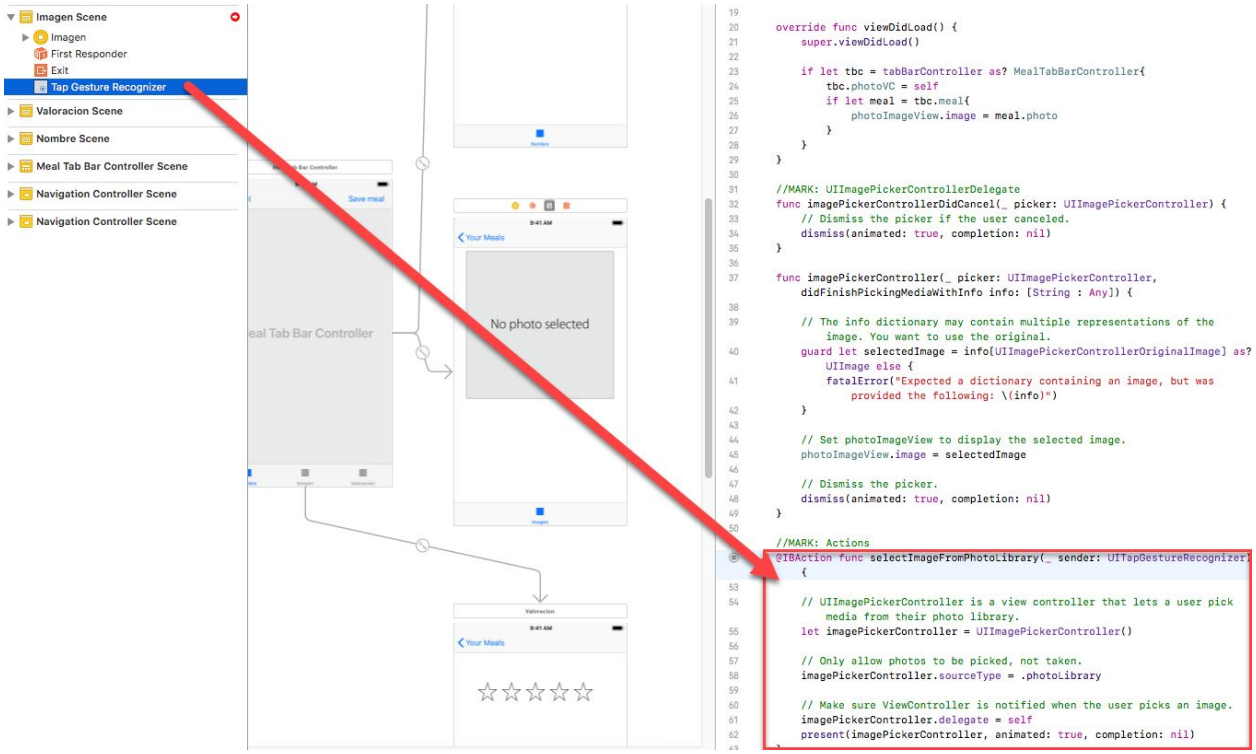
```
50      //MARK: Actions
51      @IBAction func selectImageFromPhotoLibrary(_ sender: UITapGestureRecognizer) {
52
53          // UIImagePickerController is a view controller that lets a user pick media from their photo library.
54          let imagePickerController = UIImagePickerController()
55
56          // Only allow photos to be picked, not taken.
57          imagePickerController.sourceType = .photoLibrary
58
59          // Make sure ViewController is notified when the user picks an image.
60          imagePickerController.delegate = self
61          present(imagePickerController, animated: true, completion: nil)
62      }
```

Y por último, **selectImageFromPhotoLibrary** el método en sí que hará que pueda navegar a través de la librería, creando una nueva instancia local de la clase *imagePickercontroller*. Más información en la documentación oficial.

Minitutorial 1. Implementar un Tab Bar



Ahora debemos añadir un Tap Gesture Recognizer a la imagen y, para finalizar, asignarla a la función selectImagefromPhotoLibrary para que se llame cuando se realice un tap sobre la imagen.



Hemos terminado este controlador por completo, deberías ver algo así en él:

```
import Foundation
import UIKit

class MealPhotoViewcontroller: UIViewController, UIImagePickerControllerDelegate, UINavigationControllerDelegate {

    @IBOutlet weak var photoImageView: UIImageView!

    override func awakeFromNib() {
        super.awakeFromNib()
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        if let tbc = tabBarController as? MealTabBarController{
            tbc.photoVC = self
            if let meal = tbc.meal{
                photoImageView.image = meal.photo
            }
        }
    }

    //MARK: UIImagePickerControllerDelegate
    func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
        // Dismiss the picker if the user canceled.
        dismiss(animated: true, completion: nil)
    }

    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {

        // The info dictionary may contain multiple representations of the image. You want to use the original.
        guard let selectedImage = info[UIImagePickerControllerOriginalImage] as? UIImage else {
            fatalError("Expected a dictionary containing an image, but was provided the following: \(info)")
        }

        // Set photoImageView to display the selected image.
        photoImageView.image = selectedImage

        // Dismiss the picker.
        dismiss(animated: true, completion: nil)
    }

    //MARK: Actions
    @IBAction func selectImageFromPhotoLibrary(_ sender: UITapGestureRecognizer) {

        // UIImagePickerController is a view controller that lets a user pick media from their photo library.
        let imagePickerController = UIImagePickerController()

        // Only allow photos to be picked, not taken.
        imagePickerController.sourceType = .photoLibrary

        // Make sure ViewController is notified when the user picks an image.
        imagePickerController.delegate = self
        present(imagePickerController, animated: true, completion: nil)
    }
}
```

MealRatingViewController

Abriremos el archivo Swift, que creamos anteriormente, **importar la librería *UIKit*** y **heredar de la clase:** *UIViewController*

Añadimos el Outlet correspondiente a las estrellas que servirán como valoración.

Y para terminar la sobrecarga de awakeFromNib y de viewdidLoad donde añadiremos, de la misma forma que de las 2 clases que hemos creado (MealNameViewController y MealPhotoViewController), un apartado dentro de viewDidLoad donde accederemos a la vista padre para conseguir la valoración.

```
16      override func awakeFromNib() {
17          super.awakeFromNib()
18      }
19
20      override func viewDidLoad() {
21          super.viewDidLoad()
22
23          if let tBC = tabBarController as? MealTabBarController {
24              tBC.ratingVC = self
25              if let meal = tBC.meal {
26                  ratingControl.rating = meal.rating
27              }
28          }
29      }
```

Hemos terminado este controlador por completo, deberías ver algo así en él:

```
import Foundation
import UIKit

class MealRatingViewcontroller: UIViewController {

    @IBOutlet weak var ratingControl: RatingControl!

    override func awakeFromNib() {
        super.awakeFromNib()
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        if let tBC = tabBarController as? MealTabBarController {
            tBC.ratingVC = self
            if let meal = tBC.meal {
                ratingControl.rating = meal.rating
            }
        }
    }
}
```

Para terminar

El último archivo que tendremos que editar es MealTableViewController. La razón es que los segues estaban antes hechos hacia MealViewController y ahora están hacia nuestro MealTabBarController, por lo que tendremos que cambiar 2 cosas; la primera de ellas, tal y como se ve en la próxima imagen:

```
@IBAction func unwindToMealList(sender: UIStoryboardSegue) {
    if let sourceViewController = sender.source as? MealTabBarController, let meal = sourceViewController.meal {

        if let selectedIndexPath = tableView.indexPathForSelectedRow {
```

Dentro de la función unwindToMealList, que se ejecutará al presionar el boton de Guardar.

```
121         case "ShowDetail":
122             guard let mealDetailViewController = segue.destination as? MealTabBarController else {
123                 fatalError("Unexpected destination: \(segue.destination)")
124             }
125
126             guard let selectedMealCell = sender as? MealTableViewCell else {
```

Y poco más arriba, en la sobrecarga de la función prepare es necesario para al Tab Bar le llegue la comida seleccionada.