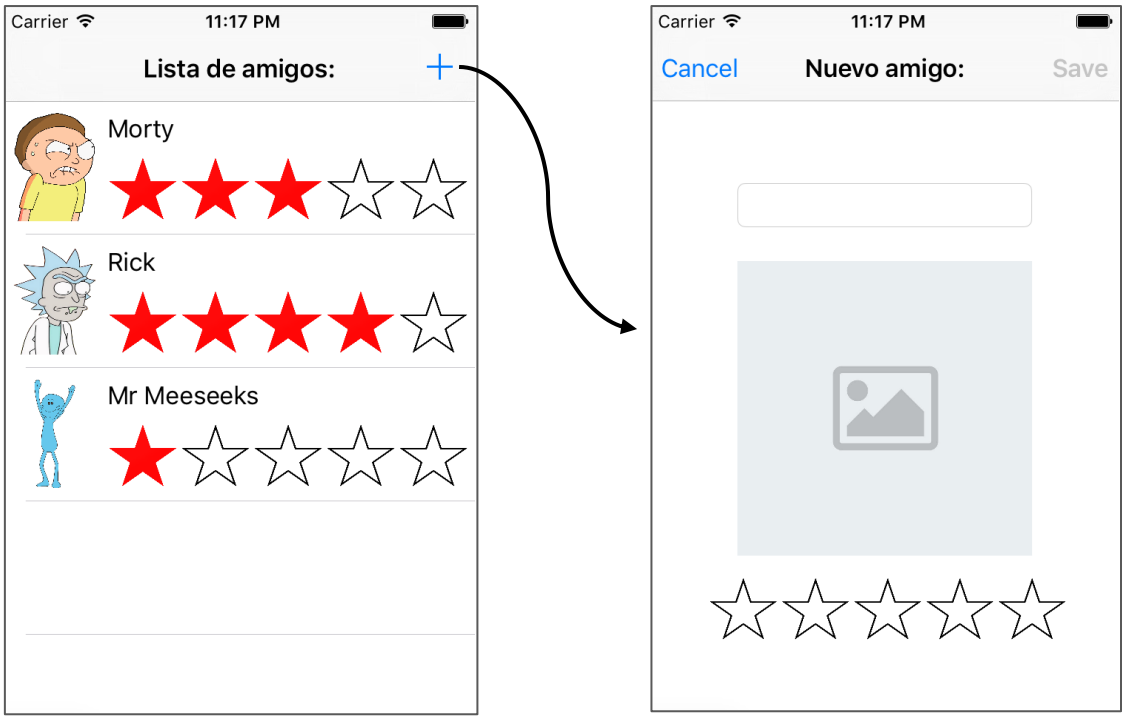
LÍNEAS DE PRODUCTOS SOFTWARE

Sesión 05. XCode, Swift, iOS: Edición, eliminación y persistencia de datos

Aitor García Luiz
Grado en Ingeniería Informática

La sesión 05 de prácticas es una extensión de lo realizado en la sesión 04. Antes de comenzar esta nueva práctica, hasta el momento hemos creado una aplicación con dos vistas, en una tenemos un listado de amigos y en la otra podemos introducir datos para añadir un nuevo amigo a la lista siempre y cuando el nombre no esté vacío:



Una vez esté todo listo podemos comenzar a seguir los pasos del guion e ir contestando a los ejercicios que se plantean:

Ejercicio 1.

¿Qué ocurre con el botón Cancel? ¿Por qué funciona si añadimos y no funciona si estamos editando?

Lo que ocurre con el botón Cancel es que funciona solamente cuando añadimos un nuevo usuario, pero no cuando se edita un usuario ya existente. Esto se debe al tipo de segue que usamos en cada caso, para añadir un usuario usamos uno de tipo modal mientras que en editar un usuario usamos uno de tipo show haciendo uso de una pila de navegación.

Ejercicio 2.

¿Qué ocurre con el botón Save? Identificad el punto de fallo en la app.

Al usar el botón Save lo que ocurre es que con la información del usuario que queríamos editar, se crea un nuevo usuario duplicando el original. Esto se debe a que la funcionalidad del botón Save es crear un usuario con esos datos, ya que aún no se refleja que se pueda guardar los datos al editar un usuario existente. Para solucionar esto se ha añadido los siguientes métodos del guion:

```
// MARK: - Unwind segue desde AmigoViewController
@IBAction func actualizaLista(sender: UIStoryboardSegue){
    let sourceViewController = sender.sourceViewController as! AmigoViewController
    if let idFilaSeleccionada = tableView.indexPathForSelectedRow {
        updateNuevoAmigo(sourceViewController.amigo!, idFila: idFilaSeleccionada)
    }else{
        addNuevoAmigo(sourceViewController.amigo!)
    }
}

func addNuevoAmigo(amigo: Amigo){
    amigos.append(amigo)
    let newIndexPath = NSIndexPath(forRow: amigos.count-1, inSection: 0)
    tableView.insertRowsAtIndexPaths([newIndexPath], withRowAnimation: .Bottom)
}

func updateNuevoAmigo(amigo: Amigo, idFila: NSIndexPath){
    amigos[idFila.row] = amigo
    tableView.reloadRowsAtIndexPaths([idFila], withRowAnimation: .Fade)
}
```

Con esta modificación, gracias a la variable tableView.indexPathForSelectedRow, podremos determinar si estamos en el modo de crear un nuevo usuario si el valor es 'nil' o en el modo de editar un usuario existente que en cuyo caso devolverá el valor de la fila que editamos.

Ejercicio 3.

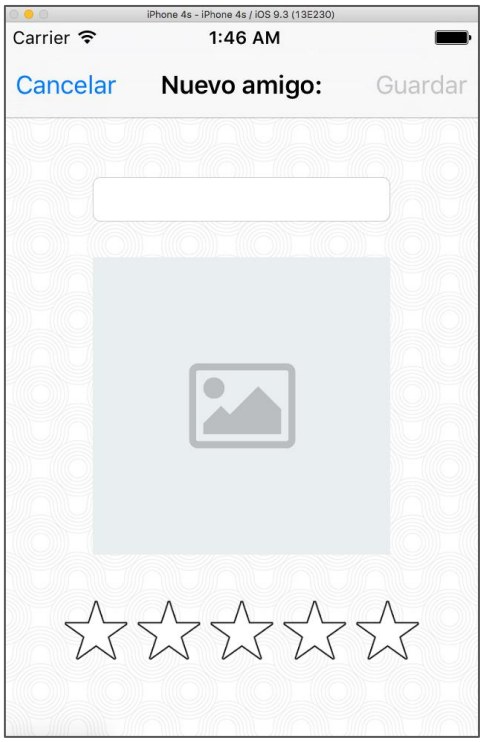
¿Os parece un poco complicado el proceso? ¿Qué sugerencias de mejoras metodológicas proponéis? Especificad y probad alguna mejora de identificación del tipo de navegación para que quede más claro el proceso de compartición de datos entre controladores de vista.

El proceso puede llegar a ser algo complicado si se realiza la primera vez, pero no pienso que la dificultad sea tan elevada como para buscar otra forma de realizarlo. La metodología usada me parece la correcta.

Ejercicio 4.

¿Edit, Cancel, Save y Done? Vamos a hacer que los botones aparezcan en español. Editar, Cancelar, Guardar y OK, respectivamente.

Para los botones Cancel y Save lo haremos usando el storyboard, seleccionando cada botón y yéndonos al inspector de atributos. Una vez en él solo tendremos que introducir el nombre deseado en el campo de texto 'Title'.



El resto de botones han sido modificados programáticamente tal y como se han creado, para ello, en la función *viewDidLoad()* de la clase correspondiente (en este caso AmigoTableViewController) además de la línea que añade el botón en sí, también incluiremos una que indique el título por defecto:

```
override func viewDidLoad() {
    super.viewDidLoad()

    //Selecciona la imagen "Background" y la pone de fondo en el listado de amigos
    self.view.backgroundColor = UIColor(patternImage: UIImage(named: "Background")!)

    navigationItem.leftBarButtonItem = editButtonItem()
    editButtonItem().title = "Editar"

    cargarDatos()
}
```

Como además se nos pide cambiar el botón 'Done' cuando se está en modo edición de la lista de amigos, debemos reflejar dichos cambios en la función '*setEditing*':

```
override func setEditing (editing:Bool, animated:Bool)    {
    super.setEditing(editing,animated:animated)
    if(self.editing){
        self.editButtonItem().title = "OK"
        self.editButtonItem().style = .Plain
    }else{
        self.editButtonItem().title = "Editar"
    }
}
```

Como se puede observar en la captura, mediante una sentencia if, comprobamos si se ha entrado en modo edición, de ser así, el título sería OK tal y como se pide, y además se ha añadido que siga un estilo 'Plain' aunque esto es meramente estético. Si no se incluyera la parte del *'else'*, una vez que se finalice el estado de edición y se vuelva al estado inicial, el título volvería a llamarse Edit, de ahí que se añada.

Por último, he traducido también al español un botón que no había sido tenido en cuenta hasta ahora: el botón 'Delete' que aparece una vez seleccionamos un usuario en el modo de edición.

Este botón rojo se ha renombrado gracias al siguiente método:

```
override func tableView(tableView: UITableView,
      titleForDeleteConfirmationButtonForRowAtIndexPath indexPath:
      NSIndexPath) -> String? {

            return "Eliminar"
}
```

Este método sobrescribe el nombre del botón de confirmación según el string que devuelva, en nuestro caso 'Eliminar'.

Los cambios de la aplicación quedarían reflejados de la siguiente forma: