

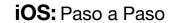
PASO A PASO 2:

UITableViewController - Interaction

Vamos a crear una aplicación que permita, a partir de una tabla, acceder a la información detallada de cada elemento. Para eso vamos a crear una aplicación de Contactos que consista de una tabla que liste todos los contactos del usuario, y en la que al seleccionar una celda (fila) de la tabla, se muestre una nueva pantalla con la información detallada.

iOS: Paso a Paso

- 1. Tomaremos como punto de partida la aplicación armada en *Paso a Paso 1*. La misma consta de <u>ContactsTableViewController</u>, un UITableViewController, que lista los contactos del usuario; <u>ContactTableViewCell</u>, una UITableViewCell custom; y <u>Person</u>, una clase que representa a un contacto de la Agenda.
- 2. En principio vamos a crear la pantalla de detalle que queremos mostrar cuando el usuario selecciona una celda (fila) de la tabla.
 - 2.1. File > NewFile > Cocoa Touch Class. Será subclase de UIViewController, y lo llamaremos <u>ContactDetailViewController</u>.
 - 2.2. En Main.storyboard agregamos un UIViewController, el cual debemos asociar a la clase creada anteriormente. En el storyboard, seleccionamos el nuevo ViewController y, en el *Identity Inspector*, indicamos que la "class" será **ContactDetailViewController**.
 - 2.3. Finalmente, configuramos la pantalla de acuerdo a nuestras necesidades, agregando los componentes necesarios, y generando los Outlets y Actions que hagan falta.
 - 2.4. A fin de poder mostrar el detalle de un contacto, la pantalla de Detalle debe obtener la información previamente. Cuando el usuario interactúa con la tabla, antes de ejecutar el segue, <u>ContactsTableViewController</u> debería indicarle a <u>ContactDetailViewController</u> cual es la Persona seleccionada. Por ello debemos crear un atributo público de tipo <u>Person</u>, que además será opcional porque hasta que no se configura no va a tener un valor.
 - 2.5. Finalmente, a partir de ese atributo de tipo Person, se configuran en el viewDidLoad los componentes de la vista.





3. Como vamos a acceder al detalle de un elemento, y luego vamos a querer volver atrás, debemos embeber la tabla en un UINavigationController. De este modo, luego vamos a poder regresar de la vista de detalle a la vista de tabla.

A continuación veremos distintas alternativas para poder acceder a la vista de detalle de un elemento a partir de la interacción con la celda (fila) de la tabla.

4. OPCIÓN 1: tableView:didSelectRowAt:indexPath sin segue

Así como nuestro UITableViewController cuenta con métodos para armar la tabla (numberOfRows, cellForRow, etc), también cuenta con métodos para interactuar con ella. Uno de esos métodos es func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath). Este método será invocado cuando el usuario interactúe con (haga tap en) una celda de la tabla, indicando como parámetro la tabla en sí misma y el índice (indexPath) de la celda afectada.

En este método debemos instanciar el controller a presentar, configurarlo con el elemento correspondiente y luego presentarlo.

- 4.1. Para presentar una pantalla (sin usar segues) debemos primero asignar un storyboardID a la pantalla que queremos presentar, es decir, a nuestro **ContactDetailViewController** en el panel *Identity Inspector* del storyboard. Le asignaremos "contactDetail".
- 4.2. Luego, en el método mencionado de <u>ContactsTableViewController</u> (didSelectRowAt:indexPath) debemos instanciar el controller a presentar: let controller = storyboard?.instantiateViewController(withIdentifier: "contactDetail") as? ContactDetailViewController
- 4.3. Luego vamos a configurarlo con el elemento correspondiente: controller?.person = persons[indexPath.row]
- 4.4. Finalmente, vamos a presentarlo utilizando el método *pushViewController* de la clase UINavigationController. Como el método recibe un UIViewController obligatorio como parámetro, debemos usar *if let* para validar que el controller se haya creado correctamente:

```
if let destination = controller {
  navigationController?.pushViewController(destination, animated: true)
}
```

5. OPCIÓN 2: segueFromCell

Así como podemos ejecutar segues automáticamente desde componentes de una vista (como un botón por ejemplo), lo mismo se puede hacer con celdas de una tabla.

5.1. Primero creamos un segue desde la celda prototipo de nuestro <u>ContactTableViewCell</u>. El destino será la pantalla de detalle, <u>ContactDetailViewController</u>.





5.2. Luego, en <u>ContactsTableViewController</u>, agregamos el método prepareForSegue. Vamos a obtener el controller destino del segue y, usando tableView.indexPathForSelectedRow obtenemos el elemento en la posición que corresponde a la fila seleccionada:

```
if let destination = segue.destination as? ContactDetailViewController,
    let selected = tableView.indexPathForSelectedRow {
         destination.person = persons[selected.row]
}
```

6. OPCION 3: tableView:didSelectRowAt:indexPath & segueFromController

Unificando las dos alternativas previas, es posible definir un segue que no se ejecute automáticamente, que vamos a ejecutar dinámicamente cuando sea necesario. De este modo, no necesitamos instanciar un controller desde el storyboard, y tampoco asociamos un segue a una celda prototipo.

- 6.1. Primero creamos un segue desde nuestro <u>ContactsTableViewController</u>. El destino será la pantalla de detalle, <u>ContactDetailViewController</u>.
- 6.2. Luego, en <u>ContactsTableViewController</u>, agregamos el método didSelectRowAt:indexPath. A partir del parámetro indexPath, vamos a obtener el elemento seleccionado, y lo guardamos en un atributo privado de la clase ("selectedPerson").
- 6.3. Además, como el segue no se inicia desde la celda, no se va a ejecutar automáticamente. Lo que queremos es ejecutar el segue manualmente selecciona ello, método cuando se una celda. Para en el didSelectRowAt:indexPath podemos forzar la ejecución del segue método performSeque(withIdentifier: el "sequeToDetail", sender: nil). Éste método ejecuta un segue a partir de un identificador, que en este caso es "segueToDetail". Para que funcione debemos ir al Storyboard, seleccionar el segue y configurar apropiadamente el identifier en el Attributes Inspector.
- 6.4. Finalmente, en <u>ContactsTableViewController</u>, agregamos el método prepareForSegue. Vamos a obtener el controller destino del segue y, usando el atributo privado donde obtuvimos el elemento seleccionado, lo configuramos.:

```
if let destination = segue.destination as? ContactDetailViewController{
    destination.person = selectedPerson
}
```