



POLITÉCNICA

ETSIT
UPM

dit
UPM

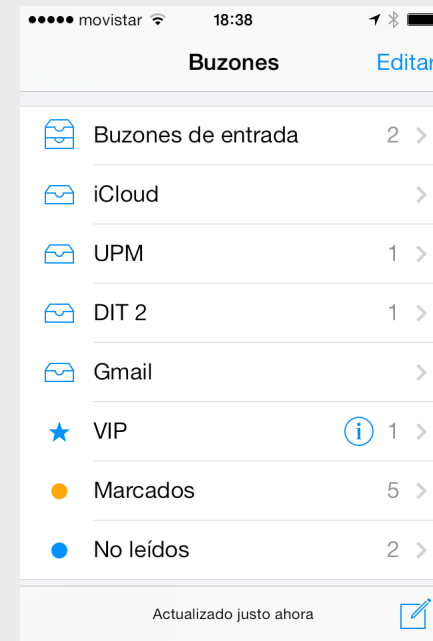
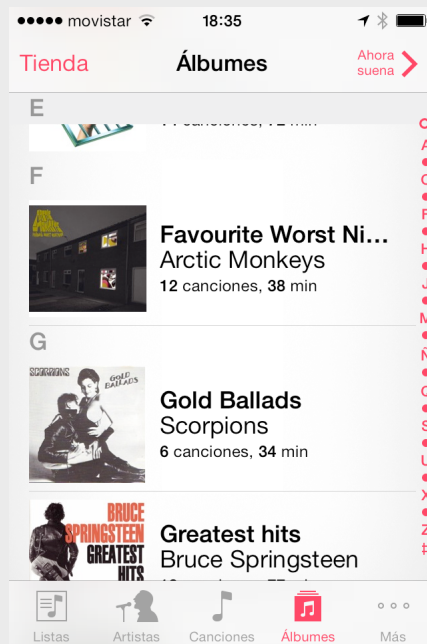
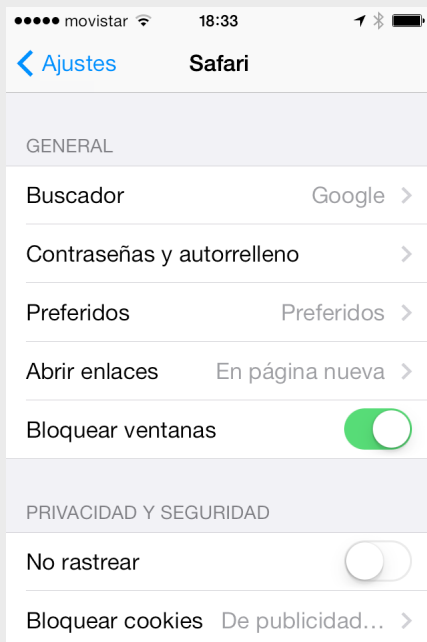
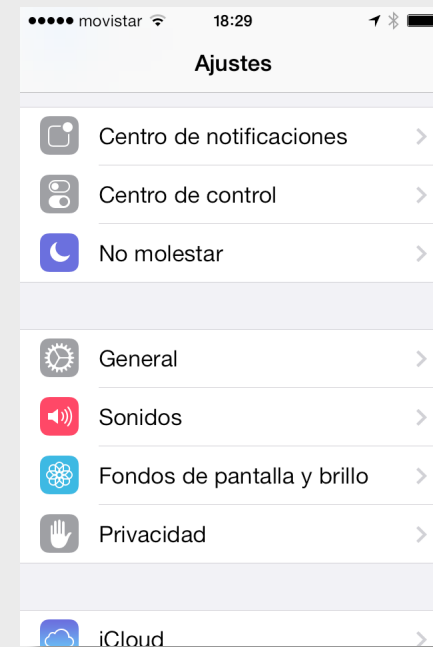
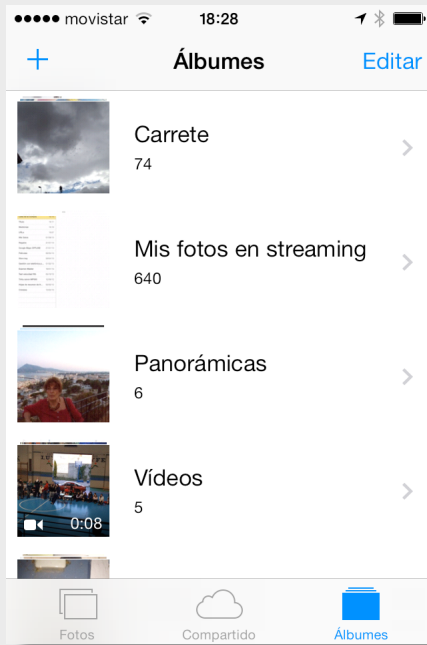
Desarrollo de Apps para iOS Table Views

IWEB 2019-2020
Santiago Pavón

ver: 2019.11.06

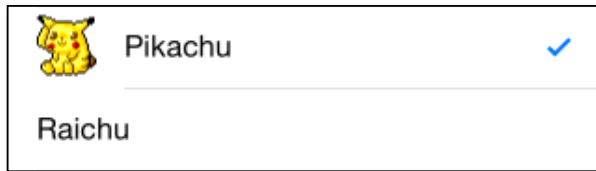
Características de las Tablas

- Mostrar la información en una tabla de una columna.
 - Cada fila es una celda.
- Apariencia:
 - Pueden tener una cabecera y pie de tabla.
 - Se pueden agrupar las celdas en secciones.
 - Cada sección: cabecera, celdas y pie.
 - Existen varios estilos predefinidos de celdas, o podemos crear celdas personalizadas.
- La tabla es un objeto **UITableView** y la celda es un objeto **UITableViewCell**.
 - **UITableView** deriva de **UIScrollView**.
 - Uso muy eficiente de las celdas reutilizando las celdas no visibles.
- Las tablas pueden ser estáticas (solo en **UITableViewController**) o dinámicas.
- Protocolos:
 - Fuente de datos: **UITableViewDataSource**.
 - Apariencia y comportamiento: **UITableViewDelegate**.



Estilos de Celdas Predefinidos

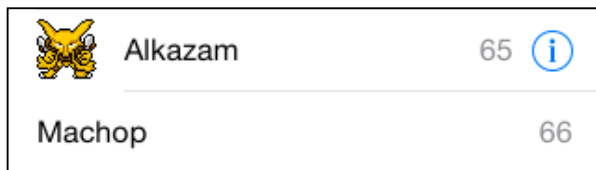
- **Basic** (`UITableViewCellStyle.default`)



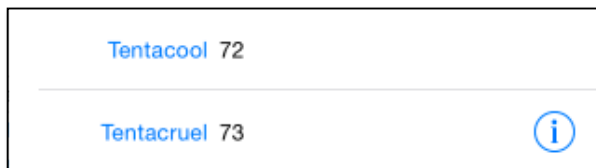
- **Subtitle** (`UITableViewCellStyle.subtitle`)



- **Right Detail** (`UITableViewCellStyle.value1`)



- **Left Detail** (`UITableViewCellStyle.value2`)



imageView: UIImageView?

textLabel: UILabel?



Feraligatr

160



detailTextLabel: UILabel?

accessoryType: UITableViewCellAccessoryType

Propiedades de las celdas

- **imageView**

- Es una **UIImageView?** que podemos personalizar cambiando sus propiedades
 - **image, highlightedImage, ...**

- **textLabel**

- Es una **UILabel?** que podemos personalizar cambiando sus propiedades
 - **text, font, ...**

- **detailTextLabel**

- Es una **UILabel?** que podemos personalizar cambiando sus propiedades
 - **text, font, ...**

- **contentView**

- Es la **UIView** donde se muestra el contenido de la celda.
- Podemos añadir nuestras propias subviews para personalizar las celdas.

- **accessoryType**

- Es el tipo de accesorio a mostrar en la celda.

- ...

Crear un objeto UITableView

- Arrastrar un objeto Table View desde la librería de objetos.
 - Configurar la tabla en el inspector.
 - Contenido: celdas creadas usando prototipos.
 - Las celdas estáticas solo se soportan en UITableViewController.
 - Número de prototipos de celdas que queremos usar.
 - Estilo de la tabla: plana, agrupada.
 - ...
 - Asignar su objeto delegado y data source.
 - ...

Crear Tabla con IB: Ready | Today at 17:19

Crear Tabla con IB > C > M > M > V > V > View > Table View

Crear Tabla con IB

- Crear Tabla con IB
 - AppDelegate.swift
 - ViewController.swift
 - Main.storyboard
 - Assets.xcassets
 - LaunchScreen.storyboard
 - Info.plist
 - Products

View Controller Scene

- View Controller
 - Top Layout Guide
 - Bottom Layout G...
 - View
 - Table View
 - First Responder
 - Exit
 - Storyboard Entry Poi...

Table View

Content: Dynamic Prototypes

Prototype Cells: 0

Style: Plain

Separator: Default

+ Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

+ Text: Default

+ Background: Default

+ Tracking: Default

Scroll View

Style: Default

Scroll Indicat... Shows Horizontal Indicat...

Horizontal Stack View - Arranges views linearly.

Vertical Stack View - Arranges views linearly.

Table View - Displays data in a list of plain, sectioned, or grouped rows.

Table View Cell - Defines the

Tabla plana.

Su contenido se creará usando los prototipos de celda que diseñe.

View as: iPhone 6s (wC hR)

Crear Tabla con IB: Ready | Today at 17:21

Crear T...la con IB > View > Table View

Crear Tabla con IB

- Crear Tabla con IB
 - AppDelegate.swift
 - ViewController.swift
 - Main.storyboard
 - Assets.xcassets
 - LaunchScreen.storyboard
 - Info.plist
 - Products
- View Controller Scene
 - View Controller
 - Top Layout Guide
 - Bottom Layout Guide
 - View
 - Table View
 - Table View Cell
 - Content View
 - First Responder

Usar un prototipo de celda.

Table View

Content: Dynamic Prototypes

Prototype Cells: 1

Style: Plain

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

Scroll View

Style: Default

Scroll Indicat... Shows Horizontal Indicat...

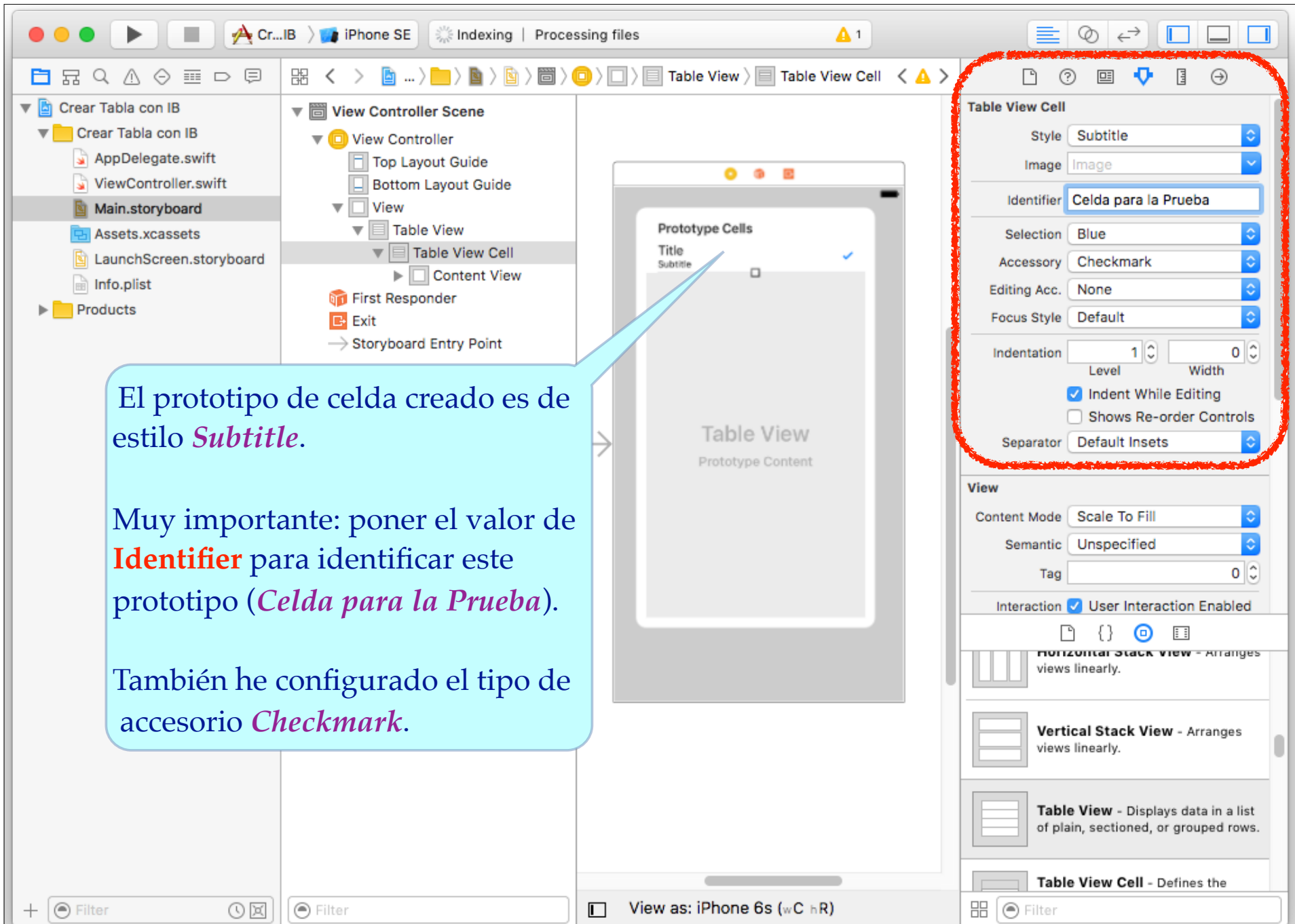
Horizontal Stack View - Arranges views linearly.

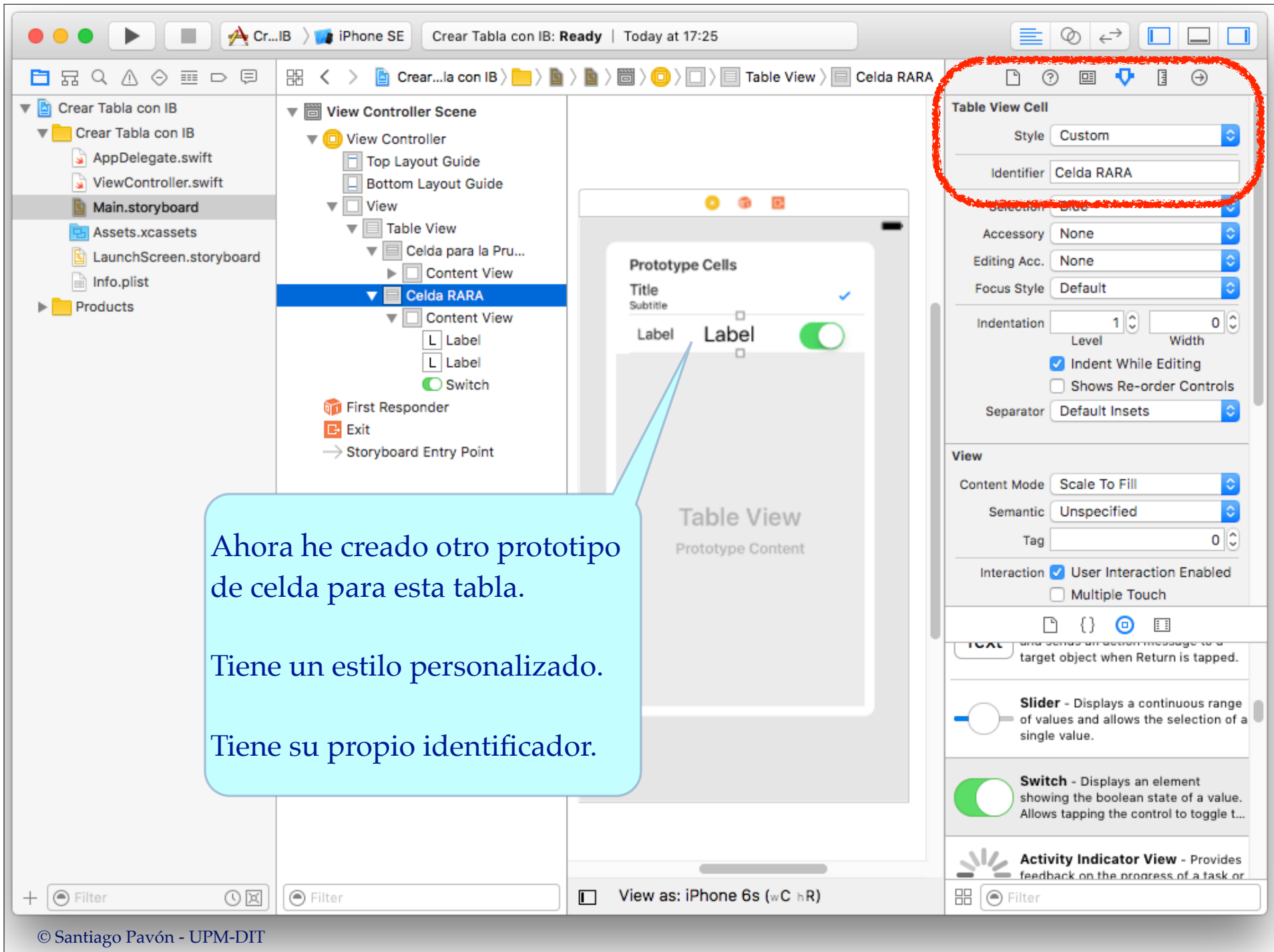
Vertical Stack View - Arranges views linearly.

Table View - Displays data in a list of plain, sectioned, or grouped rows.

Table View Cell - Defines the

View as: iPhone 6s (wC hR)





Ahora he creado otro prototipo de celda para esta tabla.

Tiene un estilo personalizado.

Tiene su propio identificador.

The screenshot shows the Xcode interface with a custom class `MyTableViewCell` defined in `MyTableViewCell.swift`. The class inherits from `UITableViewCell` and has three weak outlets: `label1`, `label2`, and `swtch`. The right-hand inspector shows the class set to `MyTableViewCell` in the `Custom Class` section. The left-hand inspector shows the `Outlets` section with `label1`, `label2`, and `swtch` connected to their respective subviews in the prototype cell. A light blue text box explains the purpose of these outlets.

```
import UIKit

class MyTableViewCell: UITableViewCell {

    @IBOutlet weak var label1: UILabel!
    @IBOutlet weak var label2: UILabel!
    @IBOutlet weak var swtch: UISwitch!

    override func awakeFromNib() {
```

Custom Class
Class: MyTableViewCell
Module: Current - Crear_Tab...

Identity
Restoration ID

User Defined Runtime Attributes

Key Path	Type	Value
----------	------	-------

Outlets

- label1: * Label1
- label2: * Label2
- swtch: * Swtch

Triggered Segues

- selection
- accessory action

Outlet Collections

- gestureRecognizers

Referencing Outlets

- New Referencing Outlet

Referencing Outlet Collections

- New Referencing Outlet Collection

View as: iPhone 6s (wC hR)

```
@IBOutlet weak var label1: UILabel!  
@IBOutlet weak var label2: UILabel!  
@IBOutlet weak var swtch: UISwitch!
```

Para acceder a las subviews añadidas en este prototipo de celda, creamos una subclase de `UITableViewCell` para la celda prototipo. Hay que cambiar la clase de la celda en el inspector.

Crearé en esta clase outlets que apunten a las subview del prototipo.

Cuando tenga un objeto de esta clase, los outlets apuntarán a las subviews.

Obtener los Datos a Mostrar

- La Table View **no posee** los datos.
 - Se los proporciona otro objeto que actúa como Data Source.
- Usa el protocolo **UITableViewDataSource** para obtener los datos.
 - Las tablas tienen una propiedad llamada **dataSource** que debe apuntar a un objeto conforme a este protocolo.
 - El objeto dataSource **no se retiene**.
 - Se está suponiendo que el objeto data source tendrá una vida superior a la de la tabla.
 - La tabla le pregunta a su dataSource:
 - cuántas secciones hay.
 - cuántas filas hay en cada sección.
 - y le pide que le proporcione las celdas a colocar en cada fila de cada sección.
 - ...

UITableViewDataSource

- Configurar la Table View

```
func tableView(_ tableView: UITableView,  
    numberOfRowsInSection section: Int) -> Int
```

```
func tableView(_ tableView: UITableView,  
    cellForRowAt indexPath: IndexPath) -> UITableViewCell
```

```
optional func numberOfSections(in tableView: UITableView) -> Int
```

```
optional func tableView(_ tableView: UITableView,  
    titleForHeaderInSection section: Int) -> String?
```

```
optional func tableView(_ tableView: UITableView,  
    titleForFooterInSection section: Int) -> String?
```

- Índice

```
optional func sectionIndexTitles(for tableView: UITableView)  
    -> [String]?
```

```
optional func tableView(_ tableView: UITableView,  
    sectionForSectionIndexTitle title: String,  
    at index: Int) -> Int
```

- Añadir o borrar filas

```
optional func tableView(_ tableView: UITableView,  
    commit editingStyle: UITableViewCellStyle,  
    forRowAt indexPath: IndexPath)
```

```
optional func tableView(_ tableView: UITableView,  
    canEditRowAt indexPath: IndexPath) -> Bool
```

- Reordenar las filas

```
optional func tableView(_ tableView: UITableView,  
    canMoveRowAt indexPath: IndexPath) -> Bool
```

```
optional func tableView(_ tableView: UITableView,  
    moveRowAt sourceIndexPath: IndexPath,  
    to destinationIndexPath: IndexPath)
```

IndexPath

- Las instancias de la estructura **IndexPath** identifican la posición de las celdas dentro de una tabla.
 - Es el número de sección y fila de una celda de la tabla.
- Tiene dos propiedades:
 - **row** y **section**.
 - devuelven un entero (**Int**).

Reutilización de celdas

- Crear y destruir celdas es costoso.
 - Las celdas no visibles, se guardan en una cola para reutilizarlas.
- Si se necesita una celda nueva, se saca de la cola de reutilización con:

```
func dequeueReusableCell(withIdentifier identifier: String,  
                          for indexPath: IndexPath) -> UITableViewCell
```

- Si la cola está vacía, construye una nueva celda copiando el prototipo.

Identificar tipo de celda o prototipo

```
override func tableView(_ tableView: UITableView,
                        cellForRowAt indexPath: IndexPath)
    -> UITableViewCell {

    // Sacar una celda de la cola para reutilizar
    let cell = tableView.dequeueReusableCell(withIdentifier: "My Cell Id",
                                           for: indexPath)
                                           as! MyTableViewCell

    // Mirar el indexPath para saber que debo mostrar en esta celda
    let row = indexPath.row
    let section = indexPath.section

    // Configurar la celda
    cell.imageView?.image = almacen.getPhoto(row)
    cell.imageView?.highlightedImage = almacen.getHPhoto(row)
    cell.textLabel?.text = almacen.getName(row)
    cell.accessoryType = .checkmark

    // Devolver la celda
    return cell
}
```

UITableView - Actualizaciones

```
func reloadData()

func insertSections(_ sections: IndexSet,
                  with animation: UITableViewRowAnimation)

func deleteSections(_ sections: NSIndexSet,
                  with animation: UITableViewRowAnimation)

func reloadSections(_ sections: IndexSet,
                  with animation: UITableViewRowAnimation)

func insertRows(at indexPaths: [IndexPath],
               with animation: UITableViewRowAnimation)

func deleteRows(at indexPaths: [IndexPath],
               with animation: UITableViewRowAnimation)

func reloadRows(at indexPaths: [AnyObject],
               with animation: UITableViewRowAnimation)

. . .
```

El delegado de la tabla

- Los objetos **UITableView** tienen una propiedad llamada **delegate**.
 - El objeto delegado no se retiene.
 - Se está suponiendo que el objeto delegado tendrá una vida superior a la de la tabla.
- Debe apuntar a un objeto que sea conforme con el protocolo **UITableViewDelegate**.
- El objeto delegado maneja la selección de celdas, ayuda en el borrado y la reordenación de las celdas, configura las cabeceras y pies de las secciones, observa las acciones realizadas sobre la tabla, controla cómo se muestra la tabla, ...

UITableViewDelegate

- Configurar las filas de la tabla:

```
tableView(UITableView, heightForRowAt: IndexPath)
tableView(UITableView, estimatedHeightForRowAt: IndexPath)
tableView(UITableView, indentationLevelForRowAt: IndexPath)
tableView(UITableView, willDisplay: UITableViewCell,
           forRowAt: IndexPath)
```

- Manejar los accesorios:

```
tableView(UITableView, editActionsForRowAt: IndexPath)
tableView(UITableView, accessoryButtonTappedForRowWith: IndexPath)
```

- Manejar la selección de celdas:

```
tableView(UITableView, willSelectRowAt: IndexPath)
tableView(UITableView, didSelectRowAt: IndexPath)
tableView(UITableView, willDeselectRowAt: IndexPath)
tableView(UITableView, didDeselectRowAt: IndexPath)
```

- Modificar la cabecera y pie de las secciones:

```
tableView(UITableView, viewForHeaderInSection: Int)
tableView(UITableView, viewForFooterInSection: Int)
tableView(UITableView, heightForHeaderInSection: Int)
tableView(UITableView, estimatedHeightForHeaderInSection: Int)
tableView(UITableView, heightForFooterInSection: Int)
tableView(UITableView, estimatedHeightForFooterInSection: Int)
tableView(UITableView, willDisplayHeaderView: UIView, forSection: Int)
tableView(UITableView, willDisplayFooterView: UIView, forSection: Int)
```

- Edición de las filas de la tabla:

```
tableView(UITableView, willBeginEditingRowAt: IndexPath)
tableView(UITableView, didEndEditingRowAt: IndexPath?)
tableView(UITableView, editingStyleForRowAt: IndexPath)
tableView(UITableView,
           titleForDeleteConfirmationButtonForRowAt: IndexPath)
tableView(UITableView, shouldIndentWhileEditingRowAt: IndexPath)
```

- Reordenar las filas de la tabla:

```
tableView(UITableView, targetIndexPathForMoveFromRowAt: IndexPath,
           toProposedIndexPath: IndexPath)
```

- Seguimiento del borrado de views:

```
tableView(UITableView, didEndDisplaying: UITableViewCell,  
          forRowAt: IndexPath)  
tableView(UITableView, didEndDisplayingHeaderView: UIView, forSection: Int)  
tableView(UITableView, didEndDisplayingFooterView: UIView, forSection: Int)
```

- Gestionar el copiado y pegado del contenido de las filas:

```
tableView(UITableView, shouldShowMenuForRowAt: IndexPath)  
tableView(UITableView, canPerformAction: Selector, forRowAt: IndexPath,  
          withSender: Any?)  
tableView(UITableView, performAction: Selector, forRowAt: IndexPath,  
          withSender: Any?)
```

- Gestionar el resaltado de las filas:

```
tableView(UITableView, shouldHighlightRowAt: IndexPath)  
tableView(UITableView, didHighlightRowAt: IndexPath)  
tableView(UITableView, didUnhighlightRowAt: IndexPath)
```

- Gestionar el foco:

```
tableView(UITableView, canFocusRowAt: IndexPath)  
tableView(UITableView, shouldUpdateFocusIn: UITableViewFocusUpdateContext)  
tableView(UITableView, didUpdateFocusIn: UITableViewFocusUpdateContext,  
          with: UIFocusAnimationCoordinator)  
indexPathForPreferredFocusedView(in: UITableView)
```

- Gestión de acciones Swipe: **Nuevo en iOS 11**

```
tableView(_: UITableView,  
    leadingSwipeActionsConfigurationForRowAt: IndexPath)  
    -> UISwipeActionsConfiguration?
```

```
tableView(_: UITableView,  
    trailingSwipeActionsConfigurationForRowAt indexPath: IndexPath)  
    -> UISwipeActionsConfiguration?
```

- etc...

Al Seleccionar una Fila ...

- Podemos crear un segue que se dispare cuando se selecciona una fila de la tabla.
- Para configurar el segue:
 - Opción 1:
 - Asignaremos un identificador al segue.
 - Adaptaremos el método **prepare(for segue:, sender:)** para:
 - Consultar qué celda está seleccionada para saber dónde hemos pulsado.
 - Configurar el VC destino del segue.
 - Opción 2:
 - Crear una acción **@IBSegueAction**.
 - Así no es necesario ni sobrescribir el método `prepare(for segue:, segue)`, ni asignar un identificador al segue.
 - Consultar detalles en el tema sobre Segues.

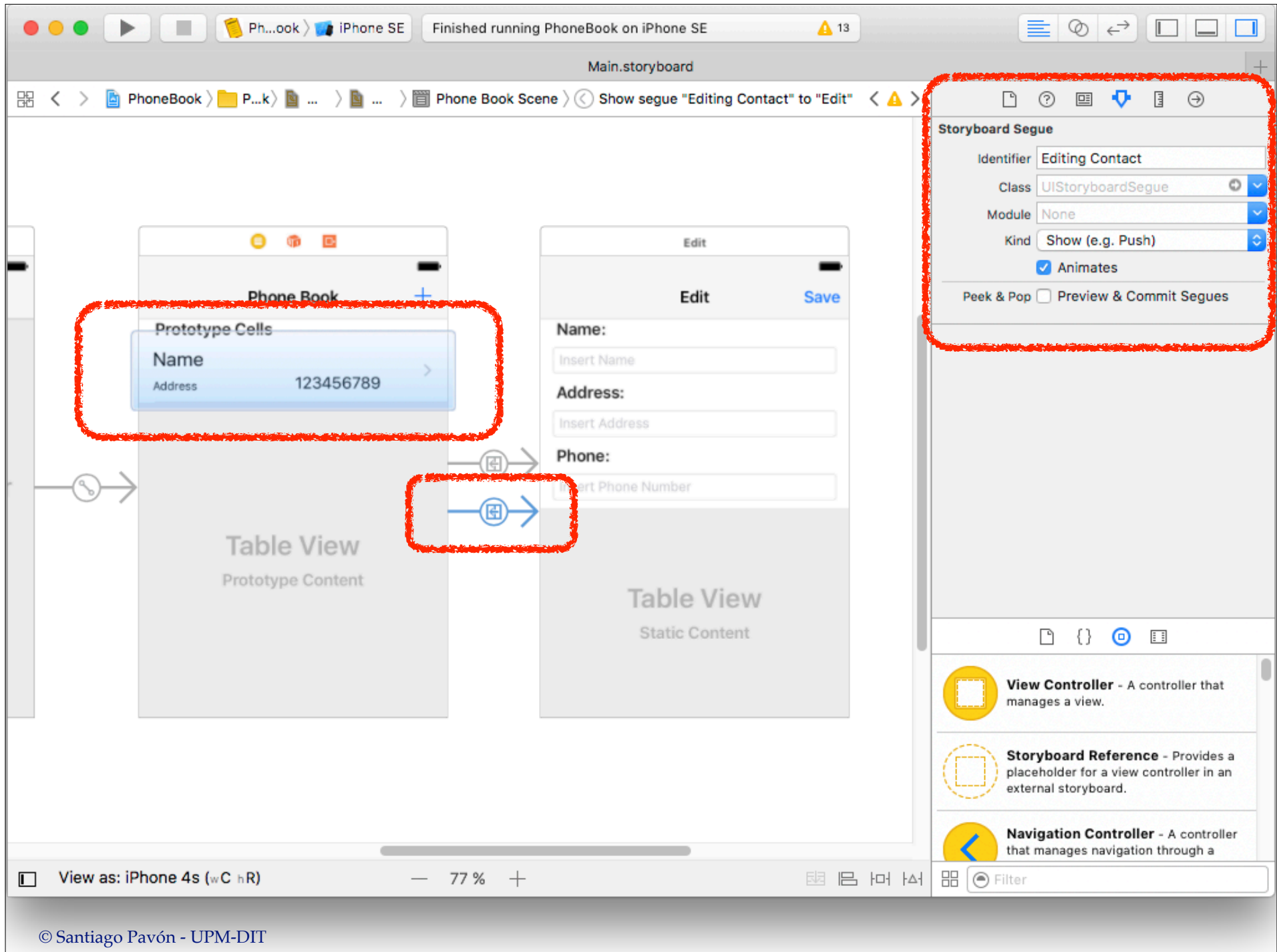
Ejemplo

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
```

```
    if segue.identifier == "Editing Contact" {  
        if let ecvc = segue.destination as? EditContactTableVC,  
           let ip = tableView.indexPathForSelectedRow {  
            ecvc.contact = contactBook.contacts[ip.row]  
        }  
    }
```

```
    } else if segue.identifier == "Adding Contact" {  
        if let ecvc = segue.destination as? EditContactTableVC {  
            let newContact = Contact(name: "", address: "", phone: "")  
            contactBook.contacts.insert(newContact, at: 0)  
            let ip = IndexPath(row: 0, section: 0)  
            tableView.insertRows(at: [ip], with: .automatic)  
            tableView.selectRow(at: ip, animated: true, scrollPosition: .top)  
            ecvc.contact = newContact  
        }  
    }
```

```
}
```



Accesorios



- Controles accesorios usados por las celdas

`UITableViewCellAccessoryType.none`

`UITableViewCellAccessoryType.checkmark`

`UITableViewCellAccessoryType.detailButton`

`UITableViewCellAccessoryType.disclosureIndicator`

`UITableViewCellAccessoryType.detailDisclosureButton`

- Desde los accesorios de tipo **Detail Disclosure Button** y **Detail Button** de las celdas también se pueden lanzar segues.
 - La celda con el accesorio hace el papel de sender del segue.

Tamaño de las Celdas

- La altura con la que pintan todas las celdas puede:
 - Configurarse desde el inspector del **Interface Builder**.
 - o indicarse **programáticamente** asignando un valor a la propiedad **tableView.rowHeight**.
- Todas las celdas se pintarán con la altura configurada, independientemente de cual sea la altura necesaria para pintar su contenido.

- Si cada celda tiene una altura distinta, se puede indicar cual es la altura de cada celda sobrescribiendo el método:

```
func tableView(_ tableView: UITableView,  
              heightForRowAt indexPath: IndexPath)  
              -> CGFloat
```

- Cuando se carga una tabla, este método se llama para todas las filas de la tabla.

- Este cálculo inicial de alturas puede tardar mucho si la tabla tiene muchas celdas.
- Para mejorar la experiencia de usuario al cargar tablas grandes, podemos estimar un valor para la altura de las celdas.

- Estamos retrasando el cálculo de las alturas reales hasta el momento de hacer un scroll.
- El valor estimado de altura se debe asignar a la propiedad **estimatedRowHeight** de las Table Views.
- Para proporcionar una estimación individual para cada una de las celdas de la tabla podemos sobrescribir en el delegado de la tabla el método:

```
func tableView(_ tableView: UITableView,  
              estimatedHeightForRowAt indexPath: IndexPath)  
              -> CGFloat
```

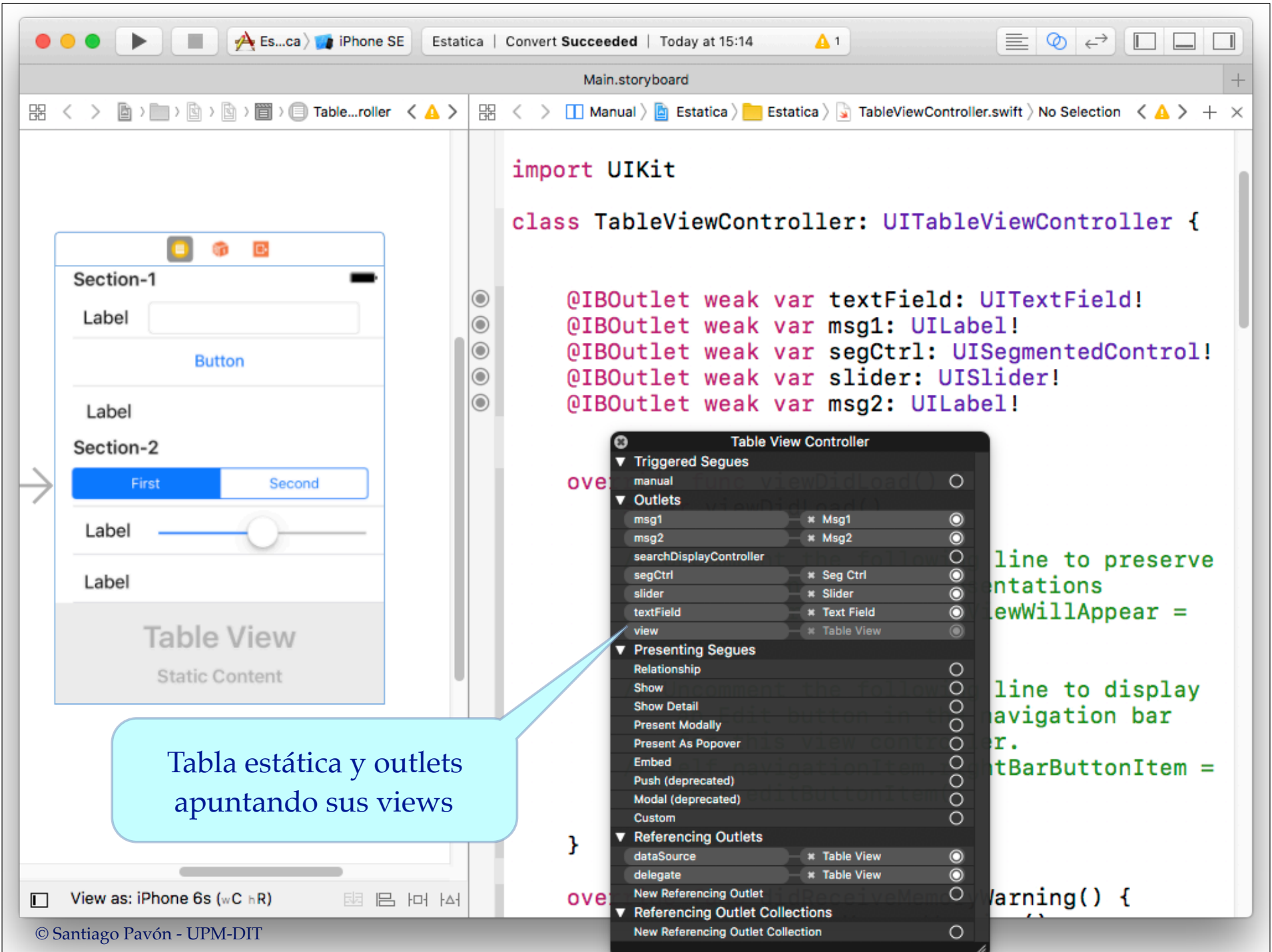
- El cálculo del tamaño individual de cada celda puede hacerse automáticamente usando autolayout.
 - Hay que asegurarse de que las restricciones de autolayout añadidas a la celda (en su prototipo) determinan cuál es su altura.
 - Las restricciones añadidas junto con el tamaño intrínseco de las subviews de cada celda determinan la altura de la celda.
 - El ancho lo determina la tableView.
 - Hay que añadir programáticamente las siguientes sentencias:
 - Decir a la TableView que las celdas calculan su propio tamaño:
`tableView.rowHeight = UITableView.automaticDimension`
 - Y proporcionar un tamaño estimado de celda distinto de cero.
 - Este valor debe ser cercano al real para evitar saltos al hacer scroll.
`tableView.estimatedRowHeight = 100`
- De esta forma, ya no hay que usar los métodos que tiene UITableView para calcular el tamaño real de cada celda, o estimarlo en función de su IndexPath.

UITableViewController

- Es una clase derivada de **UIViewController**
 - que tiene una **UITableView** como su view.
 - La propiedad **view** y **tableView** apuntan al mismo objeto: la tabla interna.
 - El propio objeto **UITableViewController** es el **data source** y **delegado** de su tabla interna.
- Proporciona facilidades:
 - carga inicial de datos
 - gestión del teclado al editar.
 - facilidades de edición.
 - etc.
- Con Interface Builder podemos añadir objetos **UITableViewController** al storyboard arrastrándolos desde la librería de objetos.
 - Crearemos para ellos subclases personalizadas que deriven de **UITableViewController**.
 - En el Inspector reasignaremos las clases.
- Y por supuesto, también se pueden crear programáticamente.

Celdas Prototipo y Estáticas

- Las tablas pueden crearse para que usen
 - Celdas **prototipo**
 - las celdas de la tabla se crean copiando las celdas prototipo.
 - Cuando se necesitan más celdas, pueden reutilizarse celdas ya usadas y no visibles.
 - Celdas **estáticas** (*Solo para UITableViewController*)
 - La tabla sólo tiene las celdas estáticas que hemos creado con IB.
 - No se usan los prototipos para crear más celdas.
 - Cuando se usan celdas estáticas no tiene sentido usar el protocolo Data Source.
 - aunque podría usarse, evitando conflictos entre el diseño estático y la información proporcionada por el data source.
- Para acceder a las subviews añadidas a las celdas, tanto en celdas estáticas y prototipos de estilo personalizado, se recomienda:
 - crear nuestras propias subclases de celda, derivadas de UITableViewCell, y crear outlets apuntando a las subviews.



```
import UIKit

class UITableViewController: UITableViewController {

    @IBOutlet weak var textField: UITextField!
    @IBOutlet weak var msg1: UILabel!
    @IBOutlet weak var segCtrl: UISegmentedControl!
    @IBOutlet weak var slider: UISlider!
    @IBOutlet weak var msg2: UILabel!
```

Table View Controller

- Triggered Segues
 - manual
- Outlets
 - msg1 * Msg1
 - msg2 * Msg2
 - searchDisplayController
 - segCtrl * Seg Ctrl
 - slider * Slider
 - textField * Text Field
 - view * Table View
- Presenting Segues
 - Relationship
 - Show
 - Show Detail
 - Present Modally
 - Present As Popover
 - Embed
 - Push (deprecated)
 - Modal (deprecated)
 - Custom
- Referencing Outlets
 - dataSource * Table View
 - delegate * Table View
- Referencing Outlet Collections
 - New Referencing Outlet
 - New Referencing Outlet Collection

Tabla estática y outlets apuntando sus views

line to preserve
ntations
ewWillAppear =

line to display
avigation bar
r.
tBarButtonItem =

arning() {