PASO A PASO 1: UITableViewController

Vamos a crear una aplicación que muestre, en una tabla, información de distintas personas. Básicamente una agenda o lista de contactos.

1. Comenzamos creando un nuevo proyecto, de tipo SingleView Application.

2. Como vamos a utilizar una tabla, creamos un ViewController que herede de la clase UITableViewController.

   2.1. File > NewFile, y crearemos un nuevo *Cocoa Touch Class*. Será subclase de `UITableViewController`, y lo llamaremos ***ContactsTableViewController***.

   2.2. Automáticamente se generará la clase en el archivo (por haber usado *Cocoa Touch Class*), junto con muchos métodos que pueden (o no) resultar de utilidad. En principio sólo necesitaremos:

      i. viewDidLoad()

      ii. numberOfSections(in tableView: UITableView) -> Int

      iii. tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int

      iv. tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell

   2.3. En Main.storyboard agregamos un `UITableViewController`, y lo marcamos como punto de entrada (o primer pantalla) de la aplicación: se puede hacer arrastrando la flecha que apunta al controller default, o marcando la opción "*Is Initial View Controller*" en el *Attributes Inspector*.

   2.4. Como no vamos a utilizarlo, eliminamos el controller default que se generó automáticamente al crear el proyecto. Lo mismo podemos hacer con la clase asociada (el archivo *ViewController.swift*).

   2.5. Finalmente, debemos asociar la nueva pantalla que creamos en el Storyboard, con la clase creada para manejarla. En el storyboard, seleccionamos el nuevo TableViewController y, en el *Identity Inspector*, indicamos que la "*class*" será ***ContactsTableViewController***.

3. A continuación, debemos configurar la celda que se utilizará en la tabla. A fin de tener mayores posibilidades de configuración, configuraremos nuestra propia celda "*custom*".

    3.1. File > NewFile, y crearemos un nuevo *Cocoa Touch Class*. Será subclase de `UITableViewCell`, y lo llamaremos ***ContactTableViewCell***. Igual que con el controller, se creará la clase con algunos métodos que pueden resultar de utilidad. Si no vamos a usarlos, podemos borrarlos directamente.

    3.2. En el storyboard, la tabla creada comienza con una celda ya definida. Ésta es la celda que vamos a configurar según nuestras necesidades. Primero debemos asociarla a la clase creada en el paso anterior, indicando su "*class*" en el *Identity Inspector* como ***ContactTableViewCell***.

    3.3. La celda se crea por defecto con el estilo "*custom*", pero podemos usar algunos de los estilos provistos (basic, subtitle, etc) si son suficientes para lo que tenemos que hacer.

    3.4. Agregamos en la celda los elementos que queremos mostrar para cada elemento. En este ejemplo agregaremos tres `UILabel` y dos `UIImageView`. La altura de la celda tambien puede configurarse en el panel *Size Inspector* de la `UITableView` seteando el atributo *Row Height*.

    3.5. Luego debemos crear los @IBOutlets correspondientes para cada componente de la celda, en la clase que creamos para tal fin: ***ContactTableViewCell***. Si al abrir el asistente no aparece el código de dicha clase, en la barra superior donde dice "*automatic*" se puede indicar "*manual*" y especificar el archivo que queremos ver.

4. A partir de este punto vamos a asumir que trabajamos con una clase Persona, y que nuestro ***ContactsTableViewController*** tiene un atributo que es un array de objetos de tipo Persona. De igual forma, pueden ser Productos, o cualquier otra cosa que requiera nuestra solución.

5. Cada celda que se muestre en la tabla, estará asociada a un elemento de nuestro array. Por lo tanto, en este caso particular, cada celda representará a una persona (o a un producto; a un elemento de nuestro array de elementos). Procedemos entonces a completar los métodos de ***ContactsTableViewController***.

    5.1. El método `numberOfSections(in tableView: UITableView) -> Int` debe devolver un valor entero, que indicará cuántas secciones conforman la tabla. En principio todos los elementos de la tabla se van a mostrar agrupados dentro de una misma sección. Por lo tanto, el método debe constar de una única línea: `return 1`. Las secciones se utilizan para separar filas que son afines entre sí, y que se diferencian de las demás (un buen ejemplo de tal uso

es la app de Contactos nativa de iPhone, en la que cada sección es una letra del alfabeto, agrupando contactos).

5.2. El método `tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int` debe devolver un valor entero, que indicará cuántas filas conforman una sección determinada (indicada con su posición en el parámetro recibido). Como queremos mostrar las personas contenidas en nuestro array, vamos a tener que mostrar tantas filas como elementos haya en el array. Por lo tanto, el método debe constar de una única línea: `return array.count`.

5.3. El método `tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell` siempre debe devolver el objeto de tipo `UITableViewCell` que se mostrará en la posición indicada por el parámetro `indexPath`.

i. `let cell = tableView.dequeueReusableCell(withIdentifier: "contact", for: indexPath)`
Con esta línea obtenemos una celda: se le pide a la tabla una celda reutilizable, para usar en la posición indicada por el indexPath. El método "*dequeueReusableCell*" se encarga de devolver una celda ya creada que no esté siendo usada, o crear una nueva celda si no hay celdas disponibles. El *identifier* que se indica como parámetro debe coincidir con el "*reuseIdentifier*" definido para la celda en el storyboard en el panel *Attributes Inspector*.

ii. `let person = peopleArray[indexPath.row]`
Con esta línea obtenemos el elemento asociado a la celda que estamos construyendo. Si estamos configurando la celda en posición 7, obtendremos el elemento en la 7ma posición del array.

iii.
```
if let contactCell = cell as? ContactTableViewCell {
    contactCell.setup(person)
}
```
En esta instrucción, usando el "`as?`", indicamos que si la celda que obtuvimos con *dequeueReusableCell* es en realidad de tipo ***ContactTableViewCell***, vamos a proceder a configurarla con el elemento obtenido del array. Si la celda no era del tipo indicado, la asignación falla y el código dentro del `if let` no se ejecuta.

6. Como queremos configurar la celda a partir del elemento, debemos definir el método `setup` en la clase ***ContactTableViewCell***: `func setup(_ person: Person)`. Este método se encargará de setear los elementos de la celda en base a la información del elemento recibido por parámetro (por ejemplo, un label que se configura con el nombre de la persona, una imagen que se configura a partir de un avatar, el fondo que depende de algún otro valor, etc).

7. Luego, si es necesario agregar información a la tabla, sólo es necesario agregar más elementos al array. Si hace falta modificar la información que se muestra, sólo es necesario modificar el código de la clase ***ContactTableViewCell***. De este modo, no hace falta modificar el código de ***ContactsTableViewController***.