PASO A PASO 5:
UITableViewController - Sections

Vamos a crear una aplicación conformada por una tabla, que va a listar información de provincias agrupadas por país. Es decir, la tabla contará con secciones, donde cada sección representa un país; y cada fila de la sección representará a una provincia del país.

1. En principio vamos a crear dos clases de nuestro modelo: País y Provincia. Un país tiene un nombre y un array de provincias. Una provincia tiene un nombre.

2. Luego crearemos nuestro ***ProvincesTableViewController***, y nuestra ***ProvinceTableViewCell***. El controller de la tabla tendrá un array de países, cada uno de los cuales tienen una cierta cantidad de provincias.

3. Como queremos que nuestra tabla tenga varias secciones, el método "`numberOfSections`" ya no debe devolver 1, sino la cantidad de secciones que queremos tener. En este caso, como cada sección es un país, la cantidad de secciones será la cantidad de elementos que hay en el array de países:
```
return countries.count
```

4. De igual forma, el método "`numberOfRowsInSection`" ahora depende del parámetro "section". Como sabemos que las filas representan provincias y las secciones representan países, la cantidad de filas en la sección N va a ser la cantidad de provincias del país en la posición N del array:
```
let country = countries[section]
return country.provinces.count
```

5. El tercer método que debemos implementar para configurar una tabla es el que arma las celdas a presentar: "`cellForRowAtIndexPath`". En este caso, todas las filas van a ser iguales, representando cada una a una provincia. Para ello usaremos nuestra ***ProvinceTableViewCell***, con identificador "*provinceCell*".

6. Luego, como el parámetro *indexPath* nos indica sección y fila de la celda que debemos configurar, obtenemos el país correspondiente a la sección indicada

(indexPath.section), y dentro de las provincias de ese país, obtenemos la que corresponde a la fila indicada (indexPath.row).

```
let cell = tableView.dequeueReusableCell(withIdentifier: "provinceCell",
                                                        for: indexPath)
if let stateCell = cell as? ProvinceTableViewCell {
      let country = countries[indexPath.section]
      let state = country.provinces[indexPath.row]
      stateCell.setup(state: state)
}
return cell
```

7. Finalmente, queremos que cada sección tenga un header (un título) custom, con la información que nosotros queremos definir. De igual forma, cada sección puede tener un footer custom.

   7.1. Para definir un header custom, se debe implementar el método:

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

   7.2. Para definir un footer custom, se debe implementar el método:

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

   7.3. En ambos casos, se debe devolver un String, que va a ser el texto a mostrar para la sección indicada por parámetro. Si queremos que el título de una sección sea el nombre del país devolvemos el nombre del elemento en la posición del array indicada como número de sección:

   ```
   let country = countries[section]
   return country.name
   ```

ALTERNATIVA

8. Además de determinar un texto para el header y/o el footer, la alternativa es definir una view custom que puede tener tantas cosas como sea necesario:

```
tableView(_ tableView: UITableView,
            viewForHeaderInSection section: Int) -> UIView?
tableView(_ tableView: UITableView,
            viewForFooterInSection section: Int) -> UIView?
```

9. En cualquiera de estos casos, en lugar de devolver un texto para que se muestre como título del header o del footer, debemos devolver una `UIView` para mostrar en su lugar. Normalmente esa vista se generará en un archivo aparte (.xib) y dentro de estos métodos sólo debemos generarla y configurarla:

```
if let headerView = Bundle.main.loadNibNamed("xibFileName", owner:
self, options: nil).first as? CustomHeaderView {
      let element = elementsArray[section]
      headerView.setup(element)
      return headerView
}
return nil
```

10. Dado que esa vista se genera en un archivo aparte, la tabla no sabe cual es su tamaño, por lo que es necesario indicar el alto que debe tener el header/footer usando un método específico para tal fin. En cualquiera de estos métodos hay que devolver un número decimal, que indicará la altura correspondiente:

```
tableView(_ tableView: UITableView,
           heightForHeaderInSection section: Int) -> CGFloat
tableView(_ tableView: UITableView,
           heightForFooterInSection section: Int) -> CGFloat
```