PASO A PASO 3:
UITableViewController - Add Element

Vamos a crear una aplicación conformada por una tabla, que va a listar información de contactos. Además, agregaremos un mecanismo que nos permita agregar nuevos contactos a la tabla en una pantalla creada para tal fin.

1. Tomaremos como punto de partida la aplicación armada en *Paso a Paso 1*. La misma consta de ***ContactsTableViewController***, un `UITableViewController`, que lista los contactos del usuario; ***ContactTableViewCell***, una `UITableViewCell` custom; y ***Person***, una clase que representa a un contacto de la Agenda.

2. En principio vamos a crear la pantalla de creación de contactos que queremos mostrar cuando el usuario necesite agregar una fila a la tabla.

    2.1. File > NewFile > *Cocoa Touch Class*. Será subclase de `UIViewController`, y lo llamaremos ***AddContactViewController***.

    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 ***AddContactViewController***.

    2.3. Finalmente, configuramos la pantalla de acuerdo a nuestras necesidades, agregando los componentes necesarios, y generando los Outlets y Actions que hagan falta. En este caso, agregaremos 3 `UITextField` para que se pueda ingresar el *nombre*, *email* y *teléfono* del nuevo contacto; y un botón para guardar la información ingresada y agregarla a la lista. Creamos los Outlets para los campos de texto, y un Action para el botón.

    2.4. La idea de esta pantalla es que el usuario ingrese la información del contacto que quiere agregar, toque el botón de *guardar* y la aplicación vuelva a la tabla, donde debería estar agregado el nuevo contacto ingresado. Para ello usaremos el patrón Listener/Observer: vamos a definir un protocolo ***AddContactObserver*** con un método: `func didSave(contact: Person)`. Como las clases que implementen este protocolo estarán "observando" nuestro nuevo ***AddContactViewController***, lo crearemos en el mismo archivo, antes de la definición de la clase.

2.5. Cuando el usuario toque el botón *guardar*, el controller va a tener que informar a su observer, indicandole que se guardó un nuevo contacto. Para lograr esto, ***AddContactViewController*** va a tener un atributo llamado "*observer*" de tipo ***AddContactObserver?***.

2.6. Luego, en el Action del botón, debemos obtener el contenido de los campos de texto, instanciar una nueva persona, y avisarle al observer mediante el método *didSave*. Asimismo, como queremos que la aplicación vuelva atrás y muestre nuevamente la tabla, debemos hacer pop del controller presentado:

```
if let name = nameTextField.text,
      let email = emailTextField.text,
      let phone = phoneTextField.text {

      let newContact = Person(theName: name,
                              thePhone: phone, theEmail: email)
      observer?.didSave(contact: newContact)
      navigationController?.popViewController(animated: true)
}
```

3. Necesitamos presentar esta nueva pantalla de alguna forma. Para ello, agregaremos un `UIBarButtonItem` a la barra de navegación de la pantalla donde está la tabla (si no está la barra de navegación, agregar el UINavigationController yendo a Editor > Embed In > NavigationController). Si aún así no permite agregar el botón, agregar un `UINavigationItem` al controller. Luego, desde el botón generamos un *segue* a la nueva pantalla. Este tipo de botón en la barra de navegación se puede configurar seteando título, o se puede elegir una de las opciones predeterminadas (add, edit, cancel, save, etc).

4. Tal como definimos nuestro protocolo ***AddContactObserver***, implica que una clase podría implementarlo y enterarse en qué momento se agregó un nuevo contacto. Como queremos que el ***ContactsTableViewController*** sea quien se notifica, definiremos que implementa el protocolo; y, como consecuencia, deberá además implementar el método que se define en el protocolo: *didSave(contact: Person)*.

4.1. Cuando se agrega un nuevo contacto hay dos cosas que debemos hacer: por un lado agregarlo a la lista de contactos, o sea, al array *persons*.

```
persons.append(contact)
```

4.2. Sin embargo, ante un cambio así la tabla no sufrirá cambios. Para forzar que la tabla se actualice debemos indicarle a la tabla que debe recargar volver a preguntar cuantas celdas hay, y recargarlas. Para ello, hay un método de la clase `UITableView` que fuerza a la tabla a regenerarse.

```
tableView.reloadData().
```

5. Finalmente, debemos indicarle al ***AddContactViewController*** quien va a ser su *observer*.

    5.1. En la clase ***ContactsTableViewController*** agregamos el método *prepareForSegue*.

    5.2. Este método se ejecutará antes de pasar a la siguiente pantalla, por lo que podemos obtener una referencia al destino del segue y configurar el observer.

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
  if let controller = segue.destination as? AddContactViewController {
      controller.observer = self
  }
}
```