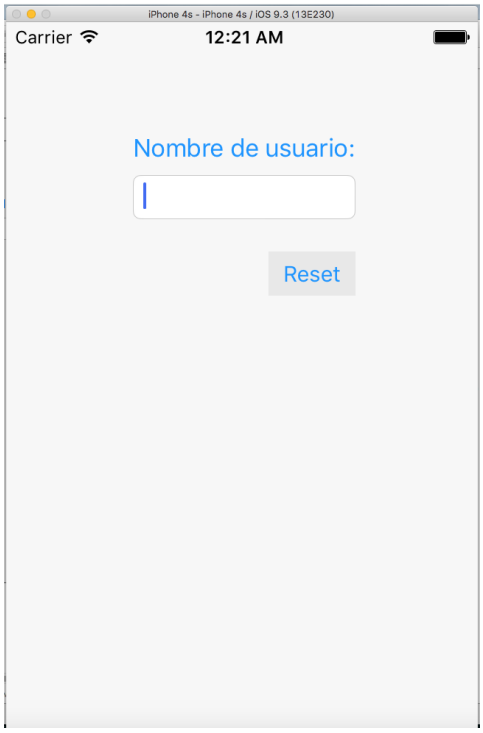
LÍNEAS DE PRODUCTOS SOFTWARE

Sesión 01. XCode, Swift, iOS: Introducción

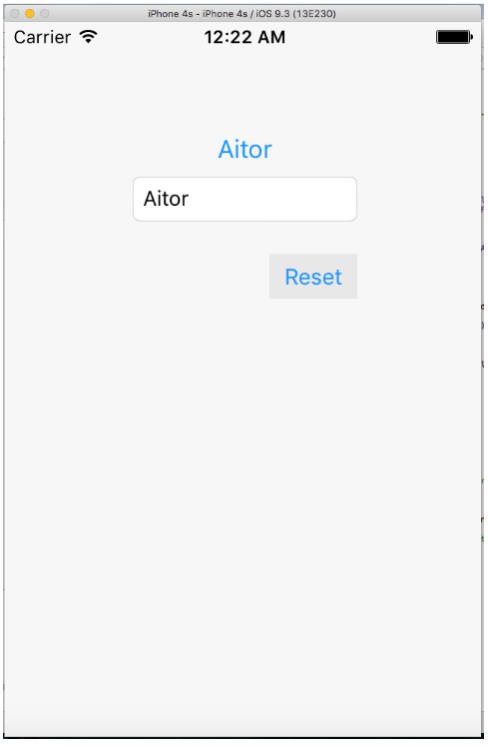
Aitor García Luiz

Grado en Ingeniería Informática

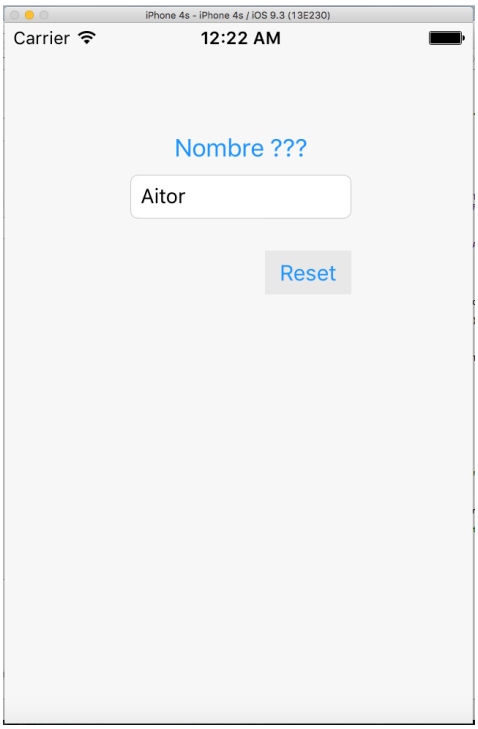
Los ejercicios de la sesión01 parten de la base de la aplicación creada gracias al guion de prácticas:



En esta aplicación se puede introducir un texto en el campo de texto y al terminar de introducirlo la etiqueta superior cambiará por dicho texto:



Y además también puede resetear el nombre de la etiqueta al pulsar el botón ‘Reset’ mostrando el siguiente texto:



Una vez tenemos claras las funcionalidades de la aplicación de origen, podemos comenzar a realizar los ejercicios del guion de prácticas:

Ejercicio 1.

Modificar la app para que exhiba el siguiente comportamiento:

- **En la etiqueta de texto aparecerá el texto:**
  "Hola desconocido"

  Para cambiar el texto que aparece en la etiqueta solamente debemos seleccionar la etiqueta e irnos al Área de utilidad, una vez allí vamos al Inspector de atributos y escribimos el texto que deseamos:

El cambio se verá de la siguiente forma en el simulador:

- **Cuando el usuario finalice la edición en el campo de texto, copiará lo escrito en la etiqueta de la siguiente forma:**

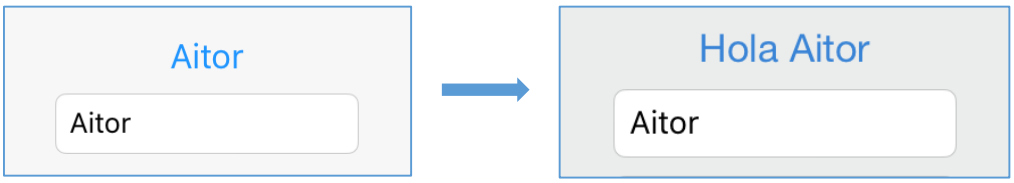
  "Hola <nombre>"

  La información que nos muestra la etiqueta la recoge del delegado del campo de texto. Para ello debemos añadir el trozo de cadena deseado en dicha función:

```
func textFieldDidEndEditing(textField: UITextField) {
    nombreLB.text = textField.text
}
```

```
func textFieldDidEndEditing(textField: UITextField) {
    nombreLB.text = "Hola " + textField.text!
}
```

Obteniendo así los siguientes cambios en nuestra aplicación:



- **El botón 'Reset' volverá a escribir la cadena "Hola desconocido", eliminando también el contenido del campo de texto (si lo hubiese).**

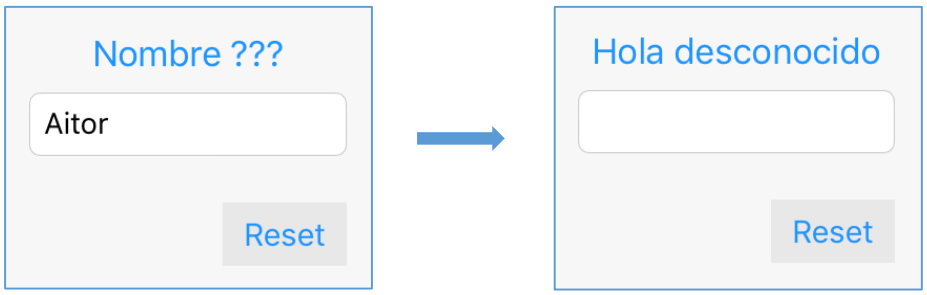
  Para realizar este apartado debemos modificar la función que declara los eventos que se realizarán al pulsar el botón:

```
//MARK: actions
@IBAction func resetButton(sender: AnyObject) {
    nombreLB.text = "Nombre ???";
}
```

  Primeramente, cambiamos la cadena de texto que aparecerá en la etiqueta y posteriormente cambiamos el texto del campo de texto por una cadena de texto vacía de forma que elimine el contenido que hubo anteriormente.

```
//MARK: actions
@IBAction func resetButton(sender: AnyObject) {
    nombreLB.text = "Hola desconocido";
    nombreTF.text = "";
}
```

  Cuando se introduzca un dato y se pulse el botón 'Reset', los cambios serán reflejados de la siguiente manera:



Ejercicio 2.

Vamos a ampliar la app con un nuevo campo de texto para el apellido del usuario. Así pues, tras rellenar el apellido, en la etiqueta aparecerá el siguiente texto:

"Hola <nombre> <apellido>"

Para introducir un apellido lo ideal es añadir un nuevo campo de texto que sea similar al campo de texto del nombre que ya habíamos creado anteriormente.

Una vez tenemos el campo de texto, haciendo uso de *"ctrl+drag"* creamos la variable del mismo en nuestro código junto con las anteriores:

```
//MARK: properties

@IBOutlet var vista: UIView!
@IBOutlet weak var nombreLabel: UILabel!
@IBOutlet weak var nombreTF: UITextField!
@IBOutlet weak var apellidoTF: UITextField!
```

A continuación, hay que crear otro delegado para el campo de texto del apellido, exactamente igual que se hizo con el delegado del nombre:

```
override func viewDidLoad() {
    super.viewDidLoad()
    nombreTF.delegate = self;
    apellidoTF.delegate = self;
    // Do any additional setup af
}
```

También habría que modificar la función encargada de colocar la información del delegado en la etiqueta de texto:

```
func textFieldDidEndEditing(textField: UITextField) {
    if (nombreTF.text == "" && apellidoTF.text! == ""){
        nombreLabel.text = "Hola desconocido";
    } else {
        nombreLabel.text = "Hola " + nombreTF.text! + " " + apellidoTF.text!
    }
}
```

Con ayuda de una sentencia condicional, controlamos los casos en los que ambos campos de texto están vacíos para que en el caso de que se seleccione el de nombre y se pase al del apellido sin introducir ningún nombre, no se refleje en la etiqueta con una cadena vacía. En los demás casos la etiqueta mostrará el nombre, el apellido, o ambos, según la información que se haya introducido en los campos de texto.

Para finalizar, en la función del botón 'Reset' añadiremos que al pulsarlo también vacíe el campo de texto del apellido, exactamente igual que se hacía con el del nombre:

```
@IBAction func setResetButton(sender: AnyObject) {
    nombreLabel.text = "Hola desconocido";
    nombreTF.text = "";
    apellidoTF.text = "";
}
```

Ejercicio 3.

Vamos a añadir un nuevo botón para cambiar el color de fondo de la ventana (color aleatorio).

Para realizar este último ejercicio de la sesión, primero de todo necesitamos un botón que realice la acción del cambio de color de fondo. Una vez tenemos el botón colocado en la interfaz debemos hacer uso de *"Ctrl+drag"* para introducir la acción que ejecutará el botón al ser pulsado.

En este caso se ha dividido en dos funciones diferentes para que sea más claro su funcionamiento:

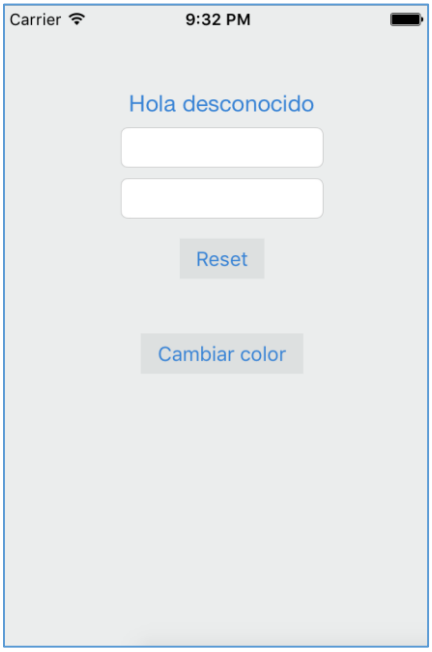
```
func getRandomColor() -> UIColor{
    let randomRed:CGFloat = CGFloat(drand48())
    let randomGreen:CGFloat = CGFloat(drand48())
    let randomBlue:CGFloat = CGFloat(drand48())

    return UIColor(red: randomRed, green: randomGreen, blue: randomBlue, alpha:
        1.0)
}
```

En esta función anterior lo que tenemos es que se generan aleatoriamente números para la variable de color rojo, verde y azul, que en conjunto definen el color del fondo.

Ya solo falta que nuestro botón inicie dicha función cada vez que es pulsado:

```
@IBAction func setColorButtom(sender: AnyObject) {
    vista.backgroundColor = getRandomColor()
}
```

Nuestra aplicación se ve así nada más emularla:



Y así se ve tras pulsar varias veces el botón que acabamos de añadir: