MOV02 - Laboratorio 2

Juan Pablo Arias Mora

Febrero 2021

1 Introducción

El objetivo del laboratorio es dar una introducción de uso sobre el entorno de desarrollo XCode utilizando un ejemplo practico sencillo, en este caso una aplicación que calcule dependiendo de una fecha seleccionada cuantos años, meses y días han transcurrido. Para referencias futuras el código fuente final del mismo esta disponible en https://github.com/pabloariasmora/MOV02-Cenfotec-Demo-Labs.git

1.1 Versiones

- MacOs 10.15.7 - Inglés
- Xcode 12.4
- Simulador iPhone 11 Pro Max

2 Instrucciones

"Sé un punto de referencia de calidad. Algunas personas no están acostumbradas a un ambiente donde la excelencia es aceptada." Steve Jobs

2.1 Creación de un proyecto básico

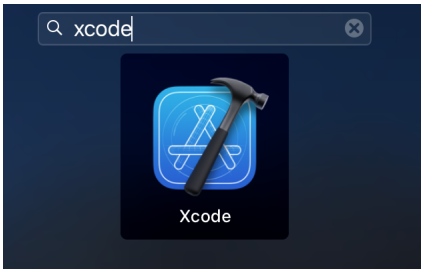

Figure 1: Icono Xcode

Paso 1: Una vez inicie la aplicación veremos una pantalla similar a la de la figura 2, si es la primera vez no se desplegarán proyectos en el panel de la derecha.

Paso 2: Seleccionamos la opción "Create a New Xcode Proyect" (ver figura 2)

Paso 3: Nos debe aparecer una ventana para poder seleccionar un nuevo Template para usarlo de base en nuestro proyecto (ver figura 3), importante revisar que en el los tabs superiores este la opción de iOS seleccionada.

Paso 4: Seleccionamos la opción "App" (ver figura 4)

Paso 5: En la siguiente ventana vemos estableceremos las opciones para la la configuración especifica de la nueva aplicación. (ver figura 5)
Product Name : El nombre de su aplicación tal como aparecerá en la App Store y aparecerá en un dispositivo cuando se instale. El nombre del producto debe tener al menos 2 caracteres y no más de 255 bytes, y preferiblemente manejar solo minúsculas y sin espacios **valor :** agecalculator
Organization Identifier : Una reverse DNS que identifica de forma exclusiva a nuestra organización. **valor :** com.cenfotec.mov02.01

Figure 2: Pantalla Inicial Xcode

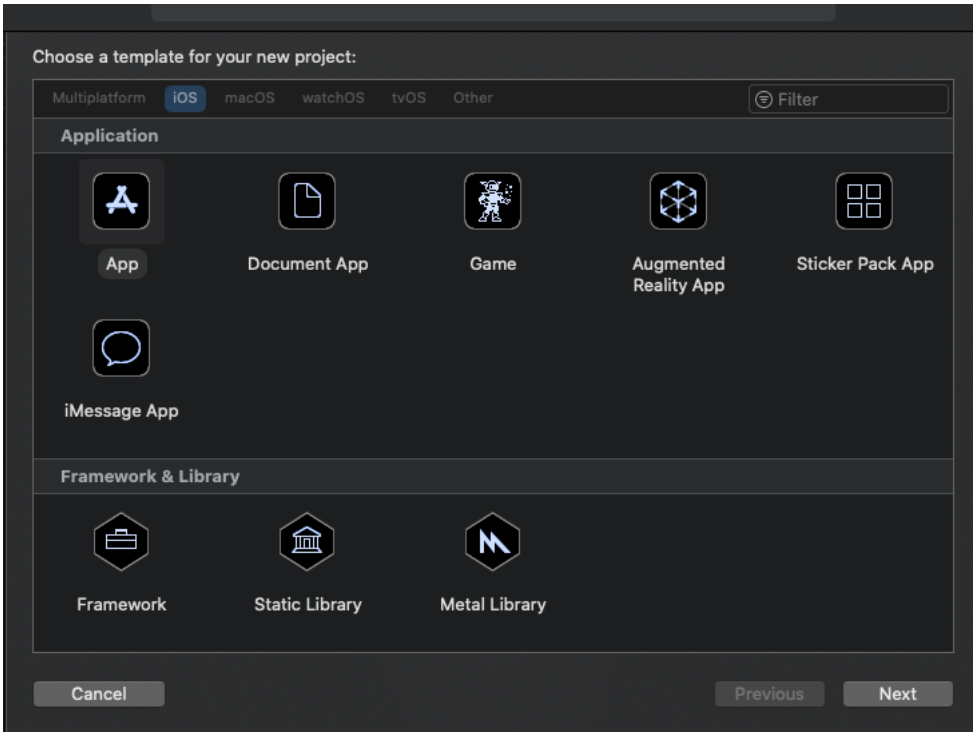

Figure 3: Selección de Template Xcode

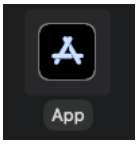

Figure 4: App Template Xcode

Bundle Identifier : Valor auto generado en base a los dos anteriores, pero dentro del ecosistema de App Store debe ser único. **Interface :** Gestor de manejo de la interfaz **valor :** StoryBoad
Life Cycle : Determina que puede o no hacer la aplicación con respecto al acceso a recursos **valor :** UIKIT App Delegate
Languge : Lenguaje de programación a utilizar **valor :** Swift
Use Core Data: Opciones de Almacenamiento interno **valor :** Sin Seleccionar

Include Test: Opciones de UnitTest **valor :** Sin Seleccionar

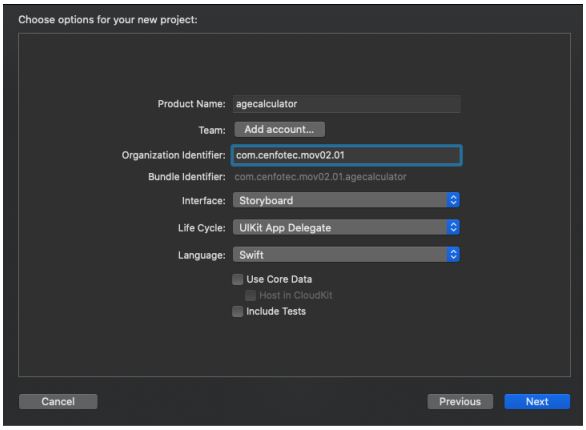

Figure 5: Opciones de Proyecto

Paso 6: Click en la opción Next, para disponernos a guardar el código dentro de nuestro computador. Seleccionar una carpeta apropiada, por el momento verificar que la opción de Source Control (Control de Versiones, figura 6) este sin habilitar. Y le damos click a Crear.

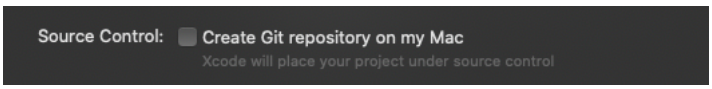

Figure 6: Opciones Control de Versiones

Paso 7: El resultado debe ser similar a la figura 7.

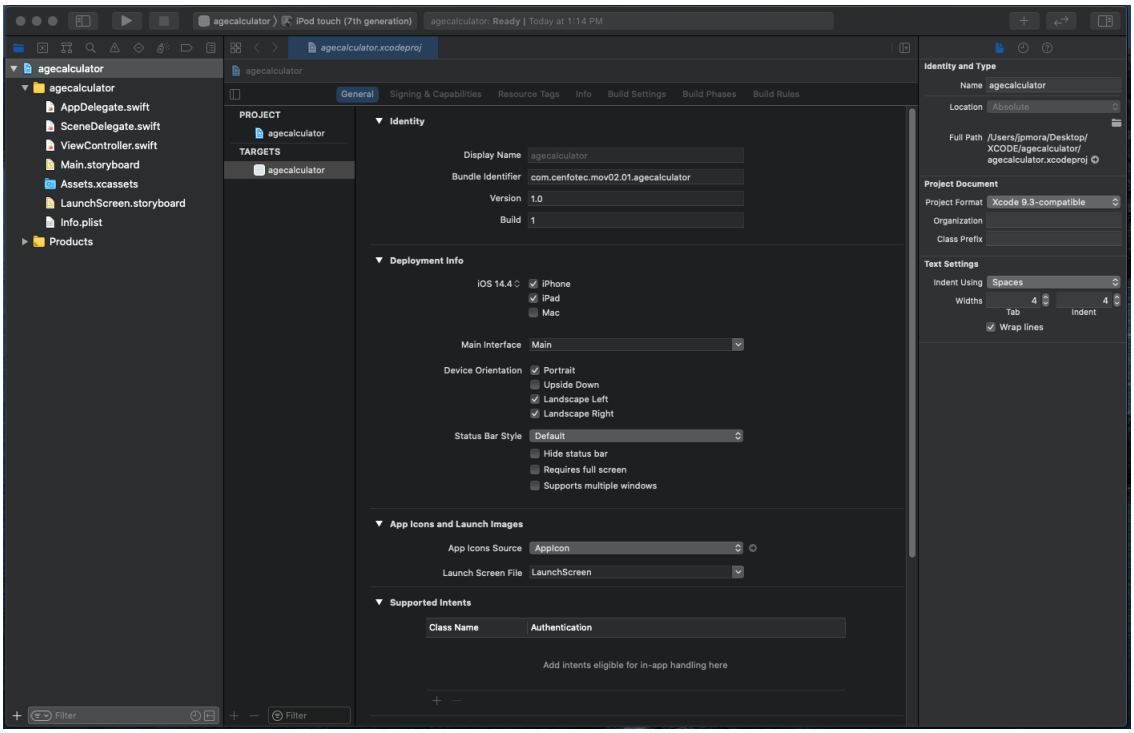

Figure 7: Pantalla Principal de un proyecto en Xcode

2.2 Configurando y Desplegando el Simulador

Paso 1: Configurar el simulador de iPhones a la versión **"iPhone 11 Pro Max"**, esto simplemente para estandarización de los resultados, si bien es cierto la gama otorgada de simuladores ejecuta la versión necesaria para este laboratorio,

pero sus dimensiones de pantalla pueden llegar a variar causando comportamientos de UI diferentes a los esperados (ex. botones en posiciones incorrecta), esto lo seleccionamos en la parte superior de la pantalla principal de Xcode, (ver figura 7, figura 8 área en color rojo y figura 9), es importante darle click sobre **"iPod touch (7th generation)"** y no sobre **agecalculator** ya que esto nos daría otras opciones y no las necesarias para cambiar el simulador.

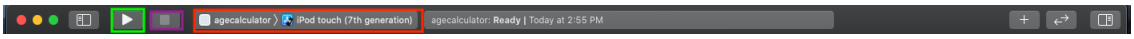

Figure 8: Selector de Simulador

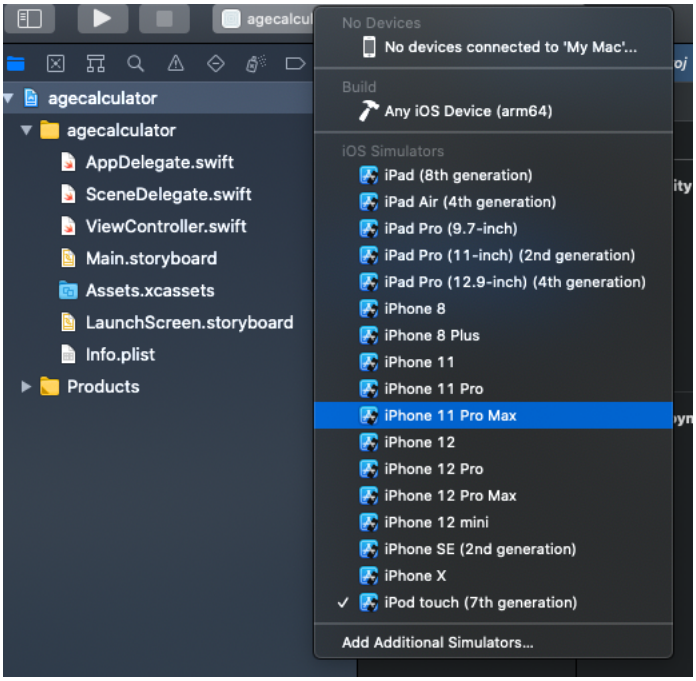

Figure 9: Selector de Simulador

Paso 2: Ejecutamos el simulador con darle click al icono de play en la parte superior de la pantalla (ver figura 8 área en color verde). El código primero es compilado y luego el simulador inicia tal cual inicia un iPhone real (ver figura 10).

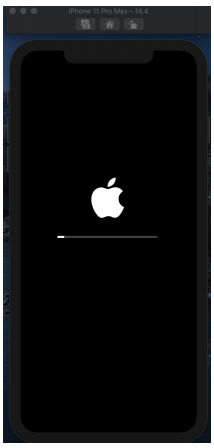

Figure 10: Simulador Iniciando

Paso 3: Luego que el simulador finalice su etapa de inicio, podemos apreciar la carga de una aplicación en blanco, la cual corresponde a nuestra primera versión de **"ageCalculator"**

Paso 4: Procedemos a cancelar la ejecución de la aplicación en el simulador con darle click al icono de Stop en la parte superior de la pantalla (ver figura 8 área en color morado). El siguiente proceso debe dejar el simulador ejecutando pero sin la aplicación en primer plano.

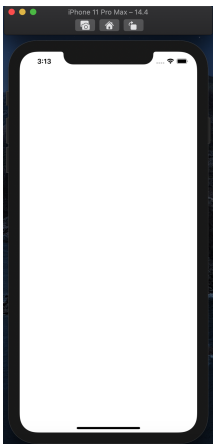

Figure 11: Simulador Ejecutando la App: ageCalculator por primera vez

2.3 Primeros pasos en Interfaz Gráfica (UI)

Paso 1: En la pantalla principal localizamos el área de **Navigation** (más detalles en figura 12) y seleccionamos el archivo **Main.storyboard**

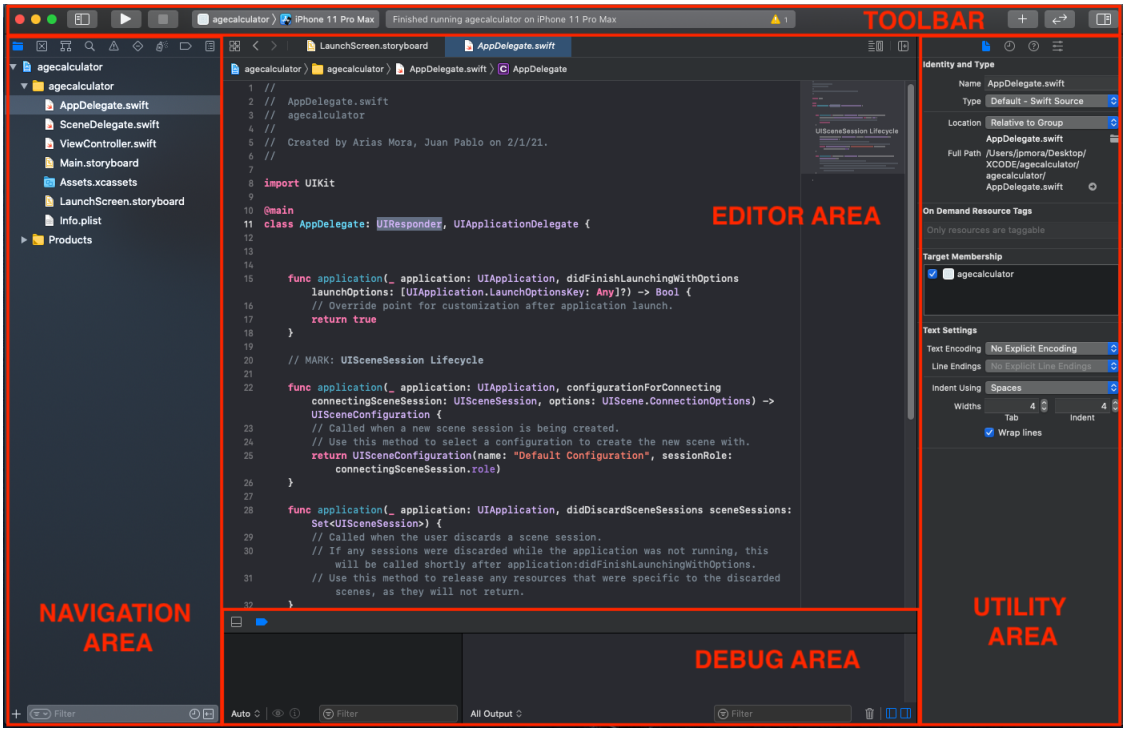

Figure 12: División general de áreas de XCode

Paso 2: En área del Editor ahora debe cambiar a mostrar una vista previa de los componentes de la interfaz gráfica (figura 13)

Paso 3: Damos click sobre el Icono "+" (figura 14 área roja) en el toolbar, para poder desplegar el panel de objetos (figura 14 área verde).

Paso 4: Seleccionamos el objeto Label y arrastramos (Drag and Drop) hacia el área color blanco.

Paso 5: Asegurándonos que el Label continúe seleccionado (ver figura 15 área verde). Seleccionamos el Inspector de Atributos en la parte superior del Utility Editor (ver figura 15 área roja), y cambiamos el nombre (label) del objeto de ser **Label** a ser **My Age**.

Paso 6: Asegurándonos que el **My Age**(Label) continúe seleccionado, vamos a cambiar los constrains del objeto, esto para asegurarnos se mantenga en proporción a los limites de la pantalla. Esto lo hacemos seleccionando en la parte

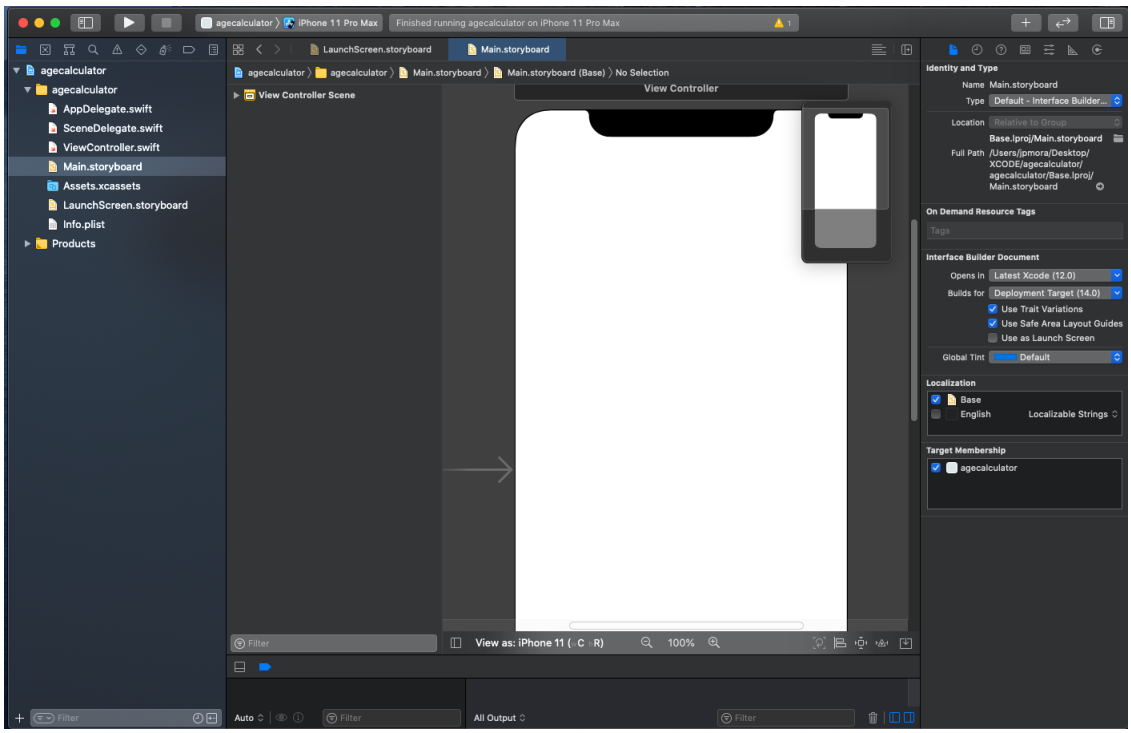

Figure 13: Editor con Preview de UI

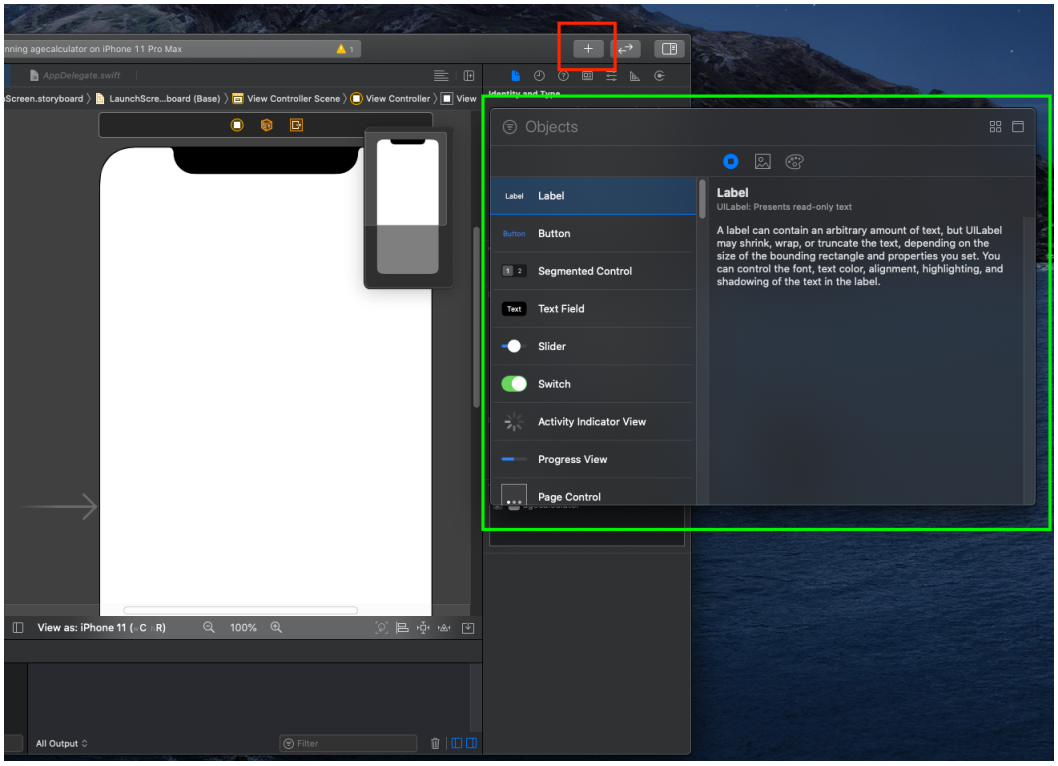

Figure 14: Agregar nuevos objetos

inferior del Editor Área (ver figura 16 área roja). Y colocamos el valor de **Top** en 16 (ver figura 17) y agregamos el constrain (ver figura 15).

Paso 7: Asegurándonos que el **My Age**(Label) continúe seleccionado, vamos a cambiar el alineamiento del objeto, esto para asegurarnos se mantenga en proporción a los limites de la pantalla. Esto lo hacemos seleccionando en la parte inferior del Editor Área (ver figura 16 área anaranjada). Marcamos el alineamiento horizontal y le damos agregar (ver figura 19).

Paso 8: Ahora siguiendo los pasos anteriores vamos a agregar un **Text Field**, al cual le pondremos en el atributo de

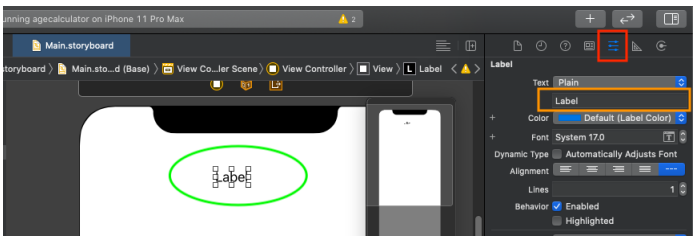

Figure 15: Inspector de Atributos

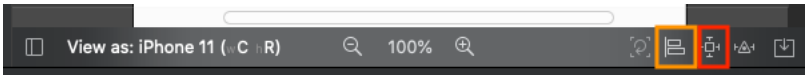

Figure 16: Object Align and Constrains

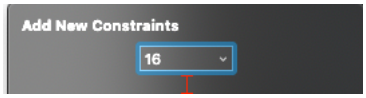

Figure 17: Top Constrain

Figure 18: Agregar Constrain

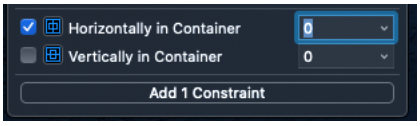

Figure 19: Agregar Alineamiento Horizontal

Placeholder el texto de **Select Date of Birth**, y en sus Contrains agregar **(Top:16, Left:16, Right:16)**

Paso 9: Ahora siguiendo los pasos anteriores vamos a agregar un **Button**, al cual le pondremos en el atributo de **Label** que actualmente dice **Button** el texto de **Calculate Age**, y en sus Contrains agregar **(Top:16, Width:200, Height:50)** y en su alineamiento marcamos el alineamiento horizontal.

Paso 10: El resultado debe ser similar a la figura 20. Si desea puede ejecutar **Build and Run** (símbolo de play) en el toolbar para comprobar que todo continúe bien.

2.4 Agregar Acciones a la UI

Paso 1: En el Navigator seleccionar el archivo **ViewController.swift**, no entraremos en detalles de porque este archivo de momento, ya que la finalidad del laboratorio es uso básico de Xcode.

Paso 2: Debemos agregar el mapeo de los objetos en el UI a las acciones que queremos se realicen, en la linea 11 agregamos el siguiente código.

```
@IBOutlet weak var ageLabel: UILabel!
@IBOutlet weak var dobTextField: UITextField!
```

Paso 3: Y agregar ahora en la linea 20 agregamos el siguiente código. Luego del cierre de la funcion **viewDivLoad** La imagen 21 muestra el resultado del código al momento.

```
@IBAction func calculateAgeBtnAxn(_ sender: Any) {
    ageLabel.text = "This is a Test"
}
```

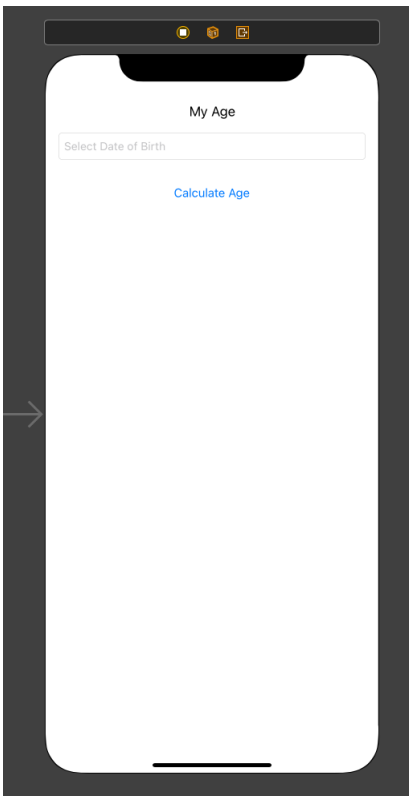

Figure 20: Resultado Final del UI

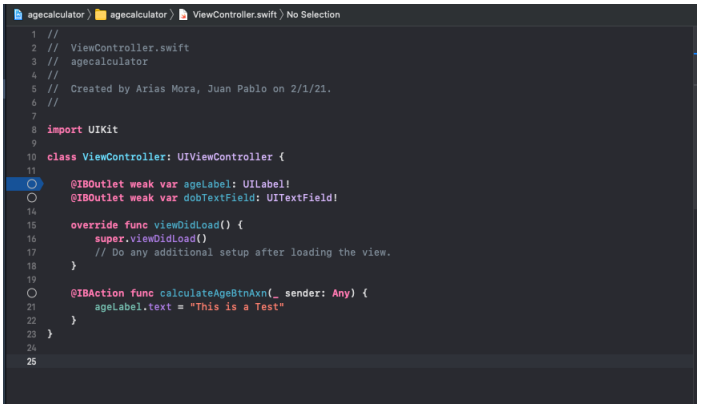

Figure 21: Resultado Final del UI

Paso 4: A este punto agregamos el código pero no hemos realizado la asociación. Es por eso que en el archivo **ViewController.swift** las lineas que agregamos se visualizan con un circulo a la izquierda en sus números de linea. Hay múltiples maneras pero la que usualmente se usa es abrir una segunda ventana de editor utilizando el botón de agregar en la esquina superior derecha del área de editor (ver figura 22). El efecto normal es que nos agregue un duplicado del archivo que tenemos ya abierto en el otro editor, entonces basta con ahora seleccionar el archivo **Main.storyboard** en el área de Navigation, tendremos ambos archivos abiertos en el área de editor.

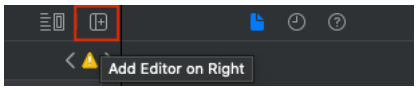

Figure 22: Agregar Editor

Paso 5: Ahora uno a uno debemos seleccionar los objetos iniciando por el **Label** dentro del **Main.storyboard**, presionar la tecla control (recuerde que en el teclado de MacOs esta en la esquina inferior izquierda)

Paso 6: Arrastrarlo hacia la variable de nombre **ageLabel** (ver figura 23), debemos comprobar que el asocie fue efectivo si el circulo a las izquierda de la linea se encuentra color blanco. Algo muy importante es que se arrastre justo al

nombre de la variable.

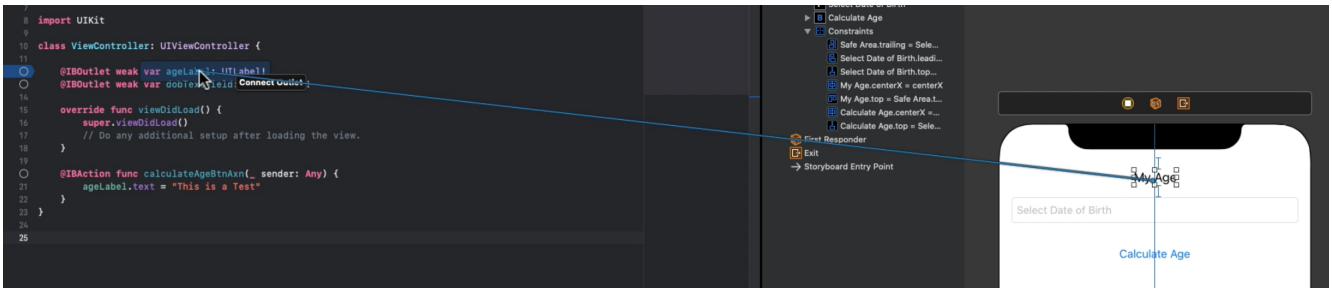

Figure 23: Arrastrar y Mapear Objetos de un Storyboard al View

Paso 7: Repetir el proceso con el Text Field dentro del **Main.storyboard**, arrastrarlo hacia la variable de nombre **dobTextField**.

Paso 8: Repetir el proceso con el Button dentro del **Main.storyboard**, arrastrarlo hacia la la función de nombre **calculateAgeBtnAxn**.

Paso 9: Verificar que todos los círculos a la izquierda del código se encuentren color blanco.

Paso 10: Ejecutar **Build and Run** (símbolo de play) en el toolbar para comprobar que todo continúe bien, una vez cargue la aplicación, podemos presionar el Button **Calculate Age** y el texto del Label superior debe cambiar y convertirse en **This is a Text**, tal cual definimos en la línea 21 de nuestro archivo **ViewController.swift** (ver figura 24)

Paso 11: Como siguiente paso vamos a agregar un Datepicker, el cual nos permita seleccionar una fecha y así poder calcular el tiempo que ha transcurrido hasta este día. En la linea 14 agregar el siguiente código, justo antes de la definición de la función de **viewDidLoad** (ver figura 24)

```
var datePicker:UIDatePicker?
var toolBar:UIToolbar = UIToolbar()
var dateOfBirth:Date?
```

Paso 12: Agregamos las funciones necesarias definimos la acciones y los botones del Datepicker justo después de la definición de la función de **viewDidLoad** pero justo antes de la función **calculateAgeBtnAxn**, recordemos que la finalidad es una primera experiencia entonces no se entraran en el laboratorio en los detalles del código como parte del documento, pero los podemos discutir durante la clase, de igual manera los comentarios deberian ayudar a predecir el comportamiento.(ver figura 24)

```
func doDatePicker(){
    // DatePicker
    // create a datepicker object with some frame
    self.datePicker = UIDatePicker(frame:CGRect(x: 0, y: 0,
                                                width: self.view.frame.size.width,
                                                height: 200))
    // set mode of date
    self.datePicker?.datePickerMode = UIDatePicker.Mode.date

    // IOS 14 Calendar https://developer.apple.com/documentation/uikit/uidatepicker
    // We need to set the default
    datePicker?.preferredDatePickerStyle = UIDatePickerStyle.wheels

    // Max date
    datePicker?.maximumDate = Date()
    // Set datepicker to textfield Input view
    dobTextField.inputView = datePicker
```

```
    // ToolBar
    toolBar.barStyle = .default
    toolBar.isTranslucent = true
    toolBar.tintColor = UIColor(red: 92/255, green: 216/255, blue: 255/255, alpha: 1)
    toolBar.sizeToFit()

    // Adding Button ToolBar
    // create done button
    let doneButton = UIBarButtonItem(title: "Done",
                                     style: .plain,
                                     target: self,
                                     action: #selector(self.doneClick))
    // space between button
    let spaceButton = UIBarButtonItem(barButtonSystemItem: .flexibleSpace,
                                      target: nil,
                                      action: nil)
    // cancel button
    let cancelButton = UIBarButtonItem(title: "Cancel", style: .plain,
                                       target: self,
                                       action: #selector(self.cancelClick))
    // add above buttons
    toolBar.setItems([cancelButton, spaceButton, doneButton], animated: true)
    toolBar.isUserInteractionEnabled = true
    // set toolbar as textfield input accessory
    dobTextField.inputAccessoryView = toolBar


    }


@objc func doneClick() {
    //Creating an object of date formater
    let dateFormatter = DateFormatter()
    //setting style of date
    dateFormatter.dateStyle = .medium
    //getting date object from picker
    dateOfBirth = datePicker?.date
    dobTextField.text = dateFormatter.string(from: dateOfBirth!)
    //close datepicker
    self.view.endEditing(true)


}

@objc func cancelClick() {
    // close datepicker
    self.view.endEditing(true)

}
```

Paso 13: Debemos instanciar el Datepicker como parte de la configuración inicial después de cargar el View. Por tanto tenemos que agregar la siguiente linea dentro de la función **viewDidLoad**, justo después de los comentarios. (ver figura 24)

Paso 14: Ejecutar **Build and Run** (símbolo de play) en el toolbar para comprobar que todo continúe bien. El resultado final debe ser similar a la figura 20, solamente que si presionamos sobre nuestro Texf Field, ahora la interfaz debe cambiar y mostrarnos el Datepicker (figura **??**), es mismo funciona con scroll para otros tipos podemos revisar la propiedad preferredDatePickerStyle dentro de nuestro código. Una vez seleccionemos la fecha damos click sobre el botón Done, y la misma debe de quedar desplegada sobre el TextField (figura **??**).

Paso 15: Finalmente debemos agregar el código responsable para el calculo de meses, días y años transcurridos desde la fecha

agecalculator 〉 agecalculator 〉 ViewController.swift 〉 M doDatePicker()

```swift
    @IBOutlet weak var ageLabel: UILabel!
    @IBOutlet weak var dobTextField: UITextField!

    var datePicker:UIDatePicker?
    var toolBar:UIToolbar = UIToolbar()
    var dateOfBirth:Date?

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        self.doDatePicker()
    }

    func doDatePicker(){
            // DatePicker
            // create a datepicker object with some frame
            self.datePicker = UIDatePicker(frame:CGRect(x: 0, y: 0, width: self.view.frame.size.width, height: 200))
            // set mode of date
            self.datePicker?.datePickerMode = UIDatePicker.Mode.date

            // IOS 14 Calendar https://developer.apple.com/documentation/uikit/uidatepicker
            // We need to set the default
            datePicker?.preferredDatePickerStyle = UIDatePickerStyle.wheels

            // Max date
            datePicker?.maximumDate = Date()
            // Set datepicker to textfield Input view
            dobTextField.inputView = datePicker

            // ToolBar
            toolBar.barStyle = .default
            toolBar.isTranslucent = true
            toolBar.tintColor = UIColor(red: 92/255, green: 216/255, blue: 255/255, alpha: 1)
            toolBar.sizeToFit()

            // Adding Button ToolBar
            // create done button
            let doneButton = UIBarButtonItem(title: "Done", style: .plain, target: self, action: #selector(self.doneClick))
            // space between button
            let spaceButton = UIBarButtonItem(barButtonSystemItem: .flexibleSpace, target: nil, action: nil)
            // cancel button
            let cancelButton = UIBarButtonItem(title: "Cancel", style: .plain, target: self, action: #selector(self.cancelClick))
            // add above buttons
            toolBar.setItems([cancelButton, spaceButton, doneButton], animated: true)
            toolBar.isUserInteractionEnabled = true
            // set toolbar as textfield input accessory
            dobTextField.inputAccessoryView = toolBar


        }


        @objc func doneClick() {
            //Creating an object of date formater
            let dateFormatter = DateFormatter()
            //setting style of date
            dateFormatter.dateStyle = .medium
            //getting date object from picker
            dateOfBirth = datePicker?.date
            dobTextField.text = dateFormatter.string(from: dateOfBirth!)
            //close datepicker
            self.view.endEditing(true)


        }

        @objc func cancelClick() {
            // close datepicker
            self.view.endEditing(true)

        }

    @IBAction func calculateAgeBtnAxn(_ sender: Any) {
        ageLabel.text = "This is a Test"
    }
}
```

Figure 24: Código con Datepicker

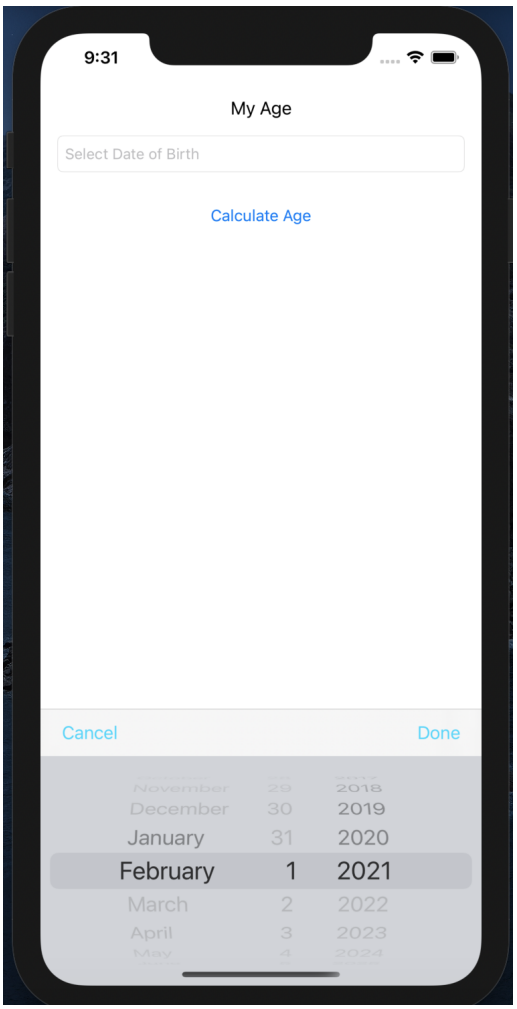

Figure 25: Datepicker desplegado al presionar el Text Field

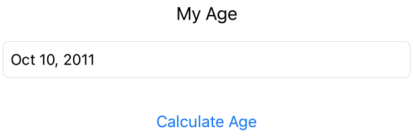

Figure 26: Valor del Datepicker en el Text Field

seleccionada y agregarlo a la acción del botón, para ellos debemos reemplazar el código actual dentro de la función **calculateAgeBtnAxn**, con el siguiente código. (ver figura 27)

```
// Check if there will be date of birth
if let dob = dateOfBirth{
    //Today Date
    let today = Date()
    // Calendar reference
    let calendar = Calendar.current
    // Calculate age from calendar
    let age = calendar.dateComponents([.year, .month, .day], from: dob, to: today)
    let ageInYear = age.year ?? 0
    let ageInMonth = age.month ?? 0
    let ageInDays = age.day ?? 0

    ageLabel.text = "\(ageInYear)yr \(ageInMonth)m \(ageInDays)d"
```

```
}
```

```swift
@IBAction func calculateAgeBtnAxn(_ sender: Any) {
    // Check if there will be date of birth
    if let dob = dateOfBirth{
        //Today Date
        let today = Date()
        // Calendar reference
        let calendar = Calendar.current
        // Calculate age from calendar
        let age = calendar.dateComponents([.year, .month, .day], from: dob, to: today)
        let ageInYear = age.year ?? 0
        let ageInMonth = age.month ?? 0
        let ageInDays = age.day ?? 0

        ageLabel.text = "\(ageInYear)yr \(ageInMonth)m \(ageInDays)d"
    }
}
```

Figure 27: Código para el calculo de días transcurridos

Paso 16: Ejecutar **Build and Run** en el toolbar para comprobar que todo continúe bien. El resultado final debe ser similar a la figura 20, solamente que cuando seleccionemos una fecha del datepicker, y presionemos sobre el boton **Calculate Age**, el Label superior de la aplicación debe cambiar y convertirse en el calculo de años, meses y días desde esa fecha al día de hoy.(figura 28).

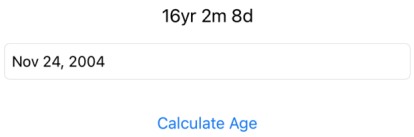

Figure 28: Resultado Final del Laboratorio