

# MOV02 - Laboratorio 5

Juan Pablo Arias Mora

Febrero 2021

## 1 Introducción

**SwiftUI** es un conjunto de herramientas de interfaz de usuario que nos permite diseñar aplicaciones de forma declarativa. Esa es una forma elegante de decir que le decimos a **SwiftUI** cómo queremos que se vea y funcione nuestra interfaz de usuario, y descubre cómo hacer que eso suceda cuando el usuario interactúa con ella.

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 12 Pro Max

## 2 Instrucciones

”Todo el mundo debería aprender a programar, porque te enseña a pensar.” Steve Jobs

### 2.1 Descarga de Recursos

Paso 1: Para el laboratorio se requieren algunas imágenes las mismas las puede encontrar en:

<https://gist.github.com/pabloariasmora/288d2f734327dcc513ebfa6de84882ac/archive/33625204642d32aa25bdbbb.zip>

Paso 2: Descargue el archivo .zip.

Paso 3: Descomprima el archivo .zip, y luego dentro de este descomprima el archivo images.zip

Paso 4: El contenido de el mismo se usara mas adelante en el Laboratorio

### 2.2 Creación de aplicación base

Paso 1: Cree un nuevo proyecto en Xcode, utilizando el template App dentro de iOS, en la ventana de opciones para la configuración específica de la nueva aplicación.

**Product Name :** swui  
**Organization Identifier :** com.cenfotec.mov02.05  
**Interface :** SwiftUI  
**Life Cycle :** SwiftUI App  
**Languge :** Swift  
**Use Core Data:** Sin Seleccionar  
**Include Test:** Sin Seleccionar

## 2.3 Análisis del código base

Paso 1: Abrir el archivo **swuiApp.storyboard**. Una aplicación que usa el **SwiftUI App Life Cycle** tiene un **Struct** que conforma el protocolo a **App**. El cuerpo del **Struct** retorna uno o más **Scenes**, las cuales en turnos proveen el contenido a desplegar. El atributo **@main** identifica el punto de entrada de la aplicación. (El código en el archivo debe ser similar al que se muestra a continuación)

```
import SwiftUI

@main
struct LandmarksApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

Paso 2: Abrir el archivo **ContentView.swift**. Por defecto los archivos **SwiftUI** declaran dos **structs**. La primera conforma el **View Protocol** y describe el contenido del **View** y su **Layout**. La segunda declara un **preview** de la vista. (El código en el archivo debe ser similar al que se muestra a continuación)

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Text("Hello, world!")
            .padding()
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Paso 3: En el **canvas**, debe hacer **click** sobre la palabra **Resume** para poder ver desplegar el **preview** de la **View** (ver figura 1 área roja). Si por alguna razón el mismo no se desplegara como en la figura 1, entonces puede utilizar la barra superior de **Xcode: Editor - Canvas** para mostrarlo. Como otro dato importante este "renderizado" de la vista consume mucho más memoria que el uso de Storyboards. Luego del "renderizado" deberíamos tener un resultado como la figura 2.

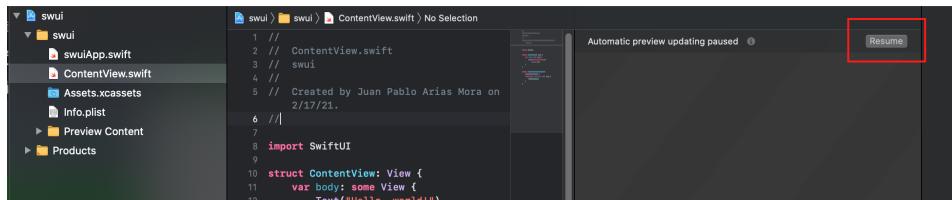


Figure 1: Vista Inicial Canvas

Paso 4: Dentro de la propiedad **body**, cambie el valor "**Hello, World!**" por un saludo hacia usted. A medida que realice cambios en el código del **View**, el **preview** se actualiza para reflejar los cambios.

Paso 5: El **SwiftUI Inspector** es una herramienta clave para el manejo de los **preview**. En el **preview** realice un **Command-click** sobre el saludo que acaba de modificar para mostrar el popup de edición de las estructuras, y seleccione **Show SwiftUI Inspector**. El popup muestra otros atributos que podríamos modificar, dependiendo del tipo de vista que inspeccionemos. (ver figura 3)

Paso 6: Utilice el **SwiftUI Inspector** para cambiar el texto del saludo a "Corcovado", este es el nombre de nuestro punto de referencia que vamos a mostrar en nuestra aplicación. (ver figura 4)

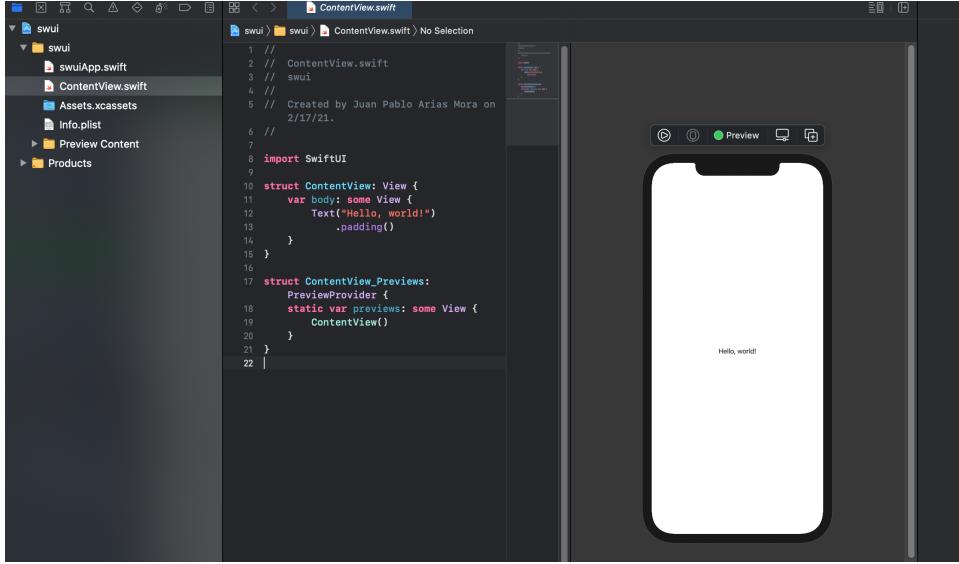


Figure 2: Preview Canvas

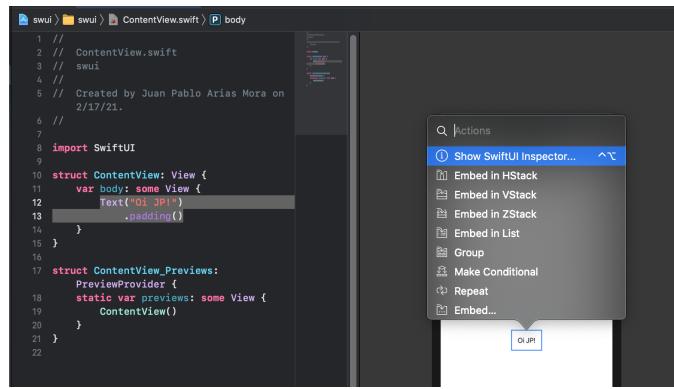


Figure 3: Abriendo el SwiftUI Inspector



Figure 4: Utilizando el SwiftUI Inspector

Paso 7: Utilice el **SwiftUI Inspector** para cambiar el **Font** a **Title**. Esto aplica la fuente del sistema al texto para que responda correctamente a los tamaños y configuraciones de fuente preferidos por el usuario.

Paso 8: Para personalizar una **View** de SwiftUI, se utilizan métodos llamados **Modifiers**. Los **Modifiers** envuelven una **View** para cambiar su visualización u otras propiedades. Cada **Modifiers** devuelve una nueva **View**, por lo que es común encadenar varios apilados verticalmente. Edite el código a para cambiar el **Modifier padding()** por el **Modifier foregroundColor(.green)** esto cambia el color del texto a verde. (El código en el archivo debe ser

similar al que se muestra a continuación)

```
struct ContentView: View {
    var body: some View {
        Text("Turtle Rock")
            .font(.title)
            .foregroundColor(.green)
    }
}
```

Paso 9: Su código es siempre la fuente de la verdad para el **View**. Cuando usa el **SwiftUI Inspector** para cambiar o eliminar un **Modifier**, Xcode actualiza su código inmediatamente para que coincida. Esta vez, abra el **SwiftUI Inspector** presionando **Command-click** sobre la declaración **Text**, y luego elija **Show SwiftUI Inspector**. Sobre la propiedad **Color**, elija **Inherited**, el color de la letra nuevamente se presenta color negro y el **Modifier** ha desaparecido.

## 2.4 Combinar Views Usando Stacks

Utilizando el **Title** creado en el **View** anterior, vamos a agregar más detalles sobre el Parque Nacional. Al crear un **View** en SwiftUI, se describe su contenido, diseño y comportamiento en el **Property** del **Body**; sin embargo, la **Property** del **Body** un único **View**. Puede combinar e incrustar varias **View** en pilas, que agrupan en **Views** de forma horizontal, vertical o de atrás hacia adelante.

Paso 1: **Command-click** sobre el **Text** en el **ContentView.swift** para mostrar popup del editor de **Struck** y seleccione **Embed in VStack**. (ver figura 5)

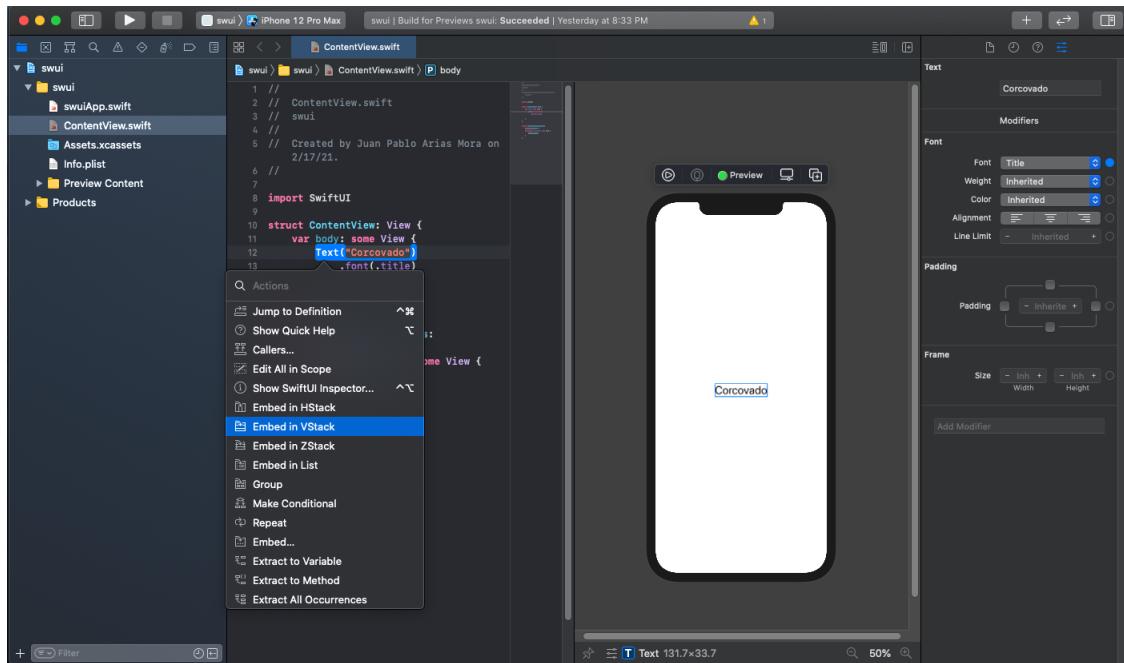


Figure 5: Agregando el V Stack

Paso 2: A continuación, agregue un **Text** desde el **Object Library**, justo por debajo del **Text** ya presente. Reemplace el **Text Placeholder** con "Parque Nacional". Cambie el **Font** a **Subheadline**.

Paso 3: Edite el **VStack initializer** para alinear el **View** a su derecha. (ver figura 6 área roja).

Paso 4: Nuevamente **Command-click** a **Parque Nacional** y elija **Embed in HStack**.

Paso 5: Dentro del nuevo **HStack** agregue un nuevo **Text** justo debajo del que contiene **Parque Nacional**, y modifique su **Placeholder** a **Puntarenas**. Cambie el **Font** a **Subheadline**. (ver figura 7).

Paso 6: Para que el diseño utilice todo el ancho del dispositivo, separe el **Parque Nacional** y el **Puntarenas** agregando un **Spacer()** entre ambos **Text**. Un **Spacer** se expande para que su **View** contenedora use todo el espacio de su **View** padre, en lugar de tener su tamaño definido solo por su contenido. (ver figura 8).

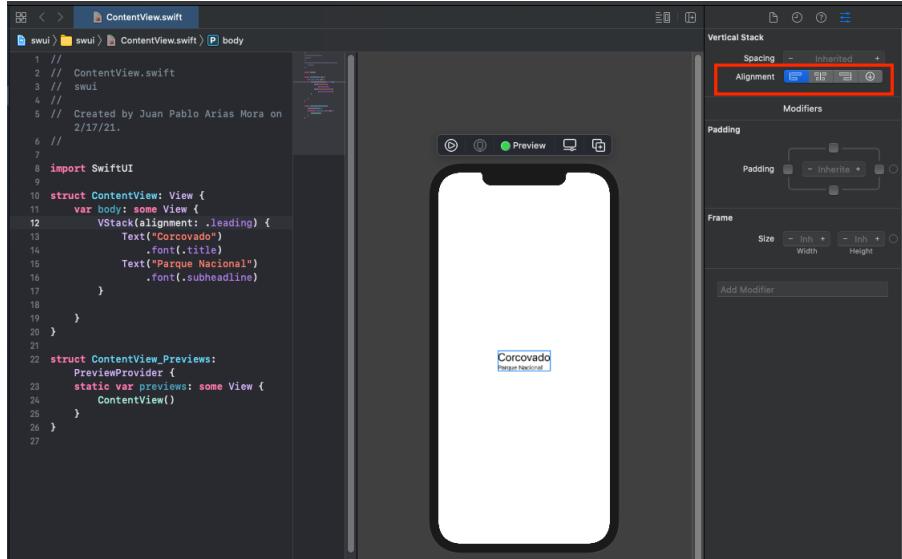


Figure 6: Alineamiento del V Stack

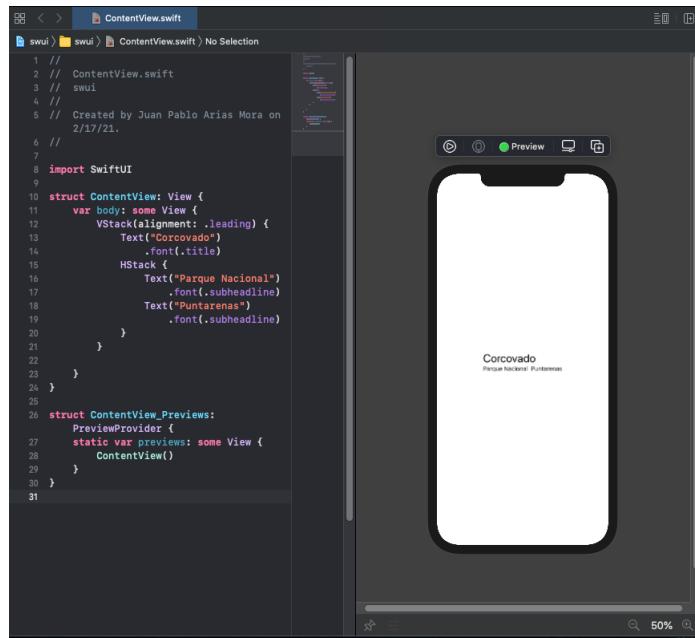


Figure 7: H Stack y Provincia

Paso 7: Agregue un **Modifier Padding()** para entregar un poco de espacio a los **Text** de los bordes, a nivel del **VStack**. (ver figura 9).

## 2.5 Agregar una Imagen

Paso 1: Abra la carpeta **Assets.xcassets** dentro del **Project navigator** y copie la carpeta **images** de los materiales que descargó al inicio del laboratorio. Como resultado debería observar algo similar a la figura 10.

Paso 2: Creamos un **View** nuevo **File - New - File**. En la sección de **User Interface** seleccione **SwiftUI View**. Asígnale de nombre **CircleImage.swift**.

Paso 3: Reemplace el **Text("Hello World!")** por **Image("corcovado")**. Como resultado debería observar algo similar a la figura 11.

Paso 4: Agregue una llamada a la propiedad **clipShape(Circle())** para agregar una forma circular a la imagen.

Paso 5: Agregaremos otro círculo con borde gris para dar a la imagen un efecto de delineado, justo después del **clipShape(Circle())**, utilizando el siguiente código:

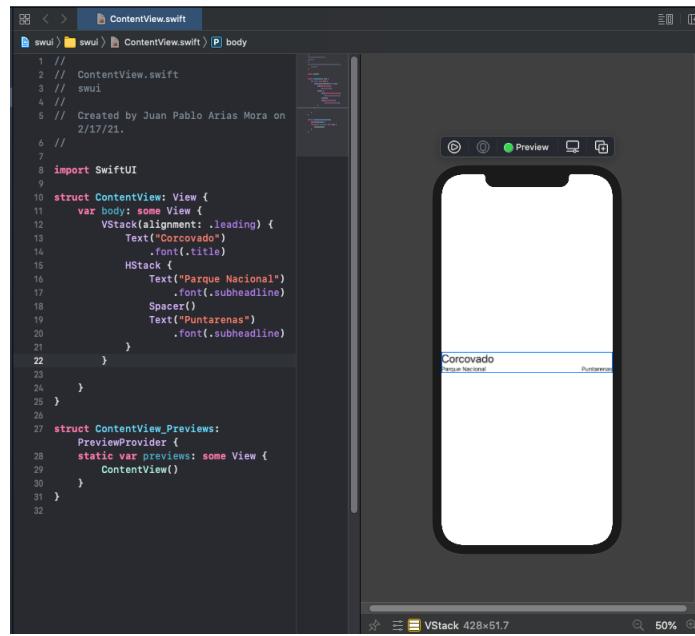


Figure 8: Efecto Spacer()

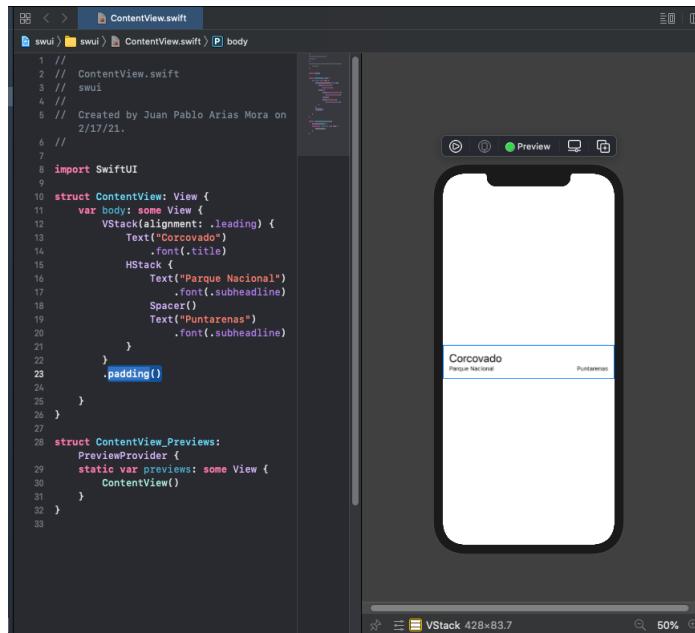


Figure 9: Efecto Padding()

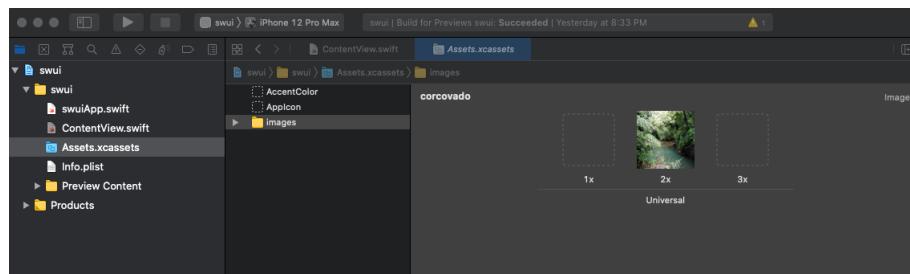


Figure 10: Imagen en la Galería

```
.overlay(Circle().stroke(Color.gray, lineWidth: 4))
```

Paso 6: Agregaremos una sombra, justo después del **overlay**, utilizando el siguiente código:

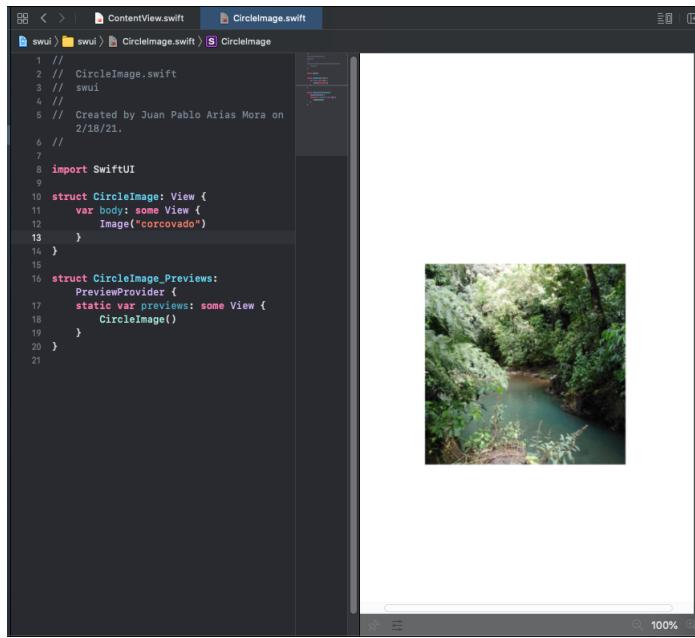


Figure 11: Primer View de la Imagen

```
.shadow(radius: 7)
```

Paso 7: Cambiemos el borde a color blanco, para dar un mejor efecto. Como resultado debería observar algo similar a la figura 12.

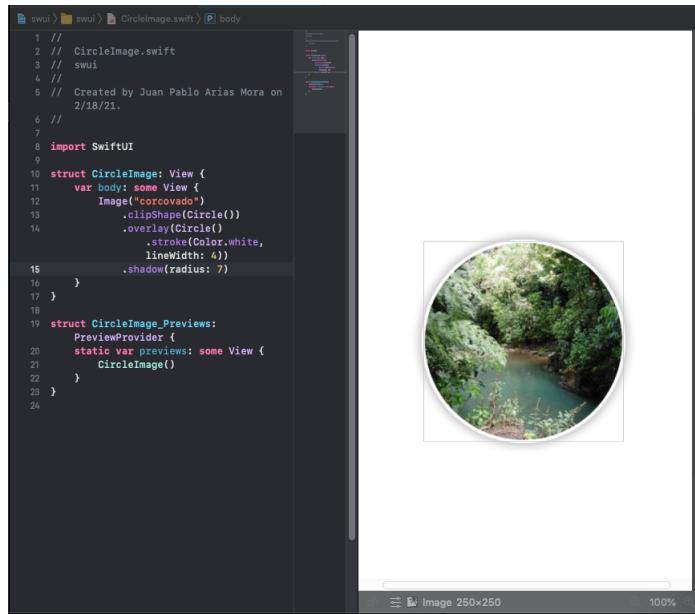


Figure 12: View de la Imagen

## 2.6 Views basados en otros Frameworks

Paso 8: Crearemos un nuevo **SwiftUI View** y le daremos por nombre **MapView.swift**. Como dato interesante cuando importa **SwiftUI** y algunos otros **frameworks** en el mismo archivo, obtiene acceso a la funcionalidad específica de **SwiftUI** proporcionada por ese **framework**.

Paso 9: Inmediatamente después del **import SwiftUI** agregaremos el siguiente código:

```
import MapKit
```

Paso 10: Cree una **Private State Variable** la cual llevara la información de la región del mapa. Agregando el siguiente código a la vista

```
@State private var region = MKCoordinateRegion(  
    center: CLLocationCoordinate2D(latitude: 8.540_800, longitude: -83.571_000),  
    span: MKCoordinateSpan(latitudeDelta: 0.2, longitudeDelta: 0.2)  
)
```

Paso 11: Reemplace el **Text("Hello World!")** por **Map(coordinateRegion: \$region)**. Al anteponer una variable de estado con \$, pasa un enlace, que es como una referencia al valor subyacente. Cuando el usuario interactúa con el mapa, el mapa actualiza el valor de la región para que coincida con la parte del mapa que está visible actualmente en la interfaz de usuario.

Paso 12: Click sobre **Resume** en el **Preview**. Como resultado debería observar algo similar a la figura 13.

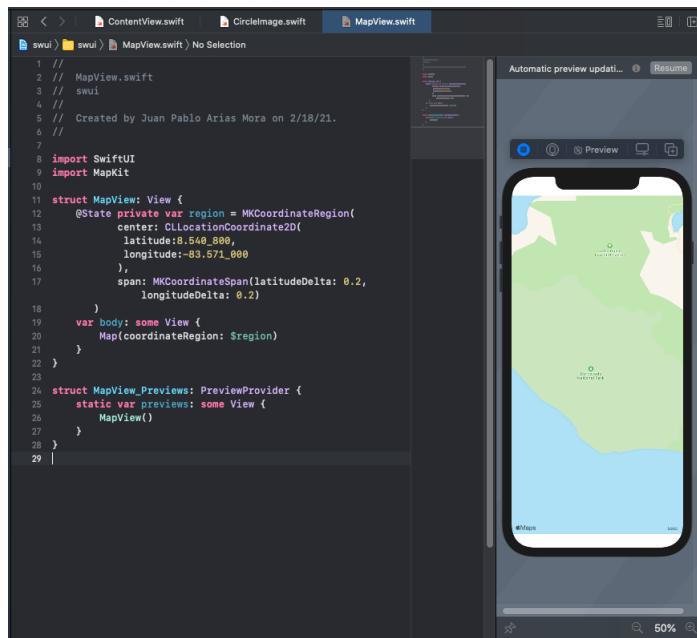


Figure 13: View del mapa

## 2.7 Unificando Views

Paso 13: En el **Project Navigator**, seleccione el archivo **ContentView.swift**. Seleccionando el **VStack** agregamos el mismo dentro de otro **VStack** ( ver figura 14)

Paso 14: Como dato importante: Cuando especifica solo el parámetro de **Height**, los **View** se ajusta automáticamente al **Width** de su contenido. En este caso, **MapView** se expande para llenar el espacio disponible. Agregue el siguiente código justo antes del segundo **VStack**

```
MapView()  
.frame(height: 300)
```

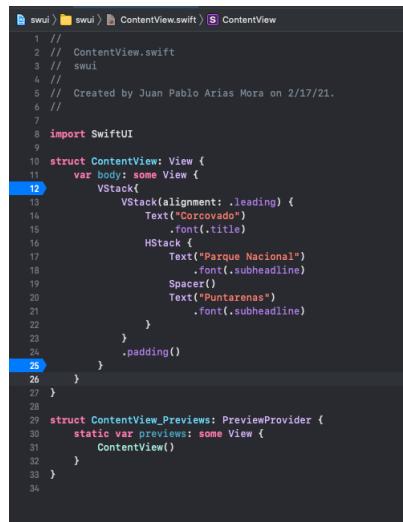
Paso 15: Si desea puede darle al **Live Preview** y ver los resultados. (ver figura 15).

Paso 16: Ahora agreguemos el **View** que contiene la Imagen. Justo debajo del **MapView**, utilizando el siguiente código.

```
CircleImage()
```

Paso 17: Si desea puede darle al **Live Preview** y ver los resultados. (ver figura 16).

Paso 18: En este momento sobrepondremos la imagen sobre el mapa, utilizando las propiedades de **offset** y **padding** de **CircleImage**, con el siguiente código (ver figura 17):



```

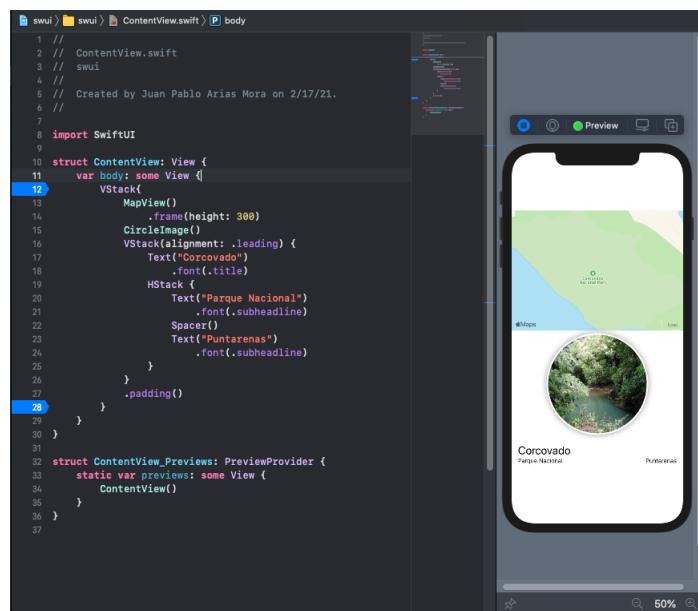
1 // Content View
2 // swui
3 // Created by Juan Pablo Arias Mora on 2/17/21.
4 //
5 import SwiftUI
6
7 struct ContentView: View {
8     var body: some View {
9         VStack {
10             VStack(alignment: .leading) {
11                 Text("Corcovado")
12                     .font(.title)
13                 HStack {
14                     Text("Parque Nacional")
15                         .font(.subheadline)
16                     Spacer()
17                     Text("Puntarenas")
18                         .font(.subheadline)
19                 }
20             }
21         }
22     }
23 }
24
25 struct ContentView_Previews: PreviewProvider {
26     static var previews: some View {
27         ContentView()
28     }
29 }
30
31 }
32
33 }
34

```

Figure 14: VStack dentro de otro VStack



Figure 15: Live Preview con MapView



```

1 // Content View
2 // swui
3 // Created by Juan Pablo Arias Mora on 2/17/21.
4 //
5 import SwiftUI
6
7 struct ContentView: View {
8     var body: some View {
9         VStack {
10             MapView()
11                 .frame(height: 300)
12                 .CircleImage()
13             VStack(alignment: .leading) {
14                 Text("Corcovado")
15                     .font(.title)
16                 HStack {
17                     Text("Parque Nacional")
18                         .font(.subheadline)
19                     Spacer()
20                     Text("Puntarenas")
21                         .font(.subheadline)
22                 }
23             }
24         }
25     }
26 }
27
28 struct ContentView_Previews: PreviewProvider {
29     static var previews: some View {
30         ContentView()
31     }
32 }
33
34 }
35
36 }
37

```

Figure 16: Live Preview con CircleImage

```

CircleImage()
.offset(y: -130)

```

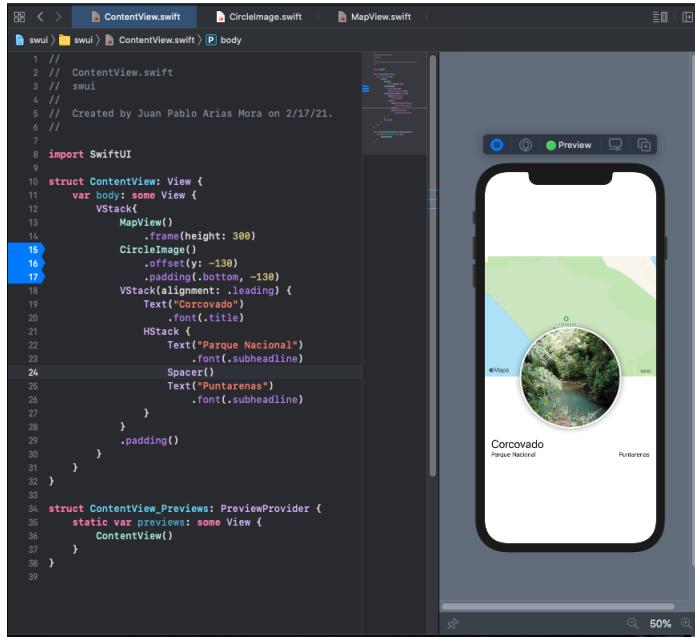


Figure 17: Live Preview con CircleImage sobrepuerto

```
.padding(.bottom, -130)
```

Paso 19: Utilicemos nuevamente un **Spacer()** para desplazar nuestro **VStack** padre más arriba en la pantalla. Este lo colocamos después del **padding()**

Paso 20: Agreguemos mas información del parque, esto para mostrar otros objetos que podemos utilizar, justo debajo del **HStack** agreguemos el siguiente código (ver resultado en figura 18):

```
Divider()
Text("Acerca de...")
    .font(.title2)
Text("Fue creado el 24 de octubre de 1975")
```

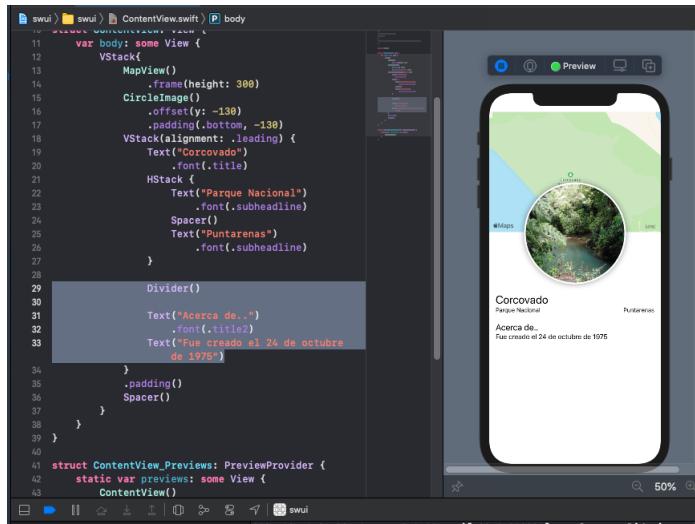


Figure 18: Usando Divider

Paso 21: Remueva ahora todas las apariciones de **.font(.subheadline)** de su actual código, y coloque el siguiente código justo debajo del **HStack**. (ver figura 19)

```
.font(.subheadline)
.foregroundColor(.secondary)
```

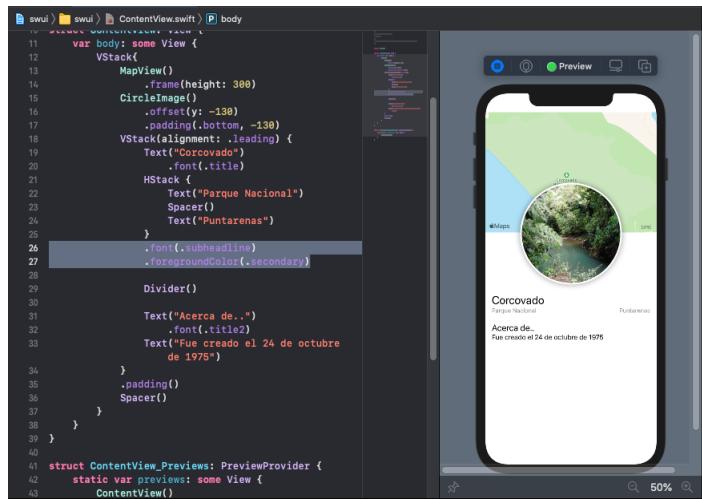


Figure 19: Herencia de Modifiers

Cuando se aplica un **Modifier** a un **Layout View** como lo es **Stack**, **SwiftUI** aplica el **Modifier** a todos los elementos contenidos dentro del grupo.

Paso 22: Por ultimo compile y ejecute la aplicación, realice movimientos sobre el área del mapa de manera tal que pueda ver los alrededores.