



App que hace un cálculo en Swift UI: Divisas

Yolanda Martínez Treviño

Descripción de la actividad

- Realizarás una aplicación para convertir de pesos a dólares, el usuario tecleará el tipo de cambio del día (la cantidad de pesos por dólar) y la cantidad de pesos que quiere convertir. La aplicación calculará la cantidad de dólares.
- Es necesario que la aplicación:
 - verifique que los campos tienen datos y si no es así que mande un mensaje de alerta para indicarlo y no truene.
 - use teclado numérico para ambos campos.
 - tenga una imagen en alguna parte de la ventana y use otra imagen para el botón.
 - El campo dólares no debe permitir la interacción con el usuario.
 - Que el teclado se quite al dar tap fuera de los campos de texto.

Vistas de la aplicación



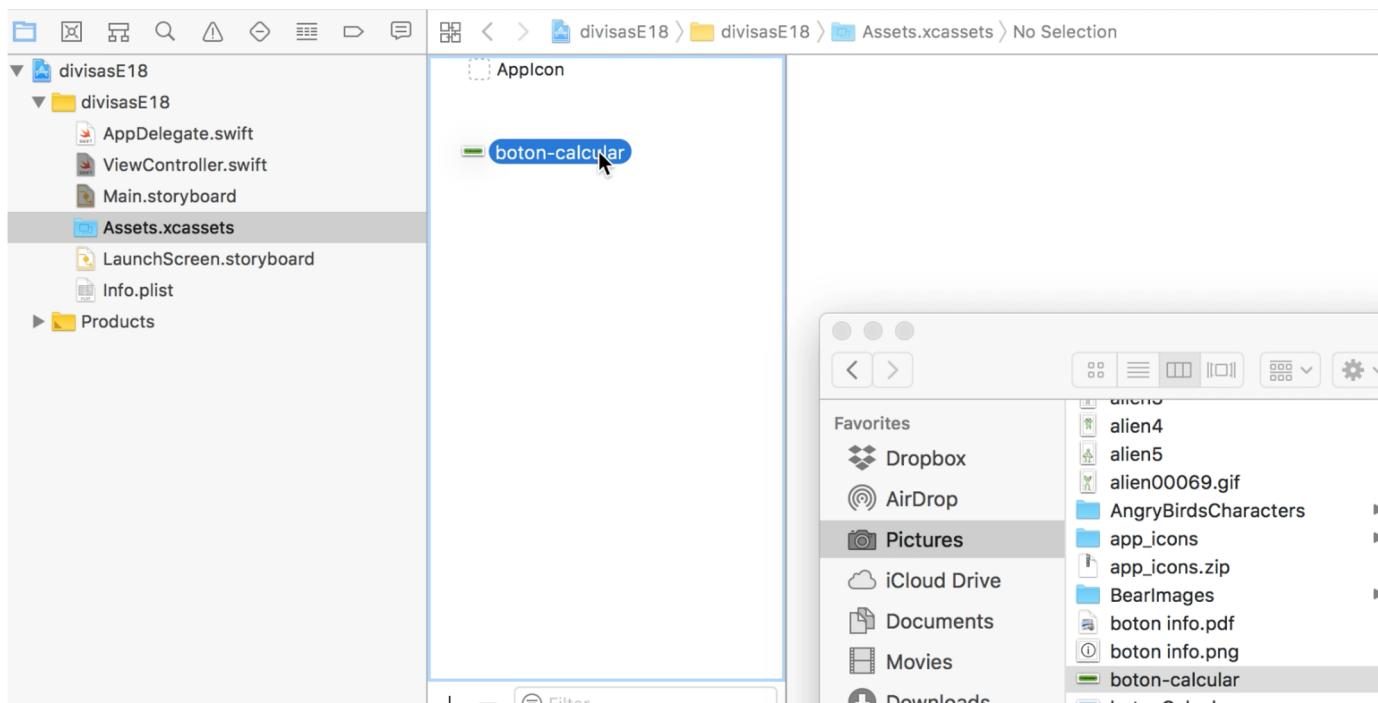
iPhone 11 Pro Max — 13.3



iPhone 11 Pro Max — 13.3

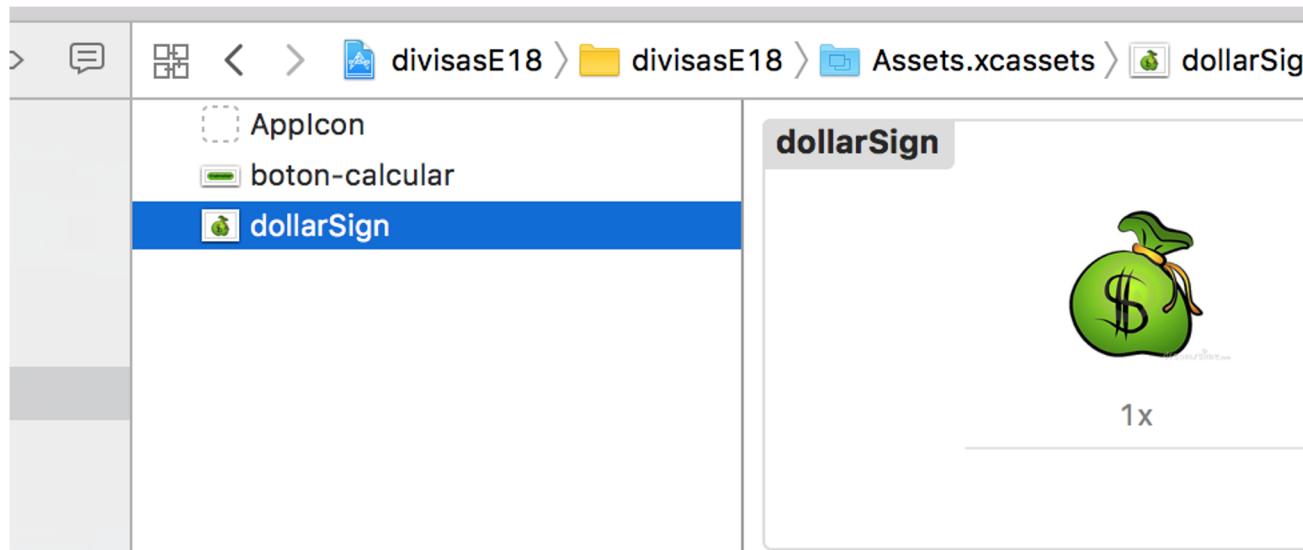
Agrega las imágenes

- Arrastra la imagen desde el finder a la zona en la que van las imágenes.

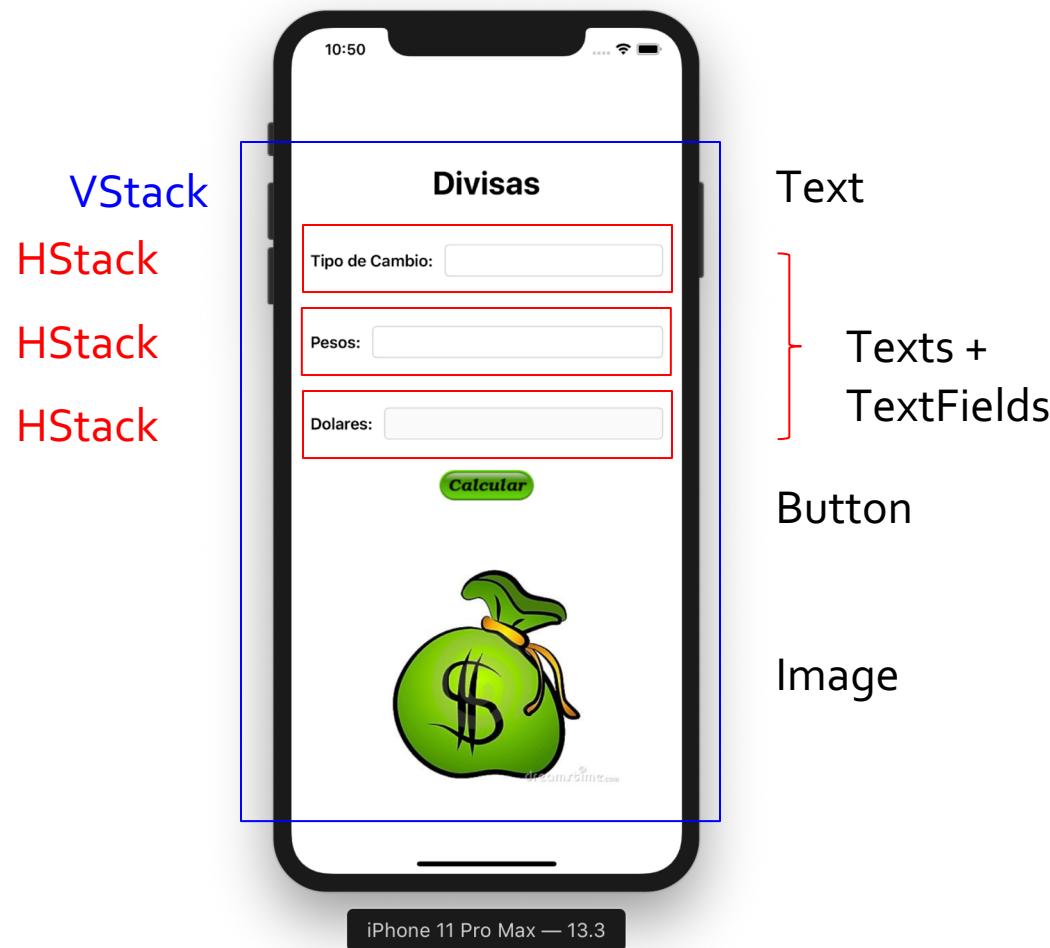


Agrega imágenes

- Agrega la imagen del signo de dólares y la del botón



Crea la interfaz visual con Stacks, Texts, TextFields, un Button y una Image



Para usar una imagen como botón...

- Reemplaza el **Text** dentro del Button con una **Image** y pasa como parámetro el nombre de la imagen como aparece en el Assets folder

```
Button(action: {}) {  
    Image("boton-calcular")  
}
```

Para la imagen...

- Para que la imagen pueda tomar diferentes tamaños de acuerdo a la pantalla, usa el modifier **resizable()**
- Además, usa el modifier **aspectRatio()** con parámetro **.fit** para que la imagen preserve sus proporciones y no se estire
- Si quieres darle un tamaño fijo, usa el modifier **frame()**

```
Image("dollarSign")
    .resizable()
    .aspectRatio(contentMode: .fit)
    .padding(60)
```

Configura los campos de texto

Asigna a cada TextField una variable @State apropiada

```
TextField("", text: $tipoCambio)  
    .keyboardType(.decimalPad)  
    .textFieldStyle(RoundedBorderTextFieldStyle())
```

Usa el modifier **keyboardType()** para especificar el teclado a desplegar

Configura los campos de texto

```
TextField("", text: $dolares)  
    .disabled(true)  
    .textFieldStyle(RoundedBorderTextFieldStyle())
```

Además, desactiva interacción con usuario para el último TextField usando el modifier **disabled()**

Completa el método de acción

- Si los campos de texto tienen dato haz el cálculo

```
Button(action: {  
    if let tipoCambioD = Double(self.tipoCambio),  
        let pesosD = Double(self.pesos) {  
            self.dolares = String(tipoCambioD / pesosD)  
    }  
}
```

Para mandar una alerta

- Agrega el modifier **alert()** al botón (después de los brackets que encierran la imagen) y quita el parámetro content

```
.alert(item: Binding<Identifiable?>, content: (Identifiable) -> Alert)
```

Para item, el modifier puede tomar un @State var de tipo Bool, que indica si la alerta se puede ver o no

```
.alert(isPresented: $alertIsVisible)
```

Crea la variable @State y lígala por medio de two-way binding

Para mandar una alerta

- Agregaremos el parámetro content, usando un **closure**
- Un closure es una función sin nombre
- Su sintaxis es la siguiente:

```
{ (parameters) -> return type in  
    statements  
}
```

Para mandar una alerta

- El closure que alert espera es una función que retorna una instancia de una alerta:

No recibimos parámetros

Se espera un objeto de tipo Alert

```
.alert(isPresented: $alertIsVisible) { () -> Alert in
    return Alert(title: Text("Error"), message: Text("Los datos no son
        adecuados"), dismissButton: .default(Text("OK")))
}
```

- Los parámetros para Alert son el texto del título y mensaje, así como el botón para cerrar la alerta.

Para mandar una alerta

- El código se puede simplificar:
 - Swift puede inferir el valor de retorno
 - Si el código solamente contiene una expresión, Swift infiere que es un return
 - Se puede omitir la lista de parámetros y la palabra 'in'
 - Nuestro método no tiene parámetros, pero en caso de tenerlos también se puede omitir enlistarlos. Dentro del cuerpo de la función el primer parámetro recibe el identificador \$0, el segundo \$1, tercero \$2, etc.

```
.alert(isPresented: $alertIsVisible) {
    Alert(title: Text("Error"), message: Text("Los datos no son adecuados"),
        dismissButton: .default(Text("OK")))
}
```

Quitar el teclado

- Envuelve el VStack en un **ZStack** y agrega una vista invisible (o blanca) en el fondo.

```
var body: some View {  
    ZStack {  
        Color(.white)  
        VStack {  
            Text("Divisas")  
        }  
    }  
}
```

- Y agrega la siguiente extensión a UIApplication

```
extension UIApplication {  
    func endEditing() {  
        sendAction(#selector(UIResponder.resignFirstResponder), to: nil, from: nil, for: nil)  
    }  
}
```

Quitar el teclado

- Finalmente, agrégale al ZStack el modifier `onTapGesture()` y llama a `endEditing()`

```
.onTapGesture(count: 1, perform: {  
    UIApplication.shared.endEditing()  
})
```

- De esta manera, presionar en cualquier elemento dentro del ZStack (incluyendo los espacios vacíos que tienen de fondo una vista Color) hará llamar a `endEditing()`, que removerá el teclado.