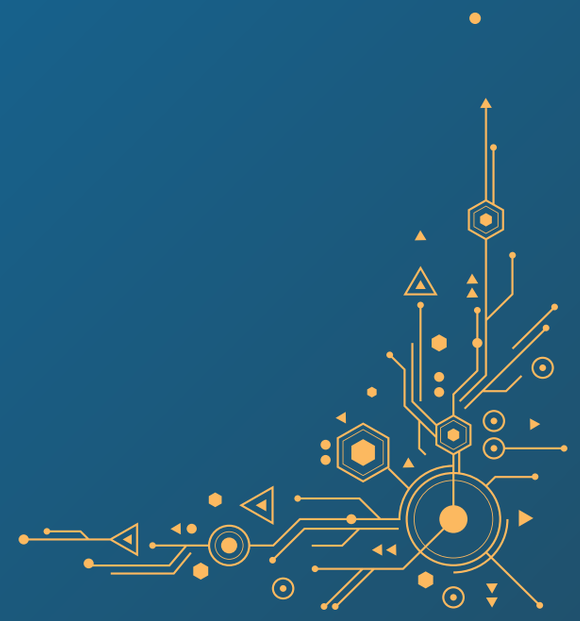
Desarrollo de aplicaciones para iOS con SwiftUI

Alberto Guerrero Martín

ConfIT Talks 2 de diciembre de 2021 confit.es/talks



¿Quién soy?

@alberto170693

albertoguerreromartin

albertoguerreromartin

Alberto Guerrero Martín

Estudios

- Graduado en Ingeniería del Software por la ETSISI de la UPM.
- Máster en Desarrollo de Aplicaciones y Servicios para Dispositivos Móviles por la ETSISI de la UPM.
- Representante de estudiantes durante casi 10 años.
    - Delegado de Alumnos de la ETSISI 2013-2015.
    - Delegado de Alumnos de la UPM 2017-2018.

Experiencia laboral

- Más de 9 años desarrollando aplicaciones para dispositivos y tecnologías Apple:
    - ETSISI.
    - bq.
    - Atresmedia.
- Multitud de proyectos como freelance.
- Actualmente trabajando en Gigigo (Econocom):
    - Wible.
    - Vivit.
    - McDonalds LATAM.

[CONF IT]

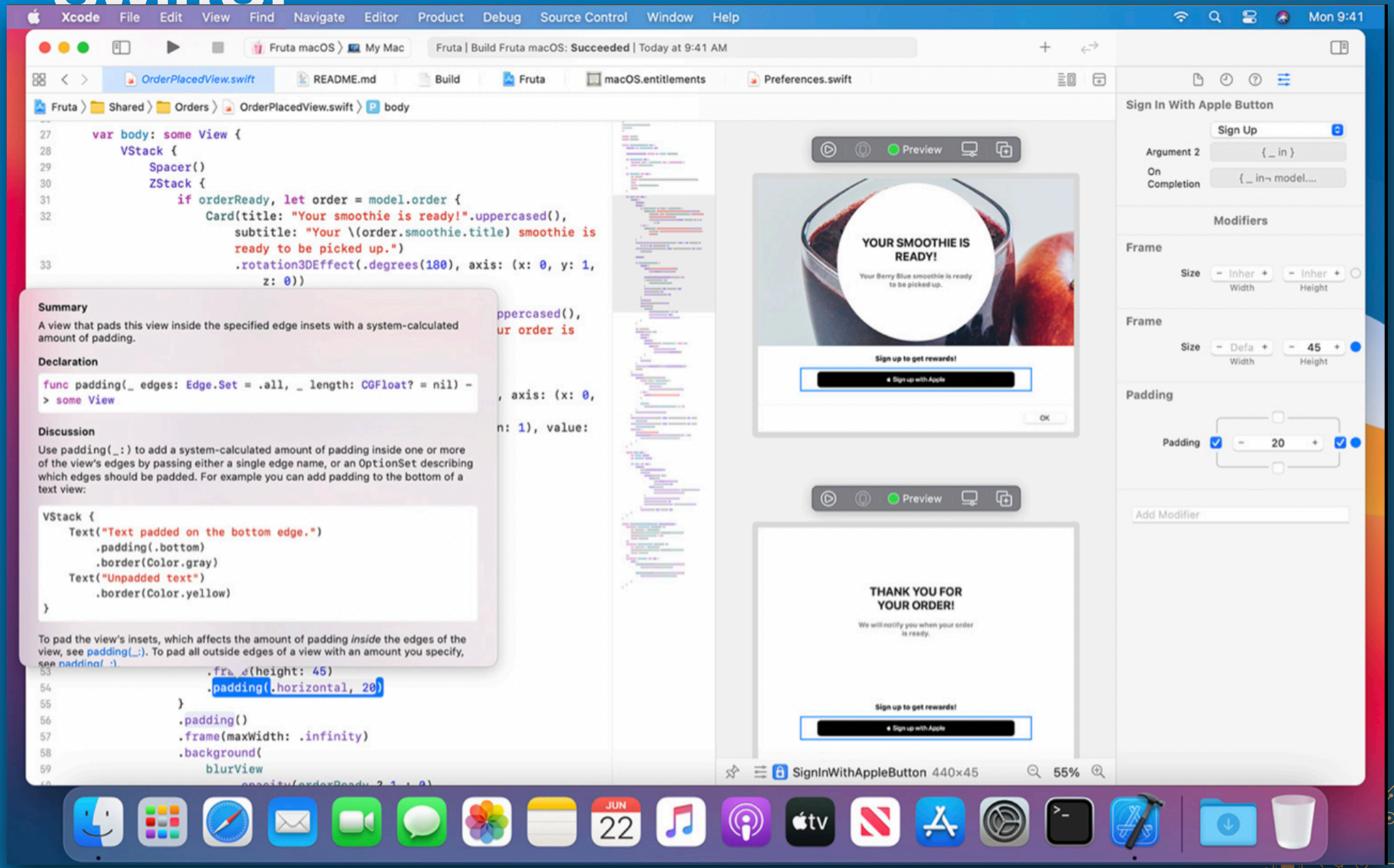
SwiftUI

SwiftUI es un framework para creación de vistas de forma declarativa.

- Diseñado por y para Swift.
- Inspirado en la gestión de estados y la reutilización de componentes de React.
- iOS 13+.
- Compatible con UIKit.

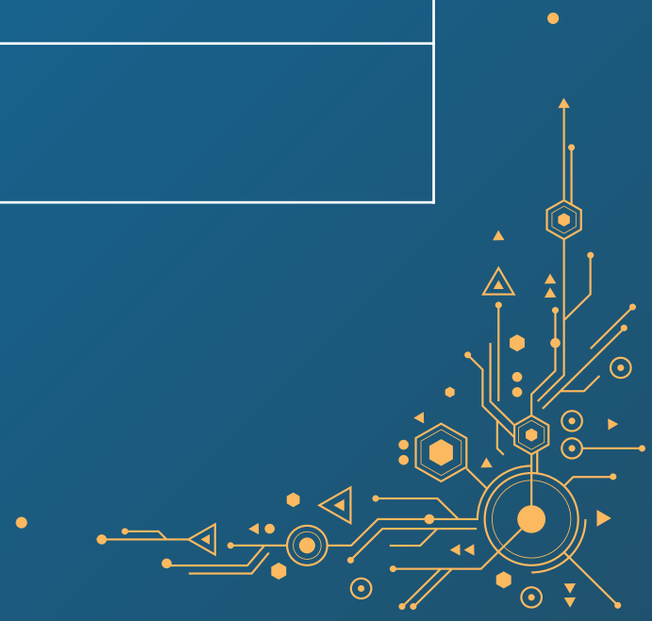
```
import SwiftUI

struct Content : View {

    @State var model = Themes.listModel

    var body: some View {
        List(model.items, action: model.selectItem) { item in
            Image(item.image)
            VStack(alignment: .leading) {
                Text(item.title)
                Text(item.subtitle)
                    .color(.gray)
            }
        }
    }
}
```

SwiftUI


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Mon 9:41
Fruta macOS My Mac
Fruta | Build Fruta macOS: Succeeded | Today at 9:41 AM
OrderPlacedView.swift
README.md
Build
Fruta
macOS.entitlements
Preferences.swift
Fruta Shared Orders OrderPlacedView.swift body
var body: some View {
VStack {
Spacer()
ZStack {
if orderReady, let order = model.order {
Card(title: "Your smoothie is ready!".uppercased(),
subtitle: "Your \(order.smoothie.title) smoothie is
ready to be picked up.")
.rotation3DEffect(.degrees(180), axis: (x: 0, y: 1,
z: 0))
Summary
A view that pads this view inside the specified edge insets with a system-calculated amount of padding.
Declaration
func padding(_ edges: Edge.Set = .all, _ length: CGFloat? = nil) -> some View
Discussion
Use padding(_:) to add a system-calculated amount of padding inside one or more of the view's edges by passing either a single edge name, or an OptionSet describing which edges should be padded. For example you can add padding to the bottom of a text view:
VStack {
Text("Text padded on the bottom edge.")
.padding(.bottom)
.border(Color.gray)
Text("Unpadded text")
.border(Color.yellow)
}
To pad the view's insets, which affects the amount of padding inside the edges of the view, see padding(_:). To pad all outside edges of a view with an amount you specify, see padding(_:).
.frame(height: 45)
.padding(.horizontal, 20)
}
.padding()
.frame(maxWidth: .infinity)
.background(
blurView
Preview
YOUR SMOOTHIE IS READY!
Your Berry Blue smoothie is ready to be picked up.
Sign up to get rewards!
Sign up with Apple
OK
THANK YOU FOR YOUR ORDER!
We will notify you when your order is ready.
Sign up to get rewards!
Sign up with Apple
SignInWithAppleButton 440×45
55%
Sign In With Apple Button
Sign Up
Argument 2
{ _ in }
On Completion
{ _ in¬ model....
Modifiers
Frame
Size
Inher
Width
Height
Frame
Size
Defa
45
Padding
20
Add Modifier

SwiftUI vs UIKit & Autolayout

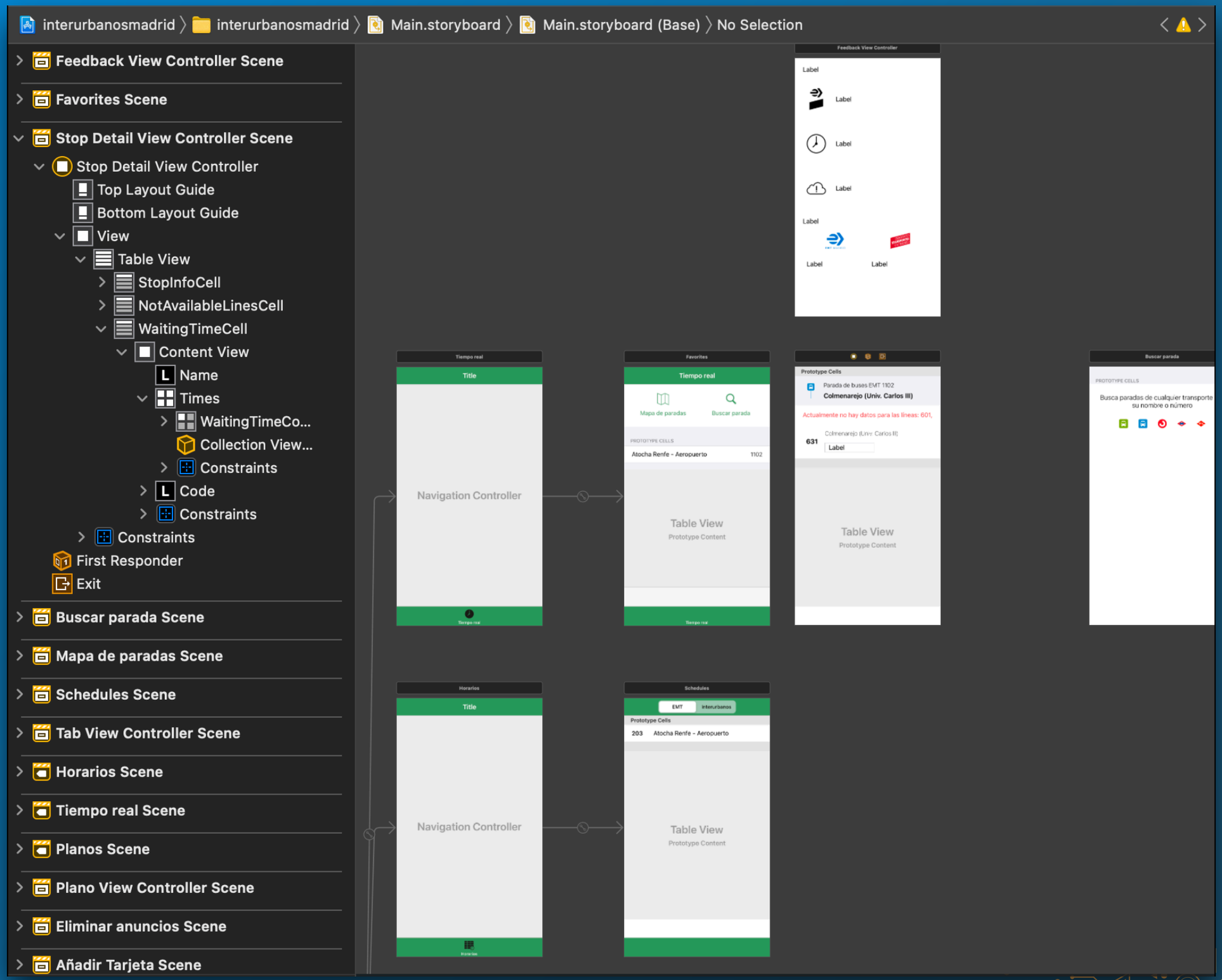
UIKit	SwiftUI
Imperativo	Declarativo
Requiere de **gestionar el estado concreto** en el que se está, y **reflejar manual y correctamente los cambios** en la UI.	**Se definen todos los posibles estados a la vez**. SwiftUI **gestiona automáticamente la actualización de la UI** cuando se producen cambios en el estado.
Rendimiento ⬆️	Rendimiento ⬆️⬆️⬆️
Interoperabilidad ✅	



SwiftUI vs IB & Storyboards

IB & Storyboards	SwiftUI
XML gigantes: legibilidad ❌	100% código Swift: legibilidad ✅ ✅
Difícil (por no decir imposible) de editar a mano	Se edita a mano por defecto
Muchas pantallas en un único archivo (Storyboards)	Una pantalla por archivo
Difícil reutilizar componentes de UI	Orientado a componentes, y diseñado para que reutilizarlos sea lo habitual
Básicamente imposible analizar cambios en GIT	100% código Swift: legibilidad ✅ ✅
IB desconoce el código en Swift	La UI se comprueba y compila constántemente
Solo se puede previsualizar una aproximación de las pantallas en IB	SwiftUI previews: permiten ejecutar y probar de forma precisa la UI (o partes de ella) directamente en el IDE

SwiftUI vs IB & Storyboards

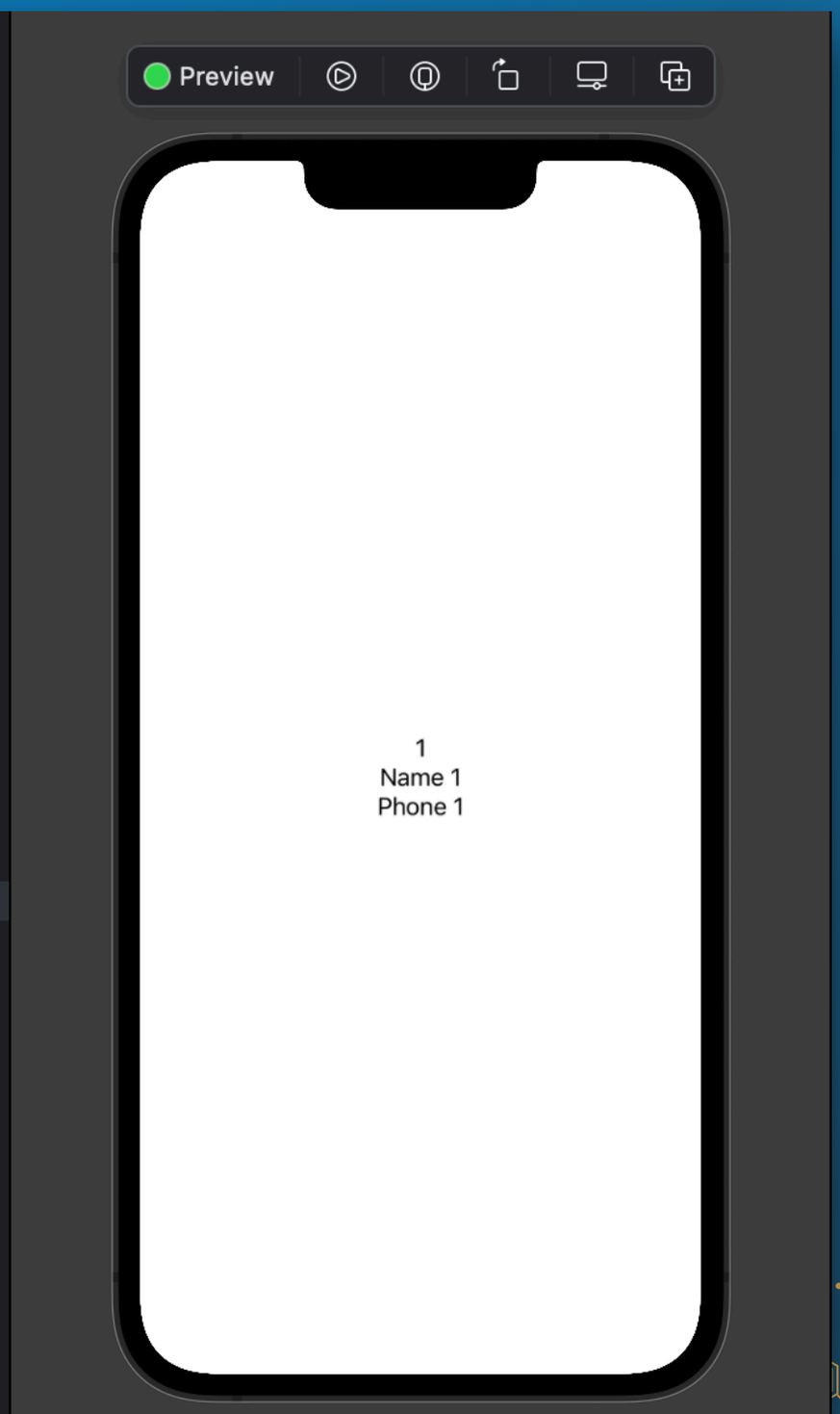


SwiftUI vs IB & Storyboards

```
<?xml version="1.0" encoding="UTF-8"?>
<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="19529" targetRuntime="iOS.CocoaTouch"
 propertyAccessControl="none" useAutolayout="YES" useTraitCollections="YES" colorMatched="YES" initialViewController="49e-Tb-3d3">
    <device id="retina4_7" orientation="portrait" appearance="light"/>
    <dependencies>
        <deployment version="4400" identifier="iOS"/>
        <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="19519"/>
        <capability name="Named colors" minToolsVersion="9.0"/>
        <capability name="System colors in document resources" minToolsVersion="11.0"/>
        <capability name="collection view cell content view" minToolsVersion="11.0"/>
        <capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>
    </dependencies>
    <scenes>
        <!--Feedback View Controller-->
        <scene sceneID="hOz-Bw-yff">
            <objects>
                <viewController storyboardIdentifier="FeedbackViewController" id="YrM-98-Vn8" customClass="FeedbackViewController"
                customModule="interurbanosmadrid" customModuleProvider="target" sceneMemberID="viewController">
                    <layoutGuides>
                        <viewControllerLayoutGuide type="top" id="yzd-ta-avp"/>
                        <viewControllerLayoutGuide type="bottom" id="efT-4R-uaR"/>
                    </layoutGuides>
                    <view key="view" contentMode="scaleToFill" id="iFc-py-wml">
                        <rect key="frame" x="0.0" y="0.0" width="375" height="667"/>
                        <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
                        <subviews>
                            <scrollView clipsSubviews="YES" multipleTouchEnabled="YES" contentMode="scaleToFill"
                            translatesAutoresizingMaskIntoConstraints="NO" id="YGh-Do-5Ex">
                                <rect key="frame" x="0.0" y="0.0" width="375" height="619"/>
                                <subviews>
                                    <view contentMode="scaleToFill" translatesAutoresizingMaskIntoConstraints="NO" id="jZr-Mt-ZH7">
                                        <rect key="frame" x="0.0" y="0.0" width="375" height="623"/>
                                        <subviews>
                                            <label opaque="NO" userInteractionEnabled="NO" contentMode="left" horizontalHuggingPriority=
                                            "251" verticalHuggingPriority="251" text="Label" textAlignment="natural" lineBreakMode="
                                            tailTruncation" numberOfLines="0" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="
                                            NO" translatesAutoresizingMaskIntoConstraints="NO" id="5U6-UM-kXC">
                                                <rect key="frame" x="20" y="20" width="335" height="20.5"/>
                                                <fontDescription key="fontDescription" type="system" pointSize="17"/>
                                                <nil key="textColor"/>
                                                <nil key="highlightedColor"/>
                                            </label>
                                            <stackView opaque="NO" contentMode="scaleToFill" axis="vertical" spacing="15"
                                            translatesAutoresizingMaskIntoConstraints="NO" id="ybY-7M-ZEI">
```

SwiftUI vs IB & Storyboards

```swift
import SwiftUI

struct ContactDetailView: View {

    let contact: Contact

    var body: some View {
        VStack {
            Text(contact.id)
            Text(contact.name)
            Text(contact.phone)
        }
    }
}

struct ContactDetailView_Previews: PreviewProvider {
    static var previews: some View {
        ContactDetailView(contact: Contact(id: "1",
                                           name: "Name 1",
                                           phone: "Phone 1"))
    }
}
```



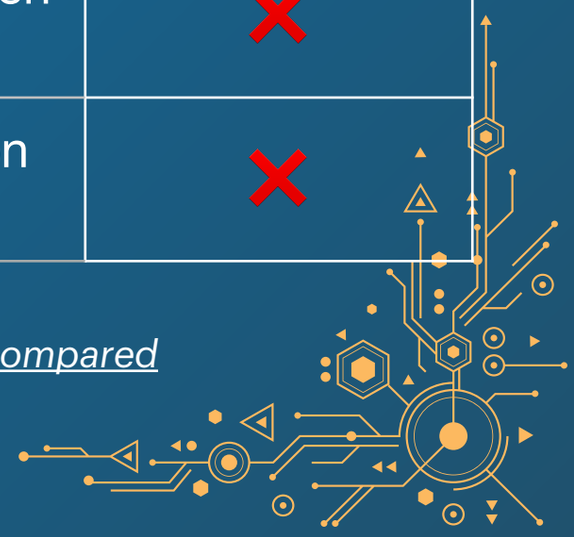
Equivalencias entre UIKit y SwiftUI

- `UITableView`: `List`
- `UICollectionView`: `LazyVGrid` and `LazyHGrid`
- `UILabel`: `Text`
- `UITextField`: `TextField`
- `UITextField` with `isSecureTextEntry` set to true: `SecureField`
- `UITextView`: `TextEditor` (plain strings only)
- `UISwitch`: `Toggle`
- `UISlider`: `Slider`
- `UIButton`: `Button`
- `UINavigationController`: `NavigationView`
- `UIAlertController` with style `.alert`: `Alert`
- `UIAlertController` with style `.actionSheet`: `ActionSheet`
- `UIStackView` with horizontal axis: `HStack`
- `UIStackView` with vertical axis: `VStack`
- `UIImageView`: `Image`
- `UISegmentedControl`: `Picker`
- `UIStepper`: `Stepper`
- `UIDatePicker`: `DatePicker`
- `UIProgressView`: `ProgressView` with a value
- `UIActivityIndicatorView`: `ProgressView` without a value
- `NSAttributedString`: Incompatible with SwiftUI; use `Text` instead.

Property wrappers

Annotation	Description	¿Source of truth?
@State	To manipulate value type data locally.	✅
@StateObject	To manipulate reference type data (must implemente ObservableObject protocol) locally.	✅
@Binding	Value type data owned by a different view. Propagates changes.	❌
@ObservedObject	Instance of an external class that conforms to the ObservableObject protocol	❌
@Published	Combine type for interoperate with SwiftUI views, and produce updates.	✅
@EnvironmentObject	For global singleton objects. Maintains reference between views.	❌
@Environment	It allows reading environment data such as presentation mode, trait collections, accessibility options, etc.	❌

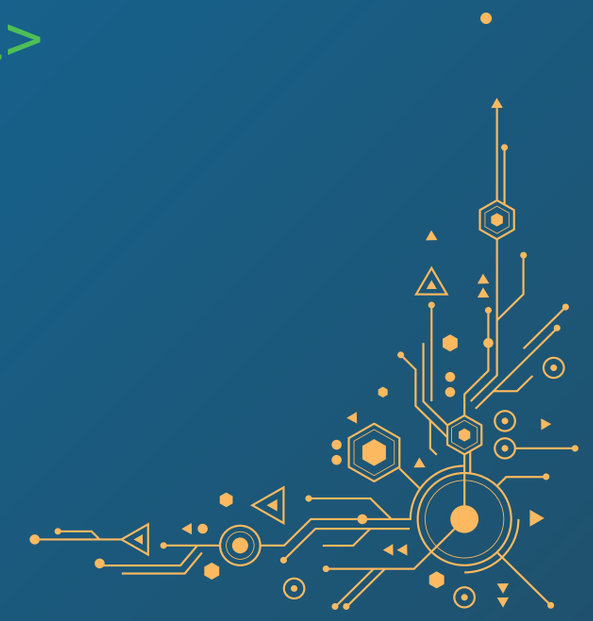
hackingwithswift.com/quick-start/swiftui/all-swiftui-property-wrappers-explained-and-compared



Two-way bindings

En SwiftUI, los property wrappers utilizan el operador **binding** ($) para escribir y leer valores de una variable y, a la vez, reaccionar a los cambios en esa variable, actualizando la vista.

```
@State var aState = false


aState        // Returns a Bool value
$aState       // Returns a Binding<Bool>
```



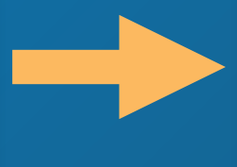
Vistas, controles y navegación

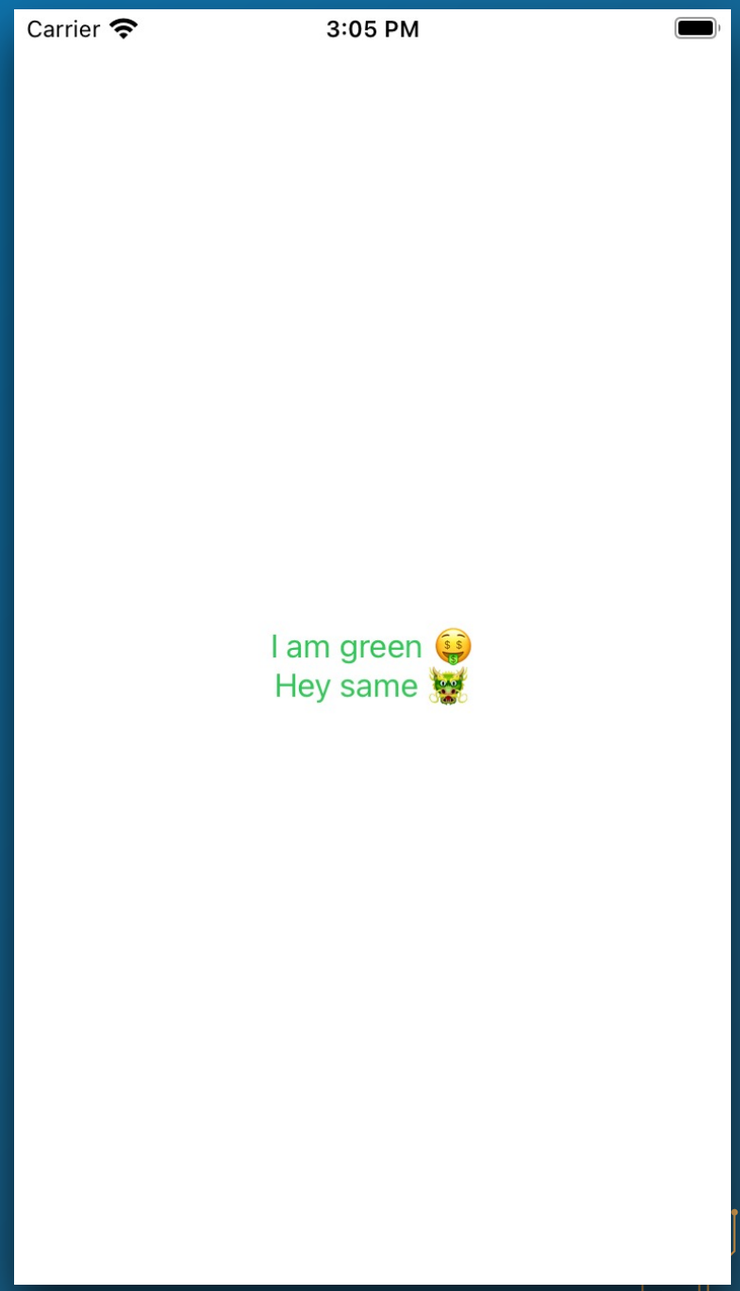
Layout de vistas	Controles	Navegación	Drawing & animation
View	Text & TextField	NavigationView	Shapes
HStack, VStack & ZStack	Button	TabView	Animations
ScrollView & List	Images	Modals	

developer.apple.com/documentation/swiftui/

ViewBuilders

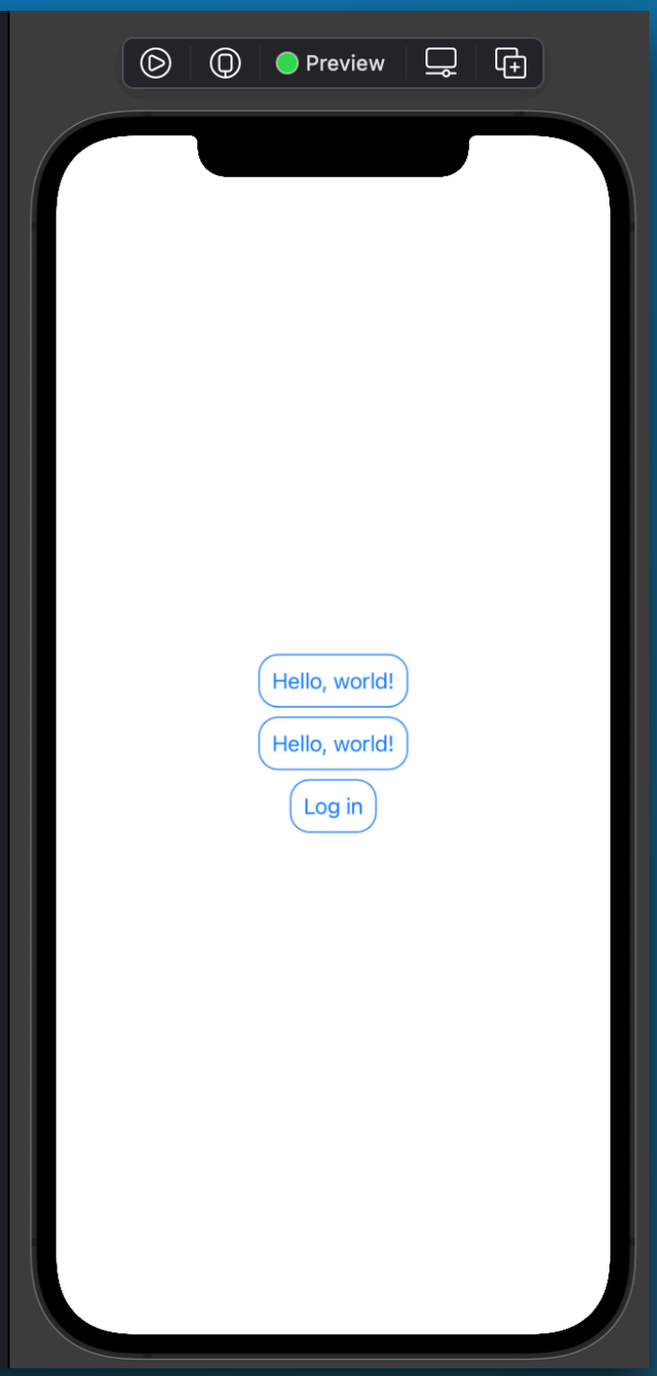
```
struct ContentView: View {
    var body: some View {
        GreenGroup {
            Text("I am green 🤑")
            Text("Hey same 🐲")
        }
    }
}

struct GreenGroup<Content>: View where Content: View {
    var views: Content

    init(@ViewBuilder content: () -> Content) {
        self.views = content()
    }

    var body: some View {
        Group {
            views.foregroundColor(.green)
        }
    }
}
```





ViewModifiers

```swift
import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {
            Text("Hello, world!")
                .modifier(BorderedCaption())

            Text("Hello, world!")
                .blueBordered()

            Button("Log in") {}
                .blueBordered()
        }
    }
}

struct BorderedCaption: ViewModifier {
    func body(content: Content) -> some View {
        content
            .padding(10)
            .overlay(
                RoundedRectangle(cornerRadius: 15)
                    .stroke(lineWidth: 1)
            )
            .foregroundColor(Color.blue)
    }
}

extension View {
    func blueBordered() -> some View {
        modifier(BorderedCaption())
    }
}
```

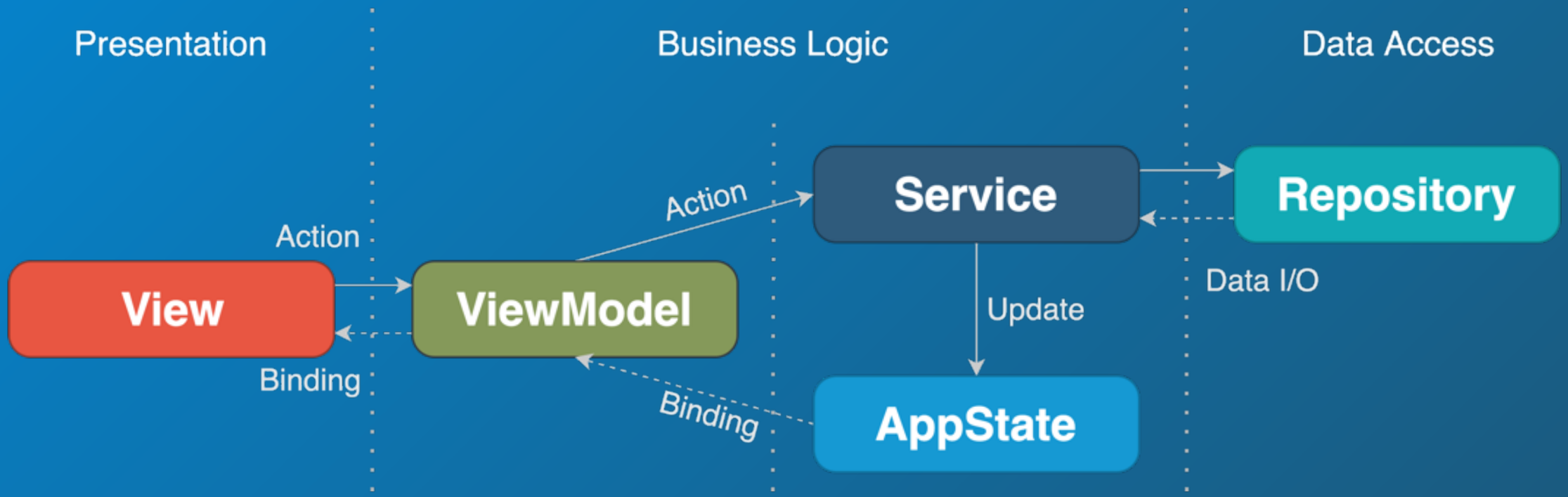


developer.apple.com/documentation/swiftui/viewmodifier

github.com/CongresoConfIT/swift-ui

Architecture

MVVM



Redux-like alternative: https://www.pointfree.co/collections/composable-architecture

Testing

Quick, Nimble & **ViewInspector**

```
let usernameTextField = try
    self.view.inspect().findAll(CustomTextField.self)[0].actualView()
let passwordTextField = try
    self.view.inspect().findAll(CustomTextField.self)[1].actualView()
usernameTextField.text = "test@test.com"
passwordTextField.text = "test"
```

https://github.com/nalexn/ViewInspector

¡Muchas gracias!

@CongresoConfIT

Congreso Conf.IT

@CongresoConfIT

CongresoConfIT

confit.es