

# Développement iOS

Première journée.

Ezequiel GOMES 23/04/2025

# Tables des matières

Journée du 23/04/2025

- Historique du développement iOS
- Présentation de l'écosystème
- Présentation de Xcode
- Présentation Swift
- Présentation Swift UI
- Premier projet





iPhone (2007)



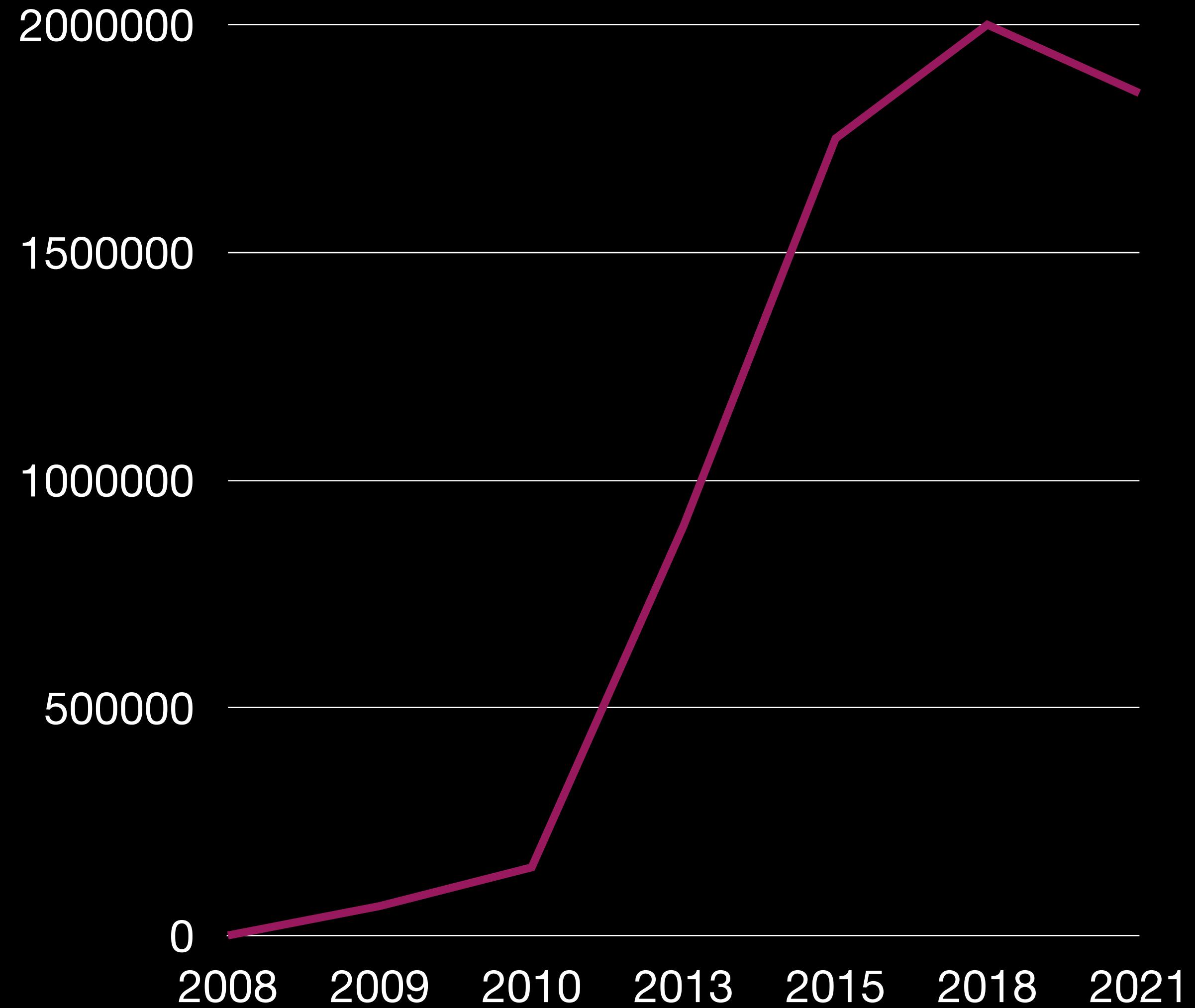
iPhone (2007)

- Premier téléphone MultiTouch
- A démocratisé les smartphone tout-écran
- iPhone OS 1.0
- Pas de 3G
- Pas de possibilité d'installer des applications tierce

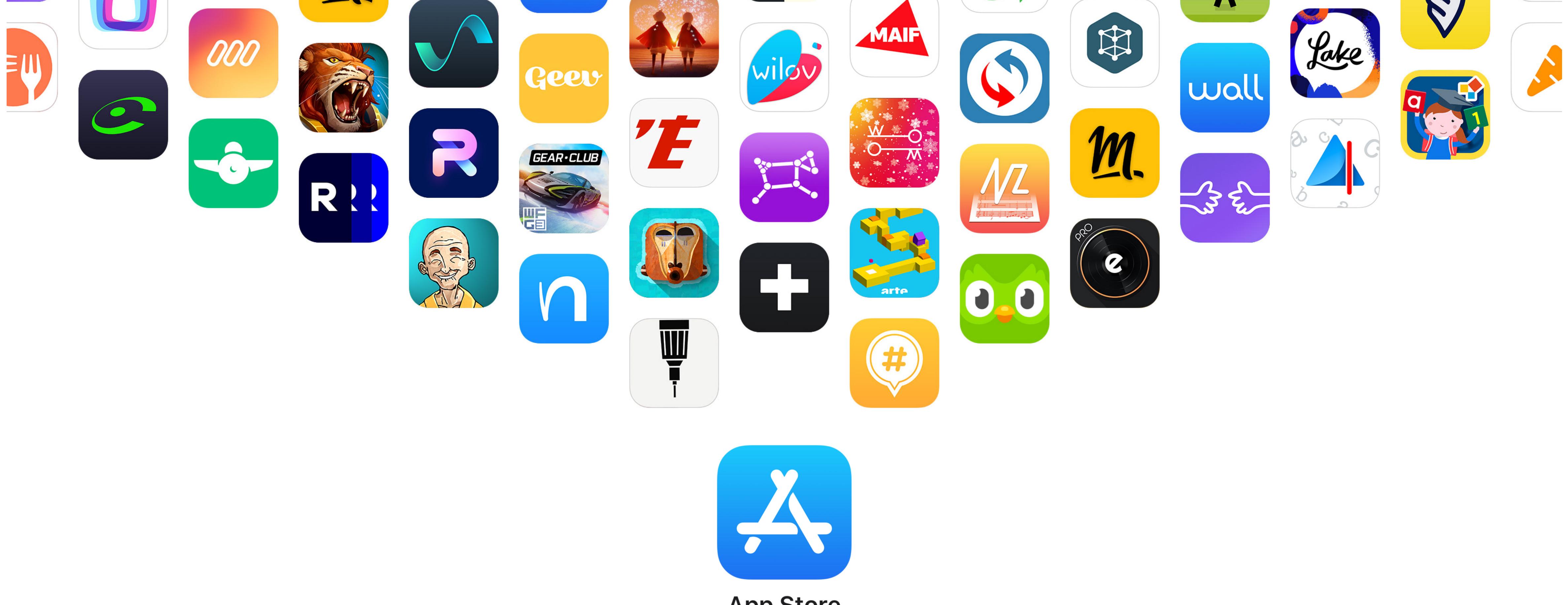


- Compatible 3G
- Possibilité d'installer des applications tierce

iPhone 3G (2008)



Evolution du nombre d'application



# 1.8 millions d'application sur L'app Store

# App Store alternatif

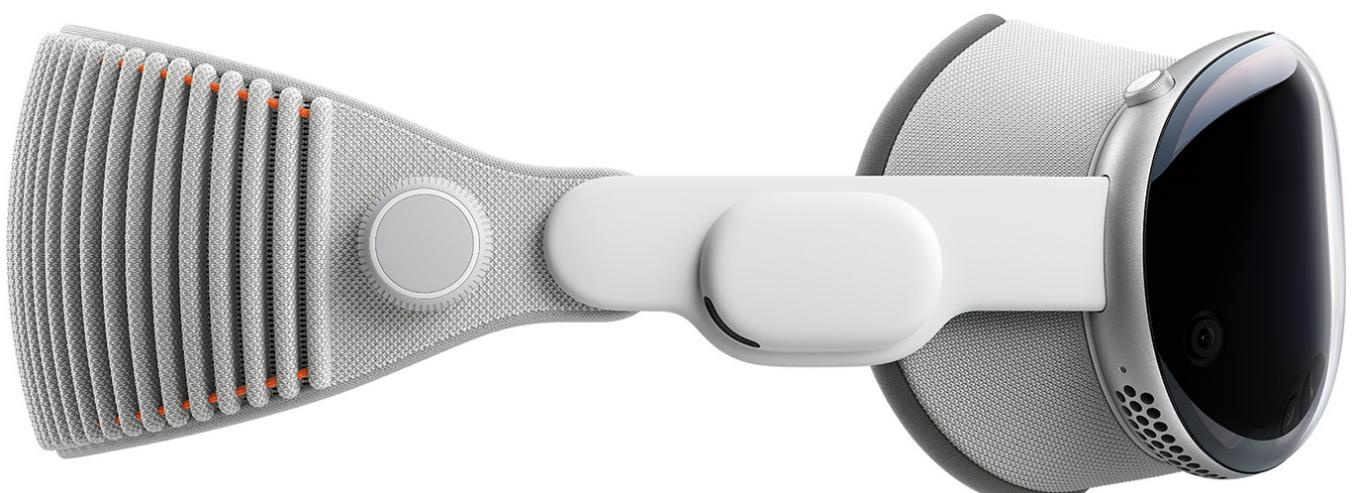
- Mise en place du DMA mars 2024
- Uniquement au sein de l'EU
- Possibilité d'installer des App Stores alternatifs



2LOKE

# L'écosystème Apple

## L'app store n'est pas unique à l'iPhone



# Swift

- Créer en 2014 remplace progressivement l'Objective-C
- Typé
- Orienté objet
- Compilé



# Swift UI

- Succède à UIKIT en 2019
- Framework déclaratif
- **Réactif** : l'UI se met à jour automatiquement quand les données changent
- Personnalisation avec des modifiers
- Composé de vues empilables
- **Multi-plateforme** : iOS, macOS, watchOS, tvOS



# UIKit VS Swift UI

## Impératif vs Déclaratif

```
let label = UILabel()  
label.text = "Bonjour"  
label.font = UIFont.systemFont(ofSize: 20)  
label.textColor = UIColor.blue  
view.addSubview(label)
```

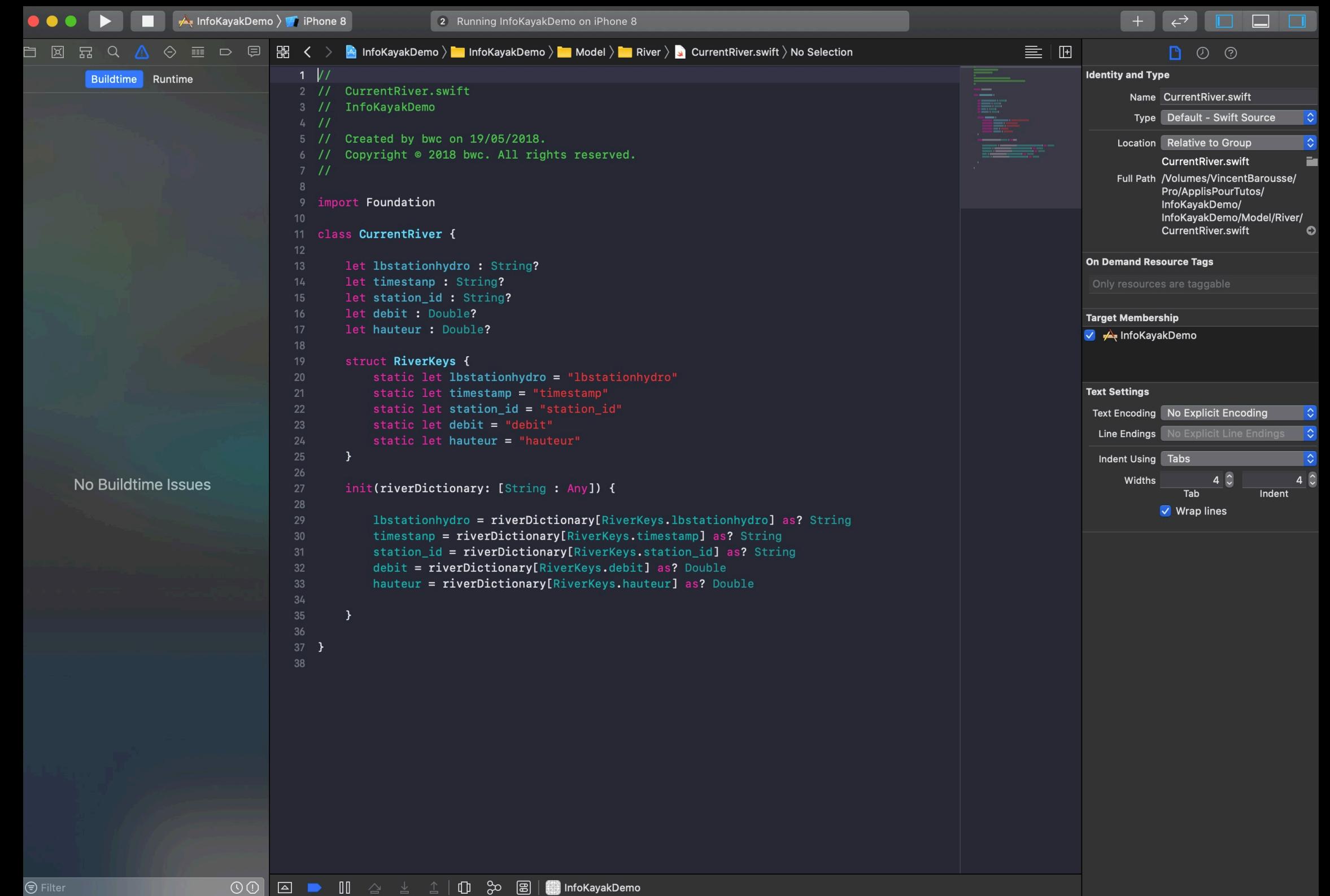
UIKit

```
Text("Bonjour")  
    .font(.title)  
    .foregroundColor(.blue)
```

Swift UI

# Xcode

- IDE officiel d'Apple
- Contient le compilateur Swift et Objective-C
- Permet de tester sur simulateur ou appareil réel
- Intègre un éditeur d'interface graphique (Storyboard pour UIKit, Canvas pour SwiftUI)



The screenshot shows the Xcode interface with the CurrentRiver.swift file open in the main editor. The code defines a class CurrentRiver and a struct RiverKeys, both used for parsing dictionary data. The Xcode interface includes the toolbar, navigation bar, file browser, and various settings panes on the right.

```
// CurrentRiver.swift
// InfoKayakDemo
//
// Created by bwc on 19/05/2018.
// Copyright © 2018 bwc. All rights reserved.

import Foundation

class CurrentRiver {

    let lbstationhydro : String?
    let timestamp : String?
    let station_id : String?
    let debit : Double?
    let hauteur : Double?

    struct RiverKeys {
        static let lbstationhydro = "lbstationhydro"
        static let timestamp = "timestamp"
        static let station_id = "station_id"
        static let debit = "debit"
        static let hauteur = "hauteur"
    }

    init(riverDictionary: [String : Any]) {
        lbstationhydro = riverDictionary[RiverKeys.lbstationhydro] as? String
        timestamp = riverDictionary[RiverKeys.timestamp] as? String
        station_id = riverDictionary[RiverKeys.station_id] as? String
        debit = riverDictionary[RiverKeys.debit] as? Double
        hauteur = riverDictionary[RiverKeys.hauteur] as? Double
    }
}
```

# Swift : Syntaxe de base

# Constante

```
let age = 30 // déclare une constante "age" ayant la valeur 30
```

# Variable

```
var nom = "John" // déclare une variable "nom" ayant la valeur « John"  
nom = "Doe" // change la valeur de la variable "nom" à "Doe"
```

# Collection : Array

```
var fruits = ["pomme", "banane", "orange"] // déclare un tableau de fruits  
fruits.append("kiwi") // ajoute un élément au tableau  
print(fruits[1]) // affiche "banane"
```

# Boucle

```
for i in 0..<5 { // boucle for qui itère de 0 à 4
    print(i)
}
```

# String : Concatenation

```
var prenom = "John"  
var nom = "Doe"  
var nomComplet = prenom + " " + nom // concatène les chaînes de caractères  
print(nomComplet) // affiche "John Doe"
```

# String : Interpolation

```
let prenom = "John"  
let age = 30  
print("Bonjour, mon nom est \$(prenom) et j'ai \$(age) ans.")
```

# Condition

```
var x = 5
if x > 10 {
    print("x est plus grand que 10")
} else {
    print("x est plus petit ou égal à 10")
}
```

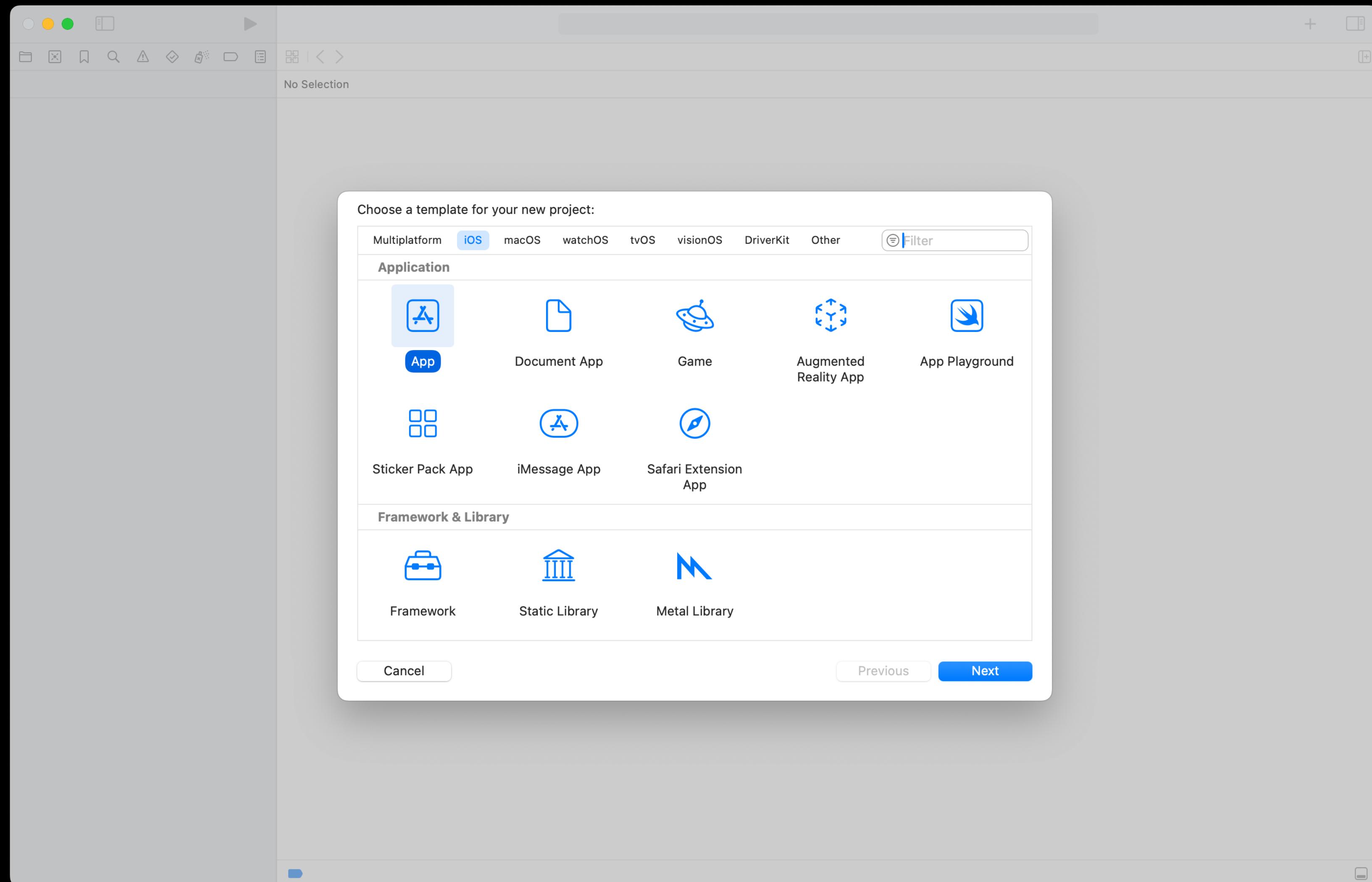
# Condition

```
let jour = "mercredi"
switch jour {
    case "lundi", "mardi", "mercredi":
        print("C'est un jour de la semaine")
    case "samedi", "dimanche":
        print("C'est un jour de week-end")
    default:
        print("Jour non reconnu")
}
```

# Swift UI

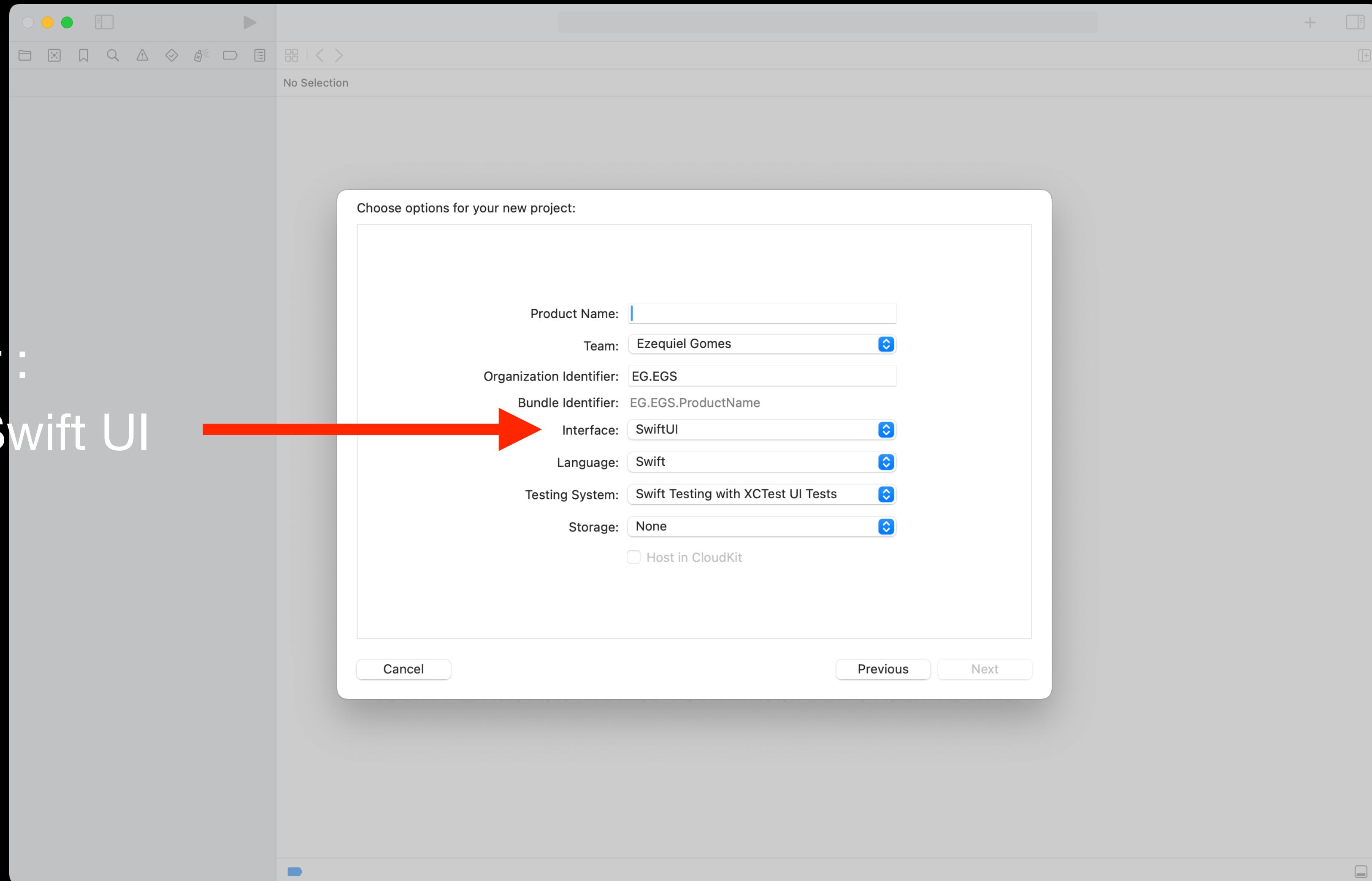
# Création d'un projet

File > New > Project ...



# Configuration du projet

Sélectionner :  
Interface : SwiftUI



Interface créée par défaut.

The screenshot shows the Xcode interface with the following details:

- Project Structure:** The left sidebar shows the project structure with files like ContentView.swift, Assets, and projetTestApp.
- ContentView.swift Code:** The main editor pane displays the code for ContentView. It includes imports, a struct definition for ContentView, and a preview block. The preview block shows a globe icon and the text "Hello, world!".
- Preview View:** To the right of the code editor is a preview view of an iPhone 16 Pro displaying the app's interface.
- Annotations:** A red arrow points from the text "Interface créée par défaut." to the project structure. A long red arrow points from the text "Canvas de Prévisualisation" to the preview view.

```
// ContentView.swift
// ContentView.swift
// projetTest
// Created by Ezequiel Gomes on 15/04/2025.

import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {
            Image(systemName: "globe")
                .imageScale(.large)
                .foregroundStyle(.tint)
            Text("Hello, world!")
        }
        .padding()
    }
}

#Preview {
    ContentView()
}
```

# Structure de base

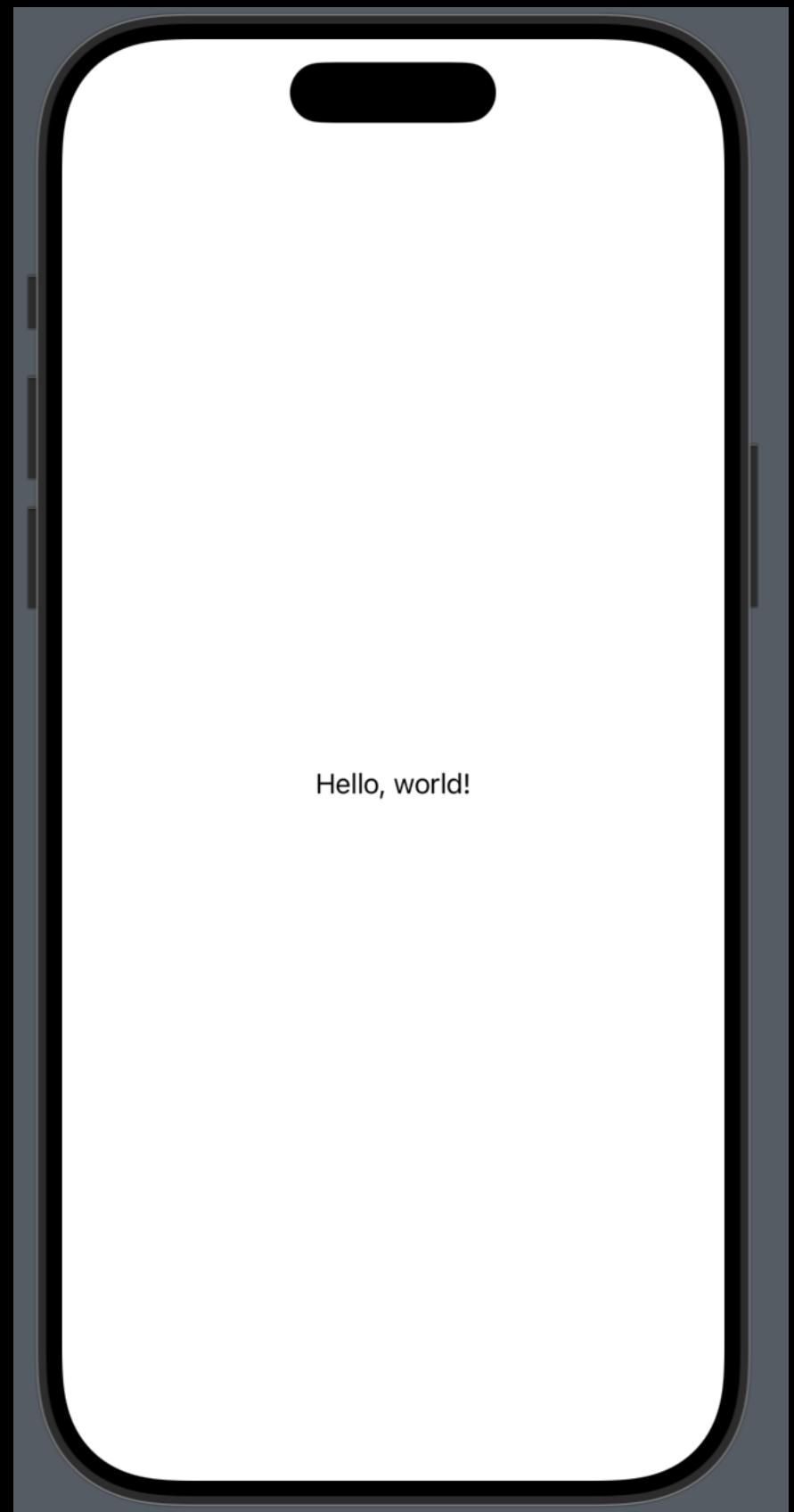
Les composants « graphique » se placent dans la variable body.

Le #Preview sert à afficher le Canva de prévisualisation

```
import SwiftUI

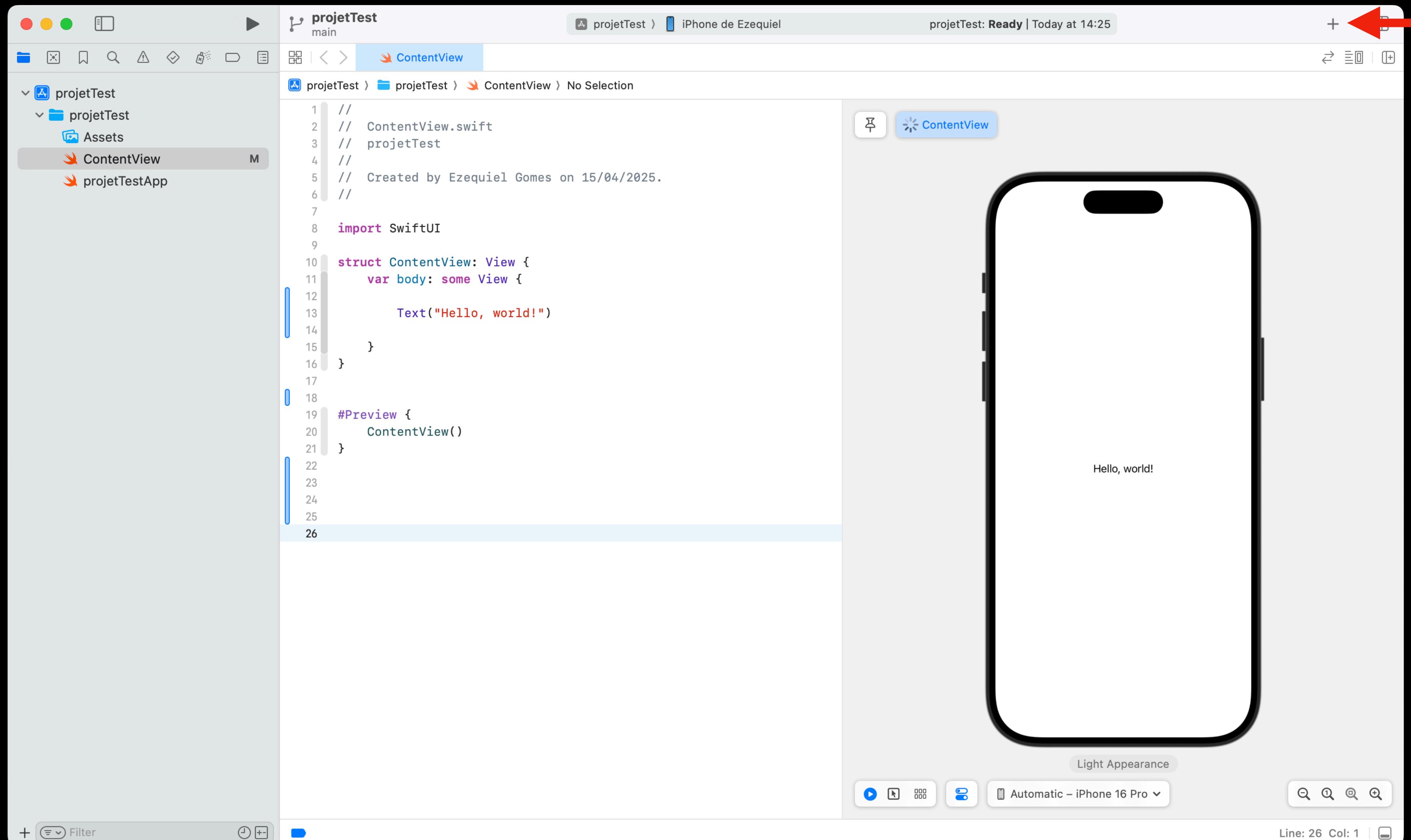
struct ContentView: View {
    var body: some View {
        Text("Hello, world!")
    }
}

#Preview {
    ContentView()
}
```

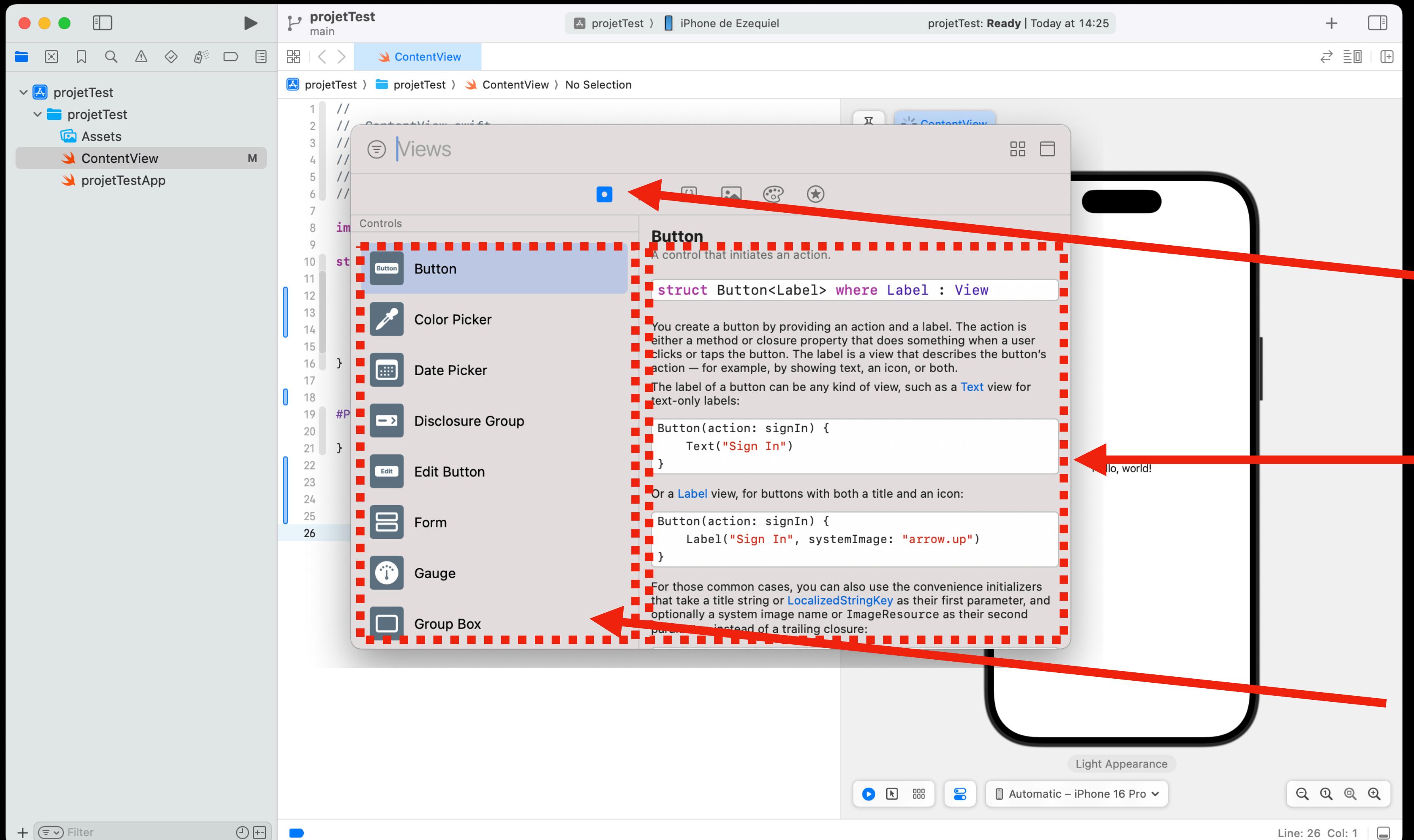


# Les différentes Views (composants)

Appuyer sur le  
« + »  
Pour afficher la  
fenêtre d'ajouts



# Les différents Views



Assurez-vous  
d'être bien dans  
cette section.

# Documentation sur l'utilisation de la view

# Liste des différentes View

# Les modifiers

Un **modifier** en SwiftUI est une **méthode qui permet de personnaliser un composant** (comme sa couleur, sa taille, sa position, etc.)

```
import SwiftUI

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

# Les modifiers

Un **modifier** en SwiftUI est une **méthode qui permet de personnaliser un composant** (comme sa couleur, sa taille, sa position, etc.)

```
import SwiftUI

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



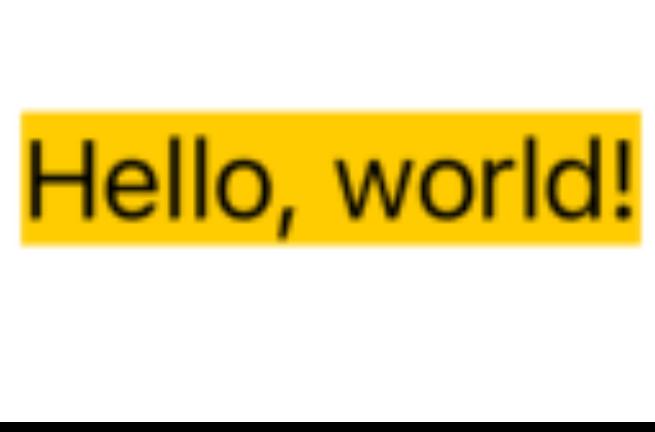
*Application  
du modifier  
fontWeight*

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

# L'ordre d'application des modifiers

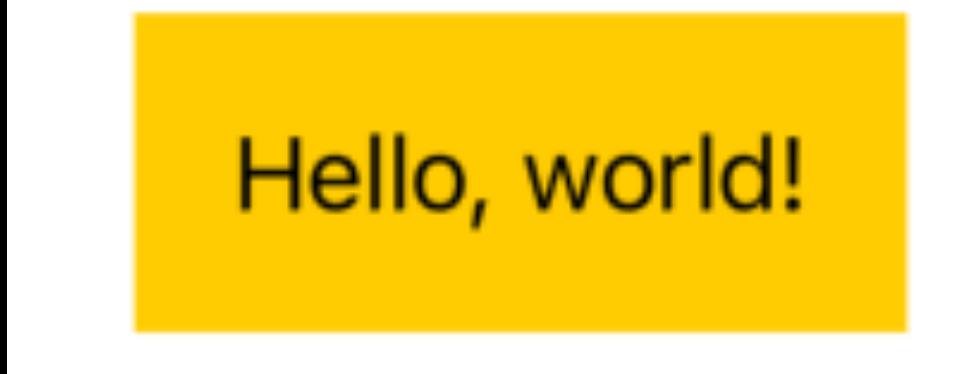
*Vous pouvez appliquer plusieurs modifier à un composant. Cependant, il est important de prêter attention à leur ordre d'application, car cela peut avoir un impact significatif sur le résultat visuel.*

```
Text("Hello, world!")  
    .background(Color.yellow)  
    |.padding()
```



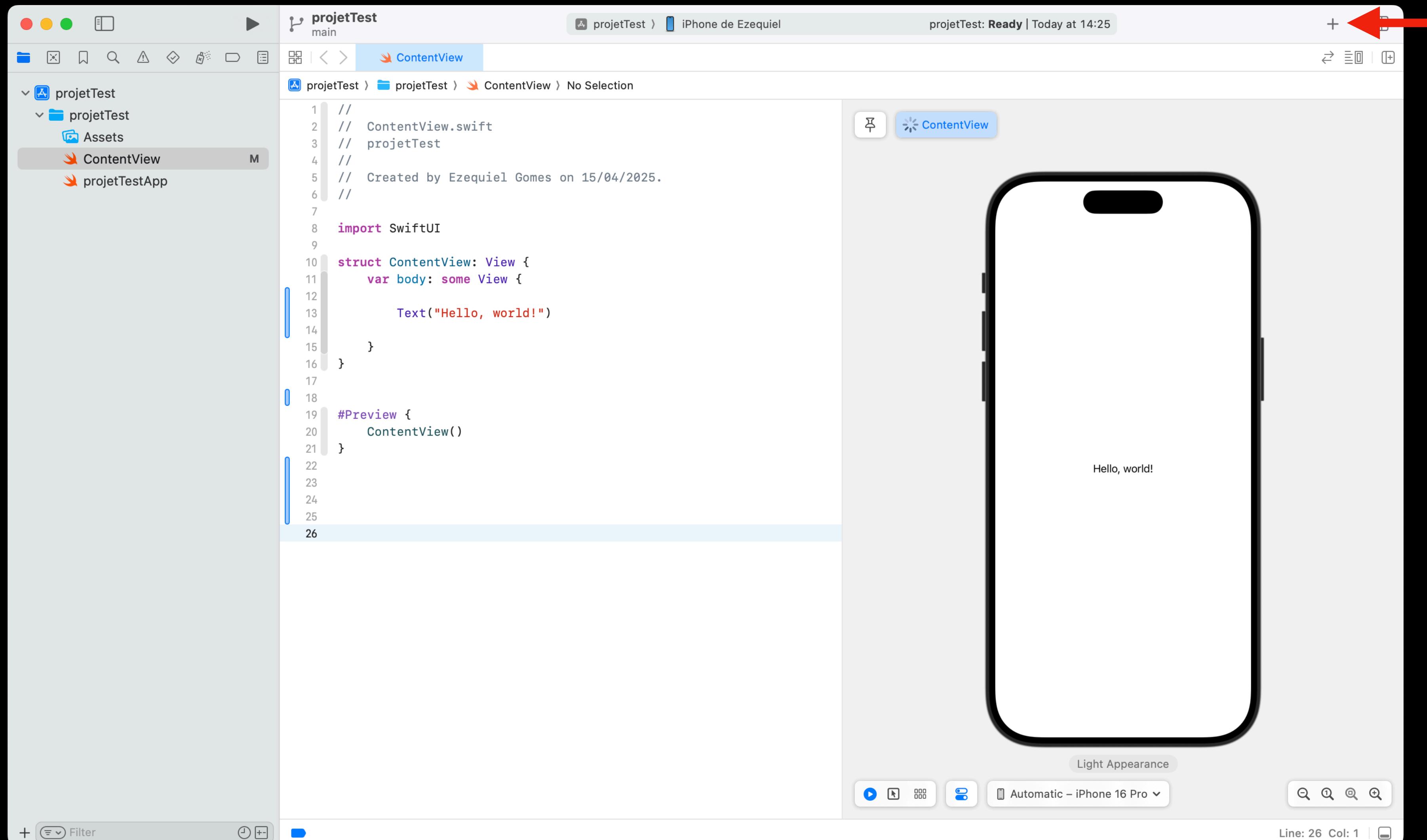
Hello, world!

```
Text("Hello, world!")  
    .padding()  
    .background(Color.yellow)
```



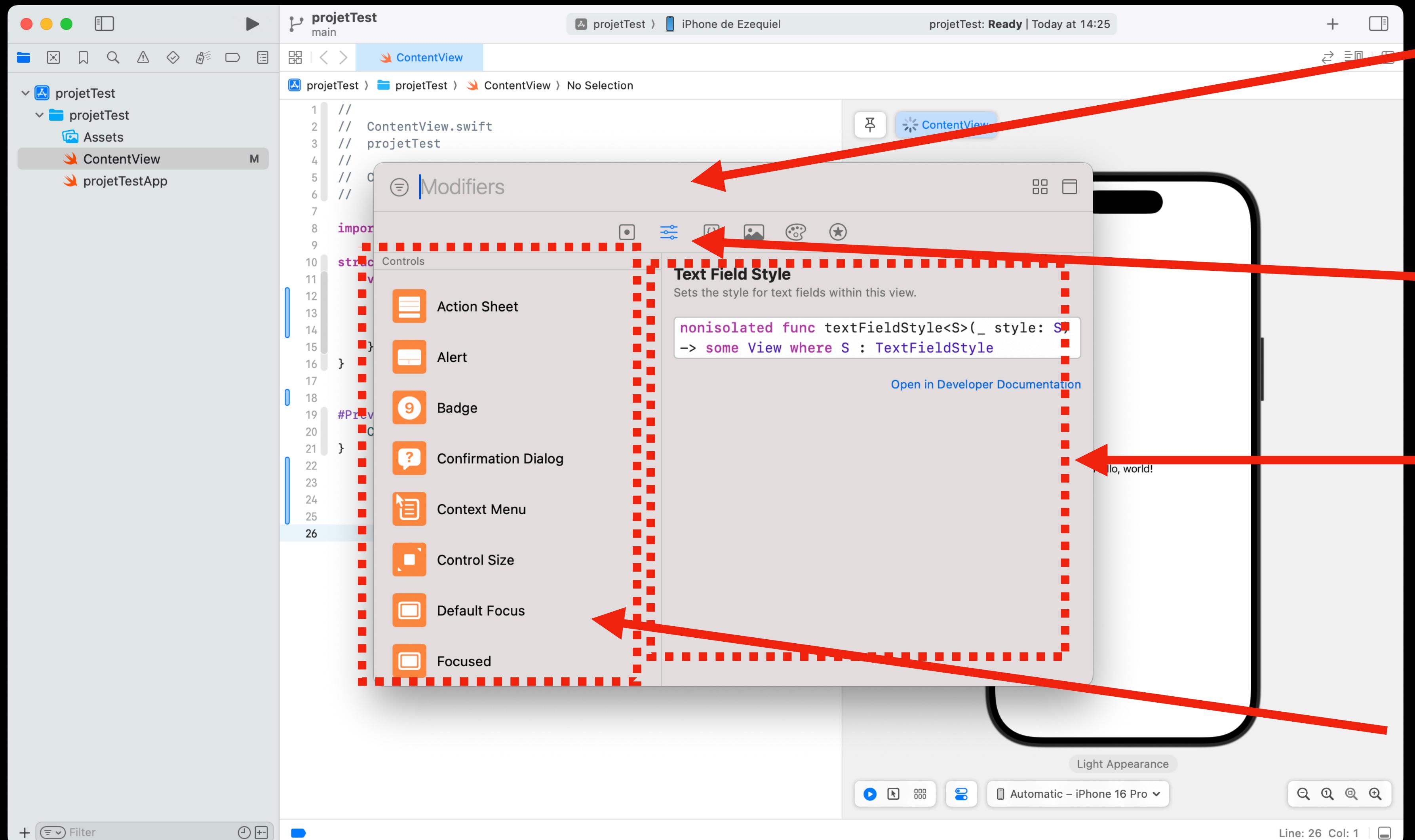
Hello, world!

# Les différents modifier



Appuyer sur le  
« + »  
Pour afficher la  
fenêtre d'ajouts

# Les différents composants



Vous pouvez rechercher un modifier en fonction d'une vue

Assurez-vous d'être bien dans cette section.

Documentation sur l'utilisation du modifier

Liste des différents modifiers

# Les Stacks

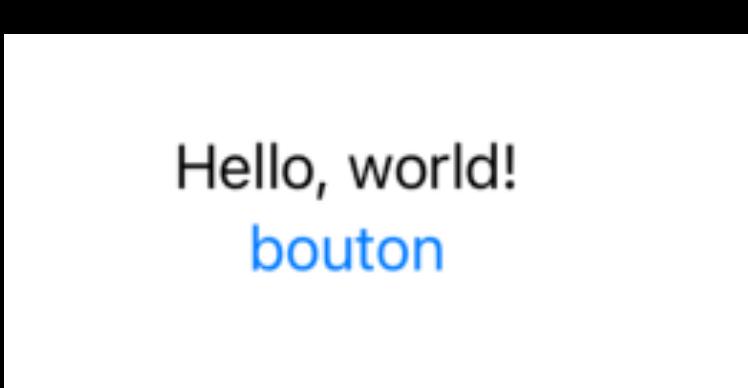
*Au-delà d'une seule vue, il est **fortement recommandé** d'encapsuler les vues dans une Stack*

*Il existe 3 principales Stacks en Swift UI :*

## VStack

Aligne les vues de  
**haut en bas**

```
 VStack {  
     Text("Hello, world!")  
     Button("bouton") {  
         print("hello")  
     }  
 }
```

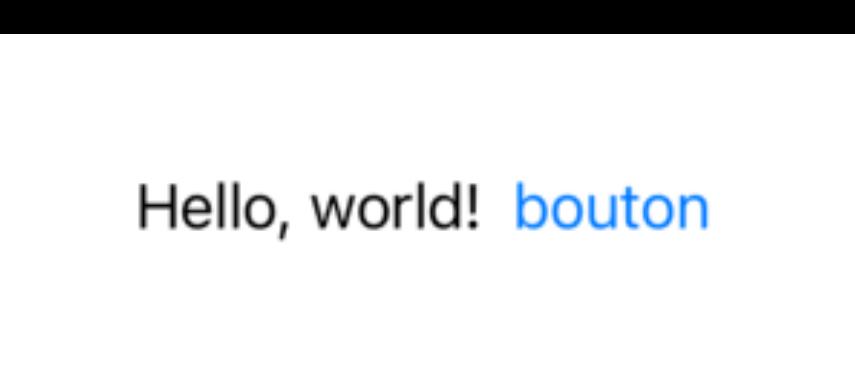


Hello, world!  
bouton

## HStack

Aligne les vues de  
**gauche à droite**

```
 HStack {  
     Text("Hello, world!")  
     Button("bouton") {  
         print("hello")  
     }  
 }
```



Hello, world! bouton

## ZStack

Superpose les vues les unes sur les autres

```
 ZStack {  
     Text("Hello, world!")  
     Button("bouton") {  
         print("hello")  
     }  
 }
```

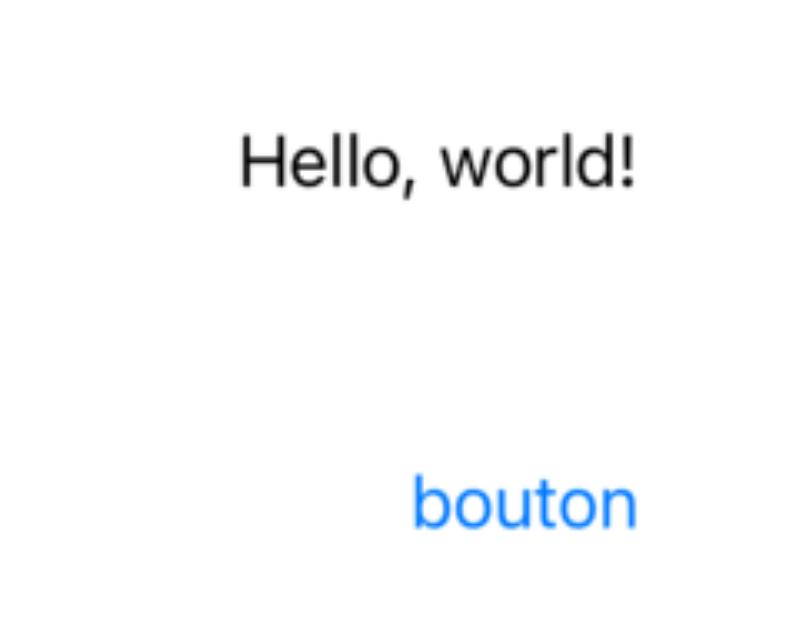


Hello, world!

# Les Stacks : Personnalisation

*Il est possible d'ajouter des paramètres pour modifier l'alignement ou l'espacement des éléments contenus dans la stack.*

```
 VStack(alignment: .trailing, spacing: 60) {  
     Text("Hello, world!")  
     Button("bouton") {  
         print("hello")  
     }  
 }  
 }
```



Hello, world!  
bouton