SwiftUI & Combine

José Luis Bustos Esteban

Tipo resultado

Tipo de resultado

Sin control de errores

```swift
enum ErrorSuma {
    case arrayVacio, datoInvalido
}

let array1 = [5,6,7,7,4,3,3,4]
let array2 = [-5,6,7,7,-4,3,3,4]

func SumaNumero(nums: [Int]) -> Int? {
    if nums.isEmpty {
        return nil
    }

    if !(nums.allSatisfy { $0 >= 0 }){
        return nil // si alguno es menor de cero
    }

    return nums.reduce(0,+)
}


SumaNumero(nums: array1)
SumaNumero(nums: array2)
SumaNumero(nums: [])
```

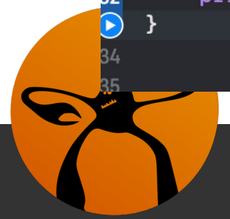
En caso de error nos devuelve nil pero no sabemos que es lo que ha pasado

Tipo de resultado

Control errores do/catch

```swift
import UIKit

enum ErrorSuma:Error {
    case arrayVacio, datoInvalido
}

let array1 = [5,6,7,7,4,3,3,4]
let array2 = [-5,6,7,7,-4,3,3,4]

func SumaNumero(nums: [Int]) throws -> Int {
    if nums.isEmpty {
        throw ErrorSuma.arrayVacio
    }

    if !(nums.allSatisfy { $0 >= 0 }){
        throw ErrorSuma.datoInvalido
    }

    return nums.reduce(0,+)
}

do {
    let suma1 = try SumaNumero(nums: array1)
    let suma2 = try SumaNumero(nums: array2)
    let suma3 = try SumaNumero(nums: [])

} catch ErrorSuma.arrayVacio {
    print("Array Vacio")
} catch ErrorSuma.datoInvalido {
    print("Array Data invalido")
} catch {
    print("\(error)")
}
```

- Creamos una enumeración de Error
- La función implementa “throws”
- Cuando hay errores se lanza el throw correspondiente
- Capturamos los errores en el do/catch en la llamada a la función.



Tipo de resultado

Enumeraciones de carga

```swift
enum  Resultado<Int,String> {
    case ok(Int)
    case error(String)
}

func SumaNumeros2(nums: [Int])  -> Resultado<Int,String> {
    if nums.isEmpty {
        return .error("Array vacio")
    }

    if !(nums.allSatisfy { $0 >= 0 }){
        return .error("Dato no valido")
    }

    return  .ok(nums.reduce(0,+))
}

let resultado = SumaNumeros2(nums: [1,2,3,45,6])

switch resultado {
    case .ok(let num) : print("Resultado ok: \(num)")
    case .error(let mensaje) : print("Error: \(mensaje)")
}
```

- Desde Swift 5 hay otra forma de gestionar los errores que es con el tipo de resultado y las enumeraciones de carga.
- Se puede crear tu propia enumeración de resultados de una operación.
- La función usa dicha enumeración en el tipo de devolución como dentro en los return
- Al llamar a la función podemos hacer un switch para los diferentes valores de la enumaracion (error y Success)

Tipo de resultado

Result type de Apple

```swift
import Foundation


enum ErrorSuma:Error {
    case arrayVacio, datoInvalido
}


func SumaNumeros(nums: [Int])  -> Result<Int, ErrorSuma> {
    if nums.isEmpty {
        return .failure(.arrayVacio)
    }

    if !(nums.allSatisfy { $0 >= 0 }){
        return .failure(.datoInvalido)
    }

    return  .success(nums.reduce(0,+))
}


let suma = SumaNumeros(nums: [1,2,3,4,5])

switch suma {
    case .success(let num) : print("Resultado ok: \(num)")
    case .failure(let mensaje) : print("Error: \(mensaje)")
}
```

- Las enumeraciones de carga es lo que ha utilizado Apple para crear el resultType.
- Se ha creado el tipo “Result” que se le indica el tipo del Success y el tipo de Error
- Es una promesa de los futuros casos que la petición asíncrona podrá tener (success y error)
- El sistema garantiza uno de ambos resultados y por ello hay que gestionar ambos