

# Universidad Nacional de La Matanza



# Swift y iOS nativo

# Agenda

- Lenguaje
- Herramientas
- Build y Release
- Analogías

*“iOS para programadores Android”*

# Lenguaje

# Dos opciones

- Objective-C
  - Primer lenguaje con soporte oficial.
  - Orientado a objetos, pero basado en C.
  - Bajo nivel de abstracción, muy *verboso*.
- Swift
  - Oficial desde 2014.
  - Más alto nivel.
  - Similar a Kotlin.

## Objective-C

```
const int count = 10;
double price = 23.55;

NSString *firstMessage = @"Swift is awesome. ";
NSString *secondMessage = @"What do you think?";
NSString *message = [NSString stringWithFormat:@"%s%s", firstMessage, secondMessage];

NSLog(@"%@", message);
```

## Swift

```
let count = 10
var price = 23.55

let firstMessage = "Swift is awesome. "
let secondMessage = "What do you think?"
var message = firstMessage + secondMessage

print(message)
```

Swift

# Variables

- Inmutables vs mutables (**let** vs **var**)
- Inferencia de tipos.

```
var myVariable = 42  
myVariable = 50  
let myConstant = 42
```

```
let implicitInteger = 70  
let implicitDouble = 70.0  
let explicitDouble: Double = 70
```



# Strings

```
let apples = 3
```

```
let oranges = 5
```

```
let appleSummary = "I have \$(apples) apples."
```

```
let fruitSummary = "I have \$(apples + oranges) pieces of fruit."
```

# Opcionales

- Los tipos opcionales pueden tener un valor concreto, o **nil** en su defecto.

```
var optionalString: String? = "Hello"  
print(optionalString == nil)  
// Prints "false"
```

```
var optionalName: String? = "John Appleseed"  
optionalName = nil
```

```
var greeting = "Hello!"
```

# Operador ??

- Equivalente al *elvis operator* (?:) de Kotlin

```
let nickname: String? = nil
let fullName: String = "John Appleseed"
let informalGreeting = "Hi \ \(nickname ?? fullName)"
```

```
// Prints "Hi John Appleseed"
```

# Condicionales

- if
- Switch
- if let
- guard let

# if

- Paréntesis opcionales en los **if**.

```
var teamScore = 100
```

```
if score > 50 {  
    teamScore += 3  
} else {  
    teamScore += 1  
}
```

```
print(teamScore)           // Prints "103"
```

# switch

```
let vegetable = "red pepper"

switch vegetable {
  case "celery":
    print("Add some raisins and make ants on a log.")
  case "cucumber", "watercress":
    print("That would make a good tea sandwich.")
  case let x where x.hasSuffix("pepper"):
    print("Is it a spicy \(x)?")
  default:
    print("Everything tastes good in soup.")
}

// Prints "Is it a spicy red pepper?"
```

# If let y guard let

- Son dos estructuras de control que nos permiten ejecutar un bloque de código dependiendo de si se pudo o no concretar una asignación de una variable (en base a nulidad)

# If let

- Puede contener tanto un bloque para cuando la condición es verdadera (asignación exitosa, no nil) como uno para cuando la condición es falsa (asignación es nil)
- La variable declarada sólo es accesible **dentro** del if let



# If let

```
var greeting: String = ""  
if let name = optionalName {  
    greeting = "Hello, \ (name)"  
} else {  
    greeting = "Hello unknown!"  
}
```

`print(name)` // No es posible!! La variable 'name' solo es accesible dentro del `if-let`

# guard let

- En caso que la condición no se cumpla, debe retornar y salir del scope de la función que lo llama (early exit)
- La variable declarada puede ser accedida **fuera** del guard let

# guard let

```
guard let name = optionalName else {  
    errorMessage = "Name cannot be null"  
    return // early exit, obligatorio  
}
```

```
println("Name is \((name)") // OK, guard let permite que se acceda a name
```

# Estructuras de repetición

- for
- for-in
- while

# for

```
var total = 0
for i in 0..<4 {
    total += i
}
```

# for-in

```
let individualScores = [75, 43, 103, 87, 12]
var teamScore = 0
for score in individualScores {
    if score > 50 {
        teamScore += 3
    } else {
        teamScore += 1
    }
}
```

# while

```
var n = 2
while n < 100 {
    n *= 2
}
print(n)
// Prints "128"
```

```
var m = 2
repeat {
    m *= 2
} while m < 100
print(m)
// Prints "128"
```

# Funciones

```
func greet(person: String, day: String) -> String {  
    return "Hello \$(person), today is \$(day)."  
}
```

```
greet(person: "Bob", day: "Tuesday")
```

- Siempre se envían los valores de los parámetros anteponiendo su nombre, salvo que la declaración de la función indique lo contrario (con un "\_" delante del nombre)
- Los parámetros de una función se deben proveer en el mismo orden que su declaración



# Clases y objetos

```
class Shape {  
    var numberOfSides = 0  
  
    func simpleDescription() -> String {  
        return "A shape with \$(numberOfSides) sides."  
    }  
}
```

...

```
var shape = Shape()  
shape.numberOfSides = 7  
var shapeDescription = shape.simpleDescription()
```

# Constructores / *self*

```
class Shape {  
    init(numberOfSides: String) {  
        self.numberOfSides = numberOfSides  
    }  
}
```

# Herramientas

# Herramientas

- Macbook
- XCode
- Swift
- Simulator
- CocoaPods



Build y Release

# Build y Release

- Todo integrado en XCode.
- Cuentas de Apple Developer.
- Obligatorio actualizar XCode y Swift cada año.
- CocoaPods no es una herramienta oficial 😞
- No se pueden distribuir archivos .ipa fácilmente.
  - Firebase App Distribution
  - HockeyApp
  - TestFlight (oficial)

# Analogías

# Analogías

		
Aplicación	Application	AppDelegate
Vistas	View/ViewGroup + layouts XML	UIKit
Pantallas	Activity, Fragment	ViewController
Ciclo de vida	onCreate(), onStart(), onStop()...	viewDidLoad(), viewWillAppear()...
Navegación	startActivity() / Jetpack Navigation component	Storyboards
Pantalla y Vistas	Jetpack Compose	Swift UI
IDE	Android Studio	XCode
Dependencias	Gradle (build.gradle en Groovy o Kotlin)	Cocoapods / Swift Package Manager
Build	Gradle	XCode

\* Evento: Apple Worldwide Developers Conference (WWDC)

<https://developer.apple.com/wwdc22/>



# Links

- [Documentación oficial de Swift](#)
- [Curso abierto de la Universidad de Stanford](#)
- [Curso de Swift 5 en español](#)
- Swift Playground en App Store

¿Preguntas?

Fin