Grupo1

March 12, 2020

1 Lenguajes Nativos para Desarrollo de Apps Móviles

Melisa Castro

Carlos Cuasqui

Lenin G. Falconí

1.1 Objetivo General

- Estudiar algunos de los lenguajes nativos de programación para desarrollo de Apps móbiles, mediante la investigación de sus características, fortalezas, debilidades y comparativa entre estos para lograr elegir el lenguaje nativo necesario según los requerimientos

1.2 Objetivos Específicos

- Estudiar Android, sus características, fortalezas y debilidades.
- Estudiar Objective C, sus características, fortalezas y debilidades.
- Estudiar Swift, sus características, fortalezas y debilidades.

1.2.1 Introducción

- El advenimiento de los **smartphones** ha generado una alta demanda de aplicaciones móviles.
- Las compañías de desarrollo se han visto en la necesidad de contratar desarrolladores para cubrir la demanda de apps intuitivas, innovadoras y útiles.
- Escoger el lenguaje de programación que mejor se adapte a los requerimientos es un factor importante para el desarrollo de aplicaciones móviles.

1.2.2 Introducción

El número de aplicaciones estimadas por plataforma es aproximadamente: de aplicaciones móviles:

- **Google Play**, 9 millones
- **App Store**, 2.5 millones
- **Windows** 700 mil

1.2.3 Conceptos Importantes

- La elección del mejor lenguaje de programación para desarrollar una app depende en gran medida del sistema operativo en el que correrá.
- Las apps móviles nativas son apps que están escritas en un lenguaje nativo que es soportado por el proveedor del sistema operativo del dispositivo.
- Las API de plataforma subyacentes están disponibles al 100% para el código de la aplicación y el sistema operativo proporciona la biblioteca de componentes de la interfaz de usuario.
- El proceso de compilación convierte este código en una aplicación ejecutable con código de bytes nativo del sistema operativo.

1.2.4 Conceptos Importantes

iOS:

- Plataforma patentada por Apple.
- Disponible para dispositivos telefónicos (iPhone) y tabletas (iPad)
- Debe disponer de una cuenta de desarrollador de Apple y Xcode IDE en una computadora Mac
- Las aplicaciones se pueden construir con iOS SDK nativo con Objective-C y Swift

1.2.5 Conceptos Importantes

Android:

- Android es una plataforma de código abierto desarrollada y promovida principalmente por Google
- Google promueve su propia marca de dispositivos móviles, Pixel (y la marca anterior, Nexus)
- Otras maracas que dependen de Android para sus Apps: Samsung, Huawei, Xiaomi y Oppo
- Para un IDE, Android Studio es, con mucho, el más popular, pero hay otros IDE igualmente populares como Netbeans e IntelliJ Idea

1.3 Android

- Es un sistema operativo móvil desarrollado por Google, basado en Kernel de Linux y otros software de código abierto.
- Fue diseñado para dispositivos móviles con pantalla táctil, como teléfonos inteligentes y tabletas.

1.3.1 Características

- Su núcleo está basado en Linux
- Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android.
- Reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android.
- La primera versión estable fue publicada en diciembre de 2014.

1.3.2 Características

- Originalmente desarrollado por una empresa llamada Android, en el año 2006 fue adquirida por Google
- Android ofrece otras particularidades muy interesantes, tales como las tiendas de aplicaciones.
- La tienda oficial de Android es Google Play donde encontraremos un sin número de aplicaciones móviles la mayoría de ellas completamente gratuitas.

1.3.3 Fortalezas

- Ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y Linux.
- Android Studio es un software de código abierto.
- Android Studio permite al desarrollador que pruebe su aplicación de forma virtual, sin la necesidad de un dispositivo móvil real ya que cuenta con un emulador.
- Gracias a la Integración de marcos de desarrollo y a la caja de herramientas Android NDK (Native Development Kit), admite la utilización de lenguajes de programación como Java.

1.3.4 Debilidades

- Si bien Java es uno de los lenguajes de programación más utilizados en el mundo, y es el lenguaje oficial de Android, existen varios motivos por los cuales Java no siempre sea la mejor opción para tus proyectos Android.
- El mayor inconveniente es que Java no es un lenguaje moderno
- Java además tiene algunos problemas bien documentados, incluyendo bloques try-catch interminables, falta de extensibilidad, inseguridad con respecto a los valores nulos
- falta de soporte para programación funcional.

1.3.5 Debilidades

- La sintaxis de Java es muy verborrágica(abundancia de texto o código)
- Para que el emulador trabaje correctamente, requiere de una elevada cantidad de recursos.

1.4 Objective C

- Lenguaje de programación nativo para los sistemas operativos OS X y iOS de Apple.
- Creado por Brad Cox y la corporación StepStone en 1980. Características.

1.4.1 Características

- Orientado a objetos
- Basado en C y Smalltalk
- Todo código C es compilable en Objective-C
- Se pueden usar las librerías de C dentro de Objective-C

- No existe un tipo booleano, se usa:

  ```
  typedef signed char BOOL;
  ```

- Existen punteros, imprescindibles para tratar con objeto (alloc devuelve un puntero).

1.4.2 Características

- Presenta herencia simple, cada clase hereda de una y solo una super clase.
- Durante la ejecución los objetos son creados siempre en memoria dinámica.
- Se entregan los mensajes usando objc_msgSend()
- Las variables instancia son normalmente (y por defecto) privadas.

1.4.3 Características

- Las clases se define en dos archivos (.h y .m)
- .h contiene la interfaz de la clase.
- No existe operador **new**.
- Para instalar un objeto se debe:
  - Llamar al método alloc
  - Llamar al método init
- No hay constructores, se usa init () para código de inicialización custom.
- Las propiedades se pueden implementar en .m

```
@synthesize name = _name;
```

- **@synthesize** genera los métodos set y get.

1.4.4 Características

- Las clases pueden tener propiedades:

```
@interface User : NSObject
@property (class, nonatomic, assign, readonly) NSInteger userCount;
@property (class, nonatomic, copy) NSUUID *identifier;
```

1.4.5 Características

- Para declarar un puntero se usa:

  ```
  type *varname;
  ```

- El valor NULL significa que el puntero apunta a nada.

```
#import <Foundation/Foundation.h>

int main () {
   int  *ptr = NULL;
   NSLog(@"The value of ptr is : %x\n", ptr  );
```

```
    return 0;
}
```

1.4.6 Características

- La gestión de memoria se da por tres métodos:
  - Manual: el número de veces que se "reclama" un objeto debe ser igual al numero de veces que se libera.
  - Semi-automatico (ARC): No llamar nunca a retain, reléase o autorelease. Se puede actuar como un GC.
  - Automático (GC): no puede ser usado en iOS
- Durante la ejecución los objetos son creados siempre en memoria dinámica.

1.4.7 Fortalezas

- Funciona de manera óptima con C++ y Objetive-C++.
- Posee características dinámicas en su funcionamiento, como el método swizzling.
- Posee soporte mejorado al momento de insertar marcos binarios.
- Un programa escrito en Objective C es más dinámica, capaz de reunir información sobre sí mismo para tomar decisiones con respecto a los tipos de memoria o de datos.

1.4.8 Debilidades

- Al estar construido en C, posee poco espacio de nombre.
- Produce errores difíciles de encontrar y corregir.
- Contiene lenguaje detallado pero complejo.
- En Objective C es que se basa en gran medida en el marco de cacao por su funcionalidad. Si bien esto es un beneficio para los desarrolladores de Apple, los desarrolladores de Windows no encontrarán el mismo éxito.

1.5 Swift

1.5.1 Introducción

- Introducido por Apple en 2014 para la plataforma **iOS**.
- Disponible para desarrollo de Apps desde **Xcode 2015**.
- Es el lenguaje por defecto de Apple para desarrollo hoy en día.

1.5.2 Características

- Es similar a lenguajes de programación tipo **script**
- Requiere computador Mac (macOS 10.11.5 o posterior) con última version Xcode
- Requiere de una cuenta **Xcode IDE** en un computador **Mac**
- Sintaxis ligera y moderna para expresar ideas complejas de manera clara concisa.

- Combina la seguridad y la velocidad en el desarrollo.
- Maneja automáticamente la memoria.
- No requiere de **;** para finalizar la instrucción.

1.5.3 Características

- Las variables siempre son inicializadas antes de su uso.
- Provee propias versiones de todos los tipos de datos fundamentales de **C** y **Objective-C**.
- Dispone de versiones mejoradas para los tipos de dato: diccionario, arreglo y conjunto.
- Los índices de los arreglos son revisados para evitar errores de desborde.
- WRT. Objective-c dispone de Tuplas, retornando múltiples valores de una función como un elemento sencillo.

1.5.4 Características

- La declaración de una clase con atributo y métodos en swift es:

```
class Square {
    var length: Int = 1

    func area() -> Int {
    return length * length
  }
    init(length: Int) {
        self.length = length
    }
}
```

- La instanciación de un objeto es:

```
var firstSquare = Square(length: 3)
println(firstSquare.length)
```

1.5.5 Características

- La herencia se implementa con la palabra reservada **Subclass**

```
class UnaSubclase: UnaSuperClase {
    // definición de la subclase
}
```

```
class Vehiculo {
    var velocidadActual = 0.0
    func sonarBocina() {
    }
}
class Bicicleta: Vehiculo {
    var tieneCesta = false
```

```
}
let bicicleta = Bicicleta()
bicicleta.tieneCesta = true
```

1.5.6 Fortalezas del Lenguaje

- Amisto y fácil de utilizar y con soporte de la ingeniería de Apple
- Compilador optimizado para el rendimiento del dispositivo
- Lenguaje optimizado para el desarrollo de la aplicación
- Es **Open Source**
- Es **type-safe** y tiene **type-inference**

```
let var1 = 42
let pi = 3.1459
```

1.5.7 Debilidades del Lenguaje

- Desarrollo exclusivo en un computador Mac.
- No puede hacer el desarrollo y *debug* en Windows.
- Swift siempre escoge **Double** antes que **Float**
- Lenguaje muy joven
- Poca interoperabilidad con herramientas de terceros e IDEs

1.5.8 Conclusiones

- Unas de las caracterízate notables del lenguaje objetive C es la forma en la que acopla la programación orientada a objetos y la compatibilidad con el lenguaje c, logrando una compatibilidad con todos los programas y librerías de C.
- Swift ofrece código más conciso, claro y limpio que facilita la escritura y la comprensión, lo que facilita el aprendizaje del lenguaje especialmente a programadores que vienen de otras plataformas.
- En las plataformas **iOS** una desventaja fuerte es lo cerrado del desarrollo que está anclado a utilizar equipos y software especifico.
- El hermeticismo de Apple puede generar que algunos programadores no conozcan o no se familiaricen con el desarrollo de sus herramientas al carecer del hardware.
- Android, como ventaja, puede ser programado en cualquier tipo de computador a diferencia de **iOS**.

1.6 Referencias e información adicional:

1. Introducion a objetive C, ferestrepoca
2. Objective C, Luis Montesano & Ana C. Murillo
3. https://domingogallardo.github.io/apuntes-lpp/teoria/tema06-programacion-orientada-objetos-swift/tema06-programacion-orientada-objetos-swift.html#herencia
4. https://medium.com/@TarunNagarDubai/8-best-programming-languages-for-mobile-app-development-59c2247c63b0

5. https://www.weheartswift.com/object-oriented-programming-swift/
6. https://www.alianzared.com/estadisticas-marketing-movil-2018/