PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso Ciência da Computação - Disciplina de Linguagens de Programação

Bruno Luiz, Gabriel Gomes, Luiz Guimarães, Luiza Ávila, Pedro Achilles

ESTUDO DA LINGUAGEM DE PROGRAMAÇÃO SWIFT:
Seminário de Linguagens de Programação

Belo Horizonte
2019

Bruno Luiz, Gabriel Gomes, Luiz Guimarães, Luiza Ávila, Pedro Achilles

ESTUDO DA LINGUAGEM DE PROGRAMAÇÃO SWIFT:
Seminário de Linguagens de Programação

Trabalho apresentado à disciplina de Linguagens de Programação, do curso de Ciência da Computação da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção de nota em Linguagens de Programação
Professor: Marco Rodrigo Costa
Coordenador: Max do Val Machado
Área de concentração: Linguagem de Programação

Belo Horizonte
2019

SUMÁRIO

1. Introdução

A linguagem de programação Swift foi anunciada em 2014 pela empresa de tecnologia Apple, com o objetivo de agilizar o processo de desenvolvimento de aplicações para os diversos dispositivos que a empresa cria. É compatível para macOS, iOS, tvOS, watchOS e Linux, Sistema Operacional caracterizado como software livre. Até o fim de 2015, os direitos sobre o Swift eram de propriedade da empresa em questão, mas a partir desta data, a linguagem passou a ser considerada “open source”.

Swift possui compatibilidade com a API Cocoa, uma interface para programação de aplicações da própria Apple para seus sistemas, desenvolvida em Objective-C, que aplica o sistema de troca de mensagens à linguagem C. O compilador do Swift utiliza a infraestrutura LLVM (Low Level Virtual Machine), utilizada em diversas linguagens, como o próprio Objective-C e Python. O compilador também é distribuído no ambiente de desenvolvimento XCode, utilizado para desenvolvimento de aplicações para dispositivos Apple.

2. Histórico

Em julho de 2010 Chris Lattner começou o desenvolvimento da linguagem Swift junto a outros desenvolvedores Apple. Em 2 de junho de 2014, na Worldwide Developers Conference (WWDC) um app realizado na linguagem foi apresentado ao público, assim como um protótipo da linguagem para os desenvolvedores apple registrados e um manual de 500 páginas gratuitamente.

A versão 1.0 foi liberada para o público em 9 setembro de 2014. Swift 1.1 foi introduzida em outubro de 2014, a versão 1.2, em abril de 2015 e na conferência WWDC de 2015 Swift 2.0 foi liberada ao público. Em setembro de 2015, foi permitido a venda de aplicativos realizados na linguagem na App Store e em dezembro do mesmo ano, o roteiro de Swift 3.0 foi anunciado em um blog da Apple. Entretanto, antes disso, a versão intermitente, 2.2, foi introduzida, com uma sintaxe e características, eliminando também componentes desatualizados.

Como o swift 1.0 veio com o objetivo de acelerar o desenvolvimento de aplicativos para iOS e MacOs ela além de mais rápida que o Objective-C possui novos recursos de linguagem, incluindo variáveis, constantes, tipo de interface, classes genéricas, funções, closures, tuplas e dicionários, a Swift era totalmente compatível com o Objective-C. Além de todas estas melhorias a Apple também lançou junto com o swift uma nova versão do Xcode que possui um recurso playground permitindo que os desenvolvedores testem e processem o código Swift em ambiente em tempo real.

Em 2015 com o lançamento do swift 2.0, teve mais melhorias para a linguagem veio novos recursos de segurança adicionais, o mais importante é que nesta versão a linguagem passou a possuir o paradigma orientado a protocolo,

permitindo que os desenvolvedores usem protocolos como interfaces, estendam protocolos e especifiquem métodos e propriedades. e no mesmo ano a Swift se tornou uma linguagem de programação open source.

Em 2016 a Apple lançou uma nova versão Swift 3.0 com grande mudanças no código, removeu palavras desnecessárias dos nomes das funções. Nesta versão passou a ser exigido a adição de rótulos para todos os parâmetros de função.

Em 2017 no swift 4.0 foi implementado Encodable, Decodable e Codable, e por fim em 2019 foi implementado a atual versão, Swift 5.0, compatível com Swift 4.0, Swift 4.1 e Swift 4.2, também com muitas melhorias: reimplementação do tipo String, imposição de acesso exclusivo à memória durante o tempo de execução, novos tipos de dados e suporte a tipos dinamicamente chamados, Atualizações do Gerenciador de Pacotes e o comando ‘swift run’ agora inclui a capacidade de importar bibliotecas em um REPL sem a necessidade de criar um executável.

3. Paradigmas

A linguagem Swift pode ser enquadrada em seis paradigmas diferentes: orientada a objetos, estruturada, imperativa, compilada, concorrente e funcional.

O paradigma de orientação a objetos é definido pela concepção de unidades que compõem as linguagens deste paradigma: os objetos. Objetos são utilizados e modificados como instâncias de si mesmos com operadores como “this” ou “self”, e, em geral, definem instâncias de classes, que definem o tipo, comportamento e estado destes objetos. Objetos também podem definir protótipos, que são cópias de outros objetos e não instâncias de classes.

A estruturação é um paradigma dado pela utilização, em geral, de estruturas como subrotinas (porções de código que executam funcionalidades específicas), laços de repetição, condicionais e estruturas em bloco. Swift utiliza deste conjunto de operações para a composição de códigos.

O paradigma imperativo é definido como a mudança de estados de um programa a partir de ações proporcionadas pelo programador. Isso torna a linguagem mais próxima da natural.

Compilação também se encaixa nos paradigmas de Swift. O compilador traduz o código para que o SO ou o processador possam executar o programa. No caso de Swift, é utilizada uma infraestrutura LLVM (máquina virtual de baixo nível), que é integrada ao XCode, IDE típica da linguagem.

O paradigma concorrente define a programação que envolve execuções simultâneas, com a execução de threads em apenas um programa ou programas separados.

Swift também se enquadra no paradigma funcional, que trata dos problemas como funções matemáticas, evitando mais a mudança de estado e focando na resolução destas funções.

4. Características Principais

As principais características da Swift sao sao Estabilidade, Modernidade, poder interatividade e rapidez. Swift incorpora várias funcionalidades de linguagens mais atuais, como opcionais, genéricos e tuples

‘ Como Swift foi projetada com o principal objetivo de se tornar a linguagem de programação principal para desenvolvimento de aplicações para os sistemas operacionais proprietários da Apple, foram priorizadas características como rapidez, estabilidade, segurança e modernidade.

Swift possui vários conceitos de linguagens modernas, como *Optionals, Generics e Tuples.* Ao longo dessas seção iremos aprofundar um pouco em cada uma dessas características citadas

4.1 Optionals

Optionals são parte fundamental da Swift. Um variável opcional pode conter um valor, ou não. Um tipo opcional é representado por **<nome-do-tipo>?**. A vantagem de ser usar tipos de opcionais está na sua segurança, já que são uma representação mais fiel do mundo real, principalmente na coerência na comunicação entre objetos que podem ou não receber parâmetros nulos. Por exemplo: um banco de dados possui uma tabela que não pode conter um valor nulo, mas a uma representação do mesmo objeto na nossa aplicação pode se anular. Caso tentasse-mos cadastrar esse objetos em nosso banco e ele contiver um valor nulo, obteremos um erro de execução, que deve ser tratado, ou mesmo causar uma interrupção. Ao declararmos nossa comunicação com um objeto não nulo, ao fazer uma chamada devemos fazer esse tratamento (unwrap), e garantir que um valor nulo não ocorra. Assim evitamos erros e possíveis interrupções na nossa aplicação.

4.2 Generics e Closures

Em Swift, podemos usar tipos genéricos em nossas funções e outras estruturas, o que nos ajuda a atingir um nível de abstração ainda mais alto. Quando declaramos uma função, podemos usar a marcação **<T>** logo após ao nome da função para definirmos um tipo genérico. Esse pode ser referenciado tanto nos parâmetros da função como em seu corpo, assim podemos receber parâmetros de tipos não especificados, mas ao mesmo tempo impor uma restrição entre eles.

Já closures são blocos de código que podemos enviar como variáveis para outras funções ou procedimentos, podendo assim alterar o seu comportamento. Por exemplo: podemos escrever uma função que ordena uma lista de inteiros, mas queremos decidir se essa ordenação será feita de maneira crescente ou decrescente. Podemos então adicionar a essa função uma closure como parâmetro,

e o teste que será efetuado na ordenação será uma chamada ao parâmetro real enviado.

5. Relacionamento com outras linguagens

5.1. Linguagens semelhantes

A linguagem Swift, no geral, é baseada em muitas linguagens de programação, Objective-C, C++, Ruby, Haskell, Python, Rust e muitas outras.

A linguagem lembra linguagens de script como Ruby e Python por usar poucas linhas de programação, linguagens como C++, Java e C# pelo uso de Generics, C# por tipos Optionals, Scala e Opa por inferência de tipos, C pelo uso de operadores e estados de controle, Haskell pelo tipo “Maybe”, tipos opcionais (na tentativa de tornar o código mais seguro, já que as assinaturas de operações, parâmetros/retornos, podem ser nulos e pode gerar interrupções anormais) como nas linguagens: Scala, Kotlin, Java 8, OCaml, F#, Haskell, Rust e muitas outras semelhanças a muitas outras linguagens.

Dentre todas as semelhanças, é importante destacar que a sintaxe de Orientação a Objetos é similar às das outras linguagens com “init” sendo o construtor da classe. É importante também pontuar que a linguagem com que mais se assemelha é aquela que deseja substituir: Objective-C.

Objective-C surgiu nos anos de 1980 com Cox e Love, para a sua empresa Stepstone, com o interesse de resolver os problemas de reusabilidade em projecto de software e programação. Era a principal linguagem usada pela Apple antes de Swift ser desenvolvida. Em essência, as duas linguagens são muito parecidas, utilizando os mesmos operadores, “&” por exemplo, é usado para lidar com operações de overflow e underflow, além da possibilidade de implementar operadores customizados no Swift para concatenação de strings e coleções como em Objective-C. Para simplificar a construção de loop, os dois idiomas usam operadores de intervalo e avaliação de lista, e utilizam o mesmo recurso “overloading de função” para reduzir a repetição do código. A sobrecarga permite que as funções forneçam valores de parâmetros padrão.

A possibilidade de ver o efeito imediato depois de mudar o código do aplicativo existe para as duas linguagens, assim como coleções digitadas para os dois e, finalmente, os dois possuem interoperabilidade, ou seja, Swift e Objective-C podem ser combinados e misturados no mesmo aplicativo.

5.2. Linguagens mais comparadas

Mesmo tendo Objective-C como a linguagem mais parecida, Swift é bastante comparado com ela, o que evidencia as suas diferenças. Dentre essas diferenças estão principalmente as falhas da linguagem Objective-C em relação às exigências do universo da computação atual.

Essas falhas geralmente estão ligadas à naturalidade da programação ou à performance da linguagem, o que o Swift satisfaz de forma precisa. Algumas dessas falhas incluem o tempo de execução do programa devido à compilação em tempo de execução, ou até mesmo a dificuldade de achar certos erros no código.

Em resumo, Swift é frequentemente comparado ao Objective-C justamente por conseguir preencher as lacunas e falhas presentes na segunda linguagem.

Figura 1 - Objective-C vs Swift comparison

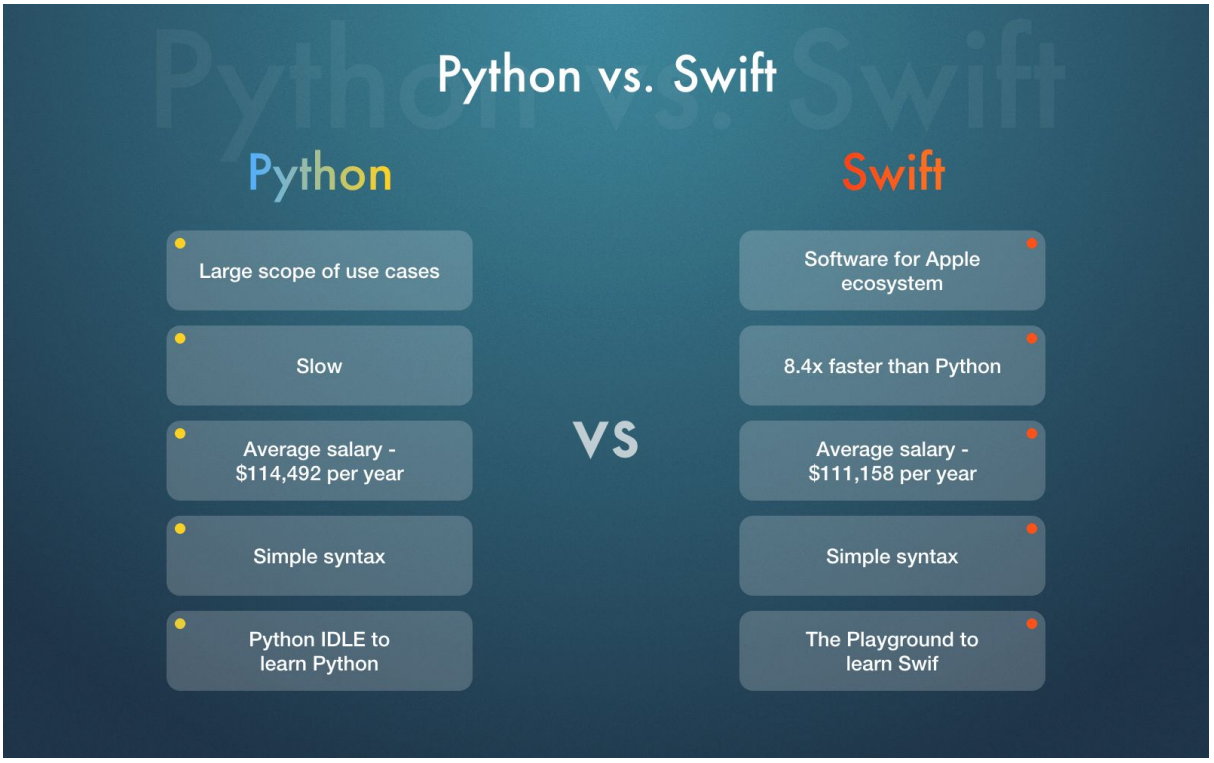
Fonte: (ALTEXSOFT…., 2018)

Outra linguagem bastante comparada com Swift é o Python, porém, o motivo da comparação é diferente da comparação com Objective-C, pois, nesse caso, a comparação é justamente devido à solução para diferentes aplicações.

Python é uma excelente linguagem para o desenvolvimento web pois é uma linguagem feita para ser usada em quase qualquer tipo de ambiente - conhecida como *general-purpose programming language* -, por esse motivo é bastante utilizada para o desenvolvimento web, porém é raramente utilizada para desenvolvimento mobile chegando a ser quase impossível encontrar aplicações feitas em Python.

Em contrapartida, Swift se destaca nesse cenário de desenvolvimento mobile, se tornando quase que uma solução perfeita para isso - em relação à Python -, porém, para o desenvolvimento web não é muito utilizada pois por ser uma linguagem muito nova e se limitar ao seu propósito principal, criar aplicações para iOS.

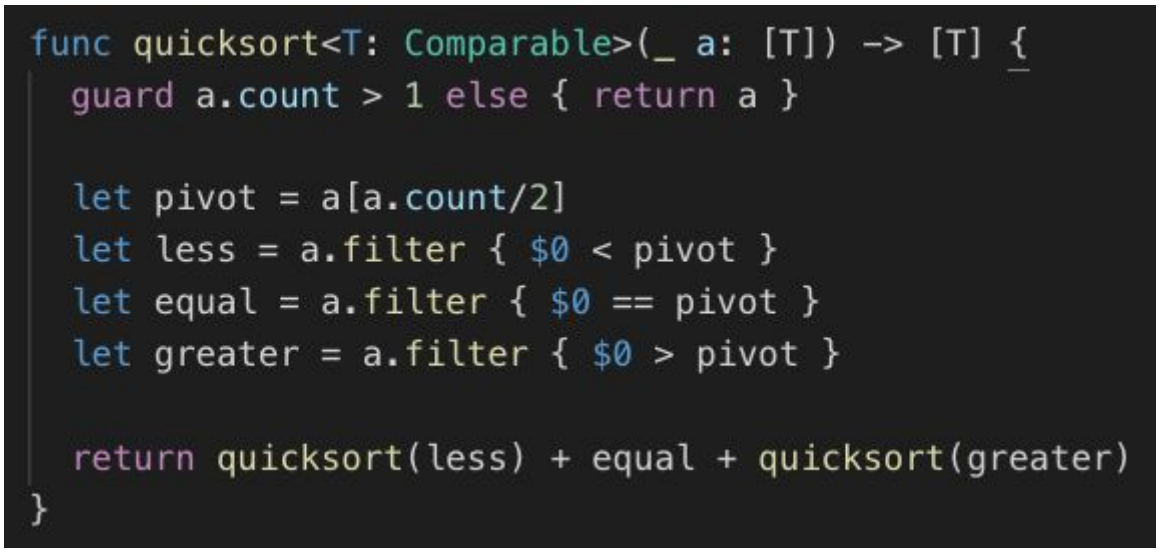
Figura 2 - Python vs Swift



Fonte: (CLEAVEROAD…., 2018)

6. Exemplos

Figura 3 - Quicksort usando tipo genérico em Swift.



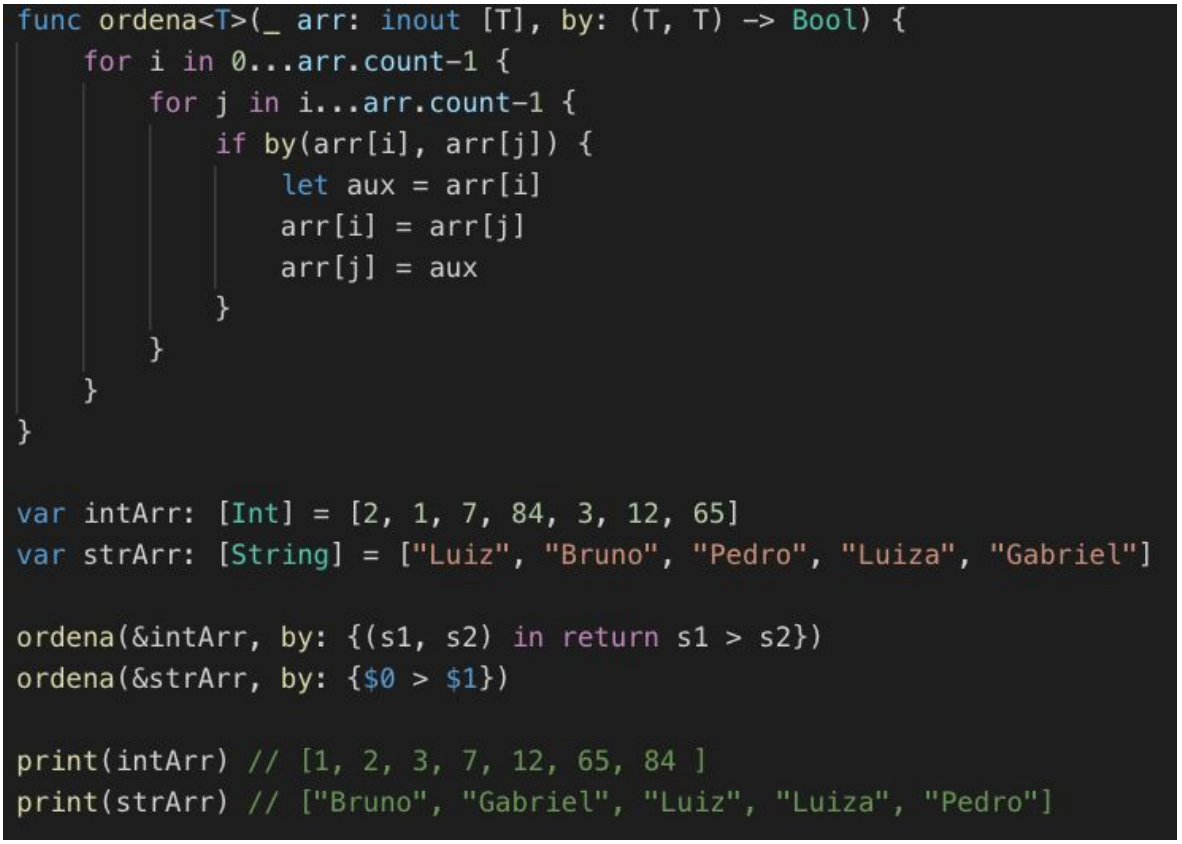
```swift
func quicksort<T: Comparable>(_ a: [T]) -> [T] {
  guard a.count > 1 else { return a }

  let pivot = a[a.count/2]
  let less = a.filter { $0 < pivot }
  let equal = a.filter { $0 == pivot }
  let greater = a.filter { $0 > pivot }

  return quicksort(less) + equal + quicksort(greater)
}
```

Fonte: GitHub

Figura 4 - Exemplo de Seleção com tipo genérico, closures e passagem por valor-resultado



```swift
func ordena<T>(_ arr: inout [T], by: (T, T) -> Bool) {
    for i in 0...arr.count-1 {
        for j in i...arr.count-1 {
            if by(arr[i], arr[j]) {
                let aux = arr[i]
                arr[i] = arr[j]
                arr[j] = aux
            }
        }
    }
}

var intArr: [Int] = [2, 1, 7, 84, 3, 12, 65]
var strArr: [String] = ["Luiz", "Bruno", "Pedro", "Luiza", "Gabriel"]

ordena(&intArr, by: {(s1, s2) in return s1 > s2})
ordena(&strArr, by: {$0 > $1})

print(intArr) // [1, 2, 3, 7, 12, 65, 84 ]
print(strArr) // ["Bruno", "Gabriel", "Luiz", "Luiza", "Pedro"]
```

Elaborada pelo grupo

Figura 5 - Exemplo de tratamento de opcionais na Swift

```swift
// Optionals e Unwrap

var pessoa: String? = nil

func comprimenta(_ nome: String = "Pessoa", comprimento frase: String) {
    print(frase + ", " + nome + "!")
}

comprimenta(comprimento: "Bom Dia") // Bom dia, Pessoa!

comprimenta(pessoa, comprimento: "Bom Dia") // Bom dia, Pessoa!
comprimenta(pessoa!, comprimento: "Bom Dia") // ERRO de execução

if let pessoa = pessoa {
    comprimenta(pessoa, comprimento: "Bom Dia") // Não é executado
}

comprimenta(pessoa ?? "Marco", comprimento: "Bom Dia") // Bom dia, Marco!
```

Elaborada pelo grupo

7. Considerações Finais

Nesse trabalho, queríamos demonstrar de forma compacta a linguagem de programação Swift. No fim, conseguimos sintetizar a linguagem demonstrando suas principais características, histórico, paradigmas e linguagens que possuem algum tipo de relacionamento com ela.

Ao longo do desenvolvimento do trabalho descobrimos a utilidade e facilidade do Swift, as diferentes tipagens, o controle de acesso a orientação a objetos e outros. Conseguimos inferir nosso aprendizado na linguagem, em relação à inferência de variável, aos paradigmas, estruturas, generalização e outros.

O que nos chamou a atenção sobre Swift é o fato de ser uma linguagem bastante nova e, por ter mais facilidade de uso e ser menos propensa a erros, conquista programadores que pertencem a gerações mais novas. Sua comunidade cresce mais todos os anos, sendo, de acordo com uma pesquisa do Stack Overflow em 2019, a 6ª linguagem mais amada por programadores. Hoje é considerada uma linguagem segura, de fácil manutenção, inovadora, com custo benefício e de alto potencial.

Por fim, esperamos que esse trabalho seja esclarecedor sobre os principais pontos da linguagem Swift, a ponto de sabermos o suficiente caso precisemos da linguagem algum dia e que desperte no leitor curiosidade para pesquisar mais ao fundo.

8. Bibliografia

Apple Inc. ***The Swift Programing Language (Swift 5 Edition).*** **Disponível em: <https://swift.org/documentation/>**

CLAITON, Craig e **WALS**, Donny. ***Complete iOS 12 Development Guide: Become a professional iOS developer by mastering Swift, Xcode 10, ARKit, and Core ML.***

WANG, Wallace. *Pro iPhone Development with Swift 5: Design and Manage Top Quality Apps.*

HOFFMAN, Jon. *Mastering Swift 5 (5a Edição)*

9. Referências

EDUCATION ECOSYSTEM. **Swift History**. Disponível em: <https://www.education-ecosystem.com/guides/programming/swift/history> Acesso em 28 de maio de 2019

CLEVERISM. **Swift**. Disponível em: <https://www.cleverism.com/skills-and-tools/swift/> Acesso em 28 de maio de 2019

DATTA, Sourav. **Which programming language is Swift similar to, and how do they compare to each other?** 14 jun. 2014. Disponível em: <https://www.quora.com/Which-programming-language-is-Swift-similar-to-and-how-do-they-compare-to-each-other> Acesso em 28 de maio de 2019

RUBENS, Paul. **10 things you should know about Apple's Swift**. 21 jul. 2014 Disponível em: <https://www.javaworld.com/article/2456964/10-things-you-should-know-about-apples-swift.html> Acesso em 28 de maio de 2019

THIAGO. **Desenvolvimento IOS: conheça a linguagem swift.** 2014. Disponível em: <https://www.devmedia.com.br/desenvolvimento-ios-conheca-a-linguagem-swift/31860> Acesso em 28 de maio de 2019

ALTEXSOFT. **Swift vs Objective-C: Out with the Old, In with the New.** 28 jun. 2018.Disponível_em: <https://www.altexsoft.com/blog/engineering/swift-vs-objective-c-out-with-the-old-in-with-the-new/> Acesso em 28 de maio de 2019

POPKO, Aleksander. **Swift vs Objective C: iOS' Programming Languages Compared.** 14 Feb.2018. Disponível em: <https://www.netguru.com/blog/swift-vs-objective-c-ios-programming-languages-compared> Acesso em 28 de maio de 2019

REDBYTES. **Importance of the Programming Language 'Swift' in iOS Development.** 21 mar. 2017. Disponível em: <https://www.redbytes.in/importance-programming-language-swift-mobile-app-development/> Acesso em 28 de maio de 2019

PATIL, Ritesh. **Similarities And Differences Between Objective-C And Swift Programming.** 1 Mar. 2017. Disponível em: <https://medium.com/@ritesh.patil732/similarities-and-differences-between-objective-c-and-swift-programming-ab15d215b136> Acesso em 28 de maio de 2019

JAGTAP, Shashikant. **Mobile Developers Takeaways From Stack Overflow Developer Survey 2019.** 11 abr. 2019. Disponível em: <https://medium.com/xcblog/mobile-developers-takeaways-from-stack-overflow-developer-survey-2019-c0d04f6eb902> Acesso em 28 de maio de 2019

MEDIUM. **The Evolution of Swift**. Disponível em: <https://medium.com/@mindfiresolutions.usa/the-evolution-of-swift-29e7a89f1a0c.> Acesso em 28 de maio de 2019

WIKIPEDIA.**Swift (linguagem de programação)**. Disponível em: <https://pt.wikipedia.org/wiki/Swift_(linguagem_de_programa%C3%A7%C3%A3o)> Acesso em 28 de maio de 2019

TUTSPLUS. **Swift from Scratch: Parâmetros, tipos e encadeamento de funções**. Disponível em: <https://code.tutsplus.com/pt/tutorials/swift-from-scratch-function-parameters-types-and-nesting--cms-23056>Acesso em 28 de maio de 2019

IMASTERS. **Falando sobre Swift – Funções.** Disponível em: <https://imasters.com.br/back-end/falando-sobre-swift-funcoes> Acesso em 28 de maio de 2019

ARSTECHNICA. **A fast look at Swift, Apple's new programming language.** Disponível em: <https://arstechnica.com/gadgets/2014/06/a-fast-look-at-swift-apples-new-programming-language/>Acesso em 28 de maio de 2019

TECNOBLOG. **Surpresa na WWDC: Apple anuncia Swift, sua nova linguagem de programação.** Disponível em: <https://tecnoblog.net/157711/apple-linguagem-programacao-swift/>

GITHUB. Disponível em: <https://github.com/raywenderlich/swift-algorithm-club/tree/master/Quicksort> Acesso em 28 de maio de 2019

CLEVEROAD. **Swift vs Python: Which of Them is More Promising in 2019?.** Disponível em: <https://www.cleveroad.com/blog/python-vs-swift> Acesso em 28 de maio de 2019