

**CENTRO UNIVERSITÁRIO CARIOCA**

**RAPHAEL LIMA VICARONE**

**THIAGO FERNANDES DA SILVA ALMEIDA**

**LINGUAGEM DE PROGRAMAÇÃO GO: HISTÓRIA, ANÁLISE E CONSTRUÇÃO**

**DE UMA APLICAÇÃO**

**RIO DE JANEIRO**

**2021RAPHAEL LIMA VICARONE**

**THIAGO FERNANDES DA SILVA ALMEIDA**

**LINGUAGEM DE PROGRAMAÇÃO GO: HISTÓRIA, ANÁLISE E CONSTRUÇÃO**

**DE UMA APLICAÇÃO**

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Carioca, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. D.Sc Sérgio Assunção Monteiro

**RIO DE JANEIRO**

**2021**

RAPHAEL LIMA VICARONE

THIAGO FERNANDES DA SILVA ALMEIDA

LINGUAGEM DE PROGRAMAÇÃO GO: HISTÓRIA, ANÁLISE E CONSTRUÇÃO DE UMA APLICAÇÃO.

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Carioca, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação

.

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Sérgio Assunção Monteiro, D.Sc - Orientador

Centro Universitário Carioca

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. André Luiz Avelino Sobral, M.Sc

Centro Universitário Carioca

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Alberto Tavares da Silva, D.Sc

Centro Universitário CariocaAGRADECIMENTOS

Jhsdgwruohg ekjgsogijsg skgjlsigja lrjhg lsjglsgj irirfuhfg itijrigj erg kerjg kejrgeirjg eigj ekjgoirgjwpoeijgpw oirgjwoeirjgwçejg wet2gweg

Wet rjehgowrgjwirgj kljgço irgj weoirg ojrlgqjorgij wreoiwjergu5uuurjg wierjgweuhoujgijr53yt298rug w985uo u hwrohwpo 8r towurt uoitworut 985 rjtiowetw0985u twoierjg weprigj w85u wlekgwçeoritjwçoit v

Weoit woeirjt wortjw peuirtj8ijtirjtijtijjirjituw´085uwe5jtp8u´t09a~pifjçofghjwoeif jg eroigjw98u03jpifjg reijg woeirjgpw9urtjwçerjtwo irjtoeirjtoeirjgo erijt oi vov otrjpwoirtpw858tujoergjwoeirtjpwoerjtwpiorejtv woeritjwçoeit´wety

E owierjt owiertpwiuerjtlekjgoeijt eijrt bijt wpeoritjeltkjy eiyjoeijgoiiiirtworutw9898gso ijg woeirjt woeirjt woeijgefjgirjgiowjíjwe5w6teg´wekjg rekthjt oirt vo toijeroughiewruthw poruht vurhtweoritjtwçoritjwrijt

Oitjowritjwoçritjwçertijtw oiuehhhhtorwuthp94htworitj thoritjwoeritjw rijtwçoerijt irojtçworitjwr tiwjretowierjtoerijtoeirjoejrwruthpwiuth ruthpowurth uthpruthpu ruhtu ruhtuqwrhtrj rhtoprth rhtopurhtpuht ourhturhtporuh.

**RESUMO**

Desenvolver software nunca foi uma tarefa fácil. Escrever um programa, não importa o quão pequeno seja, requer uma dose alta de paciência e atenção aos detalhes e as vezes parece não haver solução para o problema. E nada disso parece ter mudado ao longo do tempo. Novas linguagens de programação e novas ferramentas todos os dias são criadas por programadores em todo o mundo e o desafio não diminuiu, pelo contrário, o numero de núcleos dos processadores vem aumentado mesmo em dispositivos móveis, desenvolvedores e administradores de sistema sofrem para aproveitar ao máximo a máquina e não afetar as tarefas dos usuários.

Conforme o número de usuários aumenta, fica mais difícil encontrar alguma linguagem de programação que resolva todos esses problemas facilmente. A linguagem Go aparece em um ambiente onde tais problemas são ainda mais extremos: Google.

Embora a linguagem Go seja inspirada na linguagem C, ela possui recursos de alto nível, como abstração de certas estruturas de dados, coleta de lixo e duck typing, além de trazer métodos modernos e elegantes para a criação de aplicativos concorrentes. Go também inclui uma extensa biblioteca padrão com ferramentas para comunicação em redes, servidores HTTP, expressões regulares, leitura e escrita de arquivos e muito mais.

O objetivo desta pesquisa é apresentar os recursos da linguagem Go e partes importantes da biblioteca padrão, uma análise de sua sintaxe e semântica, apresentar sua história e objetivos em sua concepção, além da construção de uma aplicação completa, uma Rede Social.

**Palavras chave:** Google, Sintaxe, Semântica, Goroutines, Paradigma Funcional, Concorrência, Escalabilidade.

**ABSTRACT**

Developing software has never been an easy task. Writing a program, no matter how small, requires a high dose of patience and attention to detail and sometimes there seems to be no solution to the problem. And none of that seems to have changed over time. New programming languages ​​and new tools are raised every day by programmers around the world and the challenge has not diminished, on the contrary, the number of cores for those adapted has increased even on mobile devices. , developers and system administrators fully recover the machine and do not affect users' tasks.

As the number of users increases, it becomes more difficult to find any programming language that solves all these problems easily. The Go language appears in environment where such problems are even more extreme: Google.

Although the Go language is inspired by the C language, it has high-level features, such as abstraction of data structures, garbage collection and duck typing, in addition to bringing modern and elegant methods for creating competing applications. Go also includes an extensive standard library with tools for communicating over networks, HTTP servers, regular expressions, reading and writing files and more.

The objective of this research is to present the resources of the Go language and important parts of the standard library, an analysis of its syntax and semantics, to present its history and objectives in its conception, in addition to the construction of a complete application, a Social Network.

**Keywords:** Google, Syntax, Semantics, Goroutines, Functional Paradigm, Competition, Scalability.

**LISTA DE ILUSTRAÇÕES**

**Figuras**

[Figura 1 - figura 15](#_Toc515606986)

[Figura 2 - figura 20](#_Toc515606987)

[Figura 3 - figura 21](#_Toc515606988)

[Figura 4 - figura 24](#_Toc515606989)

[Figura 5 - figura 25](#_Toc515606990)

[Figura 6 - figura 26](#_Toc515606991)

[Figura 7 - I figura 27](#_Toc515606992)

[Figura 8 - T figura 28](#_Toc515606993)

[Figura 9 - figura 29](#_Toc515606994)

[Figura 10 - figura 30](#_Toc515606995)

[Figura 11 - figura 32](#_Toc515606996)

[Figura 12 - figura 34](#_Toc515606997)

[Figura 13 - figura 36](#_Toc515606998)

[Figura 14 - figura 37](#_Toc515606999)

[Figura 15 - figura 38](#_Toc515607000)

[Figura 16 - figura 39](#_Toc515607001)

[Figura 17 - figura 40](#_Toc515607002)

[Figura 18 - figura 42](#_Toc515607003)

[Figura 19 - figura 43](#_Toc515607004)

[Figura 20 - figura 44](#_Toc515607005)

[Figura 21 - figura 45](#_Toc515607006)

[Figura 22 - figura 46](#_Toc515607007)

[Figura 23 - figura 47](#_Toc515607008)

[Figura 24 - figura 48](#_Toc515607009)

[Figura 25 - figura 49](#_Toc515607010)

[Figura 26 - figura 50](#_Toc515607011)

**Sumário**

1. INTRODUÇÃO 10

1.1. Porquê GO 11

2. ESTRUTURA DO TRABALHO 12

1. **INTRODUÇÃO**

Go é uma linguagem extremamente simples que favorece a criação de programas rápidos, leves, confiáveis, fáceis de entender, documentar e de manter. A primeira  impressão é de que lhe falta funcionalidades mais complexas que outras linguagens possui, no entanto, o intuito de seus criadores foi exatamente o inverso, que GO fosse uma linguagem simples onde só existe uma forma ou poucas formas  de realizar uma tarefa, sendo assim, GO é uma linguagem fácil de aprender e que não possui muito a gerar discussão entre os programadores  do tipo, essa funcionalidade é melhor para tal tarefa.

Não é uma linguagem pensada para programas curtos, mas é muito bem otimizada para programas escaláveis, pois foi criada pela gigante Google para a própria Google, pensando em uma linguagem que otimize e melhore a produção e compilação de código por seus programadores em suas milhares de linhas de código.

Não existe linguagem que seja capaz de resolver todos os problemas de forma fácil, porém o GO foi criado em um ambiente em que esse problemas são extremamente grandes, o Google.

Apesar de bastante inspirada na linguagem C, Go possui características de mais alto nível, como abstrações para algumas estruturas de dados, garbage collection e duck typing, além de  trazer uma  abordagem moderna e elegante para a criação de aplicações concorrentes. Go também inclui uma extensa biblioteca padrão com ferramentas para comunicação em redes, servidores HTTP, expressões regulares, leitura e escrita de arquivos e muito mais.

**1.1 PORQUE GO?**

Go nasceu como um projeto interno no Google, iniciado  em 2007 por  Rob Pike, Ken Thompson e Robert Griesemer, e posteriormente lançado como um projeto de código aberto em novembro de 2009. Pike e Thompson já haviam trabalhado juntos no Bell Labs, o berço do Unix e do Plan 9, o sistema operacional que deu origem às ferramentas de compilação usadas  como base do desenvolvimento da linguagem Go.

No artigo "Go at Google: Language Design in  the Service of Software Engineering" (<http://talks.golang.org/2012/splash.article>), Rob Pike  atribui os longos períodos de compilação e a dificuldade em escalar o desenvolvimento de grandes aplicações como sendo os principais motivadores para a criação da nova linguagem.  Segundo ele, a maior parte do problema é a forma com que as linguagens C e C++ tratam o gerenciamento de dependências entre os diversos arquivos-fonte.

Dentro deste cenário, Go foi concebida com o objetivo de tornar o desenvolvimento de servidores no Google uma tarefa mais produtiva e eficiente.

Para evitar os problemas de compilação, Go  implementa um controle rigoroso e inteligente de dependências, baseado na definição e uso  de packages (pacotes).

Com uma sintaxe bastante limpa, se comparada com outras linguagens inspiradas em C, como C++ e Java, Go permite a escrita de programas concisos e legíveis, além de facilitar bastante a escrita de ferramentas que interagem com o código-fonte. Bons exemplos de tais ferramentas são o

go fmt  (formata o código de acordo com o guia de estilo da linguagem) e go fix (reescreve partes do código que usa APIs depreciadas para que usem as novas APIs introduzidas em versões mais recentes).

Go possui tipagem forte e estática, porém introduz uma forma curta de declaração de variáveis baseada em  inferência de tipos, evitando redundância e produzindo código muito mais sucinto do que linguagens estaticamente tipadas tradicionais. Além disso, Go traz uma implementação de duck typing: se faz "quack" como um pato e anda como um pato, então provavelmente é um pato. Baseado em interfaces, permitindo a criação de tipos bastante flexíveis. Alguns tipos de coleção de dados como slices, listas de tamanho dinâmico, e maps, dicionários de dados associativos, são nativos à linguagem, que também fornece um conjunto de funções embutidas para a manipulação destes tipos, arrays, listas de tamanho fixo, também estão disponíveis para casos em que um nível maior de controle é necessário.

Go suporta o uso de ponteiros, referências a endereços de memória, o que torna ainda mais fácil a criação de poderosos tipos customizados. Entretanto, aritmética sobre ponteiros não é suportada.

Apesar de toda a flexibilidade, através do uso de um coletor de lixo Go reduz drasticamente a complexidade no gerenciamento de memória das aplicações, tornando-as mais robustas e evitando vazamentos descuidados. Go também fornece recursos que permitem a escrita de programas totalmente procedurais, orientados a objetos, através da definição de novos tipos e de funções  que operam sobre tais tipos, ou funcionais, funções são membros de primeira classe em Go, que também suporta a criação de closures, funções que herdam o contexto de onde elas foram definidas.

A abordagem de Go para concorrência é um dos maiores diferenciais da linguagem. Inspirada no famoso paper de C.A.R. Hoare Communicating Sequential Processes [ref csp-hoare], Go implementa goroutines que são processos extremamente leves que se comunicam através de channels, evitando o uso de memória compartilhada e dispensando o uso de travas, semáforos e outras técnicas de sincronização de processos.