UNIVERSIDADE DE ITAÚNA

DAVI VENTURA CARDOSO PERDIGÃO
EDMILSON LINO CORDEIRO

Go e TypeScript

Linguagens de Programação

ITAÚNA
2022

SUMÁRIO

1. INTRODUÇÃO

1.1. Go:

Go é uma linguagem de programação criada pela **Google** e lançada em **código livre** em novembro de **2009**. Sua origem deu-se quando a Google estava com um problema, pois muitos dos seus sistemas eram feitos em **C++** e em **C**, e o processo de compilar esses programas para gerar um executável era **complicado** e **demorado**. Com isso, os engenheiros do Google tiveram a ideia de criar uma nova linguagem de programação. Sua **popularidade** vem crescendo cada vez mais e acredita-se que, no futuro, a Go possa **substituir** o **Java**, uma das linguagens de programação mais populares para o desenvolvimento de software.

Essa linguagem, diferentemente de outras como Python e Java, é uma **linguagem moderna**, visto que ela foi criada aproximadamente **20 anos depois** que a maioria das outras linguagens. Nesses 20 anos, a computação **evoluiu**, e essas evoluções foram **incorporadas** no Go. A primeira dessas evoluções faz com que o processo de compilação de um programa em Go seja bem **rápido**, justamente sanando o problema da Google. Outra questão é a **modularização** da linguagem, com as suas funcionalidades espalhadas em pacotes, que são importados na nossa aplicação conforme a nossa **necessidade**. Também vale ressaltar que é uma linguagem **esteticamente tipada**, além de permitir **programação simultânea** (programação paralela).

Além disso, a sua sintaxe tem em torno de **25 palavras-chave**, ou seja, uma sintaxe bem curta, fazendo com que o nosso código fique **simples** de se entender. Apesar da linguagem ser simples, ela é bem **convencionada**, já que o Go foi feito para ser uma linguagem **rápida** de **desenvolvimento**, então há certas **convenções** (algumas delas forçadas pela linguagem) que fazem com que o foco do programador seja em **desenvolver** código, e não em **discutir como** ele deve ser feito.

1.2. TypeScript:

TypeScript é uma linguagem de programação de código aberto desenvolvida pela **Microsoft** em **2012**. Mais conhecido como um **superset** do **Javascript**, ou seja, um conjunto de ferramentas, o TypeScript foi criado com o objetivo de incluir recursos que não estão presentes no JS. Por meio dele é possível definir a **tipagem estática**, parâmetros e retorno de funções. Foi considerada pelo público a **4ª** linguagem **"mais amada"**, de acordo com uma pesquisa conduzida pelo site **Stack Overflow** em **2018**, e está entre as **15** linguagens mais **populares**, de acordo com uma pesquisa conduzida pela **RedMonk**.

Além de ser uma ferramenta **orientada a objetos**, **fortemente tipada** e que pode ser **escrita** em qualquer **ambiente de desenvolvimento**, o TypeScript quando instalado via gerenciador de pacotes JS, permite **checar erros** e utilizar outros compiladores que suportam este mecanismo.

O nome TypeScript surgiu da combinação de palavras *"JavaScript"* + *"Type"* ("Tipo" em português), representa a sua finalidade mais importante: **a tipagem estática**, na qual possibilita **programar** tanto do lado do **cliente** (client-side), como no lado do **servidor** (server-side). Logo, o TypeScript eleva o nível de **produtividade** e ainda garante o desenvolvimento de aplicações **complexas**, **eficazes** e **seguras**. Em suma, o código é **"transformado"** (no termo técnico: transcompilado) em JavaScript "puro" antes de ser executado.

2. CARACTERÍSTICAS DAS LINGUAGENS

2.1. Go:

O uso de Go vem **crescendo** em todo o mundo, especialmente no ambiente de **Cloud Computing.** Alguns dos principais projetos de infraestrutura em nuvem escritos em Go são **Docker** e **Kubernetes**. O Go Wiki mantém atualizado frequentemente uma lista de **empresas** que fazem **uso** da linguagem Go, além do próprio Google, são elas:

- **Mercado Livre**
- **iFood**
- **Heroku**
- **Nokia**
- **SoundCloud**

É importante estar ciente que cada tecnologia possui suas **vantagens** e **desvantagens**. Saber os pontos fracos e fortes da linguagem ajuda a escolher sabiamente qual é a melhor linguagem para cada projeto.

Vantagens:

- **A facilidade de uso:** A linguagem apresenta uma sintaxe fácil de usar. Isso torna fácil a manutenção do código.
- **Linguagem de segurança:** como Go é uma linguagem de tipo estático, há menos probabilidades de erro. Essa é uma vantagem sobre as

linguagens tipadas dinamicamente com um grande número de tipos de variáveis e maiores chances de erros de codificação complexos. Além disso, o código escrito em Go é mais simples e fácil de depurar. A combinação desses fatores reduz as vulnerabilidades de segurança do aplicativo.

- **Excelente documentação:** Go não só tem uma excelente documentação, mas vai um passo além. Os desenvolvedores que não documentam seu código o suficiente recebem alertas para colocar em prática a documentação necessária. Isso aumenta a capacidade de manutenção do código escrito em Go.

Desvantagens:

- **Menos versátil:** embora a simplicidade do Go ofereça muitas vantagens, ele também mantém afastadas algumas funcionalidades poderosas de alto nível. Isso torna a linguagem menos versátil quando você a compara a linguagens mais complexas. Go ainda não oferece suporte a genéricos.
- **Consome mais recursos computacionais:** como não há máquina virtual para esta linguagem, os arquivos Go podem ser muito maiores do que algumas das linguagens de programação líderes de mercado. Isso faz com que o Go consuma mais memória RAM.
- **Construir um grande ecossistema para Go levará tempo:** ao contrário de algumas das principais linguagens de programação, Go não se tornou indispensável em nenhuma área. Como resultado, a adoção demora mais. Naturalmente, levará mais tempo para construir um grande ecossistema.

2.2. TypeScript:

O TypeScript é utilizado por programadores que querem criar **funções tipadas** e **bem modeladas** e que gostam de garantir que suas aplicações não tenham qualquer **problema**. Naturalmente, por possuir mais recursos do que o **JavaScript**, o **TypeScript** é considerado uma versão **melhor** pelos desenvolvedores.

Ao utilizarmos o Typescript, temos a possibilidade de aplicar a **tipagem estática** à programação juntamente com interfaces em um sistema construído unicamente com Javascript, ou seja, podemos **turbinar** as nossas aplicações. Entre esses conceitos estão:

- **Encapsulamento:** forma de estruturar o código para que um determinado bloco tenha pontos de acesso específicos para o ambiente externo, o que garante a visibilidade e acessibilidade controladas dos elementos internos da classe.

- **Herança:** segundo o princípio da herança, uma classe (filha) pode herdar de outra (pai) características e comportamentos já definidos nessa segunda, sem necessidade de redefinição.
- **Abstração:** capacidade de destacar as características dos elementos do mundo real que serão úteis para o sistema. Essas características são reunidas na forma de classes, que representam as “coisas” reais a serem utilizadas no domínio do problema.
- **Polimorfismo:** conceito a partir do qual objetos podem assumir formas diferentes em determinadas situações, mas mantendo uma relação com sua definição inicial de nível mais alto.

Vantagens

- **Apresenta erros no momento da organização**, enquanto o JavaScript, no tempo de execução.
- **Tipagem estática:** valiosa para ajudar a arquivar capacidades, explicar a utilização e diminuir a sobrecarga psicológica (dicas de tipo de interface e obtenção de erros esperados de programação contínua).
- **Executado em qualquer programa ou motor JavaScript**.
- **Ajuda na organização do código.**
- **O TypeScript tem uma documentação melhor para APIs** que estão em um estado de harmonia com o código-fonte. Algumas organizações relatam uma **diminuição nos bugs** quando mudam para o TypeScript.

Desvantagens

- Ao utilizar uma biblioteca externa, deve haver um **documento de definição** e, de vez em quando, ele **não está acessível**.
- **A natureza dos documentos de definição de tipo é uma preocupação**.
- Em qualquer ponto que o TypeScript precise ser executado em um programa, deve haver uma etapa de reunião para **alterar o TypeScript para JavaScript**.
- **TypeScript não é totalmente coexpressivo com JavaScript**. Os destaques ausentes incluem: HOFs, composição, genéricos com chave superior.

3. CRITÉRIOS DE AVALIAÇÃO DE LINGUAGENS

3.1. Go:

3.1.1. Legibilidade:

Simplicidade geral

A Go em sua criação, baseou-se no conceito de sintaxe simplificada e eficiente, fornecendo ao desenvolvedor apenas os recursos suficientes para uma aplicação desejada, bem como facilita manutenções póstumas.

Ortogonalidade

Go proporciona a liberdade para se utilizar N ferramentas para aplicações, permitindo fácil interpretação sobre códigos, graças a seus tipos de dados e sua sintaxe. Também permite declarar variável sem necessitar declarar seus tipos - exceto em métodos - e permite a manipulação de arrays através de Slices.

Instruções de controle

Na Go, as instruções de controle possuem semântica simples. Comandos como *if/else, switch/case, while e for* podem ser trabalhadas nesta linguagem.

Tipos de dados

Go dispõe de tipos inteiros, pontos flutuantes, double, booleanos, strings char e, no caso do float, deve ser declarado como float64.

Sintaxe

Durante a construção da linguagem, uma das premissas norteadoras do GoLang foi proporcionar uma linguagem cuja sintaxe fosse de fácil compreensão com as linguagens C e Java, proporcionando melhor adaptação para a linguagem.

3.1.2. Capacidade de escrita:

Suporte para abstração

Apesar de que Go não seja orientada a objeto, a linguagem realiza a abstração de dados através de suas structs de tipos, tendo o mesmo propósito de uma classe, onde a struct representa o estado e funções vinculadas representam o seu comportamento.

Go também dispõe de pacotes (packages) que une os pacotes padrão da linguagem e módulos criados pelo usuário sem a necessidade de implementar dentro das inclusões de bibliotecas.

Expressividade

Através do construtor ":=", Go associa variáveis e valores sem necessitar declarar seus tipos - exceto em Métodos.

Outra função presente em Go é o "fallthrough" (queda), que possui efeito inverso ao break na linguagem C. Este efeito ocorre ao ser declarado switch e case em GoLang, que não são resolvidos em cascata, utilizando o fallthrough somente quando for necessário.

3.1.3. Confiabilidade:

Verificação de tipos

Por ser uma linguagem fortemente tipada, Go realiza a checagem de tipos em tempo de compilação, a fim de que não ocorram bugs enquanto executa o código.

Manipulação de exceções

Go possui funções com o intuito de proteger o código de cometer bugs em caso de aplicações complexas, como sistemas de segurança. Destaca-se: *Defer, Panic, Recover, Aliasing.*

3.1.4. Custos:

Aprendizagem

Em decorrência da sua praticidade, simplicidade e ampla documentação oficial, Go proporciona melhor nível de aprendizagem para desenvolvedores, de juniors a seniors, devido sua familiaridade com C.

Compilação

Uma das premissas da linguagem, sua compilação é feita de forma rápida, diminuindo portanto o seu custo.

Execução do programa

Através de goroutines (ou threads), acaba por economizar recursos computacionais, sendo tão rápida quanto a linguagem C.

Sistema de implementação

Possui suporte a todos os sistemas operacionais, cabendo ao desenvolvedor instalar conforme sua arquitetura (x64).

Sistemas críticos

Devido a sua simplicidade, Go é recomendado para sistemas de segurança (como bancos, por exemplo), entretanto, peca na aplicação em sistemas de tempo real, como controle de tráfego aéreo e rastreamento veicular.

3.2. TypeScript:

3.2.1. Legibilidade:

Simplicidade geral

É algo extremamente simples escrever códigos em Javascript. Este é um dos muitos fatores que dão tanta popularidade ao TypeScript atualmente. Muitos desenvolvedores estão migrando suas bases de código de JavaScript para TypeScript por conta dessa simplicidade.

Ortogonalidade

TypeScript proporciona a liberdade para se utilizar N ferramentas para aplicações, permitindo fácil interpretação sobre códigos, graças a sua sintaxe. Além do mais, os tipos podem ser adicionados incrementalmente a uma aplicação JavaScript existente. Em determinadas aplicações, principalmente as maiores, isso implica em segurança ao adicionar um bom sistemas de tipos, o que pode ser de grande valor.

Instruções de controle

No TypeScript, temos algumas estruturas de decisão que podem nos ajudar na programação no lado do cliente. Comandos como *if/else* e *switch* podem ser trabalhados nesta linguagem.

Tipos de dados

Uma das maiores diferenças do TypeScript para o JavaScript é a tipagem de suas variáveis. Em JavaScript podemos atribuir um valor numérico para uma variável e logo depois mudar este valor para booleano, para uma string, um array ou mesmo um objeto. Já no TypeScript temos os tipos bem definidos e são eles: *String, Boolean, Number, Array, Tuple, Enum, Any* e *Void*.

Sintaxe

A POO, por exemplo, sempre foi um problema ao ser aplicada em JavaScript, devido a sua sintaxe não permitir escrever classes, por exemplo, de forma tão clara, além da fraca tipagem de dados. Para isso, o TypeScript oferece uma forma de corrigir ou contornar esses problemas, adicionando funcionalidades que quando compiladas resultarão em código JavaScript novamente. Porém, agora o desenvolvedor lidará diretamente com uma sintaxe simplificada, mais clara e amplamente suportada por editores de código modernos.

3.2.2. Capacidade de escrita:

Suporte para abstração

Com TypeScript dispomos de recursos que melhor suportam o uso da Programação Orientada a Objetos, que tem como base quatro princípios fundamentais, sendo um deles justamente a abstração.

Expressividade

O TypeScript está introduzindo inovações desde 2019 para realmente descrever a expressividade do JavaScript em seu sistema de tipos. Entre tipos condicionais, novas verificações mais rígidas, ferramentas gerais e algum trabalho experimental no gerenciamento de projetos cruzados.

3.2.3. Confiabilidade:

Verificação de tipos

Consiste em *classes, interfaces, herança,* etc. É estrita e tipificada estaticamente como Java. As interfaces são usadas para definir contatos no texto datilografado. Em geral, define as especificações de uma entidade.

Manipulação de exceções

O TypeScript permite tratar uma exceção, de forma similar ao C#. Basicamente temos três blocos: *try, catch, finally*.

No *"try"*, colocamos o algoritmo que deve ser executado. Já no bloco do *"catch"*, devemos colocar o algoritmo que deve ser executado quando houver uma exceção. E por último, no bloco de *"finally"* (que é opcional), podemos colocar algum algoritmo que deverá sempre ser executado, independente de haver exceção ou não.

3.2.4. Custos:

Aprendizagem

Em decorrência da sua praticidade e simplicidade, o TypeScript proporciona melhor nível de aprendizagem para desenvolvedores, de juniors a seniors, devido sua familiaridade com JavaScript.

Compilação

O arquivo TypeScript não roda diretamente no navegador enquanto o JavaScript é executado. Sendo assim, temos que compilar o arquivo TypeScript em JavaScript, então ele funcionará normalmente. Para compilar e executar primeiro, precisamos instalar o Node e usá-lo para instalar o TypeScript globalmente em seu sistema local.

Sistema de implementação

Possui suporte a todos os sistemas operacionais, cabendo ao desenvolvedor instalar conforme sua arquitetura (x64).

Sistemas críticos

Nestes casos usar tipagem forte, estática, e uma linguagem que facilite a escrita de um código bem estruturado pode fazer toda diferença entre pegar um bug em tempo de desenvolvimento, ou descobri-lo em produção após uma invasão que custe muito caro financeiramente à empresa. Sendo assim, esse seria mais um caso onde o TypeScript ganha destaque.

4. EXEMPLOS DE CÓDIGOS

1.1. Go: Link para acessar o código completo

Como já dito anteriormente, na Go, as instruções de controle possuem semântica simples e que se assemelham bastante com a linguagem C. Comandos como *if/else, switch/case e for* podem ser trabalhadas nesta linguagem. Segue o exemplo de uma aplicação de calculadora desenvolvida em Go:

```
15          for option != 3 {
16                  operation := 0
17                  menuPrincipal()
18                  fmt.Scan(&option)
19
20                  if option == 1 {
21                          menuCalcSimples()
22                          fmt.Scan(&operation)
23                          switch operation {
24                          case 1:
25                                  informarDados()
26                                  fmt.Printf("o Resultado da soma de %f e %f é : %2.f \n ", num1, num2, somar(num1, num2))
27                                  break
28                          case 2:
29                                  informarDados()
30                                  fmt.Printf("o Resultado da subtração de %f e %f é : %2.f \n", num1, num2, subtrair(num1, num2))
31                                  break
32                          case 3:
33                                  informarDados()
34                                  fmt.Printf("o Resultado da subtração de %f e %f é : %2.f \n", num1, num2, multiplicar(num1, num2))
35                                  break
36                          case 4:
37                                  informarDados()
38                                  fmt.Printf("o Resultado da divisão de %f e %f é : %f \n", num1, num2, dividir(num1, num2))
39                                  break
40                          case 5:
41                                  fmt.Println("Escolheu sair bye!!!")
42                                  break
43                          default:
44                                  fmt.Println("opção invalida !!! Tente novamente")
45                                  break
```

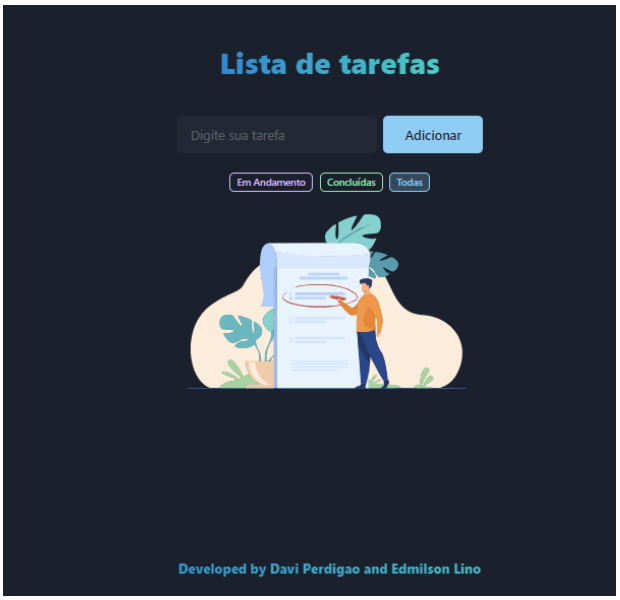
Fonte: Autoria Própria.

Durante a construção da Go, uma das premissas norteadoras foi proporcionar uma sintaxe de fácil compreensão. Um exemplo disso é esse trecho de código, onde fica bem claro a finalidade de cada função, bem como o entendimento de sua sintaxe:

```
132 func calcularRaizQuadrada() float32 {
133         resultado := (num1 / 1.0 / 2.0)
134         return resultado
135 }
136
137 func optPotencia() {
138         fmt.Print("deseja calcular a potência de :")
139         fmt.Scan(&num1)
140         fmt.Print("\nelevado á :")
141         fmt.Scan(&num2)
142 }
143
144 func optRaizQuadrada() {
145         fmt.Print("Calcular a raiz Quadrada de :")
146         fmt.Scanf("%f", &num1)
147 }
```

1.2. TypeScript: [Link para acessar o projeto completo]

O projeto desenvolvido para exemplificar o uso do TypeScript foi uma Lista de Tarefas, utilizando a tecnologia React:



Fonte: Autoria Própria.

TypeScript proporciona fácil interpretação sobre códigos, graças a sua sintaxe. Além do mais, os tipos podem ser adicionados incrementalmente a uma aplicação JavaScript existente.

```
    if (content && !statusInput) {
        setStatusInput(true);
    }

    return (
        <form onSubmit={handleSubmit}>
            <HStack mt='4' mb='4'>
                <Input
                    h='46'
                    borderColor={!statusInput ? 'red.300' : 'transparent'}
                    variant='filled'
                    placeholder='Digite sua tarefa'
                    value={content}
                    onChange={(e) => setContent(e.target.value)}
                />
                <Button
                colorScheme='blue'
                px='8'
                pl='10'
                pr='10'
                h='46'
                type='submit'>Adicionar</Button>
            </HStack>
        </form>
    );
}
```

Fonte: Autoria Própria.

5. CONCLUSÃO

1.1. Go:

Levando em consideração que esta linguagem foi criada há pouco mais de 10 anos, Go já ocupou o seu espaço no mercado de trabalho e na comunidade de Software Livre, sendo escolha de inúmeros projetos *open source* e *cloud native* que estão sendo utilizados por milhares de usuários e empresas no mercado de tecnologia atual.

Sua simplicidade, além do suporte nativo de multiprocessamento, é um dos motivos pelos quais Go ganhou o seu espaço no coração dos desenvolvedores, seguindo um dos objetivos principais da linguagem: tornar o desenvolvimento de softwares modernos mais simples, rápido e produtivo.

1.2. TypeScript:

Como vimos, o TypeScript é uma linguagem de programação de código aberto criada recentemente para aprimorar e aperfeiçoar o JavaScript. O intuito tem dado certo e, por isso, sua popularidade tem crescido a cada ano, permitindo que ela ocupe importante espaço na rotina de trabalho dos programadores.

Portanto, quem deseja ampliar suas oportunidades profissionais, bem como melhorar e otimizar suas atividades, sobretudo aqueles que já utilizam o JavaScript, o TypeScript pode ser uma boa solução.

6. REFERÊNCIA BIBLIOGRÁFICA

- ALURA. **Go - a linguagem do Google**. Disponível em: <https://www.alura.com.br/conteudo/golang>. Acesso em: 29 de agosto de 2022.

- ANNEBÄCK, Joakim; STJERNBERG, Johan. **A comparison between Go and C++**. Disponível em: <https://www.csc.kth.se/utbildning/kth/kurser/DD143X/dkand11/Group9Alexander/Joakim_Anneback_Johan_Stjernberg.finalreport.2.pdf>. Acesso em: 29 de agosto de 2022.

- CAMPOMORI, Cleber. **Precisamos falar sobre TypeScript**. Disponível em: <https://www.treinaweb.com.br/blog/precisamos-falar-sobre-o-typescript>. Acesso em: 30 de agosto de 2022.

- EDSON. **Introdução ao TypeScript**. Disponível em: <https://www.devmedia.com.br/introducao-ao-typescript/36729/>. Acesso em: 30 de agosto de 2022.

- GOULARTE, Suelen. **Golang - a Linguagem do Google.** Disponível em: <https://www.ime.usp.br/~gold/cursos/2015/MAC5742/reports/GoLang.pdf>. Acesso em: 30 de agosto de 2022.

- Redação Vulpi. **Do JavaScript ao TypeScript, Why?**. Disponível em: <https://blog.vulpi.com.br/javascript-typescript/>. Acesso em: 28 de agosto de 2022.

- ROSA, Daniel. **Go explicada e exemplos de código**. Disponível em: <https://www.freecodecamp.org/portuguese/news/genericos-em-go-explicados-com-exemplos-de-codigo/>. Acesso em: 29 de agosto de 2022.

- SEBESTA, Robert. **Conceitos de Linguagem de Programação**. Ed.Bookman. Publicado em: 17 de janeiro de 2011.

- SESTOFT, Peter. **Programming Language Concepts (Undergraduate Topics in Computer Science).** Ed.Springer. Publicado em: 31 de agosto de 2017 (2ª edição.

- MORAES, Murilo. **Go Search**. Disponível em: <https://github.com/kliff-k/go-search>. Acesso em: 26 de agosto de 2022.