O preço da simplicidade

Por que Go é uma ameaça ao status quo corporativo?

Alex Rios

GopherCon Latam 2025

Capitulo 1 - `Era uma vez...`

Mais um dia fazendo o mesmo

Fim da historia

No flamewars

Capitulo 2 - `Simplicidade` ?

Simplicidade: `Nós a reconhecemos quando a vemos` **- mas o que é, exatamente?**

> "A simplicidade é útil porque torna as coisas previsíveis e confiáveis."
>
> - George Whitesides

"Coisas simples tendem a ser baratas e fáceis de (re)produzir."

- George Whitesides

"Coisas simples podem ser usadas como building blocks para criar sistemas mais complexos."

- George Whitesides

"Durante o design, é importante encontrar o equilíbrio certo entre simplicidade e funcionalidade."

- George Whitesides

"Quando um sistema é construído com base nestes princípios, ele se torna acessível para iniciantes"

- George Whitesides

"Simplicidade envolve criar componentes que são confiáveis, fáceis de entender e que podem ser combinados ou empilhados para formar sistemas maiores e mais complexos de maneira previsível."

- George Whitesides


GO

Ecossistema

> Confiavel, Acessivel, Empilhavel

- fast builds
- fast compilation
- single binary
- package manager (que funciona)
- tooling
- libraries ecosystem
- test library
- networking
- synchronization primitives
- templating

Mas a linguagem é simples?

Sim!

e não!

Go é, na verdade, complexa, mas passa a sensação de simples.

- Os elementos que interagem se encaixam perfeitamente, sem surpresas.
- Sintaxe simples, documentação clara.
- Projetada para ser fácil de aprender.

You know when you see it.

Nenhuma referência a High Order Function na spec

Simplicidade (da linguagem) é a arte de ocultar a complexidade.

`Simples` vs `Complicado` em linguagens

Quando é simples

- É mais fácil de entender.
- É mais fácil de trabalhar.
- Se quebrar, é mais fácil de corrigir.

Quando é complicada

Você precisa entender mais coisas para:

- ler
- depurar
- alterar/corrgir

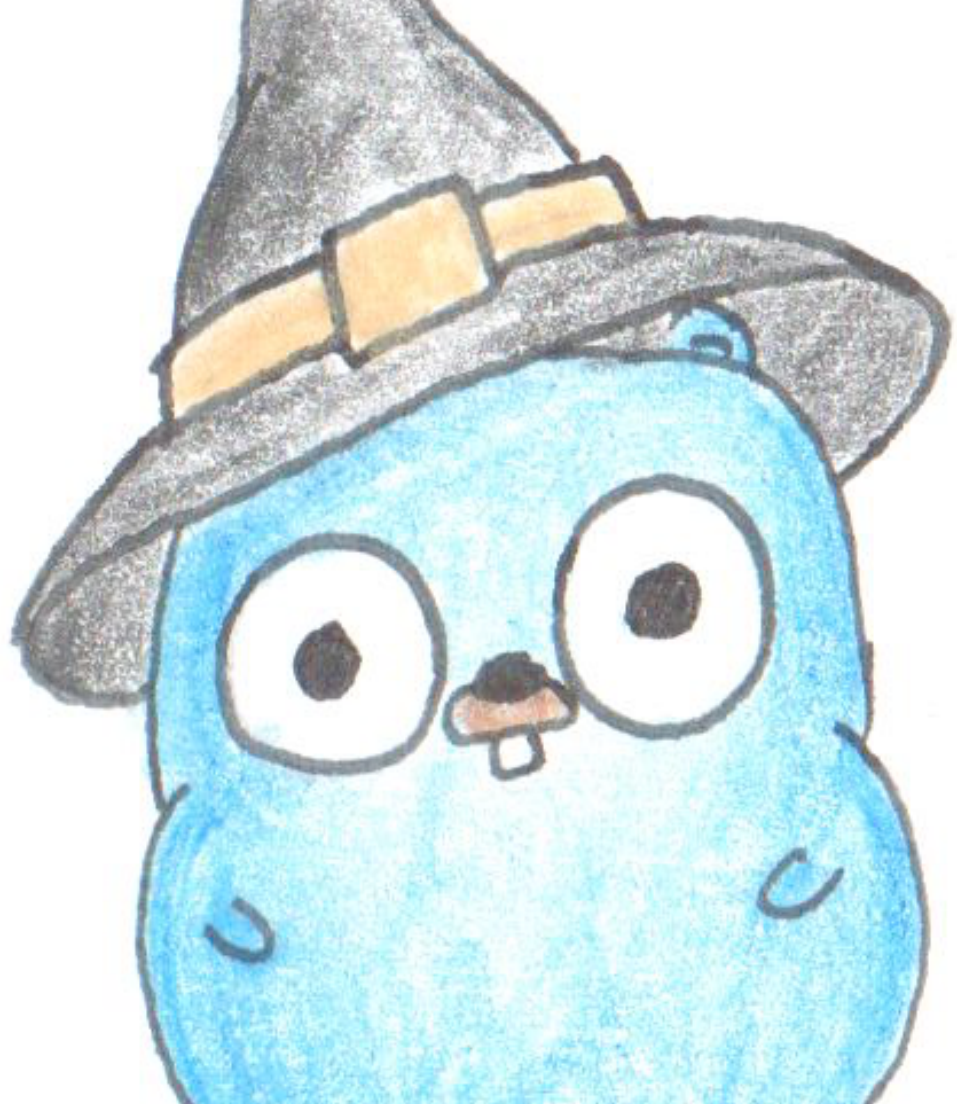
Mais divertido de escrever ou menos trabalho para manter?

Go foi projetado para ajudar a escrever grandes programas, desenvolvidos e mantidos por grandes equipes com alguns padroes em mente:

- Abstracao minima
- Legibilidade clara
- Codigo facil de manter

Cleverness ≠ Idiomatic

Quanto mais "esperto" você tenta ser, mais se desvia do estilo idiomático de Go.



Go então é uma linguagem cool?

Go é mais **enterprise** do que você imagina

Capitulo 3 - `Enterprise`

(Pensei que Go era Hipster)

Go foi projetado para ajudar a escrever grandes programas, desenvolvidos e mantidos por grandes equipes

oogle-grande

Intenção

A alegação do Go é que minimizar o esforço do programador é uma consideração mais importante.

Tab or spaces?

Chaves no final da linha ou na próxima?

GO FMT yourself!

`Enterprise?`

Tamanho e escala do escopo e processos.

Pausa para a realidade triste do Enterprise



Realidade 1

As escolhas e conhecimento técnico não são o aspecto mais importante.

"Coloca mais um pod no kubernetes"

Realidade 2

Especialistas de domínio, analistas, processos e stakeholders, às vezes, não entendem da parte técnica.

"Mas não é só ..."

Realidade 3

As pessoas técnicas muitas vezes não conhecem o domínio de negócios.

> "Quem tem que saber isso é o time de produto"

A importância da elegância, corretude e eficiência tem diminuído `cada vez mais`.

Incentivos escassos para refactoring e manutenção

O software se torna complexo e grande por várias (más) razões:

- O código permanece em diferentes níveis de qualidade
- Múltiplas equipes trabalhando em paralelo acabam produzindo código redundante

O domínio muda com o tempo

- O domínio muda com o tempo, inevitavelmente invalidando suposições anteriores e causando falhas nas abstrações.
- Quanto mais sofisticada for a abstração, maior será o risco de ela se tornar um problema quando o negócio fizer uma solicitação de mudança significativa.

Jrs vão aprender o status quo

(Certified Copy-and-Paste Engineer)

Gestão não estão preparadas para tomar decisões técnicas por falta de experiência

Naturalmente os torna avessos ao risco, levando-os a imitar principalmente o que outros players bem-sucedidos fazem no mercado ou, mais precisamente, o que os analistas afirmam que esses players fazem.

Go `prospera` mantendo a complexidade sob controle à medida que os sistemas escalam.

Jrs são mais produtivos

- Precisa de muito pouca base teorica para programar em Go.
- Não precisa saber OOP nem FP.
- Facilidade no onboarding é vital.

> Acessibilidade

Mid level engs tem dificuldade em introduzir abstrações frágeis

(que com certeza nos morderão no futuro)

Go foi `intencionalmente` criado para o cenário enterprise e nem tenta esconder.

"The key point here is our programmers are Googlers, they're not researchers. They're typically, fairly young, fresh out of school, probably learned Java, maybe learned C or C++, probably learned Python. They're not capable of understanding a brilliant language but we want to use them to build good software. So, the language that we give them has to be easy for them to understand and easy to adopt."

- Rob Pike

Uau 😮! Onde eu assino?

Capitulo 4 - `Problemas`

O contraste corporativo

ABC players distribution.

- A: Top Performer
- B: Reliable
- C: Underperformer

Consumers vs. Producers

- A: Producer
- B: Consumer
- C: Hater

Bingo do B&C player

"Dude! Where is my OO?"

"Go limita minha criatividade"

"Tem muitas opções! É difícil escolher o que usar. A comunidade não se decide."

"Idiomatic? No way Jose!"

(Me traz uma caneca cheio de SOLID e DDD)

"Tem muita regras boba!"

- Não pode ter variaveis ou imports não utilizados.
- Não pode ter arquivos de pacotes diferentes no mesmo diretorio.
- Não pode ter imports circulares.

"Para que complicar as coisas. Não precisa reinventar a roda!"

"A gente resolve os problemas dos clientes, em vez de atender a caprichos de engenharia"

"Typescript tem um type system incrivel! Não tem porque usar algo `tão atrasado`."

Welcome Anders!

No, you can't just use Go for Typescript compiler because

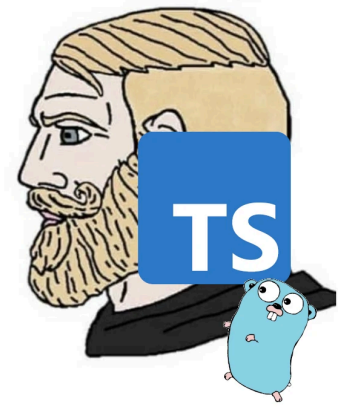
Rust has been Stack Overflow's most loved language for four years in a row, indicating that many of those who have had the opportunity to use Rust have fallen in love with it. However, the roughly 97% of survey respondents who haven't used Rust may wonder, "What's the deal with Rust?"

The short answer is that Rust solves pain points present in many other languages, providing a solid step forward with a limited number of downsides.

I'll show a sample of what Rust offers to users of other programming languages and what the current ecosystem looks like. It's not all roses in Rust-land, so I talk about the downsides, too.
Coming from dynamically-typed languages

The arguments between programmers who prefer dynamic versus static type systems are likely to endure for decades more, but it's hard to argue about the benefits of static types. You only need to look at the rise of languages like Typescript or features like Python's type hints as people have become frustrated with the current state of dynamic typing in today's larger codebases. Statically-typed languages allow for compiler-checked constraints on the data and its behavior, alleviating cognitive overhead and misunderstandings.

This isn't to say that all static type systems are equivalent. Many statically-typed languages have a large asterisk next to them: they allow for the concept of NULL. This means any value may be what it says or nothing, effectively creating a second possible type for every type. Like Haskell and some other modern programming languages, Rust encodes this possibility using an optional type, and the compiler requires you to handle the None case. This prevents occurrences of the dreaded TypeError: Cannot read property 'foo' of null runtime error (or language equivalent), instead promoting it to a compile-time error you can resolve before a user ever sees it. Here's an example of a function to greet someone whether or not we know their name. If we had forgotten the None case in the match or tried to use name as if it was an always-present String value, the compiler would complain.



Productivity and delivery time are crucial compared to a few miliseconds and minor errors, besides Go's mascot is blue, just like Typescript's

TLDR: "This is for nerds! I deliver value! Language is just a tool."

Capitulo 5 - A ameaça

Produtividade não é algo ruim.

mas a falta de pensamento critico é definitivamente ruim.
(Algumas pesoas só querem entregar solução e isso é OK.)

Uma coisa que é o Enterprise entende com um absurdo na mentalidade de times que usam Go:

🤭 `Por vezes tem que programar!` 🤭

You must be intentional

(And it’s not optional)

No `rinse and repeat` anymore

(de uma hora para outra não da mais só para repetir)

The choice has a `PRICE`

`Go` was made to solve `Google`'s problem
(😜not yours😜), so...

STOP FAANG COSPLAY

DANGER ZONE

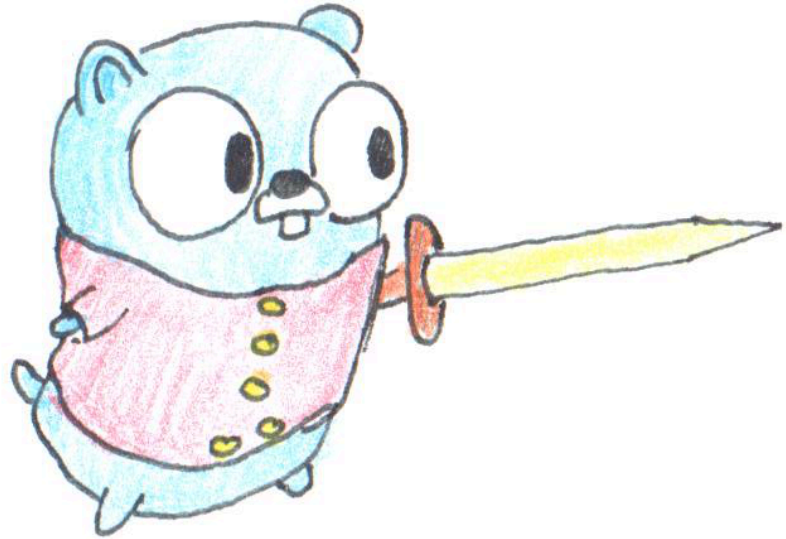
The system loathes thinking.

Go turns out has the potential to be `subversive`.

This single choice COLLAPSES the status quo

Você introduziu na empresa uma ferramenta que leva as pessoas a pensarem/terem que discutir por não ter todas as respostas prontas

Nesse processo, elas podem acabar copiando código alheio sem compreender, podem acabar traduzindo para go as libs das linguagens que conhecem, etc ...



Capitulo 5 - `Finale`

Are we doomed? Why use Go? Is the change worth it?

(Well...)

It’s a platform language, not a product language.

Passado & Prente

What if ...

Go have the language

Não é só pegar todas as boas ideais e simplicidade e colocar tudo em uma linguagem?

One language to rule them all?

"general purpose, functional, ..."

Se todas as linguagens convergissem teríamos apenas uma forma de pensar.

Essa visão é monolítica e idealista.

Queremos diversidade de pensamento

Quanto menos diferente, menos interessante.

Diversidade de pensamento

How Languages Shapes Thought, 2011.

A inteligência humana é definida por sua adaptabilidade, permitindo-nos reformular nossa compreensão do mundo para se adequar a objetivos e ambientes em constante evolução. Essa flexibilidade levou à vasta diversidade de línguas, cada uma oferecendo uma ferramenta cognitiva única moldada pela história cultural. As línguas influenciam como percebemos e categorizamos o mundo, e estudá-las ajuda a desvendar como criamos conhecimento e construímos a realidade, lançando luz sobre o que nos torna exclusivamente humanos.

People create things, not companies

(mainly the brave ones)

"You cannot carry out fundamental change without a certain amount of madness. In this case, it comes from nonconformity, the courage to turn your back on the old formulas, the courage to invent the future"

- Thomas Sankara

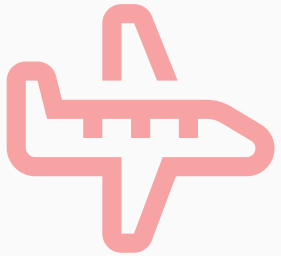
Adicionar features em Go não faz com que ela se torne melhor, apenas maior.

Features adicionam complexidade, prejudicando a legibilidade.

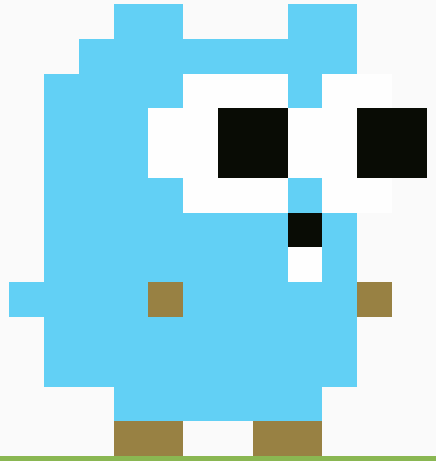
Não queremos apenas uma ferramenta.

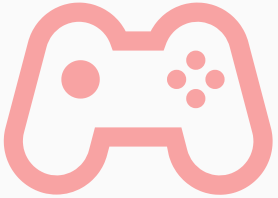
Queremos um conjunto de ferramentas simples, onde cada uma delas sendo a melhor em uma tarefa específica.

Ou simplesmente, Unix Philosophy.



API





Go é uma linguagem de propósito específico

Decida você se isso é bom ou ruim

Go é repelente de:

- FP bro (Category Theory, Monads, Cade meu map-filter?)
- "Só reescrever em Rust"
- "Não é rápido o suficiente"
- "Código é arte"
- Os adventistas do Clean Code, DDD, SOLID e etc.

Para esse recorte, o software não é apenas sobre fazer o trabalho, mas sobre fazê-lo de uma determinada maneira.

Complexidade não é corretude.

A prática é o critério da verdade.

Assim como ocorreu com JS e Python, Go cresce a cada dia e é adotado por mais e mais empresas devido sua materialidade, simplicidade e aversão ao idealismo.

You know you've achieved perfection in design, `not when you have nothing more to add, but when you have nothing more to take away.

-Saint-Exupery

Obrigado GopherCon!

Credits

Cast

(In order of appearance)

Alex Rios	Keynote Speaker Rob Pike-ish
Evento	GopherCon Latam 2025
site	alexrios.me
linkedin	alexrios.me
Slides	Slidev Unocss Vuejs