SEMINÁRIO RUST

Conjunto de recomendações, esboços e linhas de pensamento sobre o que falar em cada slide.

Introdução

Apresentar todos os membros do grupo

Sumário

Apresentar o Sumário

Histórico

Tópico: Contexto da criação de Rust

- Rust: linguagem de programação **segura**, **concorrente** e **de alto desempenho**.
- Criador: Graydon Hoare.
- Motivação: resolver **limitações e problemas em outras linguagens**.
- Objetivos: **segurança**, **evitar erros comuns**.

Linha do tempo

Pensando em **performance e segurança**, Graydon Hoare, um ex-funcionário da Mozilla criou o Rust em 2006.
Hoare estava trabalhando na Mozilla Research na época, mas devido a algumas **frustrações** com as **limitações de C++**, ele achava que era **muito propenso a erros e difícil de raciocinar**, ele queria desenvolver uma **linguagem que fosse rápida e eficiente como C++**, mas que também fosse **segura e confiável**.
Ele passou os próximos anos trabalhando nessa linguagem em seu **tempo livre**, até que em **2009 a Mozilla percebeu o potencial do projeto e começou a apoiar o seu desenvolvimento**. Com isso, o **desenvolvimento da linguagem acelerou** e em **2010, foi lançada a versão alfa de Rust**.
Em **2013, Gradon saiu do projeto por conta de um burnout**. Ele mesmo declarou que a energia dele simplesmente acabou para ser o líder técnico do projeto. Essa situação terminou inclusive com seu casamento.

Ele deu um tempo, saiu da mozilla e foi trabalhar com outras coisas.
Desde 2015, o **desenvolvimento** do Rust tem se **concentrado na estabilidade e ampliação das ferramentas**. O compilador Rust foi melhorado, e uma série de novas bibliotecas foram criadas.
E em 2016, Hoare foi trabalhar na Apple, ajudando no desenvolvimento do swift, mas dessa vez sem uma posição de liderança.

Linguagens relacionadas

Linguagem c-like, tem como grande **inspiração**(e ao mesmo tempo ódio mortal) **o C++ e o C**.

Rust e Go compartilham o objetivo de fornecer abstrações de alto nível com desempenho adequado.

O criador do rust posteriormente após um hiato devido ao burnout no seu projeto de rust foi trabalhar na apple **ajudar no desenvolvimento do swift**

Características marcantes

Apresentar os tópicos das características e **falar sobre como a linguagem é compilada e não interpretada**

Tipagem estática inferida

O **compilador pode inferir que o tipo de y é i32**, mesmo que o tipo **não** seja **explicitamente especificado**. Isso pode tornar o **código** do Rust **mais conciso e fácil de ler**.

A **tipagem estática inferida nem sempre é possível**. Por exemplo, em um código que se define uma variável chamada x e atribui a ela o valor "Olá, mundo!". **O compilador não pode inferir o tipo de x porque o valor "Olá, mundo!" pode ser de qualquer tipo**. Nesse caso, o tipo de x deve ser explicitamente especificado como String.

O fato de **ser fortemente tipada ajuda a evitar erros** em tempo de execução, como tentar adicionar uma cadeia de caracteres a um inteiro.

De forma **combinada à tipagem forte, o ownership and borrowing** (propriedade e empréstimo), **ajudam a evitar vazamentos de memória e outros erros** causados pelo gerenciamento de memória incorreto.

Esses fatores ajudam a **tornar Rust em uma linguagem muito segura**, que foi uma das **principais motivações para sua criação**.

Programação concorrente e paralela

Rust possui um **modelo de concorrência seguro e eficiente**, permitindo que os desenvolvedores **escrevam código concorrente sem se preocupar com problemas de race conditions ou deadlocks**. Como exemplo, ele suporta a criação de **threads**.

Programação funcional

Rust suporta alguns conceitos funcionais, como **imutabilidade por padrão e funções de primeira classe**. Esses recursos permitem **escrever código mais expressivo e modular**.

Programação procedural

Rust também suporta o estilo de programação procedural, onde as instruções são executadas sequencialmente. Ele **permite a definição de funções, estruturas de controle de fluxo (if, while, for) e outros elementos comuns da programação procedural**.

Programação genérica

Rust possui um sistema de tipos genéricos, o que significa que os desenvolvedores podem escrever **código que funcione com diferentes tipos de dados**. Isso permite a **reutilização de código e a criação de estruturas de dados e algoritmos flexíveis e abstratos**.

Programação orientada a objetos

FALAR SOBRE O CAPITULO 17 DO LIVRO QUE FALA:

Não existe um consenso na comunidade de programação sobre quais recursos uma linguagem precisa para ser considerada orientada a objetos. Indiscutivelmente, linguagens POO compartilham certas características comuns, nome para objetos, encapsulamento e herença.

Embora Rust não seja uma linguagem orientada a objetos tradicional, ela **oferece recursos semelhantes**, como **structs** e **traits**, que **podem ser usados para implementar abstração de objetos e polimorfismo**.

Não usa coletor de lixo

capítulo 4 do livro

https://github.com/rust-br/rust-book-pt-br/blob/master/src/ch04-01-what-is-ownership.md

nesse slide eu tava pensando em **introduzir as formas que as outras linguagens podem lidar com a memória** e no próximo slide a gente explica ownership and borrowing detalhado e falando suas qualidades

---

Recapitulando conceitos

Apresentar os conceitos de pilha e heap de forma clara e rápida

---

Ownership

Capítulo 4.01 do livro

https://github.com/rust-br/rust-book-pt-br/blob/master/src/ch04-01-what-is-ownership.md

---

Borrowing

Capítulo 4.02 do livro

https://github.com/rust-br/rust-book-pt-br/blob/master/src/ch04-02-references-and-borrowing.md

---

Slices

capítulo 4.03 do livro

https://github.com/rust-br/rust-book-pt-br/blob/master/src/ch04-03-slices.md

---

Aplicações

Falar de forma geral sobre como **Rust pode ser usado em cenários diferentes devido suas qualidades (segurança e performance)** e **muitas empresas estão adotando ele**.

Aplicações - Linux e Windows

Linus Torvalds, **o criador do Linux**, se trata de uma pessoa **muito detalhista** e que se importa com a **confiabilidade** das coisas sobre todo o resto. E **com razão para isso**, pois o **Linux** é de extrema **importância para a computação**.
Os **altos padrões esperados para o sistema** são tantos que durante toda a existência do Linux, **o kernel continuou majoritariamente em C** e **Rust conseguiu atender a essas expectativas**.
De forma análoga, aconteceu com o Windows.
Tanto a Microsoft quanto a comunidade Linux concordam que **dois terços ou mais das vulnerabilidades de segurança decorrem de problemas de segurança de memória**.

Aplicações - Discord

Uma parte do Discord foi reescrito em GO
Picos de latência a cada 2 minutos causados por pausas para o **garbage collector**
Após se esforçarem para realizar o **máximo de otimizações possíveis em GO**, eles decidiram **migrar para uma linguagem de programação sem o garbage collector**, e escolheram **o Rust**.
Rust se sobressai em performance em todos os testes, mesmo **sendo comparado com a versão mais otimizada de GO** que a equipe do Discord conseguiu produzir representado pelos gráficos.

Tutorial de instalação

Falar de forma breve como instalar e as IDEs e ferranmentas pra programar

---

Considerações finais - Wanderson

Eu tentei aprender Rust seguindo a **documentação**, e achei essa **experiência extremamente fácil e intuitiva**. Tudo é muito bem explicado e os tópicos são introduzidos com passos pequenos, o que tira um pouco desse "medo" em volta de uma linguagem diferente como Rust.

O que eu mais achei legal foi sobre como **os erros são percebidos em tempo de compilação** e sobre como eles **são bem documentados**. Utilizando a extensão rust-analyzer no VS-Code, **quando aparece um erro no código, ele faz uma explicação detalhada sobre o por que do erro, como você possivelmente poderia corrigi-lo** e não apenas uma mensagem de erro genérica como é java, por exemplo.
Me impressionou também o **engajamento** e amor **da comunidade** de Rust sobre a linguagem.

Considerações finais - Gabriel

É muito interessante a capacidade do Rust em somar performance e segurança de maneira impressionante. Fica notório que por essas vantagens claras há uma **tendência muito grande de diversos programas e aplicações** muito usados **serem reescritos nessa linguagem**, conforme mostramos em algumas notícias.
Então pra mim o que ficou de mais fascinante do Rust foi isso, como ela parece ser **poderosa e revolucionária** nesse quesito, **podendo ser uma alternativa** até aos clássicos **C** e **C++** em diversos casos
“C foi a linguagem de programação dos últimos 40 anos, Rust pode ser a dos próximos 40”
