CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE SOROCABA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LEONARDO NAOTO HIROSUE – RA: 0030481911024

ATIVIDADE 2 - LINGUAGENS PARA BACK-END

Prof. Denilce de Almeida Oliveira Veloso

Disciplina: Programação Web

Sorocaba/SP

2º Semestre/2021

SUMÁRIO

INTRODUÇÃO **3**

JAVASCRIPT **4**
Recursos e observações do JavaScript 4
Limitações do JavaScript 5

PYTHON **6**
Recursos e observações do Python 6
Limitações do Python 6

RUBY **7**
Recursos e observações do Ruby 7
Limitações do Ruby 7

PHP **8**
Recursos e observações do PHP 8
Limitações do PHP 9

JAVA **10**
Recursos e observações do Java 10
Limitações do Java 11

C# **12**
Recursos e observações do C# 12
Limitações do C# 13

PERL **14**
Recursos e observações do Perl 14
Limitações do Perl 14

C++ **16**
Recursos e observações do C++ 16
Limitações do C++ 17

KOTLIN **18**
Recursos e observações do Kotlin 18
Limitações do Kotlin 18

SCALA **19**
Recursos e observações do Scala 19
Limitações do Scala 19

CONCLUSÃO **20**

REFERÊNCIAS **21**

1. INTRODUÇÃO

Nesta pesquisa será abordado algumas das principais linguagens de programação para a carreira de desenvolvedor back-end, citando as principais características, recursos, observações e limitações de cada uma delas.

O conteúdo desta pesquisa será utilizado para o desenvolvimento do conhecimento da disciplina de Programação Web na Faculdade de Tecnologia de Sorocaba (FATEC).

2. JAVASCRIPT

JavaScript é uma das linguagens mais populares da última década. Ele permite que os desenvolvedores façam front-end e back-end com a mesma sintaxe, o que reduz significativamente a carga de trabalho.

O Express.JS e o Node.js permitem o gerenciamento e a execução das duas pontas do aplicativo, enquanto as APIs ajudam no desenvolvimento mais rápido e fácil dos aplicativos.

2.1. Recursos e observações do JavaScript

a) Desenvolvimento rápido: Usar a mesma sintaxe para o front-end e o back-end torna o JavaScript muito fácil e rápido de desenvolver.
b) Tecnologia de backend leniente: Embora haja diferentes middlewares suportados e preferidos pelo JavaScript, não há restrições para usá-los. JavaScript permite que os desenvolvedores selecionem e usem qualquer middleware que desejem implementar.
c) Custo benefício: Só porque você pode usar JavaScript para desenvolver o front-end e o back-end dos aplicativos, isso reduz significativamente os custos de desenvolvimento.
d) I/O: O Express.JS e o NodeJS permitem que os programas lidem com as solicitações de E / S e notificações dos aplicativos conectados.
e) Open-Source: JavaScript é uma tecnologia de código aberto, e é por isso que uma enorme comunidade de desenvolvedores está trabalhando nisso. Isso significa que o JavaScript estará em constante aprimoramento e comunidades ativas de desenvolvedores de back-end e front-end por um longo tempo.
f) Menos sobrecarga de script: Alguns dos recursos integrados aumentam a eficiência da codificação e também melhoram o desempenho. Os recursos que fazem isso são DOM, aros e muitos mais. Eles também melhoram a menor sobrecarga de script.

2.2. Limitações do JavaScript

- As funções orientadas a eventos são muito complicadas.
- A maioria dos programadores que usam JavaScript são incapazes de compreender o middleware usado com ele.
- O JavaScript requer MySQL para serviços de banco de dados muito complexos e desatualizados.
- A maioria dos desenvolvedores não prefere a liberdade que a estrutura de back-end JavaScript oferece.

3. PYTHON

Python foi criado em 1991 e emergiu como uma ótima linguagem multiuso desde então. Ele fornece um ambiente de desenvolvimento de back-end muito limpo e fácil. De acordo com uma pesquisa Stack Overflow realizada em 2020, esta foi a 3ª linguagem de programação mais amada pelos desenvolvedores.

3.1. Recursos e observações do Python

a) Fácil de aprender: Python fornece um ambiente de desenvolvimento muito próximo de escrever em inglês. É por isso que é muito fácil de aprender devido à sua alta legibilidade.
b) Muitas bibliotecas estão disponíveis: Existem muitas bibliotecas disponíveis que tornam a necessidade de escrever código muito menor. Você também pode usar a maioria das bibliotecas para aprimorar as tarefas implementadas.
c) IoT: Você pode usar Python para criar objetos Raspberry Pi.
d) Códigos embutidos: A regra de escrita uma vez executada em qualquer lugar (WORA) torna o Python embutido no código-fonte de linguagens como C ++.
e) Custo benefício: Tem muitos desenvolvedores trabalhando, e as bibliotecas são gratuitas, o que torna o desenvolvimento em Python muito econômico.

3.2. Limitações do Python

- A camada de acesso ao banco de dados não é tão desenvolvida quanto outras linguagens.
- Os programas desenvolvidos em Python requerem muitos testes e depuração.
- Python é altamente dependente de bibliotecas e estruturas de terceiros.
- Se a aplicação for interrompida, o tempo de execução fica muito lento.

4. RUBY

Esta linguagem de programação foi desenvolvida em 1990 por um especialista em programação japonês. O melhor dessa linguagem é que ela tem uma sintaxe semelhante a Python e Java. Não só isso, mas permite grandes recursos de automação. É a razão pela qual plataformas como Airbnb e Esty usam isso para fins de automação.

4.1. Recursos e observações do Ruby

a) Produtividade: Por causa do suporte de bibliotecas de terceiros do Ruby, é muito produtivo. Junto com as bibliotecas, Ruby é uma linguagem muito fácil e, na maioria das vezes, não há necessidade de documentação.
b) Meta-programação: Metaprogramação é uma das melhores características do uso de Ruby. Ele permite que os programas obtenham e modifiquem dados de outros programas. A melhor parte é que todo o processo é feito em tempo de execução.
c) Recursos de teste: Existe uma biblioteca em Ruby com recursos de teste. Garante o desenvolvimento de qualidade de todas as aplicações.
d) Confiável e rápido: O método de programação orientado a objetos e os resultados do processamento de consultas tornam Ruby uma plataforma de desenvolvimento muito rápida. Ele pode se desenvolver cerca de 40% mais rápido do que outras tecnologias.

4.2. Limitações do Ruby

- O tempo de execução do Ruby é comparativamente mais lento do que outras linguagens.
- O número de bibliotecas e fontes disponíveis para Ruby não é grande.
- Como Ruby é uma nova linguagem, os desenvolvedores atuais acham um pouco difícil de aprender.
- Os programas codificados em Ruby são difíceis de depurar.

5. PHP

Foi desenvolvido em 1994 e, desde então, tornou-se a melhor linguagem de desenvolvimento do lado do servidor do mundo. De acordo com uma pesquisa da W3Tech, cerca de 79% dos sites do mundo são feitos em PHP.

5.1. Recursos e observações do PHP

a) Fácil de usar: PHP é uma linguagem que você pode usar para muitos propósitos diferentes. É muito fácil de usar e implementar e pode funcionar facilmente com os requisitos modernos, como integração de banco de dados. Também é uma excelente linguagem para iniciantes porque é fácil de usar. O menor tempo necessário para ter um bom controle também motiva os desenvolvedores iniciantes.
b) Open-Source: PHP é uma linguagem de código aberto com várias bibliotecas disponíveis. Isso significa que pode fornecer grandes benefícios se você estiver procurando desenvolver um back-end.
c) Versátil: PHP é uma linguagem de programação que pode ser executada em todos os sistemas operacionais. Além disso, é tão versátil que os programas codificados em PHP também podem ser executados em qualquer navegador da web.
d) Seguro: O PHP tem muitos recursos de segurança integrados que o tornam muito seguro como uma linguagem de desenvolvimento de backend. Esses recursos de segurança permitem mitigar vários threads para melhor segurança do sistema.
e) Automação: Em termos de automação, o PHP é uma ótima linguagem de programação. É porque ele pode automatizar facilmente a autenticação, mapeamento de URL e gerenciamento de sessão.
f) Comunidade: PHP é uma linguagem que possui uma enorme comunidade de desenvolvedores e também é versátil, portanto, a maioria dos desenvolvedores sabe como usá-la. Além disso, o PHP também é muito barato. É o penúltimo lugar na lista das tecnologias mais bem

pagas, de acordo com uma pesquisa de desenvolvedores que a Stack Overflow realizou em 2020.

5.2. Limitações do PHP

Embora fosse uma linguagem muito popular, agora não é tão popular. A razão para isso é que a maioria das pessoas nem se preocupa em adicioná-la às suas habilidades.

Essa linguagem de desenvolvimento de back-end é ótima no desenvolvimento de back-end, mas se a compararmos com as tecnologias de back-end modernas, ela está muito atrás.

Por ser de código aberto, um dos maiores problemas com o PHP é que ele pode ser mal utilizado, e isso pode causar a criação de códigos com bugs e não tão bem otimizados.

6. JAVA

Java foi feito em 1991, mas foi oficialmente publicado em 1995 e, a partir dessa época, emergiu como uma das melhores linguagens de programação do mundo. Ela também é classificada como a segunda melhor linguagem de programação no índice TIOBE de 2021. Além disso, é a melhor opção para construir aplicativos móveis.

6.1. Recursos e observações do Java

a) Escalável: O framework de desenvolvimento de Java permite desenvolver aplicações com opção de escalabilidade. Isso é possível porque permite que o lado do servidor execute várias instâncias ao mesmo tempo. Isso o torna uma excelente tecnologia de desenvolvimento de back-end.

b) Compreensão fácil: A sintaxe usada na programação Java é muito simples e fácil de entender. É porque é mais fácil lembrar as palavras-chave. Isso torna a programação, atualização e depuração muito fáceis para os desenvolvedores.

c) Multi-threading com suporte: Quando falamos sobre o uso de servidores web, Java é a linguagem de programação que pode lidar com diferentes solicitações de thread independentes. No entanto, requer muitos threads de CPU para funcionar também.

d) Bibliotecas de código aberto: Suponha que você esteja procurando o suporte de bibliotecas de código aberto. Então Java é uma das melhores linguagens de programação de backend. É porque ele tem bibliotecas enormes. Eles podem ser usados para adicionar facilmente todos os recursos adicionais do servidor ao seu aplicativo.

e) Possui opções de segurança aprimoradas: Java é uma ótima opção se você está procurando construir um ótimo sistema em termos de segurança. É porque aqui a segurança é muito forte. Um exemplo disso é que a máquina virtual Java primeiro verifica o bytecode para garantir que não haja vírus e, em seguida, o processo continua.

6.2. Limitações do Java

- A linguagem de programação Java é demorada.
- Não há programação de baixo nível feita em Java.
- Não há comandos para coleta de dados de lixo.
- Java é uma linguagem que funciona melhor em sistemas caros ou de ponta. Isso torna o desenvolvimento em Java muito caro.
- Na seção GUI dos aplicativos de desenvolvimento Java, muitas coisas necessárias como ferramentas e objetos de interface da moda estão faltando.

7. C#

C-Sharp é uma das linguagens mais populares para fazer o back-end de um sistema. É por causa de seus recursos incríveis como a automação em servidores Windows. Além disso, é ótimo porque executa códigos muito rápido. Também pode ser usado para desenvolvimento de jogos e criação de aplicativos CLI.

7.1. Recursos e observações do C#

a) Suporta desenvolvimento multiplataforma: Uma das melhores coisas sobre o C-Sharp é que ele suporta o desenvolvimento de plataforma cruzada. Isso significa que os aplicativos desenvolvidos com o C# podem ser executados em diferentes sistemas operacionais. No entanto, para obter mais informações sobre estruturas de plataforma cruzada C#, você pode consultar aquele artigo que discute o desenvolvimento de plataforma cruzada em detalhes.
b) Compatibilidade extensa: C-Sharp desenvolveu suporte a aplicativos para compatibilidade com versões anteriores. Isso torna o C# a melhor opção para os aplicativos que devem ser executados em versões mais antigas de estruturas de programação.
c) Coleta de dados e valores de lixo: Para reduzir os possíveis erros no aplicativo, o C-Sharp possui o recurso de coleta de dados e valores de lixo. Ele coleta todos os dados de lixo que podem atrapalhar os valores durante a execução. Por causa disso, o programa pode fornecer os melhores resultados de forma otimizada.
d) Object-oriented programming: C-Sharp é a linguagem que oferece suporte à programação orientada a objetos. Isso significa que os desenvolvedores podem fazer bom uso das classes e relacionamentos. Isso torna o código excepcionalmente fácil de depurar, entender e até mesmo reutilizar.

7.2. Limitações do C#

Embora seja uma linguagem de alto nível, isso criou um problema que a linguagem não pode interagir diretamente com o hardware.

Não é ótimo em termos de compatibilidade com diferentes sistemas, porque você só pode instalá-lo de um computador Windows.

Além disso, ele só pode ser executado usando a estrutura .Net. Portanto, podemos dizer que essa tecnologia é muito inflexível se comparada às tecnologias de back-end mais recentes.

8. PERL

O Perl foi desenvolvido há cerca de 3 décadas, pois é uma linguagem que ainda tem um desempenho excepcionalmente bom onde é necessário. O interessante do Perl é que ele foi classificado como uma das linguagens mais lucrativas pela pesquisa de um desenvolvedor da Stack Overflow. Perl 5 é a versão que ainda é amplamente utilizada para prototipagem e automação.

8.1. Recursos e observações do Perl

a) Compatível com múltiplas plataformas: Perl é uma linguagem que funciona em muitas plataformas diferentes, como Windows, Unix, Linux e Mac. Isso torna muito fácil para diferentes tipos de desenvolvedores trabalharem.
b) Perl é uma linguagem de código aberto: Como a maioria das linguagens de programação, Perl também é open source. Ele também fornece várias bibliotecas para facilitar os desenvolvedores.
c) Extensibilidade: Perl é uma linguagem que suporta extensibilidade. Isso significa que você pode usar as bibliotecas C e C ++. Além disso, os programas Perl também podem ser integrados aos programas C e C ++.
d) Processamento de texto: Perl ainda mantém a conformidade com o POSIX. Isso significa que as funções de processamento de texto do Perl ainda são excelentes. Ele também mantém os recursos avançados de processamento de chamadas de soquete.

8.2. Limitações do Perl

- O processamento ou programas desenvolvidos na linguagem Perl não são tão bons quanto algumas outras linguagens.
- As bibliotecas disponíveis no Pearl não possuem alguns dos recursos mais necessários.
- O processo de correção de bugs em Perl não é tão fácil e é muito desafiador.

- Se seu código for extenso, não será apenas um desafio manipulá-lo, mas também será muito desafiador para o sistema lidar com o código.
- Escalabilidade não é uma opção disponível quando você usa Perl.
- A velocidade do aplicativo desenvolvido em Perl não é tão grande.
- É uma linguagem cara porque é uma tecnologia mais antiga e não há muitos desenvolvedores disponíveis. Além disso, essa linguagem leva muito tempo para ser aprendida e dominada.
- As bibliotecas disponíveis em Perl não são gratuitas, pois você tem que pagar taxas de cópia por elas.

9. C++

Podemos chamar isso de versão avançada da linguagem C porque é a mesma, mas tem algumas adições incríveis. Uma das maiores adições ao C ++ é que ele oferece suporte à programação orientada a objetos.

É uma das linguagens de programação mais antigas e, por ser uma linguagem de baixo nível, pode interagir diretamente com o hardware. Faz com que tenha um desempenho muito bom. A melhor parte é que você pode usá-lo para desenvolver os sistemas mais recentes sem faltar ou faltar os recursos necessários.

9.1. Recursos e observações do C++

a) Portabilidade: Um dos melhores recursos de usar C ++ é que ele é portátil. A linguagem pode funcionar facilmente em qualquer sistema operacional de plataforma diferente e compiladores. Isso torna o C ++ ótimo para realizar diferentes tarefas.
b) Programação orientada a objetos: C ++ oferece suporte à programação orientada a objetos. Isso torna muito fácil criar diferentes tipos de programas e aplicativos. Também torna muito fácil para os desenvolvedores atenderem a todos os requisitos do aplicativo usando sua arquitetura de programação estruturada.
c) Linguagem de baixo nível: C ++ é uma linguagem de baixo nível, o que significa que não requer muitas traduções. Desta forma, ele pode interagir facilmente com a maioria do hardware. É a razão pela qual ele é usado para codificar a maioria dos recursos de hardware do sistema.
d) O gerenciamento de memória é ótimo: O gerenciamento de memória feito em C ++ é ótimo, o que torna os aplicativos codificados em C ++ muito eficientes em termos de gerenciamento de memória. Os desenvolvedores também podem controlar como seu programa codificado acessa a memória necessária, o que o torna ainda mais eficiente em termos de memória se codificado corretamente.

9.2. Limitações do C++

Não há trabalho feito na coleta automática de dados ou valores de lixo. A segurança do hardware não é grande ao usar C ++. Por isso, se alguém quiser, pode manipular o código para interagir com o hardware do sistema.

10. KOTLIN

Kotlin é a linguagem de desenvolvimento de back-end que pode ser encontrada no Android Studio como uma opção. Embora tenha sido criado em 2011, agora está assumindo o Java porque agora a maioria dos aplicativos está sendo desenvolvida com ele. Outra coisa incrível é que agora ele oferece suporte ao desenvolvimento entre estruturas, tornando-o mais popular e amplamente aceito.

10.1. Recursos e observações do Kotlin

a) Menos código é necessário para o desenvolvimento: Kotlin fornece um ambiente de codificação muito conciso, onde muito pouco código pode fazer a maioria das tarefas. Leva muito pouco tempo para realizar as tarefas.
b) Compatibilidade Java: O código não requer margens quando usado de Java para Kotlin ou de Kotlin para Java. Isso faz com que o código funcione perfeitamente, e os desenvolvedores podem usar perfeitamente os códigos de diferentes linguagens para adicionar mais funcionalidade a seus aplicativos.
c) Fácil de gerenciar: O código do Kotlin é muito fácil de ler, editar e entender. Isso torna muito rápido e fácil para os desenvolvedores gerenciar e atualizar seu código.

10.2. Limitações do Kotlin

- Não há muitos tipos de variáveis primitivas disponíveis no Kotlin. Isso torna o gerenciamento de variáveis em todo o código um tanto difícil. Também pode causar problemas de saída às vezes.
- As funções aqui também têm tipos primitivos como linguagens tradicionais. Isso torna mais difícil para os desenvolvedores que estão acostumados com outras linguagens.
- A compilação para aplicativos Android é mais lenta se a compararmos com a do Java.

11. SCALA

Scala é conhecido por sua propriedade de combinar programação funcional e programação orientada a objetos. Faz com que ele forneça um código muito conciso. A JVM o capacita, e essa é a razão de sua ampla aceitação. Além disso, o suporte a tipos estáticos dessa linguagem mantém o desenvolvedor livre do incômodo de tipos estáticos.

11.1. Recursos e observações do Scala

a) Esquema de codificação fácil: Uma grande coisa sobre Scala é que ele combina o conceito de programação funcional e programação orientada a objetos. Isso significa que você não precisa lidar com os dois separadamente. Isso torna o processo de programação muito fácil.
b) Compatibilidade com o Java: Como a escala funciona com JVM, ela é compatível com a linguagem de programação Java. Isso significa que, quer o código seja escrito em qualquer linguagem, ele fornecerá o mesmo resultado. Isso o torna uma excelente linguagem de desenvolvimento de backend para desenvolvedores Java.
c) Natureza concisa: O código Scala tem uma natureza de codificação muito concisa, o que o torna muito fácil de manter. Isso torna o trabalho de desenvolvimento e atualização muito simples.

11.2. Limitações do Scala

A abordagem de codificação do Scala é um pouco não convencional em comparação com as linguagens tradicionais como Java. Isso significa que você pode demorar mais do que o normal para ter um bom domínio desse idioma.

Aqui, a programação orientada a objetos e a programação funcional são combinadas. Como a maioria dos conceitos principais dos desenvolvedores é baseada em C ++, esta combinação de programação orientada a objetos e programação funcional pode ser um pouco difícil para eles.

12. CONCLUSÃO

O back-end no mundo da programação está em constante evolução e aprimoramento para atender a necessidade do momento atual do mercado e tecnologias disponíveis.

Não existe a melhor linguagem de programação ou uma que se destaque mais que a outra, cada uma foi desenvolvida para atender a um tipo de cenário. Conhecendo todas elas, você identificará mais indicada para o seu projeto, para construir o melhor back-end da sua aplicação em termos de desempenho, eficiência e funcionalidade.

REFERÊNCIAS

BACK 4 APP. As dez melhores linguagens para programar um backend. Disponível em: <https://blog.back4app.com/pt/as-dez-melhores-linguagens-para-programar-um-backend/>. Acesso em: 04 de agosto de 2021.