Desenvolvimento de Sistemas

O tecnólogo com esta formação desenvolve, analisa, projeta, implementa e atualiza sistemas de informação. Tem noções de gerenciamento, mas sua especialidade é a criação de sistemas informatizados: programação de computadores e desenvolvimento de softwares para ampliar a capacidade dos recursos do equipamento. Ele implanta e desenvolve banco de dados. Conhece a estrutura física dos equipamentos e seus periféricos, e precisa se manter muito atualizado sobre aplicativos, ambientes operacionais e linguagens de programação. Além disso, precisa ter boa noção dos negócios da companhia para a qual trabalha.

Os cursos nesta área se dividem, fundamentalmente, em dois tipos: os mais generalistas e os específicos para alguma área de atuação, como banco de dados. Seja como for, o currículo de todos inclui muitas aulas de disciplinas de Ciências Exatas, como cálculo e linguagens de computação. Estuda também engenharia de software e banco de dados.

Em atividades práticas, o estudante faz análise e programação de sistemas e administra redes de computadores. Fazem parte da grade curricular matérias como aplicações em comércio eletrônico e internet.

Introdução: Tem como objetivo esclarecer aos desenvolvedores o funcionamento das metodologias no desenvolvimento de software e o modo como elas evoluíram no decorrer dos anos.

Em meados da primeira guerra mundial tivemos uma evolução significativa no segmento corporativo.

Nesta época o mundo passava por intensas transformações e isto provocou drásticas mudanças no ciclo produtivo das empresas e percebeu-se a necessidade de controlar o seu processo de trabalho.

Baseado nestas transformações houve a necessidade de se aplicar o conceito de dinamização de processos e daí surgiu à necessidade de se administrar grandes volumes de dados em organizações de todas as esferas.

Com a criação dos computadores comerciais após a segunda guerra mundial tivemos um aumento significativo na dinamização da indústria de computadores e, conseqüentemente, o processo de construção de softwares, para que os mesmos automatizassem processos manuais e pudessem avaliar situações complexas que são parte integrante do cotidiano das organizações.

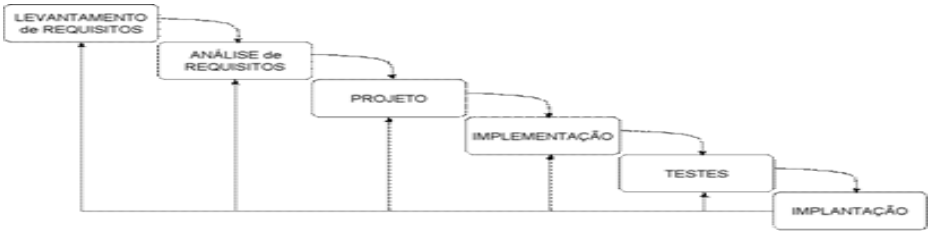
E a partir desse cenário, criou-se modelos de desenvolvimento de softwares que atendessem a determinadas necessidades específicas e ao mesmo tempo pudessem ser utilizados na elaboração softwares sem grandes complexidades.

A seguir são apresentados os modelos de desenvolvimento de softwares.

Modelo Cascata

O Modelo Cascata, também chamado de Clássico ou Linear, caracteriza-se por possuir uma tendência na progressão seqüencial entre uma fase e a seguinte. Eventualmente, pode haver uma retroalimentação de uma fase para a fase anterior, mas de um ponto de vista macro, as fases seguem fundamentalmente de forma seqüencial.

A figura abaixo nos dá uma idéia visual do conceito apresentado acima.



Modelo Iterativo e Incremental

O Modelo de ciclo de vida Iterativo e Incremental foi proposto justamente para ser a resposta aos problemas encontrados no Modelo em Cascata. Um processo de desenvolvimento, segundo essa abordagem, divide o desenvolvimento de um produto de software em ciclos. Em cada ciclo de desenvolvimento, podem ser identificadas as fases de análise, projeto, implementação e testes.

Essa característica contrasta com a abordagem clássica, na qual as fases de análise, projeto, implementação e testes são realizados uma única vez.

No Modelo de ciclo de vida iterativo e incremental, um sistema de software é desenvolvido em vários passos similares (iterativo). Em cada passo, o sistema é estendido com mais funcionalidades (incremental).

Cleanroom é uma metodologia muito utilizada no desenvolvimento de software. É considerada "pesada" pelos padrões da Engenharia de Software, mas muito difundida no desenvolvimento de grandes projetos corporativos.

O processo é baseado no projeto apurado das funções, que são analisadas pelo método de revisão-par com o objetivo de verificar se fazem realmente o que foram especificadas a fazer. Por analogia, podemos comparar esta metodologia com as **salas limpas** na fabricação de semicondutores, que eliminam a necessidade de se limpar wafers de silício pelo fato de que eles nunca começam sujos. O desenvolvimento Cleanroom remove a necessidade de depuração do programa, assegurando que os erros nunca começam introduzidos no sistema.

Programação Extrema (do inglês eXtreme Programming), ou simplesmente **XP**, é uma metodologia ágil para equipes pequenas e médias e que irão desenvolver software com requisitos vagos e em constante mudança. Para isso, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento de software.

Scrum é um método ágil que foi concebido como um estilo de gerenciamento de projetos em empresas de fabricação de automóveis e produtos de consumo, por Takeuchi e Nonaka no artigo "The New New Product Development Game".Eles notaram que projetos usando equipes pequenas e multidisciplinares (cross-functional) produziram os melhores resultados, e associaram estas equipes altamente eficazes à formação Scrum do Rugby (utilizada para reinício do jogo em certos casos).

Jeff Sutherland, John Scumniotales, e Jeff McKenna documentaram, conceberam e implementaram o Scrum, como descrito abaixo, na empresa Easel Corporation em 1993, incorporando estilos de gerenciamento observados por Takeuchi e Nonaka. Em 1995, Ken Schwaber formalizou a definição de Scrum e ajudou a implantá-lo em desenvolvimento de software em todo o mundo.

Por estarmos vivendo em constante transformação neste cenário de desenvolvimento de software precisamos realizar uma análise do que foi vivido no passado para assim aprendermos com os erros e percebemos que podemos sempre inovar.

Vejam o cenário que antes era a realidade dos desenvolvedores de software.

As linguagens de programação eram bastante complexas e, além de não existir o conceito de linguagem visual, envolviam inúmeros cálculos matemáticos e isso exigia que os desenvolvedores possuíssem conhecimentos avançados nas áreas de matemática e física, tornando, assim, uma difícil manipulação delas.

A interação do usuário ao software se dava através do teclado e isto fazia com que o usuário que fosse utilizar o computador passasse muito tempo para aprender a manipular o sistema.

Pelo que se pode perceber que com a evolução dos paradigmas o cenário mudou significativamente como apresentado abaixo.

Com a introdução do mouse nos PCs e o surgimento da interface gráfica surgiu o conceito de linguagens visuais e disso temos atualmente um cenário complexo para o desenvolvimento de sistemas.

Por isso o conhecimento das metodologias acima e de sua aplicação no cenário de desenvolvimento pode muitas vezes economizar ciclos de trabalho e como conseqüência há uma economiza significativa de dinheiro e trabalho.

Não importa se você adotar um ou outra de forma isolada ou integrada o importante que desenvolver soluções vai além de linhas de código depende mais do trabalho das pessoas e de como é tratado cada ciclo deste trabalho.

Sistemas de software têm desempenhado um papel cada vez mais preponderante no dia-a-dia das pessoas, e em muitas situações o funcionamento correto ou incorreto desses sistemas pode ser a diferença entre a vida e a morte. Entretanto, a construção de sistemas é complexa, pois deve lidar com requisitos intransigentes, restrições de integridade e a necessidade de um vasto conhecimento sobre a aplicação para que as interações esperadas entre o software e o ambiente possam ser adequadamente descritos. Quando os requisitos não são totalmente compreendidos, registrados e comunicados para a equipe de desenvolvimento, muito provavelmente, haverá discrepância entre o que o sistema construído faz e o que ele deveria fazer.

Hoje em dia o software assume um duplo papel. Ele é o produto e ao mesmo tempo o veículo para entrega do produto. Como produto ele disponibiliza o potencial de computação presente em computador, ou mais amplamente numa rede de computadores acessível pelo hardware local. Quer resida em um telefone celular, quer opere em um computador de grande porte (Mainframes) o software é transformador de informações – produzindo, gerando, adquirindo, modificando, exibindo, ou transmitindo informação, que pode ser tão simples como um bit ou tão complexa como uma apresentação multimídia. Como veículo usado para a entrega do produto, o software age como base para controle do computador – sistemas operacionais – para a comunicação da informação e para a criação e o controle de outros programas.

Atualmente, o desenvolvimento de software não ocorre como no passado, o programador solitário foi substituído por uma equipe de especialistas com cada um se concentrando numa parte da tecnologia necessária para produzir uma aplicação. No entanto, os mesmos questionamentos feitos ao programador solitário estão sendo feitos nos dias atuais:

· Por que leva tanto tempo para concluir o software?

· Por que os custos de desenvolvimentos são tão altos?

· Por que não podemos achar todos os erros antes da entrega do software aos clientes?

· Por que continuamos a ter dificuldades em avaliar o progresso enquanto o software é desenvolvido?

Há alguns anos tem-se discutido maneiras de contornar a complexidade do software, visto que a cada dia novas áreas de aplicação têm surgido e exigido mais confiabilidade e precisão dos softwares já existentes e dos que ainda virão a ser construídos. Não há consenso sobre qual a melhor prática para o desenvolvimento de sistemas de software, mais existe um esforço em encontrar soluções para reduzir as dificuldades oriundas da própria natureza do software, da sua complexidade, de necessidades de cumprir seus objetivos e da rapidez com que sofre alterações. A Engenharia de software propõe a adoção da disciplina para lidar com essas dificuldades, tentando reduzir ao máximo a influência delas no processo de desenvolvimento de software.

Engenharia de Software

A Engenharia de software é uma disciplina que reúne metodologias, métodos e ferramentas a ser utilizadas, desde a percepção do problema até o momento em que o sistema desenvolvido deixa de ser operacional, visando resolver problemas inerentes ao processo de desenvolvimento e ao produto de software.

O objetivo da Engenharia de software é auxiliar no processo de produção de software, de forma que o processo de origem a produtos de alta qualidade, produzidos mais rapidamente e a um custo cada vez menor. A Engenharia de software segue o conceito de disciplina na produção de software, fundamentado nas metodologias, que por sua vez seguem métodos que utilizam de ferramentas automáticas para englobar as principais atividades do processo de produção.

Metodologia de Desenvolvimento de Software

Já faz alguns anos que o desenvolvimento de software deixou de ser sinônimo apenas de código. Hoje em dia, sabe-se que é necessária a utilização de uma metodologia de trabalho.

Mas o que é necessariamente uma metodologia de software? Entende-se por metodologia, como a maneira – forma – de se utilizar um conjunto coerente e coordenado de métodos para atingir um objetivo, de modo que se evite, tanto quanto possível, a subjetividade na execução do trabalho. Fornecendo um roteiro, um processo dinâmico e interativo para desenvolvimento estruturado de projetos, sistemas ou software, visando à qualidade e produtividade dos projetos.

O dicionário [WEBSTERS, 1998] define metodologia como um conjunto de métodos, regras e postulados empregados por uma disciplina: um procedimento particular ou conjuntos de procedimentos.

É objetivo de uma metodologia definir de forma clara “quem” faz “o que”, “quando”, “como”, e até mesmo “onde”, para todos os que estejam envolvidos diretamente ou não com o desenvolvimento de software. Deve definir também qual o papel dos técnicos, dos usuários, e o da administração da empresa no processo de desenvolvimento. Com isso, evita-se a situação a qual o conhecimento sobre o sistema é de poucos, comumente apelidados, de “os donos do sistema”. Além disso, deve instruir um conjunto de padrões preestabelecidos, de modo a ser evitar a subjetividade na abordagem, a fim de garantir fácil integração entre os sistemas desenvolvidos. Com isso, o uso de uma metodologia possibilita:

· Ao gerente: controlar o projeto de desenvolvimento de software mantendo o rumo do projeto sobre controle para que não haja desvios de planejamentos de custos e prazos, que, se negligenciados ou mal conduzidos, podem por em risco o sucesso do projeto.

· Ao desenvolvedor: obter a base para produzir de maneira eficiente, software de qualidade que satisfaça os requisitos estabelecidos.

Muitas vezes, o uso de uma metodologia é encarado como cerceamento da criatividade dos técnicos, ou como, acréscimo de burocracia, leia-se documentações, por muitos tidos como desnecessário a construção de software. Uma metodologia não deve limitar a criatividade profissional, mas deve ser um instrumento que determine um planejamento sistemático, que harmonize e coordena as áreas envolvidas. O que limita a criatividade não é a metodologia, mas os requisitos de qualidade e produtividade de um projeto.

Como uma metodologia é um conjunto de métodos, convém definir o que é um método e qual o seu objetivo.

Um método é abordagem técnica passo a passo para realizar uma ou mais tarefas indicadas na metodologia. Ou seja, é (são) o(s) procedimento(s) necessário(s) a ser (em) adotado(s) para atingir um objetivo. Já uma técnica, pode ser compreendida como sendo um modo apropriado de se investigar sistematicamente um universo de interesse ou domínio do problema. Para tanto, utiliza-se de uma notação. Como exemplo de técnica, temos: Análise estruturada, Análise Essencial, Projeto Estruturado, Análise Orientada a Objetos.

A escolha de uma metodologia a ser utilizada no desenvolvimento, deve ser realizada com base na natureza do projeto e do produto a ser desenvolvido, dos métodos e ferramentas a serem utilizadas e dos controles e produtos intermediários desejados.

Conclusões

O uso de metodologia, mesmo que ainda não fortemente sedimentada, no desenvolvimento de software é de extrema importância, para que o sistema construído atenda as necessidades dos interessados, com um mínimo de qualidade.

O tecnólogo com esta formação desenvolve, analisa, projeta, implementa e atualiza sistemas de informação. Tem noções de gerenciamento, mas sua especialidade é a criação de sistemas informatizados: programação de computadores e desenvolvimento de softwares para ampliar a capacidade dos recursos do equipamento. Ele implanta e desenvolve banco de dados. Conhece a estrutura física dos equipamentos e seus periféricos, e precisa se manter muito atualizado sobre aplicativos, ambientes operacionais e linguagens de programação. Além disso, precisa ter boa noção dos negócios da companhia para a qual trabalha.

Os cursos nesta área se dividem, fundamentalmente, em dois tipos: os mais generalistas e os específicos para alguma área de atuação, como banco de dados. Seja como for, o currículo de todos inclui

muitas aulas de disciplinas de Ciências Exatas, como cálculo e linguagens de computação. Estuda também engenharia de software e banco de dados.

Em atividades práticas, o estudante faz análise e programação de sistemas e administra redes de computadores. Fazem parte da grade curricular matérias como aplicações em comércio eletrônico e internet. Para obtenção do diploma, algumas instituições exigem estágio e um trabalho de conclusão de curso.

Se você chegou até esse post, é muito provável que você já tenha ouvido falar em linguagens de programação. Elas são diversas, estão em constante processo de evolução e foram criadas para resolver algum problema específico da área computacional.

Sim, existem muitas linguagens de programação por aí! Contudo, quais são as principais? qual é o objetivo e filosofia de cada uma? E a origem de cada uma? Relevância no mercado de trabalho?

Pois é exatamente sobre isso que esse artigo irá tratar. Continue lendo e conheça as **15 principais linguagens de programação** do mercado de TI!

Código Binário x Linguagens de Programação

Inicialmente, os computadores foram criados para realizar cálculos matemáticos de forma muito mais rápida do que seres humanos são capazes. Para que isso fosse possível, era necessário que as devidas instruções matemáticas fossem repassadas às máquinas. Nesse momento, surgiram as linguagens de programação.

Em outras palavras, para que uma instrução seja "entendida" pelo computador, é necessário que sejam escritas em linguagem de máquina, ou seja, **códigos binários** formados por **sequências de 0 e 1**.

Isso quer dizer que para aprender a programar, preciso aprender sobre codificação binária? Bem, esta é uma decisão sua, mas para te tranquilizar, a resposta é NÃO!

Programar diretamente em linguagem de máquina é sim possível, porém, é um processo lento e difícil, praticamente inviável nos dias de hoje devido a complexidade dos sistemas modernos. Sendo assim, para que você não precise codificar de forma binária, existem as linguagens de programação. Estas são próximas as linguagens humanas e, portanto, mais fáceis de serem lidas e compreendidas.

O que são Linguagens de Programação?

Resumindo, linguagens de programação são **padrões de codificação binária**, com sintaxe e semânticas específicas. Desta forma, capazes de criar instruções para máquinas. Graças a esses conjuntos de códigos e recursos, é possível criar programas e sistemas para resolver os mais diversos problemas do cotidiano.

Para Melhor Compreensão!

O humano fala através de um idioma, já o computador entende binário. Então para que ambas as partes consigam se comunicar, é necessário um intermediário: uma linguagem de programação.

Através dela, é possível programar de uma forma que um compilador traduza as instruções para o computador (em binário). De outra forma, caso seja uma linguagem interpretada, as instruções seriam repassadas a um interpretador para a sua execução.

Resumindo, linguagens de programação existem para ser o **canal de comunicação** de um programador com o hardware (máquina).

Tipos de Linguagens de Programação

Existem diversas categorias para classificar linguagens de programação, nesse post, iremos citar as 2 principais:

Paradigma de Programação

Um paradigma de programação, a grosso modo, é a forma utilizada para resolver um problema computacional. Linguagens podem suportar mais de um paradigma (linguagens multiparadigma), este a ser escolhido conforme o problema a ser resolvido. Abaixo, os dois principais paradigmas de programação.

▪ **Paradigma Procedural ou Imperativo**

Conceito de programação que define softwares como uma sequência de comandos para o computador executar. O nome do paradigma, Imperativo, está ligado ao tempo verbal, onde repassamos "ordens" ao computador: faça isso, depois isso e depois aquilo.

▪ **Paradigma Orientado a Objetos**

Paradigma mais popular atualmente, trata-se de um conceito de programação baseado no uso de componentes individuais. Estes são chamados **objetos** e fazem parte da composição do software.

Tentamos resumir. Contudo, **paradigmas de programação** merecem um post específico para o assunto. Pensando nisso, escrevemos o **seguinte artigo** para tratar esse assunto com mais detalhes.

Alto ou Baixo Nível

As linguagens também são classificadas em níveis (alto ou baixo). Existem aquelas em que a sintaxe se aproxima a uma linguagem humana, por isso entram na categoria das **linguagens de alto nível**.

Por outro lado, existem as linguagens que possuem sintaxe e semântica próximas ao código de máquina, portando, classificadas como **linguagens de baixo nível**.

As linguagens de programação mais populares

Antes de mais nada, é importante esclarecermos algumas questões:

▪ As linguagens tratadas aqui não foram selecionadas ao acaso, nem de acordo com a nossa única e exclusiva opinião. Para essa seleção, nos baseamos nos conceituados rankings da **TIOBE, IEEE Spectrum e Redmonk**.

▪ Como os rankings passam por frequentes atualizações (e o próprio mercado de TI também pode nos surpreender), apresentaremos as linguagens de programação que ficaram entre as mais populares nos três rankings.

▪ Para essa lista, foram consideradas apenas linguagens de "programação". Não foram consideradas outros tipos de linguagens da área de tecnologia, como linguagens de marcação (HTML), linguagens de folhas de estilo (CSS), linguagens de consulta estruturada (SQL) e entre outras.

▪ A lista não está ordenada, pois existem diversos critérios de ordenamento, como: popularidade, relevância, empregabilidade e por aí vai. Além disso, as posições estão em constante mudança, pois a linguagem de programação que faz sucesso hoje, pode já não ser mais tão relevante mês que vem ou ano que vem. Pensando nisso, o post será atualizado de forma anual.

No início da década de 90, um grupo de engenheiros dava início ao Projeto Green, na empresa **Sun Microsystems** (hoje pertencente à Oracle). A ideia era criar novas tecnologias que permitissem a comunicação entre diferentes dispositivos utilizados no dia a dia das pessoas, como televisão, vídeo cassete, aparelhos de TV a cabo, eletrodomésticos e entre outros.

Sim, a ideia era promissora para a época, mas não deu certo, o mercado ainda não possuía espaço para a tecnologia. Mas nem tudo foi em vão, afinal, esse projeto deu origem à **linguagem de programação Java.** Com a expansão da internet, a linguagem foi conquistando espaço e sendo utilizada para executar aplicações nos browsers. Desta forma, transcendendo a simples interpretação de códigos HTML.

Atualmente ela é usada para diversos fins e está presente também em sistemas operacionais, dispositivos móveis, mainframes e por aí vai.

Como Explicar esse Sucesso?

Bem, existem diversas características que tornam a linguagem Java tão popular, talvez a mais em evidência nos dias de hoje seja o fato de que Java é a linguagem base para o desenvolvimento de aplicações mobile para Android, simplesmente a plataforma mobile mais utilizada no mundo. Entretanto, além disso, podemos citar:

▪ A sintaxe similar a outras linguagens mais antigas, como C e C++. Isso proporciona fácil entendimento para programadores dessas linguagens.

▪ Suporte à Orientação a Objetos – Java é uma das principais representantes das linguagens orientadas a objetos.

▪ Portabilidade – a compilação do código fonte cria um executável que, por sua vez, será interpretado por uma máquina virtual. A máquina virtual funciona como um intermediário entre o código e a plataforma na qual esse código será executado. Isso permite que a aplicação seja executada em diferentes plataformas **(Filosofia WORA: write once, run anywhere. Em português, "Escreva uma vez, execute em qualquer lugar").**

▪ Alta performance e entre outros fatores!

Enfim, existem muitas outras características que justificam a importância e popularidade dessa linguagem. Se você pretende aprender Java, na Becode possuímos um **curso completo de Orientação a Objetos com Java**, vale a pena conhecer!

Linguagem C

Criada pelo cientista da computação Dennis Ritchie em 1972, a linguagem C é derivada das antigas ALGOL 68 e BCPL. Ela surgia da necessidade de escrever programas de maneira mais fácil que a linguagem Assembly (mais próxima do código de máquina e, portanto, mais complexa de se entender).

No ano seguinte à sua criação, C é usada para escrever uma versão do sistema operacional Unix e, com isso, passou a ganhar notoriedade. Mas foi a partir do ano de 1978, com a publicação do livro **"The C Programming Language"**, que a linguagem passou a ser difundida no mercado, sendo utilizada também em outras áreas e para outros fins na programação de computadores.

Vantagens da Linguagem C:

▪ Por ser estruturada, a linguagem C torna o processo de desenvolvimento mais simplificado.

▪ Fácil portabilidade

▪ Simples, pois conta com um conjunto de bibliotecas de funções e sub-rotinas que auxiliam no desenvolvimento de sistemas.

▪ Provê recursos também de baixo nível, permitindo a incorporação de códigos Assembly.

Hoje em dia, além de grande parte dos sistemas operacionais existentes no mercado serem escritos em C, a linguagem também é muito utilizada no desenvolvimento de interpretadores, compiladores, editores de texto, softwares de computação gráfica e tratamento de imagens, banco de dados e entre outros.

Linguagem C++

Como o próprio nome já nos permite concluir, C++ é uma linguagem baseada em C, criada na década de 80 por Bjarne Stroustrup. Inicialmente, o objetivo do cientista da computação era desenvolver uma nova versão do Kernell (núcleo) do sistema operacional Unix e, para isso, escolheu a linguagem C como base. Sua escolha foi devido à performance, portabilidade e possibilidade de uso para diversos fins.

Durante o desenvolvimento, Stroustrup implementou diversas melhorias à linguagem C, incluindo alguns recursos de orientação a objetos. A linguagem criada pelo cientista da computação recebia, inicialmente, o nome de **"C com classes".** Entretanto, 3 anos mais tarde passou a ser chamada de C++.

C++ tornava-se assim uma linguagem muito poderosa, capaz de resolver problemas ainda mais complexos. Ela continua em constante evolução e hoje é utilizada nos mais diversos tipos de aplicações, como: jogos, editores de texto, editores de imagem e entre outras tantas finalidades.

Principais Características

- Multi-paradigma
- Compatibilidade com a linguagem C
- Portabilidade
- Boa performance

Linguagem C#

C# (lê-se "c sharp"), trata-se de uma linguagem de programação desenvolvida pela Microsoft e lançada em julho de 2002. A linguagem é um dos recursos da plataforma .NET (pronuncia-se "dot net"), que foi criada com o objetivo de melhorar a comunicação entre diferentes tecnologias utilizadas pela empresa.

É uma linguagem orientada a objetos, cuja sintaxe foi baseada nas precursoras C++, Java e Object Pascal. Desse modo, programadores que conhecem pelo menos uma destas linguagens, podem facilmente aprender a programar em C#.

Principais Características

- Suporte à Orientação a Objetos;
- Uso do conceito de máquina virtual (assim como Java)
- Portabilidade
- Sintaxe simples e de fácil compreensão

O sucesso da linguagem C# é muito relacionado a sua constante evolução, mas também pelo leque de recursos que a tecnologia oferece, aumentando a produtividade no processo de desenvolvimento. A linguagem pode ser utilizada na criação de diversos tipos de aplicações, com foco em soluções de alto nível.

Python

Criada no início dos anos 90, Python é uma linguagem de programação desenvolvida e distribuída pela Python Software Foundation, comumente utilizada para fins diversos na programação. Considerada uma linguagem de altíssimo nível, Python suporta diferentes paradigmas de programação e conta com recursos poderosos.

Uma das principais características da linguagem Python é a legibilidade do código. A linguagem possui uma sintaxe moderna e clara, o que favorece a escrita de códigos organizados, fáceis de compreender e manter, sem perda em produtividade.

Outras duas propriedades fazem do Python uma das principais linguagens de programação do mercado:

- Suporte a múltiplos paradigmas de programação
- Desenvolvimento comunitário, o que facilita a constante evolução da linguagem

Python é muito usado em aplicações web, soluções complexas, jogos e entre outros. Essa linguagem também é frequentemente ensinada em cursos de lógica de programação devido à simplicidade da sintaxe. Além de tudo isso, Python tem sido muito utilizado para o desenvolvimento de aplicações que se utilizam de técnicas de **Inteligência Artificial e Aprendizado de Máquina (Machine Learning – ML).**

JavaScript (JS)

Muita gente confunde JavaScript com Java, por isso é preciso deixar isso bem claro. São duas linguagens de programação distintas. Em outras palavras…

JavaScript não é Java!

Essa confusão entre as duas linguagens existe desde o lançamento do JavaScript e se deve, claro, aos seus nomes. Mas existe uma razão para elas possuírem nomes parecidos?

Sim! O JavaScript estava sendo desenvolvida pela empresa Netscape, que negociava na época uma parceria com a Sun Microsystems (do Java, lembra?). Assim, para aproveitar um pouco do sucesso da linguagem Java, em uma estratégia de Marketing, a linguagem que antes era chamada de LiveScript, foi lançada com o nome de JavaScript.

Além dos nomes, as sintaxes de Java e JavaScript também são parecidas. Mas isso também têm explicação: ambas, assim como muitas linguagens da época, foram baseadas em C, daí a semelhança.

OK, Mas o que é Javascript?

JavaScript é uma linguagem de programação criada para navegadores, com o objetivo de proporcionar maior interatividade às páginas web. Suportada hoje por todos os navegadores, o JavaScript é padronizado pela ECMA Internacional e considerado a linguagem de programação Web mais popular do mercado.

Se você buscar materiais sobre JavaScript na web, provavelmente você encontrará muita informação sobre o lado **"client-side"** da linguagem. Mas, o que é isso?

Client-side x Server-side

Grosseiramente, dizemos que o que é executado na interface do usuário é considerado **client-side (lado do cliente)**, como o que aparece na sua tela (imagens, textos, cores, etc etc). Por outro lado, tudo o que é executado no servidor é **server-side (lado do servidor)**, por exemplo, a interação de um website com o seu banco de dados, local onde estão armazenados arquivos HTML, imagens, vídeos, dados de texto e demais arquivos que compõem a página que aparece para o usuário final.

Por que o JavaScript é tão especial?

Com JavaScript você pode trabalhar os elementos no client-side: alterar a estrutura de um documento HTML, modificar estilos CSS, realizar ações conforme a interação do usuário com a sua aplicação, realizar validações de formulários e muito mais. Em outras palavras, o JavaScript é o responsável por trazer vida a uma página web no lado client-side.

Entretanto, o JS é muito mais do que isso. Atualmente, devido a constante evolução da linguagem, a tecnologia pode ser utilizada inclusive no lado server-side e aplicações mobile. Desta forma, tornando-se uma das linguagens mais versáteis existentes.

Na Becode, possuímos um **curso focado em iniciantes em HTML, CSS e JavaScript**. Em outras palavras, para aqueles que desejam aprender essas tecnologias do zero!

Perl

Perl é uma linguagem de programação criada e mantida pelo programador americano Larry Wall. A ideia de Wall era criar uma linguagem que, acima de tudo, fosse prática e proporcionasse mais liberdade ao desenvolvedor.

Principais Características

▪ Multiplataforma

▪ Código aberto

▪ Fácil de aprender, principalmente quando já se tem familiaridade com outras linguagens

▪ Usos diversos: aplicações web, interfaces gráficas, programação de redes, processamento de textos, jogos, etc.

Quanto ao mercado de trabalho, há quem diga que a linguagem esteja caindo em desuso. No entanto, Perl continua em aprimoramento e ainda se mantém boas posições nos rankings mundiais de linguagens de programação. Entretanto, talvez não seja a realidade brasileiro.

Assembly

Assembly é uma linguagem de programação de baixo nível, também definida, muitas vezes, como linguagem de montagem. Nascida em meados dos anos 50 (Sim! Anos 50), o Assembly abriu as portas para a segunda geração de linguagens de programação, quando os computadores ainda funcionavam a válvulas.

Na época, se utilizava notação binária para programar, o que era uma tarefa extremamente difícil. Assim, a linguagem Assembly surgiu como uma alternativa, pois poderia ser compreendida de uma forma menos complexa. Ainda assim, o código Assembly necessitava ser traduzido para a linguagem de máquina. Neste momento, surge o Assembler, software capaz de realizar essa “tradução”.

ATENÇÃO! não confunda a linguagem Assembly com o sistema Assembler.

Provavelmente você deve estar se perguntando:

“Se Assembly é uma linguagem tão complexa, então ninguém mais deve usar, certo?”

Errado! Embora seja uma linguagem complicada, o Assembly ainda é muito utilizado no desenvolvimento de sistemas que atuam de forma mais próxima ao hardware, como drivers de dispositivos, firmwares e microcontroladores. Além de aplicações que precisam do máximo de recursos da máquina.

PHP

PHP é uma linguagem de programação de livre distribuição, utilizada em todo o mundo para criação de sistemas web dinâmicos. OK, mas o que significa PHP? É uma longa história, mas vamos resumi-la para você.

Com a expansão da internet, o programador Rasmus Lerdorf criou uma ferramenta simples para contabilizar o número de visitantes de suas páginas. Daí surge o **Personal Home Page Tools**, o embrião da linguagem PHP, baseado nas linguagens C e Perl.

Na segunda versão do PHP – que até então ainda não era uma linguagem de programação – novas funcionalidades foram adicionadas, dentre elas a interpretação de formulários. A partir disso, Lerdorf disponibilizou o código fonte do PHP para outros programadores que passaram a trabalhar no projeto e criar novos recursos à ferramenta.

Em 1998, os programadores israelenses Zeev Suraski e Andi Gutmans reescreveram o PHP, oferecendo novas funções e características capazes de torná-la uma linguagem de programação.

Desse modo, a linguagem se mantinha com o nome PHP, devido à popularidade já conquistada, mas o significado da sigla passava a ser PHP Hypertext Preprocessor, um acrônimo recursivo.

Os códigos PHP são interpretados no servidor, logo trata-se de uma linguagem server-side. Sempre que o navegador solicitar, o interpretador processa o código da página e gera um HTML, que será enviado como resposta ao cliente. Podendo incluir, por exemplo, informações do banco de dados, já que o PHP possui essa funcionalidade.

Principais Características

▪ Suporte tanto à programação estruturada, quanto à orientação a objetos

▪ Fácil aprendizado (é necessário aprender HTML antes)

▪ Boa performance

▪ Portabilidade

▪ Código aberto, é liberado para a comunidade de programadores trabalhar na evolução da linguagem e consultar problemas já resolvidos anteriormente

Se for para resumir, podemos dizer que o PHP é uma das linguagens programação mais populares e, portanto, uma das mais requisitadas pelo mercado de Desenvolvimento Web.

Ruby

Criado em 1995 pelo programador japonês Yukihiro Matsumoto (mais conhecido como Matz), o Ruby é uma linguagem de programação orientada a objetos e de sintaxe simples. A proposta de Matz era desenvolver uma linguagem legível, fácil e agradável, daí o slogan da linguagem:

"O melhor amigo do programador"

Inspirada em linguagens como Perl, LISP e SmallTalk, Ruby é utilizada principalmente no desenvolvimento de aplicações web.

Principais características

▪ Linguagem interpretada

▪ Multiplataforma

▪ Produtividade

▪ Código aberto (open source), mantido por uma comunidade ativa de desenvolvedores de todo o mundo

Ruby on Rails não é Ruby!

É muito comum encontrar materiais sobre essa linguagem com o nome de Ruby on Rails (ou Rails ou RoR). Contudo, fique esperto! Ruby on Rails é um framework utilizado em conjunto com o Ruby. Entretanto, Ruby on Rails não é a linguagem de programação. A linguagem de programação é apenas "Ruby".

Dito isso, precisamos dar o devido mérito ao framework. O Ruby on Rails é responsável por muito do sucesso obtido pelo Ruby. Atualmente, muitas startups escolhem Ruby e Ruby on Rails para desenvolver suas aplicações, pois a tecnologia permite que se dedique mais tempo e atenção ao negócio em si do que ao desenvolvimento. Isto ocorre, pois normalmente os prazos para conclusão dos projetos que usam essas tecnologias são mais curtos, tamanho é a produtividade que a linguagem e o framework proporcionam.

Ruby já é uma das linguagens de programação mais populares e ainda possui altas perspectivas de crescimento. Trata-se de uma tendência! Aplicações web de grande relevância, como Twitter e GitHub utilizam a linguagem. Vale a pena aprender essa joia (literalmente)

Go é uma linguagem de programação criada pela Google. Possui o propósito de aumentar a produtividade em projetos. A linguagem foi lançada em 2009, surgindo como solução para atender a diversas necessidades da empresa.

O foco da linguagem é a performance, buscando ótimos desempenhos tanto da compilação, quanto de processamento da aplicação. A Go é multiplataforma, com suporte para Linux, Windows, MacOS e entre outros.

Outras Características

- Código aberto
- Combina recursos de alto e baixo nível
- Sintaxe simples, buscando facilidade para aprender e programar
- Altamente escalável
- Ótimo recurso de programação concorrente

A linguagem Go vem alcançando posições cada vez melhores nos rankings das linguagens de programação. Dropbox, Uber e SendGrid são alguns exemplos de empresas que utilizam essa tecnologia. No Brasil, a tecnologia ainda não é muito forte.

Swift

Criada pela Apple, Swift é uma linguagem de programação destinada ao desenvolvimento de aplicativos para as plataformas da marca, como Mac OS, iOS, Apple Watch e Apple TV.

Swift é open source e foi projetada também com o objetivo de proporcionar liberdade para os programadores. Possui sintaxe simples, performance e possibilidade incorporar códigos em Objective-C (antecessor ao Swift).

A linguagem Swift vem ganhando cada vez mais espaço no mercado de TI. Afinal de contas, se você deseja trabalhar com os produtos da Apple, você precisa aprender Swift ou Objective-C. E, convenhamos, a Apple ocupa uma fatia substancial do mercado de TI.

Para aprender mais sobre o Swift e dar os seus primeiros passos com o desenvolvimento de aplicações iOS, confira o nosso **curso de Swift – Aprenda a programar para iOS!**

Visual Basic (VB)

Em meados da década de 60, os matemáticos John George Kemeny e Thomas Eugene Kurtz criaram a linguagem de programação BASIC. Esse nome é um acrônimo de **Beginner's All-purpose Symbolic Instruction Code** que, em português, significa **Código de Instrução Simbólica para Iniciantes.**

O objetivo dos professores era criar uma linguagem para uso didático, mais simples de ser assimilada pelos estudantes. Assim, o BASIC, que havia sido inspirado em FORTRAN e ALGOL 60, ganhava cada vez mais popularidade, exatamente pela facilidade de aprendizado.

Já na década de 90, depois de inúmeros trabalhos realizados com o Basic, a Microsoft lançava a primeira versão do Visual Basic, uma linguagem de programação orientada a objetos baseada em Basic que, dentre diversas funcionalidades, passou a contar com recursos para criar interfaces gráficas para o usuário.

Em 2002, a linguagem passava a fazer parte da plataforma .NET da Microsoft, passando a ser chamada comumente de VB .NET. Para a empresa, tornar o VB um novo integrante da plataforma .NET foi uma forma eficaz de unir a produtividade oferecida pela linguagem aos poderosos recursos oferecidos pelo framework.

O Visual Basic .NET, embora tenha perdido espaço para outras tecnologias como Java e C#, ainda é muito utilizado no mercado de soluções de alto nível.

Linguagem R

Desenvolvida na década de 90, R é uma linguagem de programação destinada à computação estatística. O nome da linguagem vem das iniciais de seus criadores, os estatísticos neozelandeses Ross Ihaka e Robert Gentleman.

R é open source, sendo constantemente aprimorado por diversos profissionais pelo mundo. A linguagem também é multiplataforma, com suporte para sistemas operacionais Linux, Windows e Mac.

Desde seu surgimento, a linguagem R é amplamente utilizada no desenvolvimento de aplicações de estatística, sistemas para construção de gráficos, softwares de análise de dados e entre outros. R conta com grandes bibliotecas de funções específicas para a área de estatística, além de um importante Ambiente de Desenvolvimento Integrado (IDE) que recebe o mesmo nome da linguagem.

Ultimamente, a linguagem R ganhou muito destaque devido a sua constante utilização para o desenvolvimento de sistemas baseados em técnicas de Machine Learning – ML. Em outras palavras, R é uma tendência, provavelmente, já uma realidade em países mais adiantados.

Objective-C

Trata-se de uma linguagem de programação orientada a objetos, baseada em SmallTalk e C, multiplataforma, criada pelos cientistas da computação Brad Cox e Tom Love no início dos anos 80. A ideia era criar uma linguagem de programação que primasse pela reutilização de código.

O Objective-C hoje pertence à Apple, sendo utilizada no desenvolvimento de aplicações para o sistema iOS. Embora a companhia tenha criado a linguagem Swift baseada em Objective-C, a intenção, segundo a própria Apple, não é uma linguagem substituir a outra, mas sim fazer com que ambas sejam capazes de coexistir. Assim, aplicativos desenvolvidos em Swift podem ter partes do código escritas em Objective-C e vice-versa.

Apesar dessa filosofia apresentada pela Apple, o Objective-C encontra-se em declínio, devido a ascensão da linguagem Swift. Entretanto, a linguagem ainda apresenta boas colocações nos rankings de popularidade mundial.

Afinal, qual é a Melhor Linguagem de Programação?

Não há! Entenda que não existe uma linguagem melhor que outra, cada uma tem suas características que a tornam mais adequada para cada projeto. É muito importante que você identifique as necessidades da sua aplicação e busque uma linguagem que ofereça os melhores recursos para atendê-las. Claro, dependendo da sua área de atuação e especialização no mercado de desenvolvimento, você irá acabar trabalhando mais com uma linguagem ou outra. Abaixo alguns exemplos:

- Para desenvolvedores front-end: JavaScript
- Para desenvolvedores back-end: Java, PHP, C, C++, Python, Ruby, C# e por aí vai
- Para desenvolvedores mobile: Java, Swift, Objective-C e JavaScript
- Para Cientistas de Dados: Python ou R
- Para iniciantes: Python ou Ruby