Kauã – Azul

Leonardo – Vermelho

Matheus – Verde

INTRODUÇÃO

O avanço tecnológico intenso é o que característica o século na qual estamos vivendo. Tecnologias que considerávamos impossíveis de acontecerem, e que apenas as teorizávamos em filmes de ficção cientifica, se provaram com o tempo, realisticamente possíveis. Inteligências Artificiais, Carros elétricos e autônomos, robôs automatizando processos manuais em empresas, tudo isso é um pouco assustador a primeira vista, mas igualmente intrigante ao mesmo tempo.

Como chegamos a tudo isso? Para podermos recapitular nosso entendimento da evolução tecnológica humana. Por que não estudarmos como começou tudo isso, especificamente na área de Desenvolvimento de Software?

CONTEXTO DE UM SOFTWARE (RESUMO)

Um software, se trata de uma sequência de instruções escritas que serão interpretadas por um computador, e por meio dele, serão executadas as tarefas descritas pelas instruções. Tais instruções são escritas em linguagens especificas que o computador entende, essas sendo as linguagens de programação. Geralmente, o software é classificado como a parte lógica de um computador, enquanto o hardware é a parte física do computador.

Os softwares atualmente são enquadrados em tipos diferentes:

Software de Sistema: Os famosos sistemas operacionais

Softwares de Programação: Sistemas criados para se desenvolver outros sistemas, as famosas IDES

Software de Aplicativo: Programas que possuem como objetivo realizar uma série de tarefas variadas.

Softwares Web: São as paginas na internet que acessamos através de navegadores de internet.

Software IA: Esses que são uma categoria até que recente, se tratam de sistemas que possuem capacidades cognitivas próprias, como raciocínio, interpretação e entendimento.

O QUE SÃO PADRÕES E METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE

Para se desenvolver um software, além de possuir o conhecimento da linguagem empregada para a construção daquele sistema, é esperado cada vez mais dos desenvolvedores, que eles possuam conhecimento sobre metodologias e técnicas para desenvolver seus sistemas. Ou seja, com sistemas feitos seguindo padrões e regras para sua estrutura.

Sistemas seguindo padrões são mais fáceis de se entender, e mais fáceis de se fazer a devida manutenção. Consequentemente, ficando mais prático de se trabalhar, principalmente para alguém que estava por fora do projeto, coisa que em grandes empresas com grandes projetos em equipe, se torna estritamente necessário.

Devido a isso, existem os padrões de projeto, formatos de sistemas orientados a objetos criados com base em resolver problemas específicos. Embora a resolução de problemas não enquadre uma solução como padrão de projeto, já que existem características únicas que um padrão de projeto deve ter, como:

* Devem possuir um nome, que descreva o problema, as soluções e consequências. Um nome permiti definir o vocabulário a ser utilizado pelos projetistas e desenvolvedores em um nível mais alto de abstração.
* Todo padrão deve relatar de maneira clara a qual (is)problema(s) ele deve ser aplicado, ou seja, quais são os problemas que quando inserido em um determinado contexto o padrão conseguirá resolvê-lo. Alguns podendo exigir pré-condições.
* Solução descreve os elementos que compõem o projeto, seus relacionamentos, responsabilidades e colaborações. Um padrão deve ser uma solução concreta, ele deve ser exprimido em forma de gabarito (algoritmo) que, no entanto, pode ser aplicado de maneiras diferentes.
* Todo padrão deve relatar quais são as suas consequências para que possa ser analisada a solução alternativa de projetos e para a compreensão dos benefícios da aplicação do projeto.

Alguns dos padrões de projetos são o:

Abstract Factory: Esse padrão de projeto tem como objetivo facilitar a criação de heranças de objetos, ou seja, criando objetos e configurando sua herança da forma mais fácil e prática possível.

Singleton: Esse padrão tem como objetivo limitar a criação de um objeto proveniente de uma classe, esse objeto será único e não haverá nenhuma outra instancia da mesma classe senão ele. Por meio dessa limitação, instancias que serão de suma importância no projeto e que devem se manter como únicas poderão cumprir essa restrição.

Além disso, o conhecimento de metodologias que facilitam o processo de desenvolvimento com um time se torna necessário para se atuar na área. Já que, atualmente, desenvolver um software é uma tarefa comumente feita em grupo, com prazos, diferentes componentes com suas personalidades e especialidades, e com tarefas a todo momento.

A falta de organização em um projeto de desenvolvimento pode definir se aquele sistema será finalizado com sucesso, ou se virá a se tornar uma completa falha. Devido a isso, todo projeto usa uma metodologia de desenvolvimento, seja ela AGILE, SCRUM etc. Cada uma contando com suas próprias características e formatos de se trabalhar.

E ONDE ENTRA A EVOLUÇÃO DE SOFTWARE NISSO TUDO?

Pois bem, e o que tudo isso tem a ver com a evolução de software? Parando para analisar, metodologias e padrões de projeto por si só, não parecem ter muita conexão com o tema. Mas, se você parar para pensar um pouco, vai pensar que isso tudo tem á ver, já que, a criação e utilização de metodologias e projetos só foi proveniente graças a evolução do conceito de desenvolvimento de sistemas por si.

Porém, algo a mais faz uma ponte nesse assunto, sendo isso o DevOps, que se trata de um conjunto de práticas que engloba vários conhecimentos relacionados a parte de desenvolvimento de sistemas, sendo alguns dos mais importantes, o conhecimento sobre metodologias e sobre padrões de projeto.

O QUE É DEVOPS?

A palavra "DevOps" é a combinação dos termos "desenvolvimento" e "operações". No entanto, ela representa um conjunto de ideias e práticas que ultrapassam o significado desses dois termos. O DevOps inclui segurança, maneiras colaborativas de trabalhar, data analytics, e muitas outras práticas e conceitos, como as metodologias e padrões de projeto.

Ou seja, o DevOps é a combinação de filosofias culturais, práticas e ferramentas que aumentam a capacidade de uma empresa de distribuir aplicativos e serviços em alta velocidade.

Essas ideias provenientes do DevOps podem ser novas funcionalidades de software, uma solicitação de aprimoramento ou uma correção de bug, entre outros.

Tais abordagens exigem comunicação frequente entre as equipes de desenvolvimento e operações, trabalho colaborativo e empatia com os demais membros das equipes, coisa que vem do gerenciamento constante das tarefas e prazos feitos com metodologias seguras e confiáveis.

Já para as etapas de desenvolvimento, os desenvolvedores, que normalmente criam códigos em um ambiente de desenvolvimento padrão, trabalham em estreita colaboração com a equipe de operações de TI para acelerar a compilação de programas de software, a realização de testes e o lançamento de soluções, sem sacrificar a confiabilidade, principalmente utilizando de padrões de projetos para facilitar todo o procedimento.

Algumas empresas que empregam o DevOps são a Netflix e Amazon, que passaram a atribuir seu sucesso devido a aceitação dessa cultura de desenvolvimento.

RETORNANDO Á EVOLUÇÃO DE SOFTWARES

Mais uma vez, o que isso tudo tem a ver com a evolução dos softwares? Antes de se adentrar no tópico referente a evolução dos softwares é importante se entender a base do que é o cenário atual de desenvolvimento de software, visto do ponto de vista do devOps.

Sabendo como este é de forma atual, e depois disso, entender como era antigamente, facilita a visualização de todo o processo evolutivo que tivemos nessa área para chegarmos aonde estamos, reforçando assim, o quanto avançamos no espaço de tempo que tivemos.

Pois bem, com os conceitos sobre Metodologias, Padrões e DevOps postos á mesa, se trata agora de irmos ao assunto principal da palestra, a evolução dos softwares vistos do ponto de vista do DevOps.

A EVOLUÇÃO DOS SOFTWARES

**início a 1949**

Um algoritmo para o que teria sido o primeiro software foi escrito por Ada Lovelace no século 19, para o planejado Analytical Engine. Ela criou provas para mostrar como o motor calcularia os números de Bernoulli. Por causa das provas e do algoritmo, ela é considerada a primeira programadora de computador.

O termo "software" foi criado na década de 1940, e é um trocadilho com o termo hardware. "Hardware", em inglês, significa "ferramenta física". Software seria tudo o que faz o computador funcionar, através de instruções, excetuando-se a parte física dele.

A primeira teoria sobre software, anterior à criação dos computadores como os conhecemos hoje, foi proposta por Alan Turing em seu ensaio de 1936, On Computable Numbers, with an Application to the Entscheidungsproblem (problema de decisão) Isso eventualmente levou à criação dos campos acadêmicos de ciência da computação e engenharia de software; ambos os campos estudam software e sua criação. Ciência da computação é o estudo teórico do computador e do software (o ensaio de Turing é um exemplo de ciência da computação), enquanto a engenharia de software é a aplicação dos princípios da engenharia ao desenvolvimento de software.

Na década de 1940 os primeiros computadores elétricos, reconhecidamente modernos, foram criados. A velocidade limitada e a pequena capacidade da memória forçavam os programadores a escreverem à mão, economicamente, programas em linguagem de montagem (Linguagem assembly), logo, descobriu-se que programar em linguagem de máquina exigia um grande esforço intelectual e era uma atividade suscetível a erros.

**1950 a 1960**

Na década de 1950 as primeiras três linguagens de programação modernas, cujos descendentes ainda estão em uso difundido hoje, foram concebidas:

• FORTRAN (1954), a "FORmula TRANslator", inventada por John Backus e outros;

• LISP, a "LISt Processor", inventada por John McCarthy e outros;

• COBOL, a COmmon Business Oriented Language, criada pelo Short Range Committee, com grande influência de Grace Hopper.

Outro marco na década de 1950 foi a publicação, por um comitê de cientistas americanos e europeus, de "uma nova linguagem para os algoritmos", a ALGOL 60 através da publicação do relatório "The ALGOL 60 Report (the "ALGOrithmic Language")". Este relatório consolidou muitas ideias que circulavam na época e apresentou duas inovações chave quanto ao projeto de linguagens:

• Estrutura de blocos aninhados: pedaços significativos de código poderiam ser agrupados em bloco de instruções, sem ter que ser transformados em procedimentos separados e ser explicitamente chamados;

• Escopo léxico: um bloco podia ter suas próprias variáveis não acessíveis fora do bloco, e muito menos manipuláveis de fora do bloco.

**1967 a 1978**

O período compreendido entre o final dos anos 1960 à década de 1970 trouxe um grande florescimento de linguagens de programação. A maioria dos principais paradigmas de linguagem agora em uso foram inventados durante este período:

• Simula, inventada nos anos 1960 por Nygaard e Dahl como um super conjunto de Algol 60, foi a primeira linguagem a suportar os conceitos programação orientada a objetos.

• C, uma das primeiras linguagens de programação de sistemas, foi desenvolvido por Dennis Ritchie e Ken Thompson nos laboratórios da Bell entre 1969 e 1973.

• Prolog, projetada em 1972 por Colmerauer, Roussel, e Kowalski, foi a primeira linguagem de programação do paradigma lógico.

• ML, inventada por Robin Milner em 1973, é uma linguagem funcional, baseada em Lisp, estaticamente tipada.

Cada uma dessas línguas gerou toda uma família de descendentes, e linguagens mais modernas contam, pelo menos, com uma delas em sua ascendência.

**1980 a 1989**

A década de 1980 foi um tempo de relativa consolidação das linguagens imperativas. A linguagem C++ combinou orientação a objetos e programação de sistemas. O governo dos Estados Unidos padronizou a Ada, uma linguagem de programação de sistemas destinados à utilização por parte dos contratantes de defesa. No Japão e em outros lugares, vastas somas foram gastas investigando as chamadas linguagens de programação de quinta geração que incorporavam a programação lógica em suas construções. A comunidade de linguagens funcionais se dedicou a padronizar a ML e o Lisp. Ao invés de inventar novos paradigmas, todos estes esforços visaram aperfeiçoar as ideias inventadas na década anterior.

No entanto, uma tendência nova e importante na conceção de linguagens foi o aumento do foco na programação de sistemas de larga escala com o uso de módulos, ou em unidades organizacionais de código de larga-escala. As linguagens Modula, Ada e ML desenvolveram notáveis sistemas de módulos durante esta década, sendo que sistemas modulares muitas vezes eram associados com a programação genérica.

Embora os principais paradigmas novos para as linguagens de programação ainda não tivessem aparecido, muitos pesquisadores expandiram as ideias das linguagens existentes adaptando-os para novos contextos. Por exemplo, as linguagens dos sistemas Argus e Emerald adaptaram a programação orientada a objeto para os seus sistemas distribuídos, alguns exemplos das linguagem da década:

• 1983 - Ada: Criada por Jean Ichbiah e sua equipe no CII Honeywell Bull, foi padronizada pelo Departamento de Defesa dos Estados Unidos para sistemas de missão crítica, especialmente no setor de defesa. Ada é conhecida por sua robustez e suporte a sistemas modulares.

• 1983 - C++: Desenvolvida por Bjarne Stroustrup, C++ combinou programação orientada a objetos com programação de sistemas. A linguagem tornou-se amplamente utilizada para o desenvolvimento de software de sistemas e aplicativos devido ao seu controle e eficiência.

• 1986 - Objective-C: Desenvolvida por Brad Cox e Tom Love, Objective-C combinou a orientação a objetos de Smalltalk com a simplicidade do C, tornando-se a base para o desenvolvimento de software em sistemas da Apple, incluindo o macOS e o iOS.

• 1987 - Perl: Criada por Larry Wall, Perl foi projetada para ser uma linguagem de script de propósito geral, combinando as melhores características de linguagens como C, sed, awk, e sh. Perl tornou-se popular para administração de sistemas, manipulação de texto e desenvolvimento web.

• 1989 - FL (Back-ups): Desenvolvida pela equipe da IBM, FL (Functional Language) foi uma linguagem de programação funcional experimental, criada para explorar técnicas avançadas em programação funcional e serviu como base para pesquisa em linguagens de programação e compiladores.

1990 a 1999

A década de 1990 foi uma época de grandes transformações no desenvolvimento de software, marcada pela popularização da internet, pelo crescimento da indústria de software comercial e pela evolução de paradigmas e ferramentas que moldaram o futuro da programação.

Com a explosão da World Wide Web, surgiram novas demandas e oportunidades, o que levou a um foco crescente em tecnologias relacionadas à internet, como linguagens de script e desenvolvimento web. As metodologias de desenvolvimento ágil começaram a ganhar força, enquanto o open source se consolidou como um movimento poderoso.

Nos anos 90, houve um aumento significativo no desenvolvimento de software voltado para a internet e para ambientes distribuídos. O modelo cliente-servidor ganhou popularidade, substituindo a arquitetura mainframe em muitas aplicações corporativas. Tecnologias como middleware e CORBA (Common Object Request Broker Architecture) permitiram a construção de sistemas mais escaláveis e distribuídos.

O desenvolvimento orientado a objetos continuou a crescer em importância, especialmente com a popularização de linguagens como Java e C++. As práticas de reuso de código e design patterns começaram a se firmar como padrões na indústria.

A década de 1990 viu a consolidação e a popularização de várias linguagens de programação que ainda hoje são amplamente utilizadas:

- 1990 - Visual Basic: Desenvolvida pela Microsoft, o Visual Basic facilitou o desenvolvimento rápido de aplicações (RAD) com uma interface gráfica intuitiva, sendo amplamente adotado para aplicativos Windows.

- 1991 - Python: Criada por Guido van Rossum, Python se destacou por sua sintaxe clara e legibilidade, tornando-se uma das linguagens preferidas para prototipagem rápida, automação, e mais tarde, para ciência de dados.

- 1995 - Ruby: Criada por Yukihiro Matsumoto, Ruby combinou a simplicidade do Python com o poder da orientação a objetos, sendo amplamente reconhecida por seu uso na web com o framework Ruby on Rails.

- 1995 - Java: Desenvolvida pela Sun Microsystems, Java foi projetada com a filosofia de "escrever uma vez, rodar em qualquer lugar", sendo amplamente utilizada para desenvolvimento de aplicações web e móveis.

- 1995 - JavaScript: Criada por Brendan Eich, JavaScript tornou-se a linguagem padrão para scripts do lado do cliente na web, essencial para o desenvolvimento de páginas interativas.

- 1995 - PHP: Criada por Rasmus Lerdorf, PHP foi amplamente adotada para o desenvolvimento de sites dinâmicos e sistemas web, especialmente em combinação com bancos de dados MySQL.

Nos anos 90, o desenvolvimento de software ainda era em grande parte baseado em modelos de desenvolvimento em cascata (waterfall), onde as fases de planejamento, design, implementação, testes e manutenção eram realizadas de forma sequencial. Contudo, as limitações desse modelo começaram a emergir, levando à criação de metodologias ágeis no final da década, que enfatizavam a iteratividade e a colaboração constante com o cliente.

2000 a 2009

A década de 2000 foi marcada por uma explosão no uso da internet, o surgimento das redes sociais, a consolidação do software como serviço (SaaS) e o crescimento do movimento open source. A virtualização e a computação em nuvem começaram a transformar a infraestrutura de TI, e o desenvolvimento de software passou a ser ainda mais orientado pela web e pela colaboração global.

Nos anos 2000, o desenvolvimento de software foi profundamente influenciado pela expansão da internet e pelo crescimento do comércio eletrônico. A popularização da web 2.0 trouxe uma ênfase na criação de aplicações web interativas e colaborativas, permitindo que os usuários participassem ativamente na geração de conteúdo.

O conceito de desenvolvimento ágil tornou-se dominante, com metodologias como Scrum e XP (Extreme Programming) ganhando força, desafiando os métodos tradicionais de desenvolvimento em cascata. Além disso, o crescimento da computação em nuvem mudou a forma como o software era distribuído e consumido, dando origem ao modelo SaaS (Software as a Service).

A década de 2000 viu o surgimento e a consolidação de várias linguagens de programação que foram projetadas para lidar com as demandas da web, bem como para promover o desenvolvimento rápido e a escalabilidade:

- 2000 - C#: Desenvolvida pela Microsoft como parte da plataforma .NET, C# foi projetada para o desenvolvimento de uma ampla gama de aplicações, desde desktop a web, sendo amplamente utilizada para desenvolvimento corporativo.

- 2003 - Groovy: Criada como uma linguagem alternativa para a JVM, Groovy introduziu uma sintaxe mais simplificada e flexível, facilitando o desenvolvimento de aplicações Java.

- 2004 - Scala: Desenvolvida por Martin Odersky, Scala combina programação orientada a objetos e funcional, sendo projetada para ser altamente escalável, o que a tornou popular em grandes sistemas distribuídos.

- 2007 - Go: Desenvolvida pelo Google, Go (ou Golang) foi projetada para ser simples, eficiente e segura, sendo amplamente adotada para sistemas de alta performance e escalabilidade.

A década de 2000 foi marcada pela consolidação das metodologias ágeis, que passaram a ser amplamente adotadas em substituição aos modelos tradicionais. O desenvolvimento iterativo e incremental, aliado ao foco na comunicação e na entrega contínua, mudou a forma como o software era desenvolvido.

Ferramentas de controle de versão como o Subversion (SVN) e, mais tarde, o Git, popularizado pelo GitHub, transformaram a colaboração entre equipes distribuídas, permitindo o desenvolvimento de software em escala global. As práticas de DevOps começaram a emergir no final da década, com a integração entre desenvolvimento e operações, impulsionada pela necessidade de automação e entregas mais rápidas.

2010 a 2019

A década de 2010 foi marcada pela consolidação da computação em nuvem, o crescimento explosivo da inteligência artificial (IA) e aprendizado de máquina, e o domínio dos dispositivos móveis. O software se tornou onipresente, impulsionado por tecnologias como big data, internet das coisas (IoT) e o desenvolvimento contínuo de metodologias ágeis e DevOps.

- Computação em Nuvem: Serviços como AWS, Azure e Google Cloud tornaram-se fundamentais, permitindo a escalabilidade e flexibilidade das aplicações.

- Inteligência Artificial e Aprendizado de Máquina: A IA se tornou uma parte essencial do desenvolvimento de software, com aplicações em reconhecimento de voz, visão computacional e análise de dados.

- Desenvolvimento Móvel: O Android e o iOS dominaram, com milhões de aplicativos sendo desenvolvidos e distribuídos globalmente.

- DevOps e Entrega Contínua: A integração entre desenvolvimento e operações transformou a entrega de software, com práticas de automação e CI/CD se tornando padrão.

- 2010 - Rust: Focada em segurança e performance, Rust ganhou destaque em sistemas onde a confiabilidade é crucial.

- 2011 - Dart: Criada pelo Google, Dart foi projetada para o desenvolvimento web e posteriormente adaptada para mobile, com destaque para o framework Flutter.

- 2011 - Kotlin: Desenvolvida como uma alternativa ao Java, Kotlin se tornou a linguagem oficial para o desenvolvimento Android.

- 2013 - TypeScript: Desenvolvida pela Microsoft, TypeScript é um superset do JavaScript que adiciona tipagem estática e outras melhorias, facilitando o desenvolvimento de aplicações web escaláveis.

- 2014 - Swift: Criada pela Apple, Swift substituiu o Objective-C como a principal linguagem para desenvolvimento de apps iOS e macOS.

O MERCADO DE SOFTWARE ATUAL

Já hoje em dia, comparado a seu início, as coisas estão muito diferentes. Não só está mais variado com diversas áreas de serviço que foram surgindo ao longo do tempo, como tambem ganhou cada vez mais o holofote do público, aumentando assim a quantidade de pessoas ingressando todos os dias no mercado de desenvolvimento de software, que devido a isso aumenta a competitividade desse ecossistema.

As maiores tendencias na área atualmente, incluem atividades como:

* **Integração De IA e Aprendizado de Máquina**: A Inteligência Artificial e o Aprendizado de Máquina têm ganhado destaque nos últimos anos e a tendência é que essa integração se fortaleça ainda mais com a passagem do tempo. Essas tecnologias permitem o desenvolvimento de sistemas mais inteligentes e autônomos, capazes de aprender e tomar decisões com base em dados e padrões identificados.
* **Crescente demanda por segurança cibernética**: Com o aumento das ameaças cibernéticas, empresas e usuários estão cada vez mais preocupados com a segurança de seus dados. A previsão é que haja uma demanda crescente por soluções de segurança cibernética, visando proteger informações sensíveis e evitar vazamentos.
* **Expansão do desenvolvimento de aplicativos móveis**: Os aplicativos móveis se tornaram parte integrante do nosso dia a dia, facilitando atividades e oferecendo uma experiência mais conveniente aos usuários. A previsão é que o mercado de desenvolvimento de aplicativos móveis continue em expansão, acompanhando o crescimento do acesso à internet por dispositivos móveis.
* **Computação em Nuvem**: A computação em nuvem tem revolucionado a forma como empresas e usuários acessam e armazenam dados. A tendência é que o desenvolvimento de software esteja cada vez mais voltado para soluções baseadas em nuvem, oferecendo maior flexibilidade, escalabilidade e segurança aos usuários.
* **Internet das Coisas (IoT)**: A Internet das Coisas é uma tendência que está transformando diversos setores, como saúde, agricultura, transporte, entre outros. Espera-se que em o mercado de desenvolvimento de software esteja cada vez mais voltado para o desenvolvimento de soluções que integrem dispositivos e sensores, permitindo a interconexão e troca de informações entre eles.
* **Automação de processos**: A automação de processos é uma tendência que visa otimizar e agilizar tarefas repetitivas e burocráticas. Espera-se um aumento na demanda por soluções de software que possibilitem a automatização de processos, tornando as operações mais eficientes e reduzindo custos.
* **Realidade Virtual e Realidade Aumentada**: A Realidade Virtual (VR) e a Realidade Aumentada (AR) têm ganhado destaque nos últimos anos e a tendência é que essas tecnologias sejam cada vez mais utilizadas no desenvolvimento de software.   
  Essas tecnologias oferecem experiências imersivas, seja por meio de jogos, simulações ou treinamentos, e têm um grande potencial de aplicação em diversos setores.

E juntamente dessas novas tendencias, as linguagens mais utilizadas da atualidade são:

* PYTHON
* C++ E C
* JAVA
* C#

Sendo Python a principal, devido a sua praticidade para trabalhar com Ciência de Dados, Aprendizado de Máquina, e consequentemente, Inteligência Artificial.

Atualmente, se prevê que a área venha a gerar 698,80 Bilhões de Dólares em receitas com um crescimento 5,27% á cada ano que se passe.

Obstáculos

Porém, nem tudo são rosas, a área tambem está sofrendo por diversos desafios na época que vivemos. Assim como dito no início, um dos problemas é o aumento da competitividade para se adentrar no mercado, mas não é o único enfrentado pela comunidade, alguns outros são:

* Evolução tecnológica acelerada: O desenvolvimento de software está sempre em constante evolução, e acompanhar as novas tecnologias e tendências pode ser um desafio. Surgirão novas linguagens de programação, frameworks e ferramentas que exigirão atualização constante dos profissionais e empresas do setor.
* Complexidade dos projetos: Com a demanda por soluções cada vez mais avançadas, os projetos de desenvolvimento de software podem se tornar complexos. Gerenciar projetos que envolvem múltiplas tecnologias, equipes e prazos curtos pode ser um desafio, exigindo habilidades de organização e gerenciamento eficientes.
* Segurança e privacidade: A segurança cibernética continuará sendo um desafio para empresas e desenvolvedores de software. Proteger os sistemas contra ataques, garantir a privacidade dos dados dos usuários e estar em conformidade com as regulamentações vigentes serão desafios constantes.
* Rápida obsolescência de tecnologias: As tecnologias estão em constante evolução, o que pode levar à rápida obsolescência de tecnologias anteriormente utilizadas. É necessário acompanhar as tendências e atualizar constantemente as habilidades para não ficar para trás no mercado.

Facilitadores

Junto desses desafios, outro vem de forma inesperada, que é, a administração da Inteligência Artificial no ambiente de desenvolvimento de software.

Com o desenvolvimento de sistemas de Inteligência Artificial, uma variedade de ferramentas acabaram sendo desenvolvidas, o que resultou em diferentes formas de resolver problemas, porém, tudo isso em uma velocidade exorbitante, o que tornou muito complicado administrar o que poderia ou não ser feito com a ia, alguns dos exemplos dessas atividades que foram reconhecidas como benéficas são:

* **Geradores de código**: Essas ferramentas podem gerar código automaticamente, baseado em especificações fornecidas pelo desenvolvedor.
* **Ferramentas de depuração**: São ferramentas podem usar IA para analisar o código e identificar erros.
* **Ferramentas de teste automatizado**: Aquelas ferramentas que podem usar IA para gerar testes automatizados, baseados no código do software.
* **Ferramentas de análise de dados**: Essas ferramentas podem usar IA para analisar dados de uso de software, para identificar padrões e tendências.

Porém, mesmo reconhecendo os riscos e dificuldades de se usarem IA, as empresas investem pesado nessa abordagem de garantir serviços personalizados de forma atual. Alguns exemplos de empresas que possuem aplicações que usam de IA, seriam a da própria Google com o Google Maps.

Nesse caso, o Google Maps é reconhecido por sua interface intuitiva, oferecendo recursos avançados, incluindo visualização por satélite, e integração perfeita com outros serviços da empresa.

Destacando-se como um aplicativo avançado em IA, ele prevê o tempo de chegada ao destino e fornece informações em tempo real sobre deslocamento e localização dos passageiros. Além disso, sugere rotas mais eficientes com base em dados históricos.

Não só o Google Maps, como a própria Netflix tambem possui formas de utilizar inteligência artificial em sua plataforma. A Netflix se destaca globalmente ao empregar IA em diversas áreas para otimizar operações e aprimorar a experiência dos usuários.

Ela aprende essencialmente com os dados que são fornecidos e, em seguida, usa seu próprio banco de dados para fornecer o conteúdo que melhor atende às suas expectativas.

CONCLUSÃO

Como visto ao longo do estudo da evolução do software, ao longo do tempo, mais e mais formas de se desenvolver software foram existindo, e graças aos incontáveis esforços de diversos especialistas, o ato de desenvolver software evoluiu e ficou cada vez mais simples. Comparado ao seu início, o processo ficou infinitamente mais fácil, ao ponto de permitir que praticamente qualquer pessoa possa aprender e agir na área, e isso é o que tende a aumentar, principalmente com o uso de Inteligências Artificiais que auxiliam os desenvolvedores a melhorar seus códigos.

Porém, algo interessante de se perceber, é que nenhum desses avanços teria ocorrido se os desenvolvedores não tivessem optado por tentar otimizar mais ainda suas soluções e buscassem criar novas formas de se desenvolverem seus softwares. Reforçando assim, que na área de TI, estudar constantemente para melhorar suas práticas de desenvolvimento é uma obrigação.