## Introdução

No cenário atual, a automação de tarefas é uma prática comum entre os desenvolvedores de software.

Pois buscam otimizar processos e aumentar a eficiência no ambiente de trabalho.

Só que esses profissionais enfrentam um desafio regular

Que é a ocorrência de erros nos scripts desenvolvidos.

Muitas das vezes esse erro só é detectado apenas após a execução, resultando em frustração e desperdício de recursos.

## Contextualização

Esse tipo de desafio ele é encontrado na empresa do Tom.

Que em busca de uma solução para esse problema, ele nos contactou e expressou sua necessidade por um sistema de

Correção automática de scripts,

Que identifique e corrija os erros de sintaxe ou lógica

E que também forneça feedback imediato sobre a entrada e saída esperada dos scripts.

Com base nisso, foi pensado na SCRIPT INTELIGENTE

### O QUE É?

A "Script Inteligente" é uma empresa que oferece uma solução inovadora e eficaz para a correção automática de scripts de software.

A principal característica da "Script Inteligente" é sua capacidade de analisar o código fonte de scripts em busca de erros de semântica, lógica ou sintaxe, proporcionando feedback imediato e sugestões de correção.

Além disso, a plataforma pode integrar-se com APIs externas ou utilizar algoritmos de inteligência artificial para realizar correções automáticas sempre que possível.

Por meio de uma interface web intuitiva, os usuários podem submeter seus scripts para correção, visualizar os erros identificados e suas correções sugeridas, bem como aproveitar funcionalidades adicionais, como a definição de entradas e saídas esperadas para os scripts.

### QUEM REALIZA?

É realizada por uma equipe multidisciplinar de profissionais especializados em desenvolvimento de software, inteligência artificial, análise de dados e engenharia de sistemas.

Giordano

Vitor

### POR QUE É IMPORTANTE?

Em resumo, embora serviços já oferecidos com soluções amplas e poderosas, a "Script Inteligente" pode oferecer uma abordagem mais especializada, personalizada e centrada no cliente para atender às necessidades específicas de Tom e de outros desenvolvedores em busca de uma solução eficaz para correção automática de scripts.

Personalização e Integração: Pode oferecer serviços altamente personalizados e integrados, adaptados às necessidades específicas de Tom e de sua equipe. Isso pode incluir recursos como a capacidade de incorporar diretamente a ferramenta de correção automática em seus fluxos de trabalho existentes ou a personalização de algoritmos de IA para se alinhar melhor com suas preferências e requisitos.

Segurança e Privacidade: Embora serviços comumente encontrados que sejam excelentes em termos de funcionalidade, alguns usuários podem ter preocupações adicionais com segurança e privacidade ao confiar em soluções de terceiros. A "Script Inteligente" pode oferecer garantias adicionais de segurança e privacidade, especialmente se fornecer opções para hospedar a solução em infraestrutura local ou oferecer controle total sobre os dados e scripts enviados para correção.

Suporte Especializado e Atendimento ao Cliente: Ao escolher a "Script Inteligente", Tom pode se beneficiar de suporte especializado e atendimento ao cliente dedicado. Isso significa que ele terá acesso a uma equipe altamente qualificada e experiente que pode ajudá-lo com qualquer dúvida, problema ou solicitação específica relacionada à correção automática de scripts.

Dessa forma reduzindo os custos de hora/trabalhada

### QUAIS SÃO AS ETAPAS DE UTILIZAÇÃO ENVOLVIDAS?

Essas etapas demostram como será a utilização do sistema

Submissão do Script: O usuário, faz o upload ou insere o script que deseja corrigir na plataforma.

Análise do Script: A plataforma realiza uma análise detalhada do código fonte do script em busca de erros de semântica, lógica ou sintaxe, utilizando algoritmos de análise estática e dinâmica.

análise dinâmica é feita durante a execução.

A análise estática é feita por meio da análise do código, sem executar o aplicativo

Identificação de Erros: Identifica os erros encontrados no script e os apresenta ao usuário de forma clara e organizada, indicando sua localização no código e oferecendo sugestões de correção.

Feedback: O usuário analisa o feedback fornecido pela plataforma e decide quais correções deseja aplicar ao seu script. Ele também pode aproveitar sugestões adicionais de melhoria fornecidas.

Correção Automática: Se configurado pelo usuário, a "Script Inteligente" pode realizar correções automáticas nos erros identificados, utilizando algoritmos de inteligência artificial ou integração com APIs externas.

Validação e Testes: Após aplicar as correções sugeridas, o usuário pode validar o script corrigido por meio de testes automatizados para garantir que os erros tenham sido corrigidos com sucesso.

### QUAL É O ARTEFATO?

O artefato principal produzido pelo processo é o script corrigido.

Esse script é o resultado mais significativo tangível da análise, identificação e correção de erros de semântica, lógica ou sintaxe realizado pela plataforma.

Pois representa a solução para o problema inicial apresentado, fornecendo assim ao usuário um código funcional e de qualidade para seus projetos de software.

### COMO GARANTIR QUE O TRABALHO FOI REALIZADO CORRETAMENTE?

Testes Automatizados:

Após receber o script corrigido, o usuário pode executar testes de unidade automatizados para verificar se o código funciona conforme o esperado.

Revisão Manual: O usuário pode revisar manualmente o script corrigido para verificar se as correções sugeridas estão de acordo com suas expectativas e requisitos.

Comparação de Versões: O usuário pode comparar a versão original do script com a versão corrigida para identificar as alterações feitas

E dessa forma entender o impacto das correções realizadas.

Feedback da Equipe: O usuário pode solicitar feedback de outros membros da equipe para avaliar a qualidade das correções realizadas.

Monitoramento Pós-Implementação: Após implementar o script corrigido em um ambiente de produção, o usuário pode monitorar seu desempenho e comportamento ao longo do tempo para garantir que as correções realizadas pela "Script Inteligente" não tenham causado regressões ou problemas inesperados.

### Para chegar nesse produto, têm-se a primeira etapa das etapas básicas de desenvolvimento de software

A etapa de Análise

Realizada pelo Analista de Sistemas

E tem a função de fazer o Levantamento das Necessidades

Por meio de Especificação de Requisitos

Utilizando da técnica de Entrevistas para obter essas informações

E ser possível gerar a modelagem de alto nível do sistema

### E desta forma, o que seria requisito?

Requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar.

requisito funcional é o que diz respeito as funcionalidades ou serviços do sistema

requisito não funcional pode ser descrito como um atributo de qualidade, de desempenho, de segurança ou como uma restrição geral em um sistema.

A análise de requisitos resulta na especificação das características operacionais do software, indica a interface do software com outros elementos do sistema e estabelece restrições a que o software deve atender.

“Iai, levando em consideração os benefícios de implementação do requisito, custo, questões de dependência entre os requisitos e riscos técnicos, estes requisitos foram priorizados: “

### Com base nesses requisitos é possível realizar a modelagem de alto nível do sistema

Uma linguagem padrão para descrever um projeto de software é a UML ( Unified Modeling Language )

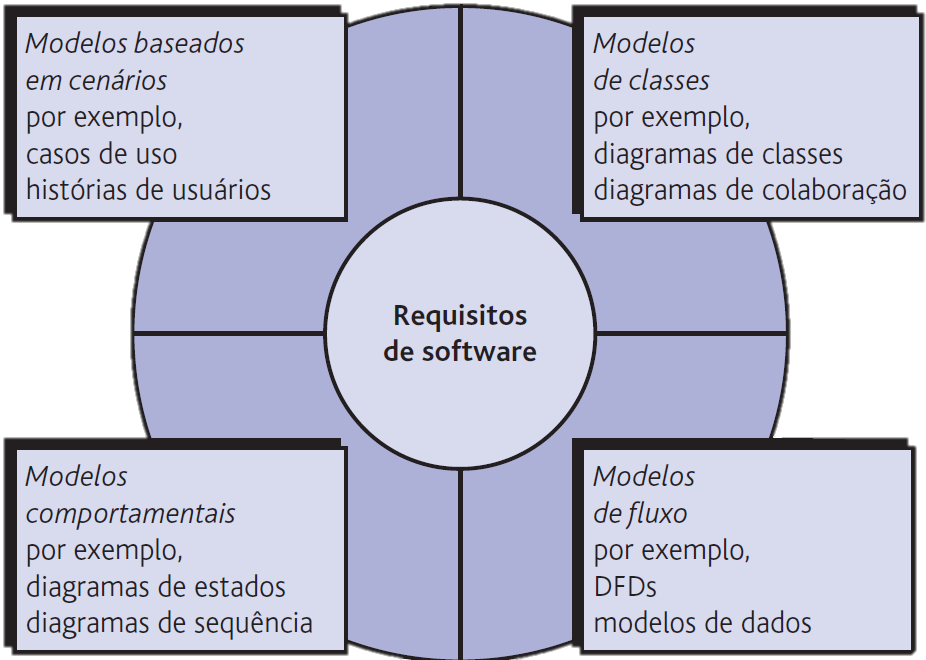
“Assim como os arquitetos criam plantas e projetos para serem utilizados por uma empresa de construção, os arquitetos de software criam diagramas UML para ajudas os desenvolvedores de software a construir o software.”

Vale ressaltar que “a omissão de uma característica particular não significa que esteja ausente, mas que foi suprimida, para evitar congestionar o diagrama com detalhes irrelevantes.”

A ideia aqui é a combinação de representações para que forneça o melhor modelo de requisitos de software e a ligação mais eficaz para o nosso projeto.

Então cada elemento do modelo de requisitos que será apresentado leva em consideração o problema segundo um ponto de vista

E a ação da modelagem de requisitos resultou nos 4 tipos de modelos



“Esses modelos dão ao projetista de software informações que podem ser transformadas em projetos de arquitetura, de interfaces e de componentes.”

Para o melhor entendimento da equipe, levou-se em consideração apenas as representações gráficas em diagrama.

“mas poderia ter a descrição de forma continua; enumerada; tabular; essencial”

### Os Modelo baseados em cenários

Possuem elementos que representam como o usuário interage com o sistema e a sequência específica de atividades que ocorre à medida que o software é utilizado.

Os “casos de uso são apena uma ferramenta para definir o que existe fora do sistema(ATORES) e o que deve ser realizado pelo sistema(CASOS DE USO)”

De forma que representa os requisitos funcionais do sistema.

Diagrama, Diagrama de Venn

Descrição gerada automaticamente

Aqui a gente tem a representação de como é a diagramação do caso de uso, onde o :

Ator: Elemento externo do sistema que sempre inicia o uso ou recebe um valor de caso de uso.

Caso de Uso: Serviço que o sistema fornece Caso de Uso aos usuários.

Interação: Estímulos recebidos pelo sistema, essa interação pode ser de :

Inclusão: para representar uma relação de inclusão entre dois casos de uso ou atividades.

Extensão: indicando que uma atividade opcional pode ser executada em determinadas condições

Generalização: Representar a relação de herança entre atividades.

Sistema: Contexto em que o caso de uso é utilizado (corresponde a uma classe ou um conjunto de classes).

### E agora os Modelo baseados em classe

Eles modelam os objetos que o sistema irá manipular, as operações que serão aplicadas aos objetos para efetuar a manipulação, os relacionamentos entre objetos e as colaborações que ocorrem entre as classes definidas.

Os elementos principais são ícones usados para representar classes e interfaces

Cada caixa é dividida em partes horizontais

A parte superior contém o nome da classe

A seção do meio lista os atributos da classe.

E ai esses atributos podem ser valores que a classe calcula a partir de suas variáveis de instância ou valores que a classe pode obter de outros objetos dos quais é composta.

Além disso cada atributo pode ter um nome, um tipo e um nível de visibilidade

O tipo e a visibilidade são opcionais.

O tipo vem após o nome e é separado por dois-pontos. A visibilidade é indicada precedendo pelo sinal –, #, ~ ou +, que indicam, respectivamente, visibilidade private, protected, package ou public

terceira seção do diagrama de classes contém as operações ou comportamentos da classe. Uma operação refere-se ao que os objetos da classe podem fazer. Usualmente é implementada como um método da classe

Cada operação também pode ser mostrada com um nível de visibilidade, parâmetros com nomes e tipos e um tipo de retorno

### E agora os Modelo baseados em comportamentos

Os elementos comportamentais representam como eventos externos mudam o estado do sistema ou suas classes, conceituando o diagrama de sequência

Mostra a interação entre objetos tendo em vista a sequência do tempo

Normalmente, um diagrama captura o comportamento de um único cenário

Facilitando o entendimento de como os objetos vão interagir dentro do caso de uso

Ele mostra a ordem temporal na qual as mensagens são enviadas entre os objetos para executar aquela tarefa

**Ator:**

Representa um usuário externo, sistema ou entidade que interage com o sistema em questão

Representado por um bloco retangular com o nome do ator.

**Objeto:**

Instância de uma classe ou entidade que participa da interação.

Representados por blocos retangulares com o nome do objeto e uma linha pontilhada.

**Linha de Vida:**

É uma linha vertical que representa a existência de um objeto ao longo do tempo.

E ela mostra o período durante o qual o objeto está ativo e participando da interação

Conectada ao bloco do objeto no topo e tem uma barra horizontal na parte inferior.

Mensagem:

comunicação entre objetos que indica uma interação específica.

As mensagens são representadas por setas entre as linhas de vida dos objetos

Fragmento de Interação:

Região do diagrama de sequência que representa uma parte específica da interação.

Apresenta diferentes tipos, como fragmento de opções (alt), fragmento de loop (loop), fragmento de paralelismo (par),

**Notação de Tempo:**

Usada para indicar restrições temporais em uma interação.

### E por fim os Modelo baseados em comportamentos

Elementos orientados a fluxo e representam o sistema como uma transformação de informações, indicando como os objetos de dados são transformados à medida que fluem pelas várias funções do sistema para isso tem-se o Diagrama de atividade

Dessa forma, a atividade representa uma ação ou etapa dentro do processo sendo modelado e é utilizada da seguinte forma:

Separados em:

**Nó de Inicialização**

O ponto de partida do processo.

**Nó de Decisão**

Usado para representar uma escolha entre duas ou mais opções de caminho.

**Nó de Fusão**

Usado para combinar diferentes caminhos de execução em um único caminho.

Ele indica o ponto onde diferentes fluxos convergem.

**Nó de Término**

Ponto de saída para o fluxo de atividades

REFERENCIAS

Os diagramas de classe também podem exibir relações entre classes.

E aqui a gente pode classificar em Generalização, Associação, Agregação, Composição, Multiplicidade

O que é? Forma de representar?

**Generalização ou herança**

Relacionamento de herança entre duas classes, indicando que uma classe (subclasse) herda atributos e métodos de outra classe (superclasse).

Uma classe que seja subclasse de outra classe é conectada a ela por uma seta com uma linha cheia como eixo e com uma ponta triangular vazia. A seta aponta da subclasse para a superclasse

**Realização ou implementação**

Indicar que uma classe implementa a interface especificada por uma ou mais interfaces.

Uma seta tendo como eixo uma linha tracejada indica implementação de interface.

Associação

Relacionamento básico entre duas classes, indicando que uma classe está de alguma forma relacionada com outra.

Pode ser bidirecional ou unidirecional

Pode ter multiplicidade

Ela pode ser rotulada, assim como cada uma de suas extremidades, para indicar o papel de cada classe na associação.

Setas em qualquer uma ou em ambas as extremidades de uma linha de associação indicam navegabilidade

Uma associação pode também conectar uma classe com ela própria, usando um laço.

A associação com uma seta em uma extremidade indica navegabilidade unidirecional. A seta significa que de uma classe pode-se facilmente acessar a segunda classe associada para a qual a associação aponta

Uma associação sem setas em geral indica uma associação bidirecional

**Agregação**

Tipo especial de associação que indica que uma classe é composta por outras classes (partes)

Representada por um losango vazio em uma extremidade do ícone, a seta aponta é considerada uma “parte” da classe na extremidade do losango da associação.

**Composição**

Forma mais forte de agregação, indicando que a vida da parte está intimamente ligada à vida do todo

Uma composição é uma agregação indicando forte relação de propriedade entre as partes

Representada por uma linha com um losango preenchido na extremidade da parte composta.

**Dependência**

Indica que uma classe depende de outra classe em algum aspecto, mas não tem uma relação estrutural direta. Ou seja, Uma classe depende de outra se alterações na segunda classe podem exigir alterações na primeira classe.

Representada por uma linha tracejada com uma seta na extremidade da classe dependente.

**Multiplicidade**

A multiplicidade serve para definir a quantidade de instâncias que uma classe pode ter em relação a outra em um relacionamento.

De uma extremidade de uma associação indica o número de objetos daquela classe associados à outra classe.

Projeto

Estudo de formas de implementação

Modelagem “implementacional” do sistema

Responsável: Projetista de Sistemas

Implementação

Codificação do sistema

Deve seguir modelos criados pelas fases anteriores

Pode seguir técnicas e padrões de codificação

Responsável: Programador

Verificação e validação

Verificação: é determinar se os produtos de software de uma atividade atendem completamente aos requisitos ou condições impostas a eles nas atividades anteriores

Validação: é garantir que o produto correto está sendo desenvolvido

Responsáveis: Analistas e Programadores /ou Equipe de Testes

Implantação

Preparação do ambiente de produção

Muitas vezes envolve o treinamento de usuários

Responsável: Analista/Programador/Suporte

Manutenção

Assegurar o correto funcionamento do sistema no ambiente de produção.

Responsável: Analista/Programador/Suporte