Aula-Tech

O que é uma GMUD?

GMUD é a sigla para **Gestão de Mudanças**, um processo formal utilizado em empresas e organizações para gerenciar e controlar as mudanças que precisam ser realizadas em sistemas de TI, processos de negócio, infraestrutura ou outros componentes importantes.GMUD é a sigla para **Gestão de Mudanças**, um processo formal utilizado em empresas e organizações para gerenciar e controlar as mudanças que precisam ser realizadas em sistemas de TI, processos de negócio, infraestrutura ou outros componentes importantes.

A GMUD é um conjunto de procedimentos e práticas que ajudam a garantir que qualquer alteração importante seja feita de maneira controlada, minimizando riscos e impactos negativos.

Etapas
1 - Pedido de Mudança: Alguém identifica a necessidade de uma mudança e faz um pedido formal, descrevendo o que precisa ser alterado e por quê.

2 - **Autorização**: Uma ou mais pessoas aprovam a solicitação de mudança, em caso de Tier 3 pra cima, precisa apenas da aprovação de coordenador, abaixo disso precisa passar pelo CAB que é basicamente um grupo de pessoas que analisam o pedido para entender os riscos e os benefícios. Eles verificam se a mudança é realmente necessária e se é segura.

3 - Implementação: A mudança é realizada conforme o plano.

4 - Verificação: Durante e após a mudança, são feitos testes para garantir que tudo está funcionando como deveria. É o famoso ˜Acompanhar deploy˜

O que é Taac?

É um tipo de teste integrado.

O principal objetivo dos testes integrados é garantir que os diferentes componentes de um sistema funcionem corretamente juntos, identificando e corrigindo problemas de integração.

Testam se os recursos estão funcionando corretamente, sem erro e considerando as regras de negócio.

São testes automatizados, ajudam a identificar se o sistema está funcionando como deveria.

O que é teste unitário/unit?

Teste unitário é uma prática de teste de software em que unidades individuais ou componentes do código são testados de forma isolada. O objetivo é validar se cada unidade de código funciona conforme o esperado. Uma unidade de código geralmente se refere a uma função, método, ou bloco de código que pode ser testado separadamente do restante do sistema.

Por exemplo: imagine a função: retornarAMenorTaxa() dentro de um serviço de uma API, esse é apenas uma das funções executadas, o teste unitário deve ser feito para testar unicamente a função retornarAMenorTaxa() para validar se ela individualmente está funcionando como deveria, é claro para chegar nessa função outras funções são executadas, mas o teste unitário se preocupe em testar funções separadas ao contrário do teste integrado.

Testes unitários devem cubrir uma porcentagem alto dos sistemas, 80% pra cima.

O que é uma API?

O que é uma API?

É um estilo de arquitetura para criar serviços web que permitem a comunicação entre diferentes sistemas pela internet.

Esses serviços funcionam como recursos que são disponibilidades pela API Através de endpoints disponíveis através dos métodos HTTP.

Métodos HTTP: A interação com esses recursos é feita utilizando os métodos HTTP. Os principais métodos são:
- **GET**: Recupera informações sobre o recurso.
- **POST**: Cria um novo recurso.
- **PUT**: Atualiza um recurso existente.
- **DELETE**: Remove um recurso.
- **PATCH**: Aplica atualizações parciais a um recurso.

Imagine um site, ele possuí um endereço base, por exemplo:
https://www.kabum.com.br
Esse é a pagina principal, mas ao clicar em produto um vc é direcionado para um recurso específico com o detalhe e opção de compra:
https://www.kabum.com.br/produto/238671/console-playstation-5-sony-ssd-825gb-controle-sem-fio-dualsense-com-midia-fisica-branco-1214a

Uma API funciona de forma semelhante há um site, você tem um endereço base e o endereço dos recursos disponíveis a partir do endereço base:

www.api-imob.tech/ -> endereço principal
www.api-imob.tech/propostas -> recurso que lista as propostas

Exemplo Prático:
Vamos imaginar uma API simples para gerenciar uma lista de tarefas:
- **URL Base**: https://api.minhastarefas.com
- **Endpoints**:
- GET https://api.minhastarefas.com/tarefas : Obter todas as tarefas.
- POST https://api.minhastarefas.com/tarefas : Adicionar uma nova tarefa.
- GET https://api.minhastarefas.com/tarefas/1 : Obter detalhes da tarefa com ID 1.
- PUT https://api.minhastarefas.com/tarefas/1 : Atualizar a tarefa com ID 1.
- DELETE https://api.minhastarefas.com/tarefas/1 : Deletar a tarefa com ID 1.

A principal diferença entre um site e uma API é o público alvo:

Site: Um site é projetado para ser acessado e interagido por seres humanos. Ele apresenta informações e conteúdo de uma maneira visualmente agradável e navegável, utilizando HTML, CSS e JavaScript para criar páginas web que podem ser visualizadas em navegadores.
API: Uma API é projetada para ser utilizada por outras aplicações e serviços, não por humanos diretamente. Ela fornece dados e funcionalidades através de chamadas programáticas, geralmente retornando informações em formatos como JSON ou XML que podem ser processados por outras aplicações.

O que é um endpoint?

Um endpoint é um ponto de acesso a um serviço ou recurso em uma API É onde uma aplicação cliente pode enviar requisições e esperar respostas.

Como Funciona um Endpoint?

1. **URL**: Cada endpoint tem um endereço específico na internet, chamado de URL. É como o endereço da sua casa, mas para dados na internet.
2. **Métodos HTTP**: Existem diferentes tipos de pedidos que você pode fazer para um endpoint, como:

- **GET**: Pedir informações. Exemplo: Ver as postagens mais recentes em uma rede social.
- **POST**: Enviar informações. Exemplo: Postar uma nova foto.
- **PUT**: Atualizar informações. Exemplo: Editar o texto de uma postagem antiga.
- **DELETE**: Deletar informações. Exemplo: Apagar uma postagem que você não quer mais.

O que é Gateway?

O gateway de API é uma ferramenta de gerenciamento de APIs que fica entre o cliente e uma coleção de serviços de back-end.

Um gateway é um dispositivo ou software que atua como intermediário entre dois sistemas ou redes diferentes, facilitando a comunicação entre eles. Ele desempenha o papel de "portão de entrada" ou "portal" que controla o acesso e direciona o tráfego de informações entre os sistemas.

Gateway de API é uma ferramenta poderosa para gerenciar, proteger e otimizar o acesso a uma API, facilitando a comunicação entre clientes e servidores de API de forma segura e eficiente.

Exemplos de Gateways:

- **Gateway de API geral**: Facilita a comunicação entre diferentes APIs, permitindo que aplicativos e sistemas usem serviços de terceiros de forma integrada.
- **Gateway de API Pagamento**: Facilita a comunicação entre um site de comércio eletrônico e o sistema de processamento de pagamentos, garantindo transações seguras e eficientes. Imagine que seu site tem um sistema de pagamentos que utiliza, PIX, cartão de crédito e débito e Pay Pal, você tem que desenvolver uma integração com cada um desses sistemas, mas para facilitar e centralizar principalmente a segurança você pode utilizar o gateway para agrupar todos os serviços de pagamentos reunindo eles em um único lugar. Além disso, vamos supor que esses diferentes sistemas de pagamento não utilizem a mesma língua, o Gateway pode realizar essa tradução e o site que consome através do Gateway não precisando falar várias línguas diferentes para cada uma das integrações de pagamento, pois elas estão agrupadas no Gateway que realiza essa tradução para uma única língua que o site possa falar.

O que é GIT?

O que é GIT?

Git é um sistema de controle de versão distribuído amplamente utilizado para gerenciar e rastrear mudanças em projetos de software. Criado por Linus Torvalds em 2005 para o desenvolvimento do kernel Linux, o Git se tornou uma ferramenta essencial no desenvolvimento de software devido à sua eficiência, flexibilidade e suporte robusto para colaboração em equipe.
Aqui estão alguns pontos principais sobre o Git:

1. **Controle de Versão**:
   - O Git rastreia e registra todas as mudanças feitas nos arquivos de um projeto ao longo do tempo. Isso permite que desenvolvedores revertam para versões anteriores, comparem diferentes versões e colaborem de forma eficiente.
2. **Distribuído**:
   - Git é um sistema distribuído, o que significa que cada desenvolvedor possui uma cópia completa do repositório, incluindo todo o histórico do projeto. Isso difere de sistemas de controle de versão centralizados, onde há um único servidor central.
3. **Ramos (Branches)**:
   - Git facilita a criação de ramos (branches), que são linhas independentes de desenvolvimento. Isso permite que desenvolvedores trabalhem em novas funcionalidades, correções de bugs ou experimentos de maneira isolada, sem afetar o código principal até que estejam prontos para mesclar suas mudanças.
4. **Mesclagem (Merge)**:
   - Quando as mudanças feitas em um ramo estão prontas para serem integradas ao código principal, elas podem ser mescladas. Git possui ferramentas poderosas para lidar com conflitos de mesclagem que podem surgir quando diferentes ramos têm alterações conflitantes.
5. **Commit**:
   - Um commit em Git é uma gravação de um conjunto de mudanças feitas nos arquivos. Cada commit tem um identificador único (hash), uma mensagem descritiva e pode incluir metadados como autor e data.
6. **Repositório (Repository)**:
   - Um repositório Git é onde o histórico do projeto é armazenado. Ele contém todos os commits, ramos e tags (pontos marcados no histórico, como versões de lançamento).
7. **Colaboração**:
   - Git facilita a colaboração em projetos de software. Ferramentas como GitHub, GitLab e Bitbucket oferecem plataformas para hospedar repositórios Git, permitindo que desenvolvedores contribuam, revisem código, relatem problemas e gerenciem projetos de maneira colaborativa.

O que é Github?

GitHub é uma plataforma de hospedagem de código-fonte e colaboração para desenvolvimento de software. Ele permite que desenvolvedores armazenem, gerenciem e compartilhem projetos de software usando o sistema de controle de versão Git. Além disso, o GitHub oferece recursos adicionais, como controle de acesso, rastreamento de problemas, integração contínua e hospedagem de páginas da web, que tornam o desenvolvimento de software colaborativo mais eficiente e organizado.

Principais Recursos do GitHub:

1. **Repositórios**: Os repositórios são usados para armazenar versões de código-fonte de um projeto, permitindo que os desenvolvedores contribuam e colaborem no desenvolvimento do software.
2. **Controle de Versão**: O GitHub usa o Git como sistema de controle de versão, o que permite que os desenvolvedores acompanhem as mudanças no código ao longo do tempo, revertam para versões anteriores e trabalhem em diferentes versões do código simultaneamente.
3. **Colaboração**: O GitHub facilita a colaboração entre desenvolvedores, permitindo que eles contribuam com código, revisem o trabalho um do outro e gerenciem tarefas e problemas do projeto.
4. **Segurança**: O GitHub oferece recursos de segurança, como autenticação de dois fatores, revisões de segurança automatizadas e alertas de segurança para vulnerabilidades conhecidas em dependências do projeto.
5. **Integração Contínua**: Os desenvolvedores podem configurar integração contínua (CI) usando ferramentas como GitHub Actions ou integração com outras plataformas de CI/CD para automatizar a construção, teste e implantação de seu código.