Semana 19 - Aula 1

Tópico Principal da Aula: Desenvolvimento e Execução de Testes Não Funcionais

Subtítulo/Tema Específico: Introdução aos Testes Não Funcionais e seu Ciclo de Vida

Código da aula: [SIS]ANO1C3B3S19A1

Objetivos da Aula:

- Compreender a importância dos testes não funcionais no ciclo de desenvolvimento de software.
- Conhecer os diferentes tipos de testes não funcionais e suas aplicações.
- Entender o processo de desenvolvimento e execução de testes não funcionais.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Introdução aos Testes Não Funcionais

- **Definição:** **Testes não funcionais avaliam os atributos de qualidade de um sistema de software, como desempenho, usabilidade, segurança e confiabilidade, em vez de sua funcionalidade específica. O objetivo principal é verificar se o produto atende às expectativas do usuário em termos de quão bem ele opera, otimizando-o antes do lançamento. Eles garantem que o software esteja em conformidade com os padrões da indústria e requisitos regulatórios.**
- **Aprofundamento/Complemento (se necessário):** Enquanto os testes funcionais verificam "o que" o software faz (suas funcionalidades), os testes não funcionais verificam "quão bem" ele faz (seus atributos de qualidade). Ambos são cruciais para a entrega de um software robusto e de alta qualidade. A automação de testes não funcionais é uma prática crescente, permitindo a execução rápida e eficiente e a detecção precoce de problemas, reduzindo custos e impactos em fases avançadas do projeto.
- **Exemplo Prático:** **Imagine um site de e-commerce. Um teste funcional garantiria que um item pode ser adicionado ao carrinho e que o processo de checkout funciona corretamente. Um teste não funcional, por outro lado, verificaria se o site permanece responsivo com 1.000 usuários simultâneos (desempenho), se as informações do cartão de crédito estão protegidas (segurança), se a navegação é intuitiva para o usuário (usabilidade), e se o sistema consegue se recuperar de uma falha de servidor (confiabilidade).**
- **Links de Vídeo:**
    - Para entender mais sobre testes funcionais e não funcionais: A diferença entre teste funcional e teste não funcional

    - Para entender mais sobre testes funcionais e não funcionais: https://youtu.be/BiMYmTybKMU?si=_zSy_dLmhTz1g0pu
    - Piramide de teste: https://youtu.be/hizne8Yc_Dg?si=HtFGfOeQ8RVaWmDD

Referência do Slide: Ciclo de Vida dos Testes Não Funcionais

- **Definição:** O ciclo de vida dos testes não funcionais geralmente envolve as seguintes fases: análise de requisitos de software, planejamento de testes, criação de casos de teste, configuração do ambiente de teste, execução de testes e repetição do ciclo para melhorias.
- **Aprofundamento/Complemento (se necessário):** Cada fase é interdependente e visa garantir que todos os aspectos não funcionais sejam adequadamente cobertos. A análise de requisitos é fundamental para definir os critérios de aceitação. A criação de casos de teste detalha como cada atributo será validado, e a execução simula condições reais para identificar gargalos ou falhas. A iteração permite refinar os testes e o software.
- ***Exemplo Prático:***
    - ***Análise de Requisitos: Para um aplicativo de streaming, definir que ele deve suportar 1 milhão de usuários simultâneos com tempo de carregamento de vídeo inferior a 2 segundos.***
    - ***Planejamento: Escolher JMeter para testes de carga e definir métricas como tempo de resposta e taxa de erros.***
    - ***Criação de Casos de Teste: Desenvolver scripts que simulem 1 milhão de usuários assistindo a diferentes vídeos simultaneamente.***
    - ***Configuração do Ambiente: Preparar um ambiente de teste que replique a infraestrutura de produção.***
    - ***Execução: Rodar os testes e coletar os dados de desempenho.***
    - ***Análise e Relato: Avaliar os resultados, identificar gargalos (por exemplo, no banco de dados) e gerar um relatório.***
    - ***Iterar e Melhorar: Se o objetivo não for alcançado, fazer ajustes no código ou infraestrutura e repetir o ciclo de teste.***

Referência do Slide: Ferramentas de Testes Não Funcionais

- **Definição:** Existem diversas ferramentas para auxiliar nos testes não funcionais, tanto de código aberto quanto comerciais, que ajudam a analisar a confiabilidade e a capacidade de resposta de um aplicativo sob variadas cargas de trabalho e cenários. **Exemplos incluem JMeter, LoadRunner, NeoLoad, Loadster e WebLoad Professional para desempenho.**
- **Aprofundamento/Complemento (se necessário):** A escolha da ferramenta depende do tipo de teste, do ambiente, do orçamento e da complexidade do sistema. **Ferramentas como JMeter são populares por serem de código aberto e versáteis para testes de carga. Ferramentas comerciais como LoadRunner oferecem recursos mais robustos para simulações em larga escala. Além disso, existem ferramentas para análise de código (SAST, DAST) para segurança, e plataformas de testes de usabilidade que registram interações do usuário.**

- **Exemplo Prático:** Uma equipe de desenvolvimento pode utilizar o Apache JMeter para simular 500.000 requisições simultâneas em uma API, avaliando o tempo de resposta e a taxa de erro para identificar gargalos de desempenho. Para segurança, pode-se usar ferramentas de varredura de vulnerabilidades para identificar falhas comuns antes que sejam exploradas.

Semana 19 - Aula 2

Tópico Principal da Aula: Tipos de Testes Não Funcionais: Desempenho e Segurança

Subtítulo/Tema Específico: Testes de Desempenho

Código da aula: [SIS]ANO1C3B3S19A2

Objetivos da Aula:

- Compreender os diferentes tipos de testes de desempenho e suas finalidades.
- Identificar métricas importantes para avaliar o desempenho de um software.
- Analisar cenários práticos de testes de desempenho.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Tipos de Testes de Desempenho

- **Definição:** T**estes de desempenho avaliam a velocidade, capacidade de resposta e estabilidade de um sistema sob uma carga de trabalho específica**. Incluem testes de carga (verificam o comportamento sob carga esperada), estresse (identificam o limite do sistema antes da falha), escalabilidade (capacidade de lidar com aumento de usuários/dados), resistência (comportamento sob carga contínua por um longo período) e concorrência (comportamento de jornadas com concorrência por recursos).
- **Aprofundamento/Complemento (se necessário):** O objetivo é identificar gargalos, determinar a capacidade máxima do sistema, verificar o tempo de carregamento e o tempo de resposta. Um atraso de apenas 100 milissegundos no carregamento de uma página pode impactar significativamente a taxa de conversão em e-commerce, por exemplo. Eles são essenciais para garantir uma boa experiência do usuário e evitar perdas de receita.
- **Exemplo Prático:**
    - **Teste de Carga:** Simular 5.000 usuários acessando um aplicativo de banco simultaneamente para garantir que as transações sejam processadas em menos de 3 segundos.

    - **Teste de Estresse:** Aumentar gradualmente o número de usuários em um servidor web até que ele comece a falhar, para determinar seu ponto de ruptura.
    - **Teste de Escalabilidade:** Adicionar mais servidores a uma aplicação para verificar se ela consegue suportar um crescimento de 20% no tráfego sem degradação do desempenho.
    -

Referência do Slide: Testes de Segurança

- **Definição:** Testes de segurança visam identificar vulnerabilidades e pontos fracos em um sistema de software que possam ser explorados por agentes mal-intencionados. Incluem testes de penetração (simulam ataques reais para encontrar brechas), testes de vulnerabilidade (identificam falhas sem necessariamente explorá-las), análise de código (revisão do código-fonte para falhas) e testes de segurança de rede.
- **Aprofundamento/Complemento (se necessário):** Esses testes buscam pôr à prova tecnologias de segurança como firewalls, autenticação, criptografia e técnicas de autorização. A detecção precoce de vulnerabilidades é crucial para proteger dados sensíveis, manter a integridade do sistema e evitar perdas financeiras ou de reputação. Testes de segurança podem ser realizados de forma manual ou automatizada, utilizando ferramentas específicas para cada tipo de avaliação.
- **Exemplo Prático:**
    - **Teste de Penetração:** Um profissional de segurança tenta explorar uma falha de "injeção SQL" em um formulário de login de um site para acessar o banco de dados e roubar informações de usuários.
    - **Teste de Vulnerabilidade:** Usar uma ferramenta automatizada para escanear um aplicativo web em busca de vulnerabilidades comuns, como "Cross-Site Scripting (XSS)" ou "Broken Authentication".
    - **Análise de Código (Code Review):** Revisar o código-fonte de um módulo de pagamento para garantir que as senhas não estão sendo armazenadas em texto simples.

Semana 19 - Aula 3

Tópico Principal da Aula: Tipos de Testes Não Funcionais: Usabilidade e Confiabilidade

Subtítulo/Tema Específico: Testes de Usabilidade

Código da aula: [SIS]ANO1C3B3S19A3

Objetivos da Aula:

- Entender o conceito de usabilidade e sua importância no design de software.
- Conhecer métodos e exemplos práticos para realizar testes de usabilidade.
- Avaliar a experiência do usuário e identificar pontos de melhoria.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Conceitos de Usabilidade e Testes

- **Definição:** Testes de usabilidade avaliam a facilidade com que os usuários podem interagir e utilizar um sistema de software para alcançar seus objetivos. O foco é na experiência do usuário, incluindo a intuitividade da interface, a eficiência na execução de tarefas, a acessibilidade e a compatibilidade com diferentes navegadores e dispositivos.
- **Aprofundamento/Complemento (se necessário):** Não se trata apenas de perguntar ao usuário se ele gostou, mas de observar seu comportamento real, identificar dificuldades e entender o porquê de certas ações. Métodos comuns incluem mapas de calor (para visualizar cliques e interações), entrevistas com usuários, testes com protótipos de papel e testes A/B. A escolha das palavras e a organização da informação na interface são cruciais para a usabilidade.
- **Exemplo Prático:**
    - Observar um usuário tentando encontrar a fatura de um mês anterior em um aplicativo bancário, notando onde ele clica, se ele se confunde com ícones ou nomenclaturas, e quanto tempo leva para completar a tarefa.
    - Utilizar um mapa de calor para identificar as áreas mais clicadas de uma página web e otimizar o design com base nesses dados.

    - Pedir a um grupo de usuários que complete uma série de tarefas específicas em um novo site, como "adicionar 3 itens ao carrinho e finalizar a compra", e registrar quaisquer problemas ou hesitações.
- **Links de Vídeo:**
    - Como realizar um teste de usabilidade remoto?: Como realizar um teste de usabilidade remoto? #AluraMais - YouTube
    - Fazendo testes de usabilidade com protótipos de papel: Vídeo: fazendo testes de usabilidade com protótipos de papel - UX Collective

Referência do Slide: Testes de Confiabilidade

- **Definição:** Testes de confiabilidade avaliam a capacidade de um sistema de software de operar sem falhas por um período de tempo especificado e sob condições definidas. Isso inclui a verificação da recuperação de desastres (quão bem o sistema se recupera de falhas), tolerância a falhas (capacidade de continuar operando mesmo com componentes falhos) e a estabilidade geral do sistema.
- **Aprofundamento/Complemento (se necessário):** A engenharia do caos é uma prática avançada que injeta deliberadamente falhas em um sistema em produção para testar sua resiliência e identificar pontos fracos antes que eles causem interrupções reais. O objetivo é construir sistemas mais imunes a falhas e melhorar a capacidade de recuperação. Métodos como "split-half reliability" podem ser usados para estimar a consistência interna de instrumentos de medição.
- **Exemplo Prático:**
    - **Recuperação de Desastres:** Desligar um servidor de banco de dados principal em um ambiente de teste para verificar se o sistema de failover (servidor de backup) assume o controle automaticamente e sem perda de dados.
    - **Tolerância a Falhas:** Desconectar um dos servidores em um cluster para observar se a aplicação continua funcionando normalmente com os servidores restantes.
    - **Engenharia do Caos:** Em um ambiente de produção controlado, simular a perda de conectividade de rede para uma pequena porcentagem de servidores para observar como o sistema reage e se recupera.
- **Links de Vídeo:**
    - Confiabilidade de Software: Esse é o SEGREDO da área de TESTES e Qualidade de Software - YouTube
    - Testes e Qualidade de Software: Many | Testes e Qualidade de Software - YouTube