PESQUISA:

Teste de back-end.

- **Nomes: Ana Júlia, Izabelle e Patrícia Gabriela.**
- **Tópicos:**
- **Conceitos:**

  Teste de codificação
- **Etapas:**

  Planejamento de testes

  Implementação de testes

  Execução de testes

  Análise de resultado dos testes

SENAI / SESI

CONCEITOS

O que é o teste Server-Side?

O teste do lado do servidor é um método de teste A/B em que as variações de um teste são renderizadas diretamente do servidor da Web e, em seguida, enviadas premeditadamente para o dispositivo dos visitantes. A implementação diretamente no servidor permite que você execute testes mais sofisticados que, de outra forma, poderiam dificultar a experiência do usuário se implementados no lado do cliente. Além disso, o teste do lado do servidor também é optado nos casos em que é simplesmente inviável experimentar no lado do cliente.

O teste do lado do servidor funciona assim: em vez de o site fazer mudanças direto no navegador da pessoa, ele já manda a versão que quer testar pronta do próprio servidor. Ou seja, quando alguém acessa o site, já recebe uma das versões que está sendo testada. Isso é bom porque permite testar mudanças mais complexas, sem deixar o site pesado ou travando. Esse tipo de teste também é usado quando não dá para fazer a mudança direto no navegador, por limitações técnicas.

Existem várias linguagens de programação que podem ser utilizadas para desenvolver o lado do servidor de uma aplicação web. Alguns exemplos populares incluem PHP, Python, Ruby, Java e C#. Cada linguagem tem suas próprias características e vantagens, e a escolha da linguagem depende das necessidades e preferências do desenvolvedor.

O server-side oferece várias vantagens no desenvolvimento web. Uma das principais vantagens é a segurança. Como o código server-side é executado no servidor, ele não é visível para os usuários finais. Isso significa que informações sensíveis, como senhas e chaves de acesso a banco de dados, podem ser mantidas em segurança. Além disso, o server-side permite o controle total sobre a lógica de negócios e a manipulação de dados, o que facilita a criação de aplicações complexas e escaláveis.

Outra vantagem do server-side é a capacidade de escalar facilmente uma aplicação web. À medida que o número de usuários e a demanda por recursos aumentam, é possível adicionar mais servidores para lidar com o aumento de tráfego. Isso é conhecido como escalabilidade horizontal. Além disso, o server-side também permite o uso de técnicas como cache de dados e balanceamento de carga para melhorar o desempenho e a disponibilidade da aplicação.

Embora existam muitas qualidades, há desvantagens da validação do lado do servidor. Um dos principais problemas é que ela pode resultar em uma experiência de usuário menos fluida, pois os usuários precisam esperar por respostas do servidor após o envio de dados. Isso pode ser mitigado com o uso de validação do lado do cliente em conjunto, permitindo que os usuários recebam feedback imediato antes de enviar o formulário.

Um exemplo comum de validação do lado do servidor é a verificação de um endereço de e-mail. O servidor pode usar expressões regulares para garantir que o formato do e-mail esteja correto antes de armazená-lo no banco de dados. Outro exemplo é a validação de senhas, onde o servidor pode verificar se a senha atende a critérios de segurança, como comprimento mínimo e a inclusão de caracteres especiais. Essas validações ajudam a garantir que os dados armazenados sejam válidos e seguros.

Em resumo, o server-side desempenha um papel fundamental no desenvolvimento web, sendo responsável por processar as solicitações dos usuários, fornecer respostas adequadas, garantir a segurança das aplicações e facilitar a manutenção e escalabilidade. É essencial entender o conceito de server-side para criar aplicações web eficientes e seguras.

Fontes usadas para pesquisa:

https://alsweb.com.br/glossario/o-que-e-server-side-lado-do-servidor/
https://vwo.com/br/teste-server-side/
https://lbodev.com.br/glossario/o-que-e-server-side-validation/

Planejamento dos Testes

Nesta etapa procura-se entender, inicialmente:

- Metas e objetivos do projeto e do cliente;
- Riscos do projeto;
- Escopo do trabalho: (o que testar? como testar?).

Os principais objetivos da etapa de planejamento são verificar a missão, definir os objetivos e as atividades de teste a serem realizadas.

É durante essa etapa que se cria o Plano de Teste, um documento que descreve o escopo, abordagem, recursos e cronograma das atividades de teste. Nele, estão documentadas as exceções quanto à abordagem do teste, recursos a serem utilizados, equipe envolvida e as técnicas a serem aplicadas.

Durante do planejamento do teste, é importante ter em mente de maneira clara o propósito dessa etapa, que visa definir:

- Os objetivos do teste;
- O escopo;
- As abordagens (técnicas, cobertura, pessoas, hardware, software, etc.)
- Os recursos;
- As políticas de teste;
- O cronograma:
    - Agendamento das atividades de análise e modelagem;
    - Agendamento das atividades de implementação, execução e avaliação.
- Os critérios de saída.

  Fonte: https://blog.onedaytesting.com.br/o-processo-de-testes-de-software-simplificado/

Implementação dos Testes

Durante a etapa de Implementação do teste, procura-se realizar a especificação dos procedimentos e/ou scripts de teste através da combinação de casos de teste em ordem particular.

Ou seja: é hora de transformar condições de teste em casos e procedimentos de teste. Casos de teste lógicos devem ser transformados em casos de teste concretos, que serão utilizados posteriormente para a execução.

Principais atividades envolvidas na Implementação do teste:

- Identificar, desenvolver, finalizar, implementar e priorizar Casos de Teste;
- Desenvolver e priorizar procedimentos de teste;
- Criar dados de teste;
- Preparar equipamento de teste;
- Automatizar casos de teste;
- Implementar e verificar o ambiente de testes corretamente;
- Verificar e atualizar a rastreabilidade bidirecional entre a base de teste e os casos de teste.

Dentro desse contexto, é muito importante ter em mente de forma clara as definições e diferenças entre erros, defeitos e falhas, visando a realização de reports e análises mais precisas:

Erros: Envolvem a ação humana e se manifestam a qualquer momento do desenvolvimento do software.

Defeitos: A manifestação de erros da ação humana, aquilo que usualmente chamamos de Bug.

Falhas: Diferenças indesejadas entre aquilo que se observa e o que é esperado.

Como funciona a execução de testes de server-side:

1. A instalação e integração do SDK:
Para executar testes server-side, é necessário instalar e integrar um Software Development Kit (SDK) nos sistemas back-end. O SDK atua como uma ponte entre o servidor do cliente e a ferramenta de teste, permitindo que a ferramenta receba informações e realize ações no servidor.

2. Chamada ao servidor:
Sempre que o código com o SDK é executado, é feita uma chamada ao servidor da ferramenta de teste. Essa chamada geralmente é feita apenas uma vez, com os dados armazenados em cache para execuções subsequentes.

3. Informações da campanha:
O SDK busca informações da campanha no servidor da ferramenta, como o número de variações, metas e a distribuição do tráfego.

4. Atribuição e rastreamento:
Uma vez que a variação é atribuída ou a meta é rastreada no lado do servidor, os dados são enviados para a ferramenta de teste para fins de rastreamento.

Análise de resultados dos testes:

1. Coleta e organização dos dados:
Os resultados dos testes, como logs, métricas de performance e dados de transações, são coletados e organizados para facilitar a análise.

2. Análise dos resultados:
Performance: Avaliação de métricas como tempo de resposta, taxa de transferência, uso de recursos (CPU, memória), e outros indicadores relevantes.

Funcionalidade: Verificação de erros, falhas, comportamento inesperado, e cumprimento das especificações funcionais.
Segurança: Identificação de vulnerabilidades e falhas de segurança.

3. Identificação de problemas:
Análise dos resultados para identificar problemas, gargalos de desempenho, e pontos que precisam ser corrigidos.

4. Ações corretivas:
Desenvolvimento de soluções para os problemas identificados, como ajustes no código, otimizações de infraestrutura, ou correção de bugs.

5. Monitoramento contínuo:
Implementação de um sistema de monitoramento para acompanhar a performance e funcionalidade da aplicação em produção.

Ferramentas e tecnologias para análise:

Ferramentas de logging: Para coletar e analisar logs de erros e eventos.
Ferramentas de monitoramento de performance: Para acompanhar o desempenho do servidor e da aplicação.
Ferramentas de teste de segurança: Para identificar vulnerabilidades e falhas de segurança.
Linguagens de programação: Para analisar o código e identificar problemas.

Fonte:

https://blog.onedaytesting.com.br/o-processo-de-testes-de-software-simplificado/