Plano de Teste Completo para o Site da Microsoft

Este documento descreve um plano de teste abrangente para o site da Microsoft (https://www.microsoft.com/pt-br/), cobrindo diversos aspectos do processo de teste, incluindo testes funcionais, de automação, performance, segurança, acessibilidade e regressão.

1. Plano de Teste Funcional (PDF)

1.1. Casos de Teste:

O plano de teste funcional incluirá casos de teste detalhados para validar todas as funcionalidades do site, incluindo:

* **Página Inicial:**
    * Verificar se a página inicial está carregando corretamente, com todos os elementos visíveis.
    * Validar a navegação para outras seções do site, como produtos, soluções, suporte e sobre a Microsoft.
    * Testar a funcionalidade do menu principal, menu dropdown e links para redes sociais.
    * Validar o conteúdo da página inicial e a presença de links para informações relevantes.
    * Testar o formulário de pesquisa.
* **Página de Produtos:**
    * Verificar a navegação para a página de produtos e a exibição correta dos produtos.
    * Validar a filtragem e ordenação dos produtos.
    * Testar o detalhamento de cada produto, incluindo informações, recursos e especificações.
    * Validar a adição de produtos ao carrinho de compras.
* **Página de Soluções:**
    * Verificar a navegação para a página de soluções e a exibição correta das soluções.
    * Validar a filtragem e ordenação das soluções.
    * Testar o detalhamento de cada solução, incluindo informações, recursos e casos de uso.
    * Validar a navegação para outras páginas relevantes, como produtos relacionados.
* **Página de Suporte:**
    * Verificar a navegação para a página de suporte e a exibição correta das opções de suporte.
    * Validar a pesquisa por soluções e a busca por artigos e FAQs.
    * Testar o contato com o suporte através do chat, telefone ou formulário.
    * Validar a funcionalidade do fórum de suporte.
* **Página de Sobre a Microsoft:**
    * Verificar a navegação para a página de sobre a Microsoft e a exibição correta das informações sobre a empresa.
    * Validar a navegação para outras páginas relevantes, como história, valores e notícias.

    * Testar a funcionalidade da área de contato para envio de mensagens.
* **Funcionalidades Específicas:**
    * Login e logout do site.
    * Registro de novo usuário.
    * Alteração de dados do usuário.
    * Recuperação de senha.
    * Criação e edição de perfil.
    * Adição de produtos ao carrinho.
    * Compra de produtos.
    * Rastreamento de pedidos.
    * Contato com suporte.
    * Inscrição em newsletters.
    * Compartilhamento de conteúdo nas redes sociais.

1.2. Critérios de Aceitação:

* Todas as páginas e funcionalidades devem carregar corretamente e sem erros.
* Os links devem funcionar corretamente e redirecionar para as páginas corretas.
* Os formulários devem funcionar corretamente, validando os dados e submetendo as informações.
* O conteúdo deve estar atualizado e livre de erros gramaticais.
* O site deve ser responsivo e adaptável a diferentes dispositivos (desktop, mobile, tablet).
* O site deve ter tempos de carregamento rápidos e fluidez na navegação.
* O site deve ser amigável e fácil de usar.

1.3. Escopo do Teste:

O escopo do teste inclui todas as páginas e funcionalidades principais do site da Microsoft. Serão testados os principais fluxos de usuários, incluindo:

* Navegação no site.
* Pesquisa de produtos e soluções.
* Compra de produtos.
* Acesso ao suporte.
* Contato com a Microsoft.

1.4. Ambientes de Teste:

Os testes serão realizados nos seguintes ambientes:

* Navegadores: Chrome, Firefox, Safari, Edge.
* Dispositivos Móveis: Android e iOS.
* Diferentes resoluções de tela: desktop, tablet, mobile.
* Diferentes versões do sistema operacional: Windows, macOS, Linux.

1.5. Riscos e Dependências:

* Riscos: Dependência de APIs externas, integração com sistemas de terceiros.
* Dependências: Acesso a dados de teste, ambiente de teste dedicado, equipe de desenvolvimento disponível para suporte.

1.6. Métricas de Sucesso:

* Taxa de defeitos: número de defeitos encontrados durante o teste.
* Cobertura de teste: porcentagem de funcionalidades e fluxos de usuários testados.
* Tempo de execução: tempo total gasto na execução dos testes.
* Tempo de resposta do site: tempo que o site leva para carregar as páginas.

2. Projeto de Automação em Cypress

2.1. Estrutura do Projeto:

```
codigo_automacao/
% % %  cypress/
%    % % %  integration/
%    %    % % %  login.spec.js
%    %    % % %  search.spec.js
%    %    % % %  cart.spec.js
%    % % %  support/
%    %    % % %  commands.js
%    %    % % %  e2e.js
%    % % %  plugins/
%    %    % % %  index.js
%    % % %  fixtures/
%    %    % % %  user.json
%    % % %  cypress.config.js
%    % % %  e2e/
%       % % %  support/
%       %    % % %  e2e.js
%       % % %  integration/
%          % % %  home.spec.js
%          % % %  products.spec.js
% % %  .env
% % %  package.json

```

2.2. Código Cypress:

* **Navegação:** Testar a navegação entre as páginas principais do site.

* **Busca:** Validar a funcionalidade da busca no site, incluindo a pesquisa por produtos, soluções e informações.
* **Login e Logout:** Testar o processo de login e logout do site, incluindo validação de credenciais, mensagens de erro e direcionamento para áreas restritas.
* **Submissão de Formulários:** Validar a submissão de formulários, incluindo validação de campos, mensagens de erro e direcionamento para páginas de sucesso.
* **Interações com Elementos Dinâmicos:** Testar a interação com elementos dinâmicos da página, como menus dropdown, carrosséis e modais.

2.3. Cobertura de Teste:

* Casos de borda: Testar cenários extremos, como campos vazios, campos com caracteres inválidos, números fora do intervalo permitido.
* Casos de erro: Testar cenários que podem gerar erros, como tentativas de login com credenciais inválidas, envio de formulários com dados incompletos, tentativas de acessar áreas restritas sem login.
* Cenários com dados dinâmicos: Testar cenários que envolvem dados dinâmicos, como a busca por produtos, a atualização de carrinhos de compras e a exibição de informações personalizadas.

2.4. Integração com Allure:

* Instalar o plugin Allure para Cypress.
* Configurar o Cypress para gerar relatórios Allure.
* Executar os testes e gerar os relatórios Allure.

2.5. Configuração de Variáveis de Ambiente:

* Criar um arquivo .env com as variáveis de ambiente para cada ambiente (desenvolvimento, homologação, produção).
* Configurar o Cypress para carregar as variáveis de ambiente.

3. Instruções de Execução (README.md)

3.1. Configuração do Ambiente:

* Instalar Node.js e npm.
* Instalar o Cypress: `npm install cypress --save-dev`.
* Abrir o Cypress: `npx cypress open`.
* Configurar as variáveis de ambiente no arquivo .env.
* Instalar o plugin Allure para Cypress.

3.2. Execução dos Testes:

* Executar os testes: `npx cypress run --env environment=staging`.
* Gerar relatórios Allure: `npx cypress run --env environment=staging --reporter

cypress-allure-reporter`.

3.3. Interpretação dos Resultados:

* Visualizar os resultados no terminal.
* Visualizar os resultados no dashboard do Cypress.
* Visualizar os relatórios Allure.

3.4. Configuração de CI/CD:

* Integrar os testes Cypress em uma pipeline CI/CD usando GitHub Actions ou Jenkins.
* Executar os testes em cada build ou pull request.
* Gerar relatórios Allure e enviar para um dashboard de testes.

4. Plano de Testes de Performance (PDF)

* **Testes de Carga:**
    * Usar ferramentas como JMeter ou k6 para simular uma grande quantidade de usuários acessando o site simultaneamente.
    * Definir cenários de carga variando o número de usuários e a taxa de requisições por segundo.
    * Monitorar o tempo de resposta, o throughput e a taxa de erros.
    * Ajustar a carga para identificar o ponto de saturação do site.
* **Testes de Estresse:**
    * Aumentar a carga gradualmente até atingir o ponto de falha do site.
    * Identificar os gargalos do sistema e as áreas mais suscetíveis a falhas.
    * Validar a capacidade do site de se recuperar após um pico de carga.
* **Testes de Capacidade:**
    * Determinar a capacidade máxima do site para lidar com um determinado número de usuários simultâneos.
    * Avaliar a degradação da performance ao aumentar a carga.
    * Identificar os recursos do sistema que precisam ser dimensionados para atender à demanda.

4.1. Métricas de Performance:

* Tempo de resposta: tempo que o site leva para responder a uma requisição.
* Throughput: número de requisições processadas por segundo.
* Taxa de erros: porcentagem de requisições que resultam em erros.

5. Plano de Testes de Segurança (PDF)

* **Testes de Vulnerabilidade:**
    * Usar ferramentas como Burp Suite, ZAP ou SQLMap para identificar vulnerabilidades comuns, como injeção de SQL, XSS, CSRF, etc.
    * Validar a implementação de medidas de segurança, como validação de entrada,

codificação de saída e controle de acesso.
* **Teste de Autenticação/Autorização:**
    * Testar o processo de autenticação, incluindo validação de credenciais, mensagens de erro e mecanismos de autenticação multifator.
    * Testar o processo de autorização, incluindo a validação de permissões de acesso a recursos e funcionalidades do site.
* **Teste de Penetração:**
    * Contratar uma empresa especializada em testes de penetração para simular ataques maliciosos e avaliar a resistência do site.
    * Validar a capacidade do site de detectar e prevenir ataques, incluindo a identificação de vulnerabilidades, a proteção de dados confidenciais e a recuperação de incidentes de segurança.

6. Plano de Testes de Acessibilidade (PDF)

* **Conformidade com WCAG:**
    * Testar o site contra as diretrizes WCAG (Web Content Accessibility Guidelines) para garantir a acessibilidade para todos os usuários.
    * Usar ferramentas como Axe, Lighthouse e NVDA para realizar testes de acessibilidade.
    * Validar a conformidade do site com as normas WCAG.
* **Ferramentas de Acessibilidade:**
    * Usar ferramentas como Axe, Lighthouse, NVDA e outros recursos de teste de acessibilidade para identificar problemas de acessibilidade.
    * Implementar as correções necessárias para garantir a acessibilidade do site.
* **Casos de Teste Acessíveis:**
    * Criar casos de teste específicos para validar a navegação com teclado, o uso de leitores de tela, a acessibilidade para pessoas com deficiências visuais e motoras, entre outros aspectos.
    * Testar os recursos de acessibilidade do site para garantir que todos os usuários possam acessar o site de forma eficaz.

7. Estratégia de Teste de Regressão

* **Seleção de Casos de Teste:**
    * Identificar os casos de teste críticos que devem ser incluídos no conjunto de regressão.
    * Priorizar os casos de teste com base na frequência de uso, na importância da funcionalidade, na probabilidade de erros e no impacto de erros.
* **Automação de Regressão:**
    * Criar scripts de regressão automatizados que serão executados em cada build.
    * Usar ferramentas como Cypress, Selenium ou outros frameworks de automação para criar os scripts de teste.
* **Planejamento de Execução:**
    * Definir a frequência de execução dos testes de regressão.
    * Executar os testes de regressão diariamente, em cada pull request, ou em outros

intervalos, dependendo do risco e da frequência de alterações no site.

8. Documentação de Integração de APIs

* **Testes de API Automatizados:**
    * Criar casos de teste para validar as respostas de APIs, usando ferramentas como Postman ou Newman.
    * Testar a funcionalidade da API, incluindo a validação de status code, mensagens de erro, headers e body da resposta.
    * Usar mockups para simular o comportamento de serviços externos e garantir a independência dos testes de API.
* **Testes de Contrato:**
    * Definir como garantir que as APIs respeitam os contratos esperados.
    * Usar ferramentas como Pact para definir e validar os contratos das APIs.
    * Testar a conformidade das APIs com os contratos definidos.
* **Testes de Carga em APIs:**
    * Detalhe como realizar testes de carga especificamente para as APIs.
    * Usar ferramentas como k6 para simular uma grande quantidade de requisições para a API.
    * Validar a performance da API sob alta carga, monitorando o tempo de resposta, o throughput e a taxa de erros.

Organização dos Artefatos

* Todos os arquivos e pastas devem ser organizados de forma clara e acessível.
* Todos os relatórios e artefatos devem ser armazenados em locais facilmente acessíveis para revisão posterior.
* É recomendado usar uma plataforma de gerenciamento de código-fonte (como GitHub) para armazenar e compartilhar os artefatos de teste.

Conclusão

Este plano de teste abrangente fornece uma estrutura completa para garantir a qualidade do site da Microsoft. A implementação deste plano permitirá a identificação e correção de defeitos, a otimização da performance, a proteção da segurança, a garantia da acessibilidade e a manutenção da qualidade do site a longo prazo.