| Estratégia de Testes |
| --- |
| 1. **Introdução** |
| **Objetivo:**  Este documento descreve a estratégia de testes para o sistema "Infojobs", garantindo que as atividades de teste sejam bem organizadas, eficientes e eficazes, alinhadas com as práticas recomendadas pela ISO 29119-3. |
| **Escopo:**  A estratégia cobre todos os tipos de teste aplicáveis ao Infojobs, incluindo testes funcionais, não funcionais, de integração, e aceitação, em diferentes plataformas e dispositivos. |
| 1. Abordagem de Teste |
| **Metodologia de Teste:**  Ágil: Os testes serão integrados no ciclo de desenvolvimento ágil, com execução contínua de testes automatizados e manuais durante cada sprint.  Cascata: Testes específicos, como testes de desempenho e segurança, serão conduzidos em fases mais avançadas do projeto, onde o sistema estará mais estabilizado. |
| **Níveis de Teste:**  Teste de Unidade: Executado pelos desenvolvedores para verificar a funcionalidade de componentes individuais.  Teste de Integração: Focado na interação entre diferentes módulos, como o cadastro de usuários e a aplicação de vagas.  Teste de Sistema: Avaliação completa do sistema em ambientes simulando produção, cobrindo tanto funcionalidades principais quanto secundárias.  Teste de Aceitação: Conduzido com usuários finais para validar se o sistema atende às necessidades do negócio. |
| **Tipos de Teste:**   * **Funcional:** Verificação das funcionalidades do sistema conforme especificado. * **Regressão:** Execução de testes repetidos para garantir que mudanças não introduzam novos defeitos. * **Desempenho:** Medição da resposta e estabilidade do sistema sob carga variada. * **Segurança:** Teste de vulnerabilidades para garantir a proteção de dados do usuário. * **Usabilidade:** Avaliação da interface do usuário para assegurar que o sistema é intuitivo e fácil de usar. |
| 1. Ferramentas de Teste |
| Automação:  Selenium: Usado para testes funcionais automatizados no front-end.  JUnit: Utilizado para testes de unidade automatizados.  Desempenho:  JMeter: Ferramenta para testes de carga e estresse.  Gerenciamento de Testes:  JIRA: Usado para rastreamento de defeitos e gerenciamento de sprints.  Segurança:  OWASP ZAP: Ferramenta para análise de vulnerabilidades de segurança. |
| 1. Critérios de Teste |
| **Critérios de Sucesso:**  Funcionalidade: Todas as funcionalidades principais e críticas devem passar com sucesso em todos os casos de teste definidos.  Desempenho: O tempo de resposta para operações-chave deve ser inferior a 2 segundos para 95% das solicitações.  Segurança: Nenhuma vulnerabilidade crítica deve ser detectada em testes de segurança. |
| **Critérios de Falha:**  Funcionalidade: Qualquer falha em casos de teste críticos será considerada um bloqueador.  Desempenho: Tempos de resposta superiores a 3 segundos em mais de 10% das solicitações durante testes de carga serão consideradas falhas.  Segurança: A detecção de qualquer vulnerabilidade crítica que possa comprometer dados sensíveis resultará em falha. |
| 1. Cobertura de Teste |
| **Cobertura Funcional:** O objetivo é alcançar uma cobertura de 100% dos casos de uso críticos definidos no Documento de Requisitos de Software.  **Cobertura de Código:** Utilização de ferramentas de cobertura de código, como SonarQube, para garantir que pelo menos 85% do código seja coberto por testes automatizados. |
| 1. Gestão de Riscos |
| **Identificação de Riscos:**  Risco: Atraso na entrega de funcionalidades críticas.  Mitigação: Priorização de testes automatizados e integração contínua para identificar defeitos o mais cedo possível.  Risco: Falha de desempenho em ambientes de produção.  Mitigação: Realização de testes de carga e estresse antecipadamente em ambientes que simulam produção. |
| 1. Papéis e Responsabilidades |
| Test Manager: Responsável por planejar e coordenar todas as atividades de teste.  Test Lead: Coordena as equipes de teste e revisa os resultados dos testes.  QA Engineers: Executam os testes, reportam defeitos e garantem a conformidade com os critérios de aceitação.  Desenvolvedores: Suportam a automação de testes e corrigem defeitos identificados. |
| 1. Cronograma de Testes |
| Sprint 1: Configuração inicial e testes de unidade.  Sprint 2: Testes funcionais e de integração das funcionalidades principais.  Sprint 3: Testes de sistema completos e início dos testes de desempenho.  Sprint 4: Testes de segurança, regressão e aceitação pelo usuário final. |
| 1. Comunicação e Relatórios |
| **Relatórios Diários:** Status dos testes e defeitos identificados serão reportados diariamente via JIRA.  **Reuniões de Revisão de Sprint:** Revisão dos resultados dos testes ao final de cada sprint, com foco em bloqueadores e riscos identificados.  **Relatório Final:** Um relatório detalhado será emitido ao final do ciclo de testes, resumindo os resultados e as lições aprendidas. |
| 1. Processo de Melhoria Contínua |
| **Revisão Pós-Projeto:** Ao término do projeto, será realizada uma sessão de retrospectiva para identificar melhorias para projetos futuros.  **Ajuste da Estratégia:** A estratégia de teste será revisada e ajustada conforme necessário ao longo do projeto, com base nos feedbacks das equipes e nos resultados dos testes. |