Plano de Teste: Sistema de Agendamento de Serviços Acadêmicos

Versão 1.0

2024

Sumário

1. Plano de teste no ASA..... 3
2. Resumo..... 3
3. Pessoas envolvidas..... 3
4. Local e ferramentas de testes..... 3
4.1. Ferramentas de testes..... 4
4.2. Ambiente de testes..... 4
5. Recursos necessários..... 4
6. Critérios usados..... 5
7. Riscos..... 5
8. Metodologia de teste..... 5
8.1. Etapas de Teste..... 6
8.2. Triagem de Bugs..... 8
8.3. Conclusão do Teste..... 9
9. Resultados de teste..... 9
10. Cronograma..... 9

1. Plano de teste no ASA

Este documento tem como objetivo descrever as atividades e planejamento para a execução dos testes no ASA - Sistema de Agendamento de Serviços Acadêmicos.

2. Resumo

No processo de Verificação e Validação de software, a atividade de testes consiste em ações de garantia de qualidade de software por meio da descoberta de erros. A partir da documentação criada para o desenvolvimento do Sistema de Agendamento de Serviços Acadêmicos, pretende-se iniciar o planejamento e execução dos testes com base nos requisitos funcionais e não funcionais previamente definidos. Para tal, o referido documento engloba a organização, objetivos, funções e responsabilidades para o gerenciamento de testes, abrangendo a metodologia e resultados obtidos a partir do que foi estabelecido.

3. Pessoas envolvidas

Devido a familiaridade com a prática de testes, e por não atuarem diretamente no desenvolvimento do sistema, a responsabilidade para a realização dos testes foi incubida aos seguintes testadores:

- ERIKY ABREU VELOSO
- GEISA MORAIS GABRIEL
- LIVIA BEATRIZ MAIA DE LIMA

4. Local e ferramentas de testes

Para a execução dos testes serão utilizadas ferramentas de apoio, que atuem em conformidade com o ambiente de desenvolvimento e a linguagem de programação empregada. Os subtópicos abaixo detalham o aspecto das tecnologias e local de testes a serem utilizados.

4.1. Ferramentas de testes

Ferramenta	Descrição da ferramenta
NUnit	Framework para realização de testes unitários.
SonarLint	Detector de problemas de qualidade do código-fonte.
Selenium	Framework portátil para testar aplicativos web.
Lighthouse	Ferramenta automatizada de código aberto para medir a qualidade de páginas da web.

4.2. Ambiente de testes

Máquina	Processador	Armazenamento	Sistema Operacional
Dell G15,	Intel Core I5 11 geração	512 GB SSD e 8 GB de RAM	Windows 11
Lenovo IdeaPad S145-15IIL	Intel Core I5 10 geração	256 GB SSD e 8 GB de RAM	Windows 11
Acer Nitro 5	AMD Ryzen 7 4800H	1 TB de HD e 256 SSD e 16 GB de RAM	Windows 10

5. Recursos necessários

Para a efetivação dos testes alguns recursos devem ser estabelecidos. Nesse sentido, os recursos foram divididos conforme o expresso na lista abaixo:

- **Pessoal (testadores):** os testes devem ser realizados por um pessoal que detenha, pelo menos, o mínimo de familiaridade possível com a aplicação e interpretação do funcionamento de testes;
- **Máquinas:** deve haver, pelo menos, uma máquina para cada testador;
- **Configuração do ambiente:** o ambiente para a realização dos testes deve ser configurado em cada máquina;
- **Documentação do sistema:** os testadores devem possuir acesso à documentação completa do sistema;

- **Ferramentas de teste:** as ferramentas de testes devem ser estabelecidas conforme o propósito para cada tipo de teste planejado.

6. Critérios usados

Para a modelagem dos testes definem-se os seguintes critérios:

- **Quantidade total de testes:** conforme aspectos definidos na documentação do sistema ASA, é pretendido a realização de, no mínimo, 120 testes no total. Cada testador, deve, portanto, analisar e produzir, no mínimo, 40 testes;
- **Quantidade mínima de casos de testes a serem realizados:** com base na quantidade de funcionalidades do sistema, tendo em vista casos válidos e inválidos, espera-se que sejam realizados, no mínimo, 30 casos de teste;
- **Critérios quantitativos de teste:** para testes estruturais a análise será feita com base na cobertura de linhas de código. Assim, procura-se obter uma quantidade de, no mínimo, 80% de cobertura total;
- **Critérios qualitativos de teste:** espera-se identificar no mínimo 15 problemas no que diz respeito à qualidade do código-fonte, a partir do uso de ferramentas de análise estática.

7. Riscos

Possíveis contingências para a efetivação do plano de teste criado:

- Incompatibilidade da ferramenta de teste com a linguagem ou IDE utilizada;
- Falta de conexão com a internet via rede wi-fi;
- Indisponibilidade de maquinário necessário.

8. Metodologia de teste

A abordagem de teste escolhida visa a completude do propósito dos testes a serem realizados. Dessa forma, para este projeto, será adotado o desenvolvimento incremental, o que permite a realização dos testes em fases sucessivas, conforme o

software é desenvolvido. Essa abordagem permite o emprego de verificações e validações contínuas até a data de entrega do projeto.

Durante o ciclo de desenvolvimento, serão realizadas técnicas de testes estruturais e funcionais para a cobertura das atividades de verificação e validação do software. Para tal, os testes serão divididos em três níveis principais, são eles: testes de unidade, integração e sistema. Outros tipos de teste também serão alocados no processo incremental, englobado e estabelecido nas etapas para teste.

8.1. Etapas de Teste

Para o gerenciamento e concretização dos testes, as funcionalidades e módulos a serem testados devem atuar conforme o estabelecido na documentação do sistema. Os módulos e funcionalidades a serem testadas atuam segundo o tipo de teste empregado em cada etapa estabelecida.

- **Teste de Unidade:** Avalia a menor unidade de código para garantir que funcione isoladamente.
    - Objetivo do teste: assegurar que as pequenas unidades do sistema funcionem como esperado.
    - Técnica: Chamar cada função e método implementado para verificar se condiz com as saídas esperadas.
    - Critério de aceitação: 80% de cobertura sobre o arquivo equivalente ao tipo de teste empregado.
    - Ferramenta: NUnit.

- **Teste de Integração:** Avalia a interação entre diferentes componentes do software para garantir que eles funcionem juntos.
    - Objetivo do teste: assegurar que os métodos de acesso ao Banco de Dados funcionem corretamente e correspondam ao mundo real.
    - Técnica: Realizar casos de teste de sucesso e de fracasso.
    - Critério de aceitação: 80% de cobertura sobre o arquivo equivalente ao tipo de teste empregado.

    - Ferramenta: NUnit.

- **Teste de Sistema:** Avalia o sistema na totalidade, validando se ele atende aos requisitos especificados, envolve simulações de uso real.
    - Objetivo do teste: assegurar que a navegação do sistema, bem como suas funcionalidades, estejam condizentes com o padrão estruturado.
    - Técnica: Criar testes em cada tela criada, lidando com cada campo, ação e objeto na qual possa ter interação.
    - Critério de aceitação: 80% de cobertura de funcionalidade lidando com a documentação do sistema.
    - Ferramentas: NUnit e Selenium.

- **Teste de Responsividade:** Avalia o sistema no quesito de adaptação em diferentes plataformas de uso.
    - Objetivo do teste: assegurar que a adaptação de tela não será prejudicada caso mude o tamanho do dispositivo.
    - Técnica: Assegurar que desktop, tablet e mobile terão a mesma experiência do usuário, diante da adaptação das telas.
    - Critério de aceitação: contabilizar no mínimo 10 problemas de responsividade, priorizando os maiores impactos na usabilidade.
    - Ferramenta: DevTools.

- **Teste de Qualidade de Página:** Avalia os aspectos da qualidade da página diante de uma auditoria automatizada.
    - Objetivo do teste: assegurar aspectos como desempenho, boas práticas e acessibilidade classificadas como boas.
    - Técnica: Auditoria automatizada.
    - Critério de aceitação: 80% dos problemas identificados estejam alinhados com as métricas e recomendações da ferramenta.
    - Ferramenta: Google Lighthouse.

- **Teste de Qualidade de Código:** Avalia os aspectos da qualidade do código diante de uma auditoria automatizada para o mesmo seguir as melhores práticas do padrão da linguagem.
    - Objetivo do teste: assegurar que a estrutura interna do código foi desenvolvida coerente às regras da linguagem.
    - Técnica: Auditoria automatizada.
    - Critério de aceitação: contabilizar 15 problemas, dando relevância aos critérios classificados com maior prioridade.
    - Ferramenta: Sonar Lint.

8.2. Triagem de Bugs

Para a triagem dos bugs encontradas a partir dos testes, definem-se as etapas de identificação dos bugs, priorização, acompanhamento e correção e registro de problemas encontrados, conforme apresentado abaixo:

- **Identificação dos bugs:** será realizada toda a análise do sistema a fim de identificar erros para resolução futura.
- **Priorização dos bugs:** serão resolvidos bugs classificados com maior prioridade, ou seja, prioridades crítica e alta.
    - **Crítica:** Problemas que devem ser solucionados imediatamente, impedem a execução de alguma tarefa. Ex: Erros que afetam a experiência do usuário.
    - **Alta:** Problemas relacionados às dependências e configurações, porém não são bloqueadores. Ex: Configuração de dependência.
    - **Média:** Erros que não afetam diretamente as funcionalidades do sistema. Ex: Erros de digitação.
- **Acompanhamento e correção:** será realizado o monitoramento e resolução dos bugs encontrados, diante das fases de entrega.
- **Registro dos bugs: serão documentados** os bugs encontrados para a validação e verificação do sistema, permitindo uma rastreabilidade completa.

8.3. Conclusão do Teste

Espera-se, a partir da execução das etapas de teste, garantir a conformidade do sistema em relação às especificações e requisitos definidos na documentação. Além disso, consoante os critérios estabelecidos neste plano, procura-se atingir ao menos 80% da cobertura total dos casos de teste propostos. Portanto, finalizado a execução dos testes, pretende-se empregar manutenção preventiva para bugs encontrados com classificação de prioridade crítica.

Com a conclusão dos testes, procura-se ainda documentar os resultados obtidos por meio do resumo proveniente do processo e solução dos testes empregados, além da análise para possíveis sugestões de melhorias futuras partindo dos defeitos encontrados.

9. Resultados de teste

A partir do planejamento e execução dos testes, espera-se entregar um relatório de testes contendo os casos de teste efetuados, assim como os resultados determinando os casos de sucesso e fracasso, bem como se o teste passou ou falhou. Além disso, os defeitos encontrados devem ser relatados com possíveis recomendações e soluções de melhoria.

Sob aspectos do ponto de vista quantitativo, os resultados devem ser contabilizados sobre a quantidade dos casos de teste executados em relação ao valor previsto. Nesse sentido, será feita a análise em relação aos casos de sucesso e fracasso calculados. Ademais, deve-se avaliar a quantidade de funcionalidades atendidas e faltantes conforme os requisitos do sistema e as métricas de desempenho da página.

10. Cronograma

Tarefa	Proprietário	Situação	Data de início	Data de término
Testes funcionais com Selenium	LIVIA BEATRIZ MAI...	Não iniciada	02/09/2024	06/09/2024

Testes funcionais com Selenium	ERIKY ABREU VELO…	Não iniciada	02/09/2024	06/09/2024
Testes estruturais com NUnit	GEISA MORAIS GAB…	Não iniciada	02/09/2024	06/09/2024
	LIVIA BEATRIZ MAI…	Não iniciada	09/09/2024	13/09/2024
	ERIKY ABREU VELO…	Não iniciada	09/09/2024	13/09/2024
	GEISA MORAIS GAB…	Não iniciada	09/09/2024	13/09/2024