Plano de Teste

Sistema de Prefeitura

Versão 3.0

Histórico das alterações

Data	Versão	Descrição	Autor
18/06/2023	0.1	Release Inicial	Leonardo Gravina Carlos

1 - Introdução

Este documento tem como objetivo descrever os requisitos a serem testados, os tipos de testes definidos para cada iteração, os recursos de hardware e software a serem empregados e o cronograma dos testes ao longo do projeto. As seções referentes aos requisitos, recursos e cronograma têm como finalidade permitir ao gerente do projeto acompanhar a evolução dos testes.

1.01 - Informações do Projeto e Componentes de Software:

O sistema de Prefeitura em questão é uma plataforma online privada, destinada ao uso exclusivo dos funcionários da prefeitura. O objetivo é fornecer aos funcionários uma interface segura e conveniente para acessar e gerenciar informações relacionadas às atividades administrativas. O sistema é composto por vários componentes, incluindo módulos de autenticação, gerenciamento de contas, solicitação de serviços internos, geração de relatórios, entre outros.

1.02 - Requisitos a Testar:

Será realizada uma análise detalhada dos requisitos do sistema de Prefeitura para identificar todos os aspectos que devem ser testados. Os requisitos podem ser categorizados em requisitos funcionais e não funcionais. Alguns exemplos de requisitos a serem testados podem incluir:

- Funcionalidades do sistema: gerenciar e verificar rotas de coleta de lixo, arrecadar impostos, liberar licenças, etc.
- Requisitos de desempenho: tempo de resposta aceitável para as operações do sistema, escalabilidade para suportar o crescimento futuro, sistema compatível com múltiplos dispositivos, etc.

- Requisitos de segurança: autenticação segura dos funcionários, proteção de dados sensíveis e confidenciais, com criptografia e controle de acesso adequados, etc.
- Requisitos de usabilidade: interface do usuário intuitiva e de fácil navegação, treinamento e suporte adequados para garantir a adoção eficiente do sistema pelos funcionários, etc.

1.03 - Estratégias de Teste:

Com base nos requisitos identificados, serão recomendadas estratégias de teste a serem empregadas. Isso pode incluir uma combinação de testes de unidade, testes de integração, testes de sistema, testes de aceitação e outros tipos de testes relevantes para garantir a cobertura adequada dos requisitos. Cada estratégia de teste será descrita em termos de seus objetivos, abordagem, escopo e critérios de sucesso.

- Testes de Unidade: serão realizados para verificar a funcionalidade correta dos componentes individuais do software, como funções específicas relacionadas às áreas de atuação da prefeitura, por exemplo, emissão de documentos, controle de licenças, entre outros.
- Testes de Integração: serão conduzidos para verificar a integração adequada entre os vários componentes do sistema e garantir que as interfaces estejam funcionando corretamente, incluindo integrações com sistemas internos da prefeitura, como bancos de dados.
- Testes de Sistema: serão executados para validar o sistema como um todo, incluindo a interação entre os diferentes módulos, o fluxo de dados correto e a conformidade com os requisitos especificados pela prefeitura, como legislações municipais.
- Testes de Aceitação: serão realizados para verificar se o software atende aos critérios de aceitação definidos pela prefeitura, considerando requisitos específicos relacionados às políticas e procedimentos administrativos, bem como às necessidades dos funcionários envolvidos na operação do sistema.

1.04 - Recursos Necessários e Estimativa de Esforços de Teste:

Para realizar os testes de forma eficiente, serão identificados os recursos necessários, tanto em termos de hardware quanto de software. Isso pode incluir equipamentos de informática, acesso a redes internas da prefeitura, sistemas operacionais, ferramentas de automação de testes, bancos de dados e outros recursos relevantes. Além disso, será fornecida uma estimativa dos esforços de teste necessários, incluindo a duração prevista para cada tipo de teste e a alocação de recursos humanos.

- Hardware: Serão necessários equipamentos de informática, como computadores desktop ou laptops, para a execução dos testes. Também pode ser necessário o acesso a redes internas da prefeitura para testes de integração com sistemas existentes.

- Software: Será necessário um conjunto de ferramentas de teste adequadas, como frameworks de automação, ferramentas de registro de bugs e software de virtualização para configurar ambientes de teste. Além disso, é importante contar com licenças de software necessárias para testar os sistemas da prefeitura.

- Recursos Humanos: Uma equipe de testes será necessária para conduzir os testes. Isso pode incluir funcionários da prefeitura com conhecimentos técnicos adequados, como analistas de sistemas, programadores e outros profissionais de TI.

1.05 - Elementos Resultantes do Projeto de Testes:

Como resultado do processo de planejamento de testes, diversos elementos serão produzidos. Isso inclui planos de teste detalhados para cada tipo de teste, cenários de teste, casos de teste, dados de teste, scripts de automação, relatórios de testes e outros artefatos relacionados ao processo de teste. Esses elementos serão documentados e organizados de forma a permitir uma execução sistemática e um acompanhamento eficaz dos resultados dos testes.

1.06 - Ambiente de Testes:

Será necessário estabelecer um ambiente de testes adequado para conduzir os testes de forma eficiente e precisa. Isso envolve a criação de ambientes de desenvolvimento e teste separados, que possam refletir com precisão o ambiente de produção do software utilizado pela prefeitura. O ambiente de teste deve ser configurado com os mesmos componentes de hardware, software, redes e configurações de segurança para garantir a reprodução fiel das condições reais de uso.

1.07 - Dados de Teste:

Serão necessários conjuntos de dados de teste realistas e representativos para cobrir os diferentes cenários e casos de uso do sistema de prefeitura. Esses conjuntos de dados devem incluir uma variedade de dados, como informações pessoais de cidadãos, registros de documentos, informações de funcionários, entre outros. Além disso, serão criados dados de teste específicos para validar diferentes funcionalidades e requisitos, como dados para testar limites, casos de exceção e situações de erro.

1.08 - Estratégia de Defeitos e Gestão de Problemas:

Uma estratégia de gestão de defeitos e problemas deve ser estabelecida para rastrear, documentar e resolver quaisquer problemas encontrados durante os testes. Isso envolve a implementação de um sistema de registro de bugs, atribuição de prioridades e severidades aos defeitos, acompanhamento do status de resolução e comunicação eficaz entre os membros da equipe de teste, desenvolvimento e gerenciamento do projeto. O objetivo é garantir que todos os defeitos sejam adequadamente registrados e tratados, visando sua resolução e prevenção em futuras versões do software.

1.09 - Plano de Teste Iterativo:

Considerando que os testes serão conduzidos em várias iterações, será elaborado um plano de teste detalhado para cada iteração. Cada plano de teste especificará os objetivos, escopo, atividades, cronograma e recursos alocados para a iteração específica. Além disso, serão estabelecidos critérios de entrada e saída para cada fase de teste, permitindo a transição adequada entre as iterações e fornecendo um roteiro claro para o progresso do teste.

1.10 - Cronograma de Teste:

Um cronograma de teste será elaborado para planejar e acompanhar as atividades de teste ao longo do projeto. O cronograma de teste definirá as datas de início e término de cada fase de teste, bem como as tarefas e marcos importantes a serem alcançados em cada fase. Ele permitirá o planejamento eficiente dos recursos, a identificação de dependências e a garantia de que o teste seja concluído dentro dos prazos estabelecidos.

1.11 - Relatórios de Teste:

Serão gerados relatórios de teste regulares para comunicar o status, os resultados e as métricas relevantes do teste. Esses relatórios podem incluir informações sobre a cobertura de teste, resultados de execução, métricas de desempenho, problemas identificados, progresso em relação ao cronograma, entre outros aspectos relevantes. Os relatórios de teste serão compartilhados com a equipe de desenvolvimento, gerentes do projeto e outras partes interessadas relevantes, permitindo uma visão clara do estado dos testes e suportando a tomada de decisões informadas.

1.12 - Atividades de Encerramento:

Ao final do ciclo de teste, serão realizadas atividades de encerramento para revisar o desempenho do teste, identificar áreas de melhoria e capturar lições aprendidas. Isso inclui a documentação de quaisquer problemas não resolvidos, atualização da documentação do sistema, arquivamento de artefatos de teste e preparação para a próxima fase do projeto, como a implantação do sistema em produção.

Sistema a ser Testado:

O sistema a ser testado é o software da Prefeitura, que abrange todos os módulos e funcionalidades mencionados anteriormente. O sistema será submetido a um processo de teste rigoroso para garantir sua confiabilidade, segurança, desempenho e usabilidade. Os testes serão conduzidos em várias iterações, com cada iteração focada em um conjunto específico de requisitos e casos de teste. Essa abordagem iterativa permite a identificação precoce de problemas e a implementação de melhorias contínuas ao longo do projeto.

2 - Requisitos a Testar

Esta seção contém os casos de uso e requisitos não funcionais identificados como objetos dos testes ao longo do desenvolvimento do projeto do sistema da Prefeitura.

Casos de uso:

Identificador do caso de uso	**Nome do caso de uso**
CDU001	ACESSAR SISTEMA
CDU002	REGISTRAR RECLAMAÇÃO
CDU003	ANALISAR ROTAS DE COLETA DE LIXO
CDU004	PERMITIR A MANUTENÇÃO DE EQUIPAMENTOS
CDU005	PERMITIR A DISPOSIÇÃO DO LIXO COLETADO
CDU006	PLANEJAR ROTAS DE COLETA DE LIXO
CDU007	ARRECADAR IMPOSTOS
CDU008	COBRAR DÍVIDAS
CDU009	ACOMPANHAR LEGISLAÇÃO TRIBUTÁRIA
CDU010	EMITIR CERTIDÕES NEGATIVAS DE DÉBITO
CDU011	ATUALIZAR LEGISLAÇÃO TRIBUTÁRIA
CDU012	FISCALIZAR EMPRESAS

CDU013	LIBERAR LICENÇAS
CDU014	CONTROLAR SERVIÇOS TERCEIRIZADOS
CDU015	ANALISAR INFRAESTRUTURAS
CDU016	GERENCIAR E FISCALIZAR PROJETOS E OBRAS
CDU017	GERENCIAR RECURSOS HUMANOS NO SISTEMA
CDU018	CONTROLAR FOLHA DE PAGAMENTO
CDU019	CONTROLAR PONTOS
CDU020	GERENCIAR CONTRATAÇÕES
CDU021	GERENCIAR CAPACITAÇÕES

Breve descrição dos casos de uso

[CDU001] ACESSAR SISTEMA

Descrição do caso de uso: Permitir o acesso do usuário ao sistema.

[CDU002] REGISTRAR RECLAMAÇÃO

Descrição do caso de uso: Permitir que o visitante possa escrever alguma reclamação no sistema.

[CDU003] ANALISAR ROTAS DE COLETAS DE LIXO

Descrição do caso de uso: Permitir a análise das rotas de coleta de lixo para identificar possíveis melhorias.

[CDU004] PERMITIR A MANUTENÇÃO DE EQUIPAMENTOS

Descrição do caso de uso: Permitir a manutenção dos equipamentos utilizados na coleta de lixo.

[CDU005] PERMITIR A DISPOSIÇÃO DO LIXO COLETADO

Descrição do caso de uso: Permitir a manutenção dos equipamentos utilizados na coleta de lixo.

[CDU006] PLANEJAR ROTAS DE COLETA DE LIXO

Descrição do caso de uso: Permitir o planejamento de rotas de coleta de lixo mais eficientes.

[CDU007] ARRECADAR IMPOSTOS

Descrição do caso de uso: Permitir a arrecadação de impostos pelos contribuintes.

[CDU008] COBRAR DÍVIDAS

Descrição do caso de uso: Permitir a cobrança de dívidas dos contribuintes.

[CDU009] ACOMPANHAR LEGISLAÇÃO TRIBUTÁRIA

Descrição do caso de uso: Permitir a visualização completa de toda base de dados da legislação tributária.

[CDU010] EMITIR CERTIDÕES NEGATIVAS DE DÉBITO

Descrição do caso de uso: Permitir a emissão de certidões negativas de débito para contribuintes.

[CDU011] ATUALIZAR LEGISLAÇÃO TRIBUTÁRIA

Descrição do caso de uso: Permitir a atualização de toda a legislação tributária da prefeitura.

[CDU012] FISCALIZAR EMPRESAS

Descrição do caso de uso: Permitir a fiscalização das empresas cadastradas na prefeitura.

[CDU013] LIBERAR LICENÇAS

Descrição do caso de uso: Gerenciar os recursos humanos da prefeitura no sistema.

[CDU014] CONTROLAR SERVIÇOS TERCEIRIZADOS

Descrição do caso de uso: Gerenciar os serviços prestados para prefeitura por terceiros.

[CDU015] ANALISAR INFRAESTRUTURAS

Descrição do caso de uso: Gerenciar os recursos humanos da prefeitura no sistema.

[CDU016] GERENCIAR E FISCALIZAR PROJETOS E OBRAS

Descrição do caso de uso: Gerenciar os recursos humanos da prefeitura no sistema.

[CDU017] GERENCIAR RECURSOS HUMANOS NO SISTEMA

Descrição do caso de uso: Gerenciar os recursos humanos da prefeitura no sistema.

[CDU018] CONTROLAR FOLHA DE PAGAMENTO

Descrição do caso de uso: Controlar a folha de pagamento dos funcionários da prefeitura.

[CDU019] CONTROLAR PONTOS

Descrição do caso de uso: Controlar o ponto dos funcionários, seja ele manual ou digital.

[CDU020] GERENCIAR CONTRATAÇÕES.

Descrição do caso de uso: Gerenciar as contratações de funcionários para a prefeitura.

[CDU021] GERENCIAR CAPACITAÇÕES

Descrição do caso de uso: Gerenciar as capacitações dos funcionários da prefeitura.

Requisitos não-funcionais:

Identificador do requisito	Nome do requisito
NF001	INTERFACE AMIGÁVEL
NF002	COMPONENTES WEB
NF003	BANCOS DE DADO MYSQL
NF004	LINGUAGEM JAVA
NF005	AGILIDADE NA EXECUÇÃO DAS OPERAÇÕES
NF006	COMPATIBILIDADE DO SISTEMA
NF007	ESCALABILIDADE DO SISTEMA
NF008	SEGURANÇA DOS DADOS E INFORMAÇÕES
NF009	SISTEMA CONFIÁVEL

NF010	COMPATIBILIDADE COM A LEI

Breve descrição dos Requisitos não-funcionais

[NF001] INTERFACE AMIGÁVEL

O sistema deve ter uma interface amigável ao usuário primário sem se tornar cansativa aos usuários mais experientes.

[NF002] COMPONENTES WEB

A interface deverá utilizar elementos comuns a usuários de sistemas web, como campos de texto, *links* e botões, sem muito rebuscamento. A ideia é focar nos aspectos operacionais sem se preocupar tanto com a beleza da tela, de modo a facilitar o uso por usuários iniciantes.

[NF003] BANCO DE DADOS MYSQL

O sistema deve utilizar um banco de dados MySQL para fazer o armazenamento de dados.

[NF004] LINGUAGEM JAVA

Visando criar um produto com maior extensibilidade, reusabilidade e flexibilidade, deve-se adotar Java como linguagem principal de desenvolvimento, seguindo cuidadosamente as técnicas de orientação a objetos.

[NF005] AGILIDADE NA EXECUÇÃO DAS OPERAÇÕES

O sistema deve executar as operações no menor tempo possível, visando dar uma maior agilidade ao processo.

[NF006] COMPATIBILIDADE DO SISTEMA

O sistema deve ser compatível com diferentes tipos de navegadores e dispositivos, tornando sua utilização mais fácil e confortável ao usuário.

[NF007] ESCALABILIDADE DO SISTEMA

O sistema deve ser escalável, permitindo o aumento da capacidade de armazenamento e processamento e sendo capaz de lidar com um grande volume de dados.

[NF008] SEGURANÇA DOS DADOS E INFORMAÇÕES

O sistema deve garantir a segurança dos dados e informações nele inseridos, tanto dos funcionários quanto dos equipamentos da prefeitura.

[NF009] SISTEMA CONFIÁVEL

O sistema deve ser confiável e garantir a disponibilidade e integridade dos dados.

[NF010] COMPATIBILIDADE COM A LEI

O sistema deve ser compatível com as normas e legislações vigentes, mostrando a todos os usuários que o sistema é seguro e confiável.

3 - Tipos de teste

Nesta seção, serão apresentados os tipos de testes escolhidos para cada iteração do projeto do Sistema de Prefeitura. Inicialmente, serão listados os tipos de testes que serão utilizados na próxima iteração, e também serão registrados eventuais tipos de teste que se espera utilizar nas demais iterações. Cada tipo de teste será descrito brevemente, indicando sua relevância em relação aos requisitos, tipo da aplicação e recursos disponíveis.

- **Teste de interface de usuário:**
    - Descrição: Esse tipo de teste verifica se a interface do usuário está funcionando corretamente e atendendo aos requisitos estabelecidos. Envolve testar a usabilidade, navegabilidade, layout, design e interação com o usuário.
    - Critérios de completude sugeridos: Todos os elementos da interface devem ser testados, incluindo botões, campos de entrada, menus e links. Verificar se a interface é intuitiva, responsiva e se as informações são apresentadas de forma clara e adequada.
- **Teste de desempenho:**
    - Descrição: Esse tipo de teste avalia o desempenho do software em diferentes condições e cargas de trabalho. O objetivo é identificar possíveis gargalos, lentidão ou problemas de resposta, garantindo que o sistema funcione de forma eficiente e atenda aos requisitos de desempenho.
    - Critérios de completude sugeridos: Analisar os tempos de resposta, a capacidade de processamento, o consumo de recursos e a estabilidade do sistema em diferentes cenários.

- **Teste de carga:**
    - Descrição: Esse tipo de teste verifica o comportamento do sistema sob uma carga de trabalho esperada, simulando a quantidade de usuários e transações que o sistema deverá suportar no ambiente de produção.
    - Critérios de completude sugeridos: Estabelecer métricas de desempenho aceitáveis, como tempos de resposta e utilização de recursos. Verificar se o sistema se mantém estável e se não há degradação significativa do desempenho durante períodos prolongados de carga.
- **Teste de segurança e controle de acesso:**
    - Descrição: Esse tipo de teste avalia a segurança do sistema, verificando se os mecanismos de autenticação, autorização e controle de acesso estão eficientes e atendem às políticas de segurança estabelecidas.
    - Critérios de completude sugeridos: Verificar se não há brechas de segurança, como vulnerabilidades conhecidas ou possibilidade de acesso não autorizado a funcionalidades ou informações sensíveis. Testar diferentes cenários de autenticação e autorização para garantir que o sistema mantenha a integridade e a confidencialidade dos dados.
- **Teste de instalação:**
    - Descrição: Esse tipo de teste verifica a instalação e configuração do software da Prefeitura em diferentes ambientes, como sistemas operacionais, servidores e dispositivos móveis.
    - Critérios de completude sugeridos: Verificar se o processo de instalação é simples, intuitivo e bem documentado. Testar a compatibilidade do software com diferentes plataformas e configurar corretamente os pré-requisitos e as dependências necessárias para a instalação.
- **Teste de usabilidade:**
    - Descrição: Esse tipo de teste visa avaliar a facilidade de uso e a experiência do usuário ao interagir com o software. O foco está na identificação de problemas de usabilidade, fluxos confusos e dificuldades de navegação.
    - Critérios de completude sugeridos: Realizar testes com usuários reais, observando sua interação com o software. Coletar feedback sobre a intuitividade da interface, clareza das informações, eficiência das ações e facilidade de aprendizado. Realizar iterações de design com base nos resultados obtidos.

- **Teste de compatibilidade:**
    - Descrição: Esse tipo de teste verifica se o software é compatível com diferentes sistemas operacionais, navegadores, dispositivos e versões. O objetivo é garantir que o aplicativo funcione corretamente em diferentes ambientes.
    - Critérios de completude sugeridos: Testar o software em uma variedade de combinações de sistemas operacionais (Windows, macOS, Linux), navegadores (Chrome, Firefox, Safari, Edge). Verificar se a interface se adapta adequadamente e se todas as funcionalidades são suportadas.
- **Teste de regressão:**
    - Descrição: Esse tipo de teste visa garantir que as alterações realizadas em novas versões do software não introduzam regressões, ou seja, não causem problemas em funcionalidades que já estavam funcionando corretamente.
    - Critérios de completude sugeridos: Desenvolver um conjunto de testes abrangente que englobe as principais funcionalidades do software. Realizar esses testes regularmente a cada nova versão, para garantir que todas as funcionalidades previamente testadas permaneçam funcionando corretamente.
- **Teste de localização:**
    - Descrição: Esse tipo de teste verifica se o software está adaptado corretamente para diferentes localidades, idiomas e culturas. Envolve verificar a tradução adequada, a formatação correta de datas, moedas e números, e a conformidade com requisitos legais e regulatórios em diferentes regiões.
    - Critérios de completude sugeridos: Testar o software em diferentes idiomas e regiões geográficas, verificando a exatidão da tradução, a formatação dos elementos de interface e a conformidade com leis e regulamentos locais.
- **Teste de recuperação de desastres:**
    - Descrição: Esse tipo de teste avalia a capacidade do sistema de se recuperar de falhas graves, como quedas de energia, falhas de hardware ou desastres naturais. O objetivo é garantir que o software possua mecanismos adequados de backup, replicação e recuperação de dados.
    - Critérios de completude sugeridos: Simular situações de desastres ou falhas graves, como desligar abruptamente o servidor principal. Verificar se o

sistema consegue recuperar os dados, migrar para um servidor secundário e retomar as operações sem perda significativa de dados ou tempo de inatividade.

- **Teste de integração:**
    - Descrição: Esse tipo de teste avalia a integração entre os diferentes componentes do software da Prefeitura, como sistemas de autenticação, processamento de dados, serviços externos (APIs), entre outros. O objetivo é garantir que esses componentes funcionem corretamente juntos.
    - Critérios de completude sugeridos: Identificar os principais componentes do software e definir casos de teste que verifiquem a integração entre eles. Testar diferentes fluxos de trabalho, desde a autenticação até a execução de processos complexos, garantindo que todas as interações entre os componentes ocorram sem problemas.

Esses são apenas alguns exemplos de tipos de teste que podem ser aplicados no Software da Prefeitura. Conforme o projeto avance, novos tipos de testes podem ser adicionados ou adaptados de acordo com os requisitos específicos e a evolução do software. É importante adaptar os critérios de completude sugeridos para cada tipo de teste de acordo com as características e necessidades do projeto em questão.

Termos Relacionados aos Testes de Software

- **Técnica de Teste:** A técnica de teste se refere ao método ou abordagem usada para projetar e executar testes em um software. Existem duas principais categorias de técnicas de teste: manual e automática.
    - Teste Manual: Nessa abordagem, os testes são realizados por testadores humanos, que executam casos de teste, verificam o comportamento do software e registram os resultados manualmente. É uma abordagem flexível, permitindo a detecção de problemas sutis e a adaptação a diferentes cenários de teste.
    - Teste Automático: Nessa abordagem, os testes são executados por meio de ferramentas de automação de teste, que executam casos de teste predefinidos sem a necessidade de intervenção humana. É uma abordagem eficiente para testes repetitivos, execução em larga escala e testes de desempenho. No entanto, pode ser limitada na detecção de problemas complexos.

- **Estágio do Teste:** O estágio do teste se refere ao nível ou fase em que os testes são executados no ciclo de vida do desenvolvimento de software. Existem vários estágios de teste, incluindo:
    - Teste de Unidade: Nesse estágio, as unidades individuais de código são testadas isoladamente para garantir que funcionem corretamente. Geralmente, é realizado pelos próprios desenvolvedores.
    - Teste de Integração: Nesse estágio, as unidades testadas individualmente são combinadas e testadas em conjunto para verificar a interoperabilidade e a comunicação correta entre elas.
    - Teste de Sistema: Nesse estágio, o sistema completo é testado como um todo para verificar se atende aos requisitos e especificações. O objetivo é garantir que todas as partes do sistema funcionem corretamente em conjunto.
    - Teste de Aceitação: Nesse estágio, os testes são realizados para validar o sistema em relação aos critérios de aceitação definidos pelo cliente ou usuário final. O objetivo é garantir que o sistema esteja pronto para ser entregue e usado pelo cliente.
- **Abordagem do Teste:** A abordagem do teste refere-se à perspectiva adotada durante o processo de teste. Duas abordagens comuns são:
    - Teste de Caixa Preta: Nessa abordagem, o testador avalia o sistema sem conhecimento detalhado de sua estrutura interna. O foco está nas entradas, saídas e comportamento funcional do software, sem considerar a implementação subjacente. O objetivo é verificar se o software atende aos requisitos e comporta-se conforme o esperado, sem se aprofundar na lógica interna ou no código-fonte.
    - Teste de Caixa Branca: Nessa abordagem, o testador tem conhecimento detalhado da estrutura interna, do código-fonte e da lógica do sistema. O objetivo é testar a lógica interna do software, garantindo que todos os caminhos e instruções sejam exercitados. São verificados aspectos como a cobertura de código, a estrutura correta das decisões e o fluxo de execução. Essa abordagem permite uma análise mais aprofundada do software, identificando possíveis erros ou falhas relacionadas à implementação interna.

3.1 - Métodos da Classe

Para teste de funcionalidade. Aqui deve-se verificar se cada classe retorna o esperado. Se possível usar teste automatizado.

Objetivo	Verificar se cada classe retorna o resultado esperado.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração ( )	Sistema ( )	Unidade (x)	Aceitação ( )
Abordagem do teste	Caixa branca (x)		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.2 - Persistência de Dados

Para teste de integridade de dados e do banco de dados. Aqui deve-se verificar se os dados não se perdem ao desligar o programa. Se o programa consegue se recuperar em caso de falha ou fechamento repentino. Se possível usar teste automatizado.

Objetivo	Verificar se os dados são mantidos após o súbito desligamento do programa.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração ( )	Sistema (x)	Unidade ( )	Aceitação ( )
Abordagem do teste	Caixa branca ( )		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.3 - Integração dos Componentes

Para teste de funcionalidade. Aqui deve-se verificar se as classes e métodos conseguem fazer a integração entre elas para uma sequência de ações do programa. Se possível usar teste automatizado.

Objetivo	Verificar se as classes e métodos se integram corretamente para realizar uma sequência de ações do programa.

Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração (x)	Sistema ( )	Unidade ( )	Aceitação ( )
Abordagem do teste	Caixa branca (x)		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.4 - Tempo de Resposta

Para teste de funcionalidade. Aqui deve-se verificar se o tempo de resposta das ações do programa é considerado aceitável. Se possível usar teste automatizado.

Objetivo	Verificar se o tempo de resposta das ações do programa é aceitável.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração (x)	Sistema (x)	Unidade (x)	Aceitação ( )
Abordagem do teste	Caixa branca (x)		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.5 - Animação

Para teste de funcionalidade (para jogos, principalmente, mas não somente). Aqui deve-se verificar se as animações existentes no programa são disparadas quando devem e se seguem uma sequência lógica. Se possível usar teste automatizado.

Objetivo	Verificar se as animações são acionadas corretamente e se seguem uma sequência lógica no programa.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração (x)	Sistema (x)	Unidade (x)	Aceitação ( )
Abordagem do teste	Caixa branca (x)		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.6 - Efeitos Sonoros

Para teste de funcionalidade. Aqui deve-se verificar se os efeitos sonoros do programa são acionados corretamente nos eventos adequados. Se possível usar teste automatizado.

Objetivo	Verificar se os efeitos sonoros são acionados corretamente nos eventos adequados do programa.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração (x)	Sistema (x)	Unidade (x)	Aceitação ( )
Abordagem do teste	Caixa branca (x)		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.7 - Usabilidade

Para teste de usabilidade. Aqui deve-se verificar se a interface do programa é intuitiva e fácil de usar, considerando a experiência do usuário. Pode-se realizar testes com usuários reais para obter feedbacks. Também pode-se utilizar ferramentas de análise de usabilidade.

Objetivo	Verificar se a interface do programa é intuitiva e fácil de usar, considerando a experiência do usuário.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração ( )	Sistema ( )	Unidade ( )	Aceitação (x)
Abordagem do teste	Caixa branca ( )		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.8 - Segurança

Para teste de segurança. Aqui deve-se verificar se o software possui medidas adequadas de segurança para proteger informações sensíveis, como dados bancários dos usuários. Pode-se realizar testes de penetração, análise de vulnerabilidades e autenticação.

Objetivo	Verificar se o software possui medidas adequadas de segurança para proteger informações sensíveis.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração ( )	Sistema (x)	Unidade ( )	Aceitação ( )
Abordagem do teste	Caixa branca ( )		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.9 - Desempenho

Para teste de desempenho. Aqui deve-se verificar se o programa é capaz de lidar com cargas de trabalho esperadas, sem apresentar degradação significativa no desempenho. Pode-se realizar testes de carga e stress.

Objetivo	Verificar se o programa é capaz de lidar com cargas de trabalho esperadas, sem degradação significativa no desempenho.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração ( )	Sistema (x)	Unidade ( )	Aceitação ( )
Abordagem do teste	Caixa branca ( )		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.10 - Compatibilidade

Para teste de compatibilidade. Aqui deve-se verificar se o programa é compatível com diferentes sistemas operacionais, dispositivos e versões de navegadores, se aplicável. Pode-se realizar testes em diferentes ambientes e configurações para identificar problemas de compatibilidade.

Objetivo	Verificar se o programa é compatível com diferentes sistemas operacionais, dispositivos e versões de navegadores, se aplicável.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração ( )	Sistema (x)	Unidade ( )	Aceitação ( )
Abordagem do teste	Caixa branca ( )		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.11- Acessibilidade

Para teste de acessibilidade. Aqui deve-se verificar se o software é acessível para pessoas com deficiência, seguindo diretrizes de acessibilidade, como as estabelecidas pelas WCAG (Web Content Accessibility Guidelines). Pode-se realizar testes com ferramentas de acessibilidade e com usuários com deficiência.

Objetivo	Verificar se o software é acessível para pessoas com deficiência.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração ( )	Sistema ( )	Unidade ( )	Aceitação ( )
Abordagem do teste	Caixa branca ( )		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

3.12 - Desempenho

Para teste de desempenho. Aqui deve-se verificar o desempenho do software em relação a tempo de resposta, escalabilidade e consumo de recursos, como memória e CPU. Pode-se

realizar testes de carga, estresse e volume para avaliar o comportamento do sistema em diferentes cenários.

Objetivo	Verificar o desempenho do software em relação a tempo de resposta, escalabilidade e consumo de recursos.			
Técnica:	( ) manual		(x) automática	
Estágio do teste	Integração ( )	Sistema (x)	Unidade ( )	Aceitação ( )
Abordagem do teste	Caixa branca ( )		Caixa preta (x)	
Responsável(is)	Leonardo Gravina Carlos			

4 - Recursos

Esta seção deve descrever os recursos humanos, ambiente de testes (hardware e software) e ferramentas de automação de testes necessários para realizar os testes descritos nas subseções a seguir.

4.1 - Ambiente de teste - Software e Hardware

- **Hardware:**
    - Computadores com capacidade de processamento e memória adequados para suportar as demandas do sistema da prefeitura.
    - Dispositivos representativos dos diferentes usuários, como computadores desktop, laptops, tablets e smartphones.
    - Servidores para hospedar o sistema da prefeitura, com capacidade adequada para lidar com o número esperado de usuários.
    - Redes e conexões de internet estáveis e confiáveis para garantir a comunicação adequada entre os componentes do sistema.
- **Software:**
    - Sistema operacional: Windows 10, macOS Big Sur, Ubuntu 20.04 LTS (ou versões mais recentes).
    - Navegadores: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge (últimas versões estáveis).

- Servidor web: Apache, Nginx (últimas versões estáveis).
- Banco de dados: MySQL, PostgreSQL (últimas versões estáveis).
- Ferramentas de virtualização: para criar ambientes de teste isolados e reproduzíveis, pode ser útil utilizar ferramentas de virtualização como o VirtualBox ou o VMware para criar máquinas virtuais com diferentes configurações de sistema operacional e software.
- Ferramentas de automação de testes: além do Selenium WebDriver, outras ferramentas populares de automação de testes incluem o Appium (para testes automatizados em dispositivos móveis), o Robot Framework (para testes de aceitação e automação de testes de interface do usuário) e o Cypress (para testes de interface do usuário baseados em JavaScript).
- Sistemas de gerenciamento de banco de dados (SGBDs): dependendo do banco de dados utilizado pelo sistema da prefeitura, pode ser necessário configurar e gerenciar instâncias de SGBDs como o MySQL, o PostgreSQL ou o Oracle Database para realizar testes de integração e validar consultas e transações.

4.2 - Ferramentas de teste

As ferramentas específicas de teste usadas no projeto podem incluir:

- Selenium WebDriver: Ferramenta de automação de teste para interagir com navegadores web. Pode ser usada para criar testes automatizados de interface do usuário, preenchendo formulários, clicando em elementos e verificando o comportamento esperado.
- JMeter: Ferramenta de teste de carga e desempenho para medir o desempenho do sistema da prefeitura sob diferentes cenários de carga. Permite simular usuários simultâneos, enviar solicitações HTTP, medir tempo de resposta e analisar métricas de desempenho.
- OWASP ZAP: Ferramenta de teste de segurança que pode ser usada para realizar testes de penetração e identificar vulnerabilidades no sistema. Pode ajudar a garantir a segurança das informações dos usuários e evitar ataques maliciosos.
- JUnit: Framework de testes unitários para Java. Pode ser usado para escrever e executar testes automatizados de unidade para garantir a corretude e funcionalidade das diferentes partes do código-fonte.

- Postman: Ferramenta para testar e depurar APIs. Permite enviar solicitações HTTP para os endpoints do sistema da prefeitura, verificar respostas, validar dados e automatizar testes de integração.

- SonarQube: Uma plataforma de análise estática de código que pode ajudar a identificar problemas de qualidade, como código duplicado, vulnerabilidades de segurança e violações de boas práticas de programação.

5 - Cronograma

Tipo de teste	**Duração**	**data de início**	**data de término**
planejar teste	15 dias	05/06/2023	21/06/2023
projetar teste	15 dias	22/06/2023	06/07/2023
implementar teste	15 dias	07/07/2023	21/07/2023
executar teste	15 dias	22/07/2023	06/08/2023
avaliar teste	15 dias	07/08/2023	22/08/2023