UNIVERSIDADE DO ESTADO DO AMAZONAS – UEA

ESCOLA SUPERIOR DE TECNOLOGIA – EST

PROJETO HEFESTO

Prof. ÁUREA MELO BRAGA

MATHEUS UCHÔA

ANA BEATRIZ

PAULO HENRIQUE MUNHOZ

KARINY OLIVEIRA

IVAN MILLER

EDUARDO RODRIGUES

PLANEJAMENTO DE TESTES

ENDEREÇO IP VÁLIDO

MANAUS

2020

1 INTRODUÇÃO

Usuários e clientes geralmente esperam um software de qualidade. Dessa forma é importante realizar a verificação e validação em todos os artefatos que devem ser entregues.

O planejamento de teste é uma atividade que visa a definir como o software será validado e quais técnicas serão utilizadas. Essa atividade tem como saída o plano de teste. Esse documento descreve as abordagens e metodologias que irão ser aplicadas aos testes de unidade, integração e sistema da aplicação “Validação de endereço IP”. Incluindo ainda os objetivos, responsabilidades de teste, critérios de entrada e saída, escopo, cronograma e abordagens.

2 ESCOPO

O documento foca nos testes de sistema e validação de dados de entrada e saída de acordo com os requisitos estabelecidos pelo cliente e Product Owner Profª Áurea Melo Braga.

2.1 Funcionalidades a serem testadas.

- Integração da função com o programa principal;
- Função valida IP;
- Tratamento para diferentes tipos de entrada;
- Sistema como um todo, por meio da função principal.

2.2 Funções que não serão testadas.

1. Qualquer outra não mencionada na seção 2.1

3 OBJETIVOS DE QUALIDADE

3.1 Objetivos primários

O objetivo primário do teste é garantir que o sistema atende aos requisitos estipulados pelo cliente, incluindo os critérios funcionais e não-funcionais e satisfaz os cenários de uso do cliente, mantendo a qualidade do produto. Ao fim do processo de desenvolvimento, o cliente deve estar satisfeito com o programa, tendo ele atingido ou excedido as suas expectativas como detalhado nos requisitos.

3.2 Objetivos secundários

Os objetivos secundários do teste são: identificar e expor todos os problemas e riscos associados, comunicando todos os problemas ao time de projeto e garantir que todos os problemas sejam tratados de maneira correta antes da liberação do sistema.

4 ABORDAGEM DE TESTE

Portanto, a abordagem usada é analítica, de acordo com a estratégia baseada em requisitos, onde uma análise da especificação de requisitos forma a base para o planejamento, estimativa de plano e design de testes. Os casos de teste serão criados durante o teste exploratório. Os tipos de teste são determinados na Estratégia de Teste.

A equipe também deve usar testes baseados na experiência e adivinhação de erros, utilizando as habilidades e intuição dos testadores, juntamente com sua experiência com aplicativos ou tecnologias semelhantes.

O projeto está usando uma abordagem ágil, com iterações constantes. No final de cada 2 dias, os requisitos identificados para essa iteração serão entregues à equipe e serão testados.

4.1 Automatização de testes

Não são utilizados testes automatizados devido ao curto espaço de desenvolvimento e escopo do sistema.

5 PAPÉIS E RESPONSABILIDADES

Papel	**Membro da Equipe**	**Responsabilidade**
Product Owner	Áurea Melo Braga	● Responsável pelo cronograma e direcionamento do projeto; ● Responsável pela coleta de requisitos; ● Ser a maior fonte de informações sobre as prioridades do projeto; ● Indicar claramente os itens necessários do Product Backlog; ● Otimizar o valor do trabalho entregue pelo time de desenvolvimento; ● Assegurar a qualidade da entrega do produto.
Scrum Master	Matheus Serrão	● Ensinar e liderar a equipe de desenvolvimento para criar produtos de alto valor; ● Remover impedimentos para o progresso do desenvolvimento; ● Facilitador do Daily Scrum; ● Guiar o time utilizando principios de Scrum.
QA	Kariny Oliveira Eduardo Rodrigues	● Entender os requisitos; ● Escrever e executar casos de teste; ● Revisar os casos de teste; ● Reportar e acompanhar defeitos; ● Retestar o software após correções; ● Preparar os dados de teste; ● Coordenar com o time qualquer problema ou defeito encontrados durante os testes; ● Avaliar se as funcionalidades desenvolvidas atendem aos requisitos estabelecidos.
Desenvolvedor	Ana Beatriz Ivan Miller Paulo Henrique Araújo	● Desenvolver soluções baseadas nos requisitos especificados; ● Compreender, executar, realizar a manutenção e corrigir possíveis erros do software.

6 CRITÉRIOS DE ENTRADA E SAÍDA

6.1 Critérios de Entrada

- Toda a documentação necessária, design e requisitos devem estar disponíveis, permitindo assim que os testadores operem o sistema e verifiquem o funcionamento correto;
- Dados de teste propriamente disponíveis;
- O ambiente de teste, como: um computador com as ferramentas necessárias para a execução do programa estar disponível;
- Os testadores terem compreendido completamente os requisitos;
- Os testadores devem ter conhecimento sobre a execução do programa;
- Cenários de teste e casos de teste revisados.

6.2 Critérios de Saída

- Os requisitos devem ser 100% cobertos;
- Nenhum bug severo ou de alta prioridade estar presente;
- Todas as áreas de alto risco devem estar testadas completamente;
- O cronograma ser atendido.

7 CRITÉRIOS DE SUSPENSÃO

7.1 Critérios de Suspensão

- O programa conter defeitos severos ou o teste não estar completo;
- Mudança significativa nos requisitos sugeridas pelo cliente, que podem afetar grande parte ou totalmente o desenvolvimento atual;
- Recursos necessários não disponíveis quando necessário pelo time de teste.

8 ESTRATÉGIA DE TESTE

8.1 Papel do QA no processo de teste

Entendendo os requisitos

- As especificações requisitos vão ser enviadas pelo cliente;
- Entender os requisitos é responsabilidade do QA.

Preparando os casos de teste

- QA vai preparar os testes baseado em um teste exploratório. Os testes devem cobrir todos os cenários dos requisitos especificados.

Preparando a matriz de testes:

- O QA vai preparar a matriz de testes que mapeia cada caso de teste com o requisito especifico. Isso vai garantir a cobertura dos requisitos.

Revisando os casos de teste e a matriz

- A revisão dos artefatos será feita em pares;
- Qualquer comentário ou sugestão nos casos de teste ou na matriz será realizada pela perspectiva do revisor;

- Sugestões e melhorias serão trabalhadas pelo autor e serão enviadas para aprovação;
- Sugestões e melhorias re-trabalhadas serão revisadas e aprovadas pelo revisor.

Criando os dados de teste:

- Os dados de teste serão criados pelo QA baseados nos cenários e casos de teste.

Executando os casos de teste:

- Os casos de teste serão executados no ambiente de teste baseados nos cenários, casos de teste e dados de teste;
- Os resultados dos testes (Resultado atual, Passou/Falhou) serão atualizados no relatório dos testes;
- O QA vai registrar os defeitos/bugs encontrados durante a execução em um documento. Depois disso o QA vai informar para os desenvolvedores a respeito dos defeitos/bugs encontrados.

Reteste:

- O reteste dos bugs corrigidos vai ser realizado pelo QA após os desenvolvedores resolverem os bugs e disponibilizarem uma nova versão do programa.

Entrega:

- Quando todos os bugs/defeitos reportados estiverem corrigidos e nenhum outro defeito ser encontrado um relatório será gerado ao cliente pelo Product Owner.

8.2 Tipos de teste

Teste de caixa-preta:

- Também chamado de teste de ambiente ou partição. Esse tipo de teste foca nos requisitos funcionais do programa. Ele permite derivar conjuntos de condições de entrada que exercitam totalmente todos os requisitos funcionais de um programa.

Teste de integração:

- O teste de integração é uma técnica sistemática para construir a estrutura do programa enquanto realiza o teste para descobrir erros associados à interação. No Relatório, o teste de integração inclui o Relatório de teste dos respectivos locais.

Teste Funcional:

- O teste funcional é realizado para descobrir um comportamento inesperado do relatório. A característica do teste funcional é fornecer correção, confiabilidade, testabilidade e precisão da saída / dados do relatório.

Teste de Sistema:

- O teste do sistema de software é realizado em um sistema completo e integrado para avaliar a conformidade do sistema com os requisitos especificados.

8.4 Definição de prioridade e severidade de bugs.

- Os campos Severidade e Prioridade dos Bugs são muito importantes para categorizar bugs e priorizar se e quando os bugs serão corrigidos. Os níveis de severidade e prioridade do bug serão definidos conforme descrito nas tabelas a seguir. O teste atribui um nível de gravidade a todos os erros.

Lista de Severidade

ID de severidade	Severidade	Descrição
1	Crítica	O módulo / produto trava ou o erro causa condições não recuperáveis. Falhas no sistema ou corrupção de banco de dados ou arquivo ou perda potencial de dados, programa travamentos que exigem reinicialização são exemplos de um bug de severidade 1.
2	Alta	Componente principal do sistema inutilizável devido a falha ou funcionalidade incorreta. Bugs de severidade 2 causam problemas sérios, como falta de funcionalidade ou mensagens de erro insuficientes ou pouco claras que pode ter um grande impacto para o usuário, impede que outras áreas do aplicativo sejam testadas etc.
3	Média	Funcionalidade incorreta do componente ou processo. Existe um solução simples para o bug, se for de severidade 3.
4	Baixa	Erros de documentação ou erros de severidade 3.

Lista de Prioridades

Prioridade	Nível de Prioridade	Descrição
1	Obrigatório consertar	Esse bug deve ser consertado imediatamente, o programa não pode ser liberado com esse bug.
2	Deveria consertar	Estes são bugs importantes que devem ser corrigidos o mais rapidamente possível.
3	Consertar quando tiver tempo	O problema deve ser corrigido dentro do tempo disponível.
4	Prioridade baixa	Não é importante (neste momento) que esses erros sejam priorizados. Devem ser corrigidos após todos os outros erros terem sido fixados.

9 NECESSIDADES DO AMBIENTE E RECURSOS

9.1 Ferramentas de teste

Processo	Ferramenta
Criação de casos de teste	Microsoft Word
Execução de casos de teste	Manual
Gerenciamento de casos de teste	Microsoft Word
Gerenciamento de defeitos	Microsoft Word
Relatório de testes	PDF

9.2 Gerenciamento de configuração

- Gerenciamento de documentos: GitLab
- Gerenciamento de código: GitLab

9.3 Ambiente de testes

- Windows 10, Linux ou Max OS X: Python 3 e accesso ao GitLab