

Laudo de Desenvolvimento de Software

SISTEMA acessidoc

CURITIBA, PARANÁ – 2023

**SUMÁRIO**

[1. INFORMAÇÕES GERAIS 4](#_Toc152402257)

[2. INTRODUÇÃO 4](#_Toc152402258)

[3. INSPEÇÕES DE VERIFICAÇÃO E VALIDAÇÃO – V&V 4](#_Toc152402259)

[**3.1. Verificações de Código e Refatoração** 5](#_Toc152402260)

[**3.2. Verificações de Qualidade** 6](#_Toc152402261)

[**3.3. Testes de Segurança** 9](#_Toc152402262)

[**3.4. Testes de Requisitos** 10](#_Toc152402274)

[**3.5. Testes de Usuário** 11](#_Toc152402293)

[**3.6. Testes de Regressão** 12](#_Toc152402310)

[**3.7. Testes de Desempenho** 13](#_Toc152402315)

[4. Relatório Técnico de Desenvolvimento de Software 14](#_Toc152402331)

# 1. INFORMAÇÕES GERAIS

|  |
| --- |
| NOME DO PROJETO: ACESSIDOC |
| DATA DO LAUDO: |
| RESPONSÁVEL PELO LAUDO: DARLEY LEAL DOS SANTOS |
| VERSÃO DO SOFTWARE: VERSÃO 1.1 |

# 2. INTRODUÇÃO

Este laudo apresenta uma análise do projeto de construção do software de edição de documentos AcessiDOC, conduzido por dois estudantes: um do curso de Engenharia de Software e outro de Análise e Desenvolvimento de Sistemas. Essa iniciativa foi realizada em colaboração com as professoras Mariane Gavioli Bergamini e Luciane Yanase Hirabara Kanashiro da ESP – Escola Superior Politécnica, UNINTER. O laudo será baseado na documentação atualizada do software, visando inspeções de qualidade com base na norma IEEE 1012-2016 (Verificação e Validação).

# 3. INSPEÇÕES DE VERIFICAÇÃO E VALIDAÇÃO – V&V

A prática de inspeção visa garantir que o produto a ser desenvolvido passe por criteriosos processos de análise e validação do ciclo de vida do software, garantindo que o projeto seja entregue no prazo. Segundo a norma IEEE 1012-2016, p. 56 (adaptado pelo autor), se um determinado elemento do sistema não for capaz de atender a todos os seus requisitos alocados, os requisitos devem ser realocados. A **Verificação** (CMMI, 2010, p.401), garantir que os produtos de trabalho selecionados cumpram os seus requisitos especificados, ajudará a equipe a construir o software corretamente, analisando se todos os requisitos e funcionalidades documentados durante a elicitação foram devidamente implementados. A **Validação** (CMMI, 2010, p.393), demonstrar que o produto ou componente cumpre o uso pretendido quando colocado em seu ambiente desejado,avalia se o produto a ser entregue atende as expectativas dos stakeholders, ou seja, se atendem a regra de negócio de forma apropriada e consistente. Após a finalização de cada inspeção nos componentes do sistema, um breve relatório deve ser escrito baseado em etapas seguidas, anomalias encontradas, justificativas e com o mais alto padrão de qualidade exigido pelas boas práticas da engenharia de software moderna. Deve-se anexar ao fim do laudo, junto a assinatura do autor.

## **3.1. Verificações de Código e Refatoração**

O código fonte deve ser revisto com o objetivo de se identificar possíveis problemas como bugs, violações de padrões e problemas de desempenho, sua importância estar em entender a lógica e a estrutura do código.

Tabela 3.1.1 – Inspeção de código-fonte

|  |  |  |
| --- | --- | --- |
| **Atividade: Validar qualidade do produto segundo diretrizes do C#** | | |
| **TAREFAS DE V&V** | ENTRADA NECESSÁRIA | SAÍDA NECESSÁRIA |
| Nomenclaturas | Código-fonte. | Escolha de nomes claros e representativos. Padrões de nomenclatura seguindo boas práticas do .NET. |
| Princípios de OO | Adesão aos princípios de Orientação a Objetos, como: encapsulamento, herança, polimorfismo e abstração. | Classes coesas entre si, não possuam forte acoplamento e suas propriedades sejam corretamente encapsuladas para que cada componente trabalhe de forma independente e reduza a chance de erros. |
| Tratamento de Exceções | Uso apropriado de blocos try-catch. | Adequação do tratamento e lançamento de exceções personalizadas. |
| Refatoração | Segundo as Leis de Lehman, código deve ser escrito pensando na sua evolução para que futuras manutenções corretivas ou evolutivas se torne mais simples e vida útil do sistema seja maior. | Código-fonte manutenível, flexível e de fácil entendimento. |

## **3.2. Verificações de Qualidade**

Tabela 3.2.1 – Inspeção de qualidade no ciclo de vida do software

|  |  |  |
| --- | --- | --- |
| **PROCESSO** | **SITUAÇÃO** | **DESCRIÇÃO** |
| **Elicitação dos Requisitos** | Aprovado | Entendido o problema que precisa ser resolvido, detalhes sobre como os requisitos devem funcionar foram devidamente coletados e analisados, visando à compreensão da necessidade dos stakeholders. |
| **Refinamento dos Requisitos Funcionais** | Aprovado | Etapa de aprimoramento e refinamento dos requisitos, garantindo que eles atendam efetivamente ao escopo do projeto. Os requisitos serão priorizados com base em sua importância e viabilidade. |
| **Verificação dos Requisitos Não Funcionais** | Aprovado | Fase crítica, onde todos os requisitos especificados foram avaliados para garantir que estejam em conformidade e coerência entre si. |
| **Validação dos Requisitos do Sistema** |  | O software será desenvolvido usando a plataforma .NET e linguagem C# para a lógica de negócios e interações com o banco de dados no back-end, tecnologias front-end como HTML, CSS, Bootstrap e Javascript serão usados para renderizar e criar o conteúdo na camada de visualização. O PostgreSQL será usado como SGBD relacional para armazenar documentos e informações do usuário. |
| **Gerenciamento dos Requisitos** | Aprovado | Requisitos serão controlados às necessidades do projeto, evitando alterações não autorizadas que não tenham passado pela devida análise. |
| **Projeto de Arquitetura** | Aprovado | A Arquitetura MVC (Model-View-Controller), irá organizar a aplicação em três componentes interconectados. Isso é feito para separar as representações internas do sistema da maneira que a informação é apresentada e aceita ao usuário. |
| **Protótipo de Alta Fidelidade** | Aprovado | Protótipo foi desenvolvido usando o software Figma, seu intuito será simular o sistema com fluxos próximos do real para melhor entendimento da proposta de solução aos stakeholders. |
| **Gerenciamento da Qualidade** | Aprovado | Processos serão definidos para que o fluxo de trabalho e produtos estejam em conformidades aos padrões de qualidade estabelecidos pela engenharia de software. |
| **Gerenciamento da Configuração** | Aprovado | Será usado o Git para o versionamento de código e o GitHub como repositório, onde serão armazenados e gerenciados todos os artefatos desenvolvidos durante o ciclo de vida do AcessiDOC. |
| **Backlog do Produto** | Aprovado | Lista de tarefas será feita e priorizada de acordo às necessidades do produto e do cliente. As atividades serão organizadas em cards no sistema Trello usando a metodologia Kanban para gerenciar esse fluxo. |
| **Desenvolvimento** | Iniciado | Esse projeto seguirá um ciclo de desenvolvimento organizado em sprints de 2 semanas cada uma. Cada sprint abordará um subprojeto específico com seus respectivos requisitos e funcionalidades a serem implementados. |
| **Refatoração** | Iniciado | Todos os desenvolvedores devem refatorar o código encontrando melhorias sempre que possível, essa prática de melhoria contínua manterá o código simples e de fácil manutenção. |
| **Testes** | Iniciado | Antes da implantação, serão efetuados testes para verificar se o produto desenvolvido atende aos seus requisitos, é necessário que haja no mínimo um teste de caixa-preta e um teste de caixa-branca para cada requisito. |
| **Implantação** | Pendente | Implantar sistema e componentes, realizar testes de aceitação e homologar junto ao usuário. |
| **Reengenharia de Software** | Iniciado | Técnicas da engenharia reversa de software serão utilizados para analisar a estrutura do AcessiDOC de forma a abstrair as funcionalidades e operações. O pensamento analítico será importante para encontrar falhas e evoluir funcionalidades reduzindo impactos nas etapas de desenvolvimento e operações. O software é uma representação lógica do mundo real, ou seja, funcionalidades precisam evoluir, podendo incluir novos requisitos ou manutenções nos já existentes. |

## **3.3. Testes de Segurança**

O Teste de Segurança tem como objetivo garantir que o funcionamento da aplicação esteja exatamente como especificado. É verificado se o software se comporta adequadamente entre as mais diversas tentativas de acesso, visando possíveis vulnerabilidades. Para isso, testa-se todos os mecanismos de proteção na aplicação de fato a protegerão de acessos indevidos. Será verificado se apenas usuários registrados no SGBD poderão ser autenticados ao acesso do sistema.

Tabela 3.3.1 – Inspeção de segurança

|  |  |  |
| --- | --- | --- |
| Atividade: Caso de teste para realizar login de acesso | | |
| TAREFAS DE V&V | **ENTRADA NECESSÁRIA** | **SAÍDA NECESSÁRIA** |
| TST-01 | Não preencher todos os campos e clicar no botão “Entrar”. | Exibir mensagem solicitando preenchimento dos campos obrigatórios, permanecendo na interface de Login. |
| 1. TST-02 | Preencher todos os campos obrigatórios e acionar o botão “Entrar”. | Exibir tela inicial ao stakeholder. |
| 1. TST-03 | Preencher campos com dados inválidos ou em branco. | Exibir mensagem solicitando o preenchimento correto dos campos, deve-se manter a tela de login aberta. |

## **3.4. Testes de Requisitos**

Durante a inspeção, cada Requisito será examinado para garantir que ele esteja completo, além disso, também será verificado se cada requisito é testável, ou seja se as operações do sistema estão funcionando como desejado. Segundo Sommerville, 2019, p 221, o teste baseado em requisitos é uma abordagem sistemática para o projeto de casos de teste, em que se considera cada requisito a fim de derivar casos de teste – o objetivo é demonstrar que o sistema implementou seus requisitos corretamente.

3.4.1 – Inspeção de Requisitos Funcionais

|  |  |  |
| --- | --- | --- |
| Atividade: Caso de testes de requisitos | | |
| TAREFAS DE V&V | **ENTRADA NECESSÁRIA** | **SAÍDA NECESSÁRIA** |
| 1. RF001 - TST-01 | Permitir ao usuário ter uma opção para upload de arquivos. Serão aceitas as extensões PDF e DOC. Caso outro formato seja anexado um aviso deve retornar ao usuário. | Gerar novo documento de acordo a formato escolhido pelo usuário. |
| 1. RF002 - TST-02 | Formatar texto do documento usando a fonte Verdana. | Texto formatado de acordo a fonte escolhida pelo usuário. |
| 1. RF003 - TST-03 | Permitir ao usuário selecionar dois tamanhos de fonte: 18 para documento padrão ou 24 para super-ampliação do texto. | Texto ampliado de acordo ao tamanho escolhido pelo usuário. |
| 1. RF004 - TST-04 | Modificar o alinhamento do texto. | Alinhamento do texto de acordo as normas da ABNT. |
| 1. RF005 - TST-05 | Modificar o espaçamento do texto. | Texto com espaçamentos de acordo às normas ABNT. |
| 1. RF006 - TST-06 | O sistema precisa disponibilizar uma tela de início com a UI para entrada de login/senha. | Disponibilizar UI inicial ao usuário. |
| 1. RF007 - TST-07 | O sistema deve ampliar imagens em documentos. Deve-se verificar se imagens ficaram uma abaixo da outra, e se não ficaram borradas. | Imagens ampliadas. |

## **3.5. Testes de Usuário**

Segundo Sommerville, 2019 p. 225, é praticamente impossível que um desenvolvedor de sistema replique o ambiente de trabalho desse software, pois os testes no ambiente de desenvolvimento são inevitavelmente artificiais. Deve ser feito um teste de aceitação junto aos stakeholders para simular a sua usabilidade e decidir se o AcessiDOC e as suas operações atendem aos requisitos.

Tabela 3.5.1 – Inspeção de usabilidade do sistema

|  |  |  |
| --- | --- | --- |
| Atividade: Caso de testes de aceitação | | |
| TAREFAS DE V&V | **ENTRADA NECESSÁRIA** | **SAÍDA NECESSÁRIA** |
| TST-01 | Selecionar opção para criar uma prova. Escolher arquivo e tamanho desejado. | Sistema deve processar prova anexada e renderizar tela de visualização onde usuário poderá fazer download ou imprimir arquivo. |
| TST-02 | Selecionar opção para criar livro. Escolher arquivo e tamanho desejado. | Sistema deve processar livro anexado e renderizar tela de visualização onde usuário poderá fazer download ou imprimir arquivo. Verifica-se imagens se comportam como esperado e se texto está na formatação adequada. |
| TST-03 | Selecionar barra de pesquisa, digitar nome de arquivo. | Mostrar arquivo procurado, se não encontrar retornar mensagem: “Arquivo não encontrado”. |
| TST-04 | Verificar fluxo do aplicativo, interação com o usuário precisa ser fluída e intuitiva. | Sistema deve seguir um “fluxo lógico” parecido com outros softwares de edição, para que isso não cause estresse mental ao usuário, ou sua manipulação se torne complexa e pouco produtiva. |

## **3.6. Testes de Regressão**

Garantem que o software atenda aos requisitos mesmo depois de sofrer manutenções. Nesse processo são verificados se alterações não afetaram outras partes do sistema. Aqui pode ser usado técnicas da engenharia reversa de software para analisar e inspecionar as mudanças de forma lógica e abstrata, caso seja identificado falhas, deve-se “quebrá-las” em problemas menores que deve ser analisado “de cima para baixo”, ou seja, cada camada deve ser entendida para que solução seja pensada, para isso também é importante que classes sejam coesas entre si, não possuam forte acoplamento e suas propriedades sejam corretamente encapsuladas para que cada componente trabalhe de forma independente e reduza a chance de erros. Para os testes de regressão não serão criados casos de teste, eles serão executados de acordo ao andamento do projeto e outros testes foram feitos, será verificado, portanto, se após uma alteração ou implementação problemas não foram encontrados, caso seja identificado alguma irregularidade deve se analisar e rastrear a causa do problema.

Tabela 3.6.1 – Inspeção de manutenções corretivas

|  |  |  |
| --- | --- | --- |
| Atividade: Análise e referência a outros testes usando a Regressão | | |
| TAREFAS DE V&V | **ENTRADA NECESSÁRIA** | **SAÍDA NECESSÁRIA** |
|  |  |  |
|  |  |  |

## **3.7. Testes de Desempenho**

Segundo Sommerville, 2019 p. 224, teste de desempenho se preocupa tanto em mostrar que o sistema cumpre os requisitos tanto em descobrir problemas e defeitos do sistema. Uma maneira eficaz de descobri-los é projetar testes em torno dos limites do sistema. Isso significa estressar o software com demandas que estão fora de limites do projeto.

Tabela 3.7.1 – Inspeção de desempenho

|  |  |  |
| --- | --- | --- |
| Atividade: Caso de testes de desempenho | | |
| TAREFAS DE V&V | **ENTRADA NECESSÁRIA** | **SAÍDA NECESSÁRIA** |
| TST-01 | Verificar como o sistema se comporta ao tentar formatar arquivos grandes. Acima de 15MB, ou com muitas páginas exemplo de livros. | Duas possibilidades:  1. Sistema trava, ou fica lento sem conseguir prosseguir com o processo; 2. Sistema mantém seu fluxo normal e faz a formatação do arquivo; |
| TST-02 | Verificar desempenho na renderização das páginas. | Analisa-se telas estão se comportando como o esperado e se nenhuma tela apresenta irregularidades. |
| TST-03 | Verificar desempenho no momento de criação dos novos arquivos. | Analisa-se se sistema está gerando corretamente os novos arquivos. |
| TST-04 | Adicionar outros formatos não aceitos, exemplos: png, jpg, txt etc. | Sistema deve retornar mensagem de erro e alertar o usuário sobre extensões de arquivo corretas. |

# 4. Relatório Técnico de Desenvolvimento de Software

Informações dos Estudantes:

Nome: Darley Leal dos Santos

Instituição de Ensino: ESP - Escola Superior Politécnica, UNINTER

Curso: Bacharelado em Engenharia de Software

Posição Atual: Estagiário em Sistemas Educacionais, Grupo UNINTER

Afiliação: Membro Corporativo do CreaJr-PR

Contato: [darleyleal2103@outlook.com](mailto:darleyleal2103@outlook.com)

Nome: Rodrigo de Souza Machado

Instituição de Ensino: ESP - Escola Superior Politécnica, UNINTER

Curso: Tecnologia em Análise e Desenvolvimento de Sistemas

Posição Atual: Estagiário em Sistemas Educacionais, Grupo UNINTER

Contato: Cwbmachado@gmail.com

1. Objetivo do Software:

O software AcessiDOC foi desenvolvido com o objetivo de ajudar os profissionais no processo de adaptação de provas e livros didáticos que serão enviados às instituições acadêmicas e utilizados pelos estudantes com deficiência visual. As cores do sistema, vermelho, verde e branco possuem referências às cores das bengalas: [Você sabia que as bengalas utilizadas pelos deficientes tem cores diferentes para facilitar a identificação de suas deficiências? - Tupãense Notícias (tupaense.com.br)](https://www.tupaense.com.br/2021/08/23/voce-sabia-que-as-bengalas-utilizadas-pelos-deficientes-tem-cores-diferentes-para-facilitar-a-identificacao-de-suas-deficiencias/#:~:text=Confira%20qual%20o%20significado%20de%20cada%20cor%3A%201,Bengala%20vermelha%3A%20pessoas%20com%20cegueira%20total%20e%20surdez.)

2. Arquitetura do Software:

A arquitetura do software foi projetada levando em consideração os seguintes princípios:

Modularidade: O sistema está dividido em módulos independentes para facilitar a manutenção e escalabilidade.

Segurança: Foram implementadas medidas de segurança para proteger os dados e garantir a integridade do sistema.

Eficiência: O software foi otimizado para garantir um desempenho eficiente, mesmo em condições de carga elevada.

3. Checklist de Inspeção:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Checklist de Inspeção de Qualidade de Software** | | | | |
| **ITEM** | DATA | CONFORME | NÃO CONFORME | N/A |
| 1. Artefatos Versionados | 1. 26/11/2023 2. 02/12/2023 3. 10/12/2023 4. 15/12/2023 | X  X  X  X |  |  |
| 1. Documentação atualizada | 1. 30/11/2023 2. 02/12/2023 3. 10/12/2023 4. 15/12/2023 5. 19/12/2023 | X  X  X  X  X |  |  |
| 1. Inspeção de código-fonte | 1. 28/11/2023 2. 10/12/2023 3. 15/12/2023 | X  X | X |  |
| 1. Refatoração | 1. 10/12/2023 2. 15/12/2023 | X  X |  |  |
| 1. Verificação de Qualidade | 1. 30/11/2023 2. 02/12/2023 3. 10/12/2023 4. 15/12/2023 5. 19/12/2023 | X  X  X  X  X |  |  |
| 1. Teste de Segurança | 1. 15/12/2023 | X |  |  |
| 1. Teste de Requisitos |  |  |  |  |
| 1. Teste de Usuário |  |  |  |  |
| 1. Teste de Regressão | 1. 10/12/2023 | X |  |  |
| 1. Teste de Desempenho |  |  |  |  |

Este laudo foi revisado e aprovado por:

DARLEY LEAL DOS SANTOS,

Estudante de Engenharia de Software

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

