CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE JAHU

CURSO DE GRADUAÇÃO EM SISTEMAS PARA INTERNET

ANDERSON LUIZ DOS SANTOS ROSA

Sistema de Agendamento com Funcionalidades Automatizadas para Otimização do Tempo

Jahu, SP

2º semestre/2024

ANDERSON LUIZ DOS SANTOS ROSA

Sistema de Agendamento com Funcionalidades Automatizadas para Otimização do Tempo

Monografia apresentada como exigência para conclusão do curso de Graduação em Sistemas para Internet pela Faculdade de Tecnologia de Jahu – FATEC – JAHU, sob orientação do Prof. Alex Paulo Lopes Batista

Jahu, SP

2º semestre/2024

**AGRADECIMENTOS**

# **RESUMO**

.

Palavras-Chave:

# **ABSTRACT**

Keywords:

# **LISTA DE ILUSTRAÇÕES**

**FIGURAS**

# **LISTA DE TABELAS**

**TABELAS**

**1. Introdução**

No cenário atual, muitos profissionais autônomos ou empresas enfrentam dificuldades em organizar seus atendimentos de forma eficiente. O agendamento de compromissos, em muitos casos, ainda é realizado de maneira manual ou com ferramentas que não atendem de forma completa às necessidades específicas desses profissionais. Além disso, o tempo investido na organização e gestão de horários poderia ser mais bem aproveitado em atividades mais produtivas. Dessa forma, surge a necessidade de sistemas automatizados que possam otimizar esse processo e agregar valor ao dia a dia de quem necessita gerenciar atendimentos, como médicos, advogados, terapeutas, e outros profissionais de prestadores de serviços.

* Para o desenvolvimento desse sistema, algumas hipóteses podem ser formuladas:
* Um sistema de agendamento que utilize métodos automatizadores pode aumentar a eficiência dos profissionais, reduzindo o tempo gasto com a organização dos atendimentos.
* A automatização de tarefas como o envio de lembretes e confirmações de presença resultará em uma redução significativa de faltas e atrasos.
* Um sistema com interface intuitiva e que permita personalização de funcionalidades conforme a área de atuação do profissional terá maior adesão e satisfação dos usuários.
* A implementação de um sistema de agendamento com integração a outras plataformas de comunicação (como e-mail e SMS) facilitará a gestão de compromissos e melhorará a interação com os clientes.

O desenvolvimento de um sistema automatizado para agendamento não só proporcionará uma solução prática para problemas de organização e gestão de horários, mas também permitirá que os profissionais possam se concentrar em outras tarefas mais importantes. A automação é uma tendência crescente, e implementar um sistema que atenda a essa demanda se torna cada vez mais necessário no ambiente competitivo e dinâmico de trabalho.

**1.1. Objetivos**

**1.1.1. Objetivos gerais**

Desenvolver um sistema de agendamento automatizado que atenda às necessidades de organização de profissionais de diferentes áreas, permitindo maior eficiência no gerenciamento de atendimentos.

**1.1.2. Objetivos específicos**

Criar uma interface simples e intuitiva para o usuário, permitindo a personalização do sistema de acordo com a área de atuação.

Implementar funcionalidades automatizadas, como envio de lembretes de compromissos e notificações de cancelamento, visando reduzir o tempo gasto com processos manuais.

Desenvolver um mecanismo de integração do sistema com outras plataformas de comunicação, como e-mails e aplicativos de mensagens, para melhorar a interação com os clientes.

Avaliar a eficácia do sistema em termos de economia de tempo e aumento da produtividade, através de testes com usuários reais.

Garantir que o sistema possa ser adaptado para diferentes tipos de profissionais, considerando suas necessidades e particularidades no agendamento de atendimentos.

**1.2. Justificativas**

A justificativa para o desenvolvimento de um sistema de agendamento automatizado está diretamente relacionada à economia de tempo e à melhoria na eficiência operacional dos profissionais. Com base em observações e conhecimento prático do cotidiano de profissionais que dependem de agendamentos para sua rotina, verificou-se que grande parte do tempo é gasto na organização manual dos atendimentos. Isso leva a uma sobrecarga de tarefas, diminuindo o tempo disponível para outras atividades essenciais. Assim, o desenvolvimento de uma aplicação que automatize o processo de agendamento não só aumentaria a produtividade, como também evitaria erros humanos e melhoraria a experiência do cliente ao garantir que os horários sejam geridos de forma eficiente.

**1.3. Problematização**

O problema a ser abordado neste trabalho é a falta de ferramentas de agendamento automatizadas e eficientes que possam atender a profissionais que precisam otimizar a organização de seus atendimentos. Muitos dos sistemas disponíveis no mercado possuem limitações, como interfaces complexas, falta de funcionalidades personalizáveis e pouca integração com outros sistemas que o profissional já possa utilizar. Além disso, a ausência de automação em processos simples, como o envio de lembretes ou notificações de cancelamento, resulta em perda de tempo e, em alguns casos, na insatisfação dos clientes. Portanto, há uma demanda real por um sistema que automatize esses processos e se adeque às necessidades específicas de cada área de atuação.

**1.4. Metodologia da pesquisa**

A seção de \*\*Metodologia\*\* de um TCC que trata do desenvolvimento de um sistema deve detalhar como a pesquisa será conduzida e como o sistema será desenvolvido. A metodologia precisa explicar os métodos, técnicas, ferramentas e processos que serão usados no desenvolvimento do sistema e na coleta de dados (caso seja necessário para a pesquisa). Abaixo, vou sugerir um modelo que você pode seguir e adaptar ao seu caso específico.

### Estrutura sugerida para a seção de Metodologia:

1. \*\*Introdução à Metodologia\*\*

- Faça uma breve introdução explicando o objetivo da metodologia: como você pretende desenvolver o sistema e como a pesquisa será conduzida.

- Explique qual é a importância da metodologia no contexto do seu TCC e o que o leitor pode esperar dessa seção.

2. \*\*Tipo de Pesquisa\*\*

- \*\*Pesquisa Aplicada\*\*: Se o objetivo é o desenvolvimento de um sistema que será utilizado na prática.

- \*\*Pesquisa Qualitativa/Quantitativa\*\*: Caso você colete dados de usuários ou faça testes, explique qual tipo de pesquisa você usará para coletar e analisar os dados.

- \*\*Pesquisa Exploratória/Descritiva\*\*: Caso você explore um problema específico ou descreva processos do sistema.

Exemplo: "A pesquisa é de caráter aplicado, com enfoque qualitativo, pois visa desenvolver um sistema de gestão para pequenas empresas e analisar como ele pode melhorar a eficiência operacional."

3. \*\*Método de Desenvolvimento\*\*

- Explique o \*\*modelo de desenvolvimento de software\*\* adotado para o projeto. Algumas opções são:

- \*\*Modelo Cascata (Waterfall)\*\*: Se o desenvolvimento será sequencial e bem estruturado.

- \*\*Modelo Ágil (Agile)\*\*: Caso o desenvolvimento seja iterativo, como no Scrum ou Kanban, em que o produto é desenvolvido em ciclos curtos e com entregas incrementais.

- \*\*Prototipação\*\*: Caso você crie protótipos do sistema para coletar feedback durante o desenvolvimento.

Exemplo: "O desenvolvimento do sistema será realizado utilizando a metodologia ágil Scrum, com ciclos de duas semanas para entregas incrementais. Cada sprint será focada em uma funcionalidade específica do sistema."

4. \*\*Fases do Desenvolvimento\*\*

- \*\*Planejamento e Levantamento de Requisitos\*\*: Definir o escopo do sistema, as funcionalidades principais, os requisitos técnicos e as necessidades dos usuários. Explique como você irá levantar esses requisitos, seja por entrevistas, questionários, ou análise documental.

- \*\*Análise e Projeto\*\*: Descrever como será feito o design do sistema, incluindo a arquitetura do software, diagramas de fluxo de dados, e protótipos de interface.

- \*\*Desenvolvimento e Implementação\*\*: Especifique como o código será implementado, que linguagens de programação e ferramentas serão utilizadas.

- \*\*Testes\*\*: Explique como será realizada a validação do sistema, por meio de testes de funcionalidade, testes de usabilidade e/ou testes de desempenho. É importante falar sobre o tipo de teste (unitário, integração, etc.).

- \*\*Implantação e Avaliação\*\*: Descreva como o sistema será implantado no ambiente real e como será feita a avaliação de sua efetividade. Isso pode incluir feedback de usuários ou métricas de desempenho.

Exemplo: "O desenvolvimento será dividido nas seguintes fases: levantamento de requisitos, onde serão coletadas informações por meio de entrevistas com stakeholders; análise e projeto, utilizando a UML para modelagem do sistema; desenvolvimento, onde serão utilizadas as linguagens Python e JavaScript; testes de funcionalidade, usabilidade e integração; e implantação, com a implementação do sistema em ambiente de produção."

5. \*\*Ferramentas e Tecnologias Utilizadas\*\*

- Explique as \*\*ferramentas de desenvolvimento\*\* (IDE, frameworks, bibliotecas) que serão utilizadas.

- Mencione as \*\*tecnologias\*\* específicas, como linguagens de programação, banco de dados, servidores web, etc.

- Caso esteja utilizando alguma ferramenta para testes ou coleta de dados, mencione também.

Exemplo: "Para o desenvolvimento do sistema, serão utilizadas as seguintes tecnologias: a linguagem de programação JavaScript com o framework React para o front-end e Node.js para o back-end. O banco de dados será MySQL e o sistema será hospedado na plataforma AWS."

6. \*\*Coleta de Dados (se aplicável)\*\*

- Caso sua pesquisa envolva a coleta de dados (como pesquisas com usuários ou testes de sistema), explique como isso será feito.

- Quais métodos você usará para coletar os dados (entrevistas, questionários, observação direta)?

- Como os dados serão analisados (análise qualitativa, quantitativa, análise de conteúdo)?

Exemplo: "Será realizada uma pesquisa com usuários do sistema, por meio de questionários, para avaliar a usabilidade e a eficácia do sistema em termos de eficiência e satisfação. A análise será feita com base em respostas qualitativas e quantitativas."

7. \*\*Cronograma\*\*

- Apresente um cronograma com as principais etapas do desenvolvimento e pesquisa.

- Estabeleça os prazos para cada fase, como levantamento de requisitos, desenvolvimento, testes, etc.

Exemplo: "O cronograma está estruturado da seguinte forma: levantamento de requisitos - mês 1; desenvolvimento e implementação - meses 2 a 4; testes e ajustes - mês 5; implantação e avaliação - mês 6."

8. \*\*Considerações Finais sobre a Metodologia\*\*

- Encerre a seção destacando a importância da abordagem metodológica adotada para garantir a qualidade e a eficácia do sistema desenvolvido.

### Exemplo de Texto de Metodologia:

> "Este trabalho adota a metodologia ágil Scrum para o desenvolvimento de um sistema de gestão de projetos. A pesquisa será de caráter aplicado e qualitativo, com o objetivo de desenvolver um protótipo funcional do sistema e avaliar sua usabilidade. O projeto será conduzido em sprints de duas semanas, nas quais serão entregues funcionalidades incrementais, sempre com base no feedback dos stakeholders. A primeira fase consistirá no levantamento de requisitos, com entrevistas e análise de documentos; em seguida, será realizada a modelagem do sistema utilizando diagramas UML. Durante o desenvolvimento, serão utilizadas as tecnologias React para o front-end e Node.js para o back-end. O sistema será testado de acordo com critérios de funcionalidade e usabilidade, e a avaliação será feita por meio de questionários aplicados a usuários finais, com foco na satisfação e na eficiência operacional. O cronograma do projeto prevê um período de seis meses, com cada fase do desenvolvimento bem delimitada."

---

Essa estrutura pode ser adaptada dependendo do escopo e do foco específico do seu TCC. Lembre-se de que a metodologia deve ser clara e objetiva, permitindo que outros pesquisadores ou desenvolvedores possam entender e reproduzir os passos tomados no seu trabalho.

**1.5. Estrutura do trabalho**

**2. Revisão bibliográfica**

Quando você está desenvolvendo um projeto de sistema e precisa de referências bibliográficas, pode se basear em diversos tipos de fontes que cobrem aspectos técnicos, metodológicos e conceituais do desenvolvimento de software. Aqui estão algumas categorias de fontes que você pode usar:

### 1. \*\*Metodologias de Desenvolvimento de Software\*\*

- \*\*Livros sobre metodologias ágeis (Agile), Scrum, Kanban, DevOps e outras abordagens\*\*. Estes textos fornecem uma base para o gerenciamento de projetos e a entrega contínua de software.

- \*\*Exemplo\*\*:

- "Scrum: A Arte de Fazer o Dobro de Trabalho na Metade do Tempo" – Jeff Sutherland.

- "The Phoenix Project" – Gene Kim, Kevin Behr, George Spafford (para DevOps).

### 2. \*\*Engenharia de Software\*\*

- \*\*Livros sobre engenharia de software\*\* são fundamentais para entender as melhores práticas de desenvolvimento, incluindo análise de requisitos, design de sistemas, testes e manutenção de software.

- \*\*Exemplo\*\*:

- "Engenharia de Software: Uma Abordagem Prática" – Ian Sommerville.

- "Software Engineering" – Roger S. Pressman.

### 3. \*\*Modelagem e Arquitetura de Software\*\*

- \*\*Livros sobre a modelagem e arquitetura de software\*\* (como UML, diagramas de classes, diagramas de sequência, arquitetura de microserviços, entre outros) são cruciais para entender a estrutura de sistemas complexos.

- \*\*Exemplo\*\*:

- "Design Patterns: Elements of Reusable Object-Oriented Software" – Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.

- "Patterns of Enterprise Application Architecture" – Martin Fowler.

### 4. \*\*Programação e Linguagens de Programação\*\*

- \*\*Textos sobre linguagens de programação específicas\*\* (como Java, Python, C++, etc.), estruturas de dados, algoritmos e técnicas de otimização de código também são essenciais.

- \*\*Exemplo\*\*:

- "Clean Code: A Handbook of Agile Software Craftsmanship" – Robert C. Martin.

- "The Pragmatic Programmer" – Andrew Hunt e David Thomas.

### 5. \*\*Testes de Software\*\*

- \*\*Livros sobre testes de software\*\* discutem técnicas de testes unitários, testes de integração, testes funcionais, testes automatizados, etc.

- \*\*Exemplo\*\*:

- "Test-Driven Development: By Example" – Kent Beck.

- "Continuous Delivery" – Jez Humble, David Farley (aborda testes em um contexto de integração contínua).

### 6. \*\*Gerenciamento de Projetos de Software\*\*

- \*\*Textos sobre gestão de projetos de software\*\* podem ajudar a entender como organizar, planejar e gerenciar o ciclo de vida do projeto.

- \*\*Exemplo\*\*:

- "Software Engineering Project Management" – Richard H. Thayer.

- "Project Management for Information Systems" – James Cadle, Donald Yeates, and John Kendall.

### 7. \*\*Tecnologias e Ferramentas Específicas\*\*

- \*\*Documentações e manuais sobre ferramentas\*\* específicas que você está utilizando no desenvolvimento do sistema, como frameworks de front-end (React, Angular), back-end (Node.js, Django), bancos de dados (MySQL, MongoDB), ou sistemas de controle de versão (Git).

- \*\*Exemplo\*\*:

- "Pro Git" – Scott Chacon e Ben Straub.

- Documentação oficial do framework ou linguagem que você estiver utilizando.

### 8. \*\*Segurança de Software\*\*

- \*\*Livros sobre segurança de software\*\* são essenciais para garantir que o sistema seja seguro e esteja protegido contra vulnerabilidades.

- \*\*Exemplo\*\*:

- "The Web Application Hacker's Handbook" – Dafydd Stuttard, Marcus Pinto.

- "Secure Programming Cookbook for C and C++" – John Viega, Matt Messier.

### 9. \*\*Design de Interface do Usuário (UI) e Experiência do Usuário (UX)\*\*

- \*\*Textos que abordam o design de interface e experiência do usuário\*\* ajudam a entender como criar interfaces que sejam fáceis de usar e visualmente agradáveis.

- \*\*Exemplo\*\*:

- "Don't Make Me Think" – Steve Krug.

- "The Design of Everyday Things" – Don Norman.

### 10. \*\*Pesquisas e Artigos Acadêmicos\*\*

- \*\*Artigos e papers acadêmicos\*\* que abordam questões específicas do desenvolvimento de sistemas, como otimização de algoritmos, novas tecnologias, e inovações na engenharia de software.

- Você pode buscar artigos em repositórios como:

- \*\*IEEE Xplore\*\*

- \*\*ACM Digital Library\*\*

- \*\*Google Scholar\*\*

### 11. \*\*Livros sobre Computação em Nuvem e Sistemas Distribuídos\*\*

- Se o seu sistema for baseado em arquitetura de nuvem ou sistemas distribuídos, pode ser útil referenciar livros especializados nesses tópicos.

- \*\*Exemplo\*\*:

- "Designing Data-Intensive Applications" – Martin Kleppmann.

- "Cloud Native Patterns: Designing change-tolerant software" – Cornelia Davis.

### 12. \*\*Documentação e Whitepapers de Tecnologias Específicas\*\*

- A documentação de tecnologias que você está utilizando pode ser uma excelente fonte de referência, especialmente para detalhes técnicos e boas práticas.

- Exemplo: Documentação do \*\*Docker\*\*, \*\*Kubernetes\*\*, \*\*AWS\*\*, \*\*Google Cloud\*\*, \*\*Microsoft Azure\*\*, etc.

### Dicas Gerais para a Escolha de Referências:

- \*\*Relevância\*\*: Verifique se o material é diretamente aplicável ao seu contexto de desenvolvimento.

- \*\*Autoridade\*\*: Prefira autores reconhecidos na área de engenharia de software, design de sistemas, ou nas tecnologias que você está utilizando.

- \*\*Atualidade\*\*: Certifique-se de que as fontes são atuais ou ainda são relevantes para a tecnologia utilizada no projeto, especialmente em áreas como desenvolvimento web, segurança, e nuvem.

Com esses tipos de fontes, você pode construir uma base sólida de referências para o desenvolvimento de sistemas de forma bem fundamentada e estruturada.

**2.1.**

**2.1.1.**

**2.1.2.**

**2.1.3.**

**2.1.4.**

**2.1.5.**

**2.1.6.**

**3. Modelo de negócios**

**3.1. Canvas**

**3.2. O que será elaborado**

**3.3. Para quem será elaborado**

**3.4. Como será elaborado**

**3.5. Quanto custará**

**4. Documentação**

**4.1. Declaração de abrangência do projeto**

**4.1.1. Requisitos funcionais**

Para expor os requisitos funcionais do seu projeto de forma clara e eficiente, você pode seguir alguns passos fundamentais. Os requisitos funcionais descrevem as funcionalidades específicas que o sistema deve ter para atender às necessidades dos usuários e aos objetivos do projeto. Abaixo, estão os passos que você pode seguir para documentá-los de forma estruturada:

### 1. \*\*Introdução ao Documento\*\*

- \*\*Objetivo\*\*: Explique de forma breve o objetivo do documento, ou seja, qual é a finalidade dos requisitos funcionais no contexto do projeto.

- \*\*Escopo\*\*: Defina qual parte do sistema ou projeto será abordada neste documento, para que os leitores saibam qual é o escopo.

Exemplo:

\*Este documento descreve os requisitos funcionais do sistema de gestão de estoque, abordando funcionalidades como cadastro de produtos, controle de entradas e saídas, e geração de relatórios.\*

---

### 2. \*\*Identificação dos Requisitos Funcionais\*\*

Cada requisito funcional deve ser descrito de maneira clara, com um formato padronizado. Para cada requisito, inclua as seguintes informações:

- \*\*Identificador único\*\* (código ou número sequencial para cada requisito, ex: RF-001)

- \*\*Descrição\*\*: Descreva a funcionalidade de forma detalhada, explicando o que o sistema deve fazer.

- \*\*Prioridade\*\*: Defina a importância do requisito (alta, média, baixa).

- \*\*Fonte\*\*: Caso necessário, indique de onde vem a exigência (cliente, regulamento, pesquisa de mercado, etc.).

Exemplo:

- \*\*RF-001: Cadastro de Produto\*\*

- \*\*Descrição\*\*: O sistema deve permitir o cadastro de novos produtos, com os seguintes dados: nome, código, descrição, preço, quantidade em estoque e categoria.

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Requisito do cliente

---

### 3. \*\*Organização dos Requisitos Funcionais\*\*

Organize os requisitos em categorias ou módulos, se aplicável. Isso facilita a leitura e compreensão, especialmente se o sistema for grande.

Exemplos de categorias:

- \*\*Cadastro de Produtos\*\*

- \*\*Controle de Estoque\*\*

- \*\*Relatórios\*\*

- \*\*Gestão de Usuários\*\*

Você pode criar subcategorias ou requisitos mais específicos dentro dessas áreas. Por exemplo, para a categoria "Cadastro de Produtos", você pode incluir requisitos como:

- \*\*Cadastro Básico\*\*

- \*\*Edição e Exclusão de Produtos\*\*

- \*\*Pesquisa e Filtros de Produtos\*\*

---

### 4. \*\*Requisitos Funcionais Detalhados\*\*

Para cada requisito funcional, inclua detalhes como:

- \*\*Fluxo de Trabalho\*\*: Se o requisito envolver interações ou processos sequenciais, descreva o fluxo de como o sistema vai funcionar, seja em formato de texto ou diagrama (ex: fluxogramas, diagramas de casos de uso).

- \*\*Entradas e Saídas\*\*: Se houver interação com o usuário ou outros sistemas, defina as entradas necessárias (informações que o usuário precisa fornecer) e as saídas esperadas (resultados ou ações que o sistema vai gerar).

- \*\*Regras de Negócio\*\*: Se o requisito depender de alguma regra específica de negócio, ela deve ser claramente descrita.

- \*\*Restrições\*\*: Caso haja limitações ou restrições, como segurança, compatibilidade ou conformidade com regulamentações, descreva-as.

Exemplo:

- \*\*RF-002: Controle de Estoque\*\*

- \*\*Descrição\*\*: O sistema deve permitir o controle de entradas e saídas de produtos no estoque. Cada movimentação deve ser registrada com data, hora, quantidade e responsável pela ação.

- \*\*Fluxo de Trabalho\*\*:

1. O usuário informa o tipo de movimentação (entrada ou saída).

2. O sistema solicita a quantidade e o produto.

3. O sistema atualiza o estoque e registra o evento.

- \*\*Entradas\*\*: Produto, quantidade, tipo de movimentação, responsável.

- \*\*Saídas\*\*: Atualização do estoque, registro da movimentação.

- \*\*Regras de Negócio\*\*: Não é possível registrar uma saída de produto se a quantidade disponível for inferior à quantidade solicitada.

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Necessidade do cliente

---

### 5. \*\*Validação e Testes\*\*

Inclua critérios de aceitação ou testes que ajudarão a validar se o requisito foi implementado corretamente. Isso é importante para garantir que o sistema atende às expectativas do cliente.

Exemplo:

- \*\*Critérios de Aceitação\*\*: O sistema deve permitir a entrada de produtos no estoque. A quantidade total de estoque deve ser atualizada corretamente após cada entrada.

---

### 6. \*\*Rastreabilidade\*\*

Se for necessário, mostre como cada requisito está relacionado a outros documentos do projeto, como casos de uso, diagramas de fluxo, ou requisitos não funcionais. Isso ajuda a garantir que todos os aspectos do sistema foram cobertos.

---

### Exemplo de Estrutura Final para um Requisito Funcional:

1. \*\*RF-001: Cadastro de Produto\*\*

- \*\*Descrição\*\*: O sistema permitirá o cadastro de produtos, incluindo informações como nome, código, preço e quantidade.

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Cliente

- \*\*Fluxo de Trabalho\*\*:

1. O usuário acessa a tela de cadastro de produtos.

2. Preenche os campos obrigatórios.

3. Clica em "Salvar" para registrar o produto.

- \*\*Entradas\*\*: Nome do produto, código, preço, quantidade.

- \*\*Saídas\*\*: Produto cadastrado no sistema.

- \*\*Regras de Negócio\*\*: O código do produto deve ser único.

- \*\*Critérios de Aceitação\*\*: O sistema deve confirmar o cadastro com uma mensagem de sucesso.

---

### Dicas Importantes:

- \*\*Seja claro e objetivo\*\*: Evite termos vagos e faça com que cada requisito seja compreensível.

- \*\*Use uma linguagem consistente\*\*: Isso ajuda a evitar ambiguidades.

- \*\*Documente as dependências\*\*: Caso um requisito dependa de outro para ser executado, isso deve ser informado.

- \*\*Valide com a equipe\*\*: Revise os requisitos com a equipe técnica e com o cliente para garantir que estão completos e corretos.

Com essa estrutura, seus requisitos funcionais serão claros, compreensíveis e prontos para serem seguidos na fase de desenvolvimento do projeto.

**4.1.2. Requisitos não funcionais**

Os \*\*requisitos não funcionais\*\* descrevem as características, atributos ou qualidades que o sistema deve ter, mas que não estão diretamente relacionados a funcionalidades específicas. Eles abordam o \*\*como\*\* o sistema deve operar, em vez de \*\*o que\*\* ele deve fazer. Esses requisitos são essenciais para garantir que o sistema atenda a expectativas de desempenho, segurança, usabilidade, entre outros aspectos.

Aqui estão algumas das principais categorias de requisitos não funcionais e como você pode estruturá-los em seu documento:

### Estrutura para Documentação de Requisitos Não Funcionais

1. \*\*Introdução\*\*

- \*\*Objetivo\*\*: Explicar o papel dos requisitos não funcionais no contexto do projeto. Eles garantem que o sistema tenha qualidades desejáveis como desempenho, segurança, usabilidade, etc.

- \*\*Escopo\*\*: Defina a área ou módulo do sistema ao qual os requisitos não funcionais se aplicam.

---

### 2. \*\*Categorias de Requisitos Não Funcionais\*\*

Você pode organizar os requisitos não funcionais em várias categorias. Abaixo estão algumas das mais comuns:

#### 2.1 \*\*Desempenho\*\*

- Relacionados a velocidade, capacidade de resposta, e eficiência do sistema.

\*\*Exemplo:\*\*

- \*\*RNF-001: Tempo de Resposta\*\*

- \*\*Descrição\*\*: O sistema deve processar e exibir os resultados de uma pesquisa em até 2 segundos.

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Requisitos do cliente

- \*\*Critérios de Aceitação\*\*: A pesquisa deve ser realizada em até 2 segundos na maior parte das consultas realizadas, mesmo com um banco de dados contendo 1 milhão de registros.

#### 2.2 \*\*Segurança\*\*

- Refere-se à proteção contra falhas de segurança, como acesso não autorizado, perda de dados ou ataques externos.

\*\*Exemplo:\*\*

- \*\*RNF-002: Criptografia de Dados\*\*

- \*\*Descrição\*\*: Todos os dados sensíveis, como senhas e informações financeiras, devem ser criptografados durante a transmissão e armazenamento.

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Regulamentações de segurança

- \*\*Critérios de Aceitação\*\*: Dados de usuários devem ser criptografados usando o algoritmo AES-256.

#### 2.3 \*\*Usabilidade\*\*

- Relacionado à facilidade de uso do sistema, interface amigável e aprendizado rápido.

\*\*Exemplo:\*\*

- \*\*RNF-003: Interface do Usuário\*\*

- \*\*Descrição\*\*: A interface do sistema deve ser intuitiva e permitir que um usuário iniciante execute tarefas básicas após 10 minutos de treinamento.

- \*\*Prioridade\*\*: Média

- \*\*Fonte\*\*: Requisitos do cliente

- \*\*Critérios de Aceitação\*\*: A interface deve passar por testes com usuários novatos, que devem conseguir realizar as principais funções sem auxílio.

#### 2.4 \*\*Escalabilidade\*\*

- Refere-se à capacidade do sistema de se adaptar ao aumento da carga de trabalho, como mais usuários ou volume de dados.

\*\*Exemplo:\*\*

- \*\*RNF-004: Escalabilidade\*\*

- \*\*Descrição\*\*: O sistema deve ser capaz de suportar 10.000 usuários simultâneos sem degradação de performance significativa.

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Necessidade de crescimento do cliente

- \*\*Critérios de Aceitação\*\*: O sistema deve ser testado em ambiente de estresse com 10.000 usuários simultâneos para garantir que o tempo de resposta permaneça abaixo de 3 segundos.

#### 2.5 \*\*Disponibilidade\*\*

- Relacionado ao tempo em que o sistema está disponível para uso, sem falhas ou interrupções.

\*\*Exemplo:\*\*

- \*\*RNF-005: Alta Disponibilidade\*\*

- \*\*Descrição\*\*: O sistema deve ter uma disponibilidade mínima de 99,9% durante o horário comercial (9h-18h).

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Contrato de SLA

- \*\*Critérios de Aceitação\*\*: O sistema deve ser monitorado e seu tempo de inatividade não pode exceder 8 horas por ano.

#### 2.6 \*\*Manutenibilidade\*\*

- Relacionada à facilidade com que o sistema pode ser mantido ou atualizado ao longo do tempo.

\*\*Exemplo:\*\*

- \*\*RNF-006: Manutenibilidade\*\*

- \*\*Descrição\*\*: O código-fonte do sistema deve ser modular e documentado, de modo que qualquer desenvolvedor possa realizar modificações ou correções com um esforço mínimo.

- \*\*Prioridade\*\*: Média

- \*\*Fonte\*\*: Requisitos de qualidade de software

- \*\*Critérios de Aceitação\*\*: O código deve seguir as boas práticas de design de software e deve ser revisado por uma equipe de desenvolvedores.

#### 2.7 \*\*Compatibilidade\*\*

- Refere-se à capacidade do sistema de funcionar em diferentes plataformas, dispositivos ou com outros sistemas.

\*\*Exemplo:\*\*

- \*\*RNF-007: Compatibilidade com Navegadores\*\*

- \*\*Descrição\*\*: O sistema deve ser compatível com as versões mais recentes dos navegadores Chrome, Firefox e Safari.

- \*\*Prioridade\*\*: Média

- \*\*Fonte\*\*: Requisitos do cliente

- \*\*Critérios de Aceitação\*\*: O sistema deve ser testado em pelo menos 3 versões de cada navegador citado e funcionar sem falhas.

#### 2.8 \*\*Localização (i18n) e Internacionalização (l10n)\*\*

- Referem-se à capacidade de adaptação do sistema para diferentes idiomas, regiões e culturas.

\*\*Exemplo:\*\*

- \*\*RNF-008: Suporte a Idiomas\*\*

- \*\*Descrição\*\*: O sistema deve ser capaz de exibir textos em pelo menos 3 idiomas: português, inglês e espanhol.

- \*\*Prioridade\*\*: Baixa

- \*\*Fonte\*\*: Requisitos do cliente

- \*\*Critérios de Aceitação\*\*: A interface do usuário deve ser capaz de exibir os textos no idioma selecionado sem erros de tradução ou formatação.

#### 2.9 \*\*Eficiência de Recursos\*\*

- Relacionado à otimização do uso de recursos do sistema, como memória, CPU e rede.

\*\*Exemplo:\*\*

- \*\*RNF-009: Uso de Memória\*\*

- \*\*Descrição\*\*: O sistema não deve consumir mais do que 500 MB de memória durante a execução de processos típicos.

- \*\*Prioridade\*\*: Média

- \*\*Fonte\*\*: Requisitos de desempenho

- \*\*Critérios de Aceitação\*\*: O sistema deve ser monitorado e não ultrapassar 500 MB de uso de memória durante operações de rotina.

#### 2.10 \*\*Conformidade\*\*

- Refere-se à aderência a normas, regulamentos ou políticas externas, como normas de segurança ou regulamentos da indústria.

\*\*Exemplo:\*\*

- \*\*RNF-010: Conformidade com GDPR\*\*

- \*\*Descrição\*\*: O sistema deve estar em conformidade com o Regulamento Geral de Proteção de Dados (GDPR) para garantir a privacidade dos dados dos usuários.

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Regulamento legal

- \*\*Critérios de Aceitação\*\*: O sistema deve garantir que os dados pessoais sejam processados de acordo com as exigências do GDPR, como consentimento explícito do usuário.

---

### 3. \*\*Estrutura de Exposição\*\*

Assim como os requisitos funcionais, cada requisito não funcional deve ser documentado de forma clara e concisa. Use uma estrutura padrão que inclua:

- \*\*Identificador único\*\* (ex: RNF-001, RNF-002, etc.)

- \*\*Descrição do Requisito\*\*

- \*\*Prioridade\*\* (Alta, Média, Baixa)

- \*\*Fonte\*\* (Cliente, Regulamentação, Requisitos técnicos, etc.)

- \*\*Critérios de Aceitação\*\* (Como saber se o requisito foi atendido)

---

### Dicas Importantes para Requisitos Não Funcionais

- \*\*Seja específico e mensurável\*\*: Como esses requisitos envolvem características de qualidade, é importante definir parâmetros mensuráveis para que possam ser avaliados objetivamente.

- \*\*Priorize de acordo com a necessidade\*\*: Embora todos sejam importantes, você pode ter requisitos não funcionais com maior prioridade dependendo do contexto do projeto (por exemplo, segurança pode ser mais importante que usabilidade em um sistema bancário).

- \*\*Considere o contexto de uso do sistema\*\*: O desempenho pode ser mais exigente em um sistema de e-commerce de grande porte do que em um aplicativo interno de pequenas empresas.

### Exemplo de Estrutura Completa:

1. \*\*RNF-001: Tempo de Resposta\*\*

- \*\*Descrição\*\*: O sistema deve processar consultas e exibir os resultados em até 2 segundos.

- \*\*Prioridade\*\*: Alta

- \*\*Fonte\*\*: Requisitos do cliente

- \*\*Critérios de Aceitação\*\*: A resposta a qualquer consulta simples no sistema deve ser de no máximo 2 segundos, medida em 95% das tentativas.

Seguindo esse formato, você poderá documentar de forma clara os requisitos não funcionais e garantir que eles sejam atendidos durante o desenvolvimento do sistema.

**4.1.3. Casos de uso**

O tópico "Casos de Uso" é uma parte essencial de um trabalho de TCC, especialmente quando você está desenvolvendo um sistema, pois descreve as interações entre os usuários (ou outros sistemas) e o sistema que está sendo projetado. Ele serve para ilustrar como o sistema será utilizado na prática, ou seja, quais são as funcionalidades que ele oferece e como os usuários irão interagir com ele.

### O que colocar no tópico "Casos de Uso":

1. \*\*Identificação dos Atores\*\*: Antes de detalhar os casos de uso, é importante identificar quem são os "atores" do sistema. Os atores podem ser usuários, sistemas ou entidades externas que interagem com o sistema. Por exemplo, em um sistema de gerenciamento de biblioteca, os atores podem ser "Usuário", "Bibliotecário", "Sistema de Pagamento", etc.

2. \*\*Descrição dos Casos de Uso\*\*: Cada caso de uso representa uma funcionalidade do sistema e descreve como os atores interagem com o sistema para atingir um objetivo específico. A descrição geralmente segue uma estrutura como:

- \*\*Nome do Caso de Uso\*\*: Um título claro e objetivo.

- \*\*Objetivo\*\*: Qual o objetivo do caso de uso.

- \*\*Ator(es)\*\*: Quais os usuários ou sistemas envolvidos.

- \*\*Pré-condições\*\*: O que precisa ser verdade antes do caso de uso acontecer.

- \*\*Fluxo Principal\*\*: Descrição passo a passo da interação entre o ator e o sistema.

- \*\*Fluxos Alternativos\*\*: Outras possibilidades que podem ocorrer durante o caso de uso.

- \*\*Pós-condições\*\*: O que ocorre no sistema após a execução do caso de uso.

3. \*\*Diagrama de Casos de Uso\*\*: Um diagrama visual pode ser incluído, representando os atores e os casos de uso de forma simplificada. Ele ajuda a visualizar rapidamente as interações principais do sistema.

### Exemplo de Caso de Uso

Imaginando que você esteja desenvolvendo um sistema de \*\*gestão de biblioteca\*\*, um caso de uso poderia ser o seguinte:

#### Caso de Uso: \*\*Empréstimo de Livro\*\*

- \*\*Objetivo\*\*: Permitir que o usuário empreste um livro na biblioteca.

- \*\*Ator(es)\*\*: Usuário, Sistema de Biblioteca

- \*\*Pré-condições\*\*: O usuário deve estar registrado no sistema. O livro deve estar disponível para empréstimo.

- \*\*Fluxo Principal\*\*:

1. O usuário acessa o sistema de biblioteca.

2. O usuário pesquisa o livro desejado.

3. O sistema verifica a disponibilidade do livro.

4. O sistema apresenta as informações do livro, incluindo disponibilidade.

5. O usuário escolhe o livro para empréstimo e confirma.

6. O sistema registra o empréstimo, atualiza o status do livro e informa ao usuário sobre o prazo de devolução.

7. O usuário recebe uma confirmação do empréstimo.

- \*\*Fluxos Alternativos\*\*:

- \*\*Livro Indisponível\*\*: Se o livro não estiver disponível, o sistema sugere a opção de reservar o livro ou procurar por outro.

- \*\*Usuário Não Registrado\*\*: Se o usuário não estiver registrado no sistema, ele será solicitado a fazer o cadastro antes de prosseguir.

- \*\*Pós-condições\*\*: O livro está marcado como "emprestado" no sistema, e o usuário tem o prazo de devolução informado.

### Considerações Finais

Ao criar os casos de uso, você deve se concentrar em \*\*como o usuário vai interagir com o sistema\*\* para atingir seus objetivos. Cada caso de uso deve ser claro, conciso e fácil de entender. Além disso, o diagrama de casos de uso ajuda a fornecer uma visão geral das funcionalidades do sistema.

Se você tiver outras dúvidas ou precisar de mais exemplos específicos, posso ajudar!

**4.1.4. Conceitos**

**4.2. Manual do usuário**

**5. Considerações finais**

**Referências**