**Engenharia de Software**

Especificação de Sistema

| **Professor** | RADAMÉS PEREIRA |
| --- | --- |

2023-02

Especificação

A engenharia de software é uma disciplina que se preocupa com todos os aspectos da produção de software, desde a concepção até a manutenção do sistema depois de estar em operação. A especificação de requisitos e a modelagem são etapas fundamentais no processo de desenvolvimento.

1. Especificação de Requisitos:

- Definição: É o processo de definir, documentar e manter as necessidades e restrições de um sistema de software. Os requisitos descrevem o que o sistema deve fazer (requisitos funcionais) e as restrições sob as quais ele deve operar (requisitos não-funcionais).

- Importância: A especificação clara e completa dos requisitos é essencial para garantir que o sistema desenvolvido atenda às necessidades do usuário e para evitar retrabalho e custos adicionais.

- Componentes comuns:

1. Requisitos Funcionais: Descrevem as funções que o sistema deve realizar. Por exemplo, "O sistema deve permitir que um usuário efetue uma venda e receba um pagamento”.
2. Requisitos Não-funcionais: Relacionam-se com características de qualidade do sistema, como desempenho, segurança, usabilidade, etc. Por exemplo, "O sistema deve ser capaz de suportar 10.000 vendas simultâneas".

2. Modelagem em Análise e Projeto de Sistemas:

- Definição: A modelagem envolve a criação de representações abstratas do sistema que está sendo desenvolvido. Essas representações podem ser usadas para entender, comunicar e documentar como o sistema irá funcionar, bem como para orientar sua construção.

- Importância: A modelagem ajuda a garantir que todos os stakeholders tenham uma compreensão clara e compartilhada do sistema proposto e de como ele se encaixa no ambiente operacional.

- Componentes comuns:

1. Modelos de Caso de Uso: Descrevem interações entre usuários (atores) e o sistema.
2. Diagramas de Classe: Representam as classes no sistema e suas relações.
3. Diagramas de Sequência: Mostram como os objetos no sistema interagem para realizar uma função específica.
4. Diagramas de Estado: Representam os estados pelos quais um objeto ou componente pode passar e os eventos que causam as mudanças de estado.
5. Diagramas de Atividade: Representam os fluxos de trabalho ou processos dentro do sistema.

Ambas as etapas, especificação de requisitos e modelagem, estão interligadas.

Uma especificação clara dos requisitos informa e orienta a modelagem, e as representações gráficas ou modelos ajudam a comunicar e validar os requisitos com os stakeholders. A combinação dessas atividades contribui para um desenvolvimento de software mais eficaz e de alta qualidade.

Atributos de uma boa Especificação de Requisitos de Projeto de Software:

* Clareza
* Não Ambígua
* Completa
* Simples
* Bem escrita

**Checklist Geral/Roteiro:**

**1 - Obtenha uma descrição textual (relatório de requisitos/especificação: documentos, entrevistas, questionários, observações, fórmulas, cursos, palestras, reuniões, etc…).**

Exemplo:

**2 - Organizar as informações, aplicar a engenharia de requisitos. Usar a decomposição.**

**3 - Modelagem Visual (simplificação da realidade: “um modelo vale por mil palavras”. Usa-se a abstração (seleção/retirada daquilo que é importante, o que interessa).**

**3.1 - Usar ferramentas diagramáticas e visuais, com uma linguagem. A linguagem padrão que usamos é a UML/SysML e BPM. (**[**www.omg.org**](http://www.omg.org)**);**

**3.1.1 - Selecionar, ou decompor os conceitos, coisas, objetos, classes, representados por substantivos, ou características do Domínio do Problema/Universo de Discurso - UD para a especificação.**

**UNIVERSIDADE COMUNITÁRIA REGIONAL DE CHAPECÓ - UNOCHAPECÓ**

**ESCOLA POLITÉCNICA**

Título

...

Equipe (engenheiro e desenvolvedores):

**...**

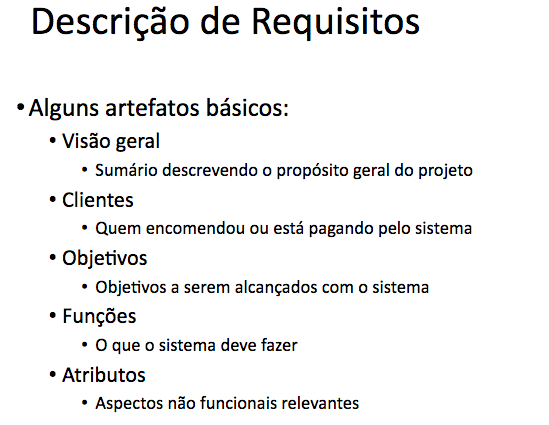
**...**

Local e data

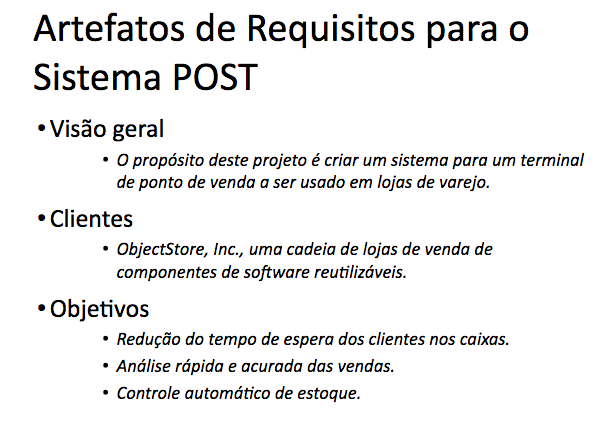
Introdução

## Resumo do Projeto (Descrição textual)

Descreve-se aqui o sistema a ser desenvolvido.



Exemplo:



## Plataforma de desenvolvimento (O equipamento dos desenvolvedores e ferramentas de software)

Descreve-se aqui uma primeira visão das tecnologias para desenvolvimento do projeto de software.

## Plataforma de operação (O equipamento do cliente/usuário do sistema)

Descreve-se aqui uma primeira visão das tecnologias para operacionalização.

## Definições e siglas (quaisquer siglas utilizadas no domínio, do vocabulário do usuário)

Descreve-se aqui a definição de todas as siglas, abreviações e termos usados.

## Perspectiva do produto

### Modos de operação (Meios de acesso ao sistema/Arquitetura do sistema)

### Identificam-se aqui os modos requeridos de operação, tais como: Back-End-Front-End, Móvel, Stand-Alone, ...

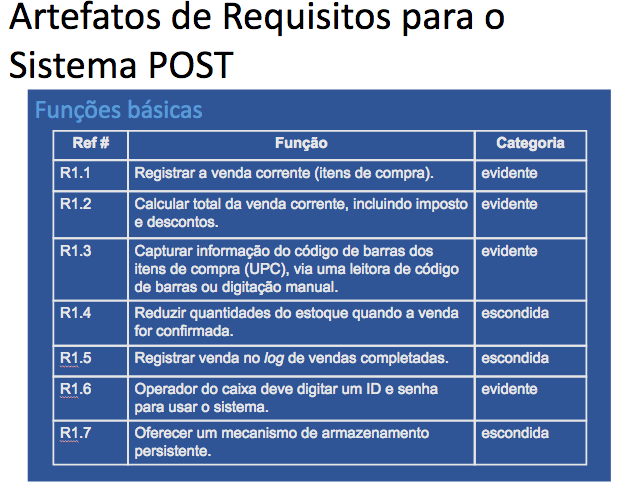
### Requisitos de adaptação ao ambiente (Aspectos legais para aderência a legislação, ambiente de operação crítica como indústria, automação, protocolos de comunicação específicos)

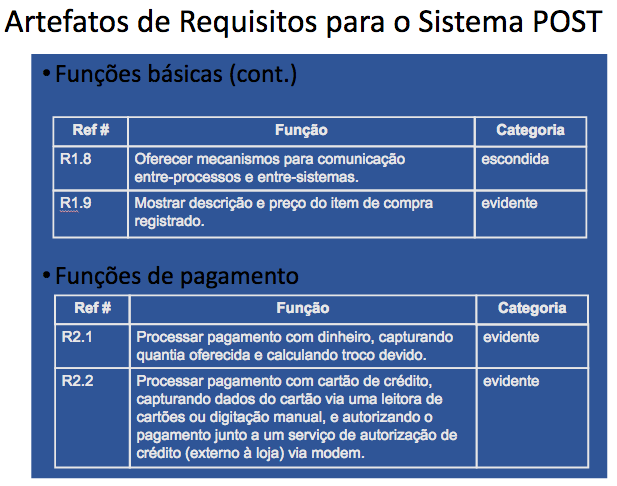
Definem-se aqui possíveis requisitos de adaptação do produto aos ambientes particulares onde ele será implantado. Por exemplo, parâmetros e métodos de configuração requeridos para ambientes específicos devem ser descritos aqui.

| Número de ordem | Requisito | Detalhes |
| --- | --- | --- |
| 1 | Configuração de ticket de venda e da Nota Fiscal Eletrônica | Configuração dos campos de formulário com interface responsiva. |

## Funções do produto (Funções básicas, R1.1 ..., R1.2 … No geral, o que o sistema deve fazer?)

Identificam-se aqui as principais funções que o produto desempenhará, descrevendo de forma sintética o objetivo de cada uma.





***Mais exemplos:***

Funções Básicas do sistema:

R1.1 – Controle de materiais: monitoramento dos ingredientes e materiais necessários para a produção de alimentos, garantindo a qualidade dos produtos em estreita colaboração com os fornecedores.

R1.2 – Garantia de qualidade e eficiência dos produtos finais: monitoramento do processo de produção e garantia de que todos os ingredientes e materiais estejam dentro dos padrões exigidos.

R1.3 – Compra, armazenamento e controle de materiais: compras de materiais e ingredientes e gerenciar o armazenamento e controle deles.

R1.4 – Produzir Alimentos: coordenar a utilização dos ingredientes e materiais no processo de fabricação dos alimentos.

R1.5 – Controlar a produção com otimização do uso de materiais: trabalhar com os líderes de cada linha de produção para planejar e otimizar a utilização dos materiais.

R1.6 – Controlar entrega de produtos e preferência de clientes: coordenar a entrega dos produtos acabados aos clientes, manter um registro de todas as vendas realizadas e fornecer informações sobre a demanda do mercado e as preferências dos clientes.

R1.7 - Negociação com fornecedores: negociar os melhores preços com fornecedores para reduzir os custos de produção.

R1.8 - Conformidade com normas de segurança alimentar e regulamentações governamentais: garantir que a produção esteja em conformidade com as normas de segurança alimentar e regulamentações governamentais.

R1.9 - Planejamento de compras futuras: planejar futuras compras de ingredientes e materiais com base nas informações sobre a demanda do mercado e as preferências dos clientes.

## Características dos usuários (identificação da formação dos usuários para suas especialidades no uso do sistema, tal como ensino-médio, graduação, especialista, engenheiro, doutor, etc...)

Descrevem-se aqui as principais características dos grupos de usuários esperados para o produto, tais como cargo ou função, permissão de acesso, frequência de uso, nível de instrução, proficiência no processo de negócio e proficiência em informática.

## Restrições (Lei Geral de proteção de dados, etc...)

Descrevem-se aqui aspectos técnicos e gerenciais que possam limitar as opções dos desenvolvedores, tais como restrições legais.

## Hipóteses de trabalho (Sistema operacional, versão de ferramentas de software, licenças de bibliotecas e de sub-sistemas)

Descrevem-se aqui fatores que não são restrições limitativas do desempenho, como na subseção anterior, mas fatores cuja alteração requer modificações na ER, como, por exemplo, versão a ser utilizada do ambiente operacional ou plataforma de desenvolvimento.

# Requisitos específicos

## Interfaces externas

### Visão geral (Tipos de interface, relatórios, gráficos, dashboards)

Descreve-se aqui, de forma detalhada, todas as entradas e saídas do produto.

### Requisitos para interfaces gráficas de usuário (mokups/wireframes)

Sugere-se, no caso de interfaces gráficas, a inclusão dos seguintes elementos:

* Um esboço do layout gráfico sugerido para a interface;
* Uma descrição dos relacionamentos com outras interfaces;
* Um diagrama de estados/atividades, caso necessário para melhor entender-se o comportamento requerido da interface;
* Uma lista dos campos de dados da interface;
* Uma lista dos comandos da interface;
* BPM;

## Requisitos funcionais

### Diagramas de casos de uso (Modelo UML de Casos de Uso)

Incluir todos os casos de uso que se pretende implementar em uma liberação. Pode-se incluir ainda: um certo caso de uso e seus relacionamentos, todos os casos de uso para um certo ator.

### Fluxos dos casos de uso (Casos de Uso Expandidos e Diagramas de Atividades)

* Pré-condições para a realização do caso de uso;
* Fluxo principal do caso de uso (sucesso), descrito na forma de uma sequência de passos;
* Fluxos alternativos do caso de uso;
* Descrições mais formais, como diagramas de estado ou de atividade, se a complexidade do caso de uso exigir;
* Observações.

## Requisitos não-funcionais

### Requisitos de desempenho (Velocidade de banda, tempo de resposta das interfaces e/ou impressão de relatórios)

Requisitos de desempenho devem ser especificados de forma quantitativa e mensurável.

### Requisitos de dados persistentes (Sistemas de gerenciamento de banco de dados e modelo de persistência)

Descrevem-se aqui estruturas lógicas de dados persistentes (que mantém seu valor após a execução do programa) que sejam usadas pelo produto. Cada estrutura de dados pode ser, por exemplo, um arquivo convencional ou uma tabela em um banco de dados.

INCLUIR AQUI O MODELO DE BANCO DE DADOS

### Restrições ao desenho

Restrições de projeto impostas por padrões externos, com influência da legislação, tipo legislação fiscal municipal, estadual e federal.

### Atributos de Qualidade

Indica os atributos de qualidade, seguindo as características e subcaracterísticas recomendadas pela norma [ISO-9126](https://drive.google.com/open?id=14-OfxrvT5pD4sblFt234r8NPBNwt2HmG), ou mais atual.

## Objetos/Classes

### Modelo Conceitual/Classes de Análise/Modelo de Domínio (Classes, Associações, nomes das associações, Multiplicidades e Atributos)

### DSS – Diagramas de Sequência do Sistema (Eventos e Operações) de Casos de Uso (denotando as mensagens entre os objetos do domínio para atender ao Caso de Uso).

### Contratos (das Operações do DSS)

### Classes de Implementação - Diagrama de Classes (Classes, Associações, nomes das associações, Multiplicidades, Atributos e Métodos). Atribuição de responsabilidades com GRASP (General Responsibility Assignment Software Patterns) que são um conjunto de princípios e diretrizes para atribuição de responsabilidades em projetos de software orientados a objetos.

# Análise de UCP

As tabelas de escopo de valor do produto e tempo de desenvolvimento com Use Case Points - UCP. (COCOMO, Planning Poker e APF).

Referências:

*IEEE Std. 830 – 1993. IEEE Recommended Practice for Software Requirements Specifications.*

*IEEE ISO/IEC/IEEE 29148 – 2011. IEEE Systems and software engineering — Life cycle processes — Requirements engineering*

**OBSERVAÇÃO: Os itens deste modelo de especificação, recomendado pela IEEE, poderão ser complementados com novos itens caso sejam justificáveis.**

**INTRODUÇÃO A FERRAMENTA DE GERÊNCIA DE CONFIGURAÇÃO PARA ENGENHARIA DE SOFTWARE**

O GitHub é uma plataforma de hospedagem de código e de projetos que utiliza o sistema de controle de versão Git. Uma funcionalidade popular do GitHub é o README.md, um arquivo formatado em Markdown que é geralmente utilizado para fornecer uma descrição amigável do repositório.

Conceitos:

1. Markdown (MD): É uma linguagem de marcação leve que pode ser convertida em HTML. É fácil de escrever e ler. No GitHub, os arquivos `.md` são automaticamente renderizados para mostrar um formato mais agradável visualmente.

2. README.md: Este é o arquivo que geralmente está na raiz de um repositório e é a primeira coisa que alguém vê quando visita um repositório no GitHub. É uma prática recomendada incluir informações como: o que o projeto faz, como configurá-lo, como contribuir e qualquer outra informação relevante.

**EXERCÍCIOS**

**1. Criar uma conta e um repositório no GitHub.**

**Aqui estão os passos detalhados:**

* + Criar uma Conta no GitHub:

1. Acesse [GitHub.com](https://github.com/).

2. Clique em “Sign up” no canto superior direito.

3. Preencha seus detalhes: nome de usuário, endereço de e-mail e senha.

4. Escolha o tipo de plano que deseja. Para projetos de código aberto, você pode escolher o plano gratuito.

5. Confirme seu endereço de e-mail clicando no link que o GitHub enviará para você.

* + Criar um Repositório:

1. Depois de fazer login na sua conta do GitHub, clique no ícone "+" no canto superior direito e selecione "New repository" (Novo repositório).

2. Nomeie seu repositório como `EngSoftI`

3. Escolha se você quer que seu repositório seja "Public" (Público) ou "Private" (Privado).

4. Marque a caixa "Initialize this repository with a README" para adicionar um arquivo README.md ao seu repositório.

5. Clique em “Create repository” (Criar repositório).

* + Editar o README.md:

Para documentações simples, diretas e que devem ser versionadas juntamente com o código, o Markdown dentro do repositório pode ser ideal.

1. Na página principal do seu repositório, você verá o arquivo README.md listado.

2. Clique no arquivo `README.md`.

3. Clique no ícone de lápis no canto superior direito do visualizador de arquivo para começar a editar.

4. Escreva ou edite o conteúdo usando a linguagem de marcação Markdown.

Por exemplo:

# EngSoftI

Bem-vindo ao projeto da disciplina EngSoftI!

## Descrição

Este repositório contém todos os materiais e projetos relacionados à disciplina EngSoftI.

## Outros informes

5. Depois de fazer suas edições, desça até a parte inferior da página. Lá, você verá uma seção chamada "Commit changes". Dê uma breve descrição do que você fez no campo "Commit changes", por exemplo, "Inicializando README".

6. Clique em “Commit changes” para salvar suas modificações.

**Markdown**

Editando online: Você pode editar o README.md diretamente no GitHub clicando no arquivo e depois no ícone de lápis (editar) no canto superior direito.

Escrevendo em **Markdown**: Algumas formatações básicas que você pode querer usar incluem:

- Títulos: Use `#` para o título principal, `##` para subtítulos, `###` para sub-subtítulos, e assim por diante.

- Listas: Use `\*` ou `-` para itens de lista não ordenados. Use números seguidos de um ponto para itens de lista ordenados.

- Links: `[texto do link](URL)`

- Imagens: `![texto alternativo](URL da imagem)`

- Código: Use acentos graves (`` ` ``) ao redor do código inline e três acentos graves (`` ``` ``) para blocos de código.

Previews: Ao editar no GitHub, você pode clicar em "Preview changes" para ver como suas mudanças aparecerão depois de renderizadas.

Ao utilizar o Markdown no README.md do GitHub, você pode fornecer informações claras e bem formatadas sobre seu projeto, tornando-o mais acessível e compreensível para outros desenvolvedores e visitantes.

**Referências completas adicionais:**

[**https://github.com/mende1/guia-definitivo-de-markdown**](https://github.com/mende1/guia-definitivo-de-markdown)

[**https://docs.github.com/pt/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax**](https://docs.github.com/pt/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax)

* + Crie o Wiki com a especificação do Projeto:

(Sua especificação com as definicões e modelos aqui)

Ex:

| **UNIVERSIDADE COMUNITÁRIA REGIONAL DE CHAPECÓ - UNOCHAPECÓ**  **ESCOLA POLITÉCNICA**  Título  ...  Equipe (engenheiro e desenvolvedores):  **...**  **...**    Local e data    Introdução Resumo do Projeto (Descrição textual) Descreve-se aqui o sistema a ser desenvolvido.  .  .  . |
| --- |

<https://docs.github.com/pt/communities/documenting-your-project-with-wikis/adding-or-editing-wiki-pages>

Pronto! Agora você tem uma conta no GitHub com um repositório chamado `EngSoftI` e um arquivo README.md inicializado, editado e Wiki da especificação do projeto criada.

**2. Compartilhar seu repositório do GitHub:**

1. Forneça acesso direto ao repositório:

Se você deseja dar a alguém (como seu professor) a capacidade de colaborar diretamente no repositório (ou seja, fazer alterações, criar branches, etc.), siga estes passos:

1. Vá ao seu repositório no GitHub.

2. Clique na guia "Settings" (Configurações).

3. No menu à esquerda, clique em "Manage access" (Gerenciar acesso).

4. Clique no botão "Invite a collaborator" (Convidar um colaborador).

5. Digite o nome de usuário do GitHub ou o endereço de e-mail (`**radamesp7@gmail.com**`) e selecione o usuário apropriado dos resultados da pesquisa.

6. Clique no botão "Add (nome de usuário) to (nome do repositório)" (Adicionar (nome de usuário) a (nome do repositório)).

O professor receberá um e-mail com o convite para colaborar no repositório e precisará aceitar o convite para começar a colaborar.

Desde que seu repositório seja público, qualquer pessoa com o link poderá visualizá-lo. Se o repositório for privado, somente colaboradores autorizados ou pessoas com quem você compartilhou o acesso poderão visualizá-lo.

**Verifique as Permissões de acesso:**

Lembre-se de que quando você convida alguém como colaborador, está dando a essa pessoa permissões para editar e fazer alterações no repositório.