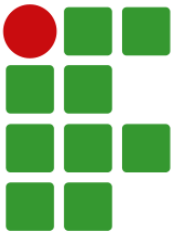

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rondônia

# Documento de Especificação de Requisitos de Software

## Nome do sistema

**Versão 1.0.0**

Nome do autor 1 <autor.um@ifro.edu.br>
Nome autor 2 <autor.dois@ifro.edu.br>

**JUNHO 2016**

# Resumo

Este documento apresenta orientações para a elaboração de projetos de sistemas a serem desenvolvidos pelo Instituto Federal de Rondônia – IFRO. Toda formatação deverá seguir estritamente às normas vigentes da ABNT para projetos de pesquisa quanto às fontes, margens, espaçamentos, elementos pré-textuais, textuais e pós-textuais.

Para usuários LaTeX, utilizamos a classe abnTeX2que contém todas as funções necessárias para a construção do documento. A documentação completa é encontrada em ⟨https://github.com/abntex/abntex2/wiki/Download⟩ e pode ser utilizada como apoio na elaboração do documento. Entretanto, para estar em conformidade com padrões únicos do IFRO, baixe o arquivo de customização em ⟨https://github.com/ifroariquemes/derstex⟩ para adequar os padrões de capa e estruturas.

**Palavras-chave**: template. dersex. ifro.

**** Esta página de resumo não é necessária no documento ****

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| ABNT | Associação Brasileira de Normas Técnicas |
| abnTeX | ABsurdas Normas para LaTeX |
| dersTeX | Documento de Especificação de Requisitos de Software para LaTeX |

# Sumário

# Introdução

A introdução serve de ponto inicial para o leitor compreender de forma rápida o problema abordado e as motivações que levaram à criação do sistema proposto. Deve ficar muito claro o método atual utilizado no processo sendo tratado (se a forma atual é manual, se há um outro sistema) e enfatizar os pontos mais críticos em que os usuários ou executantes do processo atual sentem mais dificuldades.

O projetista deve atentar para a definição de um escopo contendo resumidamente as funções do novo sistema e quais dos problemas atuais o software **NÃO** vai resolver, isto é, que ficarão pendentes para trabalhos futuros, versões futuras ou mesmo aqueles que por algum motivo não possuem desenvolvimento viável (neste caso explicar os critérios que elegeram a inviabilidade).

Os últimos parágrafos devem resumir a estrutura que se seguirá no documento adiantando ao leitor que assuntos serão abordados em cada capítulo.

Portanto, recomendamos que a introdução contemple os seguintes itens:

- problemática;
- motivações;
- método atual;
- pontos críticos;
- escopo do sistema proposto;
- e estrutura do restante do documento.

# 1 Descrição Geral

No início deste capítulo, demonstre os resultados esperados com a utilização do sistema proposto. Insista na solução dos problemas críticos enfrentados pelos usuários e outros fatores que visem a valorização do produto.

**NÃO** utilize esse espaço para:

- características tecnológicas como plataforma, arquitetura, e softwares de terceiros;
- escopo restritivo (dizer o que o software não fará);
- e assuntos relacionados a qualquer seção deste capítulo.

## 1.1 Características dos Usuários

Defina quem são os usuários em potencial do sistema. Nomeie genericamente pela atribuição e papel de cada um na organização e descreva a atuação deles no processo atual e em que momento o novo sistema entrará em cena. Demonstre como a utilização e integração de todos os envolvidos é pertinente para o bom funcionamento do sistema, e a posição de cada um no programa.

## 1.2 Restrições tecnológicas

Nesta seção você deve impor as tecnologias necessárias para a construção e implantação do sistema tais como plataformas, dispositivos e requisitos mínimos de hardware. Além disto serão expostos requerimentos de software envolvidos, como máquinas virtuais, bancos dados e softwares de terceiros. Caso uma implementação específica não seja necessária mas sim um atributo tecnológico, como, por exemplo, design responsivo ou banco de dados orientado a objeto, declarar a necessidade desta deixando aberto ao desenvolvedor a escolha da melhor forma de entregar esses requisitos.

## 1.3 Restrições funcionais

Aqui serão desenroladas as funções que não serão implementadas pelo software apesar de necessárias para a solução do problema em questão. Os motivos para a inviabilidade devem ser demonstrados com clareza.

## 1.4 Pressupostos e Dependências

Caso seja necessário, esta seção deve ser utilizada para dissertar sobre hipóteses em qualquer nível, operacional ou pessoal, que possa atrapalhar o bom andamento do desenvolvimento do produto, como a aprovação de um superior, participação de usuários durante testes e qualquer outro cenário que possa vir a se tornar um risco para o funcionamento do sistema e à execução do projeto.

# 2 Requisitos Específicos

Através das observações realizadas quanto às demandas solicitadas, levando em consideração aquilo que o departamento demandante necessita e quanto as informações gerenciadas afetam na organização, seguirá aqui a escrita de um capítulo para descrição dos requisitos necessários para satisfazer os pedidos realizados. Para cada requisito funcional elegido, haverá obrigatoriamente ao menos um requisito não-funcional e opcionalmente requisitos de interface.

Um *requisito funcional* representa uma função generalizada que o software terá de desempenhar; *requisitos não-funcionais* definem as regras que regerão determinadas funções do sistema como validações, questões de segurança, atributos para persistência e regras de negócio; enquanto que cada *requisito de interface* indica uma necessidade identificada pelo projetista como sendo essencial para a composição da interação humano-computador – IHC (principalmente nos casos em que se deseja causar menor impacto visual ao criar formulários com design semelhante àquele que os usuário já estão familiarizados).

Cada requisito funcional será identificado nos documentos pelas letras RF seguidas de um numeral sequencial e único: RF1, RF2, ..., RFn.

Os requisitos não-funcionais e de interface serão identificados pelas siglas RNF e RI respectivamente, seguidas pelo índice do RF ao qual se referencia e depois por numeral sequencial e único: RNF1.1, RNF1.2, ..., RNF1.n; RI1.1, RI1.2, ..., RI1.n.

O nome dos RF devem iniciar por verbos no infinitivo (terminados em ar, er, ir) e possuir contexto único entre todos os RF do sistema:

- RF1 - Cadastro de aulas (inválido, primeira palavra é um substantivo)
- RF2 - Cadastrar cliente (válido)
- RF3 - Alterar cadastro do cliente (inválido, contexto que trata de “cadastro do cliente” já existe)

A tabela 1 mostra um modelo de tabela montada para cada RF elicitado. No decorrer do projeto, caso seja necessário referenciar qualquer item, RF, RNF ou RI, toda a cadeia de caracteres será utilizada como identificação única, sigla e números.

Tabela 1 – Exemplo de tabela descritiva de requisito funcional

| **RF1 - Nome do requisito funcional**<br>**Descrição:** Texto descrevendo o contexto do requisito funcional. | |
|---|---|
| **Requisitos não-funcionais** | |
| RNF1.1 | Requisito não-funcional 1.1 |
| RNF1.2 | Requisito não-funcional 1.2 |
| RNF1.3 | Requisito não-funcional 1.3 |
| **Requisitos de interface** | |
| RI1.1 | Requisito de interface 1.1 |
| RI1.2 | Requisito de interface 1.2 |
| RI1.3 | Requisito de interface 1.3 |

# 3 Diagramas

O uso de diagramas é essencial para a compreensão profunda tanto do problema quanto da solução sendo desenvolvida. Em informática é comum recorrer ao Diagrama Entidade-Relacionamento (DER) e/ou à Linguagem de Modelagem Unificada (UML) durante o projeto de sistemas. Para elaboração deste documento de especificação, ambas as técnicas podem ser invocadas, sendo que a utilização de DER torna obrigatória a aplicação de um dicionário de dados, enquanto que ao usar UML requeremos ao menos um diagrama de Caso de Uso e sua descrição formal. A versão da especificação utilizada para diagramação deste projeto deve ser UML 2.3[1].

A seleção de quaisquer outros digramas é opcional e válida na composição deste documento, desde que devidamente descritos e que sejam relevantes ao projeto entregando novas perspectivas que, sem estes, seria mais complexo ou mesmo impossível de se perceber.

## 3.1 Casos de Uso

O digrama de caso de uso deve contemplar as funções globais do sistema fazendo alusão aos requisitos funcionais levantados no capítulo anterior. Cada RF deve gerar ao menos um caso de uso (UC) os quais devem estar dispostos e disponíveis aos atores exatamente como demandado pelos RNF. Processos fora dos limites do sistema (system boundary) também podem ser modelados se o projetista julgar necessário. Um exemplo pode ser visualizado na figura 1.

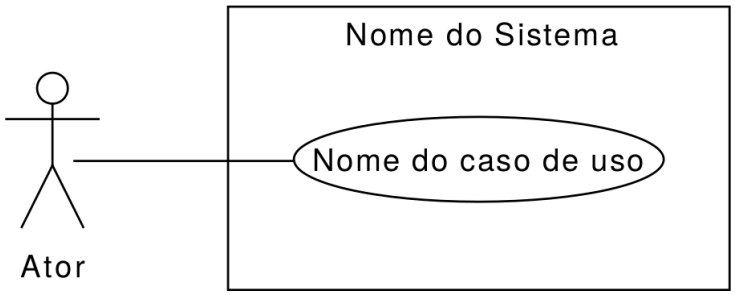


Figura 1 – Exemplo de diagrama de caso de uso

Assim como nos requisitos, UC também possuem um número de identificação sequencial e único utilizado em sua descrição formal e referenciamento no documento,

[1] Os documentos que especificam a UML 2.3 podem ser encontrados em ⟨http://www.omg.org/spec/UML/2.3/⟩.

porém não aparecem nos digramas. As regras de nomenclatura seguem as mesmas que de RF. O início da descrição de cada UC deve ser efetuado em uma nova página.

Uma das características abordadas pelos UC é a prioridade de desenvolvimento por função. Mantenha com prioridade mais alta os primeiros módulos a serem programados, como aqueles que são dependência de outros. Veja na tabela 2 os valores de prioridades.

A frequência de uso de um UC pode variar inclusive inversamente à sua prioridade de desenvolvimento dependendo do escopo do projeto. Tomando como exemplo um site de notícias, o cadastro da notícia feito pelo administrador é menos utilizados que a visualização da notícia feita pelos usuários públicos. Uma vez que o cadastro é realizado uma vez, a visualização é feita milhares de vezes, ou seja, para o sistema a frequência de uso do cadastro é baixa, enquanto que a visualização tem frequência alta. Para atribuição de uma frequência de uso razoável, deve-se pensar num período entre uma semana e um ano de uso normal do sistema e inferir tempos de uso a cada função, então utilize os itens da tabela 3 para determinar o uso.

Tabela 2 – Valores de prioridades de desenvolvimento para casos de uso

| **Prioridade** | **Descrição** |
|---|---|
| 1 - Urgente | Função extremamente necessária para o desenvolvimento de partes críticas do sistema, existe dependência de sua existência para que funções críticas possam ser criadas |
| 2 - Crítica | Função que se refere a um dos problemas críticos enfrentados pelos usuários, devem ser estar no topo das funções a implementar e disponibilizadas para produção o mais rápido possível |
| 3 - Importante | Função importante na solução de problemas críticos que deve constar ainda nas versões iniciais do sistema |
| 4 - Relevante | Função relevante para regras de negócio triviais, deve estar no sistema até à versão final |
| 5 - Desejável | Função sem muita importância para o sistema como um todo, sua ausência não afetará na solução dos problemas críticos, porém seu desenvolvimento é desejável para um futuro ainda indeterminado |

Tabela 3 – Valores de frequência de uso para casos de uso

| **Frequência** | **Descrição** |
|---|---|
| Alta | Utilização maior que 15% do tempo |
| Média | Utilização de 5% a 15% do tempo |
| Baixa | Utilização de até 5% do tempo |

### 3.1.1 UC1 - Nome do Caso de Uso

**Objetivo:** Descreva o objetivo do caso de uso.

**Requisitos:** Identifique os requisitos aos quais este UC está relacionado (e.g. RF1, RF2).

**Atores:** Identifique os atores envolvidos no UC.

**Prioridade:** Identifique qual a prioridade de desenvolvimento deste UC (ver tabela 2).

**Pré-condições:** Condições necessárias para que UC seja executado.

**Pós-condições:** Condições do sistema após a execução do UC.

**Frequência de uso:** Identifique a frequência de uso deste UC (ver tabela 3).

**Campos:** Identifique os campos necessários no contexto sendo trabalho neste UC.

**Fluxo principal:** Defina um passo-a-passo das principais ações que o usuário deve tomar dentro deste UC considerando um caso de sucesso (sempre iniciar pelo ator).

a) Ator faz isso;

b) Sistema responde;

c) Ator (A1) (A2) faz isso;

d) Sistema (E1) responde;

e) O caso de uso é encerrado.

**Fluxo alternativo:** Defina caminhos alternativos que o usuário pode iniciar por vontade própria dentro do UC (sempre iniciar pelo ator).

A1 - muda de ideia

a) Ator muda de ideia;

b) Sistema responde;

c) Retorna ao passo c) do fluxo principal.

A2 - cancela operação

a) Ator cancela operação;

b) Sistema responde;

c) O caso de uso é encerrado.

**Fluxo de exceção:** Defina cenários em que o sistema pode disparar erros causados pelo usuário ou qualquer ação dentro da UC que tenha sido iniciada por decisão do sistema (sempre iniciar pelo sistema).

E1 - captura erro

a) Sistema captura erro e mostra mensagem;

b) Retorna ao passo c) do fluxo principal.

**Validações:** Defina como os campos serão validados e todos os casos que podem fazer disparar fluxos de exceção.

**Protótipos de telas:**

Insira imagens de protótipos de telas que possam realizar as ações descritas no UC contendo os requisitos de interface dos RF relacionados.

## 3.2 Pacotes

Diagramas de pacotes são opcionais neste modelo de documento. Devem demonstrar dependências e usos entre módulos do sistema permitindo uma visão geral dos fluxos de comunicação internos e externos. Portanto haverá nesta seção a descrição da arquitetura do sistema e seus principais meios de acesso e troca de mensagens.

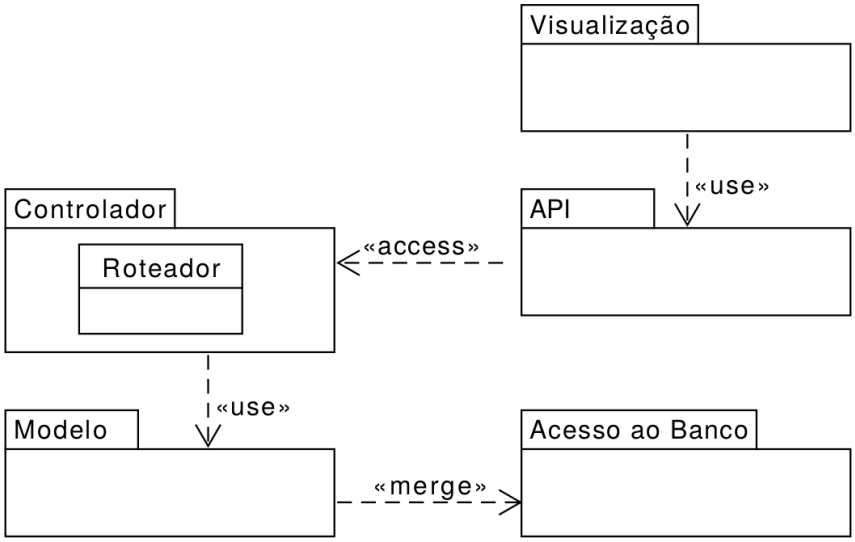


Figura 2 – Exemplo de diagrama de pacotes

Após a exposição do diagrama, um texto descritivo contendo informações sobre o mesmo deve ser redigido. Todo texto a seguir nesta seção é um **exemplo de redação para diagrama de pacotes**.

O diagrama da figura 2 demonstra a arquitetura desejada para o sistema dividido logicamente em back-end, front-end e middleware.

*Back-end* representa o conjunto de camadas de software responsável pela aplicação das regras de negócio, segurança, autenticação e persistência, o qual será composto pelas camadas *Controlador*, *Modelo* e *Acesso ao Banco*. A classe de controle *Roteador* fará recepção de requisições provindas do middleware e então identificará e redirecionará para classes de controles internas responsáveis por gerenciar o contexto das solicitações, além de montar respostas contendo o resultado do processamento obtido no formato esperado pelo front-end. As classes de controle acessarão classes da camada de modelo que por sua vez possuem características persistíveis e comportamentos que apoiam na recuperação de objetos de outros modelos relacionados, por isso sua ligação forte com a camada de acesso ao banco de dados.

*Middleware* compreende a camada que faz intermediações entre a visualização do sistema e suas regras. Através da *API* ficarão disponíveis as operações do sistema permitindo que diferentes sistemas possam utiliza-las, inclusive a própria camada de visualização. Para a produção da API deve ser utilizado o padrão RESTful.

*Front-end* é a parte que fará interface com o usuário permitindo-lhe interagir com o sistema e vice-versa. Este sistema contará com uma camada de *Visualização* implementando as operações disponíveis na API e cumprindo os requisitos de software especificados neste projeto. Apesar de a implementação desta camada estar aqui planejada, por haver API, qualquer aplicação terceira será compreendida como uma camada de visualização.

## 3.3 Classes

Classes, por definição, são abstrações de características e comportamentos em comum entre objetos do mundo real. Os diagramas devem demonstrar tantas classes quanto necessárias para o entendimento completo do sistema. Caso não seja possível a exposição de classes implementáveis, as camadas (pacotes) não desenvolvidas no projeto, porém planejadas, podem aparecer como metaclasses.

A figura 3 demonstra um diagrama de classes. Depois de exibido o diagrama, se seguirão tópicos para descrever formalmente o objetivo de cada classe com seus atributos e métodos (e os parâmetros de cada método).

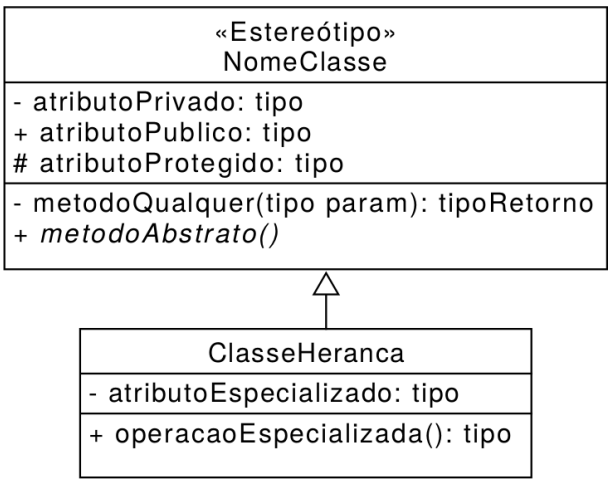


Figura 3 – Exemplo de diagrama de classes

### 3.3.1 NomeClasse

**Objetivo:** Descreva o objetivo da classe.

**Generalização de:** ClasseHeranca.

**Atributos**

| Nome | Tipo | Padrão | Tam. | Vis. | Req. | Descrição |
|---|---|---|---|---|---|---|
| atributoPrivado | tipo | - | 5 | pri | não | Descreva o objetivo do atributo |
| atributoPublico | tipo | “ola” | 1 | pub | sim | Descreva o objetivo do atributo |
| atributoProtegido | tipo | 999 | 10 | pro | sim | Descreva o objetivo do atributo |

**Métodos**

| | | | | | |
|---|---|---|---|---|---|
| metodoQualquer | Descreva o objetivo do método | | | | |
| | **Parâmetros** | | | | |
| | param | tipo | padrão | pub | Objetivo do parâmetro |
| | **Retorno** | | | | |
| | tipo | Objetivo do retorno | | | |
| metodoAbstrato | Descreva o objetivo do método | | | | |

### 3.3.2 ClasseHeranca

**Objetivo:** Descreva o objetivo da classe.

**Especialização de:** NomeClasse.

| **Atributos** | | | | | | |
|---|---|---|---|---|---|---|
| **Nome** | **Tipo** | **Tam.** | **Vis.** | **Padrão** | **Req.** | **Descrição** |
| atributoEspecializado | tipo | 5 | pri | - | não | Descreva o objetivo do atributo |

| **Métodos** | | |
|---|---|---|
| operacaoEspecializada | Descreva o objetivo do método | |
| | **Retorno** | |
| | tipo | Objetivo do retorno |

## 3.4 Atividades

## 3.5 Sequências

## 3.6 Instalação

# 4 Resultados esperados

# 5 Cronograma

# 6 Orçamento

# 7 Cronograma

# 8 Orçamento

Discriminar os recursos necessários para o desenvolvimento do projeto.

# Considerações finais

# Referências