\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Documento de Especificação de Requisitos

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Gerenciador de Lanchonete para SmartCanteen

**versão 2.1**2025

## Histórico de Versões

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autores** |
| 08/12/2025 | 1.0 | Elaboração do Documento | Vinicius Gabriel Monteiro e Ian Alvares Freitas |
| 12/12/2025 | 1.1 | Atualização dos tópicos Definição de problema e requisitos não funcionais | Vinicius Gabriel Monteiro |
| 12/12/2025 | 1.2 | Atualização dos Stakeholders | José Miguel Oliveira Pinheiro Dias |
| 17/12/2025 | 1.3 | Inserção da modelagem do banco de dados e módulos do código | Vinicius Gabriel Monteiro |
| 20/12/2025 | 1.4 | Continuação da modulagem dos códigos | José Miguel Oliveira Pinheiro Dias |
| 22/12/2025 | 2.0 | Atualização dos Requisitos funcionais e não funcionais seguindo a nova Prática Orientada | José Miguel Oliveira Pinheiro Dias |
| 23/12/2025 | 2.1 | Inserção das Bibliotecas, atualização da modelagem e documentação geral | Vinicius Gabriel Monteiro |

**Conteúdo**

**1. INTRODUÇÃO 4**

1.1 Finalidade 4

1.2 Contexto 4

**2. DESCRIÇÃO GERAL 4**

2.1 Definição do problema 4

2.2 Principais Stakeholders 5

2.3 Logo do sistema 5

**3. REQUISITOS FUNCIONAIS 6**

3.1 Requisitos gerais 6

**4. REQUISITOS NÃO FUNCIONAIS 7**

4.1 Requisitos Gerais

**5. Modelagem do banco de dados 9**

5.1 modelo de entidade relacionamento 9

5.2 modelo de entidade relacionamento melhorado 10

5.3 Resumo das Entidades 11

**6. tecnologia 11**

6.1 linguagens de programação 11

6.2 bibliotecas 12

**7. organização de pastas do projeto** 13

**8. modulo 14**

8.1 INTERFACE 14

8.2 Módulo DE relatorio 15

8.3 módulo DE sistemacontroller 16

8.4 módulo DE principalcontroller 17

**9. classes - poo 19**

9.1 cliente 19

9.2 funcionario 20

9.3 item\_venda 21

9.4 pessoa 21

9.5 produto 22

9.6 sessao 23

9.7 venda 24

**9. lINKS EXTERNOS** 25

**Documento de Especificação de Requisitos**

**1. Introdução**

**1.1Finalidade**

O documento tem como objetivo descrever os requisitos de um sistema voltado para a gestão de uma cantina de pequeno porte.

**1.2 Contexto**

Atualmente, o solicitante não realiza os processos administrativos da cantina, e deseja ter um controle maior e mais preciso sobre as vendas de lanches e demais processos. Sendo que os métodos manuais estão sujeitos a esquecimentos, erros e retrabalho, além de demandar maior tempo.

**2. Descrição geral**

O sistema proposto visa informatizar o controle de vendas de lanches do estabelecimento formalizado como lanchonete, permitindo o registro digital dos produtos, clientes e vendas realizadas. O objetivo é modernizar a rotina do profissional, tornando os processos mais rápidos, organizados e confiáveis.

**2.1Definição do Problema**

Não existe controle de nenhum processo administrativo.

| **O Problema de** | Falta de controle de vendas do estabelecimento |
| --- | --- |
| **Afeta** | Proprietário. |
| **Resultando em** | Escassez dos processos administrativos e controle da lanchonete. |
| **Benefícios** | Maior agilidade na administração das vendas do estabelecimento, tomada de decisão baseada em dados precisão nos registros da lanchonete, redução de falhas. |

**2.2 Principais Stakeholders**

|  |  |
| --- | --- |
| **Nome** | **Responsabilidade no Projeto** |
| Vinicius Gabriel da Silva Monteiro | Desenvolver o sistema e banco de dados |
| Ian Alvares Freitas | Desenvolver o sistema e banco de dados |
| José Miguel Oliveira Pinheiro Dias | Desenvolver o sistema e banco de dados |
| Tárcio Ribeiro Cavalcante | Representante do estabelecimento |

**2.3 Logo do Sistema**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**3. Requisitos Funcionais**

Neste tópico é abordado os requisitos funcionais que estão presentes no sistema de gestão de lanchonete.

**3.1 Requisitos Gerais**

|  |  |
| --- | --- |
| CÓDIGO | DESCRIÇÃO |
| RFG-01 | O sistema deve permitir o registro de clientes. |
| RFG-02 | O sistema deve permitir o registro de produtos. |
| RFG-03 | O sistema deve permitir o registro de vendas. |
| RFG-04 | O sistema deve exibir os clientes e produtos registrados durante o cadastro das vendas. |
| RFG-05 | O sistema deve fornecer um histórico de vendas. |
| RFG-06 | O sistema deve fornecer um relatório de desempenho de vendas dos produtos |
| RFG-07 | O sistema deve fornecer um relatório da quantidade de pedidos dos clientes |
| RFG-08 | O sistema deve permitir a ordenação do relatório de vendas dos produtos por quantidade vendida (ascendente e descendente) para identificar os itens mais vendidos e menos vendidos |
| RFG-09 | O sistema deve permitir a ordenação do relatório de pedidos dos clientes por quantidade de venda (ascendente e descendente). |
| RFG-10 | O sistema deve ser reestruturado para utilizar o paradigma de Orientação a Objetos, organizando suas funcionalidades em classes e métodos com responsabilidades bem definidas. |
| RFG-11 | O sistema deve permitir o cadastro de funcionários responsáveis pelo atendimento das vendas, armazenando informações básicas para identificação do atendente. |
| RFG-12 | O sistema deve permitir a listagem dos funcionários cadastrados no sistema. |
| RFG-13 | O sistema deve permitir associar cada venda realizada a um funcionário responsável pelo atendimento. |
| RFG-14 | O sistema deve registrar os atendimentos realizados pelos funcionários, vinculando cada atendimento à respectiva venda e data. |
| RFG-15 | O sistema deve permitir a geração de um relatório contendo a quantidade de atendimentos realizados por cada funcionário em um dia específico. |
| RFG-16 | O sistema deve permitir a geração de um relatório contendo o valor total de vendas realizadas por cada funcionário em um dia específico. |

**4. Requisitos Não Funcionais**

Neste tópico é abordado os requisitos funcionais que estão presentes no sistema de gestão de lanchonetes.

**4.1 Requisitos Gerais**

|  |  |
| --- | --- |
| CÓDIGO | DESCRIÇÃO |
| RNF-01 | Responsividade do terminal do sistema inferior a 15 segundos para demonstração dos resultados. |
| RNF-02 | Performance: Tempo de resposta inferior a 10 segundos para abertura de telas e cálculos, Banco de Dados leve e rápido (SQLite, por exemplo). |
| RNF-03 | Segurança: Armazenamento seguro dos dados do cliente e dos registros. |
| RNF-04 | Usabilidade: Interface amigável e intuitiva, para usuários com baixo nível de instrução tecnológica e uma navegação fluída com poucos cliques. |
| RNF-05 | Padronizar ícones, palavras, aparências e ações que possuam o mesmo significado, de acordo com as convenções da plataforma em uso. |
| RNF-06 | Manutenibilidade: Código modular para facilitar futuras atualizações ou correções. |
| RNF-07 | Adotar uma identidade visual padronizada para o sistema de gerenciamento |
| RNF-08 | Adotar linguagem em concordância com o mundo real do usuário, empregando palavras, frases, conceitos e convenções familiares ao mesmo. |
| RNF-9 | Prevenir a ocorrência de erros e, caso aconteçam, emitir mensagens que possibilitem ao usuário reconhecer, diagnosticar e se recuperar dos mesmos. As mensagens de erro devem ser expressas em linguagem direta (sem códigos), indicando precisamente o problema e construtivamente sugerir uma solução. |
| RNF-10 | Fornecer explicitamente ao usuário a opção de confirmação e cancelamento no final de todas as tarefas operacionais, evitando que surjam dúvidas, para o usuário, sobre a efetiva gravação ou aborto de cada transção. |
| RNF-11 | Não permitir a introdução, por parte do usuário, de dados inconsistentes ou que violem a integridade daqueles armazenados de maneira a requer intervenções através de ferramentas não próprias do ambiente do usuário final ou do administrador do sistema. |
| RNF-12 | Privacidade e LGPD: Termo de consentimento para coleta de dados pessoais, se armazenar dados sensíveis de clientes e opção de exclusão dos dados do cliente, caso solicitado. |
| RNF-13 | O sistema deve ser desenvolvido seguindo os princípios da Orientação a Objetos, utilizando classes, encapsulamento e modularização do código. |

**Legenda**

**Codificação dos Requisitos:**

Os requisitos estão codificados de acordo o módulo e a temática aos quais pertencem, conforme formato a seguir:

**MOD-NNN**

Onde:

**MOD** = código do módulo referente ao requisito, podendo ser: *RNF* - Requisitos não Funcionais; RFN - Requisitos Funcionais

**NNN** = sequência numérica de três algarismos.

**5. Modelagem do Banco de Dados**

O processo de modelagem de dados seguiu uma evolução interativa para garantir as regras de negócio de vendas e clientes fossem atendidas com precisão e normalização técnica.

**5.1 Modelo de Entidade Relacionamento (versão inicial)**

Diagrama Entidade-Relacionamento (ERD)

Diagrama

O conteúdo gerado por IA pode estar incorreto.  
A primeira versão do modelo (segunda imagem) estabeleceu a estrutura básica do banco de dados, focando na identificação das tabelas principais e seus atributos técnicos primários.

* **Tabelas definidas:** clientes, vendas, itens\_da\_venda, e produtos.
* **Características**: Esta versão já apresenta a estrutura de relacionamentos muitos-para-muitos (N:N) entre produtos e vendas, resolvida através da tabela associativa itens\_da\_venda.
* **Observações de Melhoria:** Nesta etapa, o foco foi a definição física (tipagem). Notou-se a necessidade de refinar a semântica das relações e a cardinalidade para melhor leitura do fluxo de negócio.

**5.2 Modelo de Entidade Relacionamento Melhorado (Versão atual)**

Diagrama de Entidade-Relação Melhorado (EER)

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

O modelo melhorado (primeira imagem) refina a abstração do sistema, focando na lógica de negócio e na clareza das associações (utilizando a notação de "Pé de Galinha" e verbos de ligação).

**Principais Melhorias e Regras de Negócio:**

1. **Relação Cliente-Venda:** Um **Cliente** "Faz" (ou realiza) uma ou várias **Vendas** (1:N). A entidade Venda agora carrega a semântica de "id\_cliente" ou registro histórico do vínculo com o cliente.
2. **Relação Venda-Itens:** Uma **Venda** "Possui" N itens. Isso garante que uma única transação possa conter múltiplos produtos distintos com suas respectivas quantidades.
3. **Relação Produto-Itens:** Um **Produto** "Pertence" a N itens de venda. Isso permite que o mesmo produto seja vendido em diferentes transações ao longo do tempo.
4. **Consistência de Atributos:** Foram validados os campos essenciais como Preço e Tipo em **Produtos**, e a Data da compra em **Vendas**, fundamentais para relatórios gerenciais

**5.3 Resumo das Entidades**

|  |  |
| --- | --- |
| Cliente | Armazena dados de identificação do comprador. |
| Vendas | Registro cabeçalho da transação (data e cliente). |
| Produtos | Catálogo de itens disponíveis com preço e categoria. |
| Itens\_da\_venda | Detalhamento da venda, especificando quais produtos e em que quantidades foram adquiridos. |

**6. Tecnologias**

**6.1 Linguagem de programação**

**Ruby** é uma linguagem de programação dinâmica, de código aberto e focada na simplicidade e na produtividade. Criada em meados da década de 1990 no Japão por **Yukihiro "Matz" Matsumoto**, ela foi desenhada com uma filosofia única: a linguagem deve ser natural para os humanos lerem e escreverem, e não apenas eficiente para as máquinas executarem.

**Principais Vantagens**

Aqui está uma mini lista com os pontos fortes que fazem desenvolvedores amarem Ruby:

* **Sintaxe Elegante e Legível:** O código Ruby lê-se quase como frases em inglês. Isso reduz a curva de aprendizado e facilita a manutenção do código por outras pessoas.
* **Ecossistema Rico (Gems):** Ruby possui um sistema de bibliotecas incrível chamado *RubyGems*. Se você precisa de uma funcionalidade (como login, upload de arquivos ou pagamentos), provavelmente já existe uma "Gem" pronta para isso.
* **Flexibilidade e Metaprogramação:** Ruby permite que o programador altere partes da própria linguagem durante a execução. Isso dá poderes incríveis para criar códigos enxutos e "mágicos".
* **Comunidade Ativa e Amigável:** A comunidade Ruby é famosa pelo lema "MINASWAN" (*Matz is nice so we are nice*), sendo muito acolhedora para iniciantes.
* **Alta Produtividade:** Graças à sua sintaxe limpa e ao ecossistema, é possível construir protótipos e produtos finais (MVPs) muito mais rápido do que em linguagens mais verbosas, como Java ou C++.

**6.2 Bibliotecas**

O **SQLite 3** é um sistema de gerenciamento de banco de dados relacional leve, amplamente utilizado em aplicações de pequeno e médio porte. Diferentemente de outros SGBDs, como MySQL ou PostgreSQL, o SQLite não necessita de um servidor para funcionar, pois todo o banco de dados é armazenado em um único arquivo no sistema. Essa característica torna sua configuração simples e prática, sendo ideal para projetos acadêmicos, aplicações desktop, sistemas embarcados e protótipos. O SQLite utiliza a linguagem SQL padrão para criação e manipulação dos dados, oferecendo suporte a tabelas, consultas, índices e relacionamentos. Apesar de suas vantagens, ele possui limitações quanto ao acesso simultâneo de múltiplos usuários, já que o controle de concorrência é feito por bloqueio do arquivo. Ainda assim, o SQLite 3 é uma excelente escolha quando se busca simplicidade, desempenho e facilidade de manutenção.

A **biblioteca tty-prompt** é uma "gem" (biblioteca) do Ruby utilizada para criar interfaces de linha de comando (CLI) interativas, bonitas e fáceis de usar. Ela fornece componentes prontos para coletar entradas do utilizador, como menus de seleção, perguntas do tipo sim/não, máscaras para senhas (onde o texto fica oculto), entre outros recursos que melhoram a experiência do utilizador no terminal.

No projeto SmartCanteen V-2.0, a tty-prompt foi introduzida para substituir a interação básica de texto (feita com puts e gets) por uma navegação mais robusta e visual. Abaixo está o detalhe de como ela está a ser aplicada:

1. Criação de Menus Interativos

Em vez de o utilizador ter de digitar um número manualmente para escolher uma opção, o sistema apresenta um menu navegável (onde se pode usar as setas do teclado).

Menu Principal: No ficheiro interface.rb, o menu principal utiliza o método .select para exibir as opções como "Cadastrar novo Produto", "Registrar Venda", etc.

Exemplo: prompt.select("Escolha uma opção:", menu\_opcoes).

Menu de Relatórios: Também no interface.rb, a biblioteca é usada para listar as opções de relatórios, incluindo a paginação (per\_page: 10) para facilitar a visualização se houver muitos itens.

2. Máscara de Senha (Segurança Visual)

A biblioteca é utilizada para ocultar a senha enquanto o utilizador a digita, exibindo bolinhas ou asteriscos em vez dos caracteres reais.

Login: Ao iniciar o sistema, a senha solicitada usa o método .mask.

Exemplo: senha = prompt.mask("Senha:").

Cadastro de Funcionários: No controlador menuPrincipalController.rb, ao registar um novo funcionário, a senha também é solicitada com máscara para garantir privacidade.

Exemplo: senha = prompt.mask("Digite a senha do funcionário:").

**7. Organização de pastas do projeto**

📁 SmartCanteen\_V-2.0/  
├── 📁 bd/  
 ├── dbConn.rb  
 ├── script.teste.rb  
 ├── sistema\_cantina.db  
├── 📁 classes/  
 ├── cliente.rb  
 ├── funcionario.rb  
 ├── item\_venda.rb  
 ├── pessoa.rb  
 ├── produto.rb  
 ├── sessão.rb  
 ├── venda.rb  
├── 📁config/  
 ├── path.rb  
├── 📁controllers/  
 ├── menuPrincipalController.rb  
 ├── menuRelatorioController.rb  
 ├── cliente.rb  
├── 📁src/  
 ├── utlis.rb  
 ├── validação.rb  
├── interface.rb  
├── .gitignore  
├── gemfile  
├── gemfile.lock

**8. Módulos**

**8.1 Interface**

Menu de opções do sistema:

* Cadastrar novo produto
* Cadastrar novo cliente
* Registrar venda
* Cadastrar novo funcionário (se Gerente)
* Relatórios
  + Listar todos os clientes
  + Listar todos os produtos
  + Listar todas as vendas
  + Apresentar o total do valor vendido em um dia específico
  + Produto mais vendido
  + Total de atendimentos por funcionário (Dia)
  + Total vendido por funcionário (Dia)
  + Voltar ao menu principal
* Sair do sistema

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.**8.2 Módulo Relatórios**

Função responsável por listar todas as vendas realizadas desde o primeiro registro de vendas.

Texto

O conteúdo gerado por IA pode estar incorreto.

Função responsável por validar a data informada pelo usuário, garantindo o formato correto (DD-MM-AAAA) e retornando uma data válida para uso no sistema.

**8.3 Módulo SistemaController**

Texto

O conteúdo gerado por IA pode estar incorreto.

Função responsável por realizar o controle de login e logout no sistema

**8.4 Módulo PrincipalController**

Texto

O conteúdo gerado por IA pode estar incorreto.

Função responsável por listar as vendas realizadas em um dia específico, exibindo a quantidade de vendas, os detalhes de cada venda e o valor total faturado no dia.

Texto

O conteúdo gerado por IA pode estar incorreto.

Função responsável por listar o produto mais vendido dentre toda o registro de vendas.

**9. Classes - POO**

A Programação Orientada a Objetos (POO) é um **paradigma de programação** (ou seja, um estilo ou forma de pensar e organizar o código) que tenta aproximar a programação do mundo real.

Se na programação antiga (procedural) o foco estava em uma lista de instruções a serem seguidas (receita de bolo), na POO o foco está em **criar objetos** que interagem entre si.

**9.1 cliente**

Texto

O conteúdo gerado por IA pode estar incorreto.

**9.2 funcionario**

Texto

O conteúdo gerado por IA pode estar incorreto.

**9.3 item\_venda**

Texto

O conteúdo gerado por IA pode estar incorreto.

**9.5 pessoa**

Tela de computador com letras e números em fundo preto

O conteúdo gerado por IA pode estar incorreto.

**9.4 produto**

Texto

O conteúdo gerado por IA pode estar incorreto.

**9.6 sessao**

Texto

O conteúdo gerado por IA pode estar incorreto.

**9.7 venda**

Texto

O conteúdo gerado por IA pode estar incorreto.

**9. LINKS EXTERNOS**

* GitHub
  + <https://github.com/V1niGabriel/SmartCanteen>