Sistema de Gestão de Loja

versao 1.0.0

Gerado por Doxygen 1.9.6

1 README

1.1 Sistema de Loja

Este é um sistema de gerenciamento para uma loja de produtos. Ele permite gerenciar produtos, realizar vendas, e manter registros de transações.

1.1.1 Proposito do projeto

O propósito desse projeto é criar um sistema de gerenciamento abrangente para uma loja de produtos. O sistema visa facilitar e otimizar diversas operações relacionadas à gestão da loja, incluindo o gerenciamento de produtos, o acompanhamento de vendas e a manutenção de registros de transações.

1.1.2 Diagrama de Classes

1.1.3 Tecnologias utilizadas

- Java 17;
- SQLite
- Gradle

1.1.4 Estrutura do Projeto

O projeto está organizado em várias camadas, cada uma responsável por diferentes aspectos do sistema:

- `model`: Contém as classes que representam as entidades do negócio (Produto, Venda, etc.).
- `dao`: Camada de acesso a dados, com classes responsáveis pela interação com o banco de dados.
- `services`: Contém a lógica de negócios e interage com a camada de acesso a dados.
- `view`: Responsável pela interface de usuário, onde as interações com o usuário acontecem.
- `exception`: Contém as exceções personalizadas usadas no projeto.
- `utils`: Fornece utilitários diversos, como manipulação de cores e arquivos.

1.1.5 Recursos Disponíveis

- Gestão de produtos: adicionar, atualizar, remover e buscar produtos.
- Realização de vendas: registrar vendas de produtos.
- Consulta de vendas: visualizar o histórico de vendas.
- Cancelar uma venda.
- Ver faturamento.

1.1.6 Como realizar o build

1. **Navegue até o Diretório do Projeto:** Abra um terminal e navegue até o diretório do seu projeto.
2. **Execute o Comando de Build:** Execute o seguinte comando para construir o projeto:

```
$ ./gradlew build
```

1.1.7 Autores

|
| --- | --- | ---
| @Raymendesc | @Leandro208 | @DougFelipe


```
|     <img alt="Raymendesc" src="https://github.com/Raymendesc.png?size=115"
width="115">
@Raymendesc| <img alt="Leandro208" src="https://github.com/leandro208.png?size=115"
width="115">
@Leandro208| <img alt="DougFelipe" src="https://github.com/DougFelipe.png?size=115"
width="115">
@DougFelipe| | :—: |:—: |:—:
```


2 Namespaces

2.1 Lista de pacotes

Esta é a lista com os pacotes e suas respectivas descrições (quando disponíveis):

3 Índice Hierárquico

3.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

4 Índice dos Componentes

4.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

5 Índice dos Arquivos

5.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

6 Namespace

6.1 Pacote br.ufrn.loja

Pacotes

- package **dao**
- package **exception**
- package **infra**
- package **model**
- package **services**
- package **utils**
- package **view**

Componentes

- class **LojaApplication**

  Classe com o metodo Main.

6.2 Pacote br.ufrn.loja.dao

Componentes

- interface **GenericDao**
- class **ItemVendaDao**

  Classe de acesso a dados (DAO) para a entidade ItemVenda.
- class **ProdutoDao**

  Implementação do DAO para a entidade Produto.
- class **VendaDao**

  Classe de acesso a dados (DAO) para a entidade Venda.

6.3 Pacote br.ufrn.loja.exception

Componentes

- class **CriacaoConexaoException**
- class **OpcaoInvalidaException**

6.4 Pacote br.ufrn.loja.infra

Componentes

- class **ConnectionFactory**

  Classe responsável por fornecer instâncias de conexão com o banco de dados SQLite.

6.5 Pacote br.ufrn.loja.model

Componentes

- class **ItemVenda**

  Representa um item de venda associado a um produto e uma quantidade.
- class **Produto**
- class **ProdutoBuilder**
- enum **StatusVenda**
- class **Venda**

  Representa uma transação de venda realizada na loja.

6.6 Pacote br.ufrn.loja.services

Componentes

- class **AbstractService**

  Classe abstrata que define operações básicas de movimentação (CRUD).
- class **ProdutoService**
- class **VendaService**

  Fornece serviços relacionados à entidade Venda.

6.7 Pacote br.ufrn.loja.utils

Componentes

- class **ArquivoUtils**

  Classe utilitária para operações relacionadas a arquivos.
- class **CorUtils**

6.8 Pacote br.ufrn.loja.view

Componentes

- class **Menu**
- class **MenuAbstract**
- class **TelaBusca**
- class **TelaCadastro**
- class **TelaFaturamento**
- class **TelaVendas**

  Classe que representa a tela de vendas e suas funcionalidades.

7 Classes

7.1 Referência do Template da Classe br.ufrn.loja.services.AbstractService< E >

Classe abstrata que define operações básicas de movimentação (CRUD).

Métodos Públicos

- void **processar** (int opcao)

  Processa a operação solicitada.
- abstract boolean **buscar** ()
- E **getObjeto** ()
- void **setObjeto** (E **objeto**)

Membros Protegidos

- void **cadastrar** ()

  Método que prepara para o cadastro de um objeto.
- abstract boolean **validar** ()
- void **alterar** ()

  Método abstrato para alterar o objeto.
- abstract void **remover** ()

  Método abstrato para remover o objeto.
- abstract void **imprimir** ()

Atributos Protegidos

- E **objeto**
- **GenericDao**< E > **dao**

7.1.1 Descrição detalhada

Classe abstrata que define operações básicas de movimentação (CRUD).

Parâmetros do template

E	Tipo de objeto manipulado pela classe.

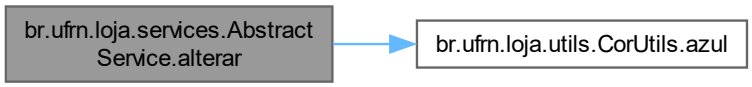
7.1.2 Documentação das funções

7.1.2.1 alterar() void br.ufrn.loja.services.AbstractService< E >.alterar ( ) [protected]

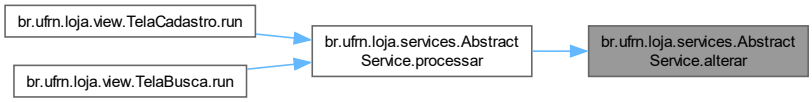
Método abstrato para alterar o objeto.

```
00065                               {
00066         dao.alterar(objeto);
00067         System.out.println(CorUtils.azul("Alteração realizada com sucesso!"));
00068     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



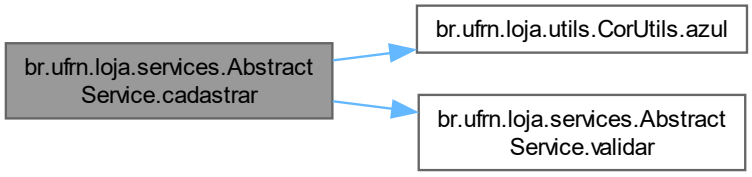
7.1.2.2 buscar() abstract boolean br.ufrn.loja.services.AbstractService< E >.buscar ( ) [abstract]

7.1.2.3 cadastrar() void br.ufrn.loja.services.AbstractService< E >.cadastrar ( ) [protected]

Método que prepara para o cadastro de um objeto.

```
00049                                {
00050         if(validar()) {
00051             dao.salvar(objeto);
00052             System.out.println(CorUtils.azul("Cadastro realizado com sucesso!"));
00053         }
00054     }
```

Este é o diagrama das funções utilizadas por essa função:



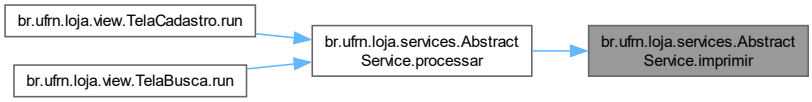
Esse é o diagrama das funções que utilizam essa função:

7.1.2.4 getObjeto() E br.ufrn.loja.services.AbstractService< E >.getObjeto ( )

```
00080                           {
00081         return objeto;
00082     }
```

7.1.2.5 imprimir() abstract void br.ufrn.loja.services.AbstractService< E >.imprimir ( ) [abstract], [protected]

Imprime a lista de produtos de forma estilizada. Esse é o diagrama das funções que utilizam essa função:



7.1.2.6 processar() void br.ufrn.loja.services.AbstractService< E >.processar (int *opcao*)

Processa a operação solicitada.

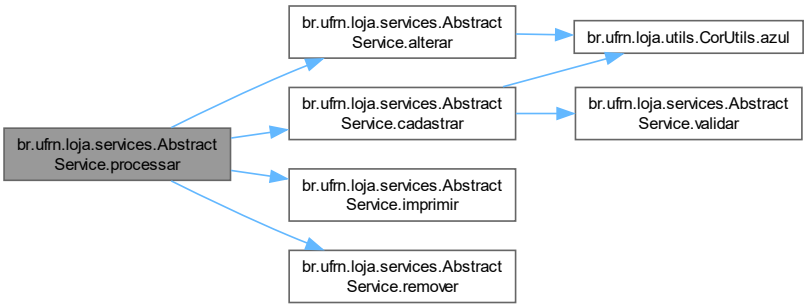
Parâmetros

opcao	A operação a ser realizada.

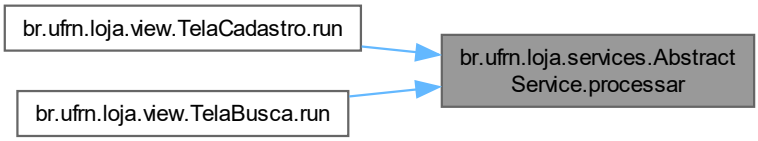
```
00023                                   {
00024         switch (opcao) {
00025         case MenuAbstract.CADASTRAR:
00026             cadastrar();
00027             break;
00028         case MenuAbstract.VER_TODOS:
00029             imprimir();
00030             break;
00031         case MenuAbstract.ALTERAR:
00032             alterar();
00033             break;
00034         case MenuAbstract.REMOVER:
```

```
00035             remover();
00036             break;
00037         default:
00038             System.out.println("Nenhuma opcao");
00039         }
00040
00041     }
```

Este é o diagrama das funções utilizadas por essa função:



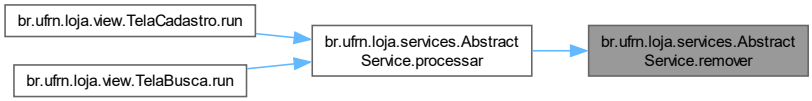
Esse é o diagrama das funções que utilizam essa função:



7.1.2.7 remover() `abstract void` **br.ufrn.loja.services.AbstractService**< E >.remover ( ) `[abstract], [protected]`

Método abstrato para remover o objeto.

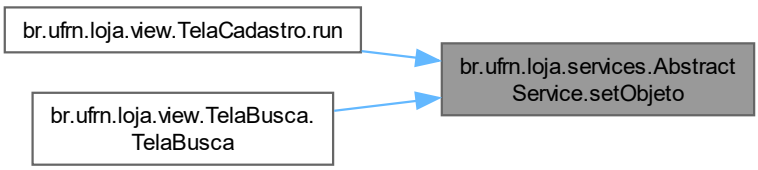
Esse é o diagrama das funções que utilizam essa função:



7.1.2.8 setObjeto() void br.ufrn.loja.services.AbstractService< E >.setObjeto (E *objeto*)

```
00084                                   {
00085         this.objeto = objeto;
00086     }
```

Esse é o diagrama das funções que utilizam essa função:



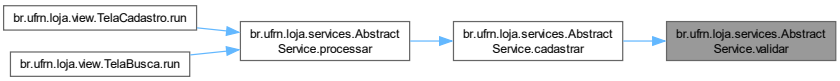
7.1.2.9 validar() abstract boolean br.ufrn.loja.services.AbstractService< E >.validar ( ) [abstract], [protected]

Valida o objeto

Retorna

true ou false

Esse é o diagrama das funções que utilizam essa função:



7.1.3 Atributos

7.1.3.1 dao GenericDao<E> br.ufrn.loja.services.AbstractService< E >.dao [protected]

7.1.3.2 objeto E br.ufrn.loja.services.AbstractService< E >.objeto [protected]

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/services/ **AbstractService.java**

7.2 Referência da Classe br.ufrn.loja.utils.ArquivoUtils

Classe utilitária para operações relacionadas a arquivos.

Membros Públicos Estáticos

- static String **lerTExtoArquivo** (final String nomeArquivo)
  Método para ler o conteúdo de um arquivo de texto.

7.2.1 Descrição detalhada

Classe utilitária para operações relacionadas a arquivos.

7.2.2 Documentação das funções

7.2.2.1 lerTExtoArquivo() `static String br.ufrn.loja.utils.ArquivoUtils.lerTExtoArquivo ( final String nomeArquivo ) [static]`

Método para ler o conteúdo de um arquivo de texto.

Parâmetros

nomeArquivo	Nome do arquivo a ser lido.

Retorna

String contendo o conteúdo do arquivo.

```
00018                                                                   {
00019         try {
00020             long time = System.currentTimeMillis();
00021             BufferedReader reader = new BufferedReader(new FileReader(nomeArquivo));
00022             StringBuilder sb = new StringBuilder();
00023             String linha;
00024 
00025             while ((linha = reader.readLine()) != null) {
00026                 sb.append(linha).append("\n");
00027             }
00028             reader.close();
00029             time = System.currentTimeMillis() - time;
00030 
00031             return sb.toString();
00032 
00033         } catch (IOException e) {
00034             e.printStackTrace();
00035         }
00036         return "";
00037     }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/utils/ **ArquivoUtils.java**

7.3 Referência da Classe br.ufrn.loja.infra.ConnectionFactory

Classe responsável por fornecer instâncias de conexão com o banco de dados SQLite.

Métodos Públicos

- Connection **getConnection** ()

Membros Públicos Estáticos

- static **ConnectionFactory** **getInstance** ()

7.3.1 Descrição detalhada

Classe responsável por fornecer instâncias de conexão com o banco de dados SQLite.

7.3.2 Documentação das funções

7.3.2.1 getConnection() `Connection br.ufrn.loja.infra.ConnectionFactory.getConnection ( )`

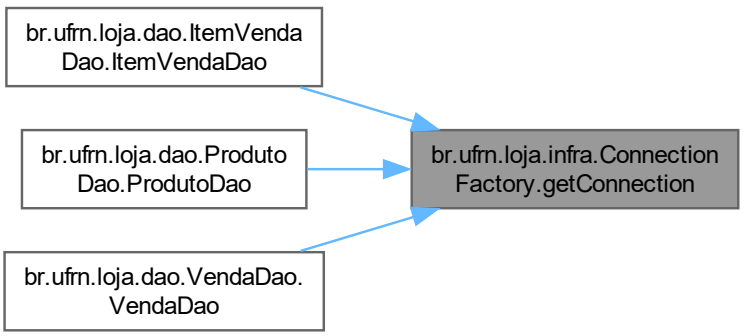
Obtém a conexão com o banco de dados.

Retorna

Conexão com o banco de dados SQLite

```
00053                                   {
00054         return this.connection;
00055     }
```

Esse é o diagrama das funções que utilizam essa função:



7.3.2.2 getInstance() static ConnectionFactory br.ufrn.loja.infra.ConnectionFactory.get↩ Instance ( ) [static]

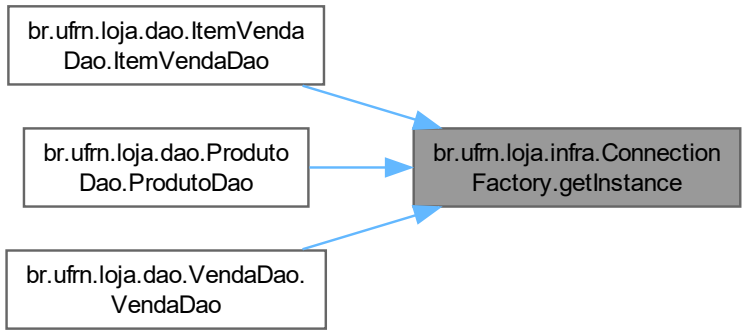
Obtém a única instância da classe, para assim evitar que se crie diversas conexões.

Retorna

Instância da ConnectionFactory

```
00041                                                {
00042         if (INSTANCE == null) {
00043             INSTANCE = new ConnectionFactory();
00044         }
00045         return INSTANCE;
00046     }
```

Esse é o diagrama das funções que utilizam essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/infra/ **ConnectionFactory.java**

7.4 Referência da Classe br.ufrn.loja.utils.CorUtils

Membros Públicos Estáticos

- static String **verde** (String texto)
- static String **azul** (String texto)
- static String **laranja** (String texto)
- static String **vermelho** (String texto)
- static String **bold** (String texto)

7.4.1 Descrição detalhada

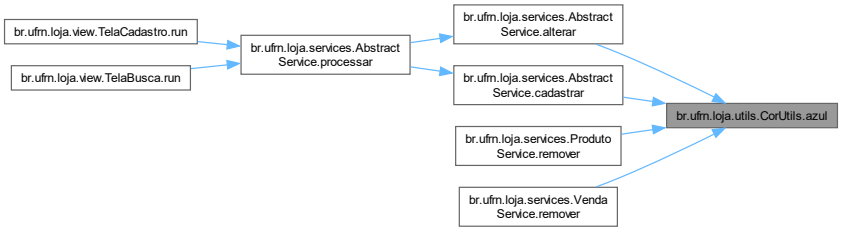
Esta classe fornece métodos utilitários para formatação de cores no console utilizando códigos ANSI escape.

7.4.2 Documentação das funções

7.4.2.1 azul() static String br.ufrn.loja.utils.CorUtils.azul (String *texto*) [static]

```
00012                                                {
00013          return "\u001B[94;1m" + texto + "\u001B[0m";
00014      }
```

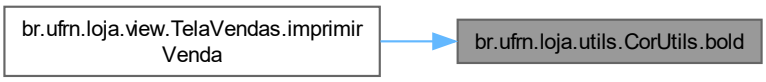
Esse é o diagrama das funções que utilizam essa função:



7.4.2.2 bold() static String br.ufrn.loja.utils.CorUtils.bold (String *texto*) [static]

```
00023                                                {
00024          return "\u001B[1m"+texto+"\u001B[0m";
00025      }
```

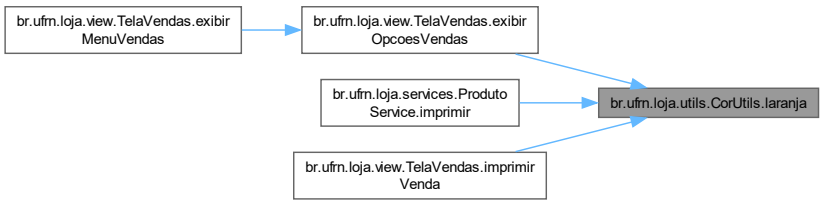
Esse é o diagrama das funções que utilizam essa função:



7.4.2.3 laranja() static String br.ufrn.loja.utils.CorUtils.laranja (String *texto*) [static]

```
00016                                                  {
00017          return "\u001B[38;5;208m" + texto + "\u001B[0m";
00018      }
```

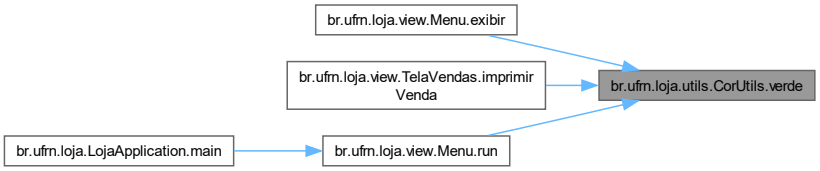
Esse é o diagrama das funções que utilizam essa função:



7.4.2.4 verde() `static String br.ufrn.loja.utils.CorUtils.verde ( String texto ) [static]`

```
00008                                          {
00009         return "\u001B[32m" + texto + "\u001B[0m";
00010     }
```

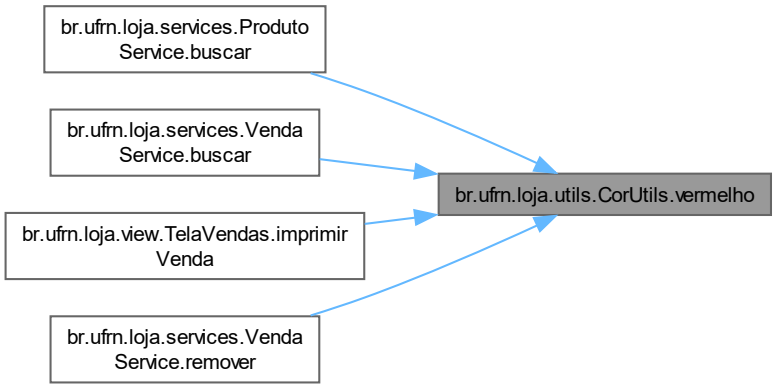
Esse é o diagrama das funções que utilizam essa função:



7.4.2.5 vermelho() `static String br.ufrn.loja.utils.CorUtils.vermelho ( String texto ) [static]`

```
00020                                          {
00021         return "\u001B[1;31m" + texto + "\u001B[0m";
00022     }
```

Esse é o diagrama das funções que utilizam essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/utils/ **CorUtils.java**

7.5 Referência da Classe br.ufrn.loja.exception.CriacaoConexaoException

Diagrama de hierarquia para br.ufrn.loja.exception.CriacaoConexaoException:

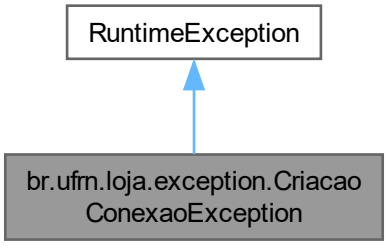
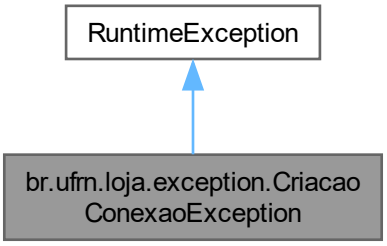


Diagrama de colaboração para br.ufrn.loja.exception.CriacaoConexaoException:



Métodos Públicos

- **CriacaoConexaoException** (String mensagem)
- **CriacaoConexaoException** (String mensagem, Throwable causa)

7.5.1 Construtores e Destrutores

7.5.1.1 CriacaoConexaoException() [1/2] br.ufrn.loja.exception.CriacaoConexaoException.Criacao↩ ConexaoException (String *mensagem*) { super(mensagem); }

```
00006                                                {
00007         super(mensagem);
00008     }
```

7.5.1.2 CriacaoConexaoException() [2/2] br.ufrn.loja.exception.CriacaoConexaoException.Criacao↩ ConexaoException (String *mensagem*, Throwable *causa*) { super(mensagem, causa); }

```
00010                                                           {
00011         super(mensagem, causa);
00012     }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/exception/ **CriacaoConexao↩ Exception.java**

7.6 Referência do Template da Interface br.ufrn.loja.dao.GenericDao< E >

Métodos Públicos

- void **salvar** (E obj)
- List< E > **buscarTodos** ()
- E **buscarPorId** (int id)
- void **delete** (int id)
- boolean **existePorId** (int id)
- void **alterar** (E obj)

7.6.1 Descrição detalhada

Interface genérica para operações básicas de persistência.

Parâmetros

<E>	Tipo da entidade manipulada pela interface.

7.6.2 Documentação das funções

7.6.2.1 alterar() `void br.ufrn.loja.dao.GenericDao< E >.alterar ( E obj )`

Altera os dados de uma entidade no banco de dados.

Parâmetros

obj	Entidade com os dados atualizados.

7.6.2.2 buscarPorId() `E br.ufrn.loja.dao.GenericDao< E >.buscarPorId ( int id )`

Busca uma entidade por seu ID.

Parâmetros

id	ID da entidade a ser buscada.

Retorna

Entidade encontrada ou null se não encontrada.

7.6.2.3 buscarTodos() `List< E > br.ufrn.loja.dao.GenericDao< E >.buscarTodos ( )`

Retorna uma lista contendo todas as entidades do tipo E no Banco de dados.

Retorna

Lista de entidades.

7.6.2.4 delete() `void br.ufrn.loja.dao.GenericDao< E >.delete ( int id )`

Exclui uma entidade do banco de dados com base no ID.

Parâmetros

id	ID da entidade a ser excluída.

7.6.2.5 existePorId() `boolean br.ufrn.loja.dao.GenericDao< E >.existePorId ( int id )`

Verifica se uma entidade com o ID especificado existe no banco de dados.

Parâmetros

id	ID da entidade a ser verificada.

Retorna

true se a entidade existir, false caso contrário.

7.6.2.6 salvar() `void br.ufrn.loja.dao.GenericDao< E >.salvar ( E obj )`

Salva a entidade no banco de dados.

Parâmetros

obj	Entidade a ser salva.

A documentação para essa interface foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/dao/ **GenericDao.java**

7.7 Referência da Classe br.ufrn.loja.model.ItemVenda

Representa um item de venda associado a um produto e uma quantidade.

Métodos Públicos

- int **getId** ()
  Obtém o identificador único do item de venda.
- void **setId** (int id)
  Define o identificador único do item de venda.
- **Produto** **getProduto** ()
  Obtém o produto associado ao item de venda.
- void **setProduto** (**Produto** produto)
  Define o produto associado ao item de venda.
- int **getQuantidade** ()
  Obtém a quantidade do produto no item de venda.
- void **setQuantidade** (int quantidade)
  Define a quantidade do produto no item de venda.
- **Venda** **getVendaId** ()
  Obtém a referência à venda à qual o item pertence.
- void **setVendaId** (**Venda** vendaId)
  Define a referência à venda à qual o item pertence.
- double **getSubtotal** ()
  Calcula e obtém o subtotal do item de venda.

7.7.1 Descrição detalhada

Representa um item de venda associado a um produto e uma quantidade.

7.7.2 Documentação das funções

7.7.2.1 getId() int br.ufrn.loja.model.ItemVenda.getId ( )

Obtém o identificador único do item de venda.

Retorna

Identificador único do item de venda.

```
00020                       {
00021         return id;
00022     }
```

7.7.2.2 getProduto() Produto br.ufrn.loja.model.ItemVenda.getProduto ( )

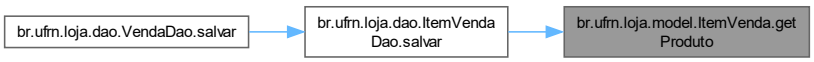
Obtém o produto associado ao item de venda.

Retorna

Produto associado ao item de venda.

```
00034                               {
00035         if (produto != null) {
00036             return produto;
00037         }
00038         return new Produto();
00039     }
```

Esse é o diagrama das funções que utilizam essa função:



7.7.2.3 getQuantidade() int br.ufrn.loja.model.ItemVenda.getQuantidade ( )

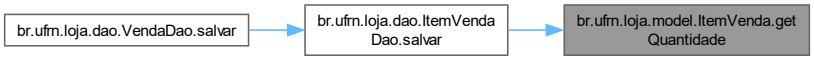
Obtém a quantidade do produto no item de venda.

Retorna

Quantidade do produto no item de venda.

```
00051                               {
00052         return quantidade;
00053     }
```

Esse é o diagrama das funções que utilizam essa função:



7.7.2.4 getSubtotal() `double br.ufrn.loja.model.ItemVenda.getSubtotal ( )`

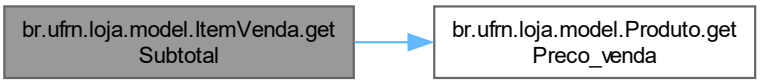
Calcula e obtém o subtotal do item de venda.

Retorna

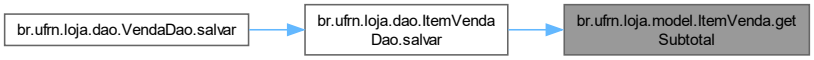
Subtotal do item de venda.

```
00079                                {
00080         if (produto != null) {
00081             return quantidade * produto.getPreco_venda();
00082         }
00083         return 0;
00084     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



7.7.2.5 getVendaId() `Venda br.ufrn.loja.model.ItemVenda.getVendaId ( )`

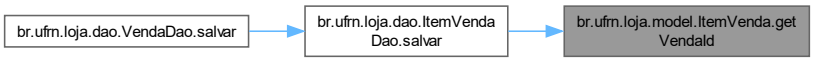
Obtém a referência à venda à qual o item pertence.

Retorna

Referência à venda à qual o item pertence.

```
00065                                {
00066         return vendaId;
00067     }
```

Esse é o diagrama das funções que utilizam essa função:



7.7.2.6 setId() `void br.ufrn.loja.model.ItemVenda.setId ( int id )`

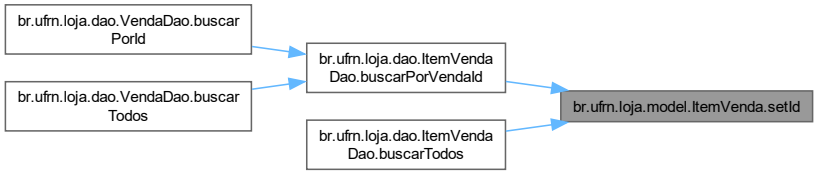
Define o identificador único do item de venda.

Parâmetros

id	Identificador único do item de venda.

```
00027                               {
00028         this.id = id;
00029     }
```

Esse é o diagrama das funções que utilizam essa função:



7.7.2.7 setProduto() `void br.ufrn.loja.model.ItemVenda.setProduto ( Produto produto )`

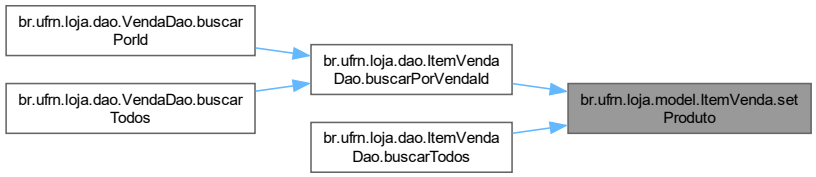
Define o produto associado ao item de venda.

Parâmetros

produto	Produto associado ao item de venda.

```
00044                                       {
00045         this.produto = produto;
00046     }
```

Esse é o diagrama das funções que utilizam essa função:



7.7.2.8 setQuantidade() `void br.ufrn.loja.model.ItemVenda.setQuantidade ( int quantidade )`

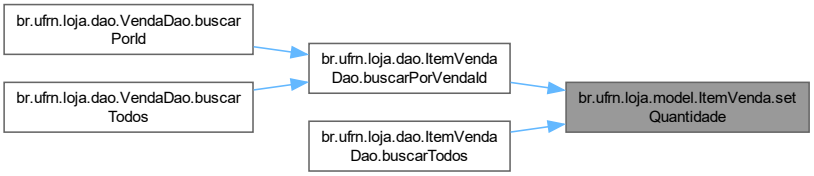
Define a quantidade do produto no item de venda.

Parâmetros

quantidade	Quantidade do produto no item de venda.

```
00058                                           {
00059         this.quantidade = quantidade;
00060     }
```

Esse é o diagrama das funções que utilizam essa função:



7.7.2.9 setVendaId() `void br.ufrn.loja.model.ItemVenda.setVendaId (`

Venda *vendaId*)

Define a referência à venda à qual o item pertence.

Parâmetros

venda↩ Id	Referência à venda à qual o item pertence.

```
00072                                       {
00073         this.vendaId = vendaId;
00074     }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/ **ItemVenda.java**

7.8 Referência da Classe br.ufrn.loja.dao.ItemVendaDao

Classe de acesso a dados (DAO) para a entidade ItemVenda.

Diagrama de hierarquia para br.ufrn.loja.dao.ItemVendaDao:

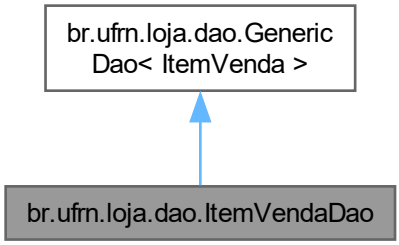
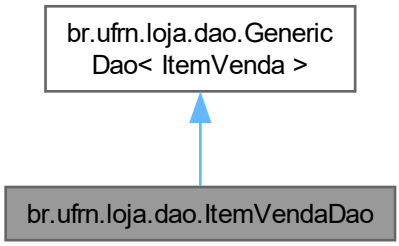


Diagrama de colaboração para br.ufrn.loja.dao.ItemVendaDao:



Métodos Públicos

- **ItemVendaDao** ()

  Construtor padrão que inicializa a conexão com o banco de dados.
- void **salvar** (**ItemVenda** obj)
- List< **ItemVenda** > **buscarTodos** ()
- **ItemVenda** **buscarPorId** (int id)
- void **delete** (int id)
- boolean **existePorId** (int id)
- void **alterar** (**ItemVenda** obj)
- List< **ItemVenda** > **buscarPorVendaId** (int vendaId)

  Busca todos os itens de uma venda especifica.

Métodos Públicos herdados de br.ufrn.loja.dao.GenericDao< ItemVenda >

- void **salvar** (E obj)
- List< E > **buscarTodos** ()
- E **buscarPorId** (int id)
- void **delete** (int id)
- boolean **existePorId** (int id)
- void **alterar** (E obj)

7.8.1 Descrição detalhada

Classe de acesso a dados (DAO) para a entidade ItemVenda.

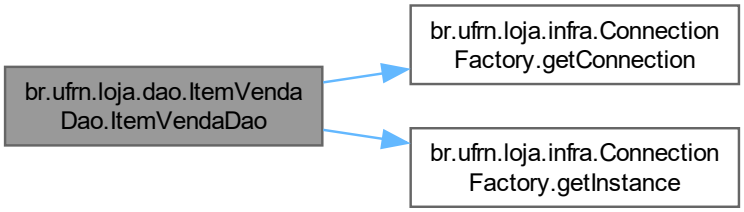
7.8.2 Construtores e Destrutores

7.8.2.1 ItemVendaDao() br.ufrn.loja.dao.ItemVendaDao.ItemVendaDao ( )

Construtor padrão que inicializa a conexão com o banco de dados.

```
00029                         {
00030         con = ConnectionFactory.getInstance().getConnection();
00031     }
```

Este é o diagrama das funções utilizadas por essa função:



7.8.3 Documentação das funções

7.8.3.1 alterar() void br.ufrn.loja.dao.ItemVendaDao.alterar (ItemVenda obj)

```
00095                                     {
00096
00097     }
```

7.8.3.2 buscarPorId() ItemVenda br.ufrn.loja.dao.ItemVendaDao.buscarPorId (int id)

```
00080                                      {
00081         return null;
00082     }
```

7.8.3.3 buscarPorVendaId() List< ItemVenda > br.ufrn.loja.dao.ItemVendaDao.buscarPorVendaId (int vendaId)

Busca todos os itens de uma venda especifica.

Parâmetros

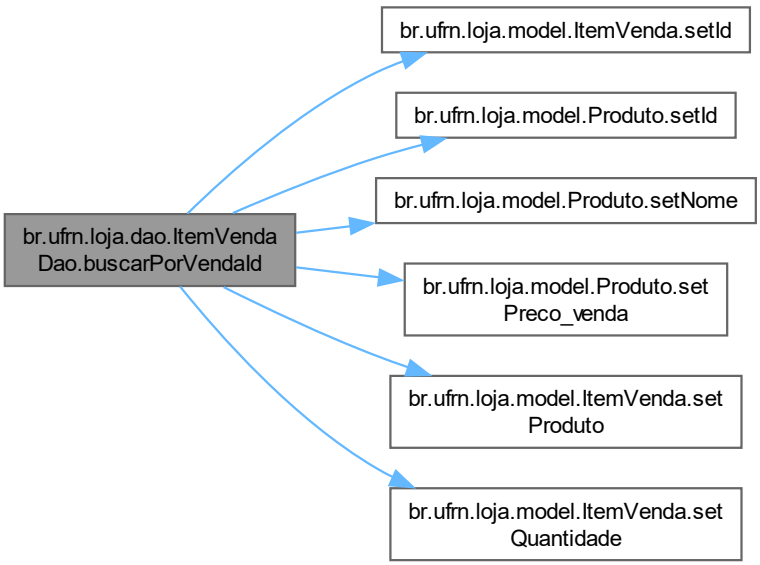
venda↩ Id	ID da venda associada aos itens.

Retorna

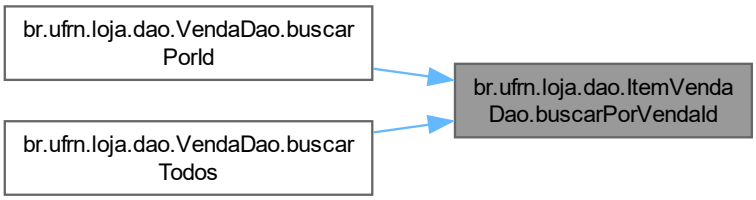
Lista de itens de venda associados à venda.

```
00118                                                   {
00119         List<ItemVenda> itens = new ArrayList<>();
00120         String sql = "SELECT v.id, v.quantidade, v.produto_id, p.nome, p.preco_venda " + "FROM
      item_venda v "
00121                 + "JOIN produto p ON p.id = v.produto_id " + "WHERE v.venda_id = ?";
00122
00123         try (PreparedStatement pstmt = con.prepareStatement(sql)) {
00124             pstmt.setInt(1, vendaId);
00125             ResultSet rs = pstmt.executeQuery();
00126
00127             while (rs.next()) {
00128                 ItemVenda item = new ItemVenda();
00129                 item.setId(rs.getInt("id"));
00130                 item.setQuantidade(rs.getInt("quantidade"));
00131
00132                 Produto produto = new Produto();
00133                 produto.setId(rs.getInt("produto_id"));
00134                 produto.setNome(rs.getString("nome"));
00135                 produto.setPreco_venda(rs.getDouble("preco_venda"));
00136
00137                 item.setProduto(produto);
00138                 itens.add(item);
00139             }
00140         } catch (SQLException e) {
00141             e.printStackTrace();
00142         } finally {
00143             fecharRecursos();
00144         }
00145
00146         return itens;
00147     }
```

Este é o diagrama das funções utilizadas por essa função:



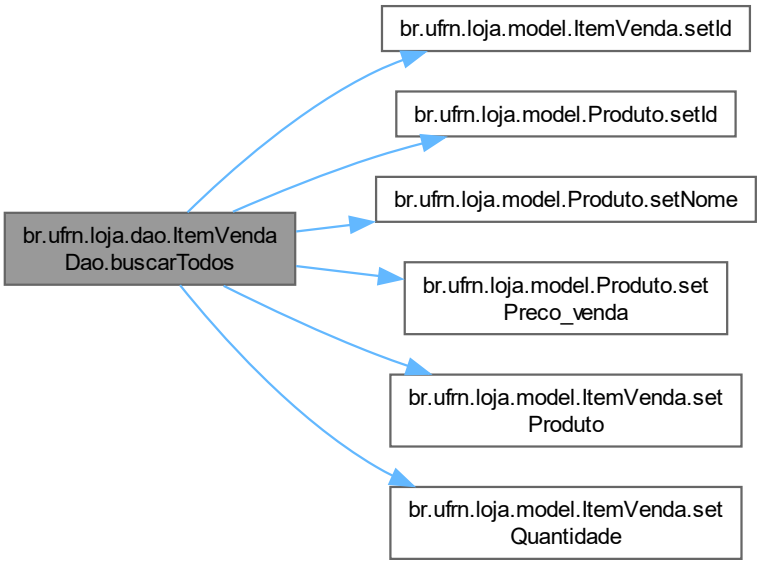
Esse é o diagrama das funções que utilizam essa função:



7.8.3.4 buscarTodos() List< ItemVenda > br.ufrn.loja.dao.ItemVendaDao.buscarTodos ( )

```
00053                                       {
00054         List<ItemVenda> resultado = new ArrayList<>();
00055         try (PreparedStatement pstmt = con.prepareStatement(SELECT_ALL)) {
00056             ResultSet rs = pstmt.executeQuery();
00057 
00058             while (rs.next()) {
00059                 ItemVenda item = new ItemVenda();
00060                 item.setId(rs.getInt("id"));
00061                 item.setQuantidade(rs.getInt("quantidade"));
00062 
00063                 Produto produto = new Produto();
00064                 produto.setId(rs.getInt("produto_id"));
00065                 produto.setNome(rs.getString("nome"));
00066                 produto.setPreco_venda(rs.getDouble("preco_venda"));
00067 
00068                 item.setProduto(produto);
00069                 resultado.add(item);
00070             }
00071         } catch (SQLException e) {
00072             e.printStackTrace();
00073         } finally {
00074             fecharRecursos();
00075         }
00076         return resultado;
00077     }
```

Este é o diagrama das funções utilizadas por essa função:



7.8.3.5 delete() void br.ufrn.loja.dao.ItemVendaDao.delete (int *id*)

```
00085                                {
00086
00087     }
```

7.8.3.6 existePorId() boolean br.ufrn.loja.dao.ItemVendaDao.existePorId (int *id*)

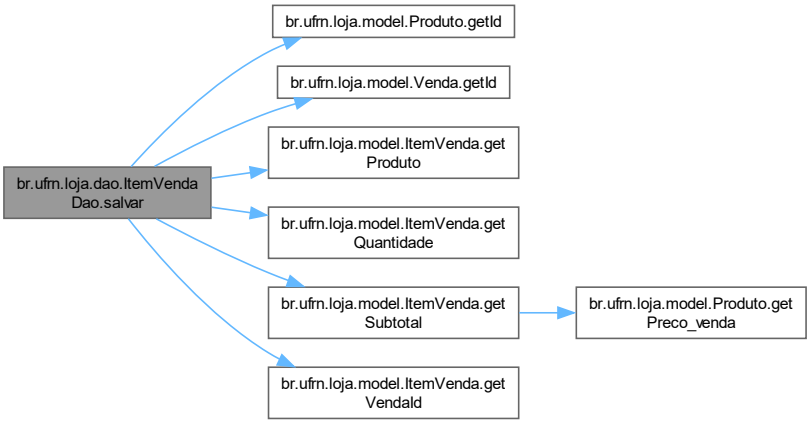
```
00090                                      {
00091         return false;
00092     }
```

7.8.3.7 salvar() void br.ufrn.loja.dao.ItemVendaDao.salvar (**ItemVenda** *obj*)

```
00034                                       {
00035
00036         try {
00037             pstmt = con.prepareStatement(INSERT);
00038             pstmt.setInt(1, obj.getVendaId().getId());
00039             pstmt.setInt(2, obj.getProduto().getId());
00040             pstmt.setInt(3, obj.getQuantidade());
00041             pstmt.setDouble(4, obj.getSubtotal());
00042
00043             pstmt.executeUpdate();
00044
00045         } catch (SQLException e) {
00046             e.printStackTrace();
```

```
00047         } finally {
00048             fecharRecursos();
00049         }
00050     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/dao/ **ItemVendaDao.java**

7.9 Referência da Classe br.ufrn.loja.LojaApplication

Classe com o metodo Main.

Membros Públicos Estáticos

- static void **main** (String[ ] args)

7.9.1 Descrição detalhada

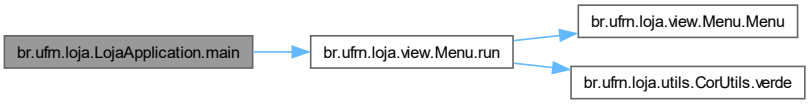
Classe com o metodo Main.

7.9.2 Documentação das funções

7.9.2.1 main() `static void br.ufrn.loja.LojaApplication.main ( String[] args ) [static]`

```
00012                                          {
00013         Menu.run();
00014     }
```

Este é o diagrama das funções utilizadas por essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/ **LojaApplication.java**

7.10 Referência da Classe br.ufrn.loja.view.Menu

Diagrama de hierarquia para br.ufrn.loja.view.Menu:

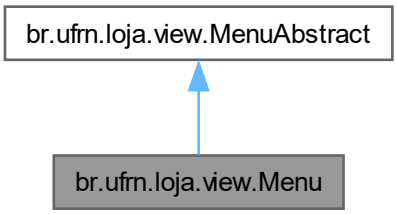
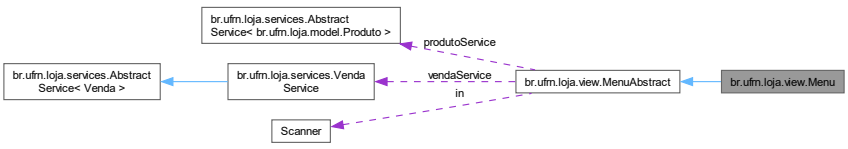


Diagrama de colaboração para br.ufrn.loja.view.Menu:



Métodos Públicos

- **Menu** ()
- void **exibir** ()

Métodos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- **Produto lerProduto** ()

  Método que lê os dados de um produto e o retorna.

Membros Públicos Estáticos

- static void **run** ()

Outros membros herdados

Atributos Estáticos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- static final int **SAIR** = 0
- static final int **CADASTRAR** = 1
- static final int **BUSCAR** = 2
- static final int **VER_TODOS** = 3
- static final int **FATURAMENTO** = 4
- static final int **VENDER** = 5
- static final int **REMOVER** = 6
- static final int **ALTERAR** = 7
- static final int **CANCELAR** = 8

Atributos Protegidos herdados de br.ufrn.loja.view.MenuAbstract

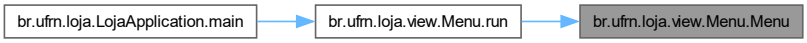
- **AbstractService**< **Produto** > **produtoService** = new **ProdutoService**()
- **VendaService vendaService** = new **VendaService**()
- int **opcao**
- boolean **saiu** = false
- Scanner **in** = new Scanner(System.in)

7.10.1 Construtores e Destrutores

7.10.1.1 Menu() br.ufrn.loja.view.Menu.Menu ( )

```
00010             {
00011         saiu = false;
00012         in = new Scanner(System.in);
00013     }
```

Esse é o diagrama das funções que utilizam essa função:



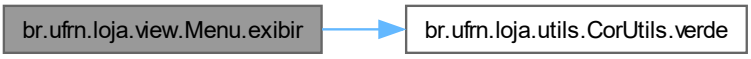
7.10.2 Documentação das funções

7.10.2.1 exibir() `void br.ufrn.loja.view.Menu.exibir ( )`

Faz a chamada do menu inicial, e mantem em um laço ate o usuario decidir encerrar.

```
00027                           {
00028         while (!saiu) {
00029             telaInicial();
00030             realizarAcao();
00031         }
00032         in.close();
00033         System.out.println(CorUtils.verde("Programa Encerrado!"));
00034     }
```

Este é o diagrama das funções utilizadas por essa função:

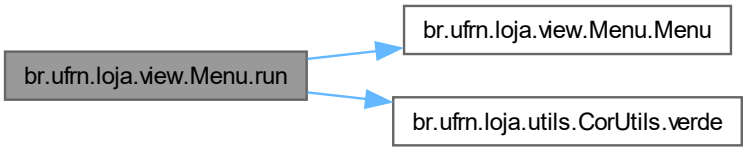


7.10.2.2 run() `static void br.ufrn.loja.view.Menu.run ( ) [static]`

Metodo responsavel por chamar o menu inicial

```
00018                               {
00019         System.out.println(CorUtils.verde("Iniciando Sistema..."));
00020         new Menu().exibir();
00021     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/ **Menu.java**

7.11 Referência da Classe br.ufrn.loja.view.MenuAbstract

Diagrama de hierarquia para br.ufrn.loja.view.MenuAbstract:

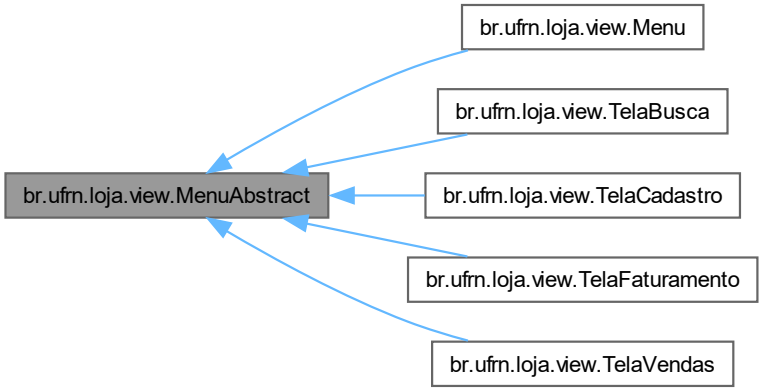
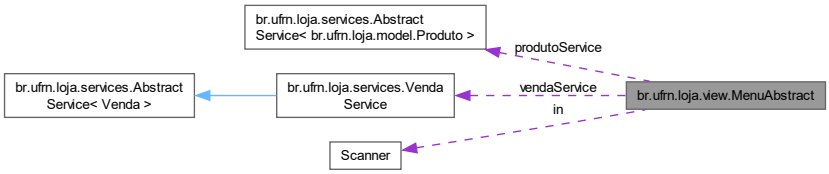


Diagrama de colaboração para br.ufrn.loja.view.MenuAbstract:



Métodos Públicos

- **Produto lerProduto** ()

  Método que lê os dados de um produto e o retorna.

Atributos Estáticos Públicos

- static final int **SAIR** = 0
- static final int **CADASTRAR** = 1
- static final int **BUSCAR** = 2
- static final int **VER_TODOS** = 3
- static final int **FATURAMENTO** = 4
- static final int **VENDER** = 5
- static final int **REMOVER** = 6
- static final int **ALTERAR** = 7
- static final int **CANCELAR** = 8

Atributos Protegidos

- **AbstractService**< **Produto** > **produtoService** = new **ProdutoService**()
- **VendaService** **vendaService** = new **VendaService**()
- int **opcao**
- boolean **saiu** = false
- Scanner **in** = new Scanner(System.in)

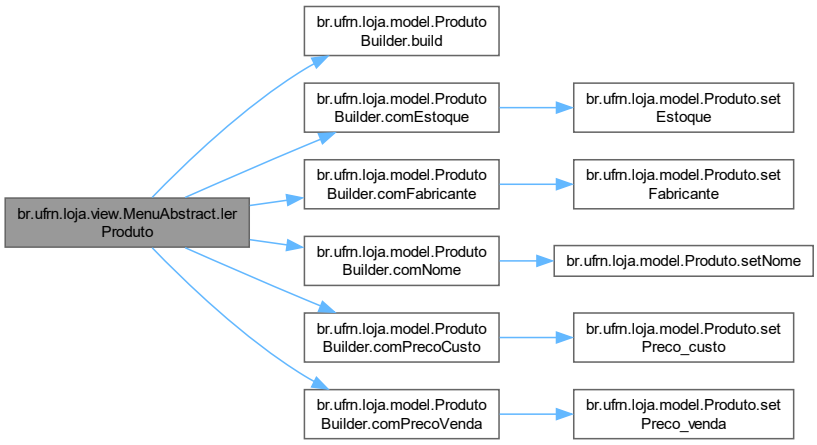
7.11.1 Documentação das funções

7.11.1.1 lerProduto() `Produto br.ufrn.loja.view.MenuAbstract.lerProduto ( )`

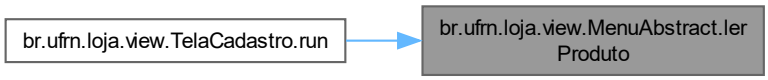
Método que lê os dados de um produto e o retorna.

```
00034                                   {
00035
00036         in.nextLine();
00037
00038         System.out.print("Digite o nome do produto: ");
00039         String nome = in.nextLine();
00040
00041         System.out.print("Digite o preço de custo do produto: ");
00042         double preco_custo = in.nextDouble();
00043
00044         System.out.print("Digite o preço de venda do produto: ");
00045         double preco_venda = in.nextDouble();
00046         in.nextLine();
00047
00048         System.out.print("Digite o estoque do produto: ");
00049         int estoque = in.nextInt();
00050         in.nextLine();
00051
00052         System.out.print("Digite o fabricante do produto: ");
00053         String fabricante = in.nextLine();
00054
00055         return new ProdutoBuilder()
00056                 .comNome(nome)
00057                 .comPrecoCusto(preco_custo)
00058                 .comPrecoVenda(preco_venda)
00059                 .comEstoque(estoque).comFabricante(fabricante).build();
00060     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



7.11.2 Atributos

7.11.2.1 ALTERAR `final int br.ufrn.loja.view.MenuAbstract.ALTERAR = 7 [static]`

7.11.2.2 BUSCAR `final int br.ufrn.loja.view.MenuAbstract.BUSCAR = 2 [static]`

7.11.2.3 CADASTRAR `final int br.ufrn.loja.view.MenuAbstract.CADASTRAR = 1 [static]`

7.11.2.4 CANCELAR `final int br.ufrn.loja.view.MenuAbstract.CANCELAR = 8 [static]`

7.11.2.5 FATURAMENTO `final int br.ufrn.loja.view.MenuAbstract.FATURAMENTO = 4 [static]`

7.11.2.6 in `Scanner br.ufrn.loja.view.MenuAbstract.in = new Scanner(System.in) [protected]`

7.11.2.7 opcao `int br.ufrn.loja.view.MenuAbstract.opcao [protected]`

7.11.2.8 produtoService `AbstractService< Produto> br.ufrn.loja.view.MenuAbstract.produtoService = new ProdutoService() [protected]`

7.11.2.9 REMOVER `final int br.ufrn.loja.view.MenuAbstract.REMOVER = 6 [static]`

7.11.2.10 SAIR `final int br.ufrn.loja.view.MenuAbstract.SAIR = 0 [static]`

7.11.2.11 saiu `boolean br.ufrn.loja.view.MenuAbstract.saiu = false [protected]`

7.11.2.12 vendaService `VendaService br.ufrn.loja.view.MenuAbstract.vendaService = new VendaService() [protected]`

7.11.2.13 VENDER `final int br.ufrn.loja.view.MenuAbstract.VENDER = 5 [static]`

7.11.2.14 VER_TODOS `final int br.ufrn.loja.view.MenuAbstract.VER_TODOS = 3 [static]`

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/ **MenuAbstract.java**

7.12 Referência da Classe br.ufrn.loja.exception.OpcaoInvalidaException

Diagrama de hierarquia para br.ufrn.loja.exception.OpcaoInvalidaException:

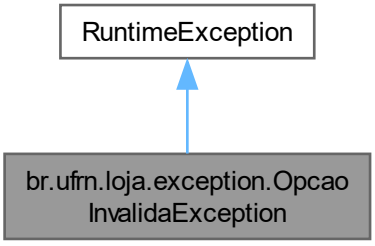
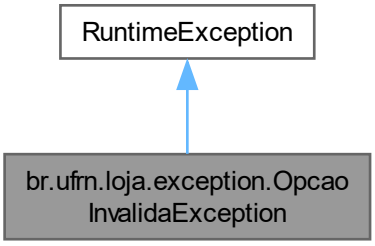


Diagrama de colaboração para br.ufrn.loja.exception.OpcaoInvalidaException:



Métodos Públicos

- **OpcaoInvalidaException** (String mensagem)

7.12.1 Construtores e Destrutores

7.12.1.1 OpcaoInvalidaException() br.ufrn.loja.exception.OpcaoInvalidaException.OpcaoInvalida↩ Exception (String *mensagem*)

```
00007                                           {
00008         super(mensagem);
00009     }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/exception/ **OpcaoInvalida↩ Exception.java**

7.13 Referência da Classe br.ufrn.loja.model.Produto

Métodos Públicos

- int **getId** ()
- void **setId** (int id)
- String **getNome** ()
- void **setNome** (String nome)
- double **getPreco_custo** ()
- void **setPreco_custo** (double preco_custo)
- double **getPreco_venda** ()
- void **setPreco_venda** (double preco_venda)
- int **getEstoque** ()
- void **setEstoque** (int estoque)
- String **getFabricante** ()
- void **setFabricante** (String fabricante)
- String **toString** ()

7.13.1 Descrição detalhada

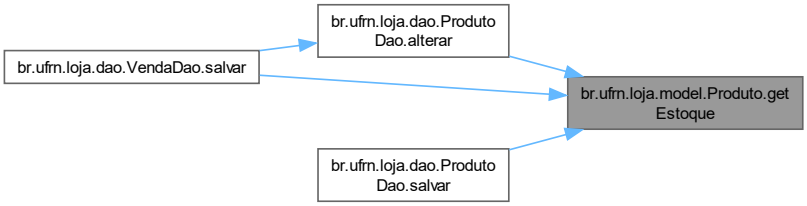
Classe responsavel pela entidade produto

7.13.2 Documentação das funções

7.13.2.1 getEstoque() `int br.ufrn.loja.model.Produto.getEstoque ( )`

```
00049                           {
00050         return estoque;
00051     }
```

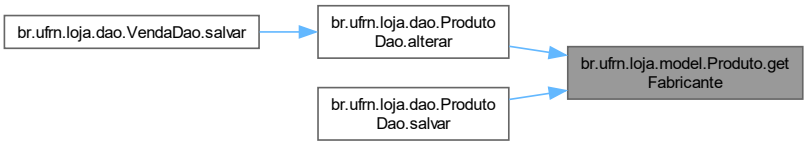
Esse é o diagrama das funções que utilizam essa função:



7.13.2.2 getFabricante() `String br.ufrn.loja.model.Produto.getFabricante ( )`

```
00057                                 {
00058         return fabricante;
00059     }
```

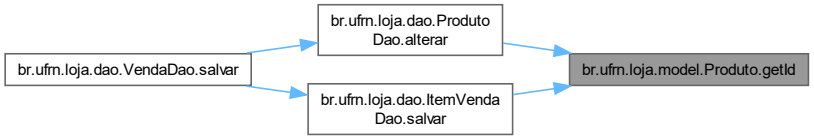
Esse é o diagrama das funções que utilizam essa função:



7.13.2.3 getId() `int br.ufrn.loja.model.Produto.getId ( )`

```
00017                       {
00018         return id;
00019     }
```

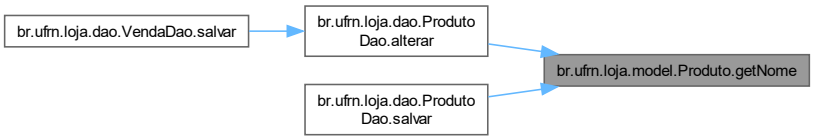
Esse é o diagrama das funções que utilizam essa função:



7.13.2.4 getNome() `String br.ufrn.loja.model.Produto.getNome ( )`

```
00025                           {
00026         return nome;
00027     }
```

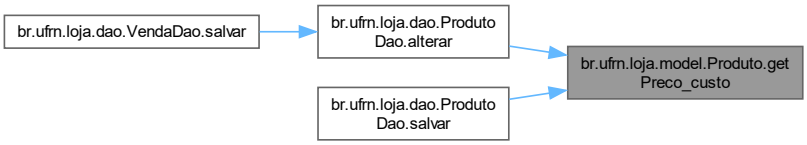
Esse é o diagrama das funções que utilizam essa função:



7.13.2.5 getPreco_custo() `double br.ufrn.loja.model.Produto.getPreco_custo ( )`

```
00033                                   {
00034         return preco_custo;
00035     }
```

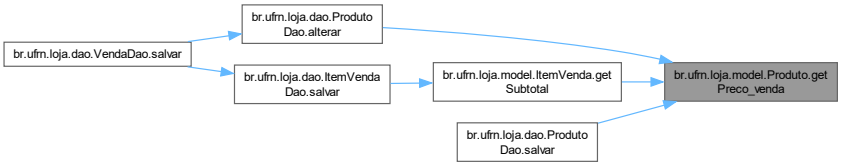
Esse é o diagrama das funções que utilizam essa função:



7.13.2.6 getPreco_venda() double br.ufrn.loja.model.Produto.getPreco_venda ( )

```
00041                                {
00042         return preco_venda;
00043     }
```

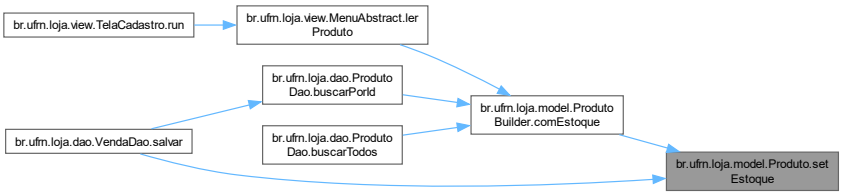
Esse é o diagrama das funções que utilizam essa função:



7.13.2.7 setEstoque() void br.ufrn.loja.model.Produto.setEstoque (int *estoque*)

```
00053                                    {
00054         this.estoque = estoque;
00055     }
```

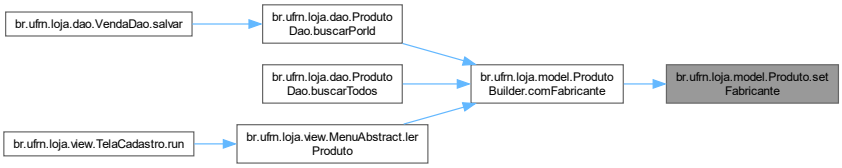
Esse é o diagrama das funções que utilizam essa função:



7.13.2.8 setFabricante() void br.ufrn.loja.model.Produto.setFabricante (String *fabricante*)

```
00061                                        {
00062         this.fabricante = fabricante;
00063     }
```

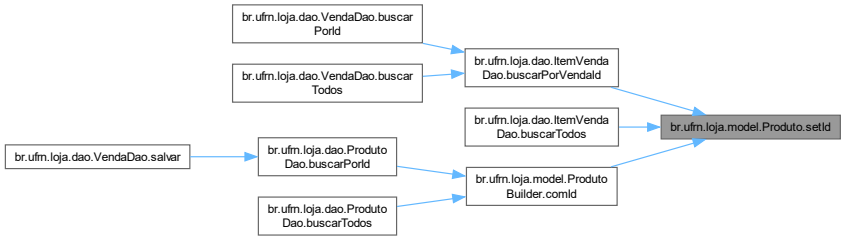
Esse é o diagrama das funções que utilizam essa função:



7.13.2.9 setId() void br.ufrn.loja.model.Produto.setId (int *id*)

```
00021                               {
00022         this.id = id;
00023     }
```

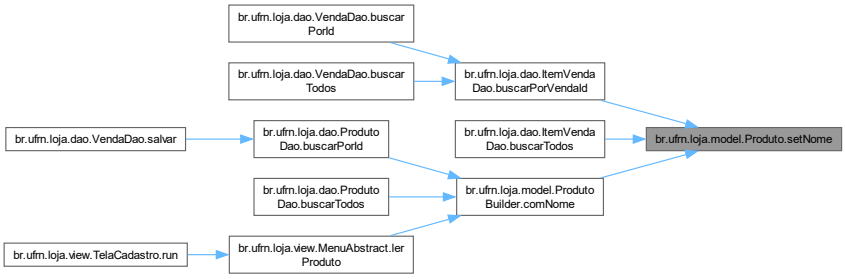
Esse é o diagrama das funções que utilizam essa função:



7.13.2.10 setNome() void br.ufrn.loja.model.Produto.setNome (String *nome*)

```
00029                                 {
00030         this.nome = nome;
00031     }
```

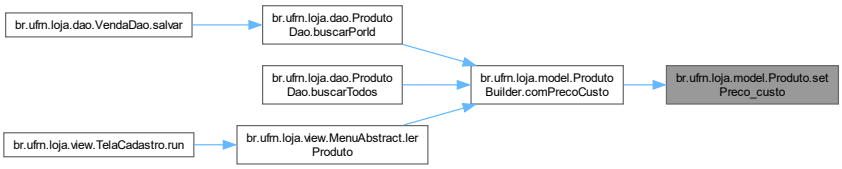
Esse é o diagrama das funções que utilizam essa função:



7.13.2.11 setPreco_custo() void br.ufrn.loja.model.Produto.setPreco_custo (double *preco_custo*)

```
00037                                           {
00038         this.preco_custo = preco_custo;
00039     }
```

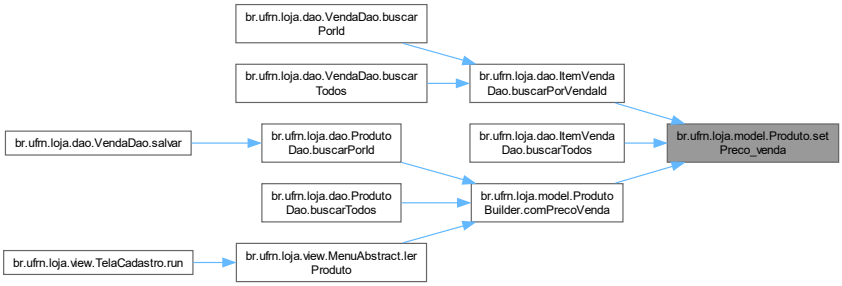
Esse é o diagrama das funções que utilizam essa função:



7.13.2.12 setPreco_venda() `void br.ufrn.loja.model.Produto.setPreco_venda ( double preco_venda )`

```
00045                                                {
00046         this.preco_venda = preco_venda;
00047     }
```

Esse é o diagrama das funções que utilizam essa função:



7.13.2.13 toString() `String br.ufrn.loja.model.Produto.toString ( )`

```
00066                                {
00067         return String.format("%-5d%-28s%-15.2f%-15.2f%-10d%s%n", id, nome, preco_custo, preco_venda,
      estoque, fabricante);
00068     }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/ **Produto.java**

7.14 Referência da Classe br.ufrn.loja.model.ProdutoBuilder

Métodos Públicos

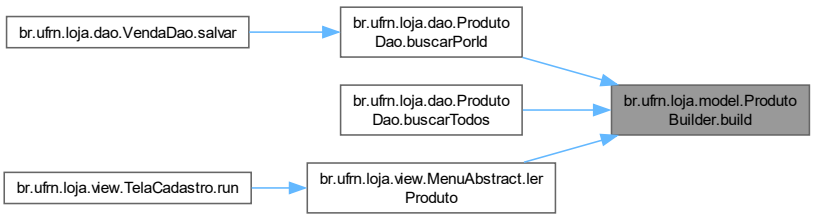
- **ProdutoBuilder comId** (int valor)
- **ProdutoBuilder comNome** (String valor)
- **ProdutoBuilder comPrecoCusto** (double valor)
- **ProdutoBuilder comPrecoVenda** (double valor)
- **ProdutoBuilder comEstoque** (int valor)
- **ProdutoBuilder comFabricante** (String valor)
- **Produto build** ()

7.14.1 Documentação das funções

7.14.1.1 build() `Produto br.ufrn.loja.model.ProdutoBuilder.build ( )`

```
00037                           {
00038         return this.produto;
00039     }
```

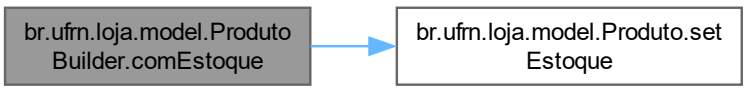
Esse é o diagrama das funções que utilizam essa função:



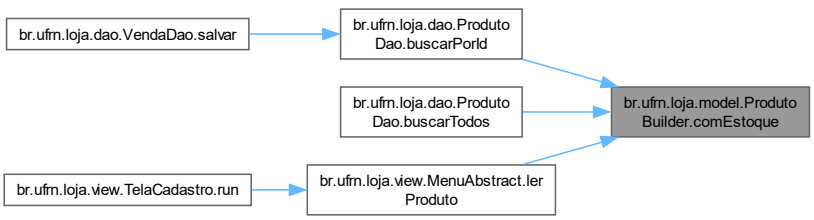
7.14.1.2 comEstoque() `ProdutoBuilder br.ufrn.loja.model.ProdutoBuilder.comEstoque ( int valor )`

```
00027                                           {
00028         produto.setEstoque(valor);
00029         return this;
00030     }
```

Este é o diagrama das funções utilizadas por essa função:



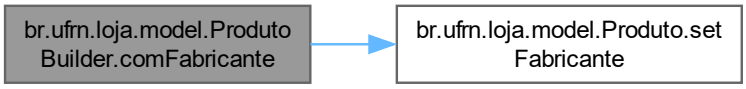
Esse é o diagrama das funções que utilizam essa função:



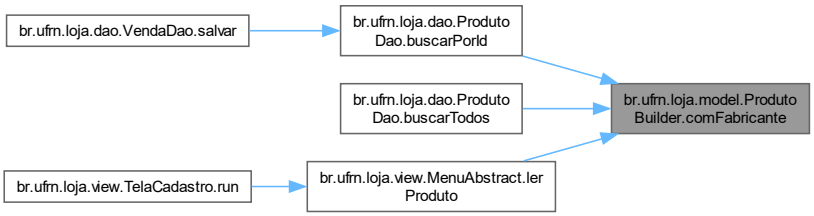
7.14.1.3 comFabricante() `ProdutoBuilder br.ufrn.loja.model.ProdutoBuilder.comFabricante ( String valor )`

```
00032                                                {
00033         produto.setFabricante(valor);
00034         return this;
00035     }
```

Este é o diagrama das funções utilizadas por essa função:



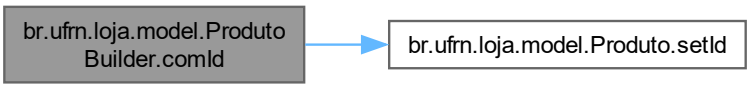
Esse é o diagrama das funções que utilizam essa função:



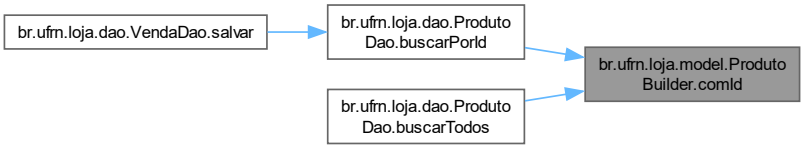
7.14.1.4 comId() `ProdutoBuilder br.ufrn.loja.model.ProdutoBuilder.comId ( int valor )`

```
00007                                        {
00008         produto.setId(valor);
00009         return this;
00010     }
```

Este é o diagrama das funções utilizadas por essa função:



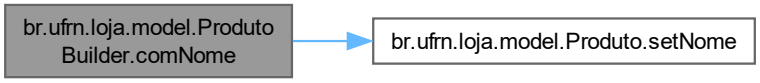
Esse é o diagrama das funções que utilizam essa função:



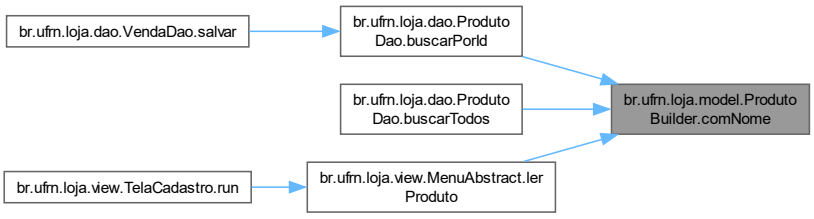
7.14.1.5 comNome() `ProdutoBuilder br.ufrn.loja.model.ProdutoBuilder.comNome ( String valor )`

```
00012                                                {
00013         produto.setNome(valor);
00014         return this;
00015     }
```

Este é o diagrama das funções utilizadas por essa função:



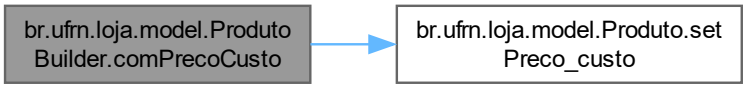
Esse é o diagrama das funções que utilizam essa função:



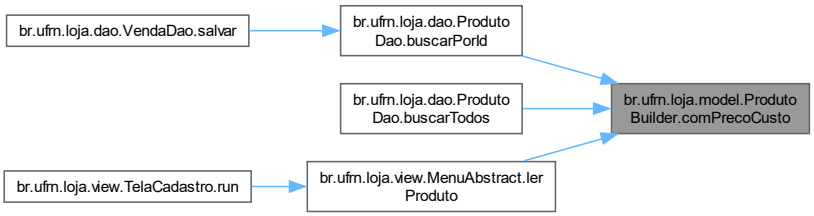
7.14.1.6 comPrecoCusto() `ProdutoBuilder br.ufrn.loja.model.ProdutoBuilder.comPrecoCusto ( double valor )`

```
00017                                                {
00018         produto.setPreco_custo(valor);
00019         return this;
00020     }
```

Este é o diagrama das funções utilizadas por essa função:



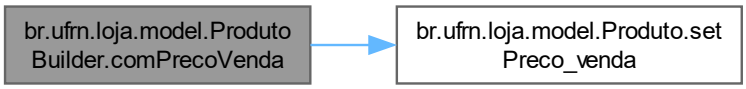
Esse é o diagrama das funções que utilizam essa função:



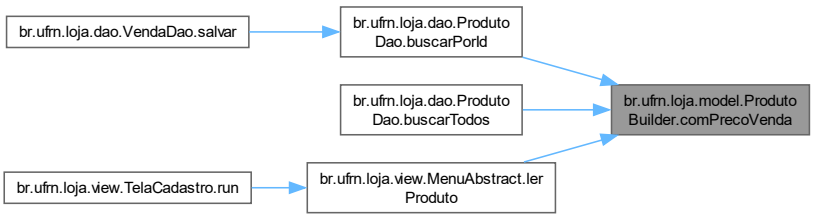
7.14.1.7 comPrecoVenda() `ProdutoBuilder br.ufrn.loja.model.ProdutoBuilder.comPrecoVenda ( double valor )`

```
00022                                                {
00023         produto.setPreco_venda(valor);
00024         return this;
00025     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/ **ProdutoBuilder.java**

7.15 Referência da Classe br.ufrn.loja.dao.ProdutoDao

Implementação do DAO para a entidade Produto.

Diagrama de hierarquia para br.ufrn.loja.dao.ProdutoDao:

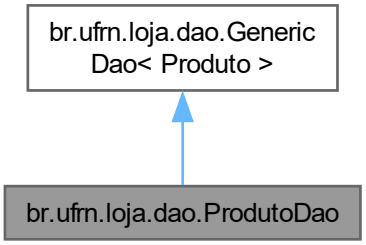
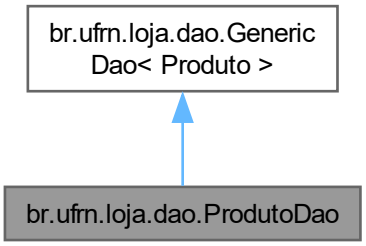


Diagrama de colaboração para br.ufrn.loja.dao.ProdutoDao:



Métodos Públicos

- **ProdutoDao** ()
- void **salvar** (**Produto** obj)
- List< **Produto** > **buscarTodos** ()
- **Produto** **buscarPorId** (int id)
- void **delete** (int id)
- boolean **existePorId** (int id)
- void **alterar** (**Produto** obj)
- void **alterarEstoque** (int id, int novoEstoque)

Métodos Públicos herdados de br.ufrn.loja.dao.GenericDao< Produto >

- void **salvar** (E obj)
- List< E > **buscarTodos** ()
- E **buscarPorId** (int id)
- void **delete** (int id)
- boolean **existePorId** (int id)
- void **alterar** (E obj)

7.15.1 Descrição detalhada

Implementação do DAO para a entidade Produto.

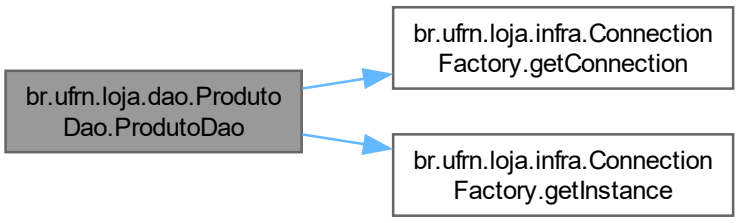
```
Esta classe é responsável por interagir com o banco de dados para
realizar operações relacionadas a produtos.
```

7.15.2 Construtores e Destrutores

7.15.2.1 ProdutoDao() `br.ufrn.loja.dao.ProdutoDao.ProdutoDao ( )`

```
00030                 {
00031         con = ConnectionFactory.getInstance().getConnection();
00032     }
```

Este é o diagrama das funções utilizadas por essa função:

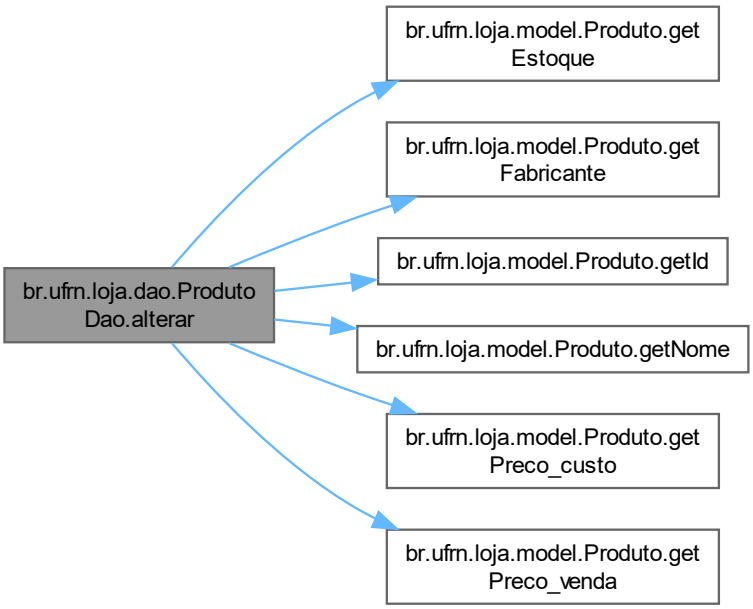


7.15.3 Documentação das funções

7.15.3.1 alterar() void br.ufrn.loja.dao.ProdutoDao.alterar (Produto obj)

```
00126                                       {
00127         try {
00128             statement = con.createStatement();
00129             String sql = "UPDATE produto SET nome = '" + obj.getNome() + "', preco_custo = " +
      obj.getPreco_custo()
00130                     + ", preco_venda = " + obj.getPreco_venda() + ", estoque = " + obj.getEstoque() +
      ", fabricante = '"
00131                     + obj.getFabricante() + "' WHERE id = " + obj.getId();
00132             statement.executeUpdate(sql);
00133         } catch (SQLException e) {
00134             e.printStackTrace();
00135         } finally {
00136             fecharRecursos();
00137         }
00138     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:

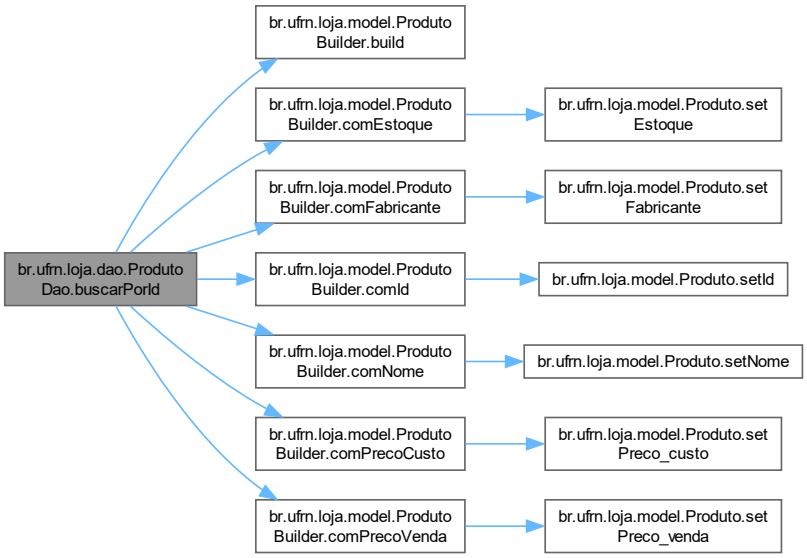
7.15.3.2 alterarEstoque() void br.ufrn.loja.dao.ProdutoDao.alterarEstoque (int *id*, int *novoEstoque*)

```
00140                                                   {
00141         try {
00142             statement = con.createStatement();
00143             String sql = "UPDATE produto SET estoque = " + novoEstoque + " WHERE id = " + id;
00144             statement.executeUpdate(sql);
00145         } catch (SQLException e) {
00146             e.printStackTrace();
00147         } finally {
00148             fecharRecursos();
00149         }
00150     }
```

7.15.3.3 buscarPorId() Produto br.ufrn.loja.dao.ProdutoDao.buscarPorId (int *id*)

```
00076                                       {
00077         try {
00078             statement = con.createStatement();
00079             ResultSet result = statement.executeQuery(SELECT + id);
00080             if (result.next()) {
00081                 Produto produto = new ProdutoBuilder()
00082                         .comId(result.getInt("id"))
00083                         .comNome(result.getString("nome"))
00084                         .comPrecoCusto(result.getDouble("preco_custo"))
00085                         .comPrecoVenda(result.getDouble("preco_venda"))
00086                         .comEstoque(result.getInt("estoque"))
00087                         .comFabricante(result.getString("fabricante")).build();
00088                 return produto;
00089             }
00090         } catch (SQLException e) {
00091             e.printStackTrace();
00092         } finally {
00093             fecharRecursos();
00094         }
00095         return null;
00096     }
```

Este é o diagrama das funções utilizadas por essa função:

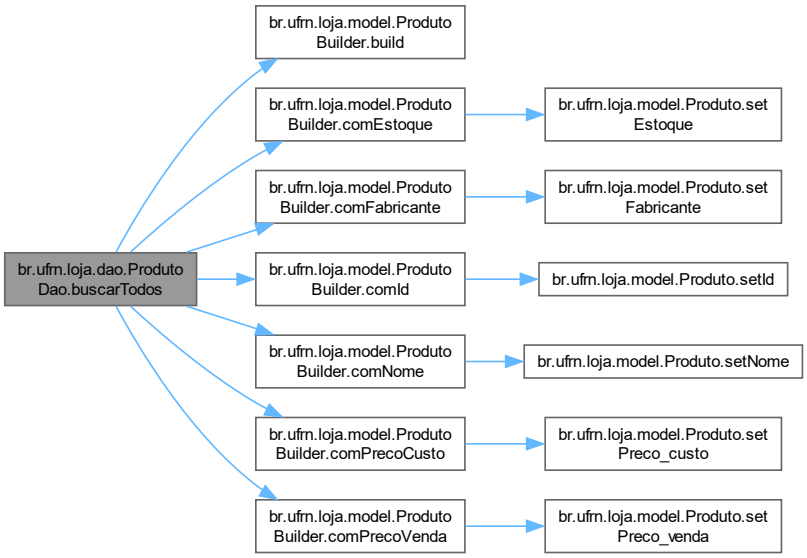


Esse é o diagrama das funções que utilizam essa função:

7.15.3.4 buscarTodos() List< Produto > br.ufrn.loja.dao.ProdutoDao.buscarTodos ( )

```
00050                                       {
00051         List<Produto> resultado = new ArrayList<>();
00052         try {
00053             statement = con.createStatement();
00054             ResultSet rs = statement.executeQuery(SELECT_ALL);
00055             while (rs.next()) {
00056 
00057                 Produto produto = new ProdutoBuilder()
00058                         .comId(rs.getInt("id"))
00059                         .comNome(rs.getString("nome"))
00060                         .comPrecoCusto(rs.getDouble("preco_custo"))
00061                         .comPrecoVenda(rs.getDouble("preco_venda"))
00062                         .comEstoque(rs.getInt("estoque"))
00063                         .comFabricante(rs.getString("fabricante")).build();
00064 
00065                 resultado.add(produto);
00066             }
00067         } catch (SQLException e) {
00068             e.printStackTrace();
00069         } finally {
00070             fecharRecursos();
00071         }
00072         return resultado;
00073     }
```

Este é o diagrama das funções utilizadas por essa função:



7.15.3.5 delete() void br.ufrn.loja.dao.ProdutoDao.delete (int *id*)

```
00099                               {
00100         try {
00101             statement = con.createStatement();
00102             statement.executeUpdate(DELETE + id);
00103         } catch (SQLException e) {
00104             e.printStackTrace();
00105         } finally {
00106             fecharRecursos();
00107         }
00108 
00109     }
```

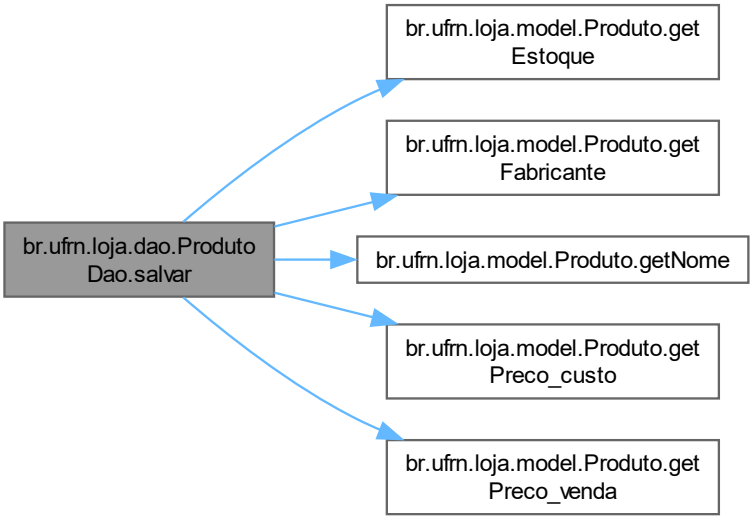
7.15.3.6 existePorId() boolean br.ufrn.loja.dao.ProdutoDao.existePorId (int *id*)

```
00112                                       {
00113         try {
00114             statement = con.createStatement();
00115             ResultSet result = statement.executeQuery(EXISTE + id);
00116             return result.next() && result.getInt(1) > 0;
00117         } catch (SQLException e) {
00118             e.printStackTrace();
00119         } finally {
00120             fecharRecursos();
00121         }
00122         return false;
00123     }
```

7.15.3.7 salvar() void br.ufrn.loja.dao.ProdutoDao.salvar (**Produto** *obj*)

```
00036                                     {
00037         try {
00038             statement = con.createStatement();
00039             statement.executeUpdate(INSERT.replace("%", "'" + obj.getNome() + "', " +
      obj.getPreco_custo() + ", "
00040                     + obj.getPreco_venda() + ", " + obj.getEstoque() + ", '" + obj.getFabricante() +
      "'"));
00041 
00042         } catch (Exception e) {
00043             e.printStackTrace();
00044         } finally {
00045             fecharRecursos();
00046         }
00047     }
```

Este é o diagrama das funções utilizadas por essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/dao/ **ProdutoDao.java**

7.16 Referência da Classe br.ufrn.loja.services.ProdutoService

Diagrama de hierarquia para br.ufrn.loja.services.ProdutoService:

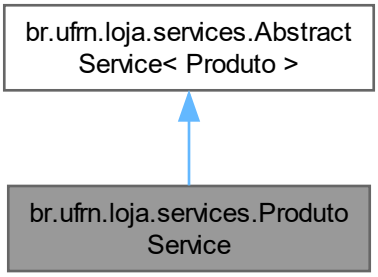
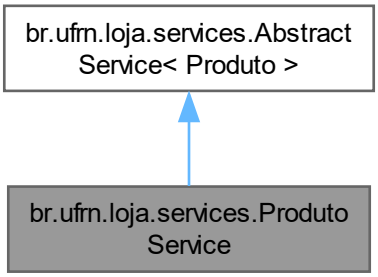


Diagrama de colaboração para br.ufrn.loja.services.ProdutoService:



Métodos Públicos

- **ProdutoService** ()
- boolean **buscar** ()

Métodos Públicos herdados de br.ufrn.loja.services.AbstractService< Produto >

- void **processar** (int opcao)
  Processa a operação solicitada.
- abstract boolean **buscar** ()
- E **getObjeto** ()
- void **setObjeto** (E objeto)

Membros Protegidos

- boolean **validar** ()
- void **imprimir** ()
- void **remover** ()

Membros Protegidos herdados de br.ufrn.loja.services.AbstractService< Produto >

- void **cadastrar** ()
  Método que prepara para o cadastro de um objeto.
- abstract boolean **validar** ()
- void **alterar** ()
  Método abstrato para alterar o objeto.
- abstract void **remover** ()
  Método abstrato para remover o objeto.
- abstract void **imprimir** ()

Outros membros herdados

Atributos Protegidos herdados de br.ufrn.loja.services.AbstractService< Produto >

- E **objeto**
- **GenericDao**< E > **dao**

7.16.1 Construtores e Destrutores

7.16.1.1 ProdutoService() br.ufrn.loja.services.ProdutoService.ProdutoService ( )

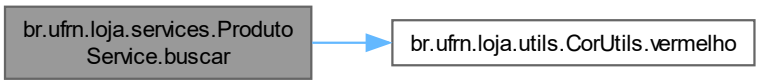
```
00010                               {
00011         dao = new ProdutoDao();
00012     }
```

7.16.2 Documentação das funções

7.16.2.1 buscar() boolean br.ufrn.loja.services.ProdutoService.buscar ( )

```
00043                               {
00044         if(dao.existePorId(objeto.getId())) {
00045             System.out.printf("\n%-5s%-20s%-15s%-15s%-10s%s\n", "ID", "Nome", "Preço Custo", "Preço
      Venda", "Estoque", "Fabricante");
00046             System.out.println(dao.buscarPorId(objeto.getId()));
00047             return true;
00048         }else
00049             System.out.println(CorUtils.vermelho("Esse produto não existe!"));
00050         return false;
00051     }
```

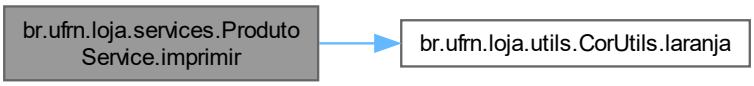
Este é o diagrama das funções utilizadas por essa função:



7.16.2.2 imprimir() void br.ufrn.loja.services.ProdutoService.imprimir ( ) [protected]

```
00027                               {
00028         System.out.println("\n------ "+CorUtils.laranja("Lista de Produtos")+"
      ----------------------");
00029         System.out.printf("%-5s%-25s%-15s%-15s%-10s%s\n", "ID", "Nome", "Preço Custo", "Preço Venda",
      "Estoque", "Fabricante");
00030         dao.buscarTodos().forEach(System.out::print);
00031         System.out.println("-----------------------------------------------");
00032 
00033     }
```

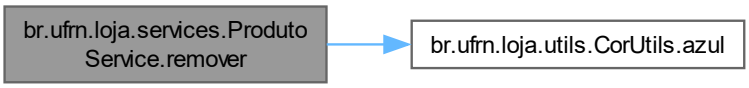
Este é o diagrama das funções utilizadas por essa função:



7.16.2.3 remover() void br.ufrn.loja.services.ProdutoService.remover ( ) [protected]

```
00036                               {
00037         if(dao.existePorId(objeto.getId())) {
00038             dao.delete(objeto.getId());
00039             System.out.println(CorUtils.azul("Produto removido com sucesso!!"));
00040         }
00041     }
```

Este é o diagrama das funções utilizadas por essa função:



7.16.2.4 validar() boolean br.ufrn.loja.services.ProdutoService.validar ( ) [protected]

```
00017                               {
00018         if(objeto.getNome() == null || objeto.getNome().isEmpty()) {
00019             System.out.println("\nVocê não digitou o nome do produto\n");
00020             return false;
00021         }
00022         return true;
00023     }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/services/ **ProdutoService.java**

7.17 Referência da enumeração br.ufrn.loja.model.StatusVenda

Métodos Públicos

- **StatusVenda** (String descricao)
- String **getDescricao** ()

Membros Públicos Estáticos

- static **StatusVenda** **fromDescricao** (String descricao)

Atributos Públicos

- **CONCLUIDA** =("Concluida")
- **CANCELADA** =("Cancelada")

7.17.1 Construtores e Destrutores

7.17.1.1 StatusVenda() br.ufrn.loja.model.StatusVenda.StatusVenda (String *descricao*)

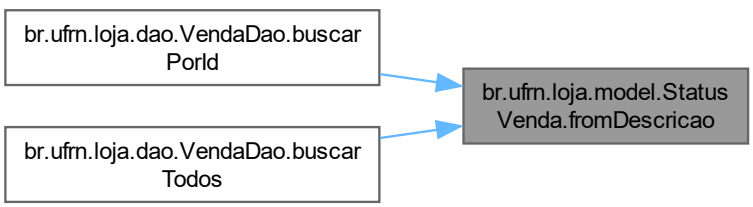
```
00008                                {
00009         this.descricao = descricao;
00010     }
```

7.17.2 Documentação das funções

7.17.2.1 fromDescricao() static **StatusVenda** br.ufrn.loja.model.StatusVenda.fromDescricao (String *descricao*) [static]

```
00016                                                    {
00017         for (StatusVenda status : StatusVenda.values()) {
00018             if (status.descricao.equalsIgnoreCase(descricao)) {
00019                 return status;
00020             }
00021         }
00022         throw new IllegalArgumentException("Nenhum status de venda encontrado com a descrição: " +
      descricao);
00023     }
```

Esse é o diagrama das funções que utilizam essa função:



7.17.2.2 getDescricao() String br.ufrn.loja.model.StatusVenda.getDescricao ( )

```
00012                                {
00013         return descricao;
00014     }
```

7.17.3 Atributos

7.17.3.1 CANCELADA br.ufrn.loja.model.StatusVenda.CANCELADA =("Cancelada")

7.17.3.2 CONCLUIDA br.ufrn.loja.model.StatusVenda.CONCLUIDA =("Concluida")

A documentação para essa enumeração foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/ **StatusVenda.java**

7.18 Referência da Classe br.ufrn.loja.view.TelaBusca

Diagrama de hierarquia para br.ufrn.loja.view.TelaBusca:

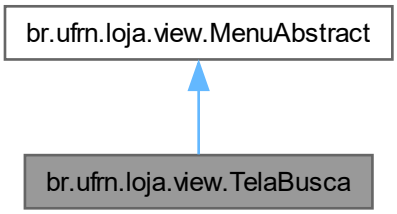
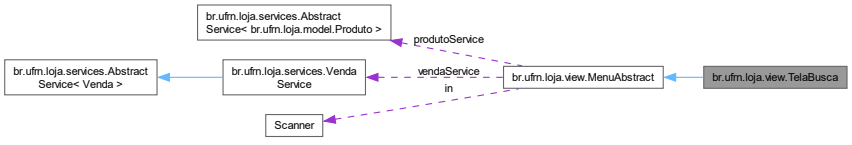


Diagrama de colaboração para br.ufrn.loja.view.TelaBusca:



Métodos Públicos

- **TelaBusca** (Scanner **in**)
- void **run** (int **opcao**)

Métodos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- **Produto lerProduto** ()

  Método que lê os dados de um produto e o retorna.

Outros membros herdados

Atributos Estáticos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- static final int **SAIR** = 0
- static final int **CADASTRAR** = 1
- static final int **BUSCAR** = 2
- static final int **VER_TODOS** = 3
- static final int **FATURAMENTO** = 4
- static final int **VENDER** = 5
- static final int **REMOVER** = 6
- static final int **ALTERAR** = 7
- static final int **CANCELAR** = 8

Atributos Protegidos herdados de br.ufrn.loja.view.MenuAbstract

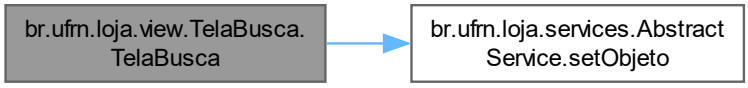
- **AbstractService**< **Produto** > **produtoService** = new **ProdutoService**()
- **VendaService vendaService** = new **VendaService**()
- int **opcao**
- boolean **saiu** = false
- Scanner **in** = new Scanner(System.in)

7.18.1 Construtores e Destrutores

7.18.1.1 TelaBusca() br.ufrn.loja.view.TelaBusca.TelaBusca (Scanner *in*)

```
00010                                   {
00011         this.in = in;
00012         produtoService.setObjeto(new Produto());
00013     }
```

Este é o diagrama das funções utilizadas por essa função:



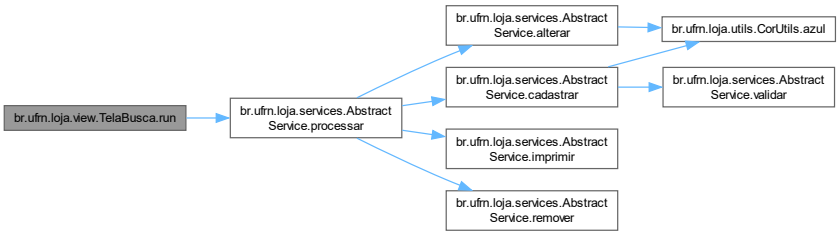
7.18.2 Documentação das funções

7.18.2.1 run() `void br.ufrn.loja.view.TelaBusca.run ( int opcao )`

Método que inicia a execução da TelaBusca.

```
00018                                {
00019         this.opcao = opcao;
00020         switch (this.opcao) {
00021         case VER_TODOS:
00022             produtoService.processar(VER_TODOS);
00023             break;
00024         case BUSCAR:
00025             buscarProduto();
00026             break;
00027         default:
00028             throw new OpcaoInvalidaException("Opção inválida! Por favor, escolha uma opção válida.");
00029         }
00030     }
```

Este é o diagrama das funções utilizadas por essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/ **TelaBusca.java**

7.19 Referência da Classe br.ufrn.loja.view.TelaCadastro

Diagrama de hierarquia para br.ufrn.loja.view.TelaCadastro:

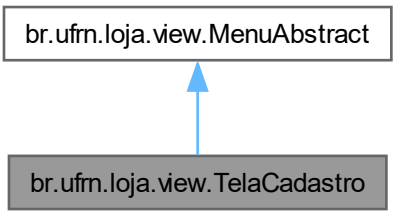
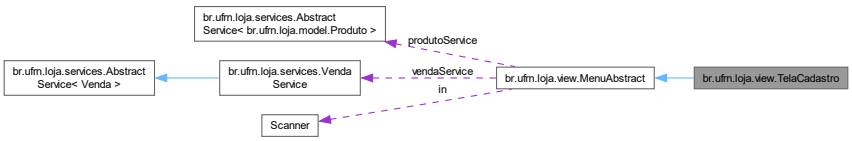


Diagrama de colaboração para br.ufrn.loja.view.TelaCadastro:



Métodos Públicos

- **TelaCadastro** (Scanner **in**)
- void **run** ()

  Método que inicia a execução do processo de cadastro.
- void **continuar** ()

  Método que pergunta ao usuário se deseja continuar cadastrando.

Métodos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- **Produto lerProduto** ()

  Método que lê os dados de um produto e o retorna.

Outros membros herdados

Atributos Estáticos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- static final int **SAIR** = 0
- static final int **CADASTRAR** = 1
- static final int **BUSCAR** = 2
- static final int **VER_TODOS** = 3
- static final int **FATURAMENTO** = 4
- static final int **VENDER** = 5
- static final int **REMOVER** = 6
- static final int **ALTERAR** = 7
- static final int **CANCELAR** = 8

Atributos Protegidos herdados de br.ufrn.loja.view.MenuAbstract

- **AbstractService**< **Produto** > **produtoService** = new **ProdutoService**()
- **VendaService vendaService** = new **VendaService**()
- int **opcao**
- boolean **saiu** = false
- Scanner **in** = new Scanner(System.in)

7.19.1 Construtores e Destrutores

7.19.1.1 TelaCadastro() br.ufrn.loja.view.TelaCadastro.TelaCadastro (Scanner *in*)

```
00007                                   {
00008         this.in = in;
00009     }
```

7.19.2 Documentação das funções

7.19.2.1 continuar() void br.ufrn.loja.view.TelaCadastro.continuar ( )

Método que pergunta ao usuário se deseja continuar cadastrando.

```
00026                                {
00027         System.out.println("\n[Digite " + Menu.SAIR + " para sair ou qualquer numero para continuar
      cadastrando.]");
00028         this.opcao = in.nextInt();
00029         if (opcao == Menu.SAIR)
00030             this.saiu = true;
00031     }
```

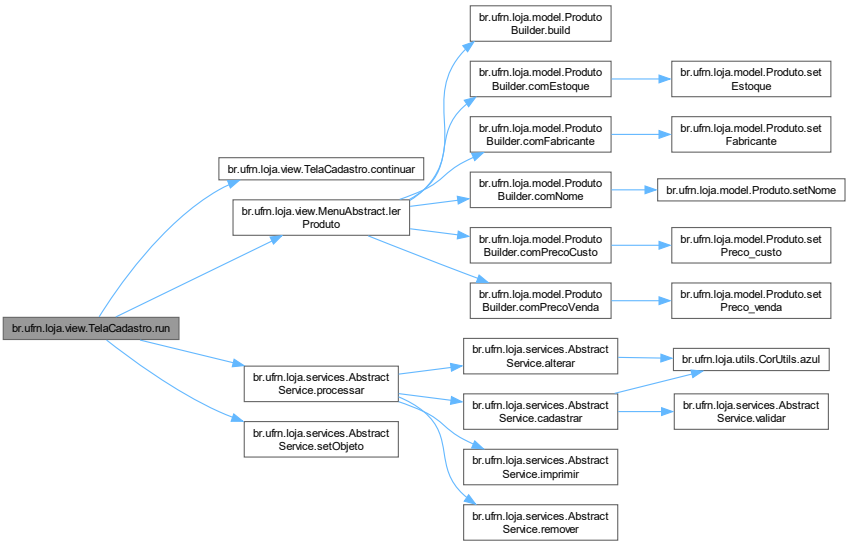
Esse é o diagrama das funções que utilizam essa função:

7.19.2.2 run() void br.ufrn.loja.view.TelaCadastro.run ( )

Método que inicia a execução do processo de cadastro.

```
00014                           {
00015         while (!saiu) {
00016             produtoService.setObjeto(lerProduto());
00017             produtoService.processar(Menu.CADASTRAR);
00018             continuar();
00019         }
00020
00021     }
```

Este é o diagrama das funções utilizadas por essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/ **TelaCadastro.java**

7.20 Referência da Classe br.ufrn.loja.view.TelaFaturamento

Diagrama de hierarquia para br.ufrn.loja.view.TelaFaturamento:

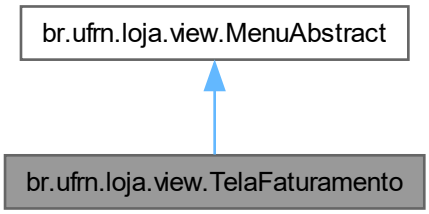
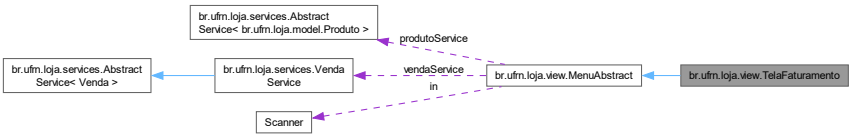


Diagrama de colaboração para br.ufrn.loja.view.TelaFaturamento:



Métodos Públicos

- **TelaFaturamento** (Scanner **in**)
- void **run** ()

Métodos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- **Produto lerProduto** ()

  Método que lê os dados de um produto e o retorna.

Outros membros herdados

Atributos Estáticos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- static final int **SAIR** = 0
- static final int **CADASTRAR** = 1
- static final int **BUSCAR** = 2
- static final int **VER_TODOS** = 3
- static final int **FATURAMENTO** = 4
- static final int **VENDER** = 5
- static final int **REMOVER** = 6
- static final int **ALTERAR** = 7
- static final int **CANCELAR** = 8

Atributos Protegidos herdados de br.ufrn.loja.view.MenuAbstract

- **AbstractService**< **Produto** > **produtoService** = new **ProdutoService**()
- **VendaService vendaService** = new **VendaService**()
- int **opcao**
- boolean **saiu** = false
- Scanner **in** = new Scanner(System.in)

7.20.1 Descrição detalhada

Tela responsável por exibir o faturamento no período especificado.

7.20.2 Construtores e Destrutores

7.20.2.1 TelaFaturamento() br.ufrn.loja.view.TelaFaturamento.TelaFaturamento (Scanner *in*)

```
00012                                   {
00013         this.in = in;
00014     }
```

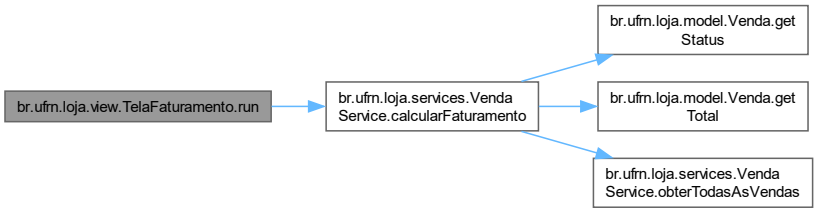
7.20.3 Documentação das funções

7.20.3.1 run() `void br.ufrn.loja.view.TelaFaturamento.run ( )`

Executa a lógica da tela para exibir o faturamento no período especificado.

```
00019                     {
00020         System.out.println("Digite o período no formato (yyyy-mm-dd)");
00021         System.out.print("Inicio: ");
00022         in.nextLine();
00023         String inicio = in.nextLine();
00024 
00025         System.out.print("Termino: ");
00026         String termino = in.nextLine();
00027 
00028         exibirResultadoFaturamento(inicio, termino, vendaService.calcularFaturamento(inicio,
      termino));
00029     }
```

Este é o diagrama das funções utilizadas por essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/ **TelaFaturamento.java**

7.21 Referência da Classe br.ufrn.loja.view.TelaVendas

Classe que representa a tela de vendas e suas funcionalidades.

Diagrama de hierarquia para br.ufrn.loja.view.TelaVendas:

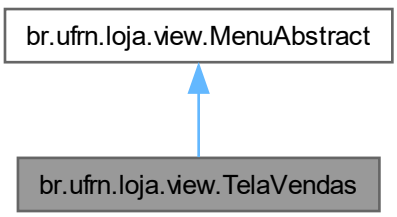
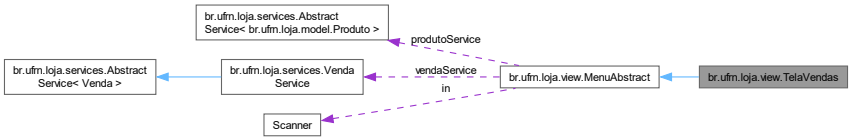


Diagrama de colaboração para br.ufrn.loja.view.TelaVendas:



Métodos Públicos

- **TelaVendas** (Scanner **in**)

  Construtor da classe TelaVendas.
- void **exibirMenuVendas** ()

  Exibe o menu de vendas e processa as opções escolhidas pelo usuário.
- void **exibirOpcoesVendas** ()

  Exibe as opções do menu de vendas.
- void **imprimirVenda** (**Venda** venda)

  Imprime os detalhes de uma venda.

Métodos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- **Produto lerProduto** ()

  Método que lê os dados de um produto e o retorna.

Outros membros herdados

Atributos Estáticos Públicos herdados de br.ufrn.loja.view.MenuAbstract

- static final int **SAIR** = 0
- static final int **CADASTRAR** = 1
- static final int **BUSCAR** = 2
- static final int **VER_TODOS** = 3
- static final int **FATURAMENTO** = 4
- static final int **VENDER** = 5
- static final int **REMOVER** = 6
- static final int **ALTERAR** = 7
- static final int **CANCELAR** = 8

Atributos Protegidos herdados de br.ufrn.loja.view.MenuAbstract

- **AbstractService**< **Produto** > **produtoService** = new **ProdutoService**()
- **VendaService vendaService** = new **VendaService**()
- int **opcao**
- boolean **saiu** = false
- Scanner **in** = new Scanner(System.in)

7.21.1 Descrição detalhada

Classe que representa a tela de vendas e suas funcionalidades.

7.21.2 Construtores e Destrutores

7.21.2.1 TelaVendas() `br.ufrn.loja.view.TelaVendas.TelaVendas ( Scanner in )`

Construtor da classe TelaVendas.

Parâmetros

in	Scanner para entrada de dados.

```
00027                                   {
00028         this.in = in;
00029     }
```

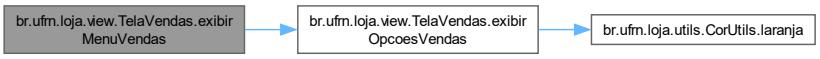
7.21.3 Documentação das funções

7.21.3.1 exibirMenuVendas() `void br.ufrn.loja.view.TelaVendas.exibirMenuVendas ( )`

Exibe o menu de vendas e processa as opções escolhidas pelo usuário.

```
00034                                   {
00035         while (!saiu) {
00036             exibirOpcoesVendas();
00037             processarOpcaoVendas();
00038         }
00039     }
```

Este é o diagrama das funções utilizadas por essa função:

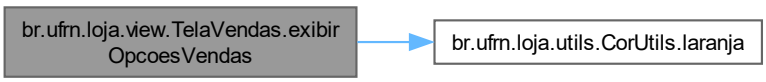


7.21.3.2 exibirOpcoesVendas() void br.ufrn.loja.view.TelaVendas.exibirOpcoesVendas ( )

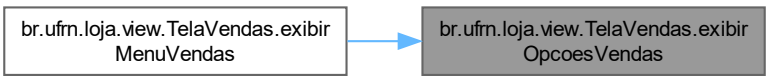
Exibe as opções do menu de vendas.

```
00044                                       {
00045
00046         System.out.println("");
00047         System.out.println("           " + CorUtils.laranja("MENU DE VENDAS") + "             ");
00048         System.out.println("");
00049         System.out.println("  " + CADASTRAR + ". Iniciar Nova Venda                ");
00050         System.out.println("  " + BUSCAR + ". Buscar uma venda                  ");
00051         System.out.println("  " + VER_TODOS + ". Exibir Vendas Realizadas          ");
00052         System.out.println("  " + SAIR + ". Sair                                ");
00053         System.out.println("");
00054         System.out.print("Escolha uma opção: ");
00055
00056         opcao = in.nextInt();
00057     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



7.21.3.3 imprimirVenda() void br.ufrn.loja.view.TelaVendas.imprimirVenda (**Venda** *venda*)

Imprime os detalhes de uma venda.

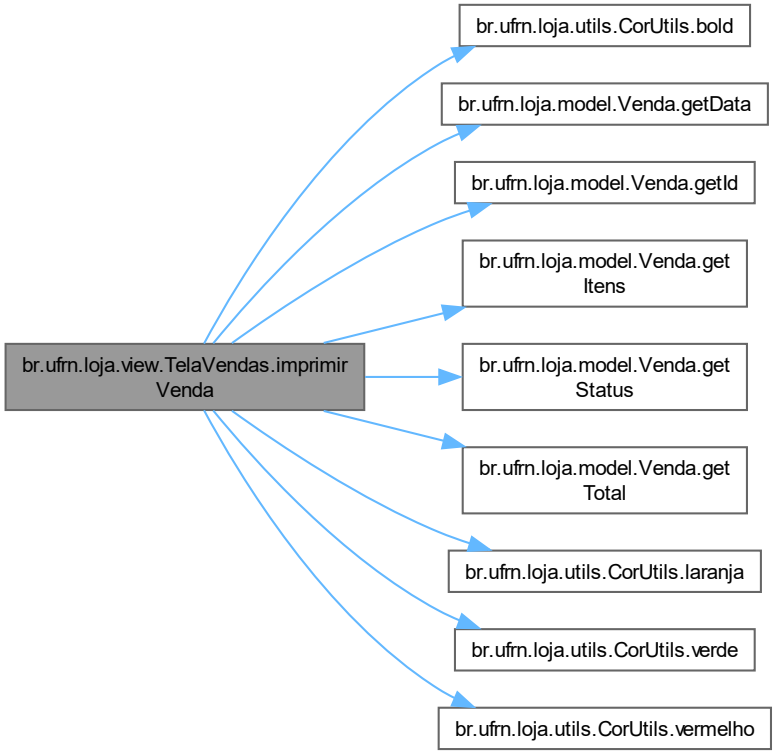
Parâmetros

venda	Venda a ser impressa.

```
00155                                       {
00156
00157         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
00158         if (venda.getId() == 0) {
00159             System.out.println("\n----- " + CorUtils.laranja("Resumo da Nova Venda") + "
      -------------------");
00160         } else {
00161             System.out.printf("ID da Venda: %d\n", venda.getId());
00162         }
00163
00164         System.out.println(CorUtils.bold("Data: ") + venda.getData().format(formatter));
```

```
00165
00166         if (venda.getStatus() == StatusVenda.CANCELADA) {
00167             System.out.println(CorUtils.bold("Status:") + CorUtils.vermelho(" " + venda.getStatus()));
00168         } else
00169             System.out.println(CorUtils.bold("Status: " + venda.getStatus()));
00170
00171         System.out.println(CorUtils.bold("Itens:"));
00172         for (ItemVenda item : venda.getItens()) {
00173             System.out.printf(CorUtils.bold("- Produto:") + " %s, Quantidade: %d, Subtotal: %.2f\n",
00174                     item.getProduto().getNome(), item.getQuantidade(), item.getSubtotal());
00175         }
00176         System.out.printf(CorUtils.verde("Total: %.2f\n"), venda.getTotal());
00177         System.out.println("-----------------------------------------\n");
00178     }
```

Este é o diagrama das funções utilizadas por essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/ **TelaVendas.java**

7.22 Referência da Classe br.ufrn.loja.model.Venda

Representa uma transação de venda realizada na loja.

Métodos Públicos

- **Venda** ()

  Construtor padrão da classe Venda.
- void **cancelarVenda** ()

  Cancela a venda, alterando seu status para CANCELADA.
- int **getId** ()

  Obtém o identificador único da venda.
- void **setId** (int id)

  Define o identificador único da venda.
- LocalDate **getData** ()

  Obtém a data da realização da venda.
- void **setData** (LocalDate data)

  Define a data da realização da venda.
- List< **ItemVenda** > **getItens** ()

  Obtém a lista de itens associados à venda.
- void **setItens** (List< **ItemVenda** > itens)

  Define a lista de itens associados à venda.
- double **calcular_total** ()

  Calcula o valor total da venda somando os subtotais de seus itens.
- double **getTotal** ()

  Obtém o valor total da venda.
- **StatusVenda** **getStatus** ()

  Obtém o status da venda.
- void **setStatus** (**StatusVenda** status)

  Define o status da venda.
- void **setTotal** (double total)

  Define o valor total da venda.
- void **addItem** (**ItemVenda** item)

  Adiciona um item à lista de itens associados à venda.
- **ItemVenda** **buscarItemPorId** (int id)

  Busca um item de venda pelo seu identificador.

7.22.1 Descrição detalhada

Representa uma transação de venda realizada na loja.

7.22.2 Construtores e Destrutores

7.22.2.1 Venda() `br.ufrn.loja.model.Venda.Venda ( )`

Construtor padrão da classe Venda.

```
00024                 {
00025         this.itens = new ArrayList<>();
00026         this.status = StatusVenda.CONCLUIDA;
00027     }
```

7.22.3 Documentação das funções

7.22.3.1 addItem() `void br.ufrn.loja.model.Venda.addItem ( ItemVenda item )`

Adiciona um item à lista de itens associados à venda.

Parâmetros

item	Item de venda a ser adicionado.

```
00118                                       {
00119         itens.add(item);
00120     }
```

7.22.3.2 buscarItemPorId() `ItemVenda br.ufrn.loja.model.Venda.buscarItemPorId ( int id )`

Busca um item de venda pelo seu identificador.

Parâmetros

id	Identificador do item de venda.

Retorna

Item de venda encontrado ou null se não existir.

```
00126                                                {
00127         for (ItemVenda item : itens) {
00128             if (item.getId() == id) {
00129                 return item;
00130             }
00131         }
00132         return null;
00133     }
```

7.22.3.3 calcular_total() `double br.ufrn.loja.model.Venda.calcular_total ( )`

Calcula o valor total da venda somando os subtotais de seus itens.

Retorna

Valor total da venda.

```
00080                                       {
00081         for (ItemVenda item : itens) {
00082             total += item.getSubtotal();
00083         }
00084         return total;
00085     }
```

7.22.3.4 cancelarVenda() `void br.ufrn.loja.model.Venda.cancelarVenda ( )`

Cancela a venda, alterando seu status para CANCELADA.

```
00031                                {
00032         this.status = StatusVenda.CANCELADA;
00033     }
```

7.22.3.5 getData() `LocalDate br.ufrn.loja.model.Venda.getData ( )`

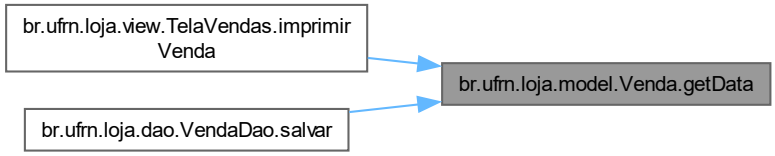
Obtém a data da realização da venda.

Retorna

Data da realização da venda.

```
00052                                {
00053         return data;
00054     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.6 getId() `int br.ufrn.loja.model.Venda.getId ( )`

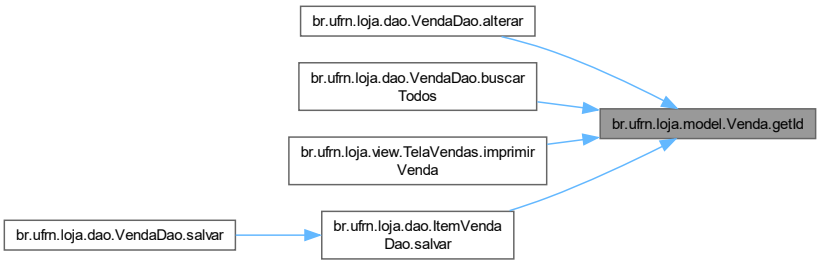
Obtém o identificador único da venda.

Retorna

Identificador único da venda.

```
00038                        {
00039         return id;
00040     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.7 getItens() List< ItemVenda > br.ufrn.loja.model.Venda.getItens ( )

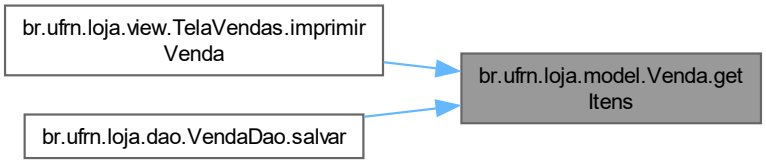
Obtém a lista de itens associados à venda.

Retorna

Lista de itens associados à venda.

```
00066                                   {
00067         return itens;
00068     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.8 getStatus() StatusVenda br.ufrn.loja.model.Venda.getStatus ( )

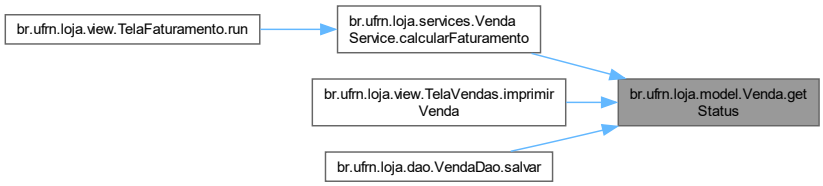
Obtém o status da venda.

Retorna

Status da venda (CONCLUÍDA ou CANCELADA).

```
00097                                   {
00098         return status;
00099     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.9 getTotal() `double br.ufrn.loja.model.Venda.getTotal ( )`

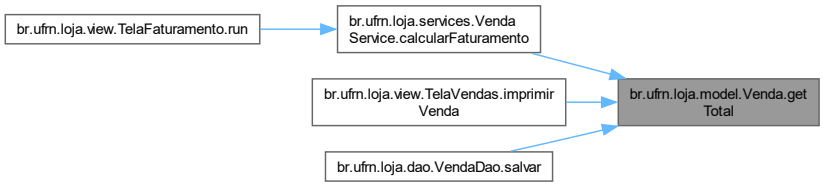
Obtém o valor total da venda.

Retorna

Valor total da venda.

```
00090                               {
00091         return total;
00092     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.10 setData() `void br.ufrn.loja.model.Venda.setData ( LocalDate data )`

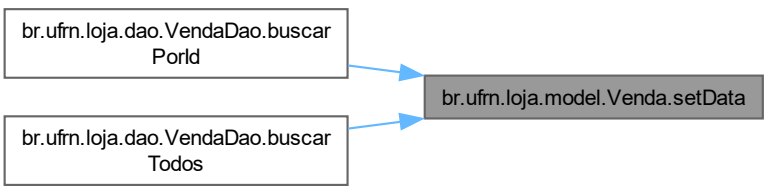
Define a data da realização da venda.

Parâmetros

data	Data da realização da venda.

```
00059                                       {
00060         this.data = data;
00061     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.11 setId() `void br.ufrn.loja.model.Venda.setId ( int id )`

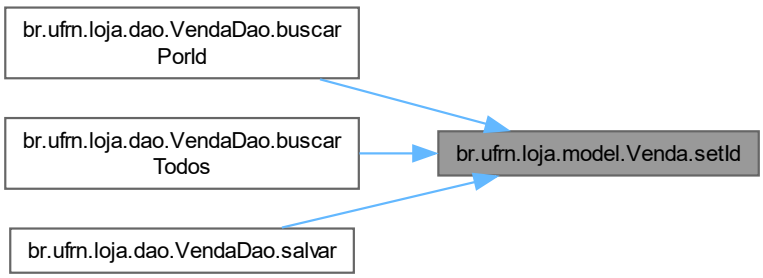
Define o identificador único da venda.

Parâmetros

id	Identificador único da venda.

```
00045                               {
00046         this.id = id;
00047     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.12 setItens() `void br.ufrn.loja.model.Venda.setItens ( List< ItemVenda > itens )`

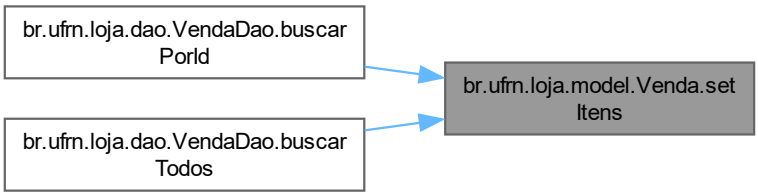
Define a lista de itens associados à venda.

Parâmetros

itens	Lista de itens associados à venda.

```
00073                                           {
00074         this.itens = itens;
00075     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.13 setStatus() `void br.ufrn.loja.model.Venda.setStatus ( StatusVenda status )`

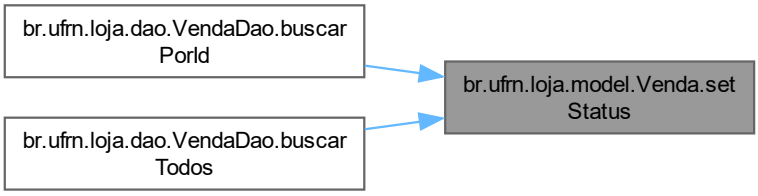
Define o status da venda.

Parâmetros

status	Status da venda (CONCLUÍDA ou CANCELADA).

```
00104                                           {
00105         this.status = status;
00106     }
```

Esse é o diagrama das funções que utilizam essa função:



7.22.3.14 setTotal() `void br.ufrn.loja.model.Venda.setTotal ( double total )`

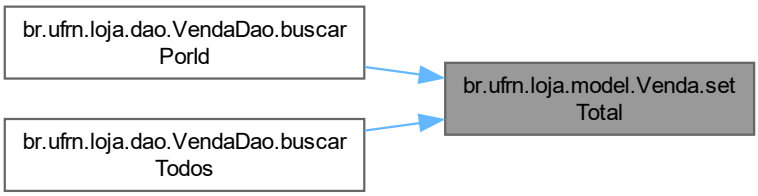
Define o valor total da venda.

Parâmetros

total	Valor total da venda.

```
00111                                       {
00112         this.total = total;
00113     }
```

Esse é o diagrama das funções que utilizam essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/ **Venda.java**

7.23 Referência da Classe br.ufrn.loja.dao.VendaDao

Classe de acesso a dados (DAO) para a entidade Venda.

Diagrama de hierarquia para br.ufrn.loja.dao.VendaDao:

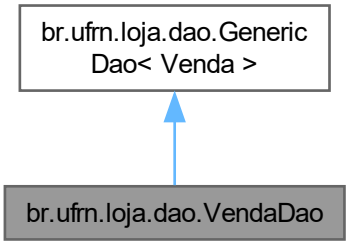
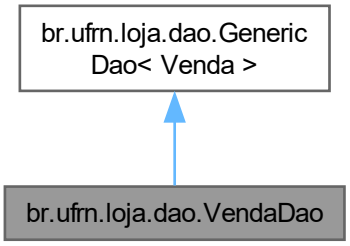


Diagrama de colaboração para br.ufrn.loja.dao.VendaDao:



Métodos Públicos

- **VendaDao** ()

  Construtor padrão que inicializa a conexão com o banco de dados.
- void **salvar** (**Venda** obj)
- List< **Venda** > **buscarTodos** ()
- **Venda** **buscarPorId** (int id)
- void **delete** (int id)
- boolean **existePorId** (int id)
- void **alterar** (**Venda** obj)
- int **buscarUltimoId** ()

  Busca o último ID inserido no banco de dados.

Métodos Públicos herdados de br.ufrn.loja.dao.GenericDao< Venda >

- void **salvar** (E obj)
- List< E > **buscarTodos** ()
- E **buscarPorId** (int id)
- void **delete** (int id)
- boolean **existePorId** (int id)
- void **alterar** (E obj)

7.23.1 Descrição detalhada

Classe de acesso a dados (DAO) para a entidade Venda.

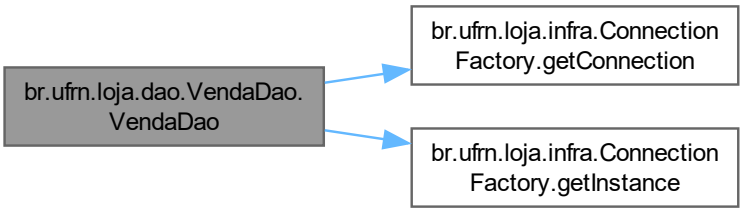
7.23.2 Construtores e Destrutores

7.23.2.1 VendaDao() `br.ufrn.loja.dao.VendaDao.VendaDao ( )`

Construtor padrão que inicializa a conexão com o banco de dados.

```
00037                     {
00038         con = ConnectionFactory.getInstance().getConnection();
00039     }
```

Este é o diagrama das funções utilizadas por essa função:



7.23.3 Documentação das funções

7.23.3.1 alterar() `void br.ufrn.loja.dao.VendaDao.alterar ( Venda obj )`

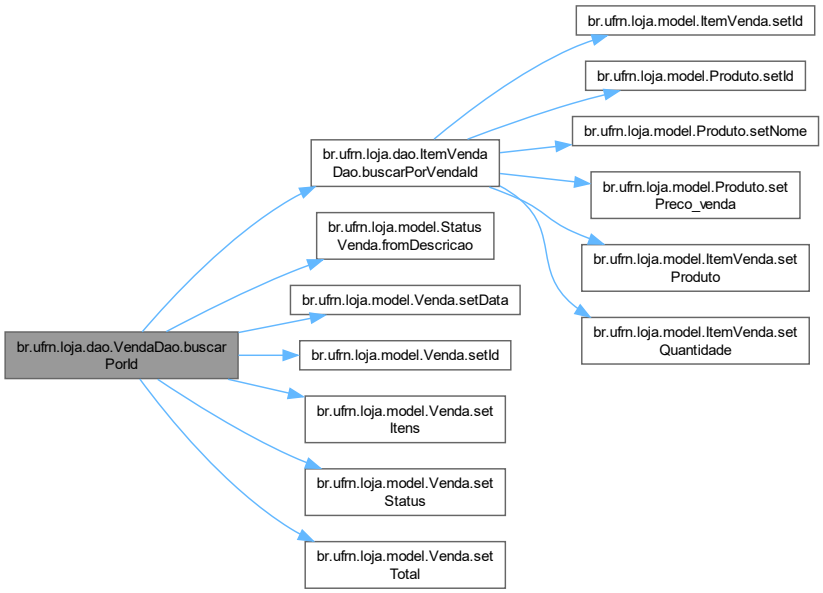
```
00143                                   {
00144         try {
00145             statement = con.createStatement();
00146             statement.executeUpdate("UPDATE venda SET status = 'CANCELADA' WHERE id = " +
      obj.getId());
00147         } catch (SQLException e) {
00148             e.printStackTrace();
00149         } finally {
00150             fecharRecursos();
00151         }
00152     }
```

Este é o diagrama das funções utilizadas por essa função:

7.23.3.2 buscarPorId() `Venda br.ufrn.loja.dao.VendaDao.buscarPorId ( int id )`

```
00093                                   {
00094         try {
00095             statement = con.createStatement();
00096             ResultSet result = statement.executeQuery(SELECT_ALL + " where id = " + id);
00097             if (result.next()) {
00098                 Venda venda = new Venda();
00099                 venda.setId(result.getInt("id"));
00100                 venda.setData(LocalDate.parse(result.getString("data")));
00101                 venda.setStatus(StatusVenda.fromDescricao(result.getString("status")));
00102                 venda.setTotal(result.getDouble("total_venda"));
00103                 List<ItemVenda> itens = new ItemVendaDao().buscarPorVendaId(id);
00104                 venda.setItens(itens);
00105 
00106                 return venda;
00107             }
00108         } catch (SQLException e) {
00109             e.printStackTrace();
00110         } finally {
00111             fecharRecursos();
00112         }
00113         return null;
00114     }
```

Este é o diagrama das funções utilizadas por essa função:

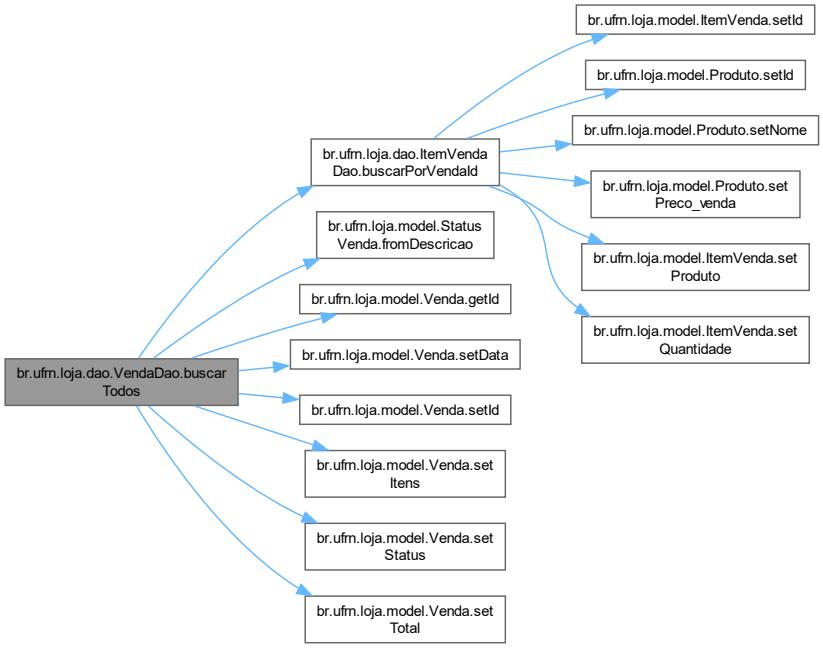


7.23.3.3 buscarTodos() `List< Venda > br.ufrn.loja.dao.VendaDao.buscarTodos ( )`

```
00067                                   {
00068         List<Venda> resultado = new ArrayList<>();
00069         try {
00070             statement = con.createStatement();
00071             ResultSet rs = statement.executeQuery(SELECT_ALL);
00072             while (rs.next()) {
00073                 Venda venda = new Venda();
00074                 venda.setId(rs.getInt("id"));
00075                 venda.setData(LocalDate.parse(rs.getString("data")));
00076                 venda.setStatus(StatusVenda.fromDescricao(rs.getString("status")));
00077                 venda.setTotal(rs.getDouble("total_venda"));
00078                 // recupere os itens da venda do banco de dados
00079                 List<ItemVenda> itens = new ItemVendaDao().buscarPorVendaId(venda.getId());
00080                 venda.setItens(itens);
00081 
```

```
00082                 resultado.add(venda);
00083             }
00084         } catch (SQLException e) {
00085             e.printStackTrace();
00086         } finally {
00087             fecharRecursos();
00088         }
00089         return resultado;
00090     }
```

Este é o diagrama das funções utilizadas por essa função:



7.23.3.4 buscarUltimoId() `int br.ufrn.loja.dao.VendaDao.buscarUltimoId ( )`

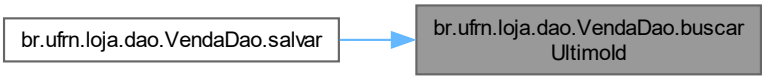
Busca o último ID inserido no banco de dados.

Retorna

Último ID inserido.

```
00158                                 {
00159         try {
00160             statement = con.createStatement();
00161             ResultSet result = statement.executeQuery("SELECT last_insert_rowid() AS novo_id;");
00162             return result.getInt("novo_id");
00163         } catch (SQLException e) {
00164             e.printStackTrace();
00165         } finally {
00166             fecharRecursos();
00167         }
00168         return 0;
00169     }
```

Esse é o diagrama das funções que utilizam essa função:



7.23.3.5 delete() void br.ufrn.loja.dao.VendaDao.delete (int *id*)

```
00117                                {
00118         try {
00119             statement = con.createStatement();
00120             statement.executeUpdate(DELETE + id);
00121         } catch (SQLException e) {
00122             e.printStackTrace();
00123         } finally {
00124             fecharRecursos();
00125         }
00126     }
```

7.23.3.6 existePorId() boolean br.ufrn.loja.dao.VendaDao.existePorId (int *id*)

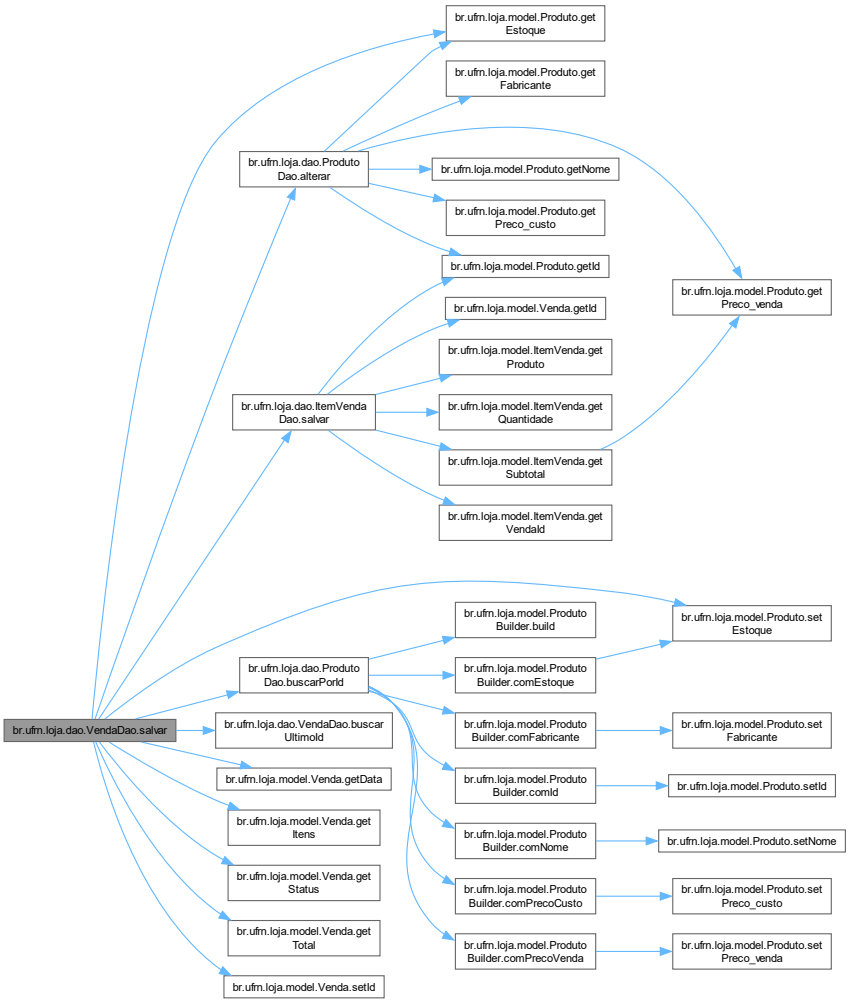
```
00129                                      {
00130         try {
00131             statement = con.createStatement();
00132             ResultSet result = statement.executeQuery(EXISTE + id);
00133             return result.next() && result.getInt(1) > 0;
00134         } catch (SQLException e) {
00135             e.printStackTrace();
00136         } finally {
00137             fecharRecursos();
00138         }
00139         return false;
00140     }
```

7.23.3.7 salvar() void br.ufrn.loja.dao.VendaDao.salvar (**Venda** *obj*)

```
00042                                  {
00043         try {
00044             statement = con.createStatement();
00045             String sql = INSERT.replaceFirst("%", obj.getData().toString())
00046                     .replaceFirst("%", obj.getStatus().toString()).replaceFirst("%",
      Double.toString(obj.getTotal()));
00047             statement.executeUpdate(sql);
00048             obj.setId(buscarUltimoId());
00049             for (ItemVenda item : obj.getItens()) {
00050                 item.setVendaId(obj);
00051                 new ItemVendaDao().salvar(item);
00052 
00053                 Produto produto = new ProdutoDao().buscarPorId(item.getProduto().getId());
00054                 produto.setEstoque(produto.getEstoque() - item.getQuantidade());
00055                 if (produto.getEstoque() >= 0) {
00056                     new ProdutoDao().alterar(produto);
00057                 }
00058             }
00059         } catch (Exception e) {
00060             e.printStackTrace();
```

```
00061         } finally {
00062             fecharRecursos();
00063         }
00064     }
```

Este é o diagrama das funções utilizadas por essa função:



A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/dao/ **VendaDao.java**

7.24 Referência da Classe br.ufrn.loja.services.VendaService

Fornece serviços relacionados à entidade Venda.

Diagrama de hierarquia para br.ufrn.loja.services.VendaService:

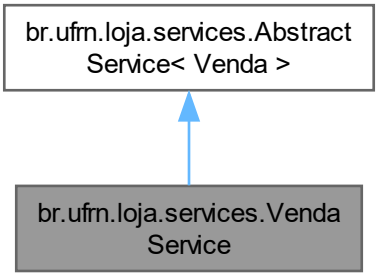
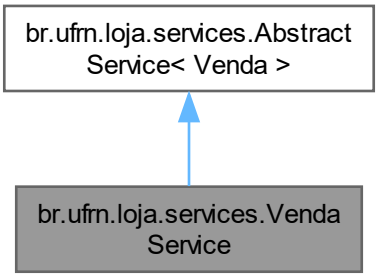


Diagrama de colaboração para br.ufrn.loja.services.VendaService:



Métodos Públicos

- **VendaService** ()

  Construtor padrão da classe VendaService.
- boolean **buscar** ()

  Busca e imprime os detalhes de uma venda pelo seu ID.
- List< **Venda** > **obterTodasAsVendas** (String inicio, String fim)

  Obtém todas as vendas no intervalo de datas especificado.
- **Venda** **buscarVendaPorId** (int idVenda)

  Busca uma venda pelo seu ID.
- double **calcularFaturamento** (String inicio, String fim)

Métodos Públicos herdados de br.ufrn.loja.services.AbstractService< Venda >

- void **processar** (int opcao)

  Processa a operação solicitada.
- abstract boolean **buscar** ()
- E **getObjeto** ()
- void **setObjeto** (E objeto)

Membros Protegidos

- boolean **validar** ()

  Valida se a venda possui itens adicionados.
- void **imprimir** ()

  Imprime os detalhes de todas as vendas cadastradas.
- void **remover** ()

  Remove a venda do sistema.

Membros Protegidos herdados de br.ufrn.loja.services.AbstractService< Venda >

- void **cadastrar** ()

  Método que prepara para o cadastro de um objeto.
- abstract boolean **validar** ()
- void **alterar** ()

  Método abstrato para alterar o objeto.
- abstract void **remover** ()

  Método abstrato para remover o objeto.
- abstract void **imprimir** ()

Outros membros herdados

Atributos Protegidos herdados de br.ufrn.loja.services.AbstractService< Venda >

- E **objeto**
- **GenericDao**< E > **dao**

7.24.1 Descrição detalhada

Fornece serviços relacionados à entidade Venda.

7.24.2 Construtores e Destrutores

7.24.2.1 VendaService() `br.ufrn.loja.services.VendaService.VendaService ( )`

Construtor padrão da classe VendaService.

```
00025                           {
00026         dao = new VendaDao();
00027     }
```

7.24.3 Documentação das funções

7.24.3.1 buscar() boolean br.ufrn.loja.services.VendaService.buscar ( )

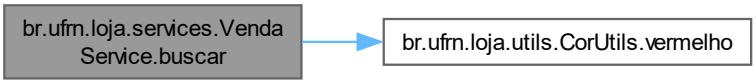
Busca e imprime os detalhes de uma venda pelo seu ID.

Retorna

True se a venda for encontrada, False caso contrário.

```
00069                           {
00070         Venda vendaEncontrada = dao.buscarPorId(objeto.getId());
00071
00072         if (vendaEncontrada != null) {
00073             System.out.println("\n----- Detalhes da Venda ----------------------");
00074             System.out.printf("%-5s%-20s%s\n", "ID", "Data", "Itens");
00075             exibirDetalhesVenda(vendaEncontrada);
00076             System.out.println("----------------------------------------------------");
00077             return true;
00078         } else {
00079             System.out.println(CorUtils.vermelho("Essa venda não existe!"));
00080             return false;
00081         }
00082     }
```

Este é o diagrama das funções utilizadas por essa função:



7.24.3.2 buscarVendaPorId() Venda br.ufrn.loja.services.VendaService.buscarVendaPorId (int *idVenda*)

Busca uma venda pelo seu ID.

Parâmetros

idVenda	ID da venda a ser buscada.

Retorna

Objeto Venda correspondente ao ID ou null se não encontrada.

```
00107                                       {
00108         return dao.buscarPorId(idVenda);
00109     }
```

7.24.3.3 calcularFaturamento() double br.ufrn.loja.services.VendaService.calcularFaturamento (String *inicio*, String *fim*)

Calcula o faturamento total das vendas no período especificado, excluindo vendas canceladas.

Parâmetros

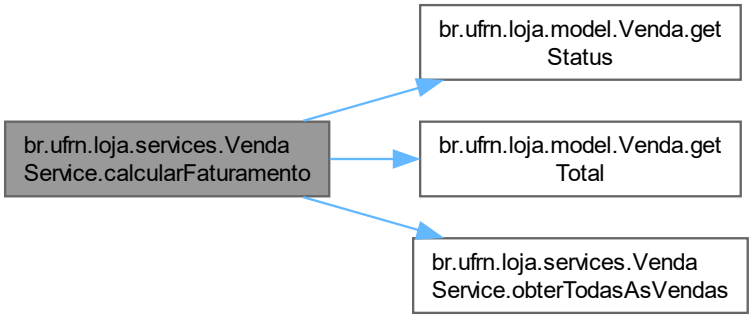
inicio	Data de início do período.
fim	Data de término do período.

Retorna

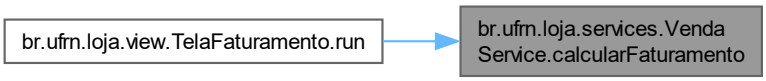
O faturamento total no período, excluindo vendas canceladas.

```
00131                                                                   {
00132         return obterTodasAsVendas(inicio, fim).stream()
00133                 .filter(venda -> venda.getStatus() != StatusVenda.CANCELADA)
00134                 .mapToDouble(Venda::getTotal)
00135                 .sum();
00136     }
```

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



7.24.3.4 imprimir() `void br.ufrn.loja.services.VendaService.imprimir ( ) [protected]`

Imprime os detalhes de todas as vendas cadastradas.

```
00044                                 {
00045         System.out.println("\n----- Detalhes da Venda ----------------------");
00046         System.out.printf("%-5s%-20s%s\n", "ID", "Data", "Itens");
00047         dao.buscarTodos().forEach(venda -> {
00048             System.out.printf("%-5s%-20s%s\n", venda.getId(), venda.getData(), venda.getItens());
00049         });
00050         System.out.println("-----------------------------------------------");
00051     }
```

7.24.3.5 obterTodasAsVendas() List< Venda > br.ufrn.loja.services.VendaService.obterTodas↩ AsVendas (String *inicio*, String *fim*)

Obtém todas as vendas no intervalo de datas especificado.

Parâmetros

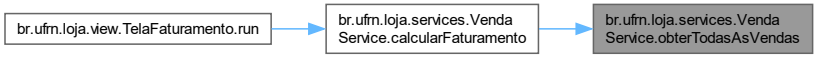
inicio	Data de início do intervalo no formato "yyyy-MM-dd".
fim	Data de fim do intervalo no formato "yyyy-MM-dd".

Retorna

Lista de vendas no intervalo de datas especificado.

```
00090                                                                    {
00091         List<Venda> listaDeVendas = dao.buscarTodos();
00092         List<Venda> retorno = new ArrayList<>();
00093         for(Venda venda : listaDeVendas) {
00094             if(venda.getData().isAfter(LocalDate.parse(inicio)) ||
      venda.getData().isEqual(LocalDate.parse(inicio))) {
00095                 if(venda.getData().isBefore(LocalDate.parse(fim)) ||
      venda.getData().isEqual(LocalDate.parse(fim))) {
00096                     retorno.add(venda);
00097                 }
00098             }
00099         }
00100         return retorno;
00101     }
```

Esse é o diagrama das funções que utilizam essa função:

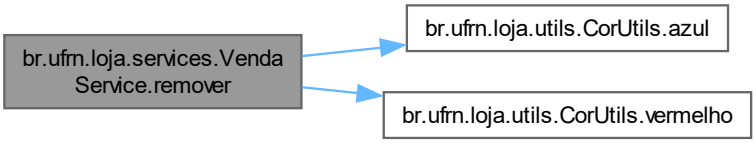


7.24.3.6 remover() void br.ufrn.loja.services.VendaService.remover ( ) [protected]

Remove a venda do sistema.

```
00056                                {
00057         if (dao.existePorId(objeto.getId())) {
00058             dao.delete(objeto.getId());
00059             System.out.println(CorUtils.azul("Venda removida com sucesso!!"));
00060         } else {
00061             System.out.println(CorUtils.vermelho("Essa venda não existe!"));
00062         }
00063     }
```

Este é o diagrama das funções utilizadas por essa função:



7.24.3.7 validar() `boolean br.ufrn.loja.services.VendaService.validar ( ) [protected]`

Valida se a venda possui itens adicionados.

Retorna

True se a venda for válida, False caso contrário.

```
00033                                {
00034         if(objeto.getItens().isEmpty()){
00035             System.out.println("Voce precisa adicionar produtos!");
00036             return false;
00037         }
00038         return true;
00039     }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/services/ **VendaService.java**

8 Arquivos

8.1 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/README.md

8.2 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/dao/GenericDao.java

Componentes

- interface **br.ufrn.loja.dao.GenericDao< E >**

Pacotes

- package **br.ufrn.loja.dao**

8.3 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/dao/ItemVendaDao.java

Implementação da interface GenericDao para a entidade ItemVenda.

Componentes

- class **br.ufrn.loja.dao.ItemVendaDao**

  Classe de acesso a dados (DAO) para a entidade ItemVenda.

Pacotes

- package **br.ufrn.loja.dao**

8.3.1 Descrição detalhada

Implementação da interface GenericDao para a entidade ItemVenda.

8.4 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/dao/ProdutoDao.java

Componentes

- class **br.ufrn.loja.dao.ProdutoDao**
  Implementação do DAO para a entidade Produto.

Pacotes

- package **br.ufrn.loja.dao**

8.5 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/dao/VendaDao.java

Implementação da interface GenericDao para a entidade Venda.

Componentes

- class **br.ufrn.loja.dao.VendaDao**
  Classe de acesso a dados (DAO) para a entidade Venda.

Pacotes

- package **br.ufrn.loja.dao**

8.5.1 Descrição detalhada

Implementação da interface GenericDao para a entidade Venda.

8.6 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/exception/CriacaoConexaoException.java

Componentes

- class **br.ufrn.loja.exception.CriacaoConexaoException**

Pacotes

- package **br.ufrn.loja.exception**

8.7 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/exception/OpcaoInvalidaException.java

Componentes

- class **br.ufrn.loja.exception.OpcaoInvalidaException**

Pacotes

- package **br.ufrn.loja.exception**

8.8 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/infra/ConnectionFactory.java

Componentes

- class **br.ufrn.loja.infra.ConnectionFactory**

  Classe responsável por fornecer instâncias de conexão com o banco de dados SQLite.

Pacotes

- package **br.ufrn.loja.infra**

8.9 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/LojaApplication.java

Componentes

- class **br.ufrn.loja.LojaApplication**

  Classe com o metodo Main.

Pacotes

- package **br.ufrn.loja**

8.10 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/ItemVenda.java

Definição da classe ItemVenda que representa um item de venda.

Componentes

- class **br.ufrn.loja.model.ItemVenda**

  Representa um item de venda associado a um produto e uma quantidade.

Pacotes

- package **br.ufrn.loja.model**

8.10.1 Descrição detalhada

Definição da classe ItemVenda que representa um item de venda.

8.11 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/Produto.java

Componentes

- class **br.ufrn.loja.model.Produto**

Pacotes

- package **br.ufrn.loja.model**

8.12 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/ProdutoBuilder.java

Componentes

- class **br.ufrn.loja.model.ProdutoBuilder**

Pacotes

- package **br.ufrn.loja.model**

8.13 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/StatusVenda.java

Componentes

- enum **br.ufrn.loja.model.StatusVenda**

Pacotes

- package **br.ufrn.loja.model**

8.14 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/model/Venda.java

Definição da classe Venda que representa uma transação de venda.

Componentes

- class **br.ufrn.loja.model.Venda**

  Representa uma transação de venda realizada na loja.

Pacotes

- package **br.ufrn.loja.model**

8.14.1 Descrição detalhada

Definição da classe Venda que representa uma transação de venda.

8.15 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/services/AbstractService.java

Componentes

- class **br.ufrn.loja.services.AbstractService**< **E** >

  Classe abstrata que define operações básicas de movimentação (CRUD).

Pacotes

- package **br.ufrn.loja.services**

8.16 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/services/ProdutoService.java

Componentes

- class **br.ufrn.loja.services.ProdutoService**

Pacotes

- package **br.ufrn.loja.services**

8.17 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/services/VendaService.java

Definição da classe VendaService que fornece serviços relacionados à entidade Venda.

Componentes

- class **br.ufrn.loja.services.VendaService**

  Fornece serviços relacionados à entidade Venda.

Pacotes

- package **br.ufrn.loja.services**

8.17.1 Descrição detalhada

Definição da classe VendaService que fornece serviços relacionados à entidade Venda.

8.18 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/utils/ArquivoUtils.java

Componentes

- class **br.ufrn.loja.utils.ArquivoUtils**

  Classe utilitária para operações relacionadas a arquivos.

Pacotes

- package **br.ufrn.loja.utils**

8.19 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/utils/CorUtils.java

Componentes

- class **br.ufrn.loja.utils.CorUtils**

Pacotes

- package **br.ufrn.loja.utils**

8.20 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/Menu.java

Componentes

- class **br.ufrn.loja.view.Menu**

Pacotes

- package **br.ufrn.loja.view**

8.21 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/MenuAbstract.java

Componentes

- class **br.ufrn.loja.view.MenuAbstract**

Pacotes

- package **br.ufrn.loja.view**

8.22 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/TelaBusca.java

Componentes

- class **br.ufrn.loja.view.TelaBusca**

Pacotes

- package **br.ufrn.loja.view**

8.23 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/TelaCadastro.java

Componentes

- class **br.ufrn.loja.view.TelaCadastro**

Pacotes

- package **br.ufrn.loja.view**

8.24 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/TelaFaturamento.java

Componentes

- class **br.ufrn.loja.view.TelaFaturamento**

Pacotes

- package **br.ufrn.loja.view**

8.25 Referência do Arquivo C:/Projetos/JAVA/trabalho-final-rayana-mendes/src/main/java/br/ufrn/loja/view/TelaVendas.java

Definição da classe TelaVendas que representa a tela de vendas e suas funcionalidades.

Componentes

- class **br.ufrn.loja.view.TelaVendas**

  Classe que representa a tela de vendas e suas funcionalidades.

Pacotes

- package **br.ufrn.loja.view**

8.25.1 Descrição detalhada

Definição da classe TelaVendas que representa a tela de vendas e suas funcionalidades.

Índice Remissivo