**01 Search**

**Manual Técnico**

Sumário

[Modelo de Dados 2](#_Toc341381866)

[Dicionário de Dados 3](#_Toc341381867)

[Intenção: Ao ter um item criado podemos: 5](#_Toc341381868)

[Modelo de Classes 8](#_Toc341381869)

[Módulo “Servidor” 11](#_Toc341381870)

[Módulo “Conexao” 15](#_Toc341381871)

[Módulo “Novidades” 18](#_Toc341381872)

[Módulo “Categoria” 19](#_Toc341381873)

[Módulo “Busca” 20](#_Toc341381874)

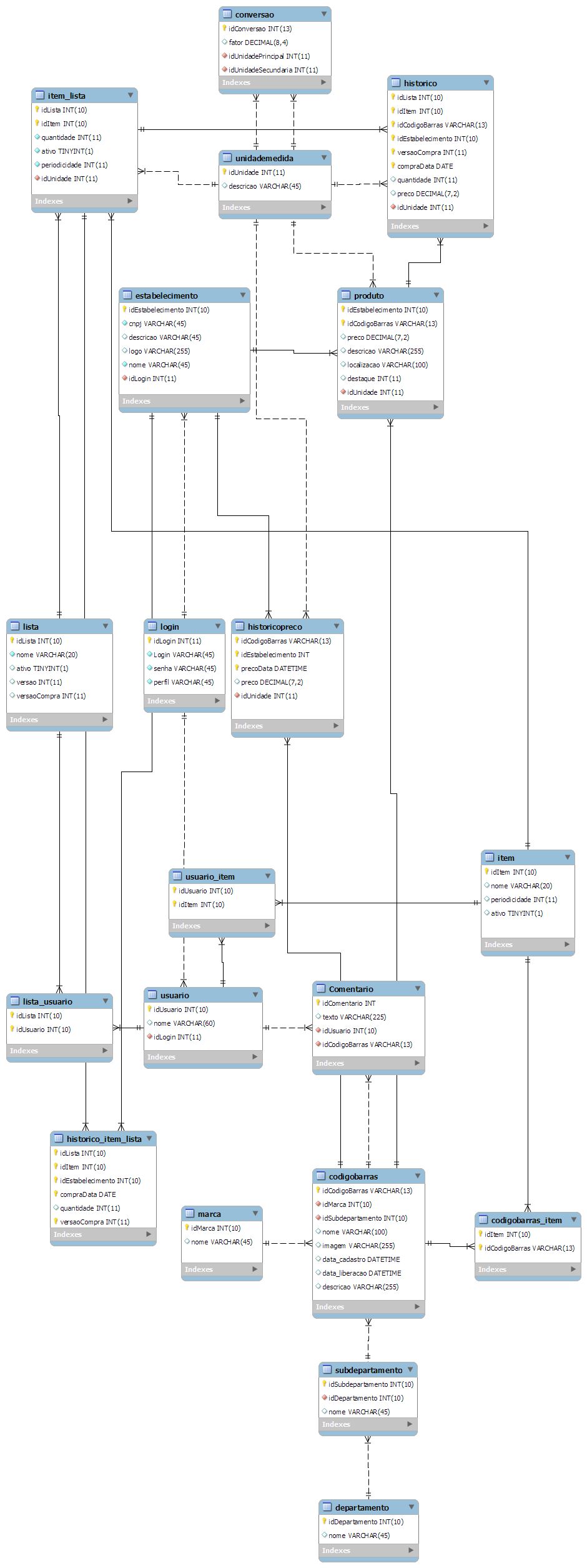
[Módulo “Listas” 21](#_Toc341381875)

[Módulo “Produto” 24](#_Toc341381876)

[Módulo “Cache” 25](#_Toc341381877)

Modelo de Dados

Para melhor visualização, recomenda-se abrir o arquivo entregue no CD.

Dicionário de Dados

**Login:** tabela de usuários do sistema.

* idLogin - PK da tabela.
* Login - Usuário para acessar o sistema, é do tipo único no banco de dados .
* Senha - Senha para acessar o sistema.
* Perfil - Código de perfil de acesso ao sistema.

**Usuario:** tabela de usuários que criarão listas no programa.

* *idUsuario* - é a chave primária da tabela.
* *Nome* – nome do usuário.
* idLogin - FK, Tabela Login, indica login do usuário.

**Lista:** tabela de listas criadas pelo usuário.

* *idLista* - é a chave primária da tabela.
* *Nome* – é o nome da lista.
* *Ativo* - Tipo Boolean, flag que indica se a lista foi excluída pelo usuário ou não.
* *Versao* - Número da versão da lista.
* *versao \_compra* - Número da versão da ultima compra.

**Lista\_Usuario:** Tabela utilizada para relacionar listas com usuários, indicando quais usuários possui acesso a determinadas listas.

* *idLista* - FK, tabela Lista.
* *idUsuario* - FK, Tabela Usuario.

**Item:** tabela de itens que um usuário cria.

* *idItem* - é a chave primária da tabela.
* *Nome* – é o nome do item.
* *Peridicidade* – é a quantidade em dias que o item dura.
* *Ativo* - Tipo Boolean, flag que indica se o item foi excluído pelo usuário ou não.

**Item\_Lista:** tabela de listas com seus respectivos itens adicionados e suas quantidades.

* *idLista* - é a chave primária da tabela.
* *IdItem* - é a chave primária da tabela.
* *Quantidade*: Quantidade que se deseja comprar do item na lista.
* *Ativo* - Tipo Bool, flag que indica se o item precisa ser comprado (V) ou se o item já foi comprado(F), é desativado quando toda a quantidade for comprada, ou a critério do usuário, e ativado em caso de periodicidade quando a validade “vence”.
* *Peridicidade* – é a quantidade em dias que o item nesta lista dura.
* *idUnidade* – é chave estrangeira da tabela UnidadeMedida.

**Departamento**: a tabela que irá armazenar os departamentos dos produtos; é independente de cada estabelecimento.

* *idDepartamento* – é a chave primária da tabela, um número.
* *nome* – é o nome do departamento.

**Subdepartamento**: a tabela que irá armazenar os subdepartamentos dos produtos; é independente de cada estabelecimento.

* *idSubdepartamento* – é a chave primária da tabela, um número.
* *idDepartamento* – é uma chave estrangeira que referencia ao Departamento, onde um departamento pode ter vários subdepartamentos, mas um subdepartamento está ligado somente a um departamento.
* *nome* – é o nome do subdepartamento.

**CodigoBarras**: é a tabela que armazena os dados invariáveis de cada produto. Ela será o registro padrão dos produtos, será preenchida pelo primeiro estabelecimento que registrar o produto e passará pela aprovação de um administrador antes de ser liberada para o usuário.

* *idCodigoBarras* – é a chave primária da tabela, o código de barras em si. É necessário que seja em texto pois o tipo int do Java só aceita até 8 dígitos, sendo que um código de barras possui 11.
* *idMarca* – uma chave estrangeira que irá referenciar a tabela Marca.
* *idSubdepartamento* – é uma chave estrangeira que irá referenciar a tabela Subdepartamento. Um produto só pode estar referenciado a um único subdepartamento.
* *nome* – o nome do produto, onde este nome é padrão/igual para todos os estabelecimentos.
* *imagem* - o caminho/url da imagem, onde esta imagem é padrão/igual para todos os estabelecimentos.
* *data\_cadastro* – é a data em que um estabelecimento cadastra um novo item ao banco de dados.
* *data\_liberacao* – é a data em que um administrador do sistema libera este produto para o público em geral.
* *descricao* – é a descrição padrão daquele produto, onde independentemente do estabelecimento, esta descrição é única.

**Marca**: é a tabela que armazena os dados das Marcas de cada produto.

* *idMarca* – é a chave primária da tabela, um número.
* *nome* – é o nome da marca.

**Estabelecimento**: tabela que armazena os dados dos estabelecimentos cadastrados em nosso sistema.

* *idEstabelecimento* - é a chave primária da tabela, um número.
* *CNPJ* – Número do CNPJ da empresa, registro único na tabela.
* *Descricao* – Descrição do estabelecimento.
* *Logo* – endereço da imagem do estabelecimento.
* *nome* – é o nome do estabelecimento.
* *idLogin* – FK, tabela Login, login do estabelecimento.

**Produto**: tabela que armazena os dados dos produtos de cada estabelecimento.

* *IdCodigoBarras* - é a chave primária da tabela.
* *IdEstabelecimento* – é a chave primaria da tabela.
* *Preço* – preço do produto.
* *Descricao* – específica do supermercado, podendo mostrar detalhes além do padronizado.
* *Localizacao* – localização do produto.
* *Destaque* – Nível de destaque do produto.
* *idUnidade* – é chave estrangeira da tabela UnidadeMedida.

**Item\_CodigoBarras:**  Relacionar a tabela ITEM com a tabela CODIGOBARRAS de muitos pra muitos.

# Intenção: Ao ter um item criado podemos:

1 - Referenciá-lo a apenas um CODIGOBARRAS. Por exemplo, um ITEM coca-cola.

2 - Referenciá-lo a varios CODIGOBARRAS. Por exemplo: refrigerante, podendo ter vários CODIGOBARRAS diferentes (fanta, coca-cola, sprit).

3 – Sem referência. Um ITEM não referenciado a um ou a vários CODIGOBARRAS. Por exemplo, detergente (caso o usuário não querer atrelar CODIGOBARRAS a esse ITEM).

* *IdItem* - é a chave primária da tabela;
* *IdCodigoBarras* - é a chave primária da tabela;

**Histórico:** Tabela de histórico dos itens comprados de uma lista.

* *idLista* – FK, tabela Lista, indica a qual lista o item pertencia;
* *idItem* – FK, tabela Item, indica qual é o item comprado;
* *idCodigoBarras* – FK, tabela CodigoBarras, código de barras do produto;
* *idEstabelecimento* – FK, tabela Estabelecimento, de qual estabelecimento veio o item;
* *versao\_compra* – flag que marca a versão da ultima compra;
* compraData – Data da compra, faz parte da PK;
* *Quantidade* – Quantidade comprada do item;
* *Preço* – Preço do produto;
* *idUnidade* – é chave estrangeira da tabela UnidadeMedida;

**HistóricoPreco:** Tabela de histórico do preço , referente ao preço de cada produto ao longo do tempo, é acionada por um trigger.

* *idCodigoBarras* – FK, tabela CodigoBarras, código de barras do produto;
* *idEstabelecimento* – FK, tabela Estabelecimento, de qual estabelecimento veio o produto;
* *precoData*– a data deste preço;
* *Preço* – Preço do produto;
* *idUnidade* – é chave estrangeira da tabela UnidadeMedida;

**Histórico\_Item\_Lista:** Tabela de histórico dos itens de uma lista, que não possuem um codigobarras agregado.

* *idLista* – FK, tabela Lista, indica a qual lista o item pertencia.
* *idItem* – FK, tabela Item, indica qual é o item comprado.
* *idEstabelecimento* – FK, tabela Estabelecimento, de qual estabelecimento veio o produto.
* compraData – Data da compra, faz parte da PK.
* *Quantidade* – Quantidade comprada do item.
* *versao \_compra* - PK, Número da versão da ultima compra.

**Unidademedida:** Tabela de unidades dos itens.

* *idUnidade* – PK, tabela Unidademedida.
* *descricao* –descrição sobre a unidade.

**Conversão:** Tabela de conversão de unidades.

* *idConversao* – PK, tabela conversao;
* *fator* – valor de quantas vezes a primeira unidade é em relação a segunda;
* *idUnidadePrincipal* – chave estrangeira da tabela Unidademedida, representa a primeira unidade na conversão;
* *idUnidadeSecundaria* – chave estrangeira da tabela Unidademedida, representa a segunda unidade na conversão;

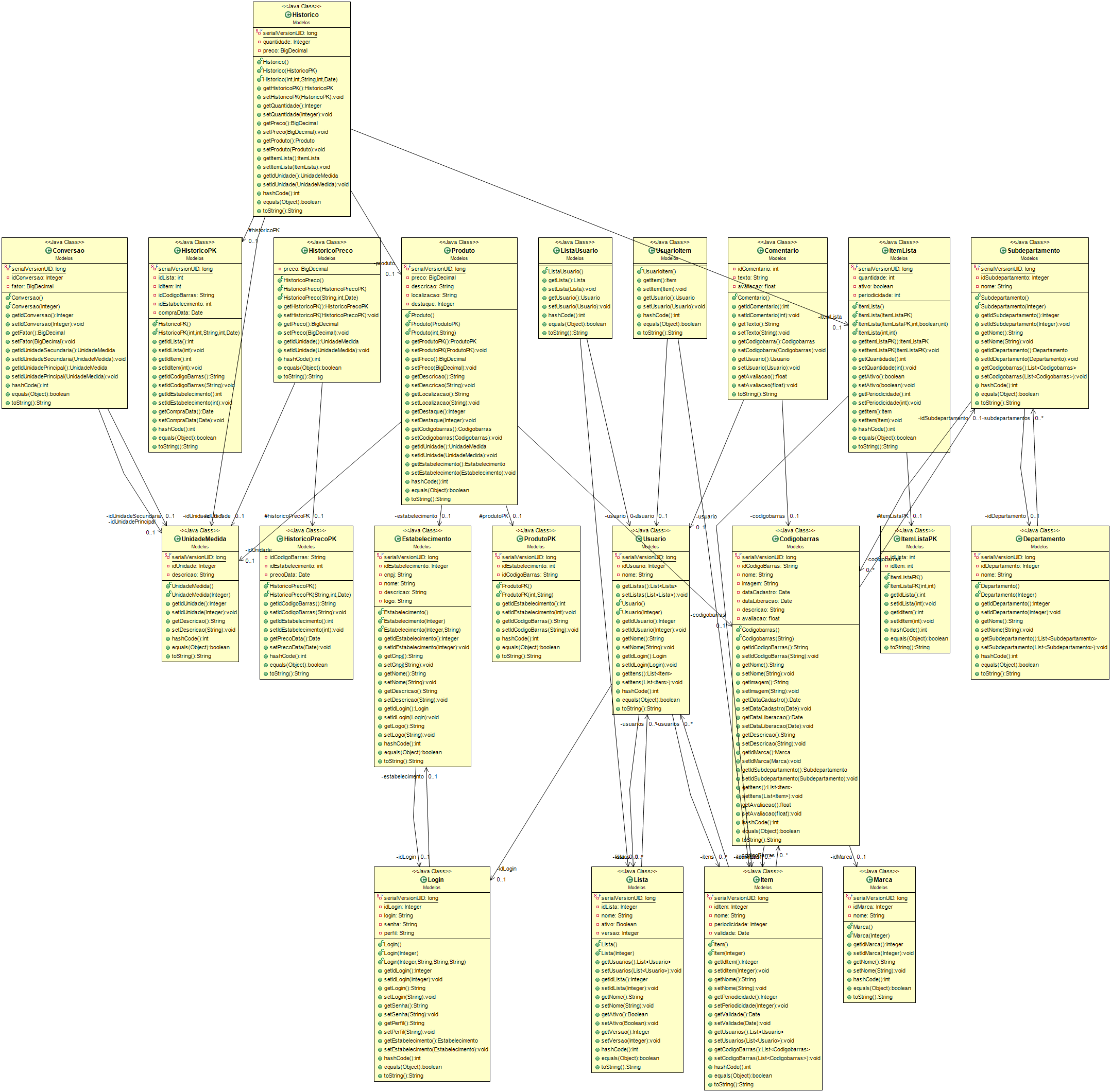
**Comentario:** Tabela referente aos comentários de cada usuário.

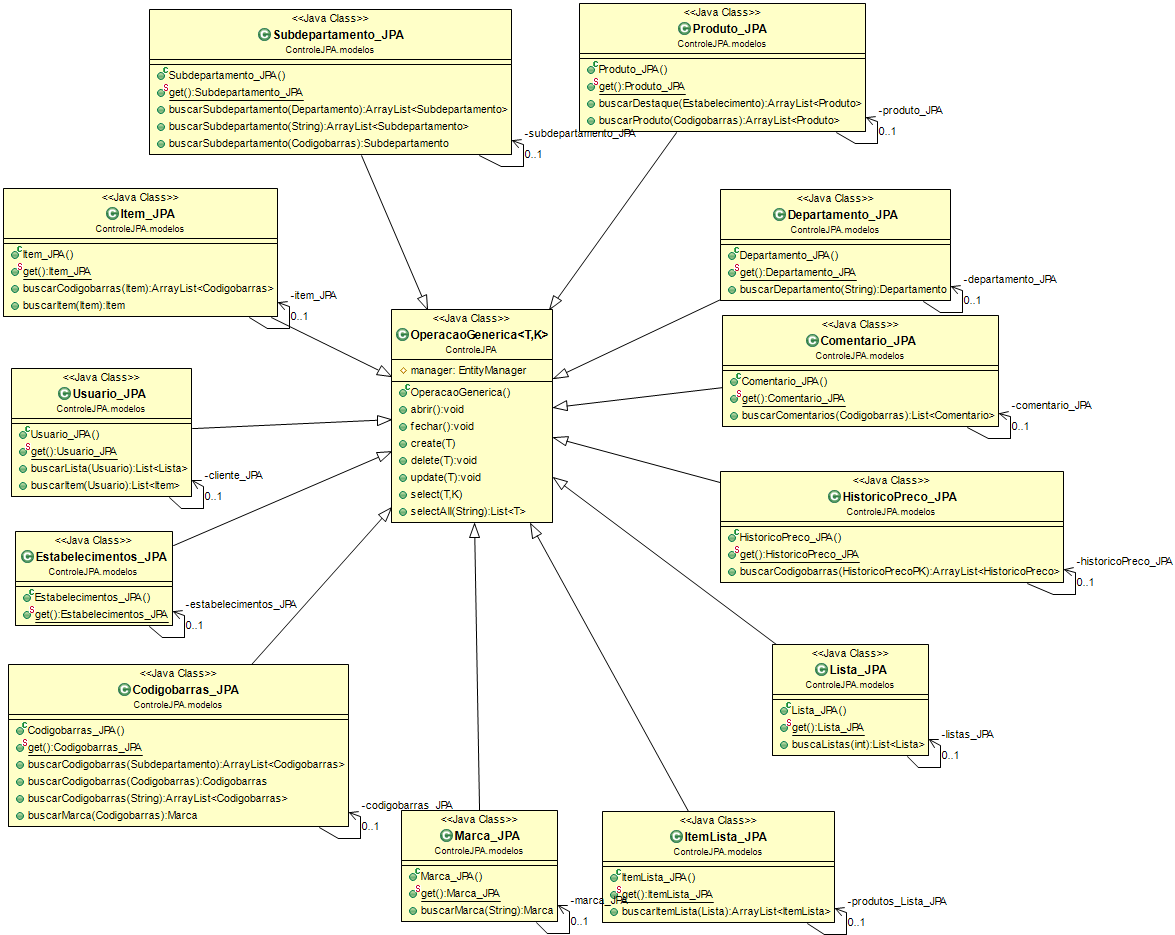
* *idComentario* – FK, tabela da tabela comentario;
* *texto* – comentário que o usuário escreveu;
* *idUsuario* – chave estrangeira da tabela usuario;
* *idCodigobarras* – chave estrangeira da tabela codigobarras;

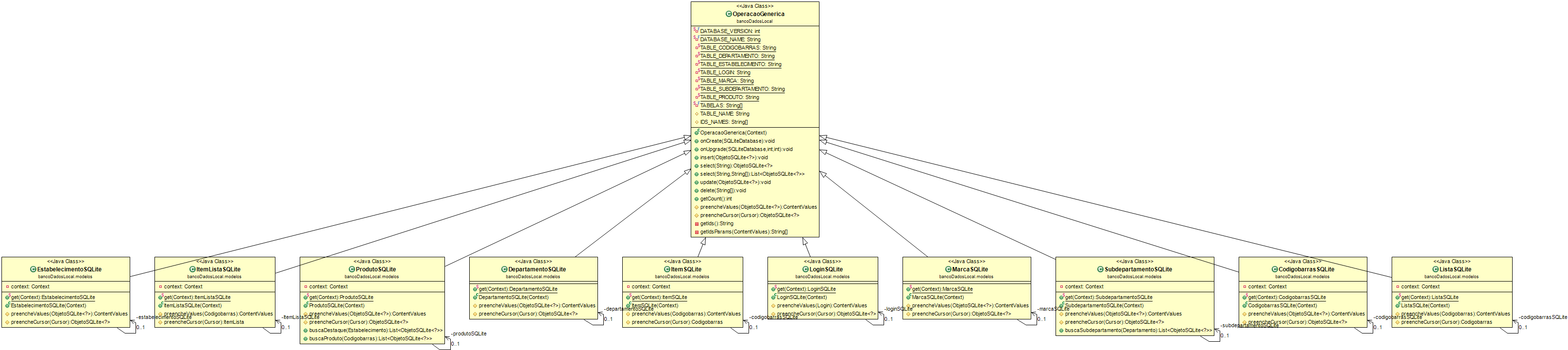
# Modelo de Classes

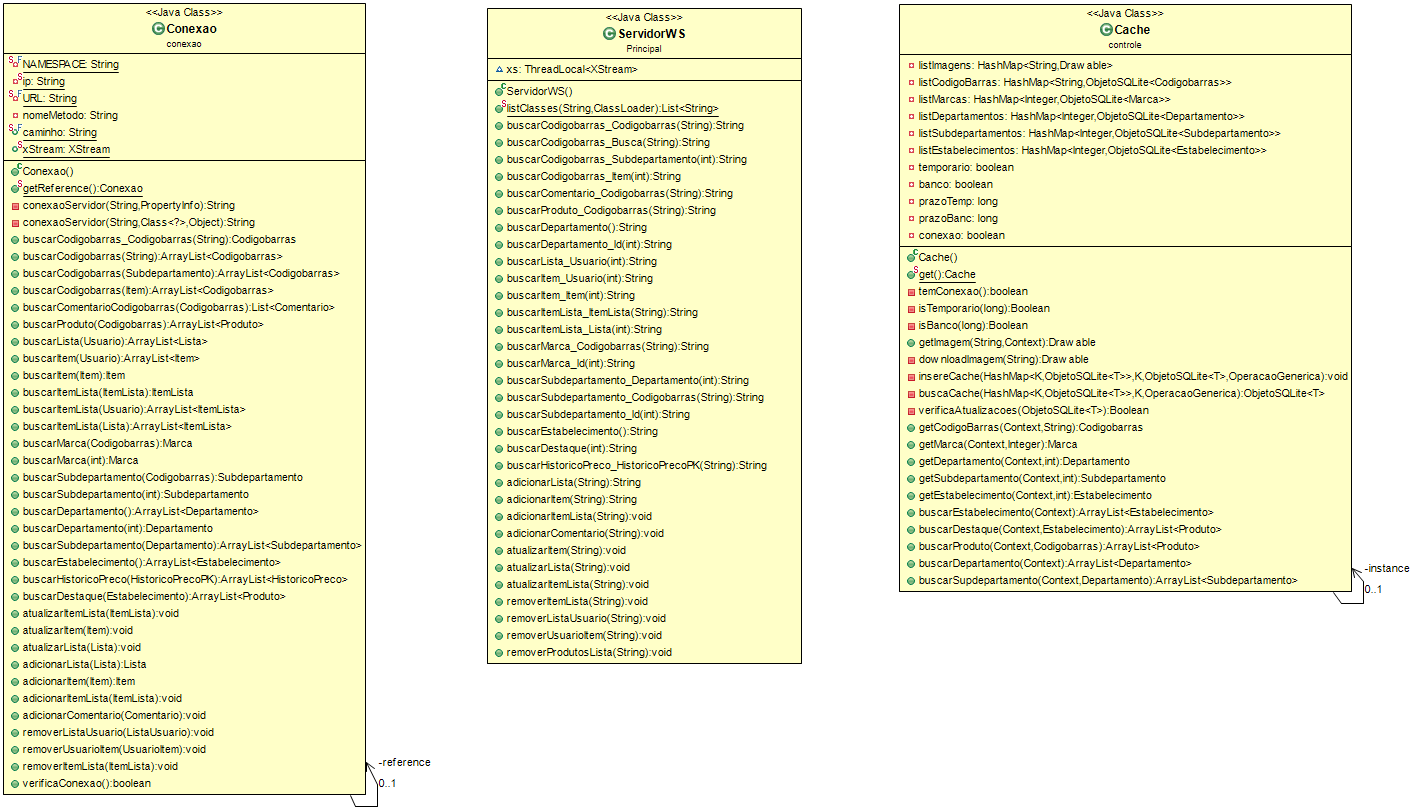
Abaixo estão os modelos de classes das principais classes do sistema. Para a melhor visualização das imagens é possível abri-las no diretório de documentações.

* *Modelos\_MySql*



* *Modelos\_JPA*
* *Modelos\_SQLite*

*Conexão*

**Módulo “Servidor”

Este módulo irá gerenciar as requisições de dados e ações, realizando a conexão com o banco de dados.

Abaixo estão os objetos pertencentes a este módulo.

**Classes de controle:**

* *Principal.Principal.java*

Irá publicar o WebService com o endereço “http://[ip]:8060/WebService/projeto", instanciando a classe ServidorWS.

* *Principal.Banco\_Dados.java*

Fornecerá dados para o gerenciamento da conexão com o banco de dados “01Search”.

* *Principal.ServidorWS.java*

Gerencia a comunicação entre a aplicação android - por meio do módulo conexão - e o servidor, convertendo os parâmetros recebidos por uma requisição para objetos java e serializando as respostas - referentes aos retornos dos objetos do banco de dados - para o protocolo de mensagem SOAP e assim poder enviar ao Android.

* *ControleJPA.Emf.java*

Fornece o método que retorna o gerenciador do banco de dados “01Search” (EntityManagerFactory).

* *ControleJPA.OperacaoGenerica.java*

É uma classe genérica que possui os principais métodos de busca, inserção, alteração e exclusão de registro em uma tabela no banco de dados.

As classes a seguir são todas filhas da classe OperacaoGenerica. Elas podem sobrepor os métodos da classe pai e ainda acrescentarem novos métodos para fazer diferentes operações no banco de dados.

* *ControleJPA.modelos.Codigobarras\_JPA.java*

Gerencia as operações da tabela Codigobarras no servidor.

* *ControleJPA.modelos.Departamento\_JPA.java*

Gerencia as operações da tabela Departamento no servidor.

* *ControleJPA.modelos.Estabelecimentos\_JPA.java*

Gerencia as operações da tabela Estabelecimento no servidor.

* *ControleJPA.modelos.HistoricoPreco\_JPA.java*

Gerencia as operações da tabela HistoricoPreco no servidor.

* *ControleJPA.modelos.ItemLista\_JPA.java*

Gerencia as operações da tabela ItemLista no servidor.

* *ControleJPA.modelos.Item\_JPA.java*

Gerencia as operações da tabela Item no servidor.

* *ControleJPA.modelos.Lista\_JPA.java*

Gerencia as operações da tabela Lista no servidor.

* *ControleJPA.modelos.Marca\_JPA.java*

Gerencia as operações da tabela Marca no servidor.

* *ControleJPA.modelos.Produto\_JPA.java*

Gerencia as operações da tabela Produto no servidor.

* *ControleJPA.modelos.Subdepartamento\_JPA.java*

Gerencia as operações da tabela Subdepartamento no servidor.

* *ControleJPA.modelos.UsuarioItem\_JPA.java*

Gerencia as operações da tabela Usuário\_Item no servidor.

* *ControleJPA.modelos.Usuario\_JPA.java*

Gerencia as operações da tabela Usuario no servidor.

As classes a seguir são todas classes-entidades, isto é, cada uma delas faz uma representação de uma tabela no banco de dados. Como o modo de conexão com o banco é a partir do framework hibernate implementado pela API JPA, todos os atributos das classes contem notações a fim de relacionar os atributos com colunas das respectivas tabelas.

* *Modelos.Codigobarras.java*

Representação da tabela codigobarras do banco de dados.

* *Modelos.CodigobarrasItem.java*

Representação da tabela Codigobarras\_Item do banco de dados.

* *Modelos.CodigobarrasItemPK.java*

Representação chave primária composta da tabela Codigobarras\_Item do banco de dados.

* *Modelos.Conversao.java*

Representação da tabela Conversao do banco de dados.

* *Modelos.Departamento.java*

Representação da tabela Departamento do banco de dados.

* *Modelos.Estabelecimento.java*

Representação da tabela Estabelecimento do banco de dados.

* *Modelos.Historico.java*

Representação da tabela Historico do banco de dados.

* *Modelos.HistoricoPK.java*

Representação chave primária composta da tabela Historico do banco de dados.

* *Modelos.HistoricoPreco.java*

Representação da tabela HistoricoPreco do banco de dados.

* *Modelos.HistoricoPrecoPK.java*

Representação chave primária composta da tabela HistoricoPreco do banco de dados.

* *Modelos.Item.java*

Representação da tabela Item do banco de dados.

* *Modelos ItemLista.java*

Representação da tabela ItemLista do banco de dados.

* *Modelo.ItemListaPK.java*

Representação chave primária composta da tabela ItemLista do banco de dados.

* *Modelos.Lista.java*

Representação da tabela Lista do banco de dados.

* *Modelos.ListaUsuario.java*

Representação da tabela ListaUsuario do banco de dados.

* *Modelos.Login.java*

Representação da tabela Login do banco de dados.

* *Modelos.Marca.java*

Representação da tabela Marca do banco de dados.

* *Modelos.Produto.java*

Representação da tabela Produto do banco de dados.

* *Modelos.ProdutoPK.java*

Representação chave primária composta da tabela Produto do banco de dados.

* *Modelos.Subdepartamento.java*

Representação da tabela Subdepartamento do banco de dados.

* *Modelos.UnidadeMedida.java*

Representação da tabela UnidadeMedida do banco de dados.

* *Modelos.Usuario.java*

Representação da tabela Usuario do banco de dados.

* *Modelos.UsuarioItem.java*

Representação da tabela UsuarioItem do banco de dados.

* *Modelos.UsuarioItemPK.java*

Representação chave primária composta da tabela UsuarioItem do banco de dados.

**Visuais:**

Este módulo não possui visuais.

# Módulo “Conexao”

O módulo Conexao é a ponte de comunicação entre a aplicação android (“01Search\_Android”) e o servidor (“01Search\_Servidor”), fazendo com que as requisições da aplicação android sejam encaminhadas ao servidor, e assim gerando uma resposta do servidor a caminho da aplicação.

Outra função deste módulo é a presença das ‘classes entidades’ - representações das tabelas do banco de dados em objetos - apenas para a aplicação android, enquanto que o servidor possui ‘classes entidades’ idênticas no que diz respeito aos atributos, mas contendo anotações de persistência da API JPA.

Abaixo estão os objetos pertencentes a este módulo.

**Classes de controle:**

* *conexao.Conexao.java*

Gerencia a comunicação entre a aplicação android e o servidor, serializando os parâmetros - referentes as requisições da aplicação android - para o protocolo de mensagem SOAP e assim poder enviar ao Web Service do servidor. Na resposta do Web Service, ele também é responsável por desserializar os pacotes e entregar à aplicação android.

* *Modelo.Codigobarras.java*

Representação da tabela “codigobarras” do banco de dados.

* *Modelo.CodigobarrasItem.java*

Representação da tabela “codigobarras\_item” do banco de dados.

* *Modelo.CodigobarrasItemPK.java*

Representação das chaves primárias da tabela “codigobarras\_item” do banco de dados.

* *Modelo.Comentario.java*

Representação da tabela “comentario” do banco de dados.

* *Modelo.Conversao.java*

Representação da tabela “conversao” do banco de dados.

* *Modelo.Departamento.java*

Representação da tabela “departamento” do banco de dados.

* *Modelo.Estabelecimento.java*

Representação da tabela “estabelecimento” do banco de dados.

* *Modelo.Historico.java*

Representação da tabela “historico” do banco de dados.

* *Modelo.HistoricoPK.java*

Representação das chaves primárias da tabela “historico” do banco de dados.

* *Modelo.HistoricoPreco.java*

Representação da tabela “historicopreco” do banco de dados.

* *Modelo.HistoricoPK.java*

Representação das chaves primárias da tabela “historicopreco” do banco de dados.

* *Modelo.ItemLista.java*

Representação da tabela “item\_lista” do banco de dados.

* *Modelo. ItemListaPK.java*

Representação das chaves primárias da tabela “item\_ lista” do banco de dados.

* *Modelo.Lista.java*

Representação da tabela “lista” do banco de dados.

* *Modelo.ListaUsuario.java*

Representação da tabela “lista\_usuario” do banco de dados.

* *Modelo.Login.java*

Representação da tabela “login” do banco de dados.

* *Modelo.Marca.java*

Representação da tabela “marca” do banco de dados.

* *Modelo.Produto.java*

Representação da tabela “produto” do banco de dados.

* *Modelo.ProdutoPK.java*

Representação das chaves primárias da tabela “produto” do banco de dados.

* *Modelo.Subdepartamento.java*

Representação da tabela “subdepartamento” do banco de dados.

* *Modelo.UnidadeMedida.java*

Representação da tabela “unidademedida” do banco de dados.

* *Modelo.Usuario.java*

Representação da tabela “usuario” do banco de dados.

* *Modelo.UsuarioItem.java*

Representação da tabela “usuario\_item” do banco de dados.

* *Modelo. UsuarioItemPK.java*

Representação das chaves primárias da tabela “usuario\_item” do banco de dados.

**Visuais:**

Este módulo não possui visuais.

# Módulo “Novidades”

É a primeira tela do programa onde são apresentadas destaques de produtos por estabelecimento. Os destaques estão cadastrados na tabela “Produto”, mas neste módulo é feita uma segunda busca para retornar os registros correspondentes da tabela “Codigobarras”.

Como o número de novidades por estabelecimento pode variar, os visuais são adicionados dinamicamente durante o tempo de execução do aplicativo.

Abaixo estão os objetos pertencentes a este módulo.

**Classes de controle:**

* *android.activity.NovidadesActivity.java*

Gerencia a requisição de novidades por estabelecimento para a classe cache e a inclusão dos elementos visuais na tela de acordo com seu estabelecimento e sua ordem.

**Visuais:**

* *layouts.NovidadesGeral.java*

É uma classe visual, estendendo um RelativeLayout. Este irá armazenar todos os objetos referentes à novidade de um estabelecimento, os quais serão discriminados abaixo.

* *layouts.NovidadesSupermercado.java*

É uma classe visual, estendendo um RelativeLayout. Este irá armazenar o cabeçalho de uma novidade, apresentando as informações (nome, logo e checkbox de favorito) do estabelecimento.

* *layouts.NovidadesProdutos.java*

É uma classe visual, estendendo um RelativeLayout. Irá agrupar “NovidadesProduto” de dois em dois

* *layouts.NovidadesProduto.java*

É uma classe visual, estendendo um RelativeLayout. Este irá armazenar as informações de um codigobarras, apresentando seu nome e imagem.

* *res/layout/novidades.xml*

É o visual xml do módulo. Apresenta apenas um scrollview para que as novidades sejam adicionadas dinamicamente quando o aplicativo for iniciado.

# Módulo “Categoria”

O módulo é acessado a partir do segundo botão da tela inicial, nele encontramos uma lista de todos os “Departamentos” criadas no sistema, a fim de dividir a disposição dos “Codigobarras”, que são os produtos sem a relação com algum estabelecimento. Além da separação em “Departamentos”, cada “Departamento” pode possuir uma lista de “Subdepartamentos” e assim dentro um "Subdepartamento” encontramos seus respectivos “Codigobarras”. Caso seja da vontade do usuário, ao clicar em um “Codigobarras” o sistema o enviará à tela “Produto” para maiores detalhes.

Abaixo estão os objetos pertencentes a este módulo.

**Classes de controle:**

* *android.activity.DepartamentosActivity.java*

Gerencia a entrada de todos os “Departamentos” e “Subdepartamentos” e a inclusão em uma lista, também é responsável por enviar o usuário à tela de “Codigobarras” de um “Subdepartamento”.

* *android.activity.CodigobarrasDepartamentosActivity.java*

Gerencia os “Codigobarras” de um determinado “Subdepartamento”, os elementos visuais, além de funções básicas como enviar o “Codigobarras” à tela com os detalhes do produto.

* *android.controle.Visual.DepartamentosControle.java*

Gerencia as requisições do aplicativo a classe de cache.

**Visuais:**

* *res/layout/categorias.xml*

É o visual xml da tela de categorias.

* *res/layout/produtoscategorias.xml*

É o visual xml da tela de produtos (“Codigobarras”) de um “Subdepartamento”.

# Módulo “Busca”

Este módulo irá gerenciar as buscas que o usuário pode digitar e retornar os “Codigobarras” referentes à sua requisição.

Abaixo estão os objetos pertencentes a este módulo.

**Classes de controle:**

* *android.activity.BuscaActivity.java*

Gerencia a requisição de buscas para a classe cache. Possui um editText no qual o usuário pode digitar sua string de busca. Ao clicar no botão de busca a activity irá buscar “Codigobarras” pelo método da classe Cache e os retornos serão listados em um listView logo abaixo do campo de busca.

O botão de fotografia tentará abrir uma nova intent, que é o aplicativo de leitura de código de barras previamente instalado no aparelho do usuário. O aplicativo irá retornar a string de retorno para o editText para que a busca possa ser feita.

**Visuais:**

* *res/layout/buscar.xml*

É o visual xml do módulo. Apresenta um editText para o usuário digitar sua busca, um botão para realizar a busca e outro para a fazer uma busca por código de barras. Além de um listView para apresentar os resultados da busca.

# Módulo “Listas”

O módulo ‘listas’ compreende uma das principais funcionalidades do projeto.

Como o próprio nome já diz, nele é possível a criação de listas customizáveis. Dentro de cada lista é possível a criação de itens também customizáveis, e, para cada item, podemos agregá-lo a um ou vários produtos - no projeto, a representação elementar de um produto é o “Codigobarras”, assim dentro de um determinado item é possível colocar produtos de marcas que agrade o usuário.

Outra funcionalidade, dentro desse módulo, é a duração que um item tem para o seu usuário – no projeto, essa duração é denominada “periodicidade”. Para exemplificar o conceito vejamos um exemplo: um usuário consome uma unidade de barbeador por semana. Assim ele pode configurar a duração do item ‘barbeador’ para 7 dias, fazendo com que o sistema mostre a ele a previsão de termino deste item.

Além dessa possibilidade, caso o usuário deseje configurar uma duração especifica para uma lista em questão, ele também o pode.

O módulo irá gerenciar ainda o compartilhamento de listas entre usuários, de modo que dois ou mais usuários possam usufruir da mesma lista.

Por fim é possível comparar o preço de compra da lista em todos os estabelecimentos disponíveis, podendo ter uma base de informação para o usuário na hora da compra.

Abaixo estão os objetos pertencentes a este módulo.

**Classes de controle:**

* *android.ListaActivity.java*

Gerencia a requisição das listas do usuário para a classe cache, a inclusão dos elementos visuais na tela e acesso ao compartilhamento de listas entre usuários.

* *android.ItemListaActivity.java*

Gerencia a requisição dos itens de uma lista, – no projeto, “ItemLista” para a classe cachê – a inclusão dos elementos visuais na tela alem da funcionamento do ‘Modo Compra’ – tela em que são apresentados os itens – e suas quantidades compradas, com o preço total da lista além das previsões de duração de cada item.

* *android.ItemActivity.java*

Gerencia a requisição dos itens do usuário para a classe cache, a inclusão dos elementos visuais na tela, além das funcionalidades de manipulação de um item.

* *android.ConfigurarItemActivity.java*

Gerencia a requisição dos “Codigobarras” de um determinado item do usuário para a classe cache, a inclusão dos elementos visuais na tela e a manipulação dos “Codigobarras” deste item, podendo adicionar novos “Codigobarras” ou excluir os já existentes.

* *controle.Visual.ListaControle.java*

Controla as requisições do visual à classe Cache.

* *controle.Visual.ItemListaControle.java*

Controla as requisições do visual à classe Cache.

* *controle.Visual.ItemControle.java*

Controla as requisições do visual à classe Cache.

* *controle.Visual.ConfigurarItemControle.java*

Controla as requisições do visual à classe Cache.

* *layouts.Lista\_Adapter.java*

Adapter para ListView customizáveis.

* *layouts.ItemListaExpandableListAdapter.java*

Adapter para ExpandableListView customizáveis.

* *layouts.Item\_Adapter.java*

Adapter para ListView customizáveis.

* *layouts.Codigobarras\_Adapter.java*

Adapter para ListView customizáveis.

* *layouts.Item\_ListViewAdapter.java*

Adapter para ListView da comparação entre estabelecimentos.

* *layouts.ItemListView.java*

Objeto de uma linha referente um estabelecimento e o seu preço da lista.

**Visuais:**

* *res/layout/compartilhar.xml*

É o visual xml do conteúdo para um Dialog referente ao compartilhamento de listas.

* *res/layout/configurar\_item.xml*

É o visual xml do conteúdo para a configuração de um determinado item.

* *res/layout/dialo\_add\_itemlista.xml*

É o visual xml do conteúdo para um Dialog ao adicionar um item na lista utilizada.

* *res/layout/dialo\_quantidade.xml*

É o visual xml do conteúdo para um Dialog referente a quantidade de um item na lista, como também da quantidade a ser comprada de um item.

* *res/layout/item\_listview.xml*

É o visual xml do conteúdo para um Dialog referente à comparação de preços entre estabelecimentos.

* *res/layout/itemlinha\_linhafilho.xml*

É o visual xml do conteúdo da linha de “Codigobarras” dentro de um item no ExpandableListView da tela de compra.

* *res/layout/itemlinha\_linhapai.xml*

É o visual xml do conteúdo da linha de “Item” no ExpandableListView da tela de compra.

* *res/layout/itemlista.xml*

É o visual xml do conteúdo da tela de itens de uma lista (tela de compra).

* *res/layout/linha\_codigobarras.xml*

É o visual xml do conteúdo da linha de “Codigobarras” da tela de configuração de um item.

* *res/layout/linha\_codigobarras.xml*

É o visual xml do conteúdo da linha para “Lista” na tela de listas e para “Item” na tela de itens.

* *res/layout/lista.xml*

É o visual xml do conteúdo da tela de listas.

# Módulo “Produto”

Este módulo irá gerenciar as informações mostradas ao usuário referente ao Codigobarrras selecionado.

Abaixo estão os objetos pertencentes a este módulo.

**Classes de controle:**

* *android.activity.CodigobarrasActivity.java*

É a classe principal do módulo. Ela irá receber um Codigobarras e mostras suas informações nos respectivos campos para o usuário.

* *android.activity.GraficoActivity.java*

Faz a requisição do servido da evolução de preço de determinado produto e, mostra em um webview o gráfico gerado pelo serviço de geração de gráfico do Google.

* *controle.Visual.CodigobarrasControle.java*

Controla as requisições do visual à classe Cache e mantem o codigobarras corrente.

* *controle.Visual.ComentariosActivity*

Gerencia a visualização de comentários de determinado codigobarras.

**Visuais:**

* *res/layout/produto.xml*

É o visual xml do módulo. Possui os principais campos de informações de um codigobarras, além do menu que irá chamar as outras telas deste módulo.

* *res/layout/grafico.xml*

Possui apenas um cabeçalho e um webview onde será mostrado o gráfico de evolução de preço de um produto.

* *res/layout/comentarios.xml*

*O layout da tela de comentários.*

* *res/layout/comentario.xml*

*O layout de um comentário único.*

# Módulo “Cache”

Este módulo irá gerenciar as buscas de dados seja na memória cache temporária (feita com hashmaps), a cache persistente (SQLite) ou buscas ao servidor externo.

Abaixo estão os objetos pertencentes a este módulo.

**Classes de controle:**

* *bancoDadosLocal.ObjetoSQLite.java*

É uma classe genérica que possui um objeto das classes Conexao.modelos e um campo de date de modificação. Essa data ira registrar a data da ultima modificação daquele objeto no SQLite.

* *bancoDadosLocal.OperacaoGenerica.java*

É uma classe genérica que possui os scripts de criação das tabelas no SQLite e os principais métodos de busca, inserção, alteração e exclusão de registro em uma tabela.

Todas as requisições e retornos dessa classe são feitas através de objetos da classe “ObjetoSQLite”.

* *controle.Cache.java*

Esta classe gerencia as requisições do programa ao servidor.

O cache possui 2 tipos: um cache temporário e um cache persistente, que utiliza o banco de dados SQLite. A ideia é que a utilização destes e seu tempo de utilização sejam parametrizados pelo usuário.

Caso o objeto solicitado não exista em nenhum dos caches o programa irá fazer uma requisição ao servidor por meio da classe “Conexao”.

Em toda requisição ao servidor externo o programa verifica se a conexão com tal está funcionando. Se uma requisição falhe o valor do campo boolean conexão será alterada para false, a fim de evitar futuras requisições à toa ao servidor.

Como regra, os métodos que recebem o id do objeto como parâmetro irão realizar as buscas primeiramente na memória cachê do aparelho. Já os métodos que retornem arraylists de objetos fazem primeiramente uma solicitação ao servidor e caso este esteja offline, é feita a busca na memória local.

As classes a seguir são todas filhas da classe ObjetoSQLite. Elas possuem os métodos “preencheValues(ObjetoSQLite<?> obj)” e “preencheCursor(Cursor c)” que são especifico de cada classe pois cada um irá preencher os valores de acordo com o seu objeto. Esta classe pode sobrepor os métodos da classe pai e pode acrescentar métodos para fazer diferentes operações no banco de dados local.

* *bancoDadosLocal.modelos.CodigobarrasSQLite.java*

Objeto pertencente: Conexão.modelos.Codigobarras

* *bancoDadosLocal.modelos.DepartamentoSQLite.java*

Objeto pertencente: Conexão.modelos.Departamento

* *bancoDadosLocal.modelos.EstabelecimentoSQLite.java*

Objeto pertencente: Conexão.modelos.Estabelecimento

* *bancoDadosLocal.modelos.LoginSQLite.java*

Objeto pertencente: Conexão.modelos.Login

* *bancoDadosLocal.modelos.MarcaSQLite.java*

Objeto pertencente: Conexão.modelos.Marca

* *bancoDadosLocal.modelos.ProdutoSQLite.java*

Objeto pertencente: Conexão.modelos.Produto

* *bancoDadosLocal.modelos.SubdepartamentoSQLite.java*

Objeto pertencente: Conexão.modelos.Subdepartamento

**Visuais:**

Este módulo não possui visuais.