

Trabalho de Pesquisa Aplicada a Novas Metodologias para Desenvolvimento de Sistemas

**Professor** : Fabio de Jesus Souza

Analise e Desenvolvimento de Sistemas

Nome : Bruno Alves de Araújo – RA:314101911

Nome: Thiago Pereira - RA : 914123856

Nome : Valber Antonio Rezende – RA: 314101911

Sumário

[1. Termo de Abertura do Projeto 1](#_Toc453193275)

[1.1. Justificativa 1](#_Toc453193276)

[1.2. Objetivo 1](#_Toc453193277)

[1.3. Intitulação do Sistema 1](#_Toc453193278)

[1.4. Restrições 1](#_Toc453193279)

[2. Requisitos 1](#_Toc453193280)

[2.1. Requisitos Funcionais 1](#_Toc453193281)

[2.2. Requisitos Não Funcionais 1](#_Toc453193282)

[3. Diagrama de Casos de Uso 1](#_Toc453193283)

[3.1 - 001 Autenticação no Sistema (Login) 1](#_Toc453193284)

[3.2 – 002 Cadastrar Produto 1](#_Toc453193285)

[3.3 – 003 Redefinir Senha 1](#_Toc453193286)

[3.4 – 004 Consultar Produtos 1](#_Toc453193287)

[3.5 – 005 Alterar Produto 1](#_Toc453193288)

[3.6 – 006 Deletar Produto 1](#_Toc453193289)

[3.7 – 007 Cadastrar Cliente 1](#_Toc453193290)

[3.8 – 008 Consultar Produto 1](#_Toc453193291)

[3.9 – 009 Alterar Cliente 1](#_Toc453193292)

[3.10 – 010 Deletar Cliente 1](#_Toc453193293)

[3.11 – 011 Criar Anúncio 1](#_Toc453193294)

[3.12 – 012 Calcular Taxa de Juros 1](#_Toc453193295)

[3.13 – 013 Cadastrar Pedido 1](#_Toc453193296)

[3.14 – 014 Incluir Movimentação Financeira 1](#_Toc453193297)

[3.15 – 015 Atualizar Caixa 1](#_Toc453193298)

[3.16 – 016 Consultar Pedido 1](#_Toc453193299)

[3.17 – 017 Cancelar Pedido 1](#_Toc453193300)

[3.18 - 018 Alterar Status do Pedido 1](#_Toc453193301)

[3.19 – 019 Emitir Relatório de Vendas 1](#_Toc453193302)

[3.20 – 020 Consultar Relatório de Vendas 1](#_Toc453193303)

[4. Diagrama de Classes 1](#_Toc453193304)

[4.1 Pacote Beans 1](#_Toc453193305)

[4.2 Pacote Model.Dao 1](#_Toc453193307)

[4.3 Pacote Controlle**r 1**](#_Toc453193309)

**5.** [**Cronograma do Projeto** 1](#_Toc453193310)

[5.1 Definição 1](#_Toc453193311)

[5.2 Modelos e Diagramas 1](#_Toc453193312)

[5.3 Diagramas UML 1](#_Toc453193313)

[5.4 Prototipação 1](#_Toc453193314)

[5.5 Protótipos 1](#_Toc453193315)

[5.6 Desenvolvimento Front-End 1](#_Toc453193316)

[5.7 Desenvolvimento Back-End 1](#_Toc453193317)

[5.8 Testes 1](#_Toc453193318)

[5.9 Implantação 1](#_Toc453193319)

[6. Modelagem de Dados 1](#_Toc453193320)

[6.1 Modelo Relacional Lógico 1](#_Toc453193321)

[6.2 Projeto Físico do Banco de Dados 1](#_Toc453193322)

[6.2.1 Criação das tabelas 1](#_Toc453193323)

[6.2.2 Procedures 1](#_Toc453193324)

[7. Elaboração do Plano de Teste 1](#_Toc453193325)

[7.1 Histórico das Revisões 1](#_Toc453193326)

[7.2 Objetivos 1](#_Toc453193327)

[7.3 Identificação de Projeto 1](#_Toc453193328)

[7.4 Requisitos a Testar 1](#_Toc453193329)

[7.4.1 Teste do Banco de Dados 1](#_Toc453193330)

[7.4.2.1 Teste Funcional 1](#_Toc453193331)

[7.4.2.2 Teste da Interface do Usuário 1](#_Toc453193332)

[7.4.4 Teste de Segurança e de Controle de Acesso 1](#_Toc453193333)

[7.4.5 Teste de Instalação 1](#_Toc453193334)

[**8.1 Introdução** 1](#_Toc453193335)

[8.2 Principais Componetes 1](#_Toc453193336)

[8.2.A Cadastrando um Pedido 1](#_Toc453193337)

[8.2.B Consultando um Pedido 1](#_Toc453193338)

[8. Desenvolvimento das Interfaces (Protótipos) 1](#_Toc453193339)

[9.1 Consulta de Pedidos 1](#_Toc453193340)

[9.2 Consulta de Pedidos Modal 1](#_Toc453193341)

[9.3 Cadastro de Pedidos 1](#_Toc453193342)

[9.4 Cadastro de Pedidos Modal 1](#_Toc453193343)

[Referências Bibliográficas 1](#_Toc453193344)

# 1. Termo de Abertura do Projeto

## Justificativa

O presente projeto surge da necessidade de controle das vendas da empresa *Cool & Cute* que comercializa produtos através do site Mercado Livre. Hoje, o controle das vendas é feito através de uma planilha eletrônica que contém os registros das vendas. Devido ao aumento da demanda, o controle está se tornando trabalhoso e, em alguns casos, gera erros em decorrência do alto volume de dados sem organização. Com base nestes problemas, este projeto visa criar um sistema que possibilite o controle das vendas de forma mais organizada e rápida.

## Objetivo

Tornar o controle de vendas da empresa *Cool & Cute* mais organizado, simples e rápido através de um sistema desenvolvido para plataforma Web onde, o acesso possa ser realizado de diferentes sistemas e dispositivos a qualquer hora e por vários usuários.

## Intitulação do Sistema

O sistema decorrente deste projeto será intitulado de Cool & Cute - Vendas.

## Restrições

* + O projeto deverá ser desenvolvido até o dia 08/06/2016.

# Requisitos

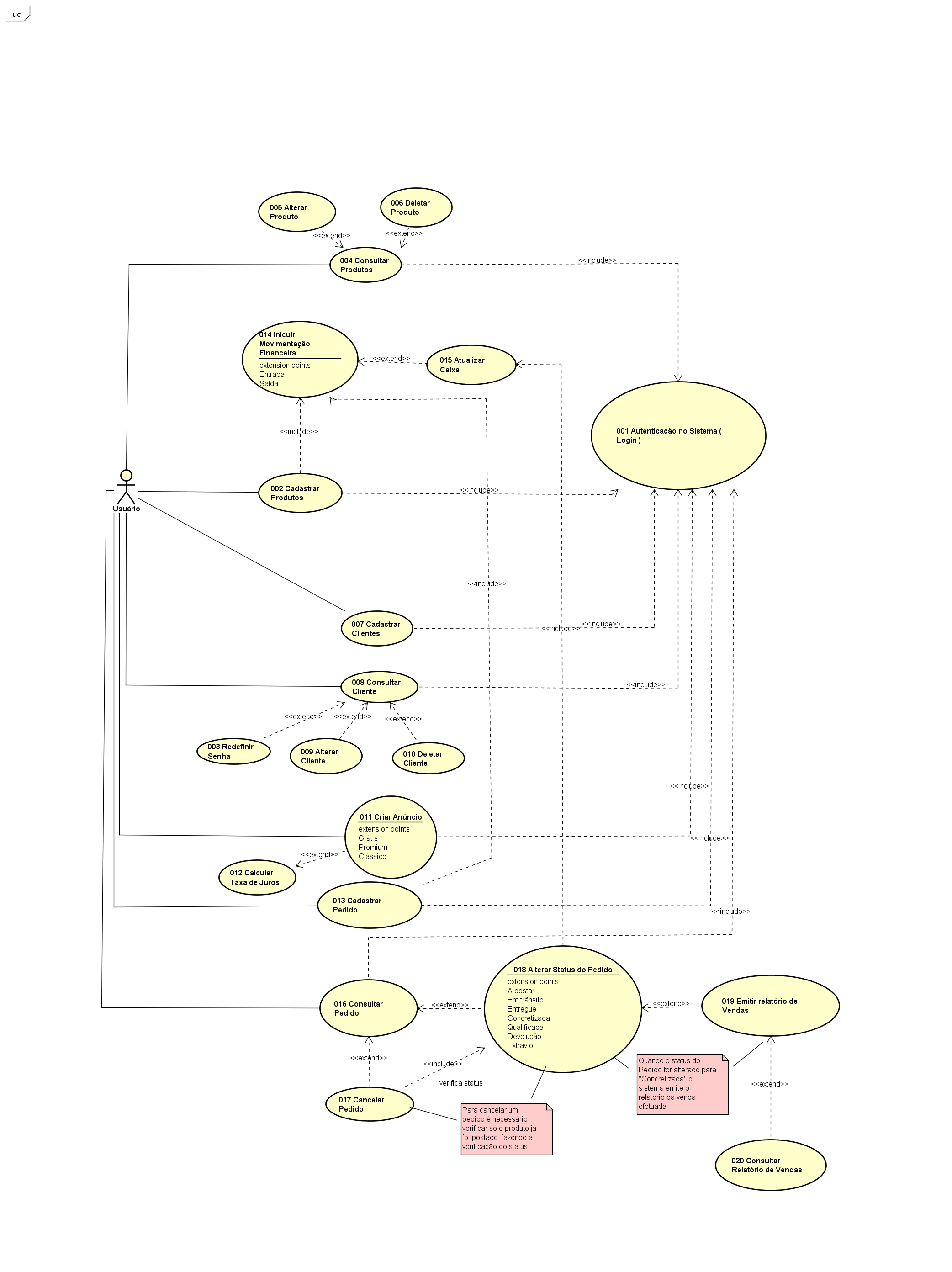
## Requisitos Funcionais

* + 1. **Em relação aos clientes**
       1. Os dados necessários são Nome e e-mail.
    2. **Em relação aos anúncios**
       1. Os dados guardados serão: Descrição, preço e o tipo de anúncio, data de criação e status.
          1. O site Mercado Livre cobra uma taxa sobre cada produto vendido no pedido. Está taxa é uma porcentagem do preço do anúncio, elas são dividas em três tipos.
* Premium – o anúncio tem visibilidade privilegiada e não oferece juros ao cliente em compras parceladas. A porcentagem descontada é de 16% sobre cada produto vendido no pedido;
* Clássico - o anúncio tem visibilidade privilegiada, porém, o juros das compras parceladas são repassadas ao cliente. A porcentagem descontada é de 10% sobre cada produto vendido no pedido; e
* Grátis – não existe desconto de taxa, porém, a visibilidade do anúncio é baixa 0%.
  + - 1. Os anúncios podem sofrer diversas alterações de preço.
    1. **Em relação aos produtos**
       1. Os dados necessários são: Nome, descrição, categoria, quantidade, estoque mínimo e preço.
       2. O mesmo produto pode ser adquirido através de vários fornecedores, portanto, o preço pode variar.
    2. **Em relação às entradas**
       1. Os dados necessários são: Produto, data, quantidade, preço e lote.
    3. **Em relação aos pedidos**
       1. Os dados necessários são: Itens do pedido, anúncio relacionado, cliente, data da venda, data de postagem, data de entrega, código da postagem, valor do frete, valor total do pedido, status e observação.
       2. Os status possíveis para o pedido são:
* A postar – Quando a postagem dos itens do pedido ainda não foi realizada.
* Em trânsito – Quando a postagem já foi realizada, porém, o cliente ainda não recebeu.
* Entregue – Quando a mercadoria chegou ao cliente.
* Concretizada – Após 48h da entrega do produto ao cliente.
* Qualificada – Quando o cliente faz a qualificação da venda. As qualificações podem ser positivas, neutras e negativas.
* Devolução – Quando a mercadoria é devolvida ou não pode ser entregue.
* Extravio – Quando a mercadoria é extraviada durante o trânsito.
  + - 1. Um pedido só poderá ser cancelado caso a postagem não tenha sido realizada.
      2. Quando a mercadoria é entregue ao cliente, a data de entrega deve ser preenchida.
      3. O sistema deve emitir relatórios de vendas com possibilidade de combinação de filtros.
    1. **Em relação às movimentações financeiras**
       1. Ao realizar uma compra, o dinheiro fica inicialmente bloqueado. A liberação ocorre dois dias após a entrega da mercadoria ou quando a negociação é avaliada positivamente pelo cliente. Desta forma, o valor da venda fica disponível em uma espécie de conta no site do Mercado Livre e pode ser movimentado para uma conta bancária.
       2. Mesmo após a liberação do pagamento o dinheiro pode ficar indisponível se o cliente registrar uma reclamação, neste caso, o valor do pedido ficará retido em um saldo chamado “em mediação” até a resolução do problema.
       3. É necessário ter o controle de 4 saldos resultantes das vendas.
* Não disponível: todas as vendas cuja entrega foi feita a menos de dois dias referente ao dia atual ou quando a entrega ainda não foi realizada.
* Disponível: quando a entrega foi realizada à mais de 48 horas ou caso o cliente avalie a venda de forma positiva.
* Em mediação: quando o cliente registra reclamação de uma venda.
* Caixa: dinheiro que se encontra na conta bancária da empresa.
  + - 1. É necessário armazenar gastos sazonais e de rotina como estacionamento, gasolina.
      2. Os sócios poderão realizar retiradas do caixa como pró-labore.
      3. Toda retirada de valor da conta do Mercado Livre é acompanhada de uma taxa fixa em cima do valor. Está taxa é estipulada pelo site Mercado Livre por isso, pode sofrer alteração a qualquer momento.
      4. As retiradas da conta do Mercado Livre compõem o caixa disponível para compras da empresa.
    1. **Em relação aos usuários do sistema**
       1. Para acessar o sistema é necessário possuir usuário e senha.

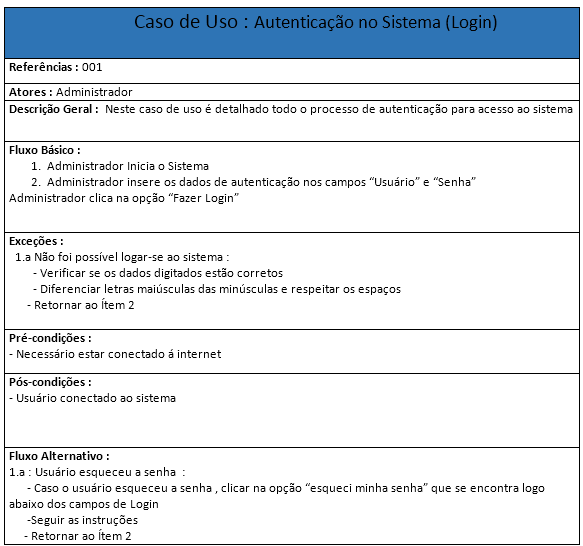
## Requisitos Não Funcionais

* + 1. Ser desenvolvido para Web;
    2. Ser portável;
    3. Funcionar corretamente nos navegadores Google Chrome, Internet Explorer versão 10 ou posterior, Firefox nos sistemas operacionais Windows 7, Windows 8, Windows 8.1 e Windows 10. Em dispositivos móveis, Safari e Google Chrome no IOS 8 e IOS 9, Google Chrome e Firefox nos sistemas Android 4.4, Android 5.0, Android 5.1 e Android 6;
    4. Ser desenvolvido com a linguagem Java pacote EE; e
    5. Utilizar o sistema gerenciador de banco de dados MySQL para o armazenamento dos dados.

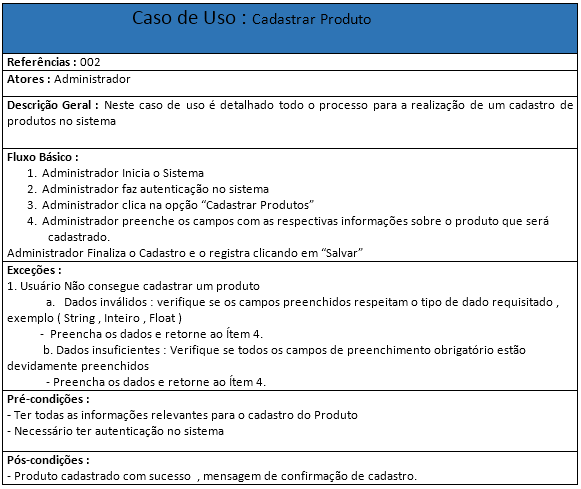
# Diagrama de Casos de Uso



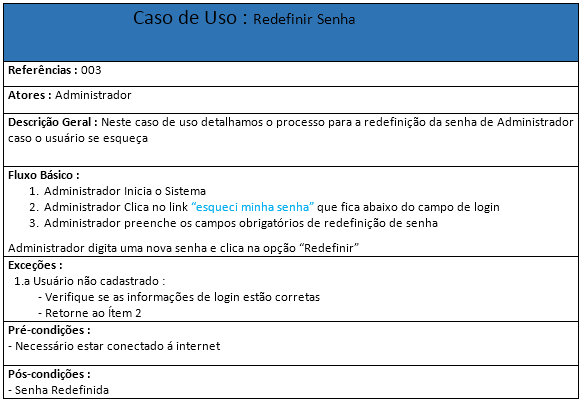
## 3.1 - 001 Autenticação no Sistema (Login)



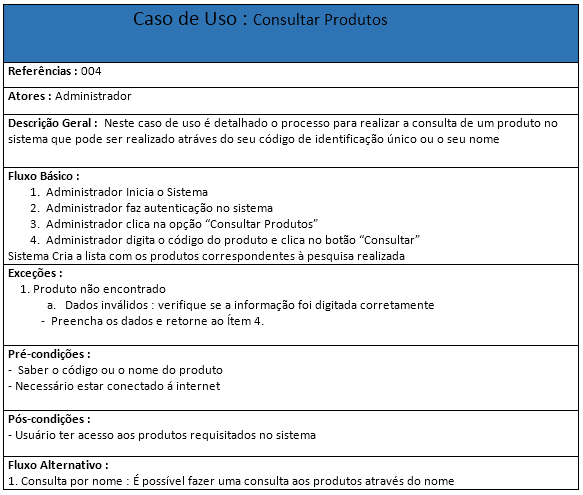
## 3.2 – 002 Cadastrar Produto



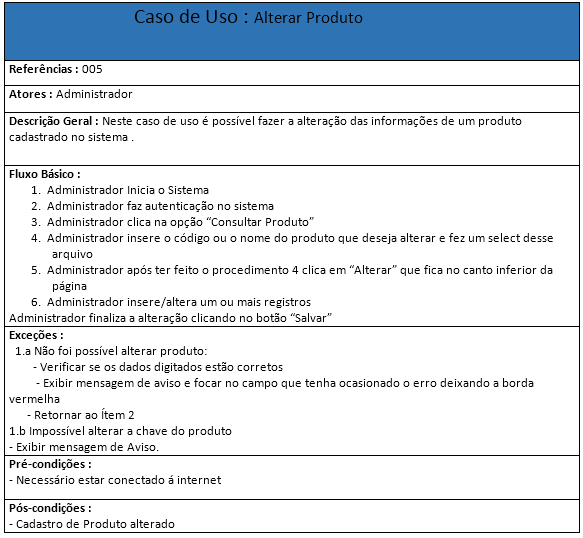
## 3.3 – 003 Redefinir Senha



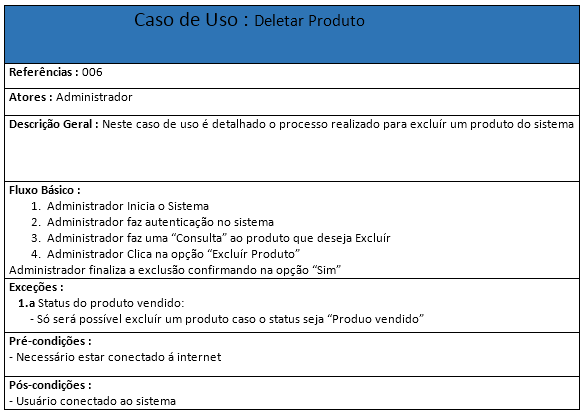
## 3.4 – 004 Consultar Produtos

****

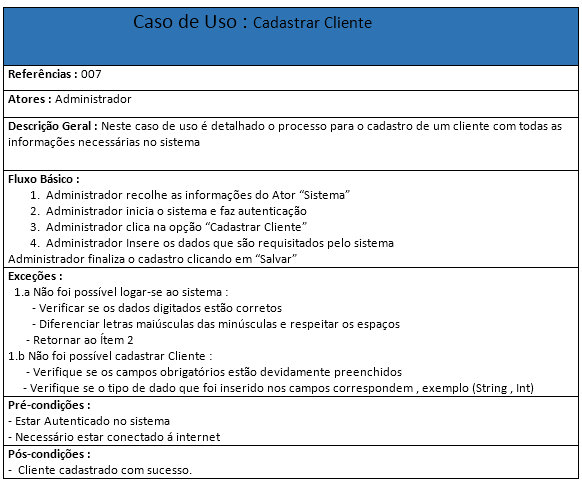
## 3.5 – 005 Alterar Produto



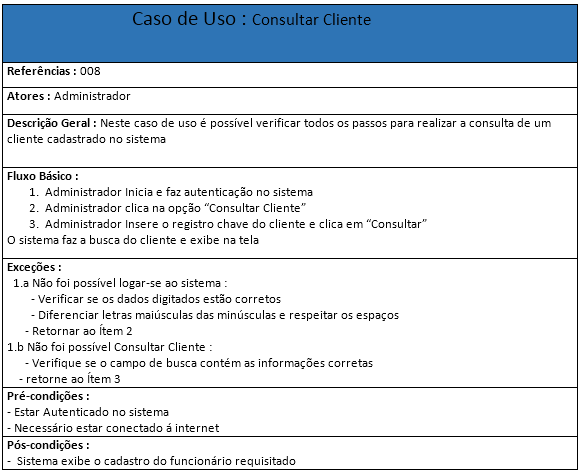
## 3.6 – 006 Deletar Produto



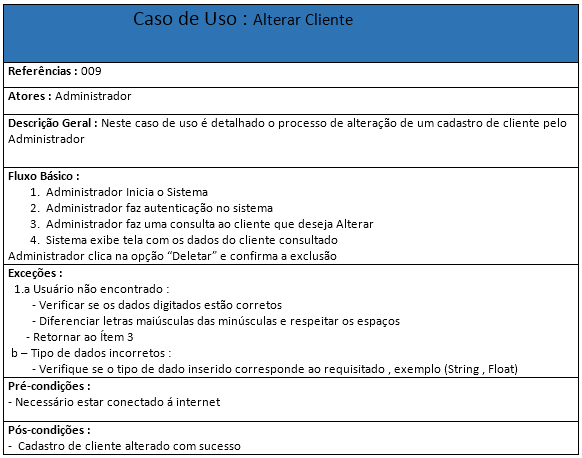
## 3.7 – 007 Cadastrar Cliente



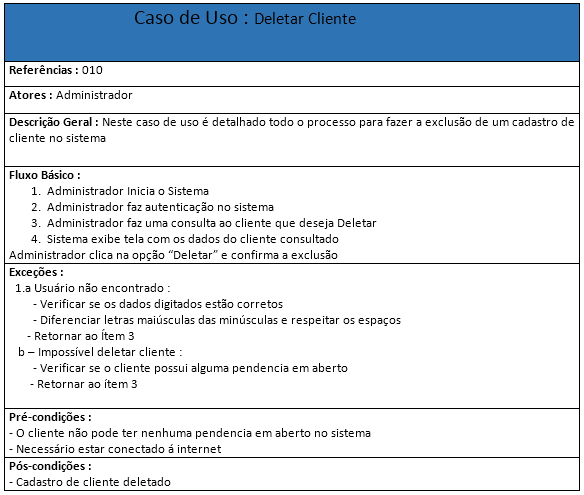
## 3.8 – 008 Consultar Produto



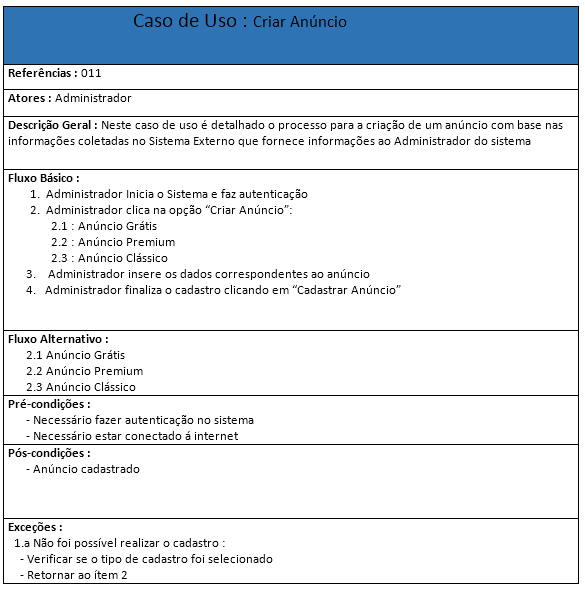
## 3.9 – 009 Alterar Cliente



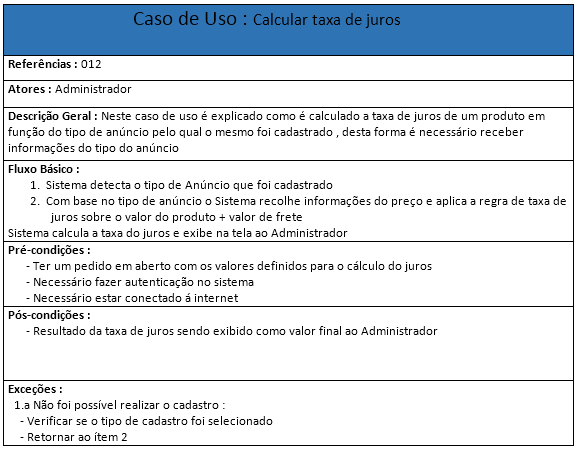
## 3.10 – 010 Deletar Cliente



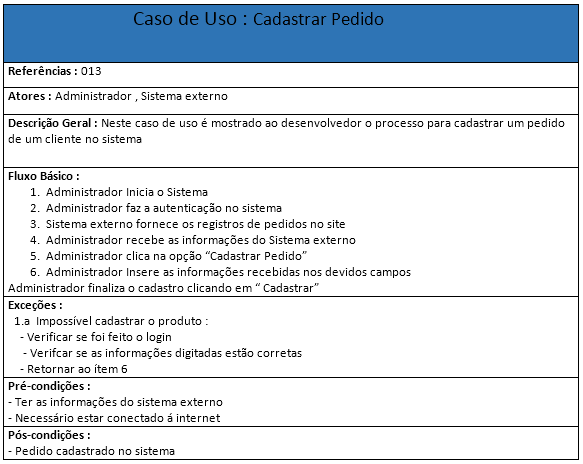
## 3.11 – 011 Criar Anúncio



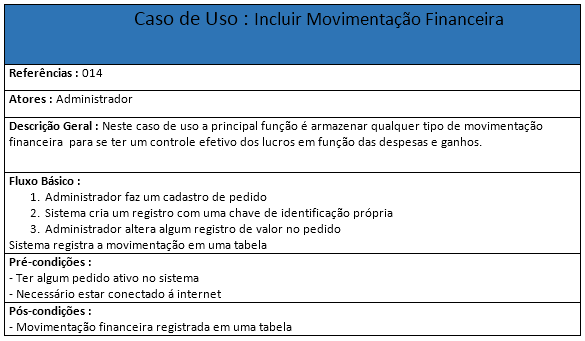
## 3.12 – 012 Calcular Taxa de Juros



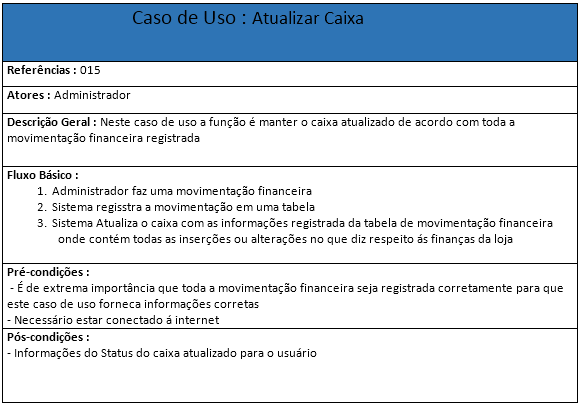
## 3.13 – 013 Cadastrar Pedido



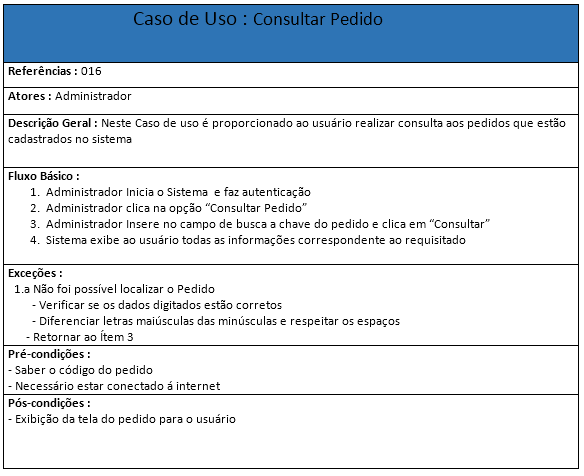
## 3.14 – 014 Incluir Movimentação Financeira



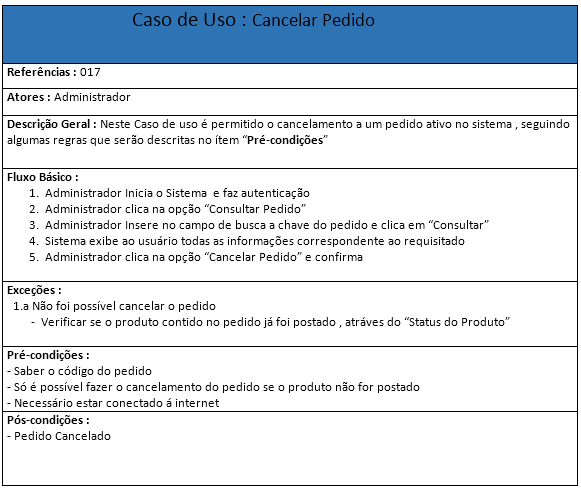
## 3.15 – 015 Atualizar Caixa



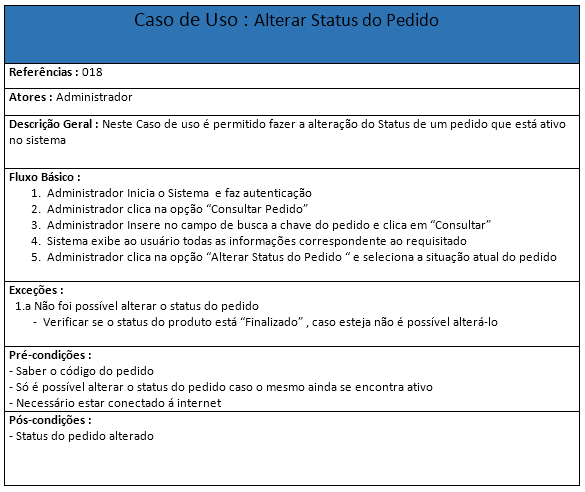
## 3.16 – 016 Consultar Pedido



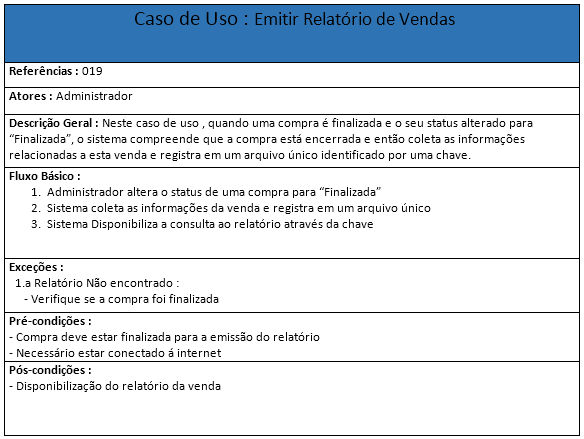
## 3.17 – 017 Cancelar Pedido



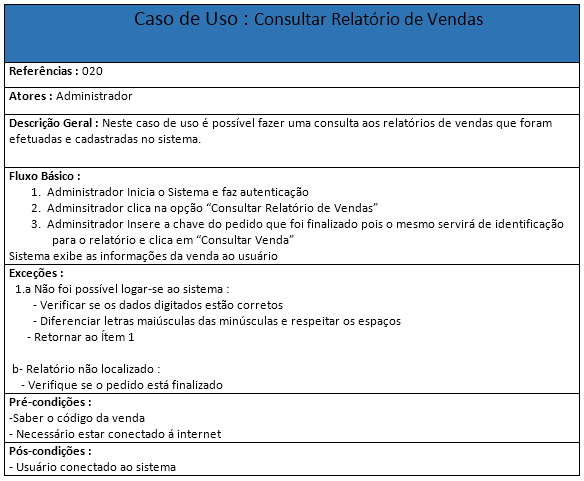
## 3.18 - 018 Alterar Status do Pedido



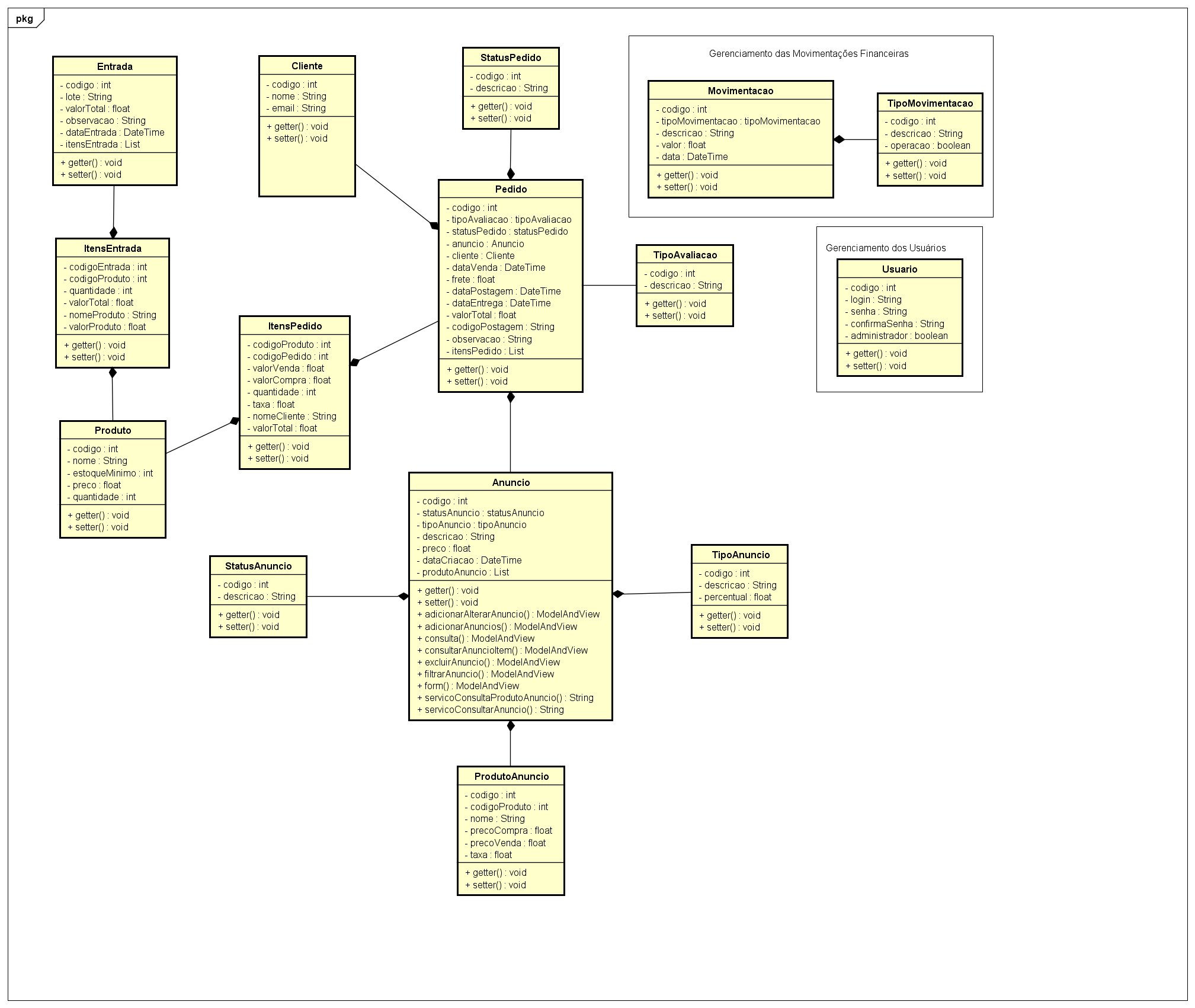
## 3.19 – 019 Emitir Relatório de Vendas



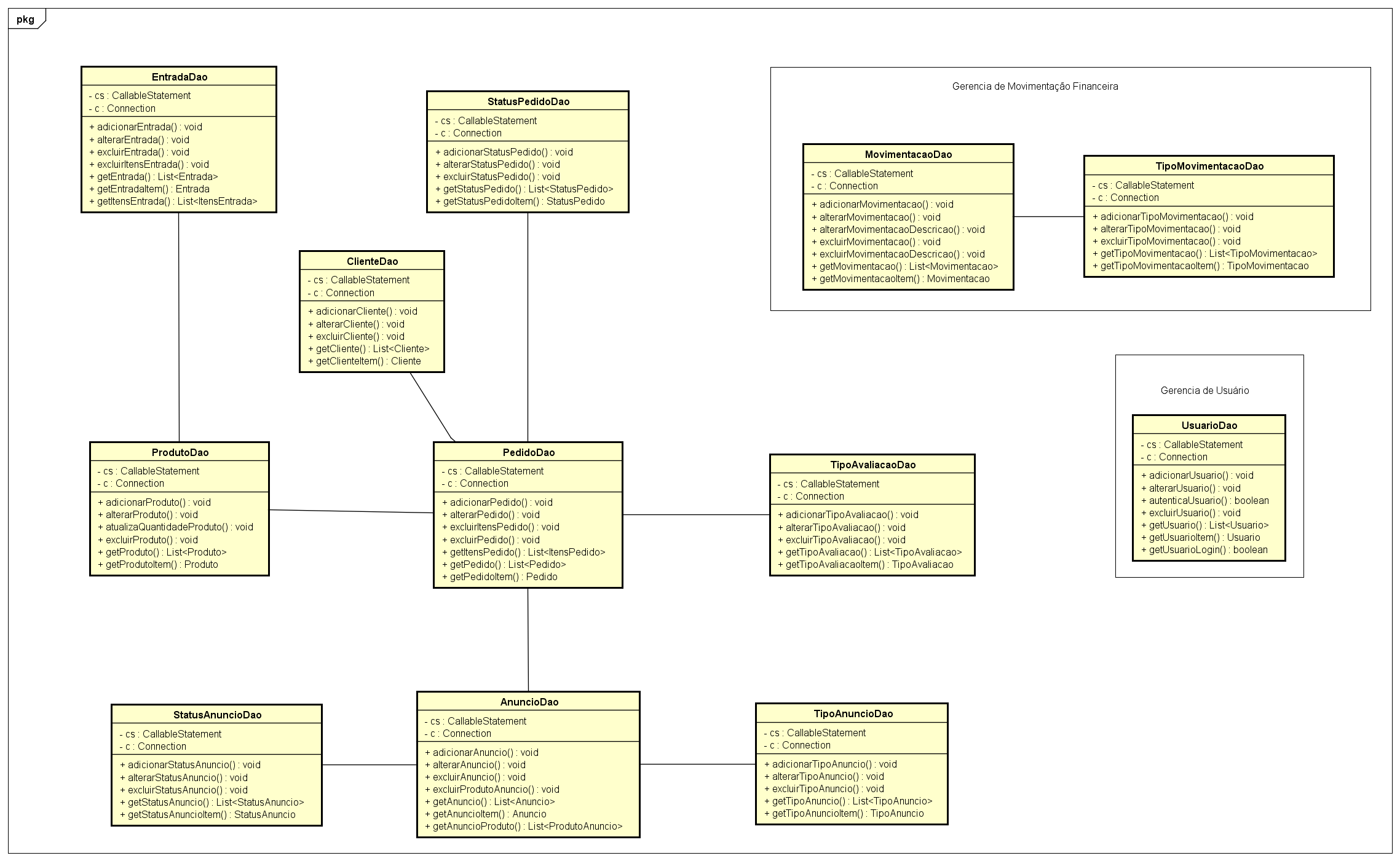
## – 020 Consultar Relatório de Vendas



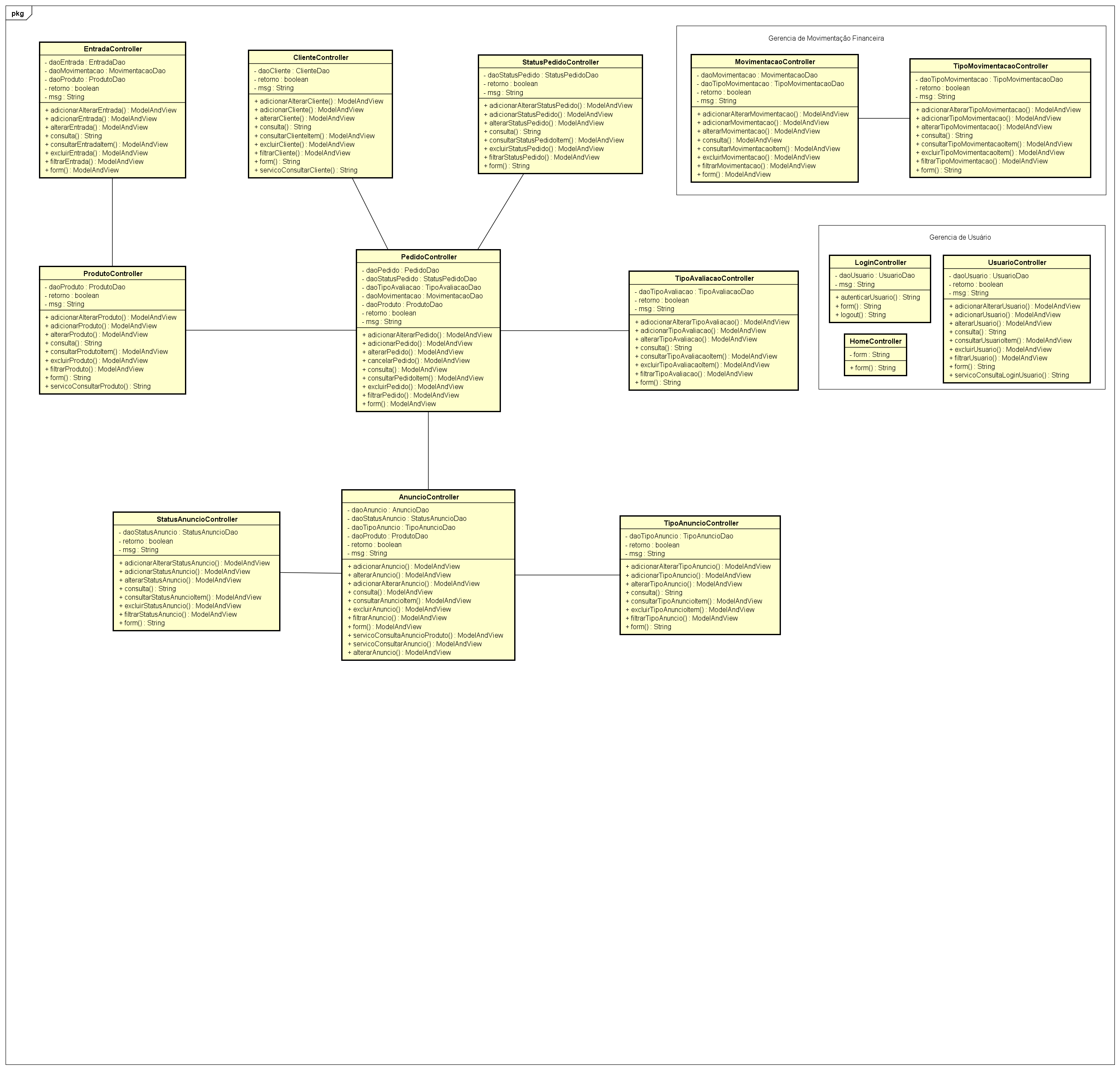
1. **Diagrama de Classes**
   1. **Pacote Beans**

****

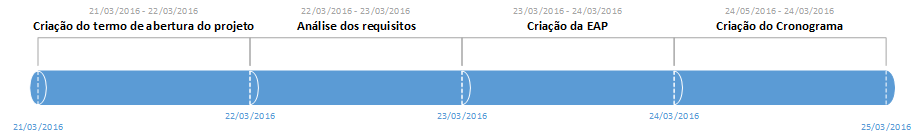
## 4.2 Pacote Model.Dao



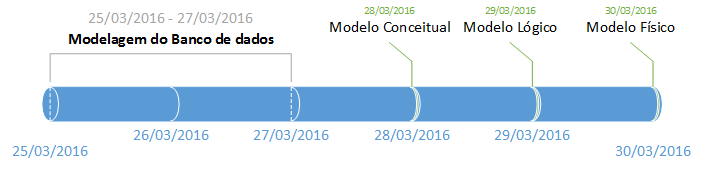
**4.3 Pacote Controller**



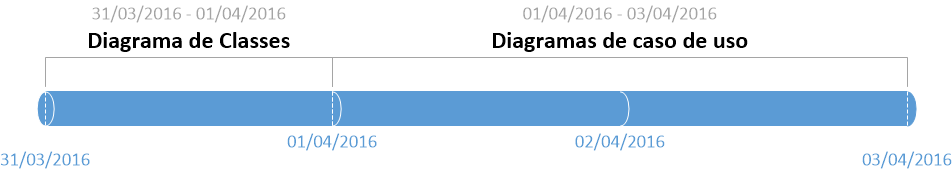
1. **Cronograma do Projeto**

**5.1 Definição**

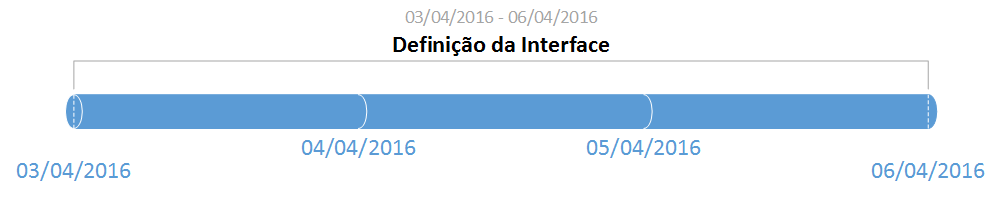
## 5.2 Modelos e Diagramas

****

## 5.3 Diagramas UML



## 5.4 Prototipação



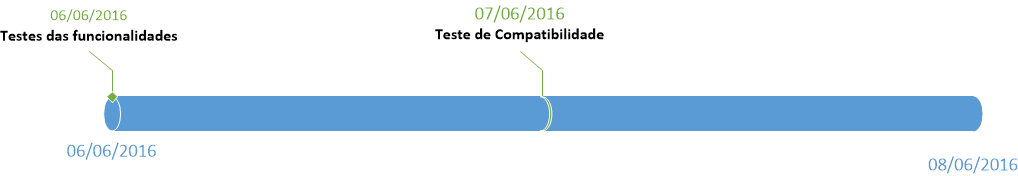
## 5.5 Protótipos

## 5.6 Desenvolvimento Front-End

## 5.7 Desenvolvimento Back-End

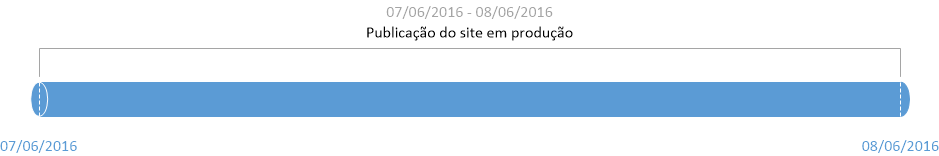
## 

## 5.8 Testes

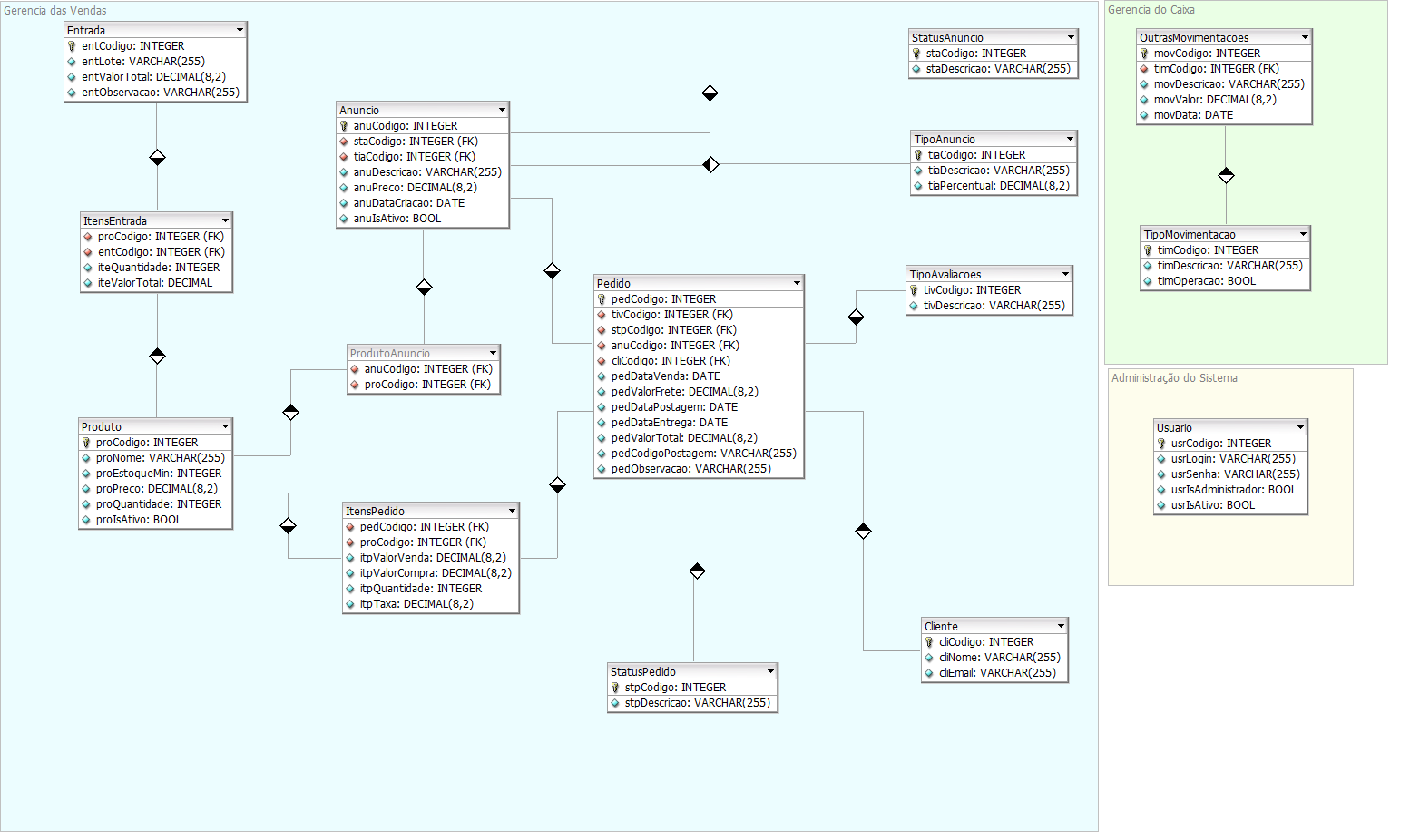


## 

## Implantação



**6. Modelagem de Dados**

**6.1 Modelo Relacional Lógico**

## 6.2 Projeto Físico do Banco de Dados

### **6.2.1 Criação das tabelas**

CREATE TABLE StatusPedido (

stpCodigo INTEGER AUTO\_INCREMENT,

stpDescricao VARCHAR(255) NOT NULL,

PRIMARY KEY(stpCodigo)

);

CREATE TABLE StatusAnuncio (

staCodigo INTEGER AUTO\_INCREMENT,

staDescricao VARCHAR(255) NOT NULL,

PRIMARY KEY(staCodigo)

);

CREATE TABLE Produto (

proCodigo INTEGER AUTO\_INCREMENT,

proNome VARCHAR(255) NOT NULL,

proEstoqueMin INTEGER UNSIGNED DEFAULT 0,

proPreco DECIMAL(8,2) NOT NULL,

proQuantidade INTEGER UNSIGNED DEFAULT 0,

proIsAtivo BOOL NOT NULL,

PRIMARY KEY(proCodigo)

);

CREATE TABLE Usuario (

usrCodigo INTEGER AUTO\_INCREMENT,

usrLogin VARCHAR(255) NOT NULL,

usrSenha VARCHAR(255) NOT NULL,

usrIsAdministrador BOOL NOT NULL,

usrIsAtivo BOOL NOT NULL,

PRIMARY KEY(usrCodigo)

);

CREATE TABLE TipoAvaliacoes (

tivCodigo INTEGER AUTO\_INCREMENT,

tivDescricao VARCHAR(255) NOT NULL,

PRIMARY KEY(tivCodigo)

);

CREATE TABLE TipoAnuncio (

tiaCodigo INTEGER AUTO\_INCREMENT,

tiaDescricao VARCHAR(255) NOT NULL,

tiaPercentual DECIMAL(8,2) NOT NULL,

PRIMARY KEY(tiaCodigo)

);

CREATE TABLE Cliente (

cliCodigo INTEGER AUTO\_INCREMENT,

cliNome VARCHAR(255) NOT NULL,

cliEmail VARCHAR(255) NOT NULL,

PRIMARY KEY(cliCodigo)

);

CREATE TABLE TipoMovimentacao (

timCodigo INTEGER AUTO\_INCREMENT,

timDescricao VARCHAR(255) NOT NULL,

timOperacao BOOL NOT NULL,

PRIMARY KEY(timCodigo)

);

CREATE TABLE Entrada (

entCodigo INTEGER AUTO\_INCREMENT,

entLote VARCHAR(255) NOT NULL,

entValorTotal DECIMAL(8,2) NOT NULL,

entObservacao VARCHAR(255),

PRIMARY KEY(entCodigo)

);

CREATE TABLE OutrasMovimentacoes (

movCodigo INTEGER AUTO\_INCREMENT,

timCodigo INTEGER NOT NULL,

movDescricao VARCHAR(255) NOT NULL,

movValor DECIMAL(8,2) NOT NULL,

movData DATE NOT NULL,

PRIMARY KEY(movCodigo),

FOREIGN KEY(timCodigo) REFERENCES TipoMovimentacao(timCodigo)

);

CREATE TABLE Anuncio (

anuCodigo INTEGER AUTO\_INCREMENT,

staCodigo INTEGER NOT NULL,

tiaCodigo INTEGER NOT NULL,

anuDescricao VARCHAR(255) NOT NULL,

anuPreco DECIMAL(8,2) NOT NULL,

anuDataCriacao DATE NOT NULL,

anuIsAtivo BOOL NOT NULL,

PRIMARY KEY(anuCodigo),

FOREIGN KEY(tiaCodigo) REFERENCES TipoAnuncio(tiaCodigo),

FOREIGN KEY(staCodigo) REFERENCES StatusAnuncio(staCodigo)

);

CREATE TABLE ItensEntrada (

proCodigo INTEGER NOT NULL,

entCodigo INTEGER NOT NULL,

iteQuantidade INTEGER UNSIGNED NOT NULL,

iteValorTotal DECIMAL(8,2),

FOREIGN KEY(entCodigo) REFERENCES Entrada(entCodigo),

FOREIGN KEY(proCodigo) REFERENCES Produto(proCodigo)

);

CREATE TABLE Pedido (

pedCodigo INTEGER AUTO\_INCREMENT,

tivCodigo INTEGER NOT NULL,

stpCodigo INTEGER NOT NULL,

anuCodigo INTEGER NOT NULL,

cliCodigo INTEGER NOT NULL,

pedDataVenda DATE NOT NULL,

pedValorFrete DECIMAL(8,2) NOT NULL DEFAULT 0,

pedDataPostagem DATE,

pedDataEntrega DATE,

pedValorTotal DECIMAL(8,2) NOT NULL,

pedCodigoPostagem VARCHAR(255),

pedObservacao VARCHAR(255),

PRIMARY KEY(pedCodigo),

FOREIGN KEY(cliCodigo) REFERENCES Cliente(cliCodigo),

FOREIGN KEY(anuCodigo) REFERENCES Anuncio(anuCodigo),

FOREIGN KEY(stpCodigo) REFERENCES StatusPedido(stpCodigo),

FOREIGN KEY(tivCodigo) REFERENCES TipoAvaliacoes(tivCodigo)

);

CREATE TABLE ItensPedido (

pedCodigo INTEGER NOT NULL,

proCodigo INTEGER NOT NULL,

itpValorVenda DECIMAL(8,2) NOT NULL,

itpValorCompra DECIMAL(8,2) NOT NULL,

itpQuantidade INTEGER NOT NULL,

itpTaxa DECIMAL(8,2) NOT NULL,

FOREIGN KEY(proCodigo) REFERENCES Produto(proCodigo),

FOREIGN KEY(pedCodigo) REFERENCES Pedido(pedCodigo)

);

CREATE TABLE ProdutoAnuncio(

anuCodigo INTEGER NOT NULL,

proCodigo INTEGER NOT NULL,

FOREIGN KEY(anuCodigo) REFERENCES Anuncio(anuCodigo),

FOREIGN KEY(proCodigo) REFERENCES Produto(proCodigo)

);

### **6.2.2 Procedures**

**-- STATUSPEDIDO**

DROP PROCEDURE IF EXISTS STATUSPEDIDO\_INSERT;

DELIMITER &&

CREATE PROCEDURE STATUSPEDIDO\_INSERT (

IN V\_STPDESCRICAO VARCHAR(255)

)

BEGIN

INSERT INTO STATUSPEDIDO (STPDESCRICAO ) VALUES (V\_STPDESCRICAO);

END

&&

DROP PROCEDURE IF EXISTS STATUSPEDIDO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE STATUSPEDIDO\_UPDATE (

IN V\_STPCODIGO INT,

V\_STPDESCRICAO VARCHAR(255)

)

BEGIN

UPDATE STATUSPEDIDO SET STPDESCRICAO = V\_STPDESCRICAO WHERE STPCODIGO = V\_STPCODIGO;

END

&&

DROP PROCEDURE IF EXISTS STATUSPEDIDO\_DELETE;

DELIMITER &&

CREATE PROCEDURE STATUSPEDIDO\_DELETE (

IN V\_STPCODIGO INT

)

BEGIN

DELETE FROM STATUSPEDIDO WHERE STPCODIGO = V\_STPCODIGO;

END

&&

DROP PROCEDURE IF EXISTS STATUSPEDIDO\_SELECT;

DELIMITER &&

CREATE PROCEDURE STATUSPEDIDO\_SELECT(

IN codigo INT(11),

IN descricao VARCHAR(150)

)

BEGIN

SELECT \* FROM statuspedido

WHERE stpCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN stpCodigo

WHEN codigo > 0 THEN codigo

END

AND stpDescricao LIKE

CASE

WHEN descricao = '' OR descricao IS NULL THEN stpDescricao

WHEN descricao <> '' THEN CONCAT('%', descricao, '%')

END;

END

&&

-- STATUSANUNCIO

DROP PROCEDURE IF EXISTS STATUSANUNCIO\_INSERT;

DELIMITER &&

CREATE PROCEDURE STATUSANUNCIO\_INSERT (

IN V\_STADESCRICAO VARCHAR(255)

)

BEGIN

INSERT INTO STATUSANUNCIO (STADESCRICAO ) VALUES (V\_STADESCRICAO);

END

&&

DROP PROCEDURE IF EXISTS STATUSANUNCIO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE STATUSANUNCIO\_UPDATE (

IN V\_STACODIGO INT,

V\_STADESCRICAO VARCHAR(255)

)

BEGIN

UPDATE STATUSANUNCIO SET STADESCRICAO = V\_STADESCRICAO WHERE STACODIGO = V\_STACODIGO;

END

&&

DROP PROCEDURE IF EXISTS STATUSANUNCIO\_DELETE;

DELIMITER &&

CREATE PROCEDURE STATUSANUNCIO\_DELETE (

IN V\_STACODIGO INT

)

BEGIN

DELETE FROM STATUSANUNCIO WHERE STACODIGO = V\_STACODIGO;

END

&&

DROP PROCEDURE IF EXISTS STATUSANUNCIO\_SELECT;

DELIMITER &&

CREATE PROCEDURE STATUSANUNCIO\_SELECT(

IN codigo INT(11),

IN descricao VARCHAR(150)

)

BEGIN

SELECT \* FROM statusanuncio

WHERE staCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN staCodigo

WHEN codigo > 0 THEN codigo

END

AND staDescricao LIKE

CASE

WHEN descricao = '' OR descricao IS NULL THEN staDescricao

WHEN descricao <> '' THEN CONCAT('%', descricao, '%')

END;

END

&&

**-- PRODUTO**

DROP PROCEDURE IF EXISTS PRODUTO\_INSERT;

DELIMITER &&

CREATE PROCEDURE PRODUTO\_INSERT (

IN V\_PRONOME VARCHAR(255),

V\_PROESTOQUEMIN INT,

V\_PROPRECO DECIMAL(8,2),

V\_PROISATIVO BOOL

)

BEGIN

INSERT INTO PRODUTO (PRONOME, PROESTOQUEMIN, PROPRECO, PROQUANTIDADE, PROISATIVO) VALUES (V\_PRONOME, V\_PROESTOQUEMIN, V\_PROPRECO, V\_PROQUANTIDADE, V\_PROISATIVO);

END

&&

DROP PROCEDURE IF EXISTS PRODUTO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE PRODUTO\_UPDATE (

IN

V\_PROCODIGO INT,

V\_PRONOME VARCHAR(255),

V\_PROESTOQUEMIN INT,

V\_PROPRECO DECIMAL(8,2),

V\_PROISATIVO BOOL

)

BEGIN

UPDATE PRODUTO SET PRONOME = V\_PRONOME, PROESTOQUEMIN = V\_PROESTOQUEMIN, PROPRECO = V\_PROPRECO, PROISATIVO = V\_PROISATIVO WHERE PROCODIGO = V\_PROCODIGO;

END

&&

DROP PROCEDURE IF EXISTS PRODUTO\_DELETE;

DELIMITER &&

CREATE PROCEDURE PRODUTO\_DELETE (

IN V\_PROCODIGO INT

)

BEGIN

DELETE FROM PRODUTO WHERE PROCODIGO = V\_PROCODIGO;

END

&&

DROP PROCEDURE IF EXISTS PRODUTO\_SELECT;

DELIMITER &&

CREATE PROCEDURE PRODUTO\_SELECT(

IN codigo INT(11),

IN nome VARCHAR(150),

IN preco DECIMAL(8,2),

IN quantidade INT

)

BEGIN

SELECT \* FROM PRODUTO

WHERE proCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN proCodigo

WHEN codigo > 0 THEN codigo

END

AND proNome LIKE

CASE

WHEN nome = '' OR nome IS NULL THEN proNome

WHEN nome <> '' THEN CONCAT('%', nome, '%')

END

AND proPreco LIKE

CASE

WHEN preco = 0 OR preco IS NULL THEN proPreco

WHEN preco > 0 THEN preco

END

AND proQuantidade LIKE

CASE

WHEN quantidade = 0 OR quantidade IS NULL THEN proQuantidade

WHEN quantidade > 0 THEN quantidade

END;

END

&&

-- USUARIO

DROP PROCEDURE IF EXISTS USUARIO\_INSERT;

DELIMITER &&

CREATE PROCEDURE USUARIO\_INSERT (

IN V\_USRLOGIN VARCHAR(255),

v\_USRSENHA VARCHAR(255),

V\_USRISADMINISTRADOR BOOL,

V\_USRISATIVO BOOL

)

BEGIN

INSERT INTO USUARIO (USRLOGIN,USRSENHA,USRISADMINISTRADOR,USRISATIVO)

VALUES (V\_USRLOGIN,V\_USRSENHA,V\_USRISADMINISTRADOR,V\_USRISATIVO);

END

&&

DROP PROCEDURE IF EXISTS USUARIO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE USUARIO\_UPDATE (

IN V\_USRCODIGO INT,

V\_USRLOGIN VARCHAR(255),

V\_USRSENHA VARCHAR(255),

V\_USRISADMINISTRADOR BOOL,

V\_USRISATIVO BOOL

)

BEGIN

UPDATE USUARIO SET USRLOGIN = V\_USRLOGIN, USRSENHA = V\_USRSENHA, USRISADMINISTRADOR = V\_USRISADMINISTRADOR, USRISATIVO = V\_USRISATIVO

WHERE USRCODIGO = V\_USRCODIGO;

END

&&

DROP PROCEDURE IF EXISTS USUARIO\_DELETE;

DELIMITER &&

CREATE PROCEDURE USUARIO\_DELETE (

IN V\_USRCODIGO INT

)

BEGIN

DELETE FROM USUARIO WHERE USRCODIGO = V\_USRCODIGO;

END

&&

DROP PROCEDURE IF EXISTS USUARIO\_SELECT;

DELIMITER &&

CREATE PROCEDURE USUARIO\_SELECT(

IN codigo INT(11),

IN login VARCHAR(150)

)

BEGIN

SELECT \* FROM USUARIO

WHERE usrCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN usrCodigo

WHEN codigo > 0 THEN codigo

END

AND usrLogin LIKE

CASE

WHEN login = '' OR login IS NULL THEN usrLogin

WHEN login <> '' THEN CONCAT('%', login, '%')

END;

END

&&

-- TIPOAVALIACOES

DROP PROCEDURE IF EXISTS TIPOAVALIACOES\_INSERT;

DELIMITER &&

CREATE PROCEDURE TIPOAVALIACOES\_INSERT (

IN V\_TIVDESCRICAO VARCHAR(255)

)

BEGIN

INSERT INTO TIPOAVALIACOES (TIVDESCRICAO ) VALUES (V\_TIVDESCRICAO);

END

&&

DROP PROCEDURE IF EXISTS TIPOAVALIACOES\_UPDATE;

DELIMITER &&

CREATE PROCEDURE TIPOAVALIACOES\_UPDATE (

IN V\_TIVCODIGO INT,

V\_TIVDESCRICAO VARCHAR(255)

)

BEGIN

UPDATE TIPOAVALIACOES SET TIVDESCRICAO = V\_TIVDESCRICAO WHERE TIVCODIGO = V\_TIVCODIGO;

END

&&

DROP PROCEDURE IF EXISTS TIPOAVALIACOES\_DELETE;

DELIMITER &&

CREATE PROCEDURE TIPOAVALIACOES\_DELETE (

IN V\_TIVCODIGO INT

)

BEGIN

DELETE FROM TIPOAVALIACOES WHERE TIVCODIGO = V\_TIVCODIGO;

END

&&

DROP PROCEDURE IF EXISTS TIPOAVALIACAO\_SELECT;

DELIMITER &&

CREATE PROCEDURE TIPOAVALIACAO\_SELECT(

IN codigo INT(11),

IN descricao VARCHAR(150)

)

BEGIN

SELECT \* FROM tipoavaliacoes

WHERE tivCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN tivCodigo

WHEN codigo > 0 THEN codigo

END

AND tivDescricao LIKE

CASE

WHEN descricao = '' OR descricao IS NULL THEN tivDescricao

WHEN descricao <> '' THEN CONCAT('%', descricao, '%')

END;

END

&&

**-- TIPOANUNCIO**

DROP PROCEDURE IF EXISTS TIPOANUNCIO\_INSERT;

DELIMITER &&

CREATE PROCEDURE TIPOANUNCIO\_INSERT (

IN V\_TIADESCRICAO VARCHAR(255),

V\_TIAPERCENTUAL DECIMAL(8,2)

)

BEGIN

INSERT INTO TIPOANUNCIO (TIADESCRICAO, TIAPERCENTUAL) VALUES (V\_TIADESCRICAO, V\_TIAPERCENTUAL);

END

&&

DROP PROCEDURE IF EXISTS TIPOANUNCIO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE TIPOANUNCIO\_UPDATE (

IN V\_TIACODIGO INT,

V\_TIADESCRICAO VARCHAR(255),

V\_TIAPERCENTUAL DECIMAL(8,2)

)

BEGIN

UPDATE TIPOANUNCIO SET TIADESCRICAO = V\_TIADESCRICAO, TIAPERCENTUAL = V\_TIAPERCENTUAL WHERE TIACODIGO = V\_TIACODIGO;

END

&&

DROP PROCEDURE IF EXISTS TIPOANUNCIO\_DELETE;

DELIMITER &&

CREATE PROCEDURE TIPOANUNCIO\_DELETE (

IN V\_TIACODIGO INT

)

BEGIN

DELETE FROM TIPOANUNCIO WHERE TIACODIGO = V\_TIACODIGO;

END

&&

DROP PROCEDURE IF EXISTS TIPOANUNCIO\_SELECT;

DELIMITER &&

CREATE PROCEDURE TIPOANUNCIO\_SELECT(

IN codigo INT(11),

IN descricao VARCHAR(150)

)

BEGIN

SELECT \* FROM TIPOANUNCIO

WHERE tiaCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN tiaCodigo

WHEN codigo > 0 THEN codigo

END

AND tiaDescricao LIKE

CASE

WHEN descricao = '' OR descricao IS NULL THEN tiaDescricao

WHEN descricao <> '' THEN CONCAT('%', descricao, '%')

END;

END

&&

**-- CLIENTE**

DROP PROCEDURE IF EXISTS CLIENTE\_INSERT;

DELIMITER &&

CREATE PROCEDURE CLIENTE\_INSERT (

IN V\_CLINOME VARCHAR(255),

V\_CLIEMAIL VARCHAR(255)

)

BEGIN

INSERT INTO CLIENTE (CLINOME, CLIEMAIL) VALUES (V\_CLINOME, V\_CLIEMAIL);

END

&&

DROP PROCEDURE IF EXISTS CLIENTE\_UPDATE;

DELIMITER &&

CREATE PROCEDURE CLIENTE\_UPDATE (

IN V\_CLICODIGO INT,

V\_CLINOME VARCHAR(255),

V\_CLIEMAIL VARCHAR(255)

)

BEGIN

UPDATE CLIENTE SET CLINOME = V\_CLINOME, CLIEMAIL = V\_CLIEMAIL WHERE CLICODIGO = V\_CLICODIGO;

END

&&

DROP PROCEDURE IF EXISTS CLIENTE\_DELETE;

DELIMITER &&

CREATE PROCEDURE CLIENTE\_DELETE (

IN V\_CLICODIGO INT

)

BEGIN

DELETE FROM CLIENTE WHERE CLICODIGO = V\_CLICODIGO;

END

&&

DROP PROCEDURE IF EXISTS CLIENTE\_SELECT;

DELIMITER &&

CREATE PROCEDURE CLIENTE\_SELECT(

IN codigo INT(11),

IN nome VARCHAR(150),

IN email VARCHAR(150)

)

BEGIN

SELECT \* FROM cliente

WHERE cliCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN cliCodigo

WHEN codigo > 0 THEN codigo

END

AND cliNome LIKE

CASE

WHEN nome = '' OR nome IS NULL THEN cliNome

WHEN nome <> '' THEN CONCAT('%', nome, '%')

END

AND cliEmail LIKE

CASE

WHEN email = '' OR email IS NULL THEN cliEmail

WHEN email <> '' THEN CONCAT('%', email, '%')

END

ORDER BY cliNome;

END

&&

**-- TIPOMOVIMENTACAO**

DROP PROCEDURE IF EXISTS TIPOMOVIMENTACAO\_INSERT;

DELIMITER &&

CREATE PROCEDURE TIPOMOVIMENTACAO\_INSERT (

IN V\_TIMDESCRICAO VARCHAR(255),

V\_OPERACAO BOOL

)

BEGIN

INSERT INTO TIPOMOVIMENTACAO (TIMDESCRICAO,TIMOPERACAO)

VALUES (V\_TIMDESCRICAO,V\_OPERACAO);

END

&&

DROP PROCEDURE IF EXISTS TIPOMOVIMENTACAO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE TIPOMOVIMENTACAO\_UPDATE (

IN V\_TIMCODIGO INT,

V\_TIMDESCRICAO VARCHAR(255),

V\_TIMOPERACAO BOOL

)

BEGIN

UPDATE TIPOMOVIMENTACAO SET TIMDESCRICAO = V\_TIMDESCRICAO, TIMOPERACAO = V\_TIMOPERACAO

WHERE TIMCODIGO = V\_TIMCODIGO;

END

&&

DROP PROCEDURE IF EXISTS TIPOMOVIMENTACAO\_DELETE;

DELIMITER &&

CREATE PROCEDURE TIPOMOVIMENTACAO\_DELETE (

IN V\_TIMCODIGO INT

)

BEGIN

DELETE FROM TIPOMOVIMENTACAO WHERE TIMCODIGO = V\_TIMCODIGO;

END

&&

DROP PROCEDURE IF EXISTS TIPOMOVIMENTACAO\_SELECT;

DELIMITER &&

CREATE PROCEDURE TIPOMOVIMENTACAO\_SELECT(

IN codigo INT(11),

IN descricao VARCHAR(150)

)

BEGIN

SELECT \* FROM TIPOMOVIMENTACAO

WHERE timCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN timCodigo

WHEN codigo > 0 THEN codigo

END

AND timDescricao LIKE

CASE

WHEN descricao = '' OR descricao IS NULL THEN timDescricao

WHEN descricao <> '' THEN CONCAT('%', descricao, '%')

END;

END

&&

**-- ENTRADA--**

DROP PROCEDURE IF EXISTS ENTRADA\_INSERT;

DELIMITER &&

CREATE PROCEDURE ENTRADA\_INSERT (

IN

V\_ENTLOTE VARCHAR(255),

V\_ENTVALORTOTAL DECIMAL(8,2),

V\_ENTOBSERVACAO VARCHAR(255)

)

BEGIN

INSERT INTO ENTRADA (ENTLOTE, ENTVALORTOTAL,ENTOBSERVACAO) VALUES (V\_ENTLOTE, V\_ENTVALORTOTAL,V\_ENTOBSERVACAO);

END

&&

DROP PROCEDURE IF EXISTS ENTRADA\_UPDATE;

DELIMITER &&

CREATE PROCEDURE ENTRADA\_UPDATE (

IN V\_ENTCODIGO INT,

V\_ENTLOTE VARCHAR(255),

V\_VALORTOTAL DECIMAL(8,2),

V\_ENTOBSERVACAO VARCHAR(255)

)

BEGIN

UPDATE ENTRADA SET ENTLOTE = V\_ENTLOTE, ENTVALORTOTAL = V\_VALORTOTAL,ENTOBSERVACAO= V\_ENTOBSERVACAO WHERE ENTCODIGO = V\_ENTCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ENTRADA\_DELETE;

DELIMITER &&

CREATE PROCEDURE ENTRADA\_DELETE (

IN V\_ENTCODIGO INT

)

BEGIN

DELETE FROM ENTRADA WHERE ENTCODIGO = V\_ENTCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ENTRADA\_SELECT;

DELIMITER &&

CREATE PROCEDURE ENTRADA\_SELECT(

IN codigo INT(11),

IN lote varchar(255),

IN valorTotal DECIMAL(8,2),

IN observacao VARCHAR (255)

)

BEGIN

SELECT \*

FROM entrada

WHERE entCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN entCodigo

WHEN codigo > 0 THEN codigo

END

AND entLote LIKE

CASE

WHEN lote = '' OR lote IS NULL THEN entLote

WHEN lote <> '' THEN CONCAT('%', lote, '%')

END

AND entValorTotal =

CASE

WHEN valorTotal = 0 OR valorTotal IS NULL THEN entValorTotal

WHEN valorTotal > 0 THEN valorTotal

END

AND entObservacao LIKE

CASE

WHEN observacao = '' OR observacao IS NULL THEN entObservacao

WHEN observacao <> '' THEN CONCAT('%', observacao, '%')

END;

END

&&

**-- OUTRASMOVIMENTAÇOES**

DROP PROCEDURE IF EXISTS OUTRASMOVIMENTACOES\_INSERT;

DELIMITER &&

CREATE PROCEDURE OUTRASMOVIMENTACOES\_INSERT (

IN V\_TIMCODIGO INT,

V\_MOVDESCRICAO VARCHAR(255),

V\_MOVVALOR DECIMAL(8,2),

V\_MOVDATA DATE

)

BEGIN

INSERT INTO OUTRASMOVIMENTACOES ( TIMCODIGO, MOVDESCRICAO,MOVVALOR,MOVDATA)

VALUES (V\_TIMCODIGO, V\_MOVDESCRICAO,V\_MOVVALOR,V\_MOVDATA);

END

&&

DROP PROCEDURE IF EXISTS OUTRASMOVIMENTACOES\_UPDATE;

DELIMITER &&

CREATE PROCEDURE OUTRASMOVIMENTACOES\_UPDATE (

IN V\_MOVCODIGO INT,

V\_TIMCODIGO INT,

V\_MOVDESCRICAO VARCHAR(255),

V\_MOVVALOR DECIMAL(8,2),

V\_MOVDATA DATE

)

BEGIN

UPDATE OUTRASMOVIMENTACOES SET TIMCODIGO = V\_TIMCODIGO, MOVDESCRICAO = V\_MOVDESCRICAO, MOVVALOR = V\_MOVVALOR, MOVDATA = V\_MOVDATA

WHERE MOVCODIGO = V\_MOVCODIGO;

END

&&

DROP PROCEDURE IF EXISTS OUTRASMOVIMENTACOES\_DESCRICAO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE OUTRASMOVIMENTACOES\_DESCRICAO\_UPDATE (

IN V\_MOVDESCRICAO VARCHAR(255),

V\_MOVVALOR DECIMAL(8,2),

V\_MOVDATA DATE

)

BEGIN

UPDATE OUTRASMOVIMENTACOES SET MOVVALOR = V\_MOVVALOR, MOVDATA = V\_MOVDATA

WHERE MOVDESCRICAO = V\_MOVDESCRICAO;

END

&&

DROP PROCEDURE IF EXISTS OUTRASMOVIMENTACOES\_DELETE;

DELIMITER &&

CREATE PROCEDURE OUTRASMOVIMENTACOES\_DELETE (

IN V\_MOVCODIGO INT

)

BEGIN

DELETE FROM OUTRASMOVIMENTACOES WHERE MOVCODIGO = V\_MOVCODIGO;

END

&&

DROP PROCEDURE IF EXISTS OUTRASMOVIMENTACOES\_DESCRICAO\_DELETE;

DELIMITER &&

CREATE PROCEDURE OUTRASMOVIMENTACOES\_DESCRICAO\_DELETE (

IN V\_MOVDESCRICAO VARCHAR(255)

)

BEGIN

DELETE FROM OUTRASMOVIMENTACOES WHERE MOVDESCRICAO = V\_MOVDESCRICAO;

END

&&

DROP PROCEDURE IF EXISTS OUTRASMOVIMENTACOES\_SELECT;

DELIMITER &&

CREATE PROCEDURE OUTRASMOVIMENTACOES\_SELECT (

IN codigo INT,

IN descricao VARCHAR(255),

IN valor DECIMAL(8,2),

IN data DATE,

IN tipoMovimentacao VARCHAR(255)

)

BEGIN

SELECT movCodigo, movDescricao, movValor, movData, outrasmovimentacoes.timCodigo, timDescricao

FROM outrasmovimentacoes

INNER JOIN tipoMovimentacao

ON outrasMovimentacoes.timCodigo = tipoMovimentacao.timCodigo

WHERE movCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN movCodigo

WHEN codigo > 0 THEN codigo

END

AND movDescricao LIKE

CASE

WHEN descricao = '' OR descricao IS NULL THEN movDescricao

WHEN descricao <> '' THEN CONCAT('%', descricao, '%')

END

AND movValor LIKE

CASE

WHEN valor = 0 OR valor IS NULL THEN movValor

WHEN valor > 0 THEN valor

END

AND movData LIKE

CASE

WHEN data = '' OR data IS NULL THEN movData

WHEN data <> '' THEN CONCAT('%', data, '%')

END

AND timDescricao =

CASE

WHEN tipoMovimentacao = '' OR tipoMovimentacao IS NULL THEN timDescricao

WHEN tipoMovimentacao <> '' THEN tipoMovimentacao

END;

END

&&

**-- ANUNCIO**

DROP PROCEDURE IF EXISTS ANUNCIO\_INSERT;

DELIMITER &&

CREATE PROCEDURE ANUNCIO\_INSERT (

IN V\_STACODIGO INT,

v\_TIACODIGO INT,

V\_ANUDESCRICAO VARCHAR(255),

V\_ANUPRECO DECIMAL(8,2),

V\_ANUDATACRIACAO DATE,

V\_ANUISATIVO BOOL

)

BEGIN

INSERT INTO ANUNCIO (STACODIGO,TIACODIGO,ANUDESCRICAO,ANUPRECO,ANUDATACRIACAO,ANUISATIVO) VALUES (V\_STACODIGO, V\_TIACODIGO, V\_ANUDESCRICAO, V\_ANUPRECO, V\_ANUDATACRIACAO, V\_ANUISATIVO);

END

&&

DROP PROCEDURE IF EXISTS ANUNCIO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE ANUNCIO\_UPDATE (

IN V\_ANUCODIGO INT,

V\_STACODIGO INT,

V\_TIACODIGO INT,

V\_ANUDESCRICAO VARCHAR(255),

V\_ANUPRECO DECIMAL(8,2),

V\_ANUISATIVO BOOL

)

BEGIN

UPDATE ANUNCIO SET STACODIGO = V\_STACODIGO, TIACODIGO = V\_TIACODIGO,ANUDESCRICAO = V\_ANUDESCRICAO, ANUPRECO = V\_ANUPRECO, ANUISATIVO = V\_ANUISATIVO

WHERE ANUCODIGO = V\_ANUCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ANUNCIO\_DELETE;

DELIMITER &&

CREATE PROCEDURE ANUNCIO\_DELETE (

IN V\_ANUCODIGO INT

)

BEGIN

DELETE FROM ANUNCIO WHERE ANUCODIGO = V\_ANUCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ANUNCIO\_SELECT;

DELIMITER &&

CREATE PROCEDURE ANUNCIO\_SELECT (

IN codigo INT,

IN descricao VARCHAR(255),

IN preco DECIMAL(8,2),

IN dataCriacao DATE,

IN tipoAnuncio VARCHAR(255),

IN statusAnuncio VARCHAR(255)

)

BEGIN

SELECT a.anuCodigo, s.staDescricao, t.tiaDescricao, a.anuDescricao, a.anuPreco, a.anuDataCriacao, s.staCodigo, t.tiaCodigo

FROM anuncio AS a

INNER JOIN statusAnuncio AS s ON a.staCodigo = s.staCodigo

INNER JOIN tipoanuncio AS t on a.tiaCodigo = t.tiaCodigo

WHERE a.anuCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN a.anuCodigo

WHEN codigo > 0 THEN codigo

END

AND a.anuDescricao LIKE

CASE

WHEN descricao = '' OR descricao IS NULL THEN a.anuDescricao

WHEN descricao <> '' THEN CONCAT('%', descricao, '%')

END

AND a.anuPreco LIKE

CASE

WHEN preco = 0 OR preco IS NULL THEN a.anuPreco

WHEN preco > 0 THEN preco

END

AND a.anuDataCriacao LIKE

CASE

WHEN dataCriacao = '' OR dataCriacao IS NULL THEN a.anuDataCriacao

WHEN dataCriacao <> '' THEN CONCAT('%', dataCriacao, '%')

END

AND s.staDescricao =

CASE

WHEN statusAnuncio = '' OR statusAnuncio IS NULL THEN s.staDescricao

WHEN statusAnuncio <> '' THEN statusAnuncio

END

AND t.tiaDescricao =

CASE

WHEN tipoAnuncio = '' OR tipoAnuncio IS NULL THEN t.tiaDescricao

WHEN tipoAnuncio <> '' THEN tipoAnuncio

END

ORDER BY a.anuDescricao;

END

&&

**-- ITENSENTRADA**

DROP PROCEDURE IF EXISTS ITENSENTRADA\_INSERT;

DELIMITER &&

CREATE PROCEDURE ITENSENTRADA\_INSERT (

IN V\_PROCODIGO INT,

V\_ENTCODIGO INT,

V\_ITEQUANTIDADE INT,

V\_ITEVALORTOTAL DECIMAL(8,2)

)

BEGIN

INSERT INTO ITENSENTRADA (PROCODIGO,ENTCODIGO,ITEQUANTIDADE,ITEVALORTOTAL)

VALUES (V\_PROCODIGO,V\_ENTCODIGO,V\_ITEQUANTIDADE,V\_ITEVALORTOTAL);

END

&&

DROP PROCEDURE IF EXISTS ITENSENTRADA\_UPDATE;

DELIMITER &&

CREATE PROCEDURE ITENSENTRADA\_UPDATE (

IN V\_PROCODIGO INT,

V\_ENTCODIGO INT,

V\_ITEQUANTIDADE INT,

V\_ITEVALORTOTAL DECIMAL(8,2)

)

BEGIN

UPDATE ITENSENTRADA SET PROCODIGO = V\_PROCODIGO, ENTCODIGO = V\_ENTCODIGO,ITEQUANTIDADE = V\_ITEQUANTIDADE, ITEVALORTOTAL = V\_ITEVALORTOTAL WHERE PROCODIGO = V\_PROCODIGO AND ENTCODIGO = V\_ENTCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ITENSENTRADA\_DELETE;

DELIMITER &&

CREATE PROCEDURE ITENSENTRADA\_DELETE (

IN V\_ENTCODIGO INT

)

BEGIN

DELETE FROM ITENSENTRADA WHERE ENTCODIGO = V\_ENTCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ITENSENTRADA\_SELECT;

DELIMITER &&

CREATE PROCEDURE ITENSENTRADA\_SELECT(

IN codigo INT(11)

)

BEGIN

SELECT itensEntrada.\*, produto.proNome, produto.proPreco

FROM itensEntrada INNER JOIN produto on itensEntrada.proCodigo = produto.proCodigo

WHERE entCodigo = codigo;

END

&&

**-- PEDIDO**

DROP PROCEDURE IF EXISTS PEDIDO\_INSERT;

DELIMITER &&

CREATE PROCEDURE PEDIDO\_INSERT (

IN V\_TIVCODIGO INT,

V\_STPCODIGO INT,

V\_ANUCODIGO INT,

V\_CLICODIGO INT,

V\_PEDDATAVENDA DATE,

V\_PEDVALORFRETE DECIMAL(8,2),

V\_PEDDATAPOSTAGEM DATE,

V\_PEDDATAENTREGA DATE,

V\_PEDVALORTOTAL DECIMAL(8,2),

V\_PEDCODIGOPOSTAGEM VARCHAR(255),

V\_PEDOBSERVACAO VARCHAR(255)

)

BEGIN

INSERT INTO PEDIDO (TIVCODIGO, STPCODIGO, ANUCODIGO, CLICODIGO, PEDDATAVENDA, PEDVALORFRETE, PEDDATAPOSTAGEM, PEDDATAENTREGA, PEDVALORTOTAL, PEDCODIGOPOSTAGEM, PEDOBSERVACAO) VALUES (V\_TIVCODIGO, V\_STPCODIGO, V\_ANUCODIGO, V\_CLICODIGO, V\_PEDDATAVENDA, V\_PEDVALORFRETE, V\_PEDDATAPOSTAGEM, V\_PEDDATAENTREGA, V\_PEDVALORTOTAL, V\_PEDCODIGOPOSTAGEM, V\_PEDOBSERVACAO);

END

&&

DROP PROCEDURE IF EXISTS PEDIDO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE PEDIDO\_UPDATE (

IN V\_PEDCODIGO INT,

V\_TIVCODIGO INT,

V\_STPCODIGO INT,

V\_ANUCODIGO INT,

V\_CLICODIGO INT,

V\_PEDDATAVENDA DATE,

V\_PEDVALORFRETE DECIMAL(8,2),

V\_PEDDATAPOSTAGEM DATE,

V\_PEDDATAENTREGA DATE,

V\_PEDVALORTOTAL DECIMAL(8,2),

V\_PEDCODIGOPOSTAGEM VARCHAR(255),

V\_PEDOBSERVACAO VARCHAR(255)

)

BEGIN

UPDATE PEDIDO SET TIVCODIGO = V\_TIVCODIGO, STPCODIGO = V\_STPCODIGO, CLICODIGO = V\_CLICODIGO, PEDDATAVENDA = V\_PEDDATAVENDA, PEDVALORFRETE = V\_PEDVALORFRETE, PEDDATAPOSTAGEM = V\_PEDDATAPOSTAGEM, PEDDATAENTREGA = V\_PEDDATAENTREGA, PEDVALORTOTAL = V\_PEDVALORTOTAL, PEDCODIGOPOSTAGEM = V\_PEDCODIGOPOSTAGEM, PEDOBSERVACAO = V\_PEDOBSERVACAO WHERE PEDCODIGO = V\_PEDCODIGO;

END

&&

DROP PROCEDURE IF EXISTS PEDIDO\_DELETE;

DELIMITER &&

CREATE PROCEDURE PEDIDO\_DELETE (

IN V\_PEDCODIGO INT

)

BEGIN

DELETE FROM PEDIDO WHERE PEDCODIGO = V\_PEDCODIGO;

END

&&

DROP PROCEDURE IF EXISTS PEDIDO\_SELECT;

DELIMITER &&

CREATE PROCEDURE PEDIDO\_SELECT(

IN codigo INT(11),

IN dataVenda date,

IN dataPostagem date,

IN dataEntrega date,

IN statusPedido VARCHAR(255),

IN tipoAvaliacao VARCHAR(255),

IN cliente VARCHAR (255),

IN email VARCHAR (255),

IN anuncio VARCHAR (255)

)

BEGIN

SELECT

ped.pedCodigo,

tiv.tivDescricao,

stp.stpDescricao,

anu.anuDescricao,

cli.cliNome,

cli.cliEmail,

ped.pedDataVenda,

ped.pedValorFrete,

ped.pedDataPostagem,

ped.pedDataEntrega,

ped.pedValorTotal,

ped.pedCodigoPostagem,

ped.pedObservacao,

tiv.tivCodigo,

stp.stpCodigo,

anu.anuCodigo,

cli.cliCodigo

FROM pedido as ped

INNER JOIN tipoAvaliacoes as tiv ON ped.tivCodigo = tiv.TivCodigo

INNER JOIN anuncio as anu ON ped.anuCodigo = anu.anuCodigo

INNER JOIN cliente as cli ON ped.cliCodigo = cli.cliCodigo

INNER JOIN statusPedido as stp ON ped.stpCodigo = stp.stpCodigo

WHERE ped.pedCodigo =

CASE

WHEN codigo = 0 OR codigo IS NULL THEN ped.pedCodigo

WHEN codigo > 0 THEN codigo

END

AND ((ped.pedDataVenda =

CASE

WHEN dataVenda = '' OR dataVenda IS NULL THEN ped.pedDataVenda

WHEN dataVenda <> '' THEN dataVenda

END)

OR (ped.pedDataVenda IS NULL))

AND ((ped.pedDataPostagem =

CASE

WHEN dataPostagem = '' OR dataPostagem IS NULL THEN ped.pedDataPostagem

WHEN dataPostagem <> '' THEN dataPostagem

END)

OR (ped.pedDataPostagem IS NULL))

AND ((ped.pedDataEntrega =

CASE

WHEN dataEntrega = '' OR dataEntrega IS NULL THEN ped.pedDataEntrega

WHEN dataEntrega <> '' THEN dataEntrega

END)

OR (ped.pedDataEntrega IS NULL))

AND stp.stpDescricao =

CASE

WHEN statusPedido = '' OR statusPedido IS NULL THEN stp.stpDescricao

WHEN statusPedido <> '' THEN statusPedido

END

AND tiv.tivDescricao =

CASE

WHEN tipoAvaliacao = '' OR tipoAvaliacao IS NULL THEN tiv.tivDescricao

WHEN tipoAvaliacao <> '' THEN tipoAvaliacao

END

AND cli.cliNome LIKE

CASE

WHEN cliente = '' OR cliente IS NULL THEN cli.cliNome

WHEN cliente <> '' THEN CONCAT('%', cliente, '%')

END

AND cli.cliEmail LIKE

CASE

WHEN email = '' OR email IS NULL THEN cli.cliEmail

WHEN email <> '' THEN CONCAT('%', email, '%')

END

AND anu.anuDescricao LIKE

CASE

WHEN anuncio = '' OR anuncio IS NULL THEN anu.anuDescricao

WHEN anuncio <> '' THEN CONCAT('%', anuncio, '%')

END;

END

&&

**-- ITENSPEDIDO**

DROP PROCEDURE IF EXISTS ITENSPEDIDO\_INSERT;

DELIMITER &&

CREATE PROCEDURE ITENSPEDIDO\_INSERT (

IN V\_PEDCODIGO INT,

V\_PROCODIGO INT,

V\_ITPVALORVENDA DECIMAL(8,2),

V\_ITPVALORCOMPRA DECIMAL(8,2),

V\_ITPQUANTIDADE INT,

V\_ITPTAXA DECIMAL(8,2)

)

BEGIN

INSERT INTO ITENSPEDIDO (PEDCODIGO,PROCODIGO,ITPVALORVENDA,ITPVALORCOMPRA,ITPQUANTIDADE,ITPTAXA) VALUES (V\_PEDCODIGO, V\_PROCODIGO, V\_ITPVALORVENDA,

V\_ITPVALORCOMPRA,V\_ITPQUANTIDADE,V\_ITPTAXA);

END

&&

DROP PROCEDURE IF EXISTS ITENSPEDIDO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE ITENSPEDIDO\_UPDATE (

IN V\_PEDCODIGO INT,

V\_PROCODIGO INT,

V\_ITPVALORVENDA DECIMAL(8,2),

V\_ITPVALORCOMPRA DECIMAL(8,2),

V\_ITPTAXA DECIMAL(8,2)

)

BEGIN

UPDATE ITENSPEDIDO SET PEDCODIGO = V\_PEDCODIGO, PROCODIGO = V\_PROCODIGO, ITPVALORVENDA = V\_ITPVALORVENDA, ITPVALORCOMPRA = V\_ITPVALORCOMPRA,

ITPTAXA = V\_ITPTAXA WHERE PEDCODIGO = V\_PEDCODIGO AND PROCODIGO = V\_PROCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ITENSPEDIDO\_DELETE;

DELIMITER &&

CREATE PROCEDURE ITENSPEDIDO\_DELETE (

IN V\_PEDCODIGO INT

)

BEGIN

DELETE FROM ITENSPEDIDO WHERE PEDCODIGO = V\_PEDCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ITENSPEDIDO\_SELECT;

DELIMITER &&

CREATE PROCEDURE ITENSPEDIDO\_SELECT(

IN codigo INT(11)

)

BEGIN

SELECT itensPedido.\*, produto.proNome, (itensPedido.itpValorVenda \* itensPedido.itpQuantidade) as valorTotal

FROM itensPedido INNER JOIN produto on itensPedido.proCodigo = produto.proCodigo

WHERE pedCodigo = codigo;

END

&&

**--PRODUTO ANÚNCIO**

DROP PROCEDURE IF EXISTS ANUNCIOPRODUTO\_INSERT;

DELIMITER &&

CREATE PROCEDURE ANUNCIOPRODUTO\_INSERT (

IN V\_PROCODIGO INT,

V\_ANUCODIGO INT

)

BEGIN

INSERT INTO PRODUTOANUNCIO (ANUCODIGO, PROCODIGO) VALUES (V\_PROCODIGO, V\_ANUCODIGO);

END

&&

DROP PROCEDURE IF EXISTS ANUNCIOPRODUTO\_DELETE;

DELIMITER &&

CREATE PROCEDURE ANUNCIOPRODUTO\_DELETE (

IN V\_ANUCODIGO INT

)

BEGIN

DELETE FROM PRODUTOANUNCIO WHERE ANUCODIGO = V\_ANUCODIGO;

END

&&

DROP PROCEDURE IF EXISTS ANUNCIOPRODUTO\_SELECT;

DELIMITER &&

CREATE PROCEDURE ANUNCIOPRODUTO\_SELECT(

IN codigo INT(11)

)

BEGIN

SELECT pro.proCodigo, pro.proNome, pro.proPreco, anu.anuPreco, ROUND((tip.tiaPercentual \* anu.anuPreco),2) as taxa

FROM tipoanuncio as tip

INNER JOIN (produto AS pro INNER JOIN (anuncio as anu INNER JOIN produtoAnuncio as prodAnu ON prodAnu.anuCodigo = anu.anuCodigo) ON pro.proCodigo = prodAnu.proCodigo) ON tip.tiaCodigo = anu.tiaCodigo

WHERE anu.anuCodigo = codigo

ORDER BY anu.anuDescricao;

END

&&

--OUTRAS PROCEDURES

DROP PROCEDURE IF EXISTS RETORNA\_INDEX;

DELIMITER &&

CREATE PROCEDURE RETORNA\_INDEX (

IN TABELA VARCHAR(80)

)

BEGIN

SELECT AUTO\_INCREMENT

FROM information\_schema.tables

WHERE table\_name = TABELA

AND table\_schema = DATABASE();

END

&&

DROP PROCEDURE IF EXISTS ATUALIZA\_QUANTIDADE\_PRODUTO;

DELIMITER &&

CREATE PROCEDURE ATUALIZA\_QUANTIDADE\_PRODUTO (

IN V\_PROCODIGO INT,

V\_QUANTIDADE INT,

V\_OPERACAO BOOL

)

BEGIN

DECLARE V\_QUANTIDADE\_ATUAL INT;

SELECT PROQUANTIDADE

INTO V\_QUANTIDADE\_ATUAL

FROM PRODUTO

WHERE PROCODIGO = V\_PROCODIGO;

IF V\_OPERACAO = TRUE THEN

UPDATE PRODUTO SET PROQUANTIDADE = V\_QUANTIDADE\_ATUAL + V\_QUANTIDADE WHERE PROCODIGO = V\_PROCODIGO;

ELSE

UPDATE PRODUTO SET PROQUANTIDADE = V\_QUANTIDADE\_ATUAL - V\_QUANTIDADE WHERE PROCODIGO = V\_PROCODIGO;

END IF;

END

&&

DROP PROCEDURE IF EXISTS AUTENTICA\_USUARIO;

DELIMITER &&

CREATE PROCEDURE AUTENTICA\_USUARIO (

IN V\_USRLOGIN VARCHAR(255),

IN V\_USRSENHA VARCHAR(32)

)

BEGIN

SELECT COUNT(USRCODIGO)

FROM USUARIO

WHERE USRLOGIN = V\_USRLOGIN

AND USRSENHA = V\_USRSENHA;

END

&&

DROP PROCEDURE IF EXISTS VERIFICA\_USUARIO;

DELIMITER &&

CREATE PROCEDURE VERIFICA\_USUARIO (

IN V\_USRLOGIN VARCHAR(255)

)

BEGIN

SELECT COUNT(USRLOGIN) FROM USUARIO WHERE USRLOGIN = V\_USRLOGIN;

END

&&

1. **Elaboração do Plano de Teste**

**7.1 Histórico das Revisões**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <26/Maio/16> | <1.0> | Release Inicial | Cool & Cute |

## 7.2 Objetivos

Este Plano de Testes da Cool & Cute visa realizar os seguintes objetivos:

• Identificar informações de projeto existentes e os componentes de software que devem ser testados.

• Listar os Requisitos a Testar recomendados (alto nível).

• Recomendar e descrever as estratégias de teste a serem empregadas.

• Identificar os recursos necessários e prover uma estimativa dos esforços de teste.

• Listar os elementos resultantes do projeto de testes.

## Identificação de Projeto

A tabela abaixo identifica a documentação e disponibilidade usados para desenvolver o plano de testes:

|  |  |
| --- | --- |
| **Documento** | **Criado ou Disponível** |
| Especificação de Requisitos | ◼ Sim 🞏 Não |
| Diagramas UML | ◼ Sim 🞏 Não |
| Modelo de Análise | 🞏 Sim ◼ Não |
| Modelo de Projeto | 🞏 Sim ◼ Não |
| Documento de Arquitetura | ◼ Sim 🞏 Não |
| Protótipo | ◼ Sim 🞏 Não |
| Manual do Usuário | 🞏 Sim ◼ Não |
| Lista de Riscos | 🞏 Sim ◼ Não |

## Requisitos a Testar

A lista abaixo identifica os itens – (casos de uso ) , requisitos funcionais e não funcionais – que foram identificados como alvos de teste. Essa lista representa o que será testado.

## 7.4.1 Teste do Banco de Dados

* Verifique que as informações do usuário podem ser cadastradas, consultadas e removidas.
* Verifique se o sistema é capaz de aplicar os filtros de pesquisa nas seguintes telas de consultas

1. Consultar Clientes
2. Consultar Status Pedidos

## Teste Funcional

* Verifique que qualquer usuário pode acessar sua própria conta através de login e senha.

## Teste da Interface do Usuário

* Navegue através de todos os use cases, verificando que cada tela de interface gráfica pode ser rapidamente entendida e facilmente utilizada.
* Verifique que todos os campos obrigatórios estão emitindo o alerta caso não sejam preenchidos devidamente.

## 7.4.4 Teste de Segurança e de Controle de Acesso

Verificar que usuários não cadastrados não podem acessar informações restritas aos cadastrados.

Verificar que além do administrador, ninguém mais pode inserir, atualizar ou remover dados do sistema.

## Teste de Instalação

Verifique que a instalação do sistema ocorre normalmente em todas as máquinas.

Verifique que o sistema é capaz de obter e atualizar as informações úteis a que se propõe a disponibilizar.

Verifique que o espaço disponível em disco para informações deve ser capaz de armazenar todos os dados/atualizações que forem cadastrados.

1. **Manual do Usuário**

**8.1 Introdução**

Este documento tem como principal finalidade esclarecer os procedimentos adequados para a utilização deste software ,cujo objetivo, é a organização de vendas da empresa Cool & Cute.

**8.2 Principais Componetes**

O sistema Cool & Cute contém diversas telas de cadastro , porém foram selecionadas de acordo com a equipe desenvolvedora as telas mais complexas para serem esclarecidos os procedimentos, a seguir a lista das telas que fazem parte deste documento :

* + - * Cadastro de Pedidos
      * Consultar de Pedidos

### **8.2.A Cadastrando um Pedido**

Para Cadastrar um pedido na tela (Pedidos/Criar) , preencha os campos detalhado na Figura 1a seguir :

Figura 1

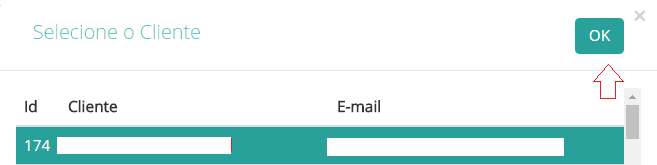
Ao clicar no ícone  o sistema disponibilizará um pop-up com a lista de todos os Anúncios e Clientes que estão cadastrados no sistema , sendo assim basta selecioná-los com um click e depois na opção OK (Figura 2)

Figura 2

Para preencher o campo *Data da Venda* o usuário tem a opção de digitar a data no campo ou tem a opção de exibir um calendário e selecionar a data desejada clicando no ícone que será exibido na figura a seguir :

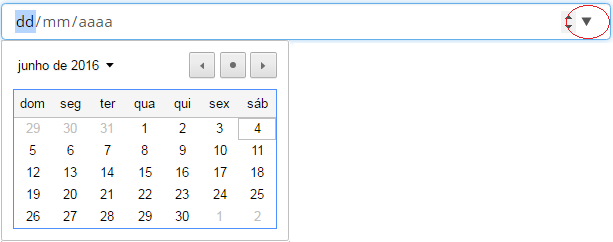


Figura 3

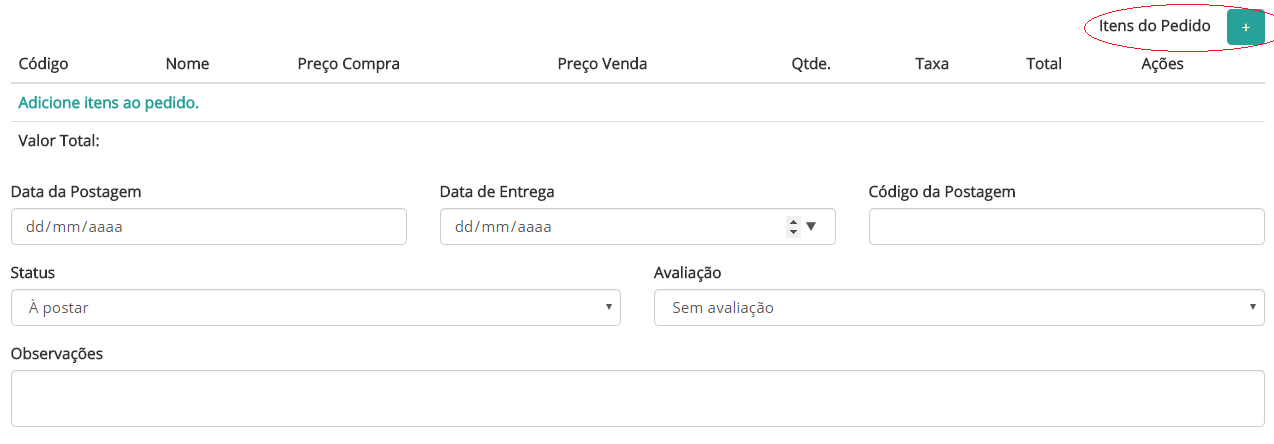
Realizados estes procedimentos, é necessário preencher a segunda parte dos campos que dizem respeito aos **Itens do Pedido**

Figura 4

**Ao clicar no botão o sistema exibirá um Modal com todos os Produtos Cadastrados , é possível fazer mais de uma seleção , basta selecioná-los com o click , em seguida clique no botão veja Figura 5



Figura 5

Após selecionar os itens do pedido que foram solicitados nos passos anteriores , insira nos campos **QTDE** (quantidade) de cada ítem selecionado o valor correspondente á quantidade de itens de entrada do pedido , veja o exemplo na Figura 6

Figura 6

Para finalizar o cadastro do pedido preenchendo os campos detalhados na figura a seguir e clique no que está detalhado na Figura 7 :

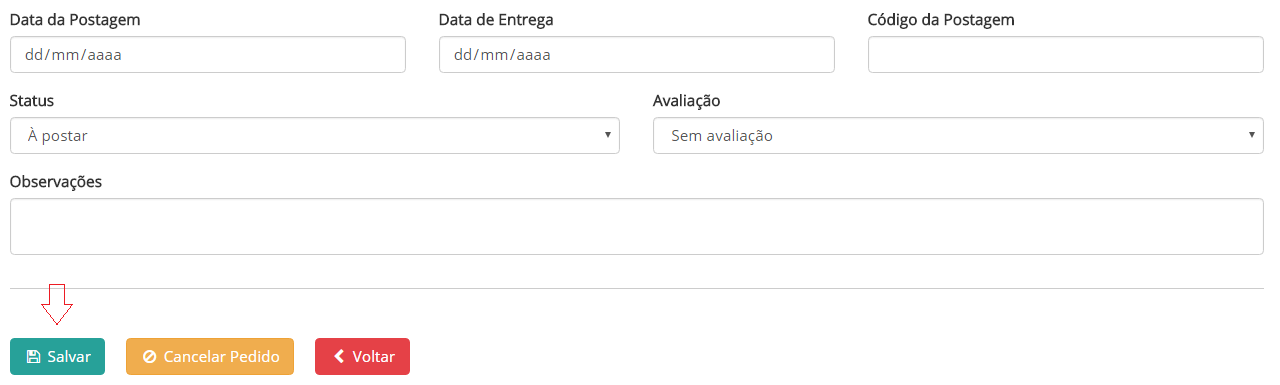


Figura 7

### **8.2.B Consultando um Pedido**

O sistema disponibiliza duas opçãos para realizar a consulta á um pedido:

* O usuário pode fazer uma combinação de filtros que o sistema disponibiliza (veja Figura 6 )em seguida o usuário clica na opção
* O usuário pode listar todos os pedidos disponíveis no sistema clicando na opção

****

Figura 8

Realizando os procedimentos detalhados anteriormente , o sistema disponibiliza uma lista ao usuário com todos os pedidos cadastrados no sistema ( veja Figura 9**.**)

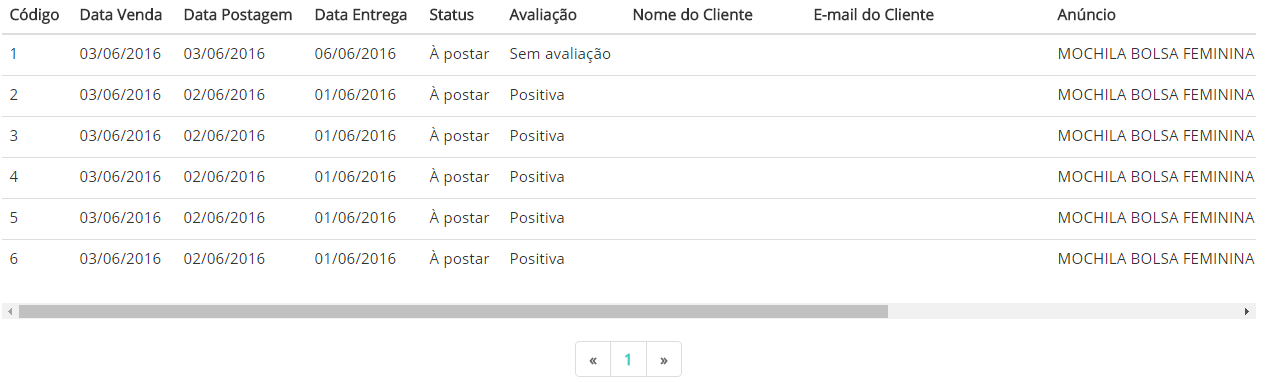
****

Figura 9

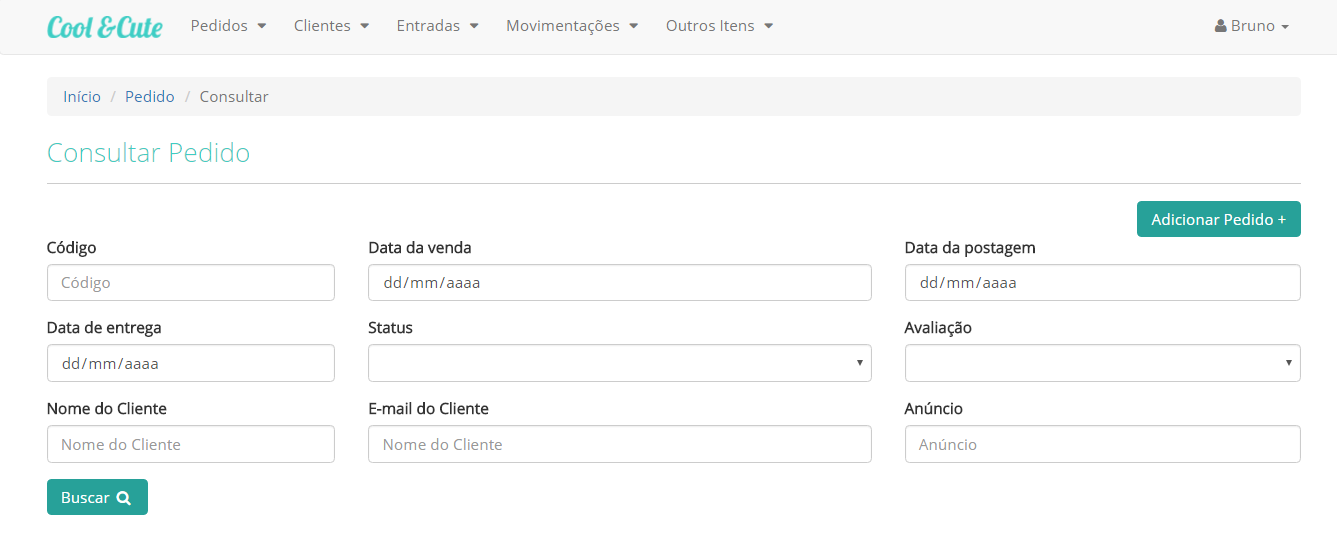
1. **Desenvolvimento das Interfaces (Protótipos)**

Para a produção das interfaces do sistema Cool & Cute foram usadas alguns protótipos que servirão como modelo ou base para a criação de todas as telas do software, confira a seguir quais as telas foram utilizadas para este protótipo :

* Consulta de Pedidos
* Consulta de Pedidos Modal
* Cadastro de Pedidos
* Cadastro de Pedidos Modal
  1. **Consulta de Pedidos**

***Versão Mobile***



***Versão desktop***

## Consulta de Pedidos Modal

***Versão Mobile***

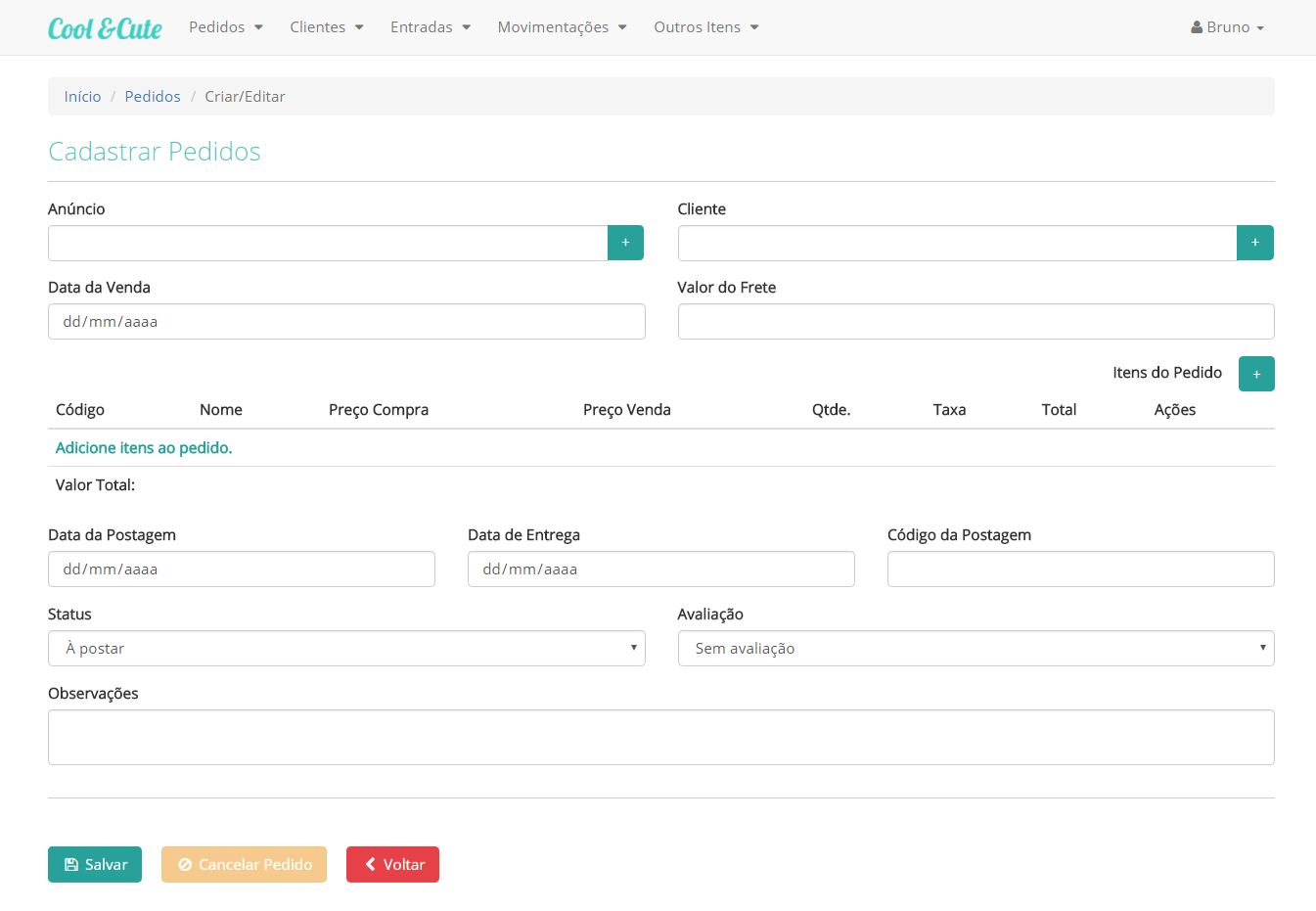


* 1. **Cadastro de Pedidos**

***Versão Mobile***

****

***Versão Desktop***



## 9.4 Cadastro de Pedidos Modal

# Referências Bibliográficas

LANZ, Luciano Quinto e LANZ, Renata. Modelos de Termo de Abertura de Projeto. 2013. Disponível em: <http://pmkb.com.br/artigo/modelos-de-termo-de-abertura-de-projeto/> Acesso em: 19/03/2016.

SILVA, Carlos Sérgio Mota. Dicas para elaboração de termos de abertura de projetos. Disponível em: <http://www.p4pro.com.br/index.php/artigos/32-dicas-para-a-elaboracao-de-termos-de-abertura-de-projetos>. Acesso em: 19/03/2016.

Sotille, Mauro. Diferenciando Requisitos, Restrições e Premissas. 2012. Disponível em: < http://www.pmtech.com.br/PMP/Requisitos\_Restricoes\_Premissas.pdf >. Acesso em: 20/03/2016