

Trabalho de Pesquisa Aplicada a Novas Metodologias para Desenvolvimento de Sistemas

**Professor** : Fabio de Jesus Souza

Analise e Desenvolvimento de Sistemas

Nome : Bruno Alves – RA: XXXXXXXXX

Nome: Thiago Pereira - RA : 914123856

Nome : Valber – RA:XXXXXXXXX

Sumário

[1. Termo de Abertura do Projeto 3](#_Toc451975468)

[1.1. Justificativa 3](#_Toc451975469)

[1.2. Objetivo 3](#_Toc451975470)

[1.3. Intitulação do Sistema 3](#_Toc451975471)

[1.4. Restrições 3](#_Toc451975472)

[2. Requisitos 4](#_Toc451975473)

[2.1. Requisitos Funcionais 4](#_Toc451975474)

[2.2. Requisitos Não Funcionais 6](#_Toc451975475)

[3. Diagrama de Casos de Uso 8](#_Toc451975476)

[3.1 - 001 Autenticação no Sistema (Login) 9](#_Toc451975477)

[3.2 – 002 Cadastrar Produto 10](#_Toc451975478)

[3.3 – 003 Redefinir Senha 11](#_Toc451975479)

[3.4 – 004 Consultar Produtos 12](#_Toc451975480)

[3.5 – 005 Alterar Produto 13](#_Toc451975481)

[3.6 – 006 Deletar Produto 14](#_Toc451975482)

[3.7 – 007 Cadastrar Cliente 15](#_Toc451975483)

[3.8 – 008 Consultar Produto 16](#_Toc451975484)

[3.9 – 009 Alterar Cliente 17](#_Toc451975485)

[3.10 – 010 Deletar Cliente 18](#_Toc451975486)

[3.11 – 011 Criar Anúncio 19](#_Toc451975487)

[3.12 – 012 Calcular Taxa de Juros 20](#_Toc451975488)

[3.13 – 013 Cadastrar Pedido 21](#_Toc451975489)

[3.14 – 014 Incluir Movimentação Financeira 22](#_Toc451975490)

[3.15 – 015 Atualizar Caixa 23](#_Toc451975491)

[3.16 – 016 Consultar Pedido 24](#_Toc451975492)

[3.17 – 017 Cancelar Pedido 25](#_Toc451975493)

[3.18 - 018 Alterar Status do Pedido 26](#_Toc451975494)

[3.19 – 019 Emitir Relatório de Vendas 27](#_Toc451975495)

[3.20 – 020 Consultar Relatório de Vendas 28](#_Toc451975496)

[4. Diagrama de Classes 29](#_Toc451975497)

[5. Cronograma do Projeto 30](#_Toc451975498)

[5.1 Definição 30](#_Toc451975499)

[5.2 Modelos e Diagramas 30](#_Toc451975500)

[5.3 Diagramas UML 30](#_Toc451975501)

[5.4 Prototipação 31](#_Toc451975502)

[5.5 Protótipos 31](#_Toc451975503)

[5.6 Desenvolvimento Front-End 31](#_Toc451975504)

[5.7 Desenvolvimento Back-End 32](#_Toc451975505)

[5.8 Testes 32](#_Toc451975506)

[5.9 Implantação 32](#_Toc451975507)

[6. Modelagem de Dados 33](#_Toc451975508)

[**6.1 Modelo Relacional Lógico** 33](#_Toc451975509)

[6.2 Projeto Físico do Banco de Dados 34](#_Toc451975510)

[6.2.1 Criação das tabelas 34](#_Toc451975511)

[6.2.2 Procedures 37](#_Toc451975512)

[7. Elaboração do Plano de Teste 52](#_Toc451975513)

# 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 torna 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 01/06/2016.

# Requisitos

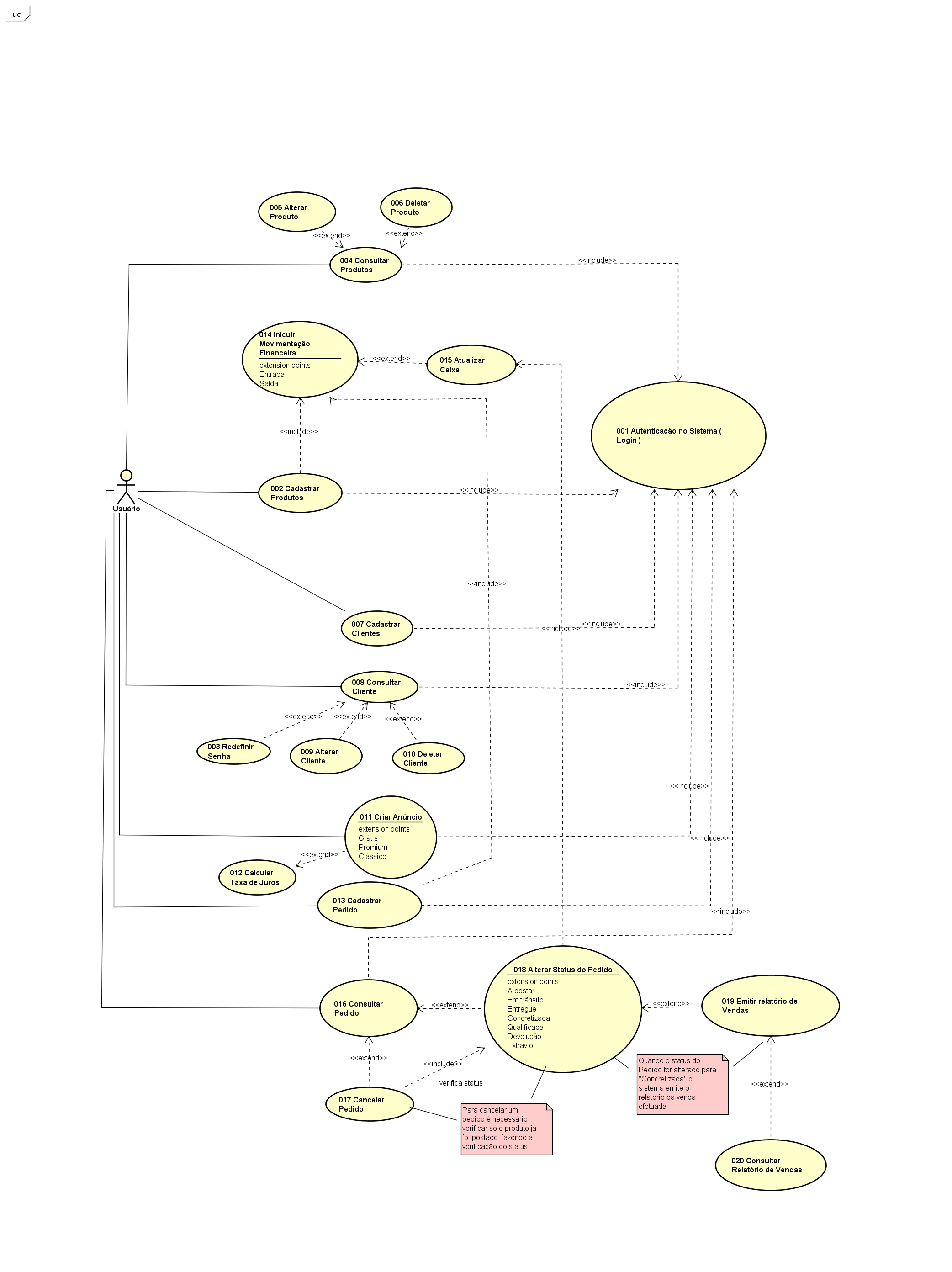
## Requisitos Funcionais

* + 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 juro 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 juro das compras parceladas é repassado 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.

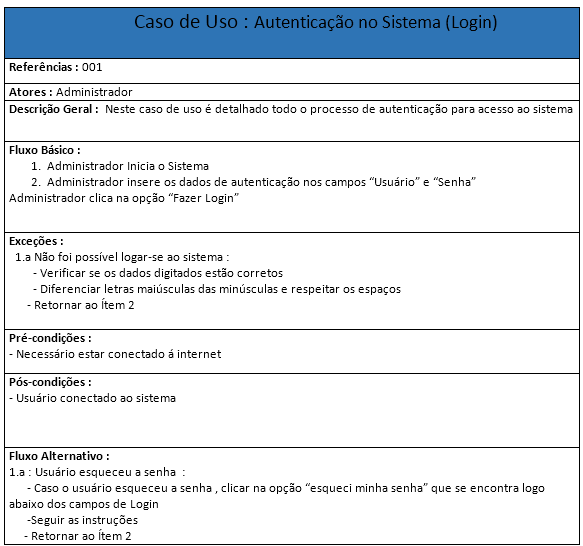
## Requisitos Não Funcionais

* + 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.

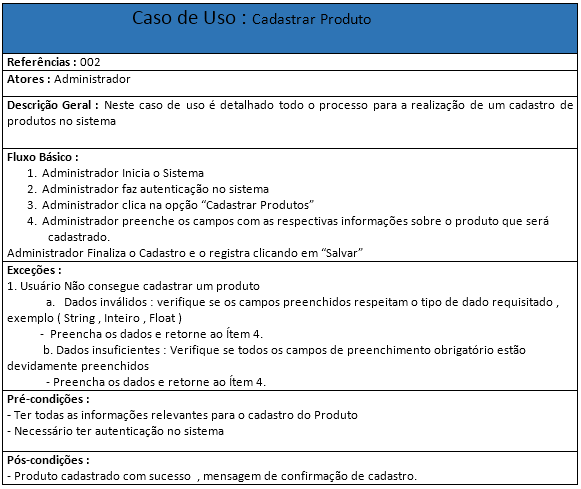
# Diagrama de Casos de Uso



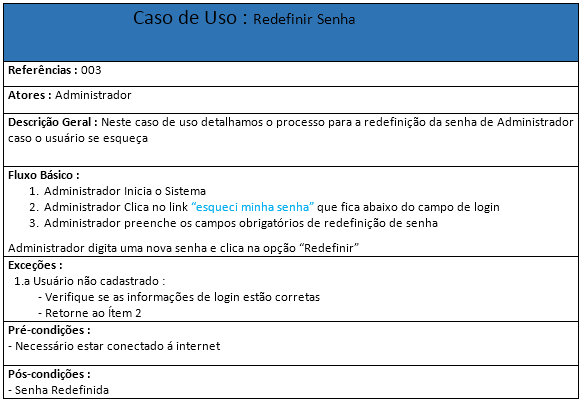
## 3.1 - 001 Autenticação no Sistema (Login)



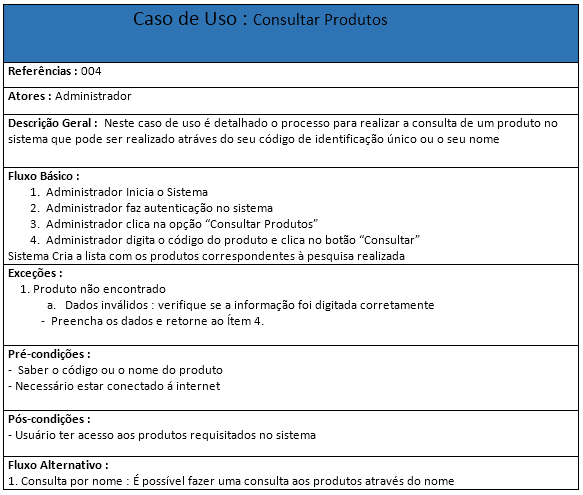
## 3.2 – 002 Cadastrar Produto



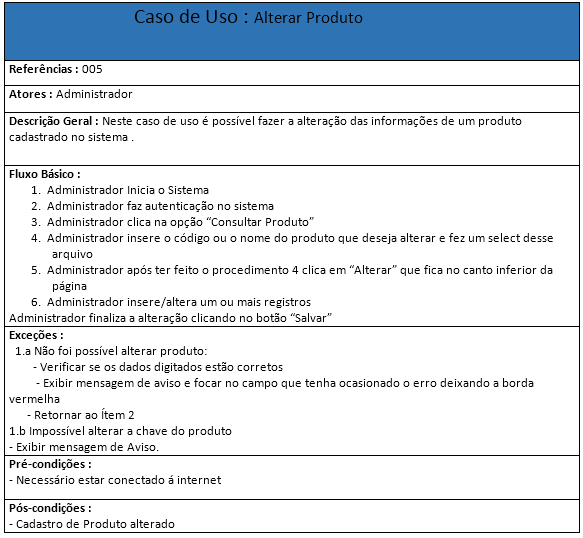
## 3.3 – 003 Redefinir Senha



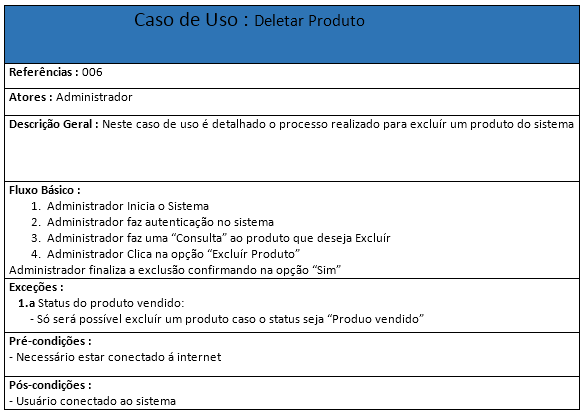
## 3.4 – 004 Consultar Produtos

****

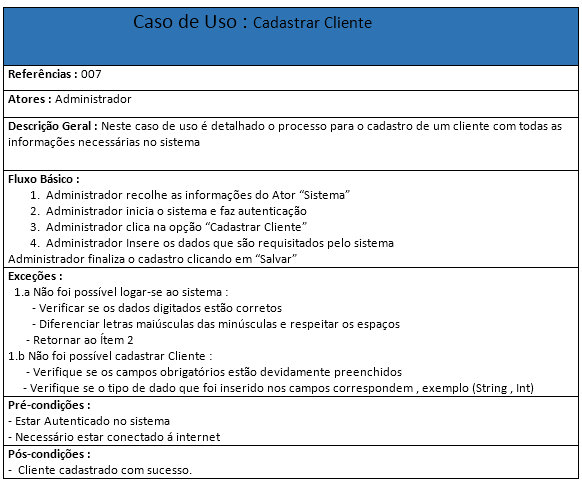
## 3.5 – 005 Alterar Produto



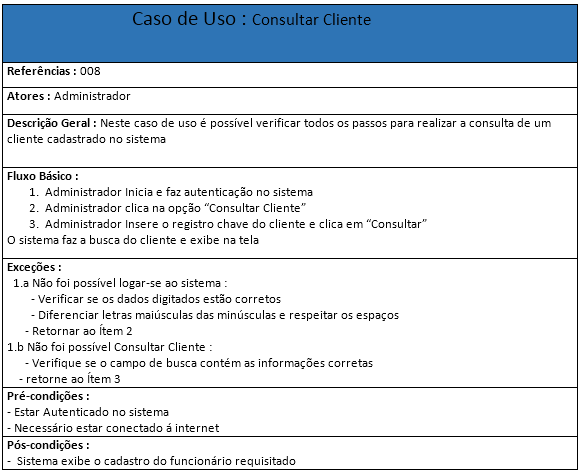
## 3.6 – 006 Deletar Produto



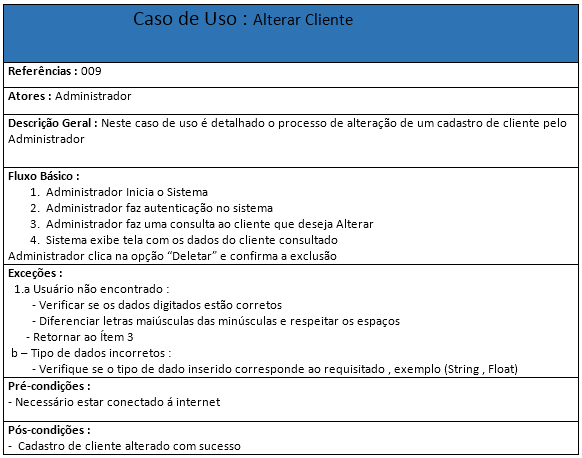
## 3.7 – 007 Cadastrar Cliente



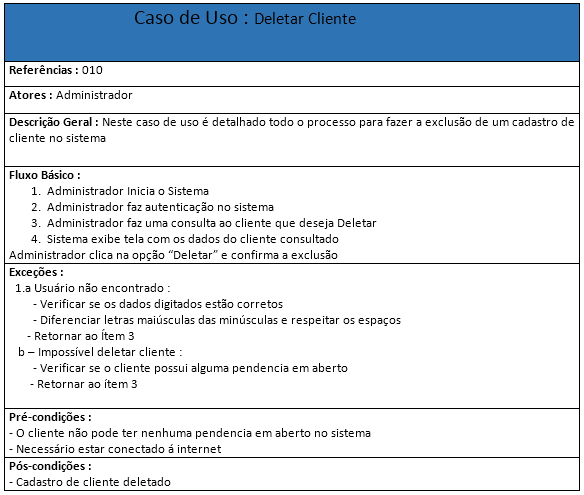
## 3.8 – 008 Consultar Produto



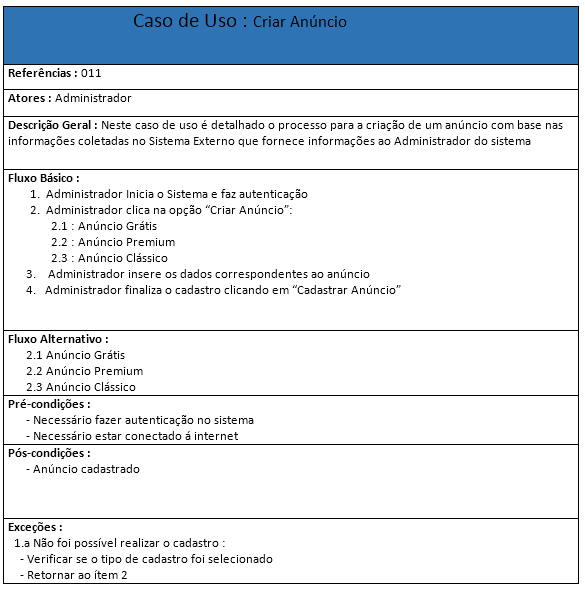
## 3.9 – 009 Alterar Cliente



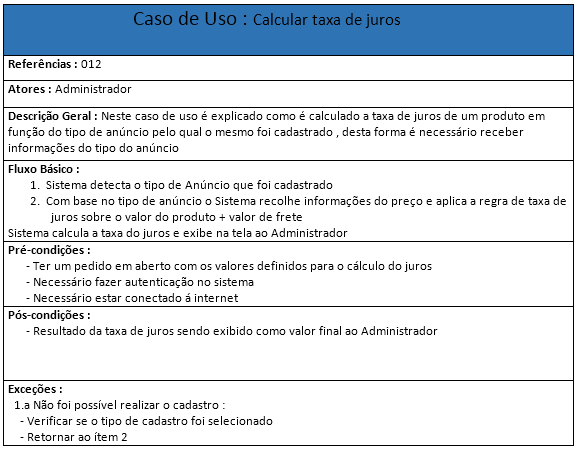
## 3.10 – 010 Deletar Cliente



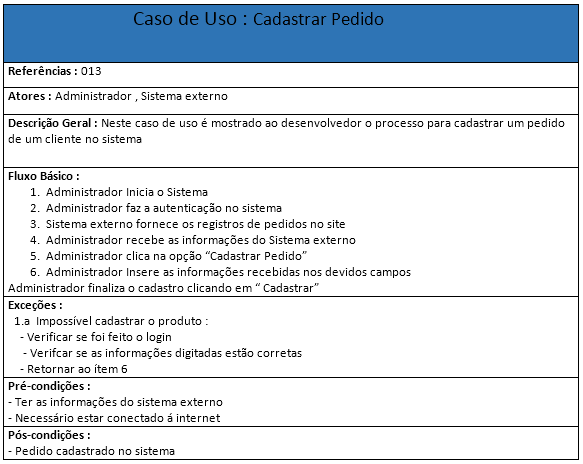
## 3.11 – 011 Criar Anúncio



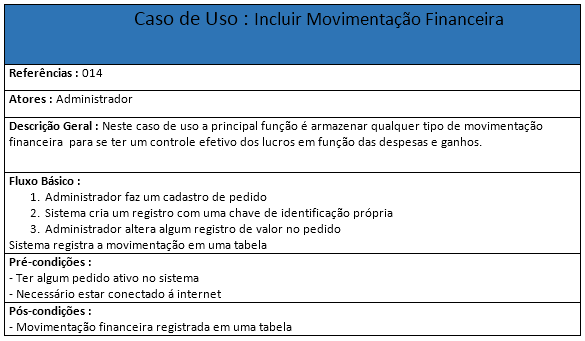
## 3.12 – 012 Calcular Taxa de Juros



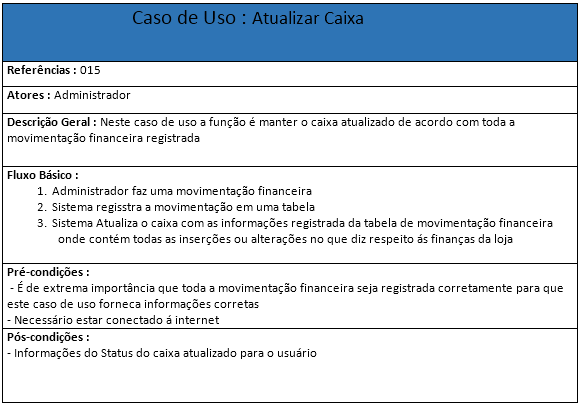
## 3.13 – 013 Cadastrar Pedido



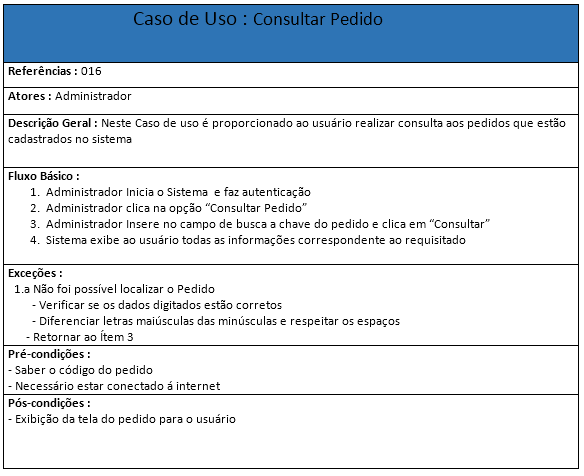
## 3.14 – 014 Incluir Movimentação Financeira



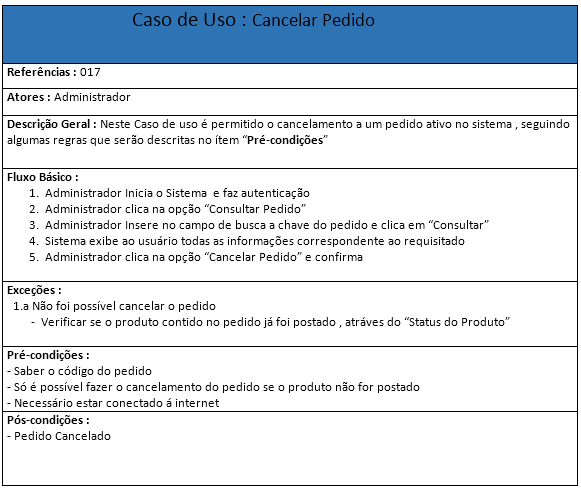
## 3.15 – 015 Atualizar Caixa



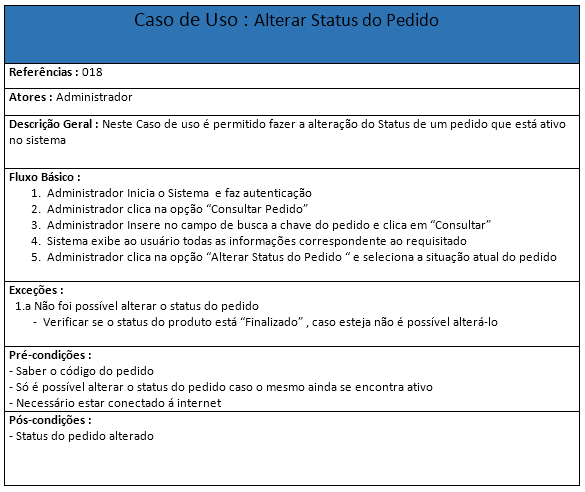
## 3.16 – 016 Consultar Pedido



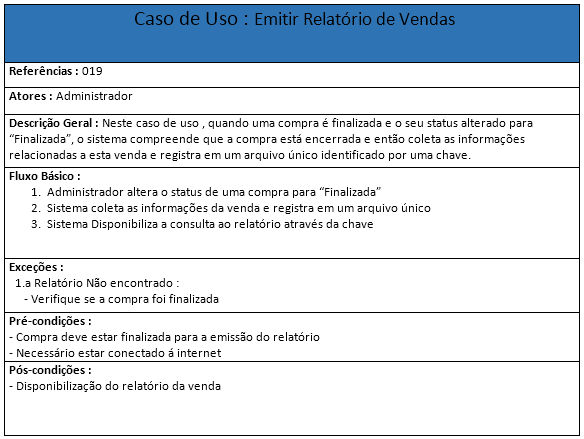
## 3.17 – 017 Cancelar Pedido



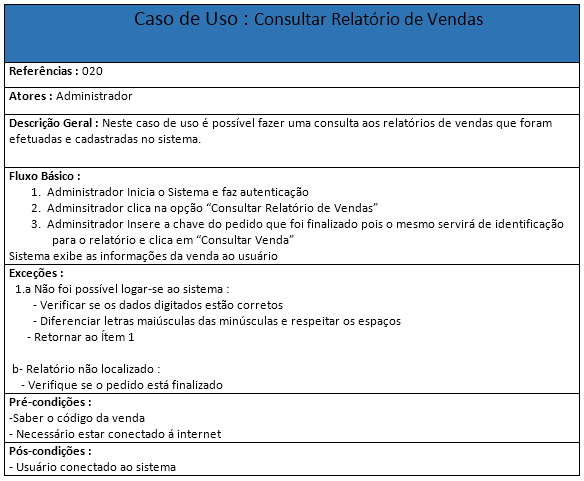
## 3.18 - 018 Alterar Status do Pedido

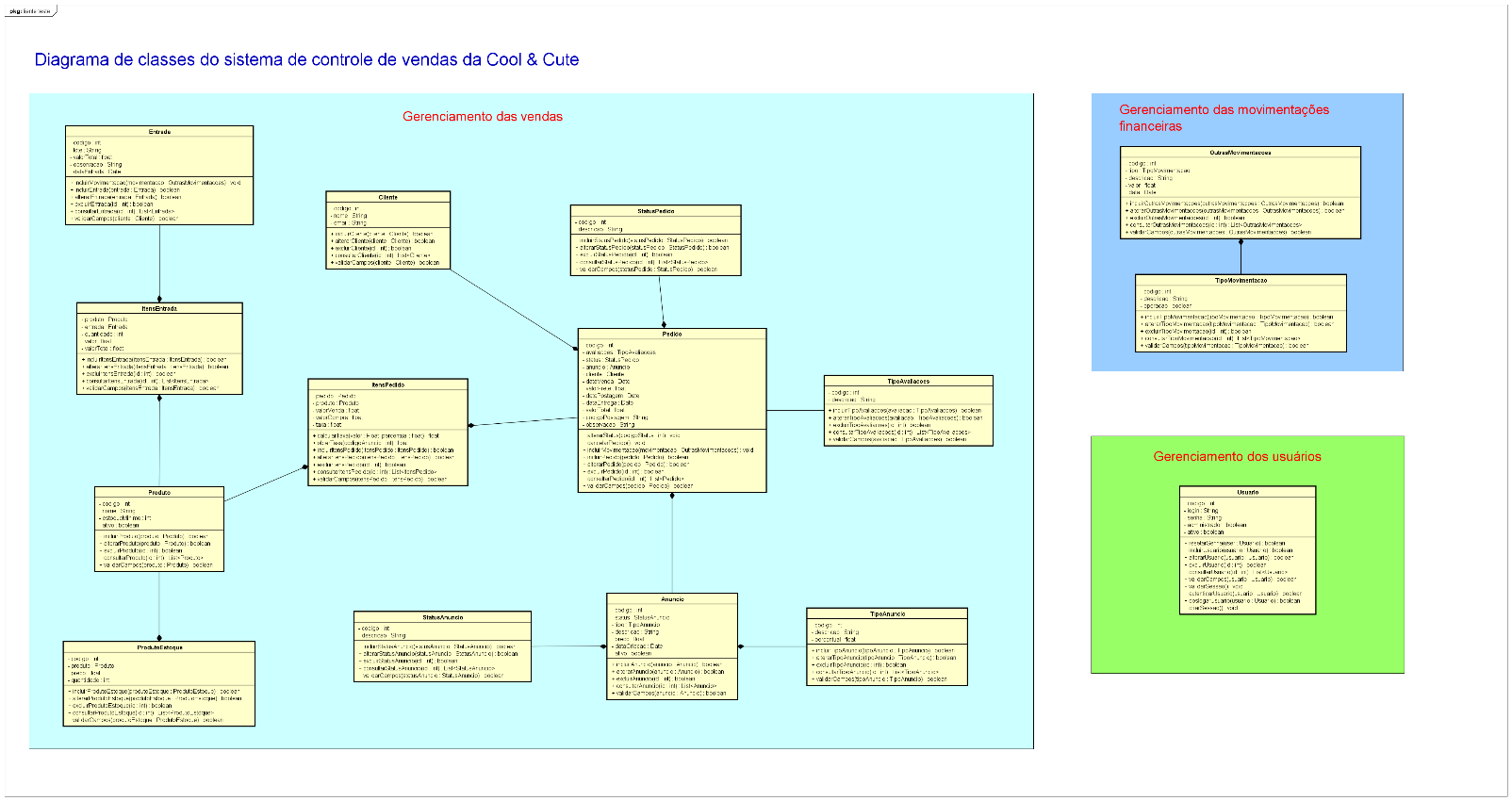


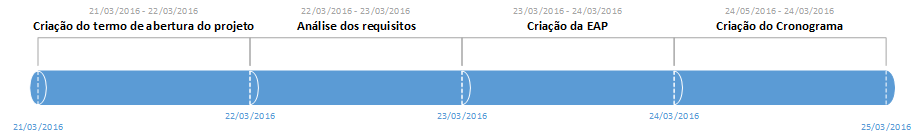
## 3.19 – 019 Emitir Relatório de Vendas



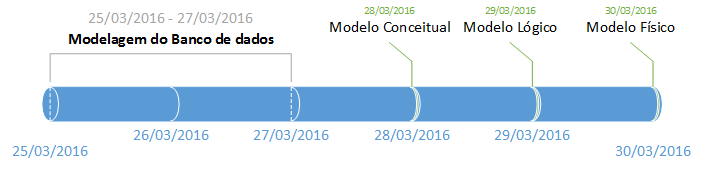
## – 020 Consultar Relatório de Vendas



1. ****Diagrama de Classes
2. Cronograma do Projeto

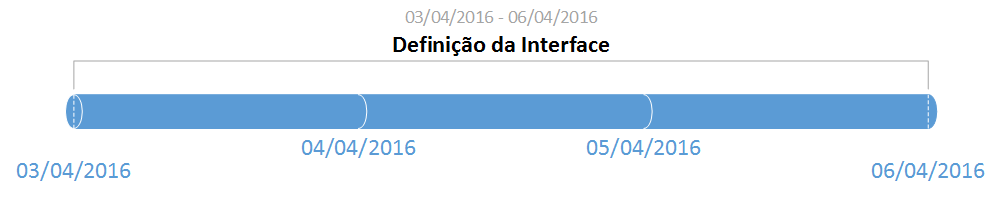
5.1 Definição

## 5.2 Modelos e Diagramas

****

## 5.3 Diagramas UML

## 5.4 Prototipação



## 5.5 Protótipos

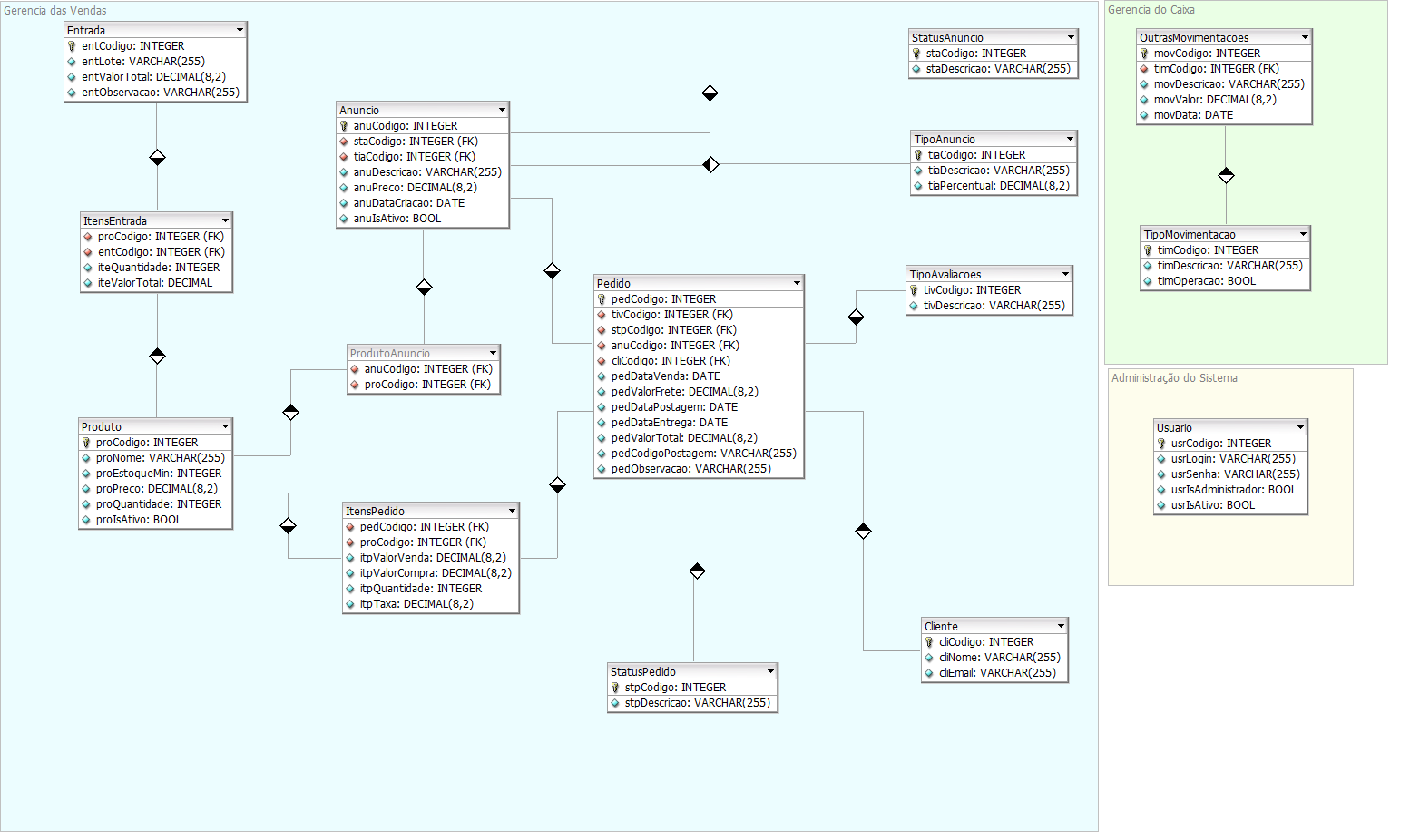
## 5.6 Desenvolvimento Front-End

## 5.7 Desenvolvimento Back-End

## 5.8 Testes

## 5.9 Implantação

1. **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 NOT NULL,

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

&&

-- 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

&&

-- PRODUTO

DROP PROCEDURE IF EXISTS PRODUTO\_INSERT;

DELIMITER &&

CREATE PROCEDURE PRODUTO\_INSERT (

IN V\_PRONOME VARCHAR(255),

V\_PROESTOQUEMIN INT,

V\_PROISATIVO BOOL

)

BEGIN

INSERT INTO PRODUTO (PRONOME, PROESTOQUEMIN, PROISATIVO) VALUES (V\_PRONOME, V\_PROESTOQUEMIN, 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\_PROISATIVO BOOL

)

BEGIN

UPDATE PRODUTO SET PRONOME = V\_PRONOME, PROESTOQUEMIN = V\_PROESTOQUEMIN, 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

&&

-- 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

&&

-- 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

&&

-- TIPOANUNCIO

DROP PROCEDURE IF EXISTS TIPOANUNCIO\_INSERT;

DELIMITER &&

CREATE PROCEDURE TIPOANUNCIO\_INSERT (

IN V\_TIADESCRICAO VARCHAR(255),

V\_TIAPERCENTUAL DECIMAL

)

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

)

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

&&

-- 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

&&

-- 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

&&

-- ENTRADA--

DROP PROCEDURE IF EXISTS ENTRADA\_INSERT;

DELIMITER &&

CREATE PROCEDURE ENTRADA\_INSERT (

IN

V\_ENTLOTE VARCHAR(255),

V\_ENTVALORTOTAL DECIMAL,

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,

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

&&

-- OUTRASMOVIMENTAÇOES

DROP PROCEDURE IF EXISTS OUTRASMOVIMENTACOES\_INSERT;

DELIMITER &&

CREATE PROCEDURE OUTRASMOVIMENTACOES\_INSERT (

IN V\_TIMCODIGO INT,

V\_MOVDESCRICAO VARCHAR(255),

V\_MOVVALOR DECIMAL,

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,

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\_DELETE;

DELIMITER &&

CREATE PROCEDURE OUTRASMOVIMENTACOES\_DELETE (

IN V\_MOVCODIGO INT

)

BEGIN

DELETE FROM OUTRASMOVIMENTACOES WHERE MOVCODIGO = V\_MOVCODIGO;

END

&&

-- PRODUTOESTOQUE

DROP PROCEDURE IF EXISTS PRODUTOESTOQUE\_INSERT;

DELIMITER &&

CREATE PROCEDURE PRODUTOESTOQUE\_INSERT (

IN V\_PROCODIGO INT,

V\_ESTPRECO DECIMAL,

V\_ESTQUANTIDADE INT

)

BEGIN

INSERT INTO produtoestoque (PROCODIGO, ESTPRECO, ESTQUANTIDADE) VALUES (V\_PROCODIGO, V\_ESTPRECO,V\_ESTQUANTIDADE);

END

&&

DROP PROCEDURE IF EXISTS PRODUTOESTOQUE\_UPDATE;

DELIMITER &&

CREATE PROCEDURE PRODUTOESTOQUE\_UPDATE (

IN

V\_ESTCODIGO INT,

V\_PROCODIGO INT,

V\_ESTPRECO DECIMAL,

V\_ESTQUANTIDADE INT

)

BEGIN

UPDATE PRODUTOESTOQUE SET PROCODIGO = V\_PROCODIGO, ESTPRECO = V\_ESTPRECO, ESTQUANTIDADE = V\_ESTQUANTIDADE WHERE ESTCODIGO = V\_ESTCODIGO;

END

&&

DROP PROCEDURE IF EXISTS PRODUTOESTOQUE\_DELETE;

DELIMITER &&

CREATE PROCEDURE PRODUTOESTOQUE\_DELETE (

IN V\_ESTCODIGO INT

)

BEGIN

DELETE FROM PRODUTOESTOQUE WHERE ESTCODIGO = V\_ESTCODIGO;

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,

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\_STACODIGO INT,

V\_TIACODIGO INT,

V\_ANUDESCRICAO VARCHAR(255),

V\_ANUPRECO DECIMAL,

V\_ANUDATACRIACAO DATE,

V\_ANUISATIVO BOOL

)

BEGIN

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

WHERE STACODIGO = V\_STACODIGO;

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

&&

-- ITENSENTRADA

DROP PROCEDURE IF EXISTS ITENSENTRADA\_INSERT;

DELIMITER &&

CREATE PROCEDURE ITENSENTRADA\_INSERT (

IN V\_PROCODIGO INT,

V\_ENTCODIGO INT,

V\_ITEQUANTIDADE INT,

V\_ITEVALOR DECIMAL,

V\_ITEVALORTOTAL DECIMAL

)

BEGIN

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

VALUES (V\_PROCODIGO,V\_ENTCODIGO,V\_ITEQUANTIDADE,V\_ITEVALOR,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\_ITEVALOR DECIMAL,

V\_ITEVALORTOTAL DECIMAL

)

BEGIN

UPDATE ITENSENTRADA SET PROCODIGO = V\_PROCODIGO, ENTCODIGO = V\_ENTCODIGO,ITEQUANTIDADE = V\_ITEQUANTIDADE,

ITEVALOR = V\_ITEVALOR, 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\_PROCODIGO INT,

V\_ENTCODIGO INT

)

BEGIN

DELETE FROM ITENSENTRADA WHERE PROCODIGO = V\_PROCODIGO AND ENTCODIGO = V\_ENTCODIGO;

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,

V\_PEDDATAPOSTAGEM DATE,

V\_PEDDATAENTREGA DATE,

V\_PEDVALORTOTAL DECIMAL,

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,

V\_PEDDATAPOSTAGEM DATE,

V\_PEDDATAENTREGA DATE,

V\_PEDVALORTOTAL DECIMAL,

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

&&

-- ITENSPEDIDO

DROP PROCEDURE IF EXISTS ITENSPEDIDO\_INSERT;

DELIMITER &&

CREATE PROCEDURE ITENSPEDIDO\_INSERT (

IN V\_PEDCODIGO INT,

V\_PROCODIGO INT,

V\_ITPVALORVENDA DECIMAL,

V\_ITPVALORCOMPRA DECIMAL,

V\_ITPTAXA DECIMAL

)

BEGIN

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

V\_ITPVALORCOMPRA,V\_ITPTAXA);

END

&&

DROP PROCEDURE IF EXISTS ITENSPEDIDO\_UPDATE;

DELIMITER &&

CREATE PROCEDURE ITENSPEDIDO\_UPDATE (

IN V\_PEDCODIGO INT,

V\_PROCODIGO INT,

V\_ITPVALORVENDA DECIMAL,

V\_ITPVALORCOMPRA DECIMAL,

V\_ITPTAXA DECIMAL

)

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,

V\_PROCODIGO INT

)

BEGIN

DELETE FROM ITENSPEDIDO WHERE PEDCODIGO = V\_PEDCODIGO AND PROCODIGO = V\_PROCODIGO;

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.

## 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

## Teste Funcional

* Verifique que as informações úteis obtidas pelo subsistema responsável são automaticamente e periodicamente atualizadas.
* Verifique que qualquer usuário pode acessar sua própria conta através de login e senha.
* Verifique que o relatório da conta do hóspede é correto.
* Verifique que as informações podem ser acessadas em qualquer dos idiomas disponíveis.

## Teste do Ciclo de Negócios

Nenhum.

## 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 toda ajuda online funciona.
* Verifique que todas as palavras e expressões apresentadas nos diversos idiomas estão em conformidade com as devidas normas sintáticas e gramaticais.

## Perfil da Performance

Verifique o tempo de resposta da rede interna, do servidor em relação aos terminais.

Verifique o tempo de consulta/atualização do subsistema de informações úteis.

Verifique que o tempo de resposta para operações que envolvam dados multimídia (imagens, vídeos, etc.) não ultrapassam 30 segundos.

## Teste de Carga

Verificar a resposta do sistema com 10 usuários.

Verificar a resposta do sistema com 50 usuários.

Verificar a resposta do sistema com 100 usuários.

Verificar a resposta do sistema com 200 usuários.

Verificar a resposta do sistema com 500 usuários.

## Teste de Stress

Nenhum.

## Teste de Volume

Nenhum.

## 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.

Verificar que os hóspedes apenas vêem suas próprias informações de conta no hotel.

Verificar que os usuários do sistema podem acessar apenas as funcionalidades e dados associados ao seu próprio tipo de usuário.

Verificar que a atualização do sistema pode ser feita apenas a partir da rede interna do hotel.

## Teste de Falha/Recuperação

Nenhum.

## Teste de Instalação

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

Verifique que qualquer terminal do sistema do hotel é capaz de rodar o TIG normalmente.

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

Verifique que a atualização dos dados no servidor se reflete em todos os terminais do hotel.

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

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