|  |
| --- |
|  |
| Universidade do Minho |
| Licenciatura em Engenharia Informática |

Relatório Final

Projecto da U.C. de Modelação e Desenvolvimento

Braga, 6 de Janeiro de 2013

**Elaborado por:**

Grupo 33

|  |  |
| --- | --- |
| C:\Users\marcelo\Desktop\piças.jpg | C:\Users\marcelo\Desktop\paxa.jpg |
| Pedro Miguel Costa Maia – A61078 | Luís Carlos Caseiro – A61009 |
| C:\Users\marcelo\Desktop\ces.jpg | C:\Users\marcelo\Pictures\EU\Pedro Faria - Cópia.jpg |
| César Morais Perdigão – A61007 | Pedro Miguel Oliveira Faria- A60998 |

Índice

[Índice Tabelas: 3](#_Toc345275216)

[Índice Figuras: 4](#_Toc345275217)

[4 – Introdução 5](#_Toc345275218)

[5 – Desenvolvimento 6](#_Toc345275219)

[5.1 – Modelo de Domínio 6](#_Toc345275220)

[5.2 – Modelo de Use-Case 8](#_Toc345275221)

[5.3 – Diagramas de Sequência 11](#_Toc345275222)

[5.4 – Diagrama de Classes 12](#_Toc345275223)

[5.5 – Diagrama da Base de Dados 13](#_Toc345275224)

[5.6 – Camadas 15](#_Toc345275225)

[5.6.1 - Camada de Apresentação (Interface do Utilizador) 15](#_Toc345275226)

[5.6.2 - Camada de Dados 23](#_Toc345275227)

[5.6.3 - Camada Business 23](#_Toc345275228)

[5.6.3.1 – Class BuyKing 23](#_Toc345275229)

[6 – Conclusão 27](#_Toc345275230)

[7 – Anexos 28](#_Toc345275231)

[5.2.1 - Use-Cases 28](#_Toc345275232)

[5.3.1 - Diagramas de sequência 37](#_Toc345275233)

[5.3.2 - Diagramas de sequência dos métodos principais: 47](#_Toc345275234)

[5.6.3.2 – Class Leilão 50](#_Toc345275235)

[5.6.3.3 – Class Produto 51](#_Toc345275236)

[5.6.3.4 – Class Suspeita 51](#_Toc345275237)

[5.6.3.5 – Class Troca 52](#_Toc345275238)

[5.6.3.6 – Class Venda 53](#_Toc345275239)

[5.6.3.7 – Class Utilizador 54](#_Toc345275240)

[5.6.3.8 – Class Licitação 55](#_Toc345275241)

[5.6.3.9 – Class Leilao Fechado Exception 55](#_Toc345275242)

[5.6.3.10 – Class Classificação 56](#_Toc345275243)

[5.6.3.11 - Class Baixa Licitação Exception 56](#_Toc345275244)

# Índice Tabelas:

[Tabela 1 - Reportar produto suspeito 28](#_Toc345275166)

[Tabela 2 – login 28](#_Toc345275167)

[Tabela 3 - Fazer licitação 29](#_Toc345275168)

[Tabela 4 - Comprar 30](#_Toc345275169)

[Tabela 5 - Analisar casos suspeitos 30](#_Toc345275170)

[Tabela 6 - Pedir sugestões 31](#_Toc345275171)

[Tabela 7 - Confirmar envio produto 31](#_Toc345275172)

[Tabela 8 - Vender 32](#_Toc345275173)

[Tabela 9 - Bloquear utilizador 32](#_Toc345275174)

[Tabela 10 - Adicionar à wishlist 33](#_Toc345275175)

[Tabela 11 - Registar 33](#_Toc345275176)

[Tabela 12 - Confirmar pagamento 34](#_Toc345275177)

[Tabela 13 - Pesquisa produto 34](#_Toc345275178)

[Tabela 14 - Leiloar 35](#_Toc345275179)

[Tabela 15 - Classificar utilizador 36](#_Toc345275180)

# Índice Figuras:

[Fig. 1– Modelo de domínio. 7](#_Toc345275181)

[Fig. 2 - Diagrama de Use-Case 10](#_Toc345275182)

[Fig. 3 - Diagrama de Classes 12](#_Toc345275183)

[Fig. 4 - Diagrama Base de Dados 13](#_Toc345275184)

[Fig. 5-Login 16](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275185)

[Fig. 6 - Favoritos 16](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275186)

[Fig. 7 - Principal 17](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275187)

[Fig. 8 - Vender 17](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275188)

[Fig. 9 – Histórico 18](#_Toc345275189)

[Fig. 10 - Leiloar 18](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275190)

[Fig. 11 - Pesquisa avançada 19](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275191)

[Fig. 12 - Produto em leilão 19](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275192)

[Fig. 13 - Registar 20](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275193)

[Fig. 14 - Produto à venda 20](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275194)

[Fig. 15 – Pesquisa Simples 21](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275195)

[Fig. 16 - A vender 21](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275196)

[Fig. 17 – Inicio 22](file:///C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Relatorio%20Fase%202.docx#_Toc345275197)

[Fig. 18 - Reportar Produto Suspeito 37](#_Toc345275198)

[Fig. 19 – Login 37](#_Toc345275199)

[Fig. 20 - Fazer Licitação 38](#_Toc345275200)

[Fig. 21 - Comprar 39](#_Toc345275201)

[Fig. 22 - Analisar casos suspeitos 40](#_Toc345275202)

[Fig. 23 - Pedir Sugestões 41](#_Toc345275203)

[Fig. 24 - Confirmar envio de produto 41](#_Toc345275204)

[Fig. 25- Vender 42](#_Toc345275205)

[Fig. 26 – Bloquear utilizador 42](#_Toc345275206)

[Fig. 27 - Adicionar à wishlist 43](#_Toc345275207)

[Fig. 28 - Registar 43](#_Toc345275208)

[Fig. 29 - Confirmar pagamento 44](#_Toc345275209)

[Fig. 30 - Pesquisar Produto 45](#_Toc345275210)

[Fig. 31 – Leiloar 46](#_Toc345275211)

[Fig. 32 - Comprar 47](#_Toc345275212)

[Fig. 33 - Vender 47](#_Toc345275213)

[Fig. 34 - Leiloar 48](#_Toc345275214)

[Fig. 35 - Licitar 49](#_Toc345275215)

# 4 – Introdução

Este relatório surge no âmbito da realização do trabalho da disciplina

de Desenvolvimento de Sistemas de Software, do curso de Engenharia

Informática da Universidade do Minho. Neste trabalho pretende-se desenvolver uma aplicação informática que suporta o comércio eletrónico, permitindo aos utilizadores comprar e vender produtos.

Desta forma procedemos á análise de requisitos do sistema com base num modelo de domínio, modelo Use-Case com a respetiva especificação textual, digramas de sequência diagrama de classes e diagrama de base de dados

Decidimos alterar um pouco com o previsto pois prescindimos de que o programa suportasse uma funcionalidade de Trocas, a única opção que nos restava era criar outro sistema para os utilizadores fazerem trocas entre si, o que não foi feito pois ia em contra com os objetivos programa de Comercio eletrónico.

O objetivo principal do nosso sistema é permitir aos seus utilizadores vender/comprar/leiloar produtos sem que estes precisem de se preocupar mais, pois o sistema trata de entregar os produtos ao comprador e o valor do produto ao vendedor.

Este relatório descreve então esses passos tomados no desenvolvimento do projecto, servindo assim de apoio a este projecto.

# 5 – Desenvolvimento

## 5.1 – Modelo de Domínio

O modelo de domínio e um esquema detalhado sobre todas as entidades de um grande sistema, e que representa todas as suas interações.

Neste trabalho prático, o desejo foi modelar uma aplicação informática que suporta o comércio eletrónico, que está representado no modelo de domínio na página que se segue (figura1).

Para a sua realização começamos então por analisar o enunciado do projeto onde debatemos ideias sobre as entidades a serem modeladas, tendo em conta as relações entre elas.

Sendo assim no nosso modelo de domínio começamos então por definir a entidade dos utilizadores estendendo-o de forma a capturar toda a essência do contexto do projeto.

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\Vendas.jpg |
| Fig. 1– Modelo de domínio. |

## 5.2 – Modelo de Use-Case

Um diagrama de use-case’s é um diagrama que representa todos os atores envolventes num determinado sistema e, mais importante que isso, todas as ações que poderão executar.

Neste modelo começamos por definir os intervenientes do sistema, designados por atores (User, Visitante, Admin.).

De acordo com o enunciado do projeto o sistema consistirá em duas aplicações, ***front end*** e ***back end,*** sendo a primeira aplicação para acesso ao sistema e a segunda para tarefas de administração. Desta forma tivemos que fazer uma distinção entre as funcionalidades relevantes do ***user*** e do ***visitante*** sendo assim as funcionalidades mais relevantes do Visitante são:

* Pesquisar Produto;
* Registar.

As principais funcionalidades do *user* são:

* Comprar;
* Vender;
* Reportar produto suspeito;
* Pesquisar produto;
* Leiloar;
* Fazer licitação;
* Login;
* Classificar utilizador;
* Adicionar à *whislist*;
* Pedir sugestões;

Relativamente às funcionalidades do *Admin*, este deverá realizar a gestão e manutenção do sistema “como um todo”, sendo algumas das suas principais atividades:

* Login
* Bloquear utilizador;
* Confirmar pagamento;
* Analisar casos suspeitos.
* Confirmar envio produto;

Como ambos os intervenientes do sistema (***User, Visitante, Admin***) são entidades virtuais, temos então o ***Admin*** que pode comportar-se como um ***User***.

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\VendasDSS.jpg |
| Fig. 2 - Diagrama de Use-Case |

## 5.3 – Diagramas de Sequência

Os diagramas de sequência são dos mais importantes diagramas que se apresentam. São eles que mais se aproximam ao código implementado.

Após da elaboração dos ***use cases*** e da sua respetiva especificação, procedemos á elaboração dos diagramas de sequência sendo estes centrados na arquitetura do sistema que estamos a desenvolver, permitindo começar a análise do sistema.

Sendo assim cada um dos diagramas de sequência irá representar os atores que interagem com o sistema, os eventos gerados pelos atores e as respostas geradas pelo sistema.

Inicialmente os diagramas de sequência foram gerados automaticamente porém verificamos que grande parte destes não se encontravam elaborados do melhor modo, desta forma procedemos há sua retificação. Sendo o resultado final as seguintes figuras (figura 18-35) que seguem em anexo.

## 5.4 – Diagrama de Classes

Com base em todos os diagramas aqui apresentados, chegamos ao Diagramade Classes, que consiste na representação de todos os packages do*BuyKing* e das respetivas classes, bem como os seus métodos e atributos.

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\Classes.jpg  Fig. 3 - Diagrama de Classes |

## 5.5 – Diagrama da Base de Dados

De seguida, encontra-se um modelo conceptual da Base de Dados criada para este projeto:

|  |
| --- |
| Fig. 4 - Diagrama Base de Dados |

**5.6 - Decisões**

Devido a conflitos existentes entre as trocas e as outras transações decidimos não implementam o mecanismo de troca no nosso projeto, as principais razões que nos levaram a tomar esta decisão foram:

- Caso uma troca fosse proposta a um produto e esse produto também recebesse uma proposta de venda como a venda tem prioridade iria gerar um conflito entre a troca e a venda podendo o utilizador ficar prejudicado por esse conflito.

Exemplo: O utilizador1 tem um carro velho à venda com o valor de 2000€, o utilizador2 oferece um carro mais recente para troca pois está a fazer coleção de carros antigos, o utilizador3 oferece 2000€ pelo carro. Como o sistema não tem suporte de escolha e a venda é sempre prioritária, a troca iria ficar sempre dependente se o produto recebia uma proposta de compra. O que iria prejudicar o vendedor que poderia querer o trocar o carro antigo pelo carro mais recente.

## 5.6 – Camadas

### 5.6.1 - Camada de Apresentação (Interface do Utilizador)

Apesar de no ano anterior na **Unidade Curricular de LI3** termos feito já uma pequena interface, neste projeto queríamos fazer uma interface mais personalizada, com melhor manuseamento por parte do utilizador e também com uma boa performance. Para isso pesquisamos sobre as utilidades do **java swing** com fim de utilizar as suas funcionalidades para realizar uma interface com as características acima referidas.

**Ícones:** Como é mais fácil para um Utilizador associar um ícone a uma função do projeto, por exemplo a função pesquisar é associada a uma lupa, tentamos colocar ícones sugestivos em toda a interface. Isto vem oferecer um melhor manuseamento e uma interface mais limpa e divertida. Alguns dos ícones foram mesmo desenhados por nós de forma a representarem melhor aquilo que nós queremos transmitir.

**Janelas:** Neste projeto tentamos utilizar o mais possível uma janela de forma a o utilizador não se confundir tanto. Para isso usamos um layout disponibilizado pelo java chamado **CardLayout**, este layout permite alterar os **JPanels** de acordo com o que nós queremos, não criando novas janelas pois estas iriam tornar a interface muito confusa porque teria um grande numero de janelas o que poderia induzir o próprio Utilizador a erro. Depois de alguma pesquisa e alguma prática conseguimos trabalhar com este layout de forma a deixar a interface da forma pretendida, ou seja mais simples e menos confusa.

Tal como pedido procedemos á criação de uma interface gráfica tendo em conta todas as funcionalidades da análise de requisitos.

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Manual\login.png  Fig. 5-Login |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\Interface\Favoritos.png  Fig. 6 - Favoritos |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Manual\depoisLogin.png  Fig. 7 - Principal |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Manual\comoPorVenda.jpg  Fig. 8 - Vender |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\Interface\Historico.png  Fig. 9 – Histórico |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Manual\comoLeiloar.jpg  Fig. 10 - Leiloar |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Manual\Pesquisa Avançada.jpg  Fig. 11 - Pesquisa avançada |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\Interface\produto leilão.png  Fig. 12 - Produto em leilão |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Manual\registar.jpg  Fig. 13 - Registar |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\Interface\produto vender.png  Fig. 14 - Produto à venda |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Manual\pesquisaSimples.png  Fig. 15 – Pesquisa Simples |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\Interface\venderleiloar.png  Fig. 16 - A vender |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Manual\inicio.png  Fig. 17 – Inicio  (Utilizador sem efectuar login) |

### 5.6.2 - Camada de Dados

Como durante as aulas da disciplina não nos foi ensinado como aceder à nossa base de dados, nós tivemos que realizar alguma pesquisa de forma a sabermos como definir os métodos para este problema.

Primeiro como nos foi sugerido, analisamos o **JDBC** do java (*Java Database Connectivity* ou JDBC é um conjunto de classes e interfaces (*API*) escritas em Java que fazem o envio de instruções *SQL* para qualquer banco de dados relacional).

Depois de analisar vimos quais seriam os métodos desta API que podíamos utilizar e que nos iriam oferecer uma maior performance e facilidade de acesso à Base de Dados.

Por fim, conciliamos o nosso código com o código *SQL*, para aceder à base de Dados com o objetivo de obter os resultados pretendidos em cada método através de *selects*, *inserts* e *updates*.

### 5.6.3 - Camada Business

A camada Business e uma camada logica que faz a ligação aos dados e à interface

### 5.6.3.1 – Class BuyKing

|  |
| --- |
| package Business;  import Data.LeiloesDAO;  import Data.ProdutosDAO;  import Data.SuspeitasDAO;  import Data.TrocasDAO;  import Data.UserDAO;  import Data.VendasDAO;  import java.math.BigInteger;  import java.security.MessageDigest;  import java.security.NoSuchAlgorithmException;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.GregorianCalendar;  import java.util.List;  import java.util.logging.Level;  import java.util.logging.Logger;  public class BuyKing {  private UserDAO \_utilizadores;  private VendasDAO \_vendas;  private LeiloesDAO \_leiloes;  private SuspeitasDAO \_suspeitas\_;  private TrocasDAO \_trocas\_;  private ProdutosDAO \_produtos;  public List<Venda> pesquisaVendasSimples(String aPchave, String aCat) throws SQLException  {  return pesquisaVendasAvançada(aPchave, aCat, 0, Float.MAX\_VALUE);  }    public List<Venda> pesquisaVendasAvançada(String aPchave, String aCat, float minP, float maxP) throws SQLException {  //throw new UnsupportedOperationException();  List<Venda> l = \_vendas.getVendasAbertas();  List<Venda> res = new ArrayList<Venda>();  for(Venda v:l)  {  if((v.getPreco()>minP)&&(v.getPreco()<maxP)&&  (aPchave.matches(v.getProduto().getNome())||aPchave.matches(v.getProduto().getDescricao()))  &&(aCat.equals(v.getProduto().getCategoria())))  res.add(v);  }  return res;  }  public List<Leilao> pesquisaLeilao(String aPchave, String aCat) {  throw new UnsupportedOperationException();  }  public boolean login(String aUsername, String aPassword) throws SQLException {  //throw new UnsupportedOperationException();  Utilizador u = \_utilizadores.get(aUsername);  String md5 = BuyKing.md5crypt(aPassword);  return ((u != null) && (u.getPassmd5().equals(md5)));  }  public boolean registar(Utilizador u) throws SQLException {  //throw new UnsupportedOperationException();  return \_utilizadores.add(u);  }  public static String md5crypt(String s) {  MessageDigest md = null;  try {  md = MessageDigest.getInstance("MD5");  } catch (NoSuchAlgorithmException ex) {  Logger.getLogger(BuyKing.class.getName()).log(Level.SEVERE, null, ex);  }  byte[] digest = md.digest(s.getBytes());  BigInteger big = new BigInteger(1, digest);  String md5 = big.toString(16);  while (md5.length() < 32) {  md5 = "0" + md5;  }  return md5;  }  public boolean registaCompra(Utilizador u, Venda v) throws SQLException {  v.setComprador(u);  GregorianCalendar d = new GregorianCalendar();  d.add(GregorianCalendar.DAY\_OF\_MONTH, 7); //7 dias para conclusao da venda  v.setDataLimiteVenda(d);  boolean res = \_vendas.update(v);  return res;  }    public boolean vender(Utilizador u, Produto p, float preco) throws SQLException  {  boolean res=false;  List<Produto> pr = \_produtos.getFromUser(u);  if(pr.contains(p)&&!\_vendas.aVenda(p)&&!\_leiloes.emLeilao(p))  {  Venda v = new Venda(VendasDAO.getNewId(),preco, new GregorianCalendar(), null, null, null,p, u, null);  res=\_vendas.add(v);  }  return false;  }    public boolean licitar(Utilizador u, Leilao l, float v) throws LeilaoFechadoException, BaixaLicitacaoException, SQLException  {  boolean res=false;  if(!l.fechado())  {  Licitacao lit = new Licitacao(u, v, new GregorianCalendar());  res=l.registaLicitacao(lit);  if(res&&(v>=l.getTecto()))  {  GregorianCalendar d = new GregorianCalendar();  d.add(GregorianCalendar.DAY\_OF\_MONTH, 7);  l.setDataLimiteLeilao(d);  \_leiloes.update(l);  }  }  return res;  }    public boolean leiloar(Utilizador u, Produto p, float base, float tecto) throws SQLException  {  boolean res=false;  List<Produto> pr = \_produtos.getFromUser(u);  if(pr.contains(p)&&!\_vendas.aVenda(p)&&!\_leiloes.emLeilao(p))  {  GregorianCalendar hoje = new GregorianCalendar();  GregorianCalendar fecho = (GregorianCalendar) hoje.clone();  fecho.add(GregorianCalendar.DAY\_OF\_MONTH,14);  GregorianCalendar limite = (GregorianCalendar) fecho.clone();  limite.add(GregorianCalendar.DAY\_OF\_MONTH, 7);  Leilao l = new Leilao(LeiloesDAO.getNewId(), u, p, hoje, fecho, limite, null, null, base, tecto);  res=\_leiloes.add(l);  }  return res;  }  } |

# 6 – Conclusão

Ao longo da realização deste trabalho, deparamo-nos com inúmeras dificuldades, que fomos superando à medida que nos tornamos mais familiarizados com a UC.

Como esperávamos a elaboração desta modelação gerou várias discussões, principalmente a nível da escolha das entidades *vs* atributos do modelo de domínio, na escolha correcta das actividades dos actores,???????????????????. Após várias tentativas e optando pela simplicidade, ficamos com os diversos modelos simples e de fácil compreensão.

O grupo conseguiu ter sempre em mente os objectivos a que se propôs e a cumpri-los com sucesso, contribuindo assim para um bom trabalho e uma boa formação individual.

Após a realização deste trabalho, chegamos à conclusão que adquirimos conhecimentos importantes sobre a planificação e modelação de projectos complexos. Os conhecimentos adquiridos durante a realização do projecto e a frequência da Unidade Curricular permitir-nos-ão abordar problemas de uma forma mais metódica e standardizada.

# 7 – Anexos

### 5.2.1 - Use-Cases

**-Tabelas de descrição textual dos use cases:**

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\reportar produto.png |
| Tabela 1 - Reportar produto suspeito |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\login.png |
| *Tabela 2 – login* |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\fazer licitação1.png  C:\Users\marcelo\Dropbox\DSS\tabelas\fazer licitação2.png |
| *Tabela 3 - Fazer licitação* |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\comprar.png |
| Tabela 4 - Comprar |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\analisa caso suspeito.png |
| Tabela 5 - Analisar casos suspeitos |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\pedir sugestões.png |
| Tabela 6 - Pedir sugestões |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\confirma envio produto.png |
| Tabela 7 - Confirmar envio produto |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\vender.png |
| Tabela 8 - Vender |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\bloquear utilizador.png |
| Tabela 9 - Bloquear utilizador |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\adic wishlist.png |
| Tabela 10 - Adicionar à wishlist |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\registar.jpg |
| Tabela 11 - Registar |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\confirma pagamento.png |
| Tabela 12 - Confirmar pagamento |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\pesquisar.jpg |
| *Tabela 13 - Pesquisa produto* |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\leiloar1.png  C:\Users\marcelo\Dropbox\DSS\tabelas\leiloar2.png |
| Tabela 14 - Leiloar |

|  |
| --- |
| C:\Users\marcelo\Dropbox\DSS\tabelas\classificar users.png |
| Tabela 15 - Classificar utilizador |

### 5.3.1 - Diagramas de sequência

**Imagens relativas aos diagramas de sequência:**

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Reportar produto suspeito - Main.jpg |
| Fig. 18 - Reportar Produto Suspeito |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Login - Main.jpg |
| Fig. 19 – Login |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Fazer licitação - Main.jpg |
| Fig. 20 - Fazer Licitação |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Comprar - Main.jpg |
| Fig. 21 - Comprar |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Analisar casos suspeitos - Main.jpg |
| Fig. 22 - Analisar casos suspeitos |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Pedir Sugestões - Main.jpg |
| *Fig. 23 - Pedir Sugestões* |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Confirmar envio de produto - Main.jpg |
| *Fig. 24 - Confirmar envio de produto* |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Vender - Main.jpg |
| Fig. 25- Vender |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Bloquear utilizadores - Main.jpg |
| *Fig. 26 – Bloquear utilizador* |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Adicionar à wishlist - Main.jpg |
| Fig. 27 - Adicionar à wishlist |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Registar - Main.jpg |
| Fig. 28 - Registar |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Confirmar pagamento - Main.jpg |
| Fig. 29 - Confirmar pagamento |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Pesquisar produto - Main.jpg |
| Fig. 30 - Pesquisar Produto |

|  |
| --- |
| C:\Users\Pedro\Dropbox\DSS\prints(atualizado)\Leiloar - Main.jpg |
| Fig. 31 – Leiloar |

### 5.3.2 - Diagramas de sequência dos métodos principais:

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Tarefas2Semana\comprar.jpg |
| *Fig. 32 - Comprar* |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Tarefas2Semana\vender.jpg |
| *Fig. 33 - Vender* |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Tarefas2Semana\leiloar.jpg |
| *Fig. 34 - Leiloar* |

|  |
| --- |
| C:\Users\marcelo\Documents\GitHub\DSS\2-Fase\Tarefas2Semana\licitar.jpg |
| *Fig. 35 - Licitar* |

### 5.6.3.2 – Class Leilão

|  |
| --- |
| package Business;  import Data.LicitacoesDAO;  import java.sql.SQLException;  import java.util.GregorianCalendar;  public class Leilao {    private int id;  private Utilizador leiloador;  private Produto p;  private GregorianCalendar dataLeilao;  private GregorianCalendar \_dataFecho;  private GregorianCalendar \_dataLimiteLeilao;  private GregorianCalendar dataPagamento;  private GregorianCalendar dataEnvioProduto;  private float \_base;  private float \_tecto;  private LicitacoesDAO licitacoes;  public Leilao(int id, Utilizador leiloador, Produto p, GregorianCalendar dataLeilao, GregorianCalendar \_dataFecho, GregorianCalendar \_dataLimiteLeilao, GregorianCalendar dataPagamento, GregorianCalendar dataEnvioProduto, float \_base, float \_tecto) {  this.id = id;  this.leiloador = leiloador;  this.p = p;  this.dataLeilao = dataLeilao;  this.\_dataFecho = \_dataFecho;  this.\_dataLimiteLeilao = \_dataLimiteLeilao;  this.dataPagamento = dataPagamento;  this.dataEnvioProduto = dataEnvioProduto;  this.\_base = \_base;  this.\_tecto = \_tecto;  this.licitacoes=new LicitacoesDAO(id);  }  } |

### 5.6.3.3 – Class Produto

|  |
| --- |
| package Business;  import Data.SuspeitasDAO;  import java.awt.image.BufferedImage;  public class Produto {  private int \_id;  private String \_nome;  private BufferedImage \_imagem;  private String descricao;  private String categoria;  private SuspeitasDAO suspeitas;  public Produto(int \_id, String \_nome, BufferedImage \_imagem, String descricao, String categoria) {  this.\_id = \_id;  this.\_nome = \_nome;  this.\_imagem = \_imagem;  this.descricao = descricao;  this.categoria = categoria;  suspeitas = new SuspeitasDAO(\_id);  }  } |

### 5.6.3.4 – Class Suspeita

|  |
| --- |
| package Business;  public class Suspeita {  private Utilizador \_queixoso;    private String \_just;  public Suspeita(Utilizador \_queixoso, String \_just) {  this.\_queixoso = \_queixoso;    this.\_just = \_just;  } |

### 5.6.3.5 – Class Troca

|  |
| --- |
| package Business;  import java.util.GregorianCalendar;  public class Troca {  private Utilizador \_prop;  private Utilizador \_convidado;  private Produto desejado;  private Produto oferta;  private GregorianCalendar dataProposta;  private GregorianCalendar dataConfirmacao;  private GregorianCalendar dataConclusao;  private GregorianCalendar dataLimite;  private int id;  public Troca(Utilizador \_prop, Utilizador \_convidado, Produto desejado, Produto oferta, GregorianCalendar dataProposta, GregorianCalendar dataConfirmacao, GregorianCalendar dataConclusao, GregorianCalendar dataLimite, int id) {  this.\_prop = \_prop;  this.\_convidado = \_convidado;  this.desejado = desejado;  this.oferta = oferta;  this.dataProposta = dataProposta;  this.dataConfirmacao = dataConfirmacao;  this.dataConclusao = dataConclusao;  this.dataLimite = dataLimite;  this.id = id;  }  } |

### 5.6.3.6 – Class Venda

|  |
| --- |
| package Business;  import java.util.GregorianCalendar;  public class Venda {  private int id;  private float \_preco;  private GregorianCalendar dataVenda;  private GregorianCalendar \_dataLimiteVenda;  private GregorianCalendar dataEnvioProduto;  private GregorianCalendar dataPagamento;  private Produto \_produto;  private Utilizador vendedor;  private Utilizador comprador;      public Venda(int id, float \_preco, GregorianCalendar dataVenda, GregorianCalendar \_dataLimiteVenda, GregorianCalendar dataEnvioProduto, GregorianCalendar dataPagamento, Produto \_produto, Utilizador vendedor, Utilizador comprador) {  this.id = id;  this.\_preco = \_preco;  this.dataVenda = dataVenda;  this.\_dataLimiteVenda = \_dataLimiteVenda;  this.dataEnvioProduto = dataEnvioProduto;  this.dataPagamento = dataPagamento;  this.\_produto = \_produto;  this.vendedor = vendedor;  this.comprador = comprador;  }  } |

### 5.6.3.7 – Class Utilizador

|  |
| --- |
| package Business;  import Data.ClassificacaoDAO;  import Data.FavoritosDAO;  import Data.ProdutosDAO;  import java.awt.image.BufferedImage;  import java.util.GregorianCalendar;  public class Utilizador {  private String \_username;  private String \_passmd5;  private String email;  private String localidade;  private GregorianCalendar dataNascimento;  private GregorianCalendar dataRegisto;  private BufferedImage imagem;  private ClassificacaoDAO \_classificacao;  private FavoritosDAO \_wishlist;  private ProdutosDAO meusProds;      public Utilizador(String name, String pass, String mail, String l, GregorianCalendar dN, GregorianCalendar dR, BufferedImage i)  {  \_username=name;  \_passmd5=pass;  email=mail;  localidade=l;  dataNascimento= dN;  dataRegisto=dR;  imagem=i;  \_classificacao=new ClassificacaoDAO(name);  \_wishlist=new FavoritosDAO(name);  meusProds=new ProdutosDAO();  }  } |

### 5.6.3.8 – Class Licitação

|  |
| --- |
| package Business;  import java.util.GregorianCalendar;  public class Licitacao {    private Utilizador u;  private float v;  private GregorianCalendar data;    public Licitacao(Utilizador user, float valor,GregorianCalendar data)  {  u=user;  v=valor;  this.data = data;  }  } |

### 5.6.3.9 – Class Leilao Fechado Exception

|  |
| --- |
| package Business;  public class LeilaoFechadoException extends Exception{  public LeilaoFechadoException() {  }  public LeilaoFechadoException(String message) {  super(message);  }    } |

### 5.6.3.10 – Class Classificação

|  |
| --- |
| package Business;  import java.util.GregorianCalendar;  public class Classificacao {  public Classificacao(Utilizador classificador, GregorianCalendar data, int valor) {  this.classificador = classificador;  this.data = data;  this.valor = valor;  } |

### 5.6.3.11 - Class Baixa Licitação Exception

|  |
| --- |
| package Business;  public class BaixaLicitacaoException extends Exception{  public BaixaLicitacaoException() {  }  public BaixaLicitacaoException(String message) {  super(message);  }    } |