UNIVERSIDADE POLITÉCNICA

A POLITÉCNICA

Escola Superior De Gestão Ciências E Tecnologias

CURSO DE: Informática de Gestão

TEMA: Sistema de Registo de Activos e Ocorrências de Tecnologia de Informação (TI) para Grandes Empresas: O Caso da Petromoc.

Clério Alfredo Faife

Maputo

2015

UNIVERSIDADE POLITÉCNICA

A POLITÉCNICA

Escola Superior De Gestão Ciências E Tecnologias

CURSO DE: Informática de Gestão

TEMA: Sistema de Registo de Activos e Ocorrências de Tecnologia de Informação (TI) para Grandes Empresas: O Caso da Petromoc.

Estudante: Clério Alfredo Faife

Supervisor: Dr. Stélio Loforte

Maputo, Março de 2015

DECLARAÇÃO DE HONRA

Eu, Clério Alfredo Faife, declaro que este trabalho de fim de curso foi exclusivamente realizado por mim. O mesmo é agora submetido de acordo com todos os requisitos e exigências para obtenção de grau de Licenciatura em Informática de Gestão, na Universidade Politécnica, em Maputo.

Assinatura_____

Data:___/___/_____

Parecer do Tutor

O trabalho de investigação do estudante CLÉRIO ALFREDO FAIFE cujo tema é SISTEMA DE REGISTO DE ACTIVOS E OCORRÊNCIAS DE TECNOLOGIA DE INFORMAÇÃO (TI) PARA GRANDES EMPRESAS: O CASO DA PETROMOC, foi por mim acompanhado durante a sua elaboração.

O estudante mostrou vontade e dedicação tendo apresentado o trabalho várias vezes correspondendo a cada etapa da sua elaboração.

Ter o inventário do parque informático actualizado é preocupação de várias empresas e para algumas médias e grandes empresas uma dor de cabeça.

Evitar grandes paragens no funcionamento do sistema através de detenção de algumas avarias em tempo útil para posterior correcção também se mostra uma tarefa de grande importância nas organizações.

O tema é actual, pertinente e vital importância para organizações que trabalham com Bancos.

O trabalho elaborado apresenta os requisitos necessários para ser submetido para a obtenção do grau de licenciatura.

Maputo, aos 09 de Março de 2015

O Tutor

_____

(Lic. Stélio Rosa Loforte)

DEDICATÓRIA

Dedico primeiramente este trabalho aos meus pais, Alfredo Francisco Faife e Tolita Arone Guiliche, que puseram neste magnífico mundo e que sempre me apoiaram, incondicionalmente, nos meus estudos e na minha vida pessoal. À minha irmã Adélia Suzete Faife e ao meu irmão Flávio Alfredo Francisco Faife que contribuíram muito para minha formação académica. E muito em especial dedico à minha namorada Elizabeth Lídia Alexandre Chiale e ao nosso bebé Mallon Clério Faife.

AGRADECIMENTOS

Em primeiro lugar, agradecer ao docente Dr. Stélio Loforte que logo de primeira aceitou este desafio como tutor, e que sempre se mostrou disponível, para me direccionar na realização desta monografia. Aos colaboradores da Petromoc Dr. Cremildo Bernardo e a Dra. Cremilde Patrício, do Gabinete de Sistemas e Tecnologias que facultaram as informações necessárias que precisei para a realização deste trabalho.

Agradeço igualmente aos meus amigos que coincidentemente são colegas de profissão, Sinai Cuna, Celso César João e Marcel Andela (primo), que me forneceram dicas úteis no âmbito da programação de sistemas informáticos.

ABSTRACTO

Autor: Clério Alfredo Faife

Grau Académico: Licenciatura em Informática de Gestão

Titulo: Sistema de Registo de Activos e Ocorrências de Tecnologia de Informação (TI) para Grandes Empresas: O Caso da Petromoc

Universidade: A Politécnica

Faculdade: Gestão Ciências e Tecnologia

Supervisor da Proposta: Dr. Stélio Loforte

Data 10 de Março de 2015

Palavras chave: Sistema, incidente, ocorrência, activos de TI, assistência *Helpdesk, sockets*, máquinas, rede, base de dados, *java*, *oracle*, inventário, utilizadores, administrador, *software*. Protótipo, grandes empresas, infraestrutura informática, utilizador final, boas práticas.

A tarefa de inventário de Tecnologias de Informação (TI) nas empresas faz parte de um grupo de elementos que suportam os gestores na tomada de decisão. Existem algumas formas de melhorar a relação entre os profissionais de TI e os utilizadores finais de informática quando se trata de incidente.

Em Moçambique os *softwares* de gestão de incidentes ainda não são encarados de forma séria pelos colaboradores das empresas. Um dos factores é forma como estes interagem com os próprios utilizadores.

A empresa Petróleos de Moçambique (PETROMOC) é uma empresa voltada para combustíveis, que está distribuída em todo território moçambicano. Possui uma infraestrutura informática robusta que requere boas práticas de Information Technology Infastruture Library (ITIL). Contudo este trabalho apresenta um protótipo que responde às questões levantadas acima.

ÍNDICE

Lista de Figuras

Lista de Tabelas

Glossário

ArrayList – classe java usada para armazenar um conjunto de objectos.

Begin – palavra reservada usa por algumas linguagens de programação, utilizada para indicar o início de um bloco de instruções.

Bit – é a menor unidade de informação que pode ser armazenada ou transmitida.

Broadcast – é o processo pelo qual se transmite ou difunde determinada informação, tendo como principal característica que a mesma informação está sendo enviada para todos receptores de uma rede ao mesmo tempo.

Byte – é uma unidade de informação digital, um *byte* é equivalente a oito bits.

Datagrama – é uma unidade de transferência básica associada a uma rede de comutação de pacotes em que a entrega, hora de chegada, e a ordem não são garantidos.

Eclipse – é uma *IDE* para desenvolvimento *Java* e para outras linguagens de programação.

End – palavra reservada de algumas linguagens de programação, utilizada para indicar o fim de um bloco de instruções.

Excel – Trata-se de um programa de computador que permite realizar tarefas contabilísticas.

FileStore – classe java que armazena os nomes das unidades de armazenamento.

FileSystem – classe java responsável pelo retorno de directórios

For – Estrutura de controlo usada por linguagens de programação para definir ciclos de instruções.

Foreach – *for* aprimorado ou simplificado (existente em *Java*).

getDefault – método da classe utilitária *FileSystems* que retorna um *FileSystem*.

getFileStore – método que instancia unidades de armazenamento.

getOutputStream – método responsável captura de saída de informação.

getTotalSpace – método que captura capacidade da unidade de armazenamento.

getUsableSpace – método que captura o espaço ocupado da unidade de armazenamento.

Hardware – parte física do computador.

Helpdesk – é um serviço de atendimento aos clientes que buscam solicitações**,** esclarecimentos e soluções**,** para diversos problemas relacionados aos produtos e serviços das empresas**.**

Host – é qualquer máquina ou computador conectado a uma rede, podendo oferecer informações, recursos, serviços e aplicações.

Host name – nome de máquina.

Incidente – episódio repentino que reduz, significativamente, as margens de segurança.

Inputstream – é uma classe abstracta da linguagem de programação Java usada para leitura de dados.

Interface – é uma espécie de classes abstractas que possibilitam outras classes de terem herança múltipla.

Internet – rede informática de acesso público.

Java SE – é uma ferramenta de desenvolvimento para a plataforma Java.

JFrame – classe java responsável exibição da janela no ecrã do utilizador.

Login – termo usado quando se acede um sistema computorizado.

Máscara de rede – é um número de 32 bits usado em um IP para separar a parte correspondente à rede pública, à sub-rede e aos *hosts*.

Microsoft SQL Server – sistema gerenciador de Base de dados desenvolvido pela *Microsoft*.

Microsoft Word – é um processador de texto produzido pela empresa *Microsoft***.**

Ocorrência – o acontecimento, facto sucedido, eventualidade, circunstância, coincidência.

Office 2010 – é um conjunto de aplicativos para escritório que contém programas como processador de texto, planilha de cálculo, base de dados, apresentação gráfica e gerenciador de tarefas, de e-mails e contactos.

OJDBC5.jar – ficheiro responsável pela conexão com base de dados.

OJDBC14.jar – ficheiro responsável pela conexão com base de dados.

OJDBC14_g.jar – ficheiro responsável pela conexão com base de dados.

Oracle 10g Express Edition (Oracle 10 XE) – Sistema gerenciador de base de dados.

Outputstream – classe *Java* que trata de saída/escrita de dados para parte interior e exterior do programa.

Petromoc – Petróleos de Moçambique.

PrintWriter – classe java responsável pela escrita de *stream* de caracteres.

Reader – classe que trata de leitura de dados.

Runnable – *interface* utilizada para transformar uma classe em *Thread.*

Scanner – classe *Java* responsável pela leitura de dados.

Servicedesk – gerenciamento de serviços de TI, que tem como objectivo prover um serviço com qualidade e alinhado às necessidades do negócio, buscando sempre uma redução de custos em longo prazo.

Serversocket – classe que faculta recursos de rede na programação *Java*.

Sockets – classe que faculta recursos de redes na programação *Java*.

Software – é uma sequência de instruções escritas para serem interpretadas por um computador com o objetivo de executar tarefas específicas.

split – método da classe *String* usado para dividir um conjunto de caracteres.

Stored Procedure – procedimentos armazenados dentro das bases de dados.

Stream – é tanto uma fonte de dados como também um destino para dados.

String – classe *Java* que representa conjunto de caracteres.

Telnet – é um protocolo de rede utilizado na Internet ou redes locais para proporcionar uma facilidade de comunicação baseada em texto interactivo bidireccional usando uma conexão de terminal virtual.

Threads – tarefa codificada na e executada no computador.

Ticket – notificação de incidente ou ocorrência.

Trigger – procedimento armazenado que é chamado a cada evento accionado na base dados.

Troubleshooting – resolução de problemas.

Stored procedure – procedimentos armazenados.

Unix – Sistema operativo.

Video-aulas – é uma aula gravada e distribuída em forma de vídeo na internet.

Visio – programa utilizado para documentação de projectos**.**

Windows – Sistema operativo desenvolvido pela empresa Microsoft.

Writer – classe Java que trata de escrita de dados.

Lista de Abreviaturas

API – Application Programming Interface

CIDR – Class less Inter Domain Routing

CPU – Central Processor Unit

DCL – Data Control Language

DDL – Data Definition Language

DML – Data Manipulation Language

FK – Foreign Key

FTP – File Transfer Protocol

HTTP – Hypertext Transfer Protocol

IDE – Integrated Development Environment

I/O – Input/Output

IP – Internet Protocol

IPv4 – Internet Protocol Version 4

IT – Information Technology

ITIL – Information Technology Infrastructure Library

JDBC – Java Data Base Connectivity

PETROMOC – Petróleos de Moçambique

PK – Primary Key

PL/SQL – Procedural Language for Structured Query Language

S/d – Sem data

SGBD – Sistema Gerenciador de Base de Dados

SQL – Structured Query Language

TCP/IP – Transmission Control Protocol/Internet Protocol

TI – Tecnologia de Informação

UML – Unified Modeling Language

XP – Extreme Programming

CAPITULO I - Introdução

Como é do nosso conhecimento o bem mais importante de uma organização é a sua informação e nada melhor que tê-la de uma forma organizada, protegida, actualizada, íntegra e acessível.

Um grande desafio que os departamentos de informática das grandes empresas têm enfrentado a cada ano é a consolidação de inventário de activos de informática. Esta tarefa tem consumido muito tempo de execução, dependendo da complexidade de informação exigida para se levantar e do tamanho da sua infraestrutura.

Outra questão levantada pelos utilizadores de informática é a forma como o Serviço de Atendimento aos Utilizadores (*helpdesk*) interagem perante os incidentes e ocorrências que habitualmente se encontram envolvidos. Na maioria dos casos nota-se atrasos ou falta de empatia para com os mesmos.

Algumas empresas têm aplicativos informáticos (*softwares*) que permitem os utilizadores abrirem "chamados" vulgo *tickets* quando existe um incidente. Desta forma os *helpdesk* são obrigados a agir sobre a situação, mas estes aplicativos geralmente não são automatizados, isto é, a detenção do incidente é identificada por um utilizador final ou profissional de Tecnologia de Informação (TI) e não pelo *software*. Portanto, o desfecho do incidente é editado por um colaborador e não pelo *software*, estando propenso a manipulações, como é o caso verificado na Petróleos de Moçambique (PETROMOC).

A proposta deste trabalho é a criação de uma ferramenta que detecta activos de TI e ocorrências/incidentes dentro deles, por forma a notificar e editar o seu estado, colocar como resolvido quando realmente estiver resolvido automático e manualmente dependendo do caso, interagindo directamente com os dispositivos. Numa primeira fase, tratando-se de um protótipo, o programa vai-se focalizar em computadores e seus elementos internos.

O esperado do sistema é possuir capacidade de inventariar a informação das máquinas na rede, sendo, assim, uma plataforma de inventário em tempo real e sempre actualizando toda informação dos dispositivos que se encontram na rede da organização, identificando alterações. Desta forma, notificará o utilizador do sistema sobre o estado de cada unidade e deverá fazer o relatório actual de toda a informação existente sempre que o utilizador do sistema solicitar.

O registo das alterações ou modificações dos activos no domínio ajuda os profissionais de TI a responderem com melhor *performance* os incidentes que ocorrem dia-a-dia na organização, o que torna visível o impacto e as prioridades dos mesmos. Assim sendo, os incidentes são assistidos gradualmente, de acordo com o nível de urgência e em última instância, os gestores podem extrair relatórios de um período e visualizar as fragilidades e pontos críticos da infraestrutura informática da sua empresa.

Torna-se muito mais fácil para os gestores tomar decisões futuras que visam melhorar o desempenho da arquitectura informática, devido à visibilidade aprimorada do sistema.

2. Objectivos

2.1. Objectivos Gerais

Desenvolver um protótipo de sistema que inventaria equipamento informático e controla incidentes de TI para a PETROMOC.

2.2. Objectivos Específicos

- Aprimorar os processos existentes na recolha de inventário a nível da organização de forma eficiente e automática;
- Garantir que haja uma visão ampla da infra-estrutura informática organizacional para ajudar na tomada de decisão;
- Permitir que seja visível o histórico de ocorrências dentro da empresa a nível de TI;
- Garantir que o serviço de *helpdesk* tenha desempenho personalizado e a resolução de problemas (*trouble-shooting*) seja priorizada pelo seu nível de gravidade e que o tempo de *trouble-shooting* se torne reduzido;
- Melhorar o desempenho do Gerenciamento de Serviços de TI (*Servicedesk*) na assistência aos utilizadores finais de TI.

2.3. Problema

Quais os motivos que provocam a insatisfação dos utilizadores finais de informática em relação à assistência *helpdesk,* mesmo existindo aplicativos de registo de ocorrências/incidentes nas organizações?

2.4. Hipóteses H0 e H1

H0 – Os *helpdesk* preocupam-se em seguir os procedimentos dos aplicativos de TI e o suporte ao utilizador é relegado para o segundo plano.

H1 – Os *helpdesk* não se preocupam em seguir os procedimentos dos aplicativos de TI e suporte ao utilizador não é relegado para o segundo plano.

2.5. Justificativa

A importância deste tema verifica-se a partir da sua aplicação na área profissional, pois não só a PETROMOC poderá usar o sistema, assim como qualquer outra empresa que disponha

de equipamento informático que pretenda monitorar a sua infraestrutura para uma melhor gestão e controlo de activos.

Desde modo, o inventário de TI realizar-se-á em curto espaço de tempo e com uma margem de falhas de registo muito reduzida. Visto que se trata de um sistema automatizado, a sua implementação ajuda na rapidez dos relatórios de infraestrutura informática e a visão dos incidentes que ocorrem a cada período é ainda mais personalizada.

2.6. Delimitação do Tema

Este trabalho tem como foco documentar o *software* e o seu protótipo que é testado num laboratório montado para esta investigação.

Trata-se de um protótipo desenvolvido pelo Sistema Gerenciador de Base de Dados (SGBD) *Oracle 10g Express Edition* e em Java *Standard Edition* versão 1.7. O ambiente de desenvolvimento foi em *Windows*, para testar, foi criada uma rede de computadores em que protótipo faz inventário e detecta incidentes e ocorrências, dando a possibilidade de o utilizador final participar.

CAPITULO II - Leitura Bibliográfica

Este capítulo consiste na revisão da literatura de vários autores que se debruçam sobre o tema em análise, o que serviu de base para a realização desta monografia.

1.1. Definição dos Conceitos de Acidente, Desastre, Incidente e Ocorrência

Ao abordar o termo *acidente*, Lourenço (2010) descreve-o como sendo o acontecimento repentino e imprevisto, provocado pela acção do homem ou da natureza, com danos significativos e efeitos muito limitados no tempo e espaço, susceptíveis de atingir pessoas, bens e o ambiente, implicando a revisão de modelos.

Por seu turno, o mesmo autor define *desastre* como sendo o acontecimento súbito ou extraordinário concentrado no tempo e no espaço, que provoca prejuízos severos na vida do indivíduo, afectando as principais funções da sociedade em determinada área, obrigando a repensar tudo, em função da gravidade, desde as finalidades (acidente grave), as regras (catástrofe) até aos sistemas de valores (calamidades).

Para este autor, o fenómeno *incidente* consiste no episódio repentino que reduz, significativamente, as margens de segurança sem, contudo, anulá-las, apresentando apenas, potencias consequências para a segurança, levando a uma actualização de base de dados sem, no entanto, acrescentar uma revisão de modelos, finalidades, regras e valores.

Por fim, o autor considera *ocorrência* como sendo o acontecimento, facto sucedido, eventualidade, circunstância, coincidência, falso alarme que origina a mobilização dos meios de bombeiros.

1.2. Abordagem Sobre Incidente

Nas grandes organizações é comum a prática de ITIL[1] e, em várias situações os colaboradores aplicam sem a noção de estar a usar a boa prática, visto que, na maioria das vezes alguns processos são ocultados. Carvalho (2011) descreve como devem ser sequenciados os processos na operação de serviço em caso de identificação de um incidente, conforme mostra a figura.

[1]=O modelo ITIL busca promover a gestão com foco no cliente e na qualidade dos serviços de tecnologia da informação (TI). O ITIL lida com estruturas de processos para a gestão de uma organização de TI apresentando um conjunto abrangente de processos e procedimentos gerenciais, organizados em disciplinas, com os quais uma organização pode fazer sua gestão táctica e operacional em vista d alcançar o alinhamento estratégico com os negócios.

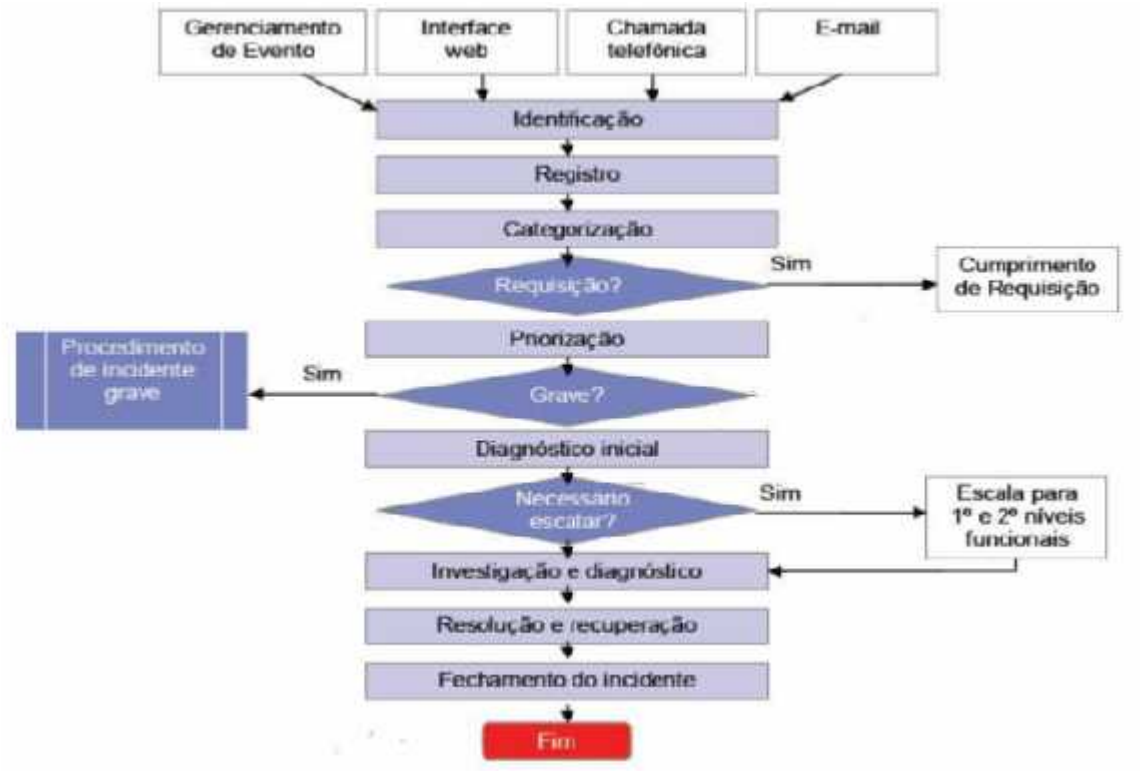

Figura 1- Processos de tratamento de incidente - Carvalho (2011)

1.3. Metodologia de Desenvolvimento Software

São as formas padronizadas de desenvolvimento. Dentre as várias metodologias Soares (s/d) argumenta que as metodologias Ágeis têm se usado como uma alternativa às abordagens tradicionais para o desenvolvimento de *software*. As metodologias tradicionais, conhecidas também como pesadas ou orientadas a planeamentos, devem ser aplicadas apenas em situações em que os requisitos do sistema são estáveis e requisitos futuros são previsíveis.

Este autor acrescenta que dentre todas as metodologias ágeis já existentes, a que vem se destacando em número de adeptos e projectos é a *Extreme Programming* (XP). As metodologias ágeis surgiram com a proposta de aumentar o enfoque nas pessoas e não nos processos de desenvolvimento. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com resolução de problemas de forma iterativa.

E concretamente para este trabalho a metodologia usada foi a *Extreme Programming* (XP), pois existe a preocupação de focalizar mais com resolução de problemas de forma iterativa com os futuros utilizadores do sistema.

A metodologia XP enfatiza o desenvolvimento rápido do projecto e visa garantir a satisfação do cliente, além de favorecer o cumprimento das estimativas. As regras, práticas e valores da

XP proporcionam um agradável ambiente de desenvolvimento de software para os seus seguidores, que são conduzidos por quatro valores: comunicação, simplicidade, *feedback* e coragem. (Libardi et al,2010). Os passos desta metodologia são:

- Planeamento/Escutar – XP é baseado em comunicação, menor importância na documentação formal, maior necessidade de uma comunicação verbal de qualidade. Para esta pesquisa, esta fase consistiu na comunicação com os colaboradores da Petromoc, escutando as dificuldades encontradas na prática e visando como o sistema proposto poderia resolver ou minimizar. Este processo foi contínuo durante o desenvolvimento com inúmeras chamadas telefónicas, troca de emails e visitas à sede da Petromoc em Maputo.
- Projecto ("Designing") – não é estático, aceita evolução natural do sistema, à medida que foi se desenvolvendo o sistema, notou-se uma “transformação” na sua modelação, isto é, alguns diagramas foram alterados com tempo.
- Codificação – fase da escritura do código, até que o sistema considera-se “acabo”, a codificação foi feita *Java* e em SGBD *Oracle*.
- Testes – é um passo integrado no processo de desenvolvimento, os testes foram feitos à medida que foi se terminando trechos de código, para assegurar que no final do trabalho a margem de erros fosse reduzida. Por exemplo: o módulo que percorre as máquinas na rede.

1.4. Diagramas de UML

A modelação de sistemas evolui juntamente com os paradigmas de programação, ao tratar do conceito de Modelo, Souza (2013) define como sendo a maneira de projectar, comunicar, documentar soluções computacionais.

O diagrama de *Unified Modeling Language* (UML) é descrito por Souza (2013), como sendo o padrão de facto para especificar, visualizar, documentar e construir artefactos de um sistema desenvolvido sob o paradigma Orientado a Objectos.

Para este autor, existem diferentes tipos de diagramas para a versão 2.4.1, nomeadamente, diagramas de Casos de Uso, de Classes, de Objectos, de Estrutura Composta, de Sequência, de Comunicação, de Estados, de Actividades, de Componentes, de Implantação, de Pacotes, de *Interface* Geral e de Tempo.

Por seu turno, Guedes (2010) afirma que o Diagrama de Caso de Uso é o mais genérico e auxilia no levantamento de análise de requisitos, sendo neste diagrama onde se compreendem as necessidades do utilizador, que vê o sistema como um todo, embora venha a ser consultado durante todo o processo de modelagem e sirva de base para todos os outros diagramas.

Adiante o autor acrescenta que o diagrama de *Casos de Uso* apresenta uma linguagem simples e fácil de perceber para que os utilizadores possam ter uma visão geral de como o sistema se comporta. O diagrama procura identificar os actores (utilizadores, outros *softwares* que interagem com o sistema ou até mesmo algum *hardware*[2] especial), que utilizarão de alguma forma o *software*, bem como os serviços, ou seja, as opções que o sistema disponibilizará aos actores, conhecidos neste diagrama como Casos de Uso.

Ainda na sequência da descrição dos diagramas UML o autor caracteriza o diagrama de Classes como sendo o diagrama mais utilizado e o mais importante da UML, servindo de apoio para a maioria dos outros diagramas. Como o próprio nome diz, este diagrama define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos possuídos por cada classe, além de estabelecer como as classes se relacionam e trocam informações entre si.

Por fim Guedes (2010) acrescenta que o diagrama de Sequência preocupa-se com a ordem temporal em que as mensagens são trocadas entre os objectos envolvidos num determinado processo. Em geral, baseia-se num Caso de Uso definido pelo diagrama de mesmo nome e apoia-se no Diagrama de Classes para determinar os objectos das classes envolvidas num processo, bem como os métodos disparados entre os mesmos. Um Diagrama de Sequência costuma identificar o evento gerador do processo modelado, bem como o actor responsável por este evento, e determina como o processo deve se desenrolar e como pode ser concluído por meio do envio de mensagens, que em geral disparam métodos entre os objectos.

1.5. Linguagem de Programação Java

1.5.1 Threads

Segundo Jaques (2007) programação *Multithread* (multi-tarefa) é um paradigma conceitual da programação pelo qual se divide os programas em dois ou mais processos que podem ser executados paralelamente. Muitas vezes, o programa não exige recursos completos do

[2] *Hardware* – parte física do computad .

computador. Por exemplo, o computador pode levar um minuto para ler os dados do utilizador do teclado, mas o tempo em que a CPU[3] estará envolvida nesse processo é mínimo.

Ainda na abordagem sobre o *multiprocessamento*, a autora ressalta que para determinar que certas classes de um programa possam correr paralelamente, é necessário definir como *thread*[4]. Para tal, podemos definir de duas maneiras: implementando a *interface Runnable*[5] ou derivando da classe *Thread*. Toda *thread* deve ter um método *run*()[6]. É neste método que é iniciada a sua execução. Podemos pensar no seu método *run*() como o equivalente ao método *main*()[7] para as classes. Uma *thread* inicia a sua execução pelo método *run*(), assim como uma aplicação Java inicia a sua execução pelo método *main*().

1.5.2. Pacote I/O – Input/Output (E/S – Entrada e saída) da linguagem de programação Java

O envio e recepção de mensagens entre as máquinas na rede neste protótipo foram, também suportados pelas classes do pacote *IO* da linguagem de programação Java. Neste contexto, Jaques (2007) afirma que, a maioria dos programas precisam aceder e enviar dados externos. Os dados de entrada podem ser provenientes de um arquivo em disco, de um teclado ou de uma conexão de rede. Da mesma maneira, o destino de saída pode ser um arquivo em disco ou uma conexão em rede. Java permite lidar com todos os tipos de entrada e saída através de uma abstracção conhecida como *stream*, que é tanto uma fonte de dados como também um destino para dados. O pacote *java.IO* define um grande número de classes para ler e escrever *streams*. As classes *InputStream*[8] e *OutputStream*[9], bem como as suas subclasses, são usadas para ler e escrever *stream* de *bytes*[10], enquanto as classes *Reader*[11] e *Writer*[12] são usadas para ler e escrever *streams* de caracteres.

[3] Central processor unit (unidade central de processamento).
[4] Tarefa a ser executada pelo CPU.
[5] Uma espécie de classe 100% abstracta da linguagem programação, utilizada para transformar classes normais em *Threads.*
[6] Método onde devem ser colocadas as instruções a serem executadas pela classe *Thread*.
[7] Método principal de uma classe onde as instruções são executadas quando programa está em execução.
[8] Classe abstracta da linguagem de programação Java usada para leitura de dados.
[9] Classe Java que trata de saída/escrita de dados para parte interior e exterior do programa.
[10] É uma unidade de informação digital, um byte é equivalente a oito bits.
[11] Classe que trata de leitura de dados.
[12] Classe Java que trata de escrita de dados.

1.5.3. Sockets (Transmission Control Protocol) TCP

Socket é um mecanismo de comunicação (de dois sentidos) entre dois programas se comunicar numa rede informática normalmente.

Tratando de programação para redes este autor que temos vindo a citar refere que os *sockets Transmission Control Protocol/Internet Protocol* (TCP/IP), são usados para programar conexões confiáveis, bidireccionais e ponto a ponto, com base em um *stream* entre computadores. Os *sockets* podem ser usados para conectar o sistema de E/S de um programa Java a outros programas em outras máquinas na rede informática de acesso público *(Internet).* Para estabelecer uma conexão via *socket* entre duas máquinas é necessário que uma máquina contenha um programa que fique esperando por uma conexão e a outra tenha um programa que tente se conectar a primeira.

Por último Jaques (2007) afirma que existem duas classes principais: *ServerSocket* que fica ouvindo a porta à espera que um cliente se conecte a ele. *Socket* cliente que se conecta a um servidor para comunicação. Assim como para fazer uma ligação precisamos do número de telefone de uma pessoa, para um programa se conectar a uma máquina remota, ele precisa saber o endereço na *internet* daquele programa que é o endereço *Internet Protocol* (IP). Além disso, é necessário também especificar o número de porta do programa servidor. Os números de porta em *TCP/IP* são números de 16 *bits*[13] que variam de 0-65535. Na prática, portas com número menor que 1024 são reservadas para serviços pré-definidos, tais como *File Transfer Protocol* (FTP), *telnet*[14] e *Hypertext Transfer Protocol (http),* por isso, não devem ser usadas.

O construtor *ServerSocket* cria um servidor que fica esperando por conexão de um cliente. Quando um cliente requisita uma conexão, o servidor abre um *socket* com o método *accept*()[15]. O servidor se comunica com o cliente usando *InputStream* (recebe dados) e *OutputStream* (envia dados). O servidor fica continuamente lendo as mensagens do cliente até que a execução seja finalizada. Neste caso, fica esperando a conexão de um novo cliente. Para finalizar o servidor é necessário teclar CTRL+C, já que este está preso em um laço infinito a espera de novas conexões.

[13] É a menor unidade de informação que pode ser armazenada ou transmiti .

[14] É um protocolo de rede utilizado n Internet ou redes locais para proporcionar uma facilidade de comunicação baseada em texto interactivo bidireccional usando uma conexão c terminal virtual**.**

[15] Método que facilita conexões entre *sockets.*

1.5.4. Conexão com base de dados

O *Java Database Connectivity* (JDBC) é uma API que permite a um programa Java aceder uma base de dados, ou seja, uma interface entre a linguagem Java e a linguagem que as outras bases de dados suportam. A maior vantagem de JDBC sobre os outros ambientes de programação de base de dados é que os programas desenvolvidos em Java com JDBC são independentes de plataforma ou da base de dados. Isso significa que um programa Java que aceda a uma base de dados pode funcionar, tanto em *Windows* como *Unix*[16], assim como pode aceder a qualquer base de dados, como *Oracle, o Microsoft SQL Server* [17]e outros. Mas como resolver o problema de Java aceder diferentes bancos de dados com diferentes protocolos de comunicação. É impossível fazer com que Java aceda a base de dados directamente com "Java puro", pois existem muitas bases de dados e, por isso, muitos protocolos que deveriam ser implementados. (Jaques, 2007).

1.6. Base de Dados

Para garantir que os dados fiquem armazenados permanentemente na memória do computador, os desenvolvedores guardam a informação numa base de dados. Greenberg (2004) afirma que num sistema eficiente os dados são divididos em entidades ou categorias discretas. Um modelo Entidade Relacionamento é uma ilustração de várias entidades numa empresa e dos relacionamentos entre elas.

O mesmo autor refere que cada linha de dados de uma tabela é identificada com exclusividade por uma *Primary key* (PK) Chave Primária. É possível relacionar logicamente dados de várias tabelas através de *Foreign Key* (FK) Chave Estrangeira. Uma chave estrangeira é uma coluna ou um conjunto de colunas que faz referência a uma chave primária na mesma tabela ou em outra tabela. Esta estrutura é criada e mantida por uma linguagem específica para base de dados *Structured Query Language* (SQL)

Para cada propósito existem subtipos de linguagens dentro desta linguagem, elas se diferenciam em:

a) *Data Definition Language* (DDL) - utilizada para criar, apagar e alterar objectos como *views*, *databases*, *stored procedures*, etc.
b) *Data Control Language* (DCL) - permite controlar a segurança de dados definindo quem pode aceder cada operação em cada objecto.

[16] *Unix* e *Windows* - Sistemas operativo para computadores.

[17] *Oracle, o Microsoft SQL Server* — Sistemas Gerenciadores de Base de Dados (SGBD).

c) *Data Manipulation Language* (DML) - permite a manipulação de dados.

Júnior (2006) acrescenta que o perfil de uso de base de dados relacionais foi evoluindo e com o passar do tempo, a estrutura SQL já não respondia às necessidades. Agregou-se então extensões ao SQL que pretendiam responder às novas necessidades, adicionando comandos processuais à linguagem declarativa do SQL. No caso da *Oracle* essa extensão foi chamada de *Procedural Language for Structured Query Language* (PL/SQL)

Para algumas versões antigas do SGBD *Oracle* o auto incremento do índice das linhas dentro tabelas é feito de forma manual, isto é, não existe uma função de auto incremento para estes sistemas. Desta forma, verifica-se a funcionalidade das *triggers*[18], as quais Júnior (2006) define como sendo tipos especiais de *stored procedures*[19] que são activados, automaticamente, como efeito de um comando de *DML* numa tabela. As *triggers* não podem ser chamadas de forma explícita ou passar parâmetros dentro delas. Não é possível associar uma *trigger* a qualquer objecto que não seja uma tabela. Uma *trigger* pode ser activada por qualquer um dos comandos de *DML*.

1.7. Redes de Computadores

As redes de computadores permitem que as máquinas se comuniquem entre si e também partilhem informação, para tal, é necessário que cada máquina tenha uma configuração de rede. Essa configuração é baseada em um endereço de *Internet Protocol* (IP) que é uma espécie de identificador da máquina dentro de uma rede e, máscara de rede que define o escopo da rede. O endereço IP tem 4 octetos e a sua máscara pode ser representada em forma de barra (/).

Abaixo temos uma legenda do intervalo do IP em parâmetros onde Costa, J. (2010) descreve:

CIDR - Bloco de Endereços	**Descrição**
0.0.0.0/8	Rede corrente (só funciona como endereço de origem)
10.0.0.0/8	Rede Privada
14.0.0.0/8	Rede Pública
39.0.0.0/8	Reservado
127.0.0.0/8	*Localhost* (máquina local)
128.0.0.0/16	Reservado
172.16.0.0/12	Rede Privada

[18] Procedimento armazenado que é chamado a cada evento accionado na base dados.

[19] Procedimentos armazenados dentro das bases de dados.

191.255.0.0/16	Reservado
192.0.2.0/24	Documentação
192.88.99.0/24	*IPv6* para *IPv4* (versões de IP)
192.168.0.0/16	Rede Privada
198.18.0.0/15	Teste de redes
223.255.255.0/24	Reservado
224.0.0.0/4	*Multicast* (rede Classe D)
240.0.0.0/4	Reservado (rede Classe E)
255.255.255.255	*Broadcast (*transmissão de informação por difusão*)*

De acordo com Costa (2010) o conceito de máscara de endereçamento é uma abordagem existente no protocolo IP com o objectivo de melhorar o desempenho no roteamento dos *datagramas*[20]. Uma máscara é uma técnica que ajuda a determinar se o endereço de um *datagrama* é local ou se precisa de um roteamento para uma outra rede. Aplicando um *AND* lógico com os endereços da máscara e do *datagrama*, fazemos uma eliminação do endereço físico. Em outras palavras, resta apenas o endereço de rede. Depois deste resultado, fica fácil saber se é necessário ou não efectuarmos um roteamento do *datagrama*.

Para definir o número de máquinas desejadas em uma determinada rede, tendo a ideia de custo e melhor desempenho, realizam-se cálculos de *IP*, com esse procedimento conseguimos ajustar a rede de forma mais adequada. Podemos expressar o número de sub-redes possíveis com a fórmula:

Número de sub-redes = 2M

Onde: M é o número de *bits* usados para definir a sub-rede ou o número de *bits* de *hosts*[21] cobertos pela máscara. Também podemos calcular o número de *hosts* por sub-rede com uma fórmula similar:

Número de *hosts* por sub-rede = 2U - 2

Onde: U é o número dos *bits* de *hosts* restantes ou *bits* de *hosts* não-cobertos pela máscara (quantidade de zeros). Devemos tirar 2, pois o primeiro e último endereço são reservados para a rede e para o *Broadcast*. Costa, J. (2010).

[20] É uma unidade de transferência básica associada a uma rede de comutação de pacotes em que a entrega, hora de chegada, e a ordem não são garantidos.

[21] É qualquer máquina ou computador conectado a uma rede, podendo oferecer informações, recursos, serviços e aplicações**.**

1.8. Validação de Dados

Diferentes factores podem comprometer a confiabilidade de dados anemométricos medidos em campo, podendo alterar as medições, tornando-as incorrectas. A disponibilização de dados de torres anemométricas requer que estes sejam submetidos a um rigoroso processo de controlo de qualidade, cujo objectivo é identificar dados inconsistentes ou suspeitos. O desenvolvimento de um conjunto de critérios de validação destes dados é elemento crítico de sua avaliação. Amanajás, J., et al. (2013).

Os parâmetros de validação de métodos analíticos envolvem Especificidade/Selectividade, Função da Resposta (gráfico analítico), Intervalo de Trabalho, Linearidade, Sensibilidade, Exactidão, Precisão (repetitividade, precisão intermediária e reprodutividade), Limite de Detecção (LD), Limite de Quantificação (LQ) e Robustez. Brito, N., et al. (2003).

CAPÍTULO III - Metodologia

1.1. Metodologia de Pesquisa

Para a presente pesquisa foi utilizada como técnica de Colecta de Dados (o Questionário) para o levantamento da informação da infraestrutura da PETROMOC, e observou-se como o *software* se comporta num ambiente similar.

Durante o processo de elaboração da pesquisa, tomou-se como exemplo a estrutura informática da PETROMOC, onde se fez o levantamento dos seus activos durante o processo de estágio pré-profissional, a nível dos departamentos nas cidades de Maputo e Matola.

A planilha em Excel do inventário de activos TI da PETROMOC serviu de modelo para desenvolver a base de dados. Tendo em conta a existência um *Data Center* (centro de dados) localizado na Sede da empresa com vários servidores físicos e lógicos, baseou-se nas partes que compõem um computador, nomeadamente, unidades de processamento, unidades de memória, entrada e saída de dados, para a criação do protótipo.

1.2. Processo de Investigação

As disciplinas de Programação, Base de Dados, Redes e Análise de Sistemas leccionadas no curso de Informática de Gestão da Universidade Politécnica, foram os “pilares” para a construção do projecto.

1.2.1. Fase I - Levantamento de Requisitos

Consistiu na observação, colecta de dados via questionário, análise de fluxo de informação actual, incluindo os objectivos fundamentais do sistema e seus agentes envolvidos.

1.2.2. Fase II - Leitura Bibliográfica

Esta fase incidiu-se na leitura de obras relacionadas com o tema, assim como outras áreas complementares, nomeadamente livros em suporte físico, *vídeo-aulas* e inúmeras apostilas acedidas pela internet.

1.2.3. Fase III: Desenvolvimento do Projecto

Esta fase dedicou-se à programação do *software*, tendo-se recorrido às seguintes ferramentas:

- Sistema Gerenciador de Base de Dados (SGBD) Oracle 10g XE;

- *Integrated Development Enviroment* (IDE) Eclipse, versão Kepler, usando códigos Java SE 1.7;
- Projecto em Microsoft Word do Office 2010 com auxílio do programa Microsoft *Visio 2010* para a modelação de figuras e diagramas em ambiente Windows.

CAPÍTULO IV- Desenvolvimento

1.1. Diagramas de UML

Os Diagramas de UML - *Unified Modeling Language* - foram utilizados para modelar esta estrutura de *software* utilizando o conceito padrão de Orientação a Objecto (OO). Esta não é uma ferramenta de desenvolvimento de aplicativos, mas sim uma linguagem que apoia a engenharia de *software*.

Para descrever os processos e actividades que correm dentro protótipo foram seleccionados alguns destes diagramas para detalhar como o *software* e o seu meio envolvente interagem no seu escopo:

1.1.1. Diagrama de Caso de Uso

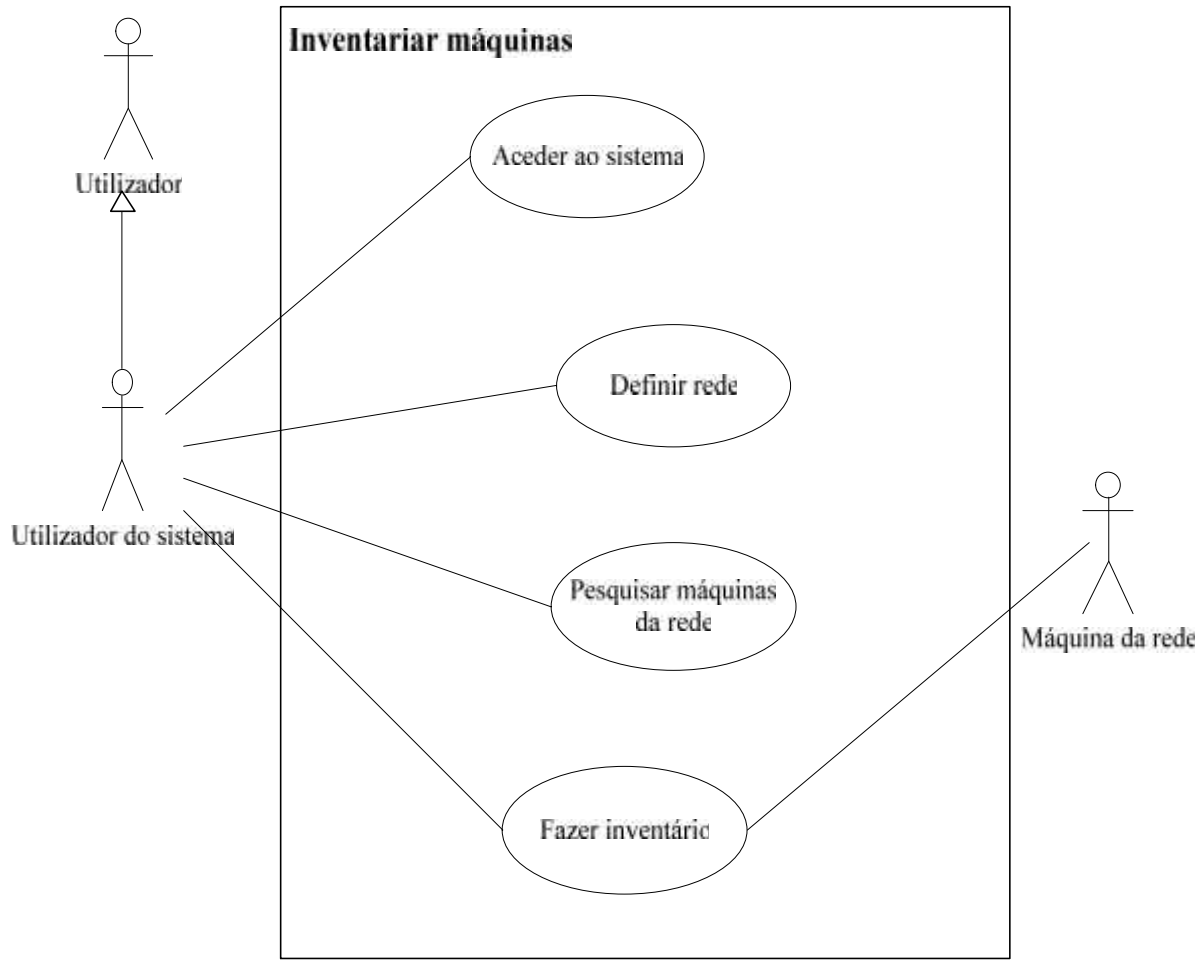

Figura 2. Caso de Uso – Aprendendo Infra-estrutura.

Identificação: UC001	
Nome: Aceder ao sistema	
Actores: Utilizador do sistema	
Tipo: Primário	
Pré-condição: Utilizador cadastrado na base de dados do sistema	
Pós-condição: Acesso para o sistema concebido ao utilizador	
Fluxo principal	
Actor	**Sistema**
1. Abre o programa	2. Exibe janela de acesso
2. Introduz dados na janela de acesso	4. Autentica utilizador
3. Pressiona o botão "Entrar"	5. Disponibiliza os seus recursos ao utilizador
Fluxo alternativo	
4. a) Dados incorrectos	
1. Informar para introduzir novamente	

Identificação: UC002	
Nome: Definir rede	
Actores: Utilizador do sistema	
Tipo: Primário	
Pré-condição: Acesso ao sistema concebido ao utilizador	
Pós-condição: Máscara e endereço de rede definidos no sistema pelo utilizador com sucesso	
Fluxo principal	
Actor	**Sistema**
1. Pressiona no botão "definições de rede"	2. Mostra janela de definições de rede
3. Introduz endereço de rede	6. Exibe intervalo de IP's
4. Introduz máscara de rede	
5. Pressiona no botão "Mostrar intervalo"	
7. Fecha a janela	
Fluxo alternativo	
5. a) Máscara de rede ou endereço de rede introduzidos incorrectamente	
1. Introduza novamente	

Identificação: UC003	
Nome: Pesquisar máquinas da rede	
Actores: Utilizador do sistema	
Tipo: Primário	
Pré-condição: Máscara e endereço de rede definidos no sistema com sucesso	
Pós-condição: Nome das máquinas na rede identificadas pelo sistema	
Fluxo principal	
Actor	**Sistema**
1. Pressiona botão "Pesquisar"	2. Retorna a relação dos nomes das máquinas e *IP's* de rede
Fluxo alternativo	
1. a) Teste de conectividade falhou	
1. Tente novamente	

Identificação: UC004	
Nome: Fazer inventário	
Actores: Utilizador do sistema, Máquina da rede	
Tipo: Primário	
Pré-condição: Nome das máquinas identificadas e ligadas à rede	
Pós-condição: Inventário concluído com sucesso	
Fluxo principal	
Actor	**Sistema**
1. Pressiona botão "Inventariar"	2. Solicita detalhes das máquinas da rede
3. Envia dados ao sistema	4. Recebe dados
5. Navega pelas máquinas no sistema	6. Exibe dados das máquinas navegadas no sistema
7. Pressiona botão "gravar"	8. Insere informação na base de dados
Fluxo alternativo	
2 a) Máquina desconectada	
1. Não possível estabelecer comunicação com a máquina	
8. a) Erro ao gravar	
1. Não foi possível estabelecer comunicação com a base de dados	

As ocorrências e incidentes detectados pelo sistema no processo de inventário são: ficheiros que apresentam insegurança para a máquina, espaço de armazenamento insuficiente no disco duro e processos “encravados” no sistema operativo. Segundo a definição de Lourenço (2010), para este trabalho podemos classificar incidente como todos os ficheiros que apresentam insegurança para a máquina, e ocorrência o espaço insuficiente de disco e tarefas pendentes no sistema operativo.

Todos estes dados são adquiridos via o *Windows* que fornece a aplicação, por meio de uma instrução Java *Runtime.getRuntime.exec(“*comando*”).* Esta instrução retorna um processo (*Process*) que pode ser lido pelas classes e métodos do pacote *I/O* Java. A instrução acima referenciada aceita qualquer comando executável na linha de comando *Windows*.

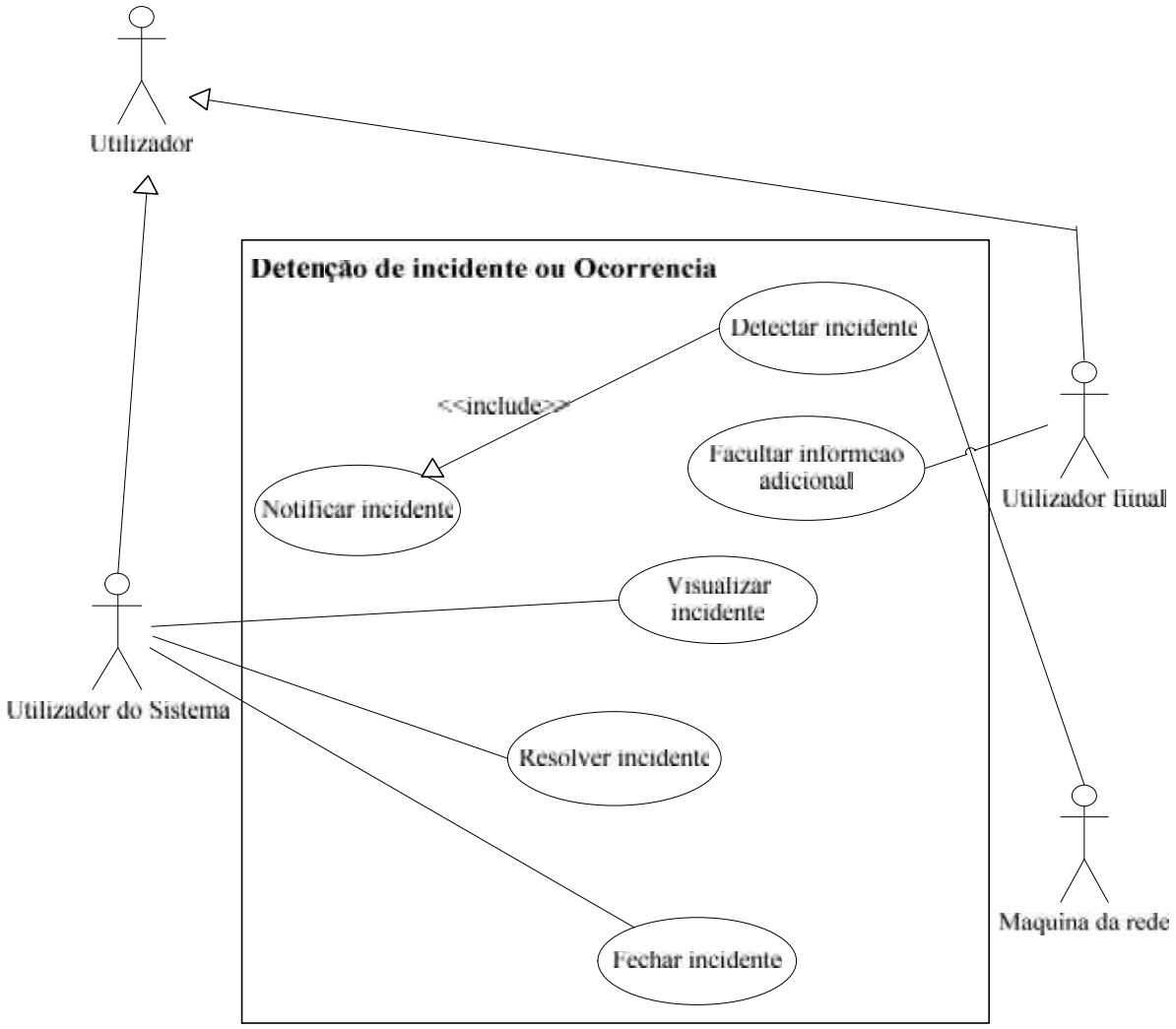

Figura 3. Caso de uso – detenção de Incidente ou Ocorrência

Identificação: UC005	
Nome: Detectar incidente ou ocorrência	
Actores: Máquina da rede	
Tipo: Primário	
Pré-condição: Dados da máquina fora do parâmetro estabelecido pelo sistema	
Pós-condição: Incidente ou ocorrência detectado e em estado aberto	
Fluxo principal	
Actor	**Sistema**
2. Envia os dados para o sistema	1. Solicita dados no sistema operativo da máquina
	3. Verifica conformidades dos dados recebidos
	4. Detecta incidente/ocorrência
Fluxo alternativo	
2. a) Erro de envio	
1. Não foi possível enviar os dados da máquina	

Identificação: UC006	
Nome: Notificar Incidente ou ocorrência	
Actores: Máquina da rede	
Tipo: Secundário	
Pré-condição: Incidente ou ocorrência detectado e em estado aberto	
Pós-condição: Notificação enviada	
Fluxo principal	
Actor	**Sistema**
2. Fica visível como máquina afectada	1. Envia a notificação para o programa servidor

Identificação: UC007	
Nome: Facultar informação adicional	
Actores: Utilizador final	
Tipo: Primário	
Pré-condição: incidente/ocorrência detectado	
Pós-condição: informação adicional enviada para o sistema	
Fluxo principal	
Actor	**Sistema**
3. Escreve no campo de texto da janela exibida	1. Exibe a janela de informação adicional ao utilizador final
4. Pressiona o batão "Enviar"	2. Solicita-o para descrever o impacto do registo
Fluxo alternativo	
4. a) Erro de envio	
1. Não foi possível enviar a descrição do incidente, verifique se a máquina está ligada a rede	

Identificação: UC008	
Nome: Visualizar incidente	
Actores: Utilizador do sistema	
Tipo: Primário	
Pré-condição: Notificação visível no sistema	
Pós-condição: Máquina afectada alcançada	
Fluxo principal	
Actor	**Sistema**
1. Visualiza a notificação	3. Monstra detalhes da máquina afectada
2. Localiza máquina afectada no sistema	

Identificação: UC009	
Nome: Resolver incidente	
Actores: Utilizador do sistema	
Tipo: Primário	
Pré-condição: incidente/ocorrência detectado	
Pós-condição: incidente/ocorrência resolvido	
Fluxo principal	
Actor	**Sistema**
1. Resolve o incidente/ocorrência no dispositivo afectado.	3. Altera o estado do incidente/ocorrência para resolvido
2. Edita o estado para resolvido	

Identificação: UC010	
Nome: Fechar incidente	
Actores: Utilizador do sistema	
Tipo: Primário	
Pré-condição: incidente/ocorrência resolvido	
Pós-condição: incidente/ocorrência fechado	
Fluxo principal	
Actor	**Sistema**
1. Pressiona o botão “fechar”	2. O sistema exibe uma janela para o utilizador escrever comentar
3. O utilizador digita comentário	
4. pressiona o botão concluir	

1.1.2. Diagrama de Classe

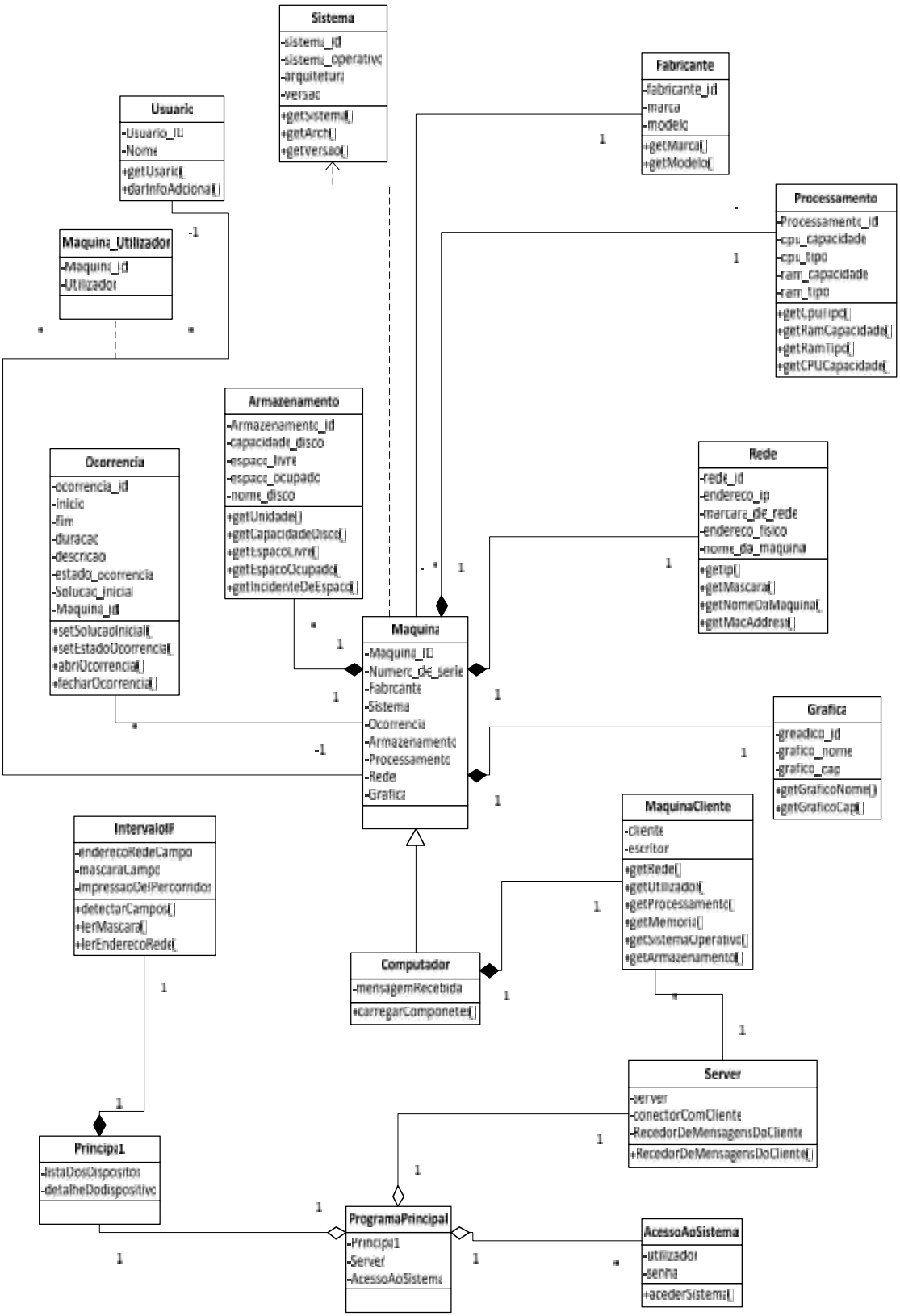

Figura 4. Diagrama de Classes

1.1.3. Diagramas de Sequência

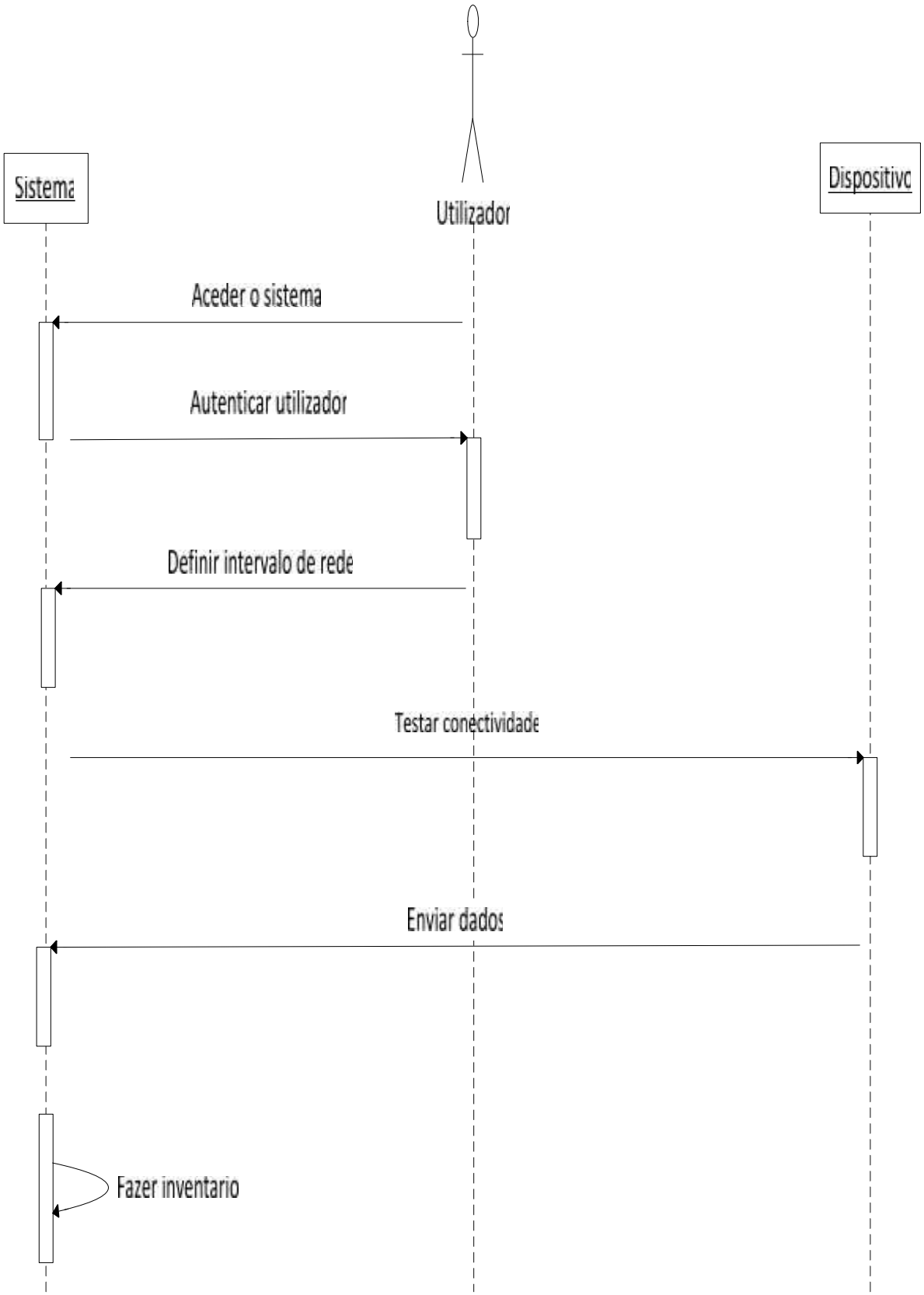

Figura 5. Diagrama de Sequência para inventário.

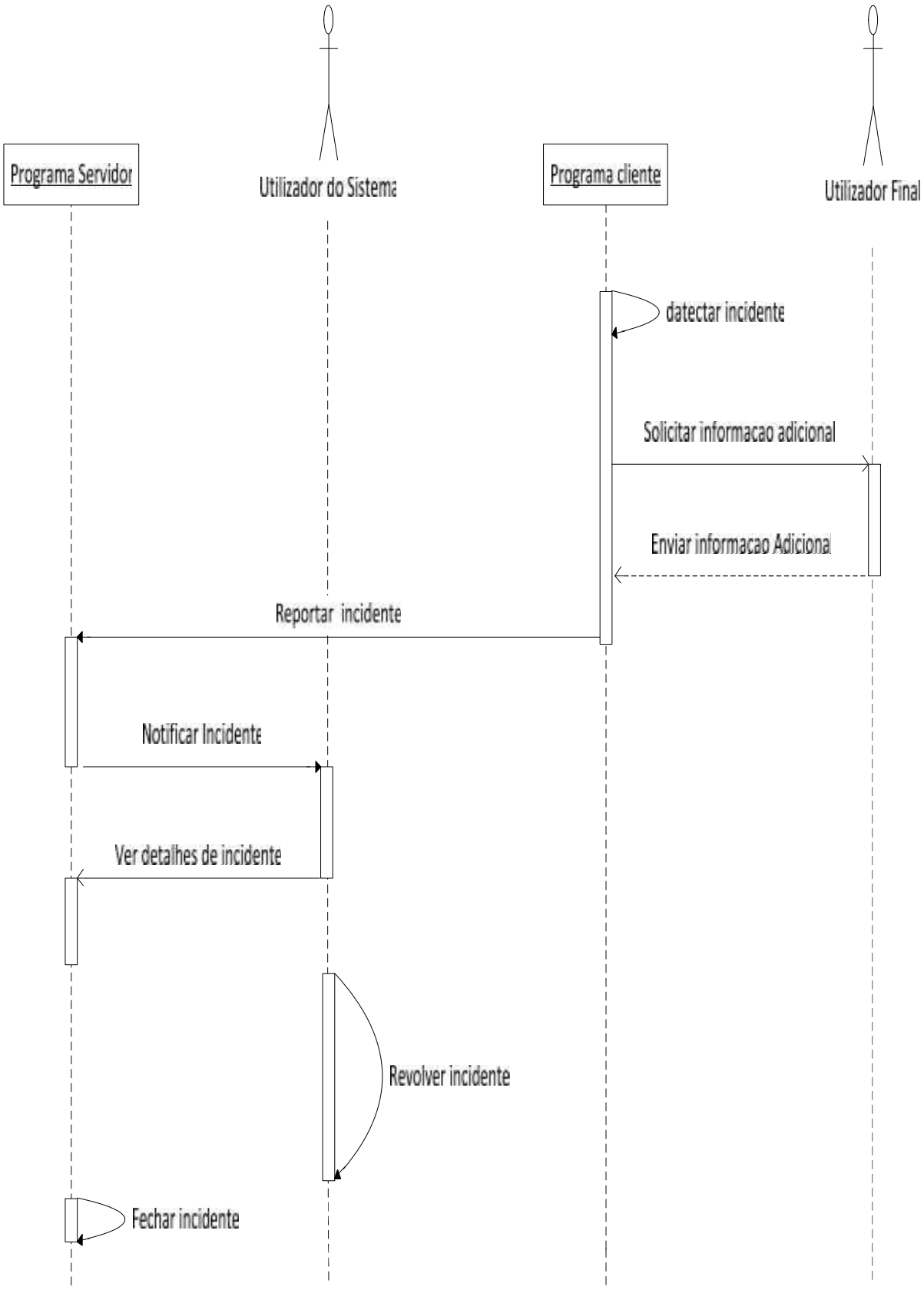

*Figura 6. Diagrama de Sequência para incidente***.**

1.1.4. Diagrama de Estado para incidente

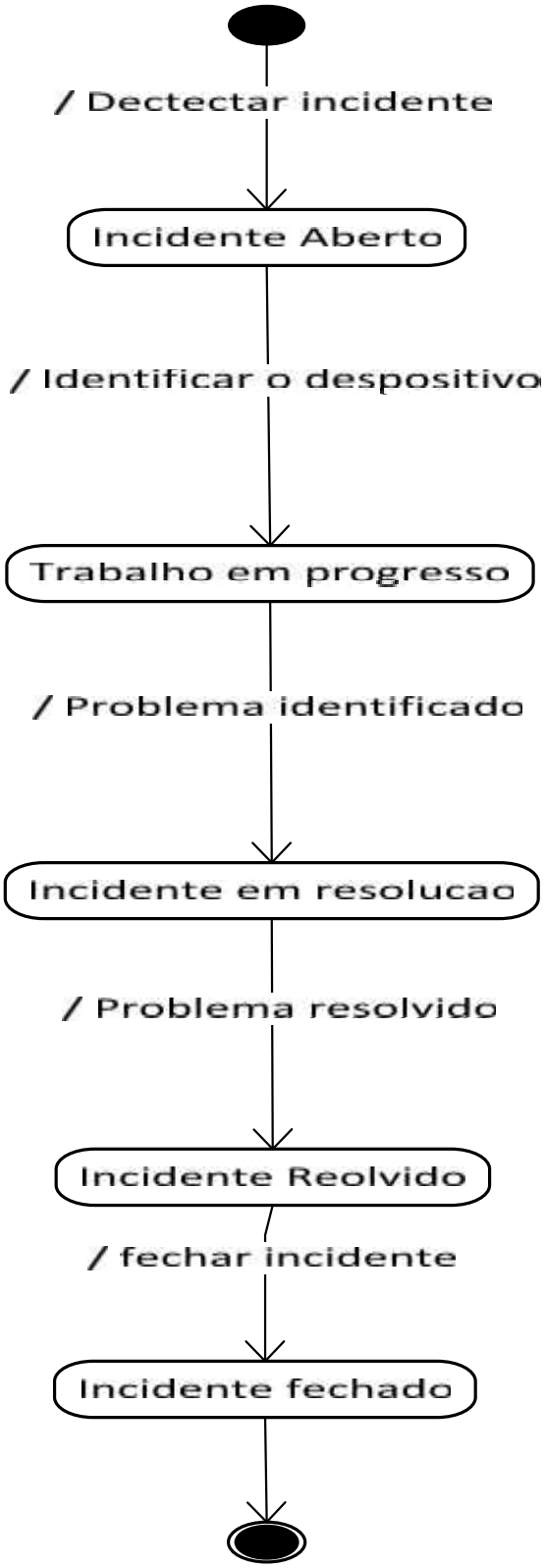

Figura 7. Diagrama de Estado do incidente ou ocorrência

1.2. Base de Dados do Sistema

A Base de Dados do aplicativo foi modelada em *Microsoft Visio 2010* em ambiente *windows 7 Professional 32 bits*, constituída por dez tabelas que representam a delimitação do fluxo de dados que decorre dentro do aplicativo.

O desenvolvimento da base de dados foi em *Oracle 10g express edition* como abaixo a figura 7 mostra, em ambiente *windows 7 Professional 32 bits*

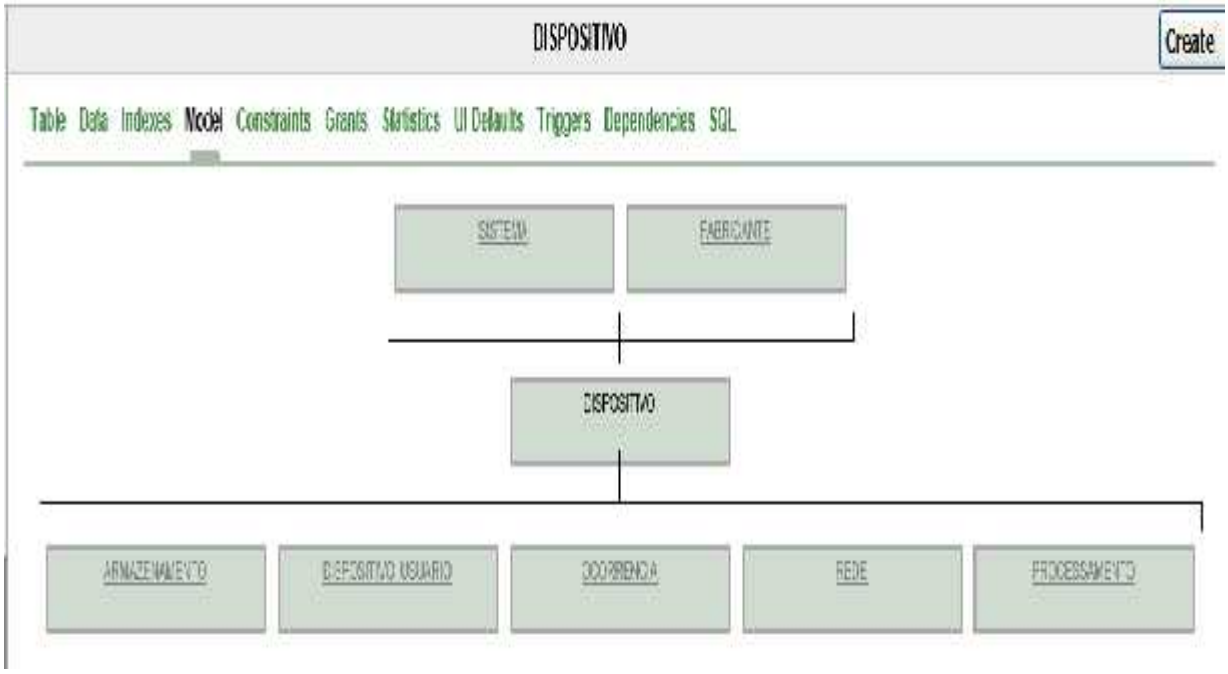

Figura 8. Esquema da base de dados em Oracle 10g express edition

1.2.1. Descrição de Esquema de Base de Dados

Após instalado o SGBD na máquina de teste, acedeu-se com utilizador SYSTEM[22] onde foi criado outro utilizador com o nome "SEBASTIAO", que contém esquema do sistema, ao qual atribuiu-se privilégios. A razão de criação de um novo utilizador visa garantir que não se gere confusão com as tabelas já existentes nos utilizadores SYS[23] e SYSTEM visto que estes já contêm tabelas criadas por padrão.

O foco deste trabalho é construir um aplicativo que inventaria e identifica alterações de activos de TI para máquinas dentro de uma rede. O esquema da base de dados foi,

[22] Utilizador criado por padrão com alto nível de gerenciamento a seguir do SY .
[23] Utilizador criado por padrão com máximo nível de privilégi

primeiramente, baseado na tabela *dispositivo* que, após o processo de normalização, gerou-se mais dez (9) tabelas, nomeadamente:

- **i)** **ARMAZENAMENTO** – tabela que contém toda a informação das unidades de armazenamento do dispositivo.
- **ii)** **DISPOSITIVO –** nesta tabela é guardada a informação do dispositivo (*hostname* e número de série).
- **iii)** **DISPOSITIVO_USUARIO** – esta é uma tabela associativa entre a tabela dispositivo e a tabela utilizador, visto tratar-se de uma relação de "muitos para muitos".
- **iv)** **FABRICANTE** – armazena informação do fabricante de dispositivo, modelo, série e o identificador do produto.
- **v)** **OCORRENCIA –** esta tabela regista toda a informação dos incidentes e ocorrências que afectam as máquinas na rede.
- **vi)** **PROCESSAMENTO –** contém informação das unidades de processamento e a frequência da memória *RAM* do dispositivo em questão.
- **vii)** **REDE –** tabela que contem dados de rede da máquina.
- **viii)** **SISTEMA –** esta tabela guarda os dados do sistema operativo corrente que corre na máquina.
- **ix)** **UTILIZADOR –** tabela que tem informação de utilizador que usa o dispositivo.

Essas tabelas foram codificadas pelos códigos SQL DDL como monstra o exemplo a baixo a criação da tabela Dispositivo:

```
CREATE TABLE DISPOSITIVO
  (         DISPOSITIVO_ID NUMBER (8,0),
            TIPO_DE_DISPOSITIVO VARCHAR2 (20),
            NUMERO_DE_SERIE VARCHAR2 (20),
            FABRICANTE_ID NUMBER (8,0),
            SISTEMA_ID NUMBER (8,0),
            NOME_DISPOSITIVO VARCHAR2 (50),
             CONSTRAINT PK_HOST PRIMARY KEY (DISPOSITIVO_ID),
             CONSTRAINT FK_MAKE FOREIGN KEY (FABRICANTE_ID)
              REFERENCES FABRICANTE (FABRICANTE_ID),
             CONSTRAINT FK_OS FOREIGN KEY (SISTEMA_ID)
              REFERENCES SISTEMA (SISTEMA_ID")
  );
```

Depois de criar as tabelas é possível, com o comando *select table_name from user_tables* linha de comando, listar as tabelas que foram criadas. Neste ambiente não é possível verificar a disposição das mesmas no esquema como na figura 7, por se tratar da linha de comando.
No ambiente gráfico, os objectos (tabelas) são exibidos em forma de esquema. Apenas os que estão directamente ligados ao objecto seleccionado aparecem visíveis, enquanto usando o comando *SQL* acima se tem a visão de todas as tabelas seleccionadas.

Para automatizar o incremento das chaves primárias e facilitar a gestão dos seus identificadores na base de dados, são criadas *Sequences*[24] e *Triggers,* isso para a versão do *Oracle 10g* porque não possui auto incremento. As *triggers* podem ser programadas para "disparar" a cada evento de inserção em linhas de uma tabela. Estas são códigos PL/SQL com blocos delimitados com *begin*[25] e *end*[26]. As *Sequences* são estruturas que contém um procedimento interno de auto incremento cada vez que ela é chamada ou utilizada, podendo começar de 1 até ao maior número inteiro suportado, incrementando de um em um, com ou sem ciclos. Para este caso as foram criadas 9 sequências para auto incremento das chaves primárias das 9 tabelas existentes na base dados como ilustra a figura.

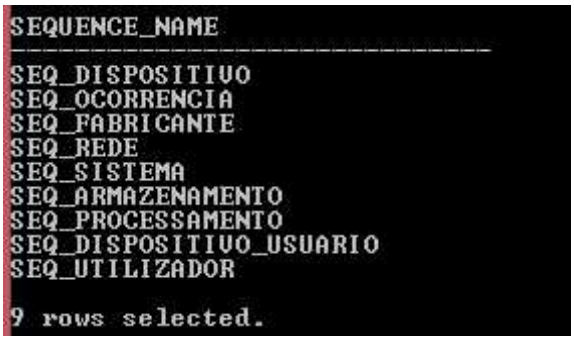

Figura 9. Ilustração de sequências na linha de comando

[24] Objecto sequencial de uma base de dados.
[25] Delimitador de um bloco de instruções (início).
[26] Delimitador de um bloco de instruções (fim).

1.2.2. Conexão com a Base de Dados Oracle

Como foi descrito no início desde capítulo as principais ferramentas para o desenvolvimento do aplicativo foram *JDK 1.7* e *Oracle 10g XE*, contudo, esta última é relativamente antiga em relação à versão de Java.

Os *drivers ojdbc14.jar* e *ojdbc14_g.jar*[27] padrão do *JDBC* do *Oracle 10g XE* armazenados no *path* na unidade do disco "*c*", na máquina de teste (*C:\oraclexe\app\oracle\product\10.2.0\server\jdbc\lib*), são incompatíveis para esta versão JDK 1.7, a versão seguinte '*Oracle 11g*' contém *drivers JDBC* compatíveis para esta versão java. Para corrigir, foi feito o *download* no site da *Oracle*. http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html. O *driver ojdbc5.jar,* foi o que tornou possível a conexão com o base dados. Os *drivers ojdbc14.jar* e *ojdbc14_g.jar* quando são usados para a conexão com a base de dados na IDE *Eclipse*, o compilador dá um erro como se os drivers não tivessem sido importados, isto é, ausência de *drivers*. Depois de fazer o *download*[28] do ficheiro do *Oracle* 11g foi possível estabelecer a conexão.

1.3. Desenvolvimento do Aplicativo

1.3.1. Descrição das classes

De acordo com a natureza do projecto, as classes em *Eclipse* foram construídas se baseando nas entidades que existem dentro dos processos no sistema. O pacote objectos contém a classe Computador.java que estende a classe Dispositivo.java, isto é, a classe Computador.java herda propriedades da classe Dispositivo.java que contêm classes internas como Rede, Fabricante, Processamento.java, Armazenamento.java e SistemaOperativo.java. Essas classes têm seus atributos internos que representam as características de um computador real conforme mostra a figura abaixo. A classe Server.java é responsável por escutar conexões dos programas clientes. A mesma implementa a *interface Runnable* e corre em paralelo com a classe Pricnipa1.java do pacote janelas que também implementam *Runnable.*

[27] Ficheiros utilizados par conectar base de dados às aplicações.
[28] Transferência de ficheiros de uma rede remota para o computador local.

Cada mensagem enviada de uma máquina cliente para o programa servidor é instanciada um objecto Computador que instancia, também, todas as classes internas dentro do mesmo, carregando todas as propriedades recebidas pela máquina cliente.

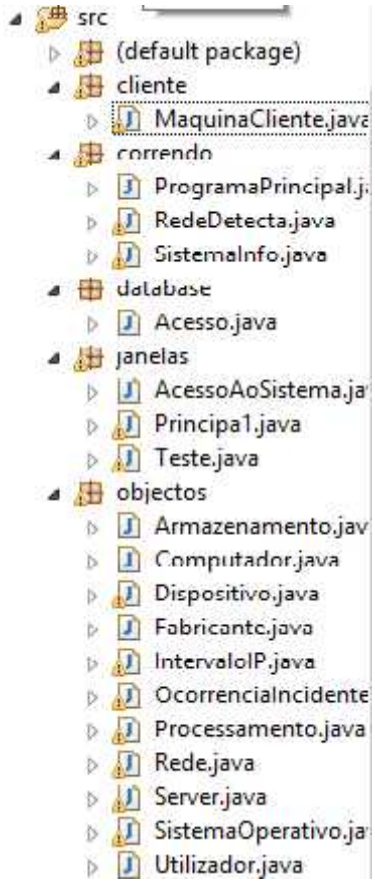

Figura 10. Ilustração das classes no Eclipse

1.3.2. Aceder ao sistema – classe: AcessoAoSistema.java

Para garantir que haja segurança no uso do sistema e como é habitual na maior parte dos *softwares* corporativos. A primeira janela que aparece no programa servidor quando este é chamado, é uma janela de *Login* que autentica o utilizador no sistema. O que esta classe faz é verificar se os dados introduzidos pelo utilizador são correspondentes aos dados que existem dentro da tabela utilizador na base de dados. Se essa condição for verdadeira o programa principal abre, caso contrário esta classe de *login* informa ao utilizador que as credenciais estão incorrectas e solicita-o para introduzir novamente.

Figura 11. Formulário de acesso ao sistema.

1.3.3. Descrição da comunicação entre as máquinas – classe: Server.java

Para garantir que haja comunicação entre o sistema e os dispositivos na rede, a classe *Socket* e *ServerSocket* do pacote *java.net* foram utilizadas neste protótipo. Desta forma, em cada máquina pertencente à rede, existe um programa Java que roda e comunica-se com o programa servidor, enviando descrições suas ao programa servidor.

Esse programa servidor tem de criar condições de forma que cada máquina possa se conectar a ele, usando um leitor *Scanner* (classe do pacote *Java.util* responsável pela leitura de dados) exclusivo para cada conexão. Assim múltiplas conexões são disponibilizadas por *Threads* java.

A ligação entre o programa servidor e o programa cliente é garantido através de *sockets* Java. As *threads* possibilitam que um cliente se conecta ao seu servidor e, ao mesmo tempo, o servidor pode escutar outras conexões de outros clientes e conectar-se a eles.

Do lado da máquina cliente temos um objecto *socket* que é instanciado com o endereço *IP* do servidor e o número de porta *TCP* no qual o programa corre. É também instanciado um objecto da classe *PrintWriter* (classe Java responsável pela escrita de dados) que captura, com o método *getoutputStream(),* a saída de dados que é enviado ao aplicativo. Esse objecto escreve com o método *println()* toda informação da máquina capturada pelo programa cliente e em seguida é encaminhada para o programa servidor.

1.3.4. Definição da dimensão da rede no aplicativo – classe IntervaloIP.java

Esta classe é chamada por um botão com o título de "*Definições de Rede*" dentro da classe *Principa1.java*.

O *Jframe*[29] abaixo é importante e fundamental para o utilizador do sistema e para o próprio sistema em si, porque dá dimensão de rede e dos dispositivos que ele mesmo deve percorrer e inventariar. Foi criada uma classe que "estende" a classe *JFrame* com o título "definições de rede" como mostra a figura a baixo.

Esta classe tem dois campos que recebem textos introduzidos pelo utilizador do sistema, cujo primeiro é para endereço de rede o e segundo para a máscara de rede. Ambos devem ser fornecidos ao sistema em formato decimal nos campos de texto.
A classe criada tem dois métodos, um que trata a máscara de rede para número de *IP's* por percorrer a partir do IP de rede fornecido. Este tratamento é feito por uma estrutura *switch*[30] que verifica as 24 possibilidades de máscara de rede numa estrutura de IP versão 4. O outro método divide a *string* do endereço de rede fornecido pelo utilizador com o método *split()* da classe *String* em quatro pedaços. Em seguida, transforma em inteiro cada um destes pedaços (octeto que está em formato decimal), depois o último pedaço (quarto pedaço) é percorrido até chegar ao número 255. Assim, ao terceiro pedaço é incrementado uma unidade e o quarto pedaço volta para zero (0). O mesmo acontece com o segundo pedaço, que é incrementado uma unidade quando o terceiro pedaço chega ao número 255. Deste modo é possível percorrer redes de máscara /8 (255.0.0.0) até /30 (255.255.255.252).

O Botão "Mostrar resultado" da figura, ilustra na área de texto a lista de *IP's* percorridos que são armazenados num *ArrayList*[31] de *String* para que futuramente possam ser usados para fazer inventário no programa principal.

[29] Classe Java responsável pela interface gráfica com o utilizad
[30] Classe que recebe conjunto de caracteres.
[31] Classe Java pertencente as colecções.

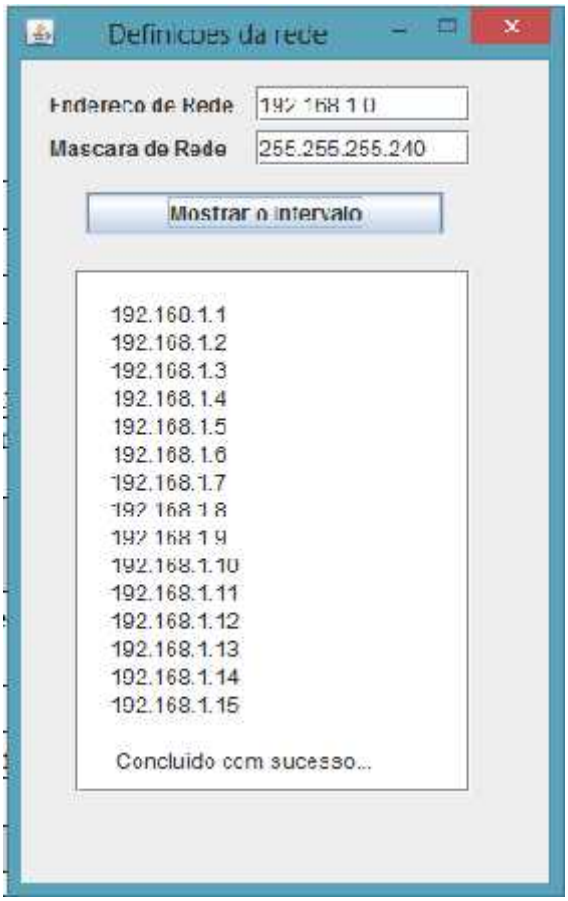

Figura 12. Janela de definições do programa.

1.3.5. Inventariando unidades de armazenamento das máquinas – classe: Armazenamento.java

Depois da conexão bem-sucedida com o dispositivo, é possível levantar relação das unidades de armazenamento existentes em cada máquina com uso da biblioteca *IO – Input output* da linguagem de programação Java.

Esta biblioteca tem *classes* e métodos que nos permitem manipular arquivos e directórios a partir do conceito de "entrada e saída" ou "leitura e escrita". A classe tem métodos que retornam objectos que é possível percorrer, usando estrutura cíclica *foreach* (ciclo *for*[32] aprimorado), cada unidade de armazenamento e, ao mesmo tempo, capturar informações de cada unidade de armazenamento.

Uma variável que percorre os discos da máquina passa a ser a unidade de armazenamento naquele índice do ciclo. Visto que a mesma tem a função de recuperar, fica muito mais fácil

[32] Estrutura de controlo usado por varias linguagens de programação.

adquirir informações como: nome da unidade, capacidade do disco, espaço livre e espaço ocupado.

1.3.6. Tipo de incidente e ocorrência nas máquinas

A captura de ocorrências é feita a partir 3 métodos que verificam conformidades da máquina que está ser inventariada.

Cada um destes métodos tem a sua funcionalidade:

* O primeiro verifica se as unidades de armazenamento da máquina que está ser inventariada tem espaço suficiente para armazenar uma quantidade de ficheiros, isto é, se o espaço ocupado for superior a 85% do espaço total, o sistema notifica uma *ocorrência*[33].

* O segundo verifica se na presente máquina existem ficheiros que possam comprometer a sua segurança (ficheiros com extensão: **.com, *.exe, *.bat, *.scr* e **.pif*), se sim, o sistema notifica um *incidente*[34]. Este método faz verificações nos directórios do disco onde está instalado o sistema operativo.

* E o último verifica se existem tarefas computacionais que estejam "pendentes" e que podem de alguma forma impossibilitar o utilizador de trabalhar na máquina. Assim o sistema notifica uma ocorrência.

Se uma destas condições ou todas elas forem verdadeiras, o programa exibe uma janela para o utilizador final colocar o impacto da ocorrência/incidente, e o solicita para enviar para o sistema. De seguida o comentário é enviado juntamente com os dados do incidente e os dados da máquina para o programa servidor em forma de notificação para o utilizador do sistema.

[33] Acontecimento, facto sucedido, eventualidade, circunstância, coincidência,

[34] Episódio repentino que reduz, significativamente, as margens segurança.

CAPÍTULO V - Colecta de Dados e Informação, Análise e Validação de Dados

1.1. Colecta de Dados e Informação

Tratando-se de uma pesquisa quantitativa, os instrumentos usados para a este tipo devem ser adequados.

Este tipo de pesquisa usa instrumentos a partir dos quais se obtém valores em números sobre características específicas de um grupo de pessoas ou objectos que se está a medir. Desta forma, é possível criar o relatório que dá a visão conclusiva num determinando aspecto em estudo.

1.1.1. Técnica de colecta escolhida

A técnica de colecta de dados escolhida foi o Questionário, pois garante o anonimato para quem está ser pesquisado e permite elaborar questões mais objectivas que estão directamente ligadas ao tema. Dá ainda espaço de tempo para quem está a responder o questionário reflectir sobre a sua resposta e seu custo é razoável comparativamente aos outros métodos, como é o caso da Entrevista.

1.1.2. Recolha de dados

Para avançar com esta pesquisa e validar as hipóteses, foi necessário saber primeiro dos utilizadores finais de informática a nível desta empresa qual é o seu parecer em relação ao serviço de assistência em *helpdesk.* Todos por que foram abordados não mostraram nenhuma dificuldade em contribuir, pelo contrário estavam muito satisfeitos em dar o seu contributo.

O questionário usado apurou dentro de uma amostra de quinze (15) colaboradores da PETROMOC qual é o seu nível de satisfação. Em primeiro lugar, procurou-se saber se existe ou não satisfação na empresa em relação a este serviço. Se existe as hipóteses H0 e H1 servem de motivo para o surgimento dessa insatisfação.

1.2. Análise e Validação de Dados

Depois de colher os dados da amostra de colaboradores, os resultados não estavam longe daquilo que era suspeito antes de se fazer a pesquisa.

Num grupo de quinze (15) colaboradores apenas dois (2) correspondentes a 13% do mesmo departamento mostraram-se satisfeitos em relação a desempenho dos *helpdesk*, afirmando que estes têm agido de forma activa e empática as solicitações dos seus serviços dentro da empresa.

Os restantes treze (13) que correspondem a 87% mostraram-se insatisfeitos, alegando que há centralização do pessoal na província de Maputo concretamente na cidade de Maputo (Sede da PETROMOC na Baixa da Cidade de Maputo), onde existem técnicos da empresa *E-business Systems* (EBS) que prestam *outsourcing* em *helpdesk* e IT. Desta forma, estão na sede do edifício os quadros IT da Petromoc, que em algum momento também prestam serviços em *helpdesk* nesta empresa. Então, as filiais de Matola e as restantes províncias do país não têm pessoal para dar assistência.

Esta situação faz com que o tempo de resposta torne-se muito lento, mesmo com existência do aplicativo *web* que regista *chamados/incidentes* manualmente. O que foi sublinhado pelos colaboradores desta empresa é que os incidentes registados no aplicativo *web* tem prioridade de serem resolvidos devido a sua visibilidade, isto é, por causa da sua existência formal, enquanto os que não existem formalmente (não existem dentro do aplicativo mas existentes fisicamente) tendem a ser deixados para depois ou nunca são resolvidos.

Contudo a hipótese H0 é válida, os *helpdesk* priorizam os incidentes visíveis no aplicativo actual, porque este gera relatórios e nele é possível ver:

- Quem resolveu?
- Quando resolveu?
- Quais os chamados pendentes que estão atribuídos ao técnico x,
- Quais os chamados que o técnico *x* resolveu em tempo *y*.

Mas a sua abertura, pré-solução e solução é ainda manual.

CAPÍTULO VI - Conclusões e Recomendações

6.1. Conclusão

Com término deste projecto é possível verificar que o protótipo é capaz inventariar computadores ligados a uma rede informática que correm sistemas operativos *Windows*.

Este protótipo por sua vez possibilita detectar ocorrências e incidentes de cada máquina ligada a rede, registando numa base de dados por forma a extrair relatórios para a aplicação.

Portanto, pode-se concluir que este sistema aproxima o profissional de *IT* aos seus colegas, proporcionando uma relação saudável, visto que em cada incidente, existe uma possibilidade de o utilizador afectado interagir com o profissional de *IT*, colocando o seu comentário. Garante visão ampla da infraestrutura informática organizacional, permite visibilidade de histórico de ocorrências, reduz o tempo de resolução de problemas (visto que o sistema lança pré-soluções ao detectar incidente) e melhora o desempenho do Gerenciamento de Serviços de TI.

Contudo, o motivo que provoca a insatisfação dos utilizadores finais de informática em relação à assistência *helpdesk* na Petromoc, é a priporidade colocada aos incidentes que se encontram registados no sistema.

6.2. Recomendações

Muitas vezes os utilizadores de informática saturam os *helpdesk* com questões “banais” e a tendência das empresas grandes é ter um número muito reduzido colaboradores que dão suporte em TI para um grande número de utilizadores finais.

Se todos os colaboradores nas empresas que usam a informática como ferramenta de trabalho tivessem uma boa prática de uso e este protótipo fosse implementado, a assistência *helpdesk* seria aprimorada.

Para esta situação, recomenda-se que haja treinamento contínuo para os utilizadores finais de informática, visto que as tecnologias de informação tendem a evoluir dia-após-dia. Estes treinos poderiam ajudar, fazendo com que os utilizadores se familiarizassem ainda mais com a informática, sendo esta a sua ferramenta de trabalho.

Existem algumas questões, das inúmeras levantadas como:

- Cabo de rede solto. Ex.: pessoal da limpeza desconectou sem intenção;
- Botão de *Power* de monitor desligado, ou botão *Power* da UPS desligado;
- Impressora sem *toner*;
- Impressora sem papel;
- Etc.;

Estas questões são abordadas para o *IT* como um problema e, por vezes, grave pelo *end user*. Em várias situações o *IT* é obrigado a se deslocar do seu local, parar com o seu trabalho, para atender a uma questão muito simples que mesmo o próprio utilizador poderia resolver.

De acordo com a natureza de negócio de cada empresa, há uma necessidade de capacitar os seus colaboradores para que sejam capazes de responder aos pequenos constrangimentos que encontram no desempenho do seu trabalho.

Da mesma maneira que existem campanhas ou lembretes de boas práticas de consumo de água e energia para redução do custo, é necessário que a informação sobre a utilização de um computador ou impressora e a base do seu funcionamento esteja sempre visível e explícita, usando os meios de comunicação que a empresa dispõe, como formações, induções, cartazes de parede, panfletos, *emails*, etc.

Bibliografia

1. AMANAJÁS, J. et al. (2013) *Definição de Critérios de Validação de Dados Anemométricos a partir de Padrões Espaço-Temporais de Vento em Superfície*. s.l.: s.e.., disponível em http://cascavel.ufsm.br/revistas/ojs-2.2.2/index.php/cienciaenatura/article/view/11624; acedido a 13-08-2014

2. BARBOSA FILHO, Manuel. (1980) *Introdução à Pesquisa: Métodos, técnicas e instrumentos.* Rio de Janeiro: Livros Técnicos e Científicos, 2.ed.

3. BOAVENTURA, Jorge. (1979) *O acidente traído*. São Paulo: Impres/Lithographica Ypiranga.

4. BRITO, N., et al (2003) *Validação de métodos analíticos: estratégia e discussão* Disponível em: http://www.google.co.mz/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=3&ved=0CDYQFjAC&url=http%3A%2F%2Fojs.c3sl.ufpr.br%2Fojs%2Findex.php%2Fpesticidas%2Farticle%2FviewFile%2F3173%2F2546&ei=_R5dVLPxIc2v7Abdl4HICg&usg=AFQjCNH8EKp57knF5960OczuufCMjEeV8Q, acedido a 07-11-2014

5. CARVALHO, P.F. (2011) *Resumo sobre ITIL V3 – Governancia de TI*. s.l.: s.e.. disponível em http://www.pedrofcarvalho.com.br/PDF/ITIL_OPERACAO_SERVICOS.pdf, acedido a 02-10-2014.

6. CERVO, Armando Luiz, BERVIAN, Pedro Alcino. (1978) *Metodologia científica: Para uso dos universitários.* São Paulo: McGraw-Hill. 2.ed.

7. COSTA, J. (2010) *Apostila de Redes de Computadores*. s.l.: s.e.. disponível em http://www.jeffersoncosta.com.br/Redes.pdf; acedido a 13/08/2014.

8. CRUZ, Carla, RIBEIRO, Uirá. 2004. *Metodologia científica: Teoria e prática.* Rio de Janeiro: Axcel Books. 2.ed.

9. GREENBERG, N. (2004) *Banco de Dados Oracle 10g: Fundamentos de SQL1*. s.l.: s.e.., disponível em http://tororodeideias.files.wordpress.com/2011/02/apostila1.pdf; acedido a 10/01/2014

10. GUEDES, G.T.A. (2010) *UML2 Guia Prático.* s.l.: s.e.. disponível em https://www.novatec.com.br/livros/uml2/capitulo9788575221457.pdf; acedido a 08/04/2014.

11. JAQUES, P. A. 2007. *Programação Avançada em Java.* s.l.: s.e.. disponível em http://professor.unisinos.br/pjaques/index.php?option=com_content&view=article&id=21&Itemid=20&lang=en; acedido a 15/08/2014

12. JUNIOR, E. A. (2006) Oracle *PLSQL*. s.l.: s.e.., disponível em http://www.jeffersoncosta.com.br/redes.pdf; acedido a 10/01/2014.

13. LORENÇO, L. (2010) *Riscos Naturais e Protecção do Ambiente*. s.l.: s.e.. disponível em http://www.uc.pt/fluc/nicif/Publicacoes/Colectaneas_Cindinicas/Download/Colecao_I/Artigo_I.pdf; acedido 30/09/2014.

14. MARION, José Carlos, DIAS, Reinaldo, TRALDI, Maria Cristina. (2002) *Monografia para os cursos de Administração, Contabilidade e Economia.* São Paulo: Atlas.

15. MOTTA-ROTH, Désirée (org.). 2001. *Redação acadêmica: Princípios básicos.* Santa Maria: Universidade Federal de Santa Maria, Imprensa Universitária.

16. LIBARDI et al, (2010) *Métodos Ágeis* http://www.ft.unicamp.br/liag/Gerenciamento/monografias/monogafia_metodos_ageis.pdf ; acedido a 19/10/2014.

17. SINCIC, M. (2012) *Apostila de SQL Básico.* s.l.: s.e.. disponível em http://blog.segr.com.br/wpcontent/uploads/2013/09/Apostila-de-SQL-B%C3%A1sico1.pdf; acedido a 06/05/2014

18. SOARES, M.S. (s/d) Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software http://www.dcc.ufla.br/infocomp/artigos/v3.2/art02.pdf Data de Acesso: 19/10/2014.

19. SOUZA, V. E. S. (2013) *Modelagem OO com UML*. s.l.:s.e.. disponível em http://www.inf.ufes.br/~vitorsouza; acedido a 23/07/2014.

Links:

20. http://187.7.106.14/tcc2012_2/lib/exe/fetch.php?media=projeto10:paulotcc_final.pdf; acedido a 24/05/2014 as 01:34

21. http://www.bibliotecavirtual.celepar.pr.gov.br/arquivos/File/MonografiaseArtigos/Mono_Dario_Final.pdf acedido a 16/03/2014 as 14:23

22. http://fatecsjc.edu.br/trabalhos-de-graduacao/wp-content/uploads/2012/03/BDR1_alessandra_douglas2009.pdf; acedido a 16/03/2014 às 14:23

23. http://www.oraclehome.com.br/2012/02/13/como-ler-e-gravar-arquivos-textos-pelo-oracle-10g-ou-superior/; acedido a 10/06/2014 às 20:14

24. http://eduardolegatti.blogspot.com/2008/06/um-pouco-do-oracle-text.html; acedido a 18/06/2014 às 18:08

25. http://alexandresl.blogspot.com/2013/08/banco-de-dados-oracle-tipos-de-dados.html#!/2013/08/banco-de-dados-oracle-tipos-de-dados.html; acedido a 18/06/2014 às 19:56

26. http://minhateca.com.br/henriqueflusao/Aulas/Universidade+XTI+-+Curso+de+JAVA+7+(116+aulas),4; acedido a 02/07/2014 às 20:48

27. http://www.dmo.fee.unicamp.br/~henrique/cursoc++/diagrama.pdf; acedido a 04/08/2014 às 11:26

28. http://pt.wikipedia.org/wiki/Information_Technology_Infrastructure_Library; acedido a 22/10/2014 às 20:04

29. http://www.colorconsole.de/console/br/index.htm acedido a 20/10/2014 às 10:16

30. http://technet.microsoft.com/en-us/library/bb491010.aspx acedido a 20/10/2014 às 0:46
31. http://pt.wikihow.com/Criar-um-V%C3%ADrus-Falso-e-Inofensivo acedido a 20/10/2014 às 16:18

32. http://dicasdalenalopez.blogspot.com/2011/07/arquivos-e-extensoes-perigosos-para-o.html acedido a 21/10/2014 às 10:16

33. http://windows.microsoft.com/pt-pt/windows-vista/recognizing-dangerous-file-types acedido a 25/11/2014 às 17:15

34. http://pt.wikipedia.org/wiki/Deadlock acedido a 10/11/2014 às 13:28

35. http://dti.unilab.edu.br/detectar-e-remover-possiveis-ameacas/ acedido a 23/10/2014 às 8:19

36. https://www.google.co.mz/?gws_rd=cr&ei=hC1vVJLLIoPksASCzYH4Dg#q=detec%C3%A7%C3%A3o+de+incidentes+no+computador&spell=1 acedido a 20/11/2014 às 19:07
37. http://forum.techtudo.com.br/perguntas/75408/qual-comando-no-windows-uso-para-mostrar-todos-os-processos-do-user acedido a 20/11/2014 às 15:18

38. http://pt.wikipedia.org/wiki/Byte acedido a 20/11/2014 às 10:16

Anexo

Universidade Politécnica

Questionário

1. Existe uma insatisfação no diz respeito a assistência de *helpdesk* na sua empresa?

☐ Sim ☐ Não

Se ***Sim*** seleccione um dos motivos abaixo com ***X*** ou descreva em número **2** o outro motivo.

___ Os *helpdesk* preocupam se em seguir os procedimentos dos aplicativos de TI e o suporte ao utilizador fica em segundo plano.

___ Os *helpdesk* não preocupam se em seguir os procedimentos dos aplicativos de TI e suporte ao utilizador não fica em segundo plano.

2. Quais os outros motivos que provocam a sua insatisfação?