Mestrado em

Engenharia Eletrotécnica

Implementação de Sistema de Gestão de Manutenção numa Indústria Metalomecânica

Relatório de Projeto apresentado para a obtenção do grau de Mestre em Engenharia Eletrotécnica – Área de Especialização em Automação e Comunicações em Sistemas Industriais

Autor

José Manuel Rodrigues Seabra

Orientador

Doutor Inácio Sousa Adelino Fonseca

Professor do Departamento de Engenharia Eletrotécnica

Instituição

Instituto Superior de Engenharia de Coimbra

Coimbra, maio, 2017

Agradecimentos

Quero deixar o meu apresso e meu reconhecimento:

- À minha família pelo apoio e paciência demonstrada ao longo de todo o Mestrado, tendo a família aumentado para o dobro nesse período;
- Ao meu orientador de projeto, Professor Doutor Inácio Fonseca pela sua disponibilidade, sabedoria e proporcionando-me uma oportunidade de alargar o meu campo de visão sobre os assuntos abordados nas reuniões de orientação;
- À empresa Miralago S.A., especialmente à sua Administração pela abertura ao desenvolvimento do projeto;
- Aos meus colegas de trabalho na área de manutenção e, em especial ao Sr. Júlio Chio pelas suas ideias e apoio na fase de testes da aplicação.

Resumo

Nenhuma empresa moderna consegue sobreviver no mercado atual sem pensar na eficiência dos seus recursos. O Departamento de Manutenção ainda é visto, em muitas empresas, como um quartel de bombeiros, a quem se recorre quando há um problema. Esta postura não auxilia a eficiência de recursos e dificulta o trabalho dentro do Departamento. Os sistemas informáticos de gestão de recursos e planeamento, são muito rígidos e o custo de implementação de soluções por vezes é considerado demasiado para as empresas, fundamentalmente quando essas soluções servem a Manutenção.

Neste trabalho procurou-se criar uma solução informática simples e barata, para a organização do trabalho no Departamento de Manutenção. Foi desenvolvida uma aplicação em Microsoft Access® cujos formulários permitem a gestão dos pedidos de intervenção nos equipamentos ou instalações. São geradas Ordens de Trabalho (OT) que, por sua vez, são utilizadas para criar os indicadores gerais de manutenção como os tempos de reparação e tempos entre falhas. Nestas OTs são registados os custos de mão-de-obra individuais, permitindo também criar um indicador de ocupação do recurso humano. A lista de artigos utilizados nas OTs é também atualizada na base de dados com a introdução do seu preço, imagem e fornecedor.

No caminho da Indústria 4.0, é feita a recolha de dados dum Serrote Automático, por forma a acompanhar o trabalho desenvolvido e visualizar a estatística da sua utilização. Fazendo-se o acompanhamento horário da máquina é possível estabelecer planos de manutenção focados na utilização de que é alvo e não no tempo decorrido deste a última intervenção ou nas horas de trabalho.

A aplicação, em Access®, que foi criada neste trabalho, permite gerir Manutenção Correctiva, gerar documentos de Manutenção Preventiva, inventariar artigos nos armazéns de manutenção, gerir a lista de máquinas, recursos humanos e fornecedores. Os documentos utilizados na manutenção, com a inclusão dos artigos e seus fornecedores e o tempo utilizado pelos Recursos Humanos na execução das tarefas, permitem a obtenção de indicadores gerais do funcionamento do departamento. O formulário da máquina Serrote Automático apresenta as estatísticas horárias do seu funcionamento para facilitar a tarefa de ajuste do plano de manutenção à realidade de funcionamento do equipamento.

Palavras-chave: Eficiência; Manutenção; Base de Dados; Indústria 4.0, Microsoft Access.

Abstract

No modern company can survive in today's market without thinking about the efficiency of its resources. The Maintenance Department is still seen, in many companies as a fire station, which is used when there is a problem. This posture does not help the efficiency of resources and hinders the work within the Department. The Information Systems (IT) for resource management and planning, are very strict and the cost of implementing solutions is sometimes considered too much for companies, fundamentally when these solutions serve Maintenance.

This project created a simple and cheap computer solution for the organization and work in the Maintenance Department. An application was developed in Microsoft Access® whose forms allow the management of requests for intervention in equipment or installations. Work Orders (WO) are generated which, in turn, are used to create general maintenance indicators, such as repair times and times between failures. These WOs record the individual labor costs, allowing the creation of indicators such as the occupation of the human resource or the overall cost in a given intervention. The list of items used in WOs is also updated in the database with the introduction of its price, image and supplier.

In the way of the Industry 4.0, the data collection of an Automatic Saw, allow the visualization of the work done and collect some statistics of it. By following the machine's schedule, it is possible to establish maintenance plans focused on the use of it and not in the time elapsed from the last intervention or the working hours.

The Microsoft Access® application, that was created in this work, manages Corrective Maintenance, generates Preventive Maintenance documents, inventory items in Maintenance warehouses, manages machine list, human resources, and suppliers. The documents used in the maintenance, including the items and their suppliers and the time used by the Human Resources in the execution of the tasks, enable general indicators of the department functioning. The Automatic Saw machine form, shows the hourly statistics of its operation to facilitate the task of adjusting the maintenance plan to the reality of operation.

Key-words: efficiency; Maintenance; database; Industry 4.0, Microsoft Access

Índice

Agradecimentos iii
Resumo v
Abstract vii
Índice ix
Índice de Figuras xi
Índice de Tabelas xiii
Índice de Códigos xv
Simbologia e Abreviaturas xix
1 Introdução 1
1.1 Motivação e contexto 1
1.2 Objetivos 4
1.3 Organização do documento 4
2 Estado da Arte 7
3 Arquitetura do sistema 9
3.1 Enquadramento 9
3.2 Arquitetura do sistema 9
3.2.1 Elementos do sistema de Gestão de Manutenção 10
3.3 Sumário 12
4 Ferramentas e Equipamentos Utilizados 15
4.1 Raspberry Pi ® 3 Model B+ 15
4.2 Autómato Omron® CP1L-M40DR-A 15
4.2.1 Software programação CX-Programmer 16
4.3 Consola Omron® NB5Q-TW01B 17
4.3.1 Software NB-Designer 18
4.4 MODBUS TCP 18
4.5 Computador com Sistema Operativo Windows® 7 19
4.5.1 Microsoft Access® 2016 19
4.6 Sumário 20
5 Desenvolvimento de software 21
5.1 Arranque do servidor MODBUS TCP (Raspberry Pi®) 21
5.2 Servidor MODBUS TCP (Slave) 23
5.3 Gestão da Base de Dados Microsoft Access® 27
5.3.1 Organização dos Formulários 31
5.4 Sumário 67
6 Conclusões 69
6.1 Conclusões 69
6.2 Melhorias no Hardware 69

6.3 Melhorias no Software 70
6.4 Trabalhos Futuros 70
Bibliografia 71
Anexos 77
Anexo I – Código para o MODBUS Server (modbusserver.sh) 77
Anexo II – Servidor ModBus em Python 78
Anexo III – Descrição do MBAP Header 81
Anexo IV – Formulários da Aplicação 82
Anexo V – Impresso de Registo das Intervenções da Manutenção 83
Anexo VI – VBA de verificação de número de intervenção 84
Anexo VII – Adição de Novas Ordens de Trabalho 85
Anexo VIII – Verificação da existência de registo de OT 86
Anexo IX – Escrever LogFile em formato .txt 87
Anexo X – Procura de Ordens de Trabalho 88
Anexo XI – Gerar gráfico em Excel com os dados do tipo de manutenção 89
Anexo XII – Gerar gráfico com mais do que uma série de dados em VBA 91
Anexo XIII – Procurar ID de material com base na referência interna da empresa usando a classe Recordset 94
Anexo XIV – Evento de abertura do formulário frmMaterial 95
Anexo XV – Processo de Fecho do Formulário frmMaterial 96
Anexo XVI – Comando para pesquisar referências de artigos 97
Anexo XVII – Determinar o número de dias úteis 98
Anexo XVIII – Encontrar imagens já carregadas na base de dados 99
Anexo XIX – Associar Imagem a Material 100
Anexo XX – Envio de e-mails para Logística 101
Anexo XXI – Indicador Tempo Médio Entre Intervenções 104

Índice de Figuras

Figura 1-1 - Impresso orientador para a realização da Manutenção Preventiva de 1º Nível 2
Figura 1-2 - Impresso para registo de intervenções de Manutenção Preventiva de 1º Nível..... 2
Figura 1-3 – Impresso Orientador para Manutenção Preventiva a executar pela Manutenção 3
Figura 1-4 – Impresso para registo da Manutenção Preventiva de maior periodicidade 4
Figura 3-1 – Diagrama de Blocos geral do fluxo de informação 10
Figura 3-2 - Diagrama de blocos do sistema de informação. Serrote Automático FAT (Máquina 23117) 11
Figura 3-3 – Representação das portas de comunicação na consola NB5 do Serrote 12
Figura 4-1 – Ligações ao minicomputador Raspberry Pi® 3 [26] 15
Figura 4-2 – Imagem do Getting Starded Guide para o SYSMAC CP1L [29] 16
Figura 4-3 – Ilustração das possibilidades dos PLCs CP1 da Omron 16
Figura 4-4 – Texto Estruturado no CX-Programmer 17
Figura 4-5 – Possibilidades de ligação com consola NB5 17
Figura 4-6 – Configuração Inicial Consola 18
Figura 4-7 – Arquitetura de uma rede MODBUS onde se podem interligar redes diferentes [31]. 18
Figura 4-8 – Exemplo de uma transação em MODBUS [31] 19
Figura 4-9 – Trama MODBUS em TCP/IP [32] 19
Figura 5-1 – Diagrama de blocos com os vários elementos do software criado 21
Figura 5-2 – Diagrama de atividade no servidor MODBUS [32] 23
Figura 5-3 – Organização de tabelas da base de dados. 28
Figura 5-4 – Organização e distribuição dos formulários 30
Figura 5-5 – Dashboard do Departamento de Manutenção 31
Figura 5-6 – Caixas de quantidade de intervenções não terminadas..... 32
Figura 5-7 – Relatório de intervenções que aguardam material 32
Figura 5-8 – Informação agrupada no relatório de Manutenção..... 33
Figura 5-9 – Lista de Formulários Relacionados com Pedidos de Intervenção 34
Figura 5-10 – Formulário “Pedido de Intervenção” 34
Figura 5-11 – Formulário de pedido de intervenção parcialmente preenchida. 36
Figura 5-12 – Escolha de bibliotecas de outras aplicações. 39
Figura 5-13 – Ordens de Trabalho 41
Figura 5-14 – Formulário Ordens Trabalho..... 42
Figura 5-15 – Formulário para encontrar referências de artigos 45
Figura 5-16 – Resultado da pesquisa executada no formulário frmFindMaterial 46
Figura 5-17 – Escolha do tipo de intervenção é feito pelo Departamento de Manutenção 47
Figura 5-18 – Desenho de SELECT query usando Access 48
Figura 5-19 – Tipo de querys disponíveis com Access 48
Figura 5-20 – Formulário de distribuição temporal mensal por tipo de intervenção..... 50
Figura 5-21 – Gráfico no Excel apresentado, ao ser pressionado o botão “Excel” em `frmOcupacaoTotal` 50

Figura 5-22 – Ocupação do recurso 293 no mês de Outubro 2016 53
Figura 5-23 – Gráfico em Excel com os registos de trabalho dos vários recursos no mês de Outubro53
Figura 5-24 – Query para organização das horas por recursos humanos e data 54
Figura 5-25 – Formulários associados com material no Departamento de Manutenção 54
Figura 5-26 – Formulário de adição de observação de características de materiais 55
Figura 5-27 – Formulário de adição de imagens 56
Figura 5-28 – Formulário de Escolha de imagens previamente carregadas na tabela tblImagens 58
Figura 5-29 – Exemplo de mensagem enviada para a secção de compras..... 59
Figura 5-30 – Pesquisa de E-mails enviados..... 62
Figura 5-31 – Formulário para gestão do inventário 62
Figura 5-32 – Formulário para editar artigos do inventário..... 63
Figura 5-33 – Formulários acessíveis na listagem de máquinas..... 63
Figura 5-34 – Formulário com informação das Máquinas 64
Figura 5-35 – Quadro indicadores máquina 23117 (1-11-2016)..... 64
Figura 5-36 – Formulário de adição de pontos de manutenção 65
Figura 5-37 – Listagem de pontos de manutençãoError! Bookmark not defined. 66
Figura 5-38 – Tabela com a lista de endereços MODBUS e os dados associados 66
Figura 5-39 – Relatório com os dados recolhidos da máquina..... 67
Figura A-1 – Organização dos formulários da aplicação de Gestão de Manutenção 82
Figura A-2 – Impresso “Pedido de Intervenção” 83

Índice de Tabelas

Tabela 5-1 – Tabela de endereçamento para os dados recebidos da máquina 27

Índice de Códigos

Código 5-1 – Shell script de arranque de outro script modbusserver 22
Código 5-2 – Shell script para arranque do servidor em Python (`/home/pi/modbusserver`) 22
Código 5-3 – Cabeçalho do ficheiro server.py. Módulos importados. 24
Código 5-4 – Main de server.py (servidor MODBUS TCP) 24
Código 5-5 – Gestor dos pacotes MODBUS. Avaliação da quantidade de bytes recebidos 25
Código 5-6 – Tarefa para escutar ligações na porta 502 do computador (Raspberry Pi® 3) 25
Código 5-7 – Decomposição dos bytes de informação pelos respetivos nomes 25
Código 5-8 – Verificação do MBAP 26
Código 5-9 – Execução do comando MySQL para inserir dados na tabela tblLog 27
Código 5-10 – Consulta para dados do gráfico de distribuição de intervenções não executadas 31
Código 5-11 – Evento ao clicar no identificador “Aguarda Material” no dashboard 32
Código 5-12 –Gerar data do pedido e datas para o ano em análise 33
Código 5-13 – Código SQL para selecionar dados para relatório 33
Código 5-14 – Linguagem SQL para recolha de dados para máquina e intervenção do sistema da empresa 35
Código 5-15 – Evento BeforeUpdate para o campo máquina 35
Código 5-16 – Consulta SQL do número mais pequeno da intervenção máquina “10101” 35
Código 5-17 - SQL para inserir dados nas tabelas 36
Código 5-18 – Função do módulo Logging para registo na tabela tblRegistoAlteracaoestado 37
Código 5-19 – Opção de enviar e-mail ao Responsável pelo Pedido 37
Código 5-20 – Função para envio de e-mail de alteração de estado da intervenção 38
Código 5-21 – Guardar dados após sair do formulário 39
Código 5-22 – Criação de diretoria e ficheiro texto para guardar dados do pedido de intervenção 40
Código 5-23 – Criação da mensagem a colocar no ficheiro Log 40
Código 5-24 – Verificação da existência de determinado registo de número de OT 41
Código 5-25 – Abrir formulário da Ordem de Trabalho pesquisada 41
Código 5-26 . Evento Form_Load() do formulário Ordens Trabalho 43
Código 5-27 – Avaliação BeforeUpdate após pesquisa de existência de referência 43
Código 5-28 – Dados de OpenArgs são utilizados para abrir nova referência 44
Código 5-29 – Envio de dados para o campo do formulário frmMaterialOrdemTrabalho 44
Código 5-30 – Ao executar duplo clique na caixa de texto txtRefInternaMiralago é desencadeado este evento 45
Código 5-31 – SQL para pesquisa da referência de artigos 45
Código 5-32 – Preenchimento da lista no formulário frmFindMaterial 46
Código 5-33 – Preenchimento dos dados da consulta nas caixas de texto respetivas 46
Código 5-34 – SQL para obtenção das horas registadas nas Ordens de Trabalho 48
Código 5-35 – Informação do tipo de intervenção ordenada por soma total de horas 49
Código 5-36 – Soma total de horas para o mês em análise 49

Código 5-37 – Criação dos objetos Excel e preenchimento dos dados 49
Código 5-38 – Construção do gráfico em VBA 51
Código 5-39 – Determinação de dias úteis incluindo as datas definidas 52
Código 5-40 – Query usando a função Crosstab do Access para organizar os dados por data/recurso humano 54
Código 5-41 – Procurar ficheiro com extensão .jpg [44] 57
Código 5-42 – Evento para escolha de imagem JPG do sistema de ficheiros 58
Código 5-43 – Pesquisa de texto dentro do endereço de imagem 58
Código 5-44 – Obtenção do nome do ficheiro da imagem a ser enviada por e-mail 59
Código 5-45 – Construção da mensagem a enviar à secção das compras 60
Código 5-46 – Envio de mensagens de E-mail utilizando diretamente o servidor externo 61
Código 5-47 – Evento “cmdClose_Click” promove a gravação dos valores de todo o formulário na tabela 63
Código 5-48 – Obtenção de número de registos com base nos critérios e somatório da diferença de datas 65
Código 5-49 – Consulta à base de dados para obtenção de informação de funcionamento da máquina 66
Código A-1 – Código do ficheiro colocado para fazer o arranque do programa 77
Código A-2 – Servidor MODBUS 80
Código A-3 – Código completo para verificação da existência de relação entre máquina e intervenção 84
Código A-4 – Criação de nova OT com base no pedido de intervenção. 85
Código A-5 – Função para encontrar número da OT na tabela tblOrdensTrabalho 86
Código A-6 – Criar diretoria e ficheiro de registo de eventos nos pedidos de intervenção 87
Código A-7 – Procura de Ordens de Trabalho e abertura de formulário respectivo 88
Código A-8 – Criação de ficheiro Excel com o gráfico de dados do tipo de manutenção 90
Código A-9 – VBA para criar gráfico de barras com mais do que uma série de dados 92
Código A-10 – VBA para criar segundo gráfico de colunas com a distribuição de horas por intervenção 93
Código A-11 – Procura ID na tabela tblMaterial que corresponde à referência na Miralago 94
Código A-12 – Evento Form_Open() no formulário frmMaterial 95
Código A-13 – Evento cmdClose_Click() no formulário frmMaterial 96
Código A-14 – Preenchimento das caixas de textos e listas para pesquisa de artigos 97
Código A-15 – Determinar o número de dias uteis entre datas 98
Código A-16 – Pesquisa de texto no endereço de imagens da tabela tblImagens 99
Código A-17 - Associar imagem a determinado material 100
Código A-18 – Método para adicionar ID da Imagem ao ID do Material 100
Código A-19 – Primeira parte de código para envio de e-mail e respetivo ficheiro anexo 101
Código A-20 – Envio de e-mail usando cliente do sistema e usando cliente externo (Gmail). 102
Código A-21 – Envio de E-mail usando CDO e Gmail 103
Código A-22 - Inicialização de variáveis e obtenção de datas para avaliação para o MTBF 104

Código A-23 - Querys para obtenção de informação da base de dados e aplicação da fórmula utilizada para MTBF 105

Simbologia e Abreviaturas

CDO - *Collaboration Data Objects*

CID – *Content-ID*

CMMS – *Computerized Maintenance Management System*

EAM – *Enterprise Asset Management*

EBIDTA – *Earnings Before Interest, Taxes, Depreciation and Amortization*

ERP – *Enterprise Resource Planning*

HTML – *Hyper Text Markup Language*

MPS - Manutenção Preventiva Sistemática

PLC – *Programmable Logic Controller*

SQL – *Structured Query Language*

URL – *Uniform Resource Locator*

1 Introdução

Segundo dados do Banco de Portugal, em 2013, a indústria metalomecânica representava cerca de um quarto do número de empresas, do volume de negócios e do número de pessoas ao serviço das indústrias transformadoras [1].

Este sector é constituído em 73% por microempresas (< 10 efetivos e <= 2 milhões de euros de volume de negócios) [2] [3] (Anexo do Decreto-lei 372/2007 de 6 de Novembro, Artigo 2º ponto 3).

Em 2013, cerca de 60% do volume de negócios teve origem no mercado externo [1].

Em termos financeiros, a pressão sobre o sector é relativamente reduzida, a informação individual referente a 2013 mostra que 25% das empresas da indústria metalomecânica não geram EBITDA suficiente para suportar os juros resultantes da sua divida financeira [1].

A Miralago S.A. [4], constituída em 1956, iniciou a sua atividade ligada à conceção, produção e comercialização de componentes para motociclismo e ciclismo. Hoje integra também a área *fitness*, comercialização de bicicletas e sistema *bike sharing* ligadas ao sector da mobilidade sustentável.

Em 2007 a função manutenção surgia dependente do departamento de produção pelo que todo o serviço prestado estava sob os critérios da mesma. As manutenções executadas eram registadas em documentos escritos, colocados nos dossiês das máquinas. A filosofia desses documentos era saber o que, e como agir, caso o mesmo problema surgisse. Não havia registo de tempos de paragem ou da intervenção realizada nem o custo total incluindo mão-de-obra.

Todas as empresas pretendem que os sistemas de produção e os equipamentos funcionem e sejam operados de forma fiável. Nenhuma organização pretende que os sistemas de produção parem, que produzam produtos de baixa qualidade ou que operem de forma ineficiente. Infelizmente nenhum ativo físico funciona sem falhas para sempre. Em muitas organizações, as paragens por quebra de funcionamento são a norma. Perdas de qualidade e de produtividade são elevadas. Os atrasos no envio de mercadoria são frequentes. Como a maioria destas deficiências são manifestadas como problemas relacionados com o equipamento, por exemplo, avarias ou ações de manutenção corretiva, a manutenção é demasiadas vezes culpabilizada por todos os problemas que proliferam neste tipo de instalações industriais. Na verdade, as razões para estes problemas inerentes são partilhados por todos os grupos funcionais [5].

A maioria do trabalho é feito numa base reativa.

O papel da manutenção deve mudar para suportar a crescente pressão global para crescer. Se os sistemas corretos, infraestruturas, processos e procedimentos forem seguidos e executados corretamente, as perdas podem ser minimizadas; haverá estabilidade nas operações; maximizar-se-á o volume de produção; e uma consistente qualidade de produto será a norma [5].

1.1 Motivação e contexto

Este trabalho foi realizado após identificada uma necessidade na empresa Miralago, local onde o autor do presente trabalho exerce funções de Responsável da Manutenção. A Manutenção Preventiva na Miralago está organizada em dois níveis. No “1º Nível” a manutenção é realizada pelo operador e no “2º Nível”, chamada de “Preventiva”, realizada a intervalos temporais maiores, pela Manutenção, face ao nível mais elevado de capacidade técnica necessário para a sua execução.

Lista de Instruções a Executar

Nº Equip.: 101.20			Designação Equipamento : Prensa hidraulica	
Marca : Mécanica Exacta			Modelo/Ano Fabrico : CC116	
Características :				
Instruções				
Ponto	**Freq.**	**Código**	**Material a utilizar**	**Observações**
1	SEM	VN410	Óleo GRXP150	Verificação dos níveis óleo de lubrificação

Indicação da localização onde Executar

Desenho/Croqui do Equipamento :

Figura 1-1 - Impresso orientador para a realização da Manutenção Preventiva de 1º Nível

Para cada um dos equipamentos foi criado um documento orientador chamado "Ficha de Intervenção de 1º Nível", ilustrado na figura 1-1, onde consta a periodicidade de execução das ações ("Freq.") e o material a utilizar.

O impresso "Registo de Intervenção de 1º Nível", ilustrado na figura 1-2, é preenchido com base na "Ficha de Intervenção de 1º Nível". É colocada uma cruz no quadro correspondente à linha do código da ação e à coluna do dia do mês.

A frequência com que é executada a intervenção/observação fica registada pela diferença de dias entre registos.

MIRALAGO — **Registo de Intervenção de 1º Nível** — Equip. Nº 101. 20 Marca: Mecanica Exacta — Modelo/Ano: CC160/2009

Mês : Setembro — Ano : 2014

Ponto	Data / Código	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	VN410																														
2	VN410																														
3	VN401																														
4	LZ802					X																									
5	LZ802									X																					
6	VC598					X																									
7	LU115				X	X																									
Rubrica																															

Observações :

MIR-MOD-52.7.192/02

Figura 1-2 - Impresso para registo de intervenções de Manutenção Preventiva de 1º Nível

Quanto à Manutenção Preventiva, realizada pelos técnicos de manutenção, ela apresenta uma organização similar. A figura 1-3 apresenta o impresso orientador para as intervenções com o título "Operações de Manutenção Preventiva". Estas são registadas no documento ilustrado na figura 1-4 ("Registo de Manutenção Preventiva") onde, no quadrado correspondente à linha do código e à coluna da semana é colocada uma cruz.

Com base nos documentos disponíveis, a informação que fica é muito escassa. Apesar do histórico existente de intervenções nas máquinas, pouco se consegue planear pois não existia uma base de dados de apoio para registo de informação e seu tratamento estatístico.

Em 2010 foi introduzido um sistema informático tipo ERP, com módulo de manutenção. Começou a ser possível introduzir dados das intervenções corretivas realizadas e fazer um acompanhamento estatístico das mesmas.

Foi decidido passar todos os impressos de Manutenção Preventiva para o sistema informático, sem qualquer alteração. Este iria gerar as ordens de manutenção do equipamento com base na periodicidade indicada. O problema é que as ordens estavam a ser geradas não pelo período de funcionamento da máquina, mas pelo período temporal decorrido.

Com a Manutenção Preventiva Sistemática (MPS) a periodicidade das intervenções é fixa [6]. O tempo contado está relacionado com o funcionamento da máquina. Apesar deste tipo de manutenção não contemplar o regime de funcionamento [6] é evidente que se o equipamento estiver parado a taxa de falhas será menor, não por melhoria da fiabilidade, mas por diminuição da taxa de ocupação da máquina.

Há a possibilidade de introduzir os dados de recursos utilizados bem como o material e a data de intervenção.

Lista de instruções a Executar

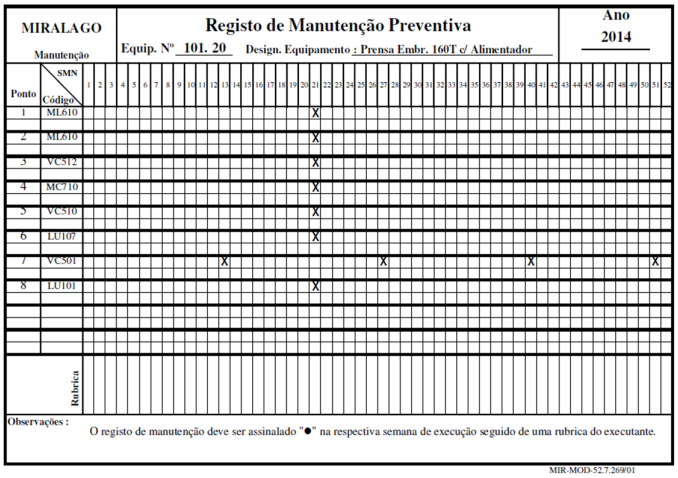
Nº Equip.:	**101.20**	**Designação Equipamento :**	Prensa Embr. 160T c/ Alimentador
Marca :	Mecânica Exacta	**Modelo/Ano Fabrico :**	CC160/2009 SR:3283
Características :			

Instruções

Ponto	Freq.	Código	Material a utilizar	Observações
1	ANO	ML610	Óleo GRXP150	Mudança do óleo sistema lubrificação

Indiçação do local onde executar a Intervenção

Figura 1-3 – Impresso Orientador para Manutenção Preventiva a executar pela Manutenção



MIRALAGO

Manutenção

Registo de Manutenção Preventiva

Equip. Nº 101. 20 Design. Equipamento : Prensa Embr. 160T c/ Alimentador

Ano 2014

Ponto	SMN / Código	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
1	ML610																					X																														
2	ML610																					X																														
3	VC512																					X																														
4	MC710																					X																														
5	VC510																					X																														
6	LU107																					X																														
7	VC501													X													X																				X					
8	LU101																					X																														
	Rubrica																																																			

Observações : O registo de manutenção deve ser assinalado "●" na respectiva semana de execução seguido de uma rubrica do executante.

MIR-MOD-52.7.269/01

Figura 1-4 – Impresso para registo da Manutenção Preventiva de maior periodicidade

A descrição do funcionamento do sistema não pode ser mais pormenorizada do que o indicado pois a aplicação limita-se a guardar os dados de registo numa base de dados. Não emite relatórios percetíveis, não permite exportar para outro tipo de ficheiro sem que seja necessário solicitar à empresa a criação dessas funcionalidades.

Na Manutenção Corretiva o funcionamento é muito similar. Os dados ficam registados no sistema para posterior consulta, mas não é possível obter outro tipo de informação. Trata-se de um caso de perpetuar a Manutenção Corretiva pois a não análise dos dados introduzidos ou o trabalho necessário para os analisar inviabiliza um melhor tratamento dos mesmo para planear futuras ações.

As soluções existentes no mercado, fora do ERP, nunca foram defendidas dentro da Empresa pois este *software* possuía um módulo de manutenção básico. Foi com base nas dificuldades de utilização do *software* e fundamentalmente nos sucessivos atrasos na implementação de soluções, por razões que não interessam aqui elencar, que surgiu uma necessidade de melhoria de recolha de dados e formas de análise.

1.2 Objetivos

O projeto desenvolvido pretende desenvolver uma solução de gestão de manutenção que preencha o vazio existente nas aplicações utilizadas na empresa em que o autor trabalha.

A solução idealizada recolhe dados de uma máquina para análise de tempos de funcionamento, número de ciclos e tempos de paragem.

Foi desenvolvida uma base de dados para introdução, por equipamento, das ações de manutenção realizadas. Inclui tempos gastos pelos recursos humanos, material utilizado e classificação das intervenções.

Com base nos dados recolhidos nas ordens de trabalho, pretende-se gerar indicadores de manutenção que facilitem o desenvolvimento da função manutenção.

1.3 Organização do documento

Este documento está dividido em sete capítulos tal como se sintetiza seguidamente:

- O primeiro capítulo contém a introdução à monografia, a contextualização, os objetivos e a organização do documento;
- O segundo capítulo contém o estado da arte relativamente aos *softwares* informáticos para Gestão de Ativos e Sistemas Computorizados de Gestão de Manutenção;
- O terceiro capítulo contém a arquitetura global do sistema e a forma como os vários blocos se interrelacionam;
- O quarto capítulo faz uma apresentação mais aprofundada do sistema em todo o *hardware* que permite o funcionamento global do sistema;
- O quinto capítulo faz uma descrição aprofundada do *software* desenvolvido para recolha de dados e a sua gestão;
- O sexto capítulo apresenta como está a ser utilizado o projeto dentro da empresa, as conclusões e trabalhos a desenvolver no futuro;
- O final da monografia é constituído pelas referências bibliográficas e os vários anexos mencionados ao longo dos capítulos.

2 Estado da Arte

A crescente importância da função de Manutenção dentro das organizações empresariais está a levar as empresas de *software* a apostar mais nos seus módulos de Manutenção.

Segundo Pinto (2013) [6] a generalidade dos sistemas comercialmente disponíveis suporta as diversas funcionalidades necessárias para a implementação, organização e gestão de um departamento de Manutenção avançado. Querendo com isto dizer que conduza ao melhor aproveitamento possível do tempo de vida dos equipamentos.

Numa pesquisa rápida sobre Software para Gestão de Manutenção [7] poderemos encontrar vários, com diversas características. De seguida apresenta-se apenas os que poderiam acarretar menos custos de implementação na orgânica da empresa em que o autor trabalha:

- **ManWinWin**– É possível fazer o download para teste e é de fácil utilização. Permite fazer controlo de stocks, gestão de ativos, planeamento de intervenções (Manutenção Preventiva), mapa de ordens de trabalho [8];
- **CentralGest Gestão da Manutenção e Gestão de Equipamentos** [9] – "O software de gestão de manutenção visa agilizar e gerir todo o tipo de ações que ocorrem no dia-a-dia com os recursos e equipamentos da empresa, podendo atuar ao nível de manutenções preventivas, corretivas e correntes (diárias). A gestão de equipamentos é fundamental para operações que têm como objetivo, alcançar as melhores decisões de rentabilidade e rapidez na alocação de recursos e equipamentos a determinados centros de custo, setores e departamentos, fábricas e obras ou até outros bens. Possui as características seguintes:
  - Gestão das manutenções preventivas ou planeadas;
  - Intervenção nas ações corretivas (Avarias) e Correntes (Diárias);
  - Plano de manutenções ao longo do ciclo de vida dos recursos e equipamentos;
  - Alertas associadas à execução do plano de manutenção, por listagem e e-mail;
  - Visualização das manutenções em atraso e as revisões que faltam realizar;
  - Realização de encomendas internas e externas;
  - Faturação associada à manutenção;
  - Fichas de identificação ajustadas a cada tipo de equipamento;
- O módulo **PHC CS Manufactor Gestão de Manutenção** permite a definição dos componentes constituintes de centros de trabalho e a interligação com o módulo PHC CS *Touch-Screen* com a possibilidade dos pedidos serem realizados diretamente do chão de fábrica pelos operadores [10];
- **Excel** – Esta ferramenta apresenta características suficientes para a criação de sistema de análise dos dados recolhidos. Em muitas empresas os dados podem estar subdivididos por vários documentos e não estar totalmente disponíveis para análise pelo Departamento de Manutenção. Em Administrações com pouca sensibilidade para as vantagens em ter um Departamento de Manutenção informado e com custos burocráticos mais baixos, colocar os dados numa tabela e gerar alguns gráficos que demonstrem o estado da Manutenção, ou falta dela, podem ser um argumento para que se tomem medidas que poderão levar à redução de custos com a manutenção;
- **Engeman** – O módulo Industrial permite o planeamento da manutenção preventiva e recolha de dados a partir de equipamentos móveis, bem como a solicitação de serviços utilizando o *Browser*. Permite programar as manutenções gerando cronogramas e respetivos alertas. Também podem ser adicionadas ou removidas caixas de texto dentro das janelas do programa consoante as necessidades de cada empresa [11];
- **DataStream**– *Software* de gestão de ativos com informação online de consumos do tipo de energia que se pretender. Apresenta gráficos de consumo e eficiência e propõem a

troca de equipamento com base nos rácios de eficiência medida. É possível visualizar dados de vibrações e outras métricas de operação. As ordens de trabalho são comunicadas para sistemas portáteis utilizados pelos técnicos [12];

- **Maximo** – Solução de gestão de ativos da IBM que permite gerir todos os aspetos relacionados com a manutenção. Desde edifícios até à verificação da máquina, passando pelos vários tipos de manutenção, aprovisionamento, inventário e prestadores de serviços externos. Utilizado em grandes grupos no ramo da energia, transporte ou até hospitais [13];
- ***Fleet Maintenance Pro*** – permite fazer o acompanhamento da manutenção corretiva e preventiva utilizando critérios como o tempo ou com recolha de dados diretamente do equipamento. Gera avisos por e-mail e permite a comunicação de Ordens de trabalho aos técnicos de manutenção [14];
- ***MP Software*** – Software de gestão do departamento com gestão de ativos. Não permite fazer recolha de dados dos equipamentos. Estes são registados manualmente caso se pretenda que a manutenção preventiva ocorra com base nas variáveis recolhidas da máquina. É possível gerar gráficos com base nos registos de dados das máquinas que, entretanto, se vão recolhendo [15];
- ***Manager Plus*** – Disponível em versão *Desktop* e *Cloud* e com três edições, sendo a *Enterprise* a mais completa. Os dados são gravados em MySQL. A apresentação é simples, com informações básicas sobre os ativos onde é possível adicionar qualquer documento com ele relacionado. Os dados têm que ser introduzidos manualmente mas tem a possibilidade de registo direto das ordens com o M+ [16];
- ***Fastmaint*** – Utiliza base de dados Microsoft Access ou Microsoft SQL Server mas não inclui a licença. A versão base permite o envio de SMS (depende do país) e e-mail para os técnicos de manutenção e o suporte para leitura de código de barras é um extra. Permite fazer uma gestão básica de ativos com adição de documentos e catalogar componentes das máquinas [17];
- **SigmaPDCA** – *Software* de gestão de ativos que utiliza a base de dados Firebird 2.0 (gratuito) [18] . A empresa comercializa soluções de *hardware* para comunicação de dados das máquinas para a base de dados (SIGMA Telemetria). Trata-se de um *software* muito completo se lhe for adicionada uma quantidade de opções pois a versão base não permite a utilização de um calendário. Tem versões gratuitas para estudantes testarem as suas funcionalidades [19];
- **CWorks *Easy*** – Solução completa, com acesso via *browser*, de gestão de ativos. Contempla a possibilidade de exportar os dados para folhas de cálculo (Excel). Nesta versão da empresa inclui a gestão de manutenções preventivas e emissão das respetivas ordens de trabalho [20];
- **Gmac.2** – Gestão de ativos *online* é a proposta de *software* desta empresa portuguesa. A base de dados pode ser MySQL, Microsoft SQL Server, *PostgreSQL* ou Oracle. Contém uma ferramenta de comunicação para smartphone que pode ser instalada nos vários sistemas operativos (Android, iOS, Windows ou *Blackberry*) e funcional também a partir do *browser*. Permite a recolha *online* de dados sobre consumos ou outras variáveis dos ativos [21].

3 Arquitetura do sistema

3.1 Enquadramento

Ao longo do meu percurso dentro da empresa sempre houve alguma dificuldade em implementar soluções modernas de planeamento na Manutenção. A escassez de recursos humanos levou sempre a manutenção para o caminho das manutenções corretivas de emergência.

O Microsoft Excel® tem sido utilizado para gerir e analisar grande quantidade de dados nas organizações, sejam elas empresas como a Miralago sejam outras de maior dimensão. Numa primeira fase, dentro da empresa, o autor iniciou a introdução de dados utilizando esta ferramenta. O objetivo era ficar com o registo de tempo gasto por intervenção, para que ao longo do tempo as manutenções planeadas pudessem ser calendarizadas num cronograma.

O Microsoft Access® é uma ferramenta de gestão de base de dados que (muitos dirão) é o passo seguinte para o analista que se depara com o constante aumento no volume de dados. Não tem número de colunas pré-determinado e consegue gerir as relações entre tabelas diferentes. Para além disso, o Access® vem com as ferramentas necessárias que nos ajudam a distribuir aos utilizadores as aplicações desenvolvidas [22]. Como é uma ferramenta acessível em computadores com o Office o caminho parecia estar traçado para que fosse esta a ferramenta a utilizar de seguida. O caminho estava traçado, falta agora construí-lo.

3.2 Arquitetura do sistema

Antes de iniciar a descrição da arquitetura convém esclarecer por que razão foi proposta esta solução de gestão de Manutenção.

Apesar de existirem muitas ferramentas informáticas para fazer gestão da manutenção, estas nunca foram implementadas na empresa. O autor pensa que as vantagens de implementar uma solução, de raiz, para a gestão da manutenção é a flexibilidade e a principal desvantagem será não ter a quem recorrer quando se pretende fazer uma correção ou melhoria.

Os requisitos definidos para o sistema a desenvolver foram os seguintes:

- Sistema personalizado de Gestão de Manutenção (para uso profissional);
- Recolha de dados de máquinas através de MODBUS TCP;
- Base de dados MySQL para armazenamento dos dados das máquinas;
- *Aplicação Access® com dashboard* dos principais indicadores instantâneos de Manutenção e indicador de número de intervenções pendentes e sua classificação;
- Possibilidade de gerar vários tipos de relatórios sobre as intervenções realizadas e sobre os tipos de intervenções pendentes;
- Lista de Equipamentos com listagem instantânea das Ordens de Trabalho executadas;
- Lista de Materiais/Produtos com possibilidade de adicionar imagem para mais rápida identificação, *datasheet*, preço e fornecedor;
- Possibilidade de envio de e-mail para o Departamento de Logística para aquisição dos materiais e proposta de abertura de novas referências;
- Gestão dos dados associados aos módulos tradicionais de um programa de manutenção, como por exemplo, fornecedores, recursos humanos, pedidos de intervenção, ordens de trabalho, etc …

Com base nos requisitos precedentes, foi desenhado um diagrama geral da Base de Dados Access, cujo diagrama de blocos é apresentado na figura 3-1. A coloração do diagrama procura identificar níveis de função para estabelecer uma relação entre a Manutenção Corretiva Planeada e não Planeada e a Manutenção Preventiva Planeada.

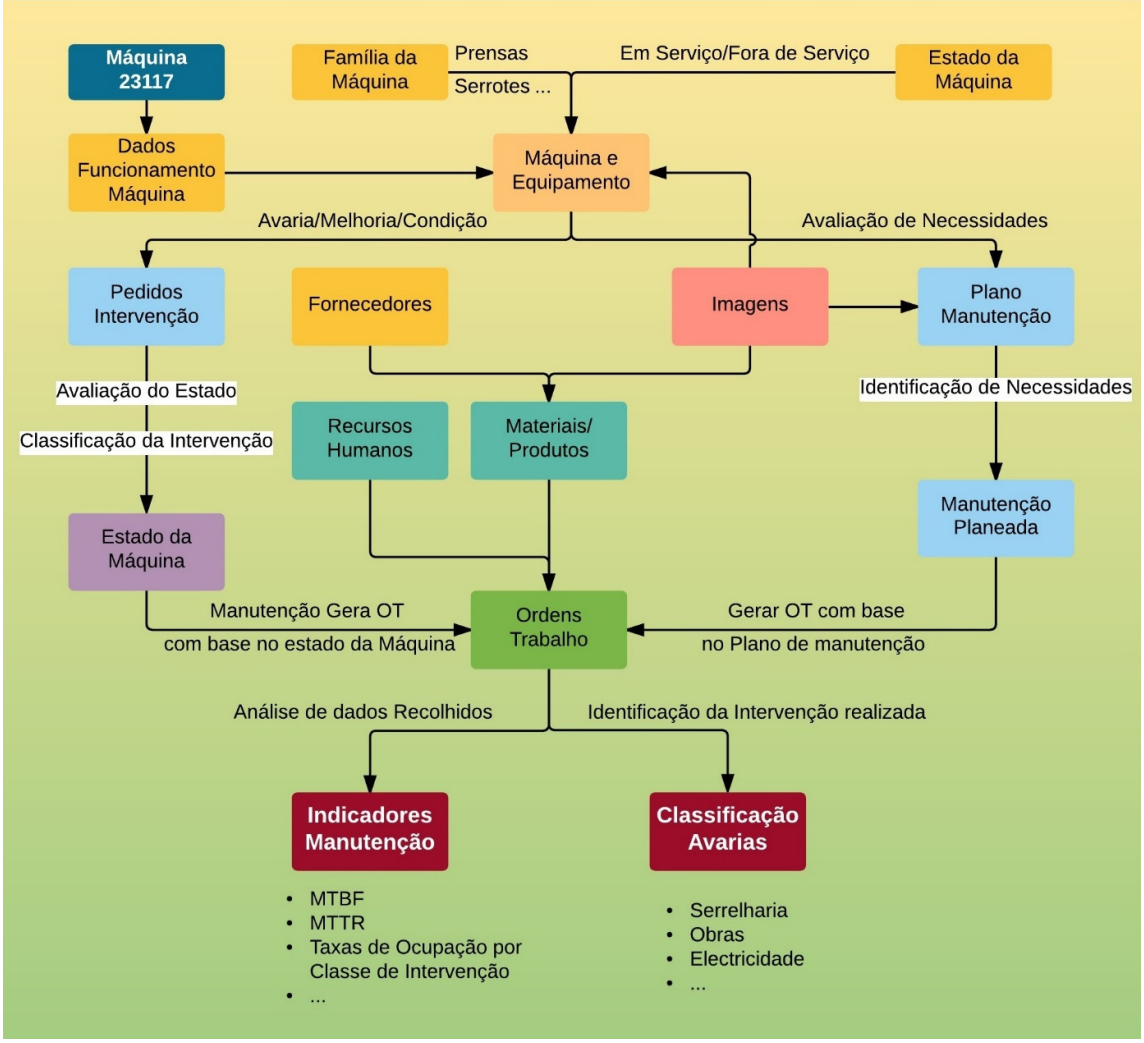


Figura 3-1 – Diagrama de Blocos geral do fluxo de informação

3.2.1 Elementos do sistema de Gestão de Manutenção

Na figura 3-2 encontra-se o diagrama de blocos de todo o sistema, nomeadamente, base de dados, Raspberry PI®, autómatos instalados no equipamento escolhido como caso de estudo – Serrote Automático

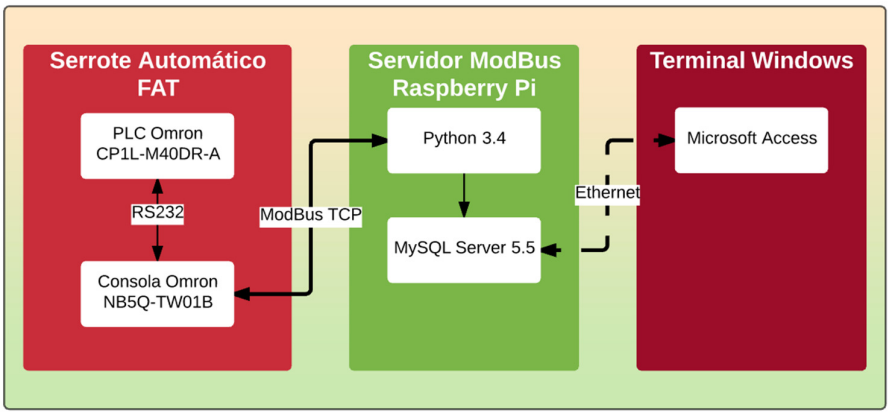


Figura 3-2 - Diagrama de blocos do sistema de informação. Serrote Automático FAT (Máquina 23117)

O Serrote Automático FAT apresenta matrícula de 1999. Sofreu uma intervenção em 2014 que consistiu na renovação e modernização do sistema de controlo. Foi-lhe adicionada uma consola Omron® NB5Q-TW01B e trocado o autómato pelo Omron® CP1L-M40DR-A.

Este equipamento permite cortar tubos em ferro ou aço até 100mm de lado ou 80mm de diâmetro com o comprimento máximo de 6000mm. Permite cortes perpendiculares, ou em ângulo, com um comprimento máximo de 3000mm

Resumidamente, este Serrote opera da seguinte forma (ignorando o procedimento de *setup*):

1. O tubo é carregado manualmente na bandeja de carregamento do Serrote;
2. A máquina carrega um tubo para iniciar o ciclo;
3. Ao ser detetado, o tubo é puxado até ao fim-de-curso que foi afinado no *setup*;
4. Neste momento o tubo fica fixo por dois tornos pneumáticos acionados pela máquina;
5. O disco de corte desce até à posição definida no *setup* cortando o tubo;
6. O tubo cortado é então descido para o caixote de receção e escorrimento do óleo de corte;
7. Após retomada a posição da bandeja de descarregamento do tubo cortado, a máquina puxa novamente o tubo a cortar, reiniciando o ciclo;
8. Após o tubo restante passar determinado ponto, assinado por um primeiro sensor indutivo, é carregado novo tubo que empurra o que estava a ser cortado;
9. Desta forma diminui-se o comprimento de tubo para sucata pelo aproveitamento máximo do comprimento de tubo disponível.

O autómato instalado não possui porta Ethernet, foi necessário utilizar a consola como *gateway* de comunicação dos dados.

Relativamente à consola Omron®, as portas RS232C, RS485, RS-422A e Ethernet estão disponíveis para comunicação [23]. Neste projeto está a ser utilizada a porta COM1 RS232 da consola para ligação com o PLC da máquina (figura 3-3 OMRON C Series PLC0:0) e a porta “Net, ligada à rede *Ethernet* das máquinas na Miralago. O equipamento “PLC1:1”, fornece o serviço de MODBUS Slave dentro do protocolo MODBUS TCP [24] e a consola NB5 (“HMI0”) é o MODBUS Master.

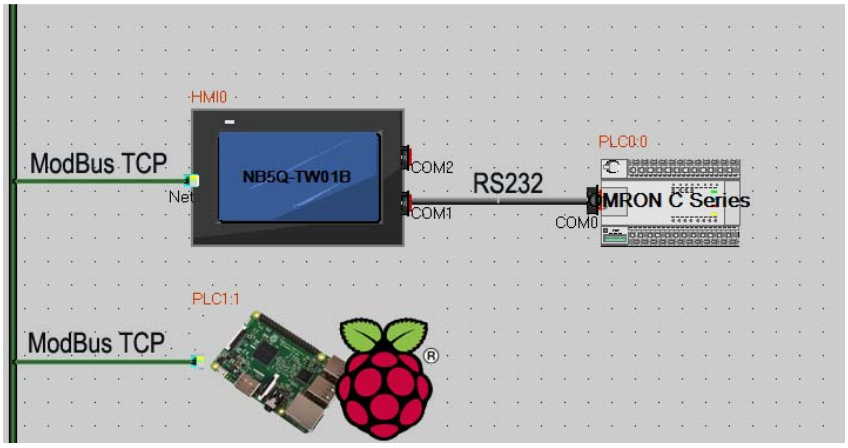


Figura 3-3 – Representação das portas de comunicação na consola NB5 do Serrote

Sabendo que a consola apenas permite trabalhar em modo *Master*, dentro do protocolo escolhido, tornou-se necessário implementar uma solução *Slave* para recolha da informação. Existem soluções comerciais dentro da Omron® mas, no sentido de controlo de custos, optou-se por utilizar o Raspberry Pi® para fazer a função de servidor de *MODBUS* TCP utilizando a linguagem Python.

3.2.1.1 Servidor MODBUS TCP

Os dados enviados a partir da consola, por *MODBUS* TCP, são trabalhados pela aplicação Python que os trata e valida. Daqui os dados são enviados para a tabela respetiva da base de dados MySQL.

Para correr o servidor *MODBUS* foi utilizado o miniPC (Raspberry Pi®) com o sistema operativo *Raspbian Jessie* (https://www.rasberrypi.org) [25] onde foi instalada o MySQL Server 5.5 [26] e a versão 3.4 do Python.

O Raspberry Pi® 3 Model B [25] apresenta agora maior capacidade de processamento (CPU com 1,2Ghz 64-bit quad-code ARMv8 enquanto na versão 2 o CPU tinha 900Mhz quad-code ARM Cortex-A7), o que o torna um pouco melhor para o que se pretende com este trabalho.

Foi estabelecido o arranque automático do Raspberry PI® em caso de quebra de energia assim como da aplicação em Python que ao receber os dados da máquina os coloca nas tabelas designadas.

3.2.1.2 Terminal (PC) Windows

Qualquer computador correndo Windows é suficiente para usar a aplicação criada. Não é necessária uma instalação completa do Access 2016 para poder usar a aplicação nem licença Office [27]. Em computadores sem o Microsoft Access instalado deve ser instalado o Access 2016 Runtime para correr a aplicação.

3.3 Sumário

Neste capítulo foi apresentada uma perspetiva geral do sistema que corresponde ao dispositivo implementado no âmbito do projeto de mestrado, as suas funcionalidades, bem como o funcionamento genérico do dispositivo.

Foi ainda apresentado um diagrama de blocos do sistema e uma descrição de cada um dos seus módulos, fazendo referência às características funcionais a que este tem de responder

4 Ferramentas e Equipamentos Utilizados

Neste capítulo são apresentadas todas as ferramentas utilizadas, tanto a nível de *hardware* como de *software*.

4.1 Raspberry Pi ® 3 Model B+

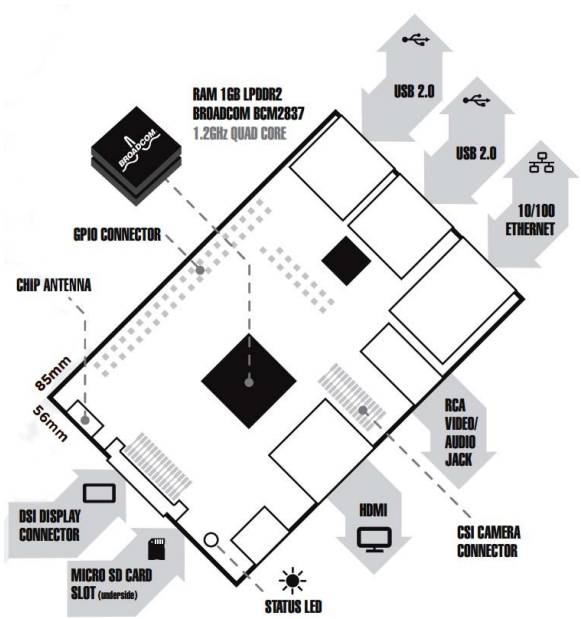


Figura 4-1 – Ligações ao minicomputador Raspberry Pi® 3 [25]

O Raspberry Pi® 3 é um computador do tamanho de um cartão de crédito com maior capacidade e mais funcionalidades que os seus antecessores. Para além das ligações existentes em modelos anteriores, o Raspberry Pi® 3 apresenta conectividade por wireless LAN e Bluetooth [25].

Para este projeto foi utilizada a porta Ethernet, uma fonte de alimentação e um cartão microSD onde se encontra instalado o Sistema Operativo Raspbian Jessie®

A comunicação com o computador é feita através do servidor SSH que vem pré-ativado para arrancar quando o computador é ligado.

4.2 Autómato Omron® CP1L-M40DR-A

Segundo a Omron esta série de PLCs permite automatizar máquinas compactas e executar qualquer tarefa de automação simples rápida e facilmente. Pode ser ligado a consolas, servo motores, inversores, controladores de temperatura e outros dispositivos para criar sistemas de baixo custo e eficientes (Figura 4-3).

Da figura 4-2 destaca-se 24 entradas DC (8), a ligação USB (2) que permite programar e monitorizar a actividade do PLC através de um computador, *slots* para ligar placas opcionais (10) onde podem ser ligadas placas com RS232C (16) ou RS-422A/485 (17).

No Serrote, foi instalada a placa com o módulo RS232C (16). É através deste porto que está ligada a consola Omron NB5.

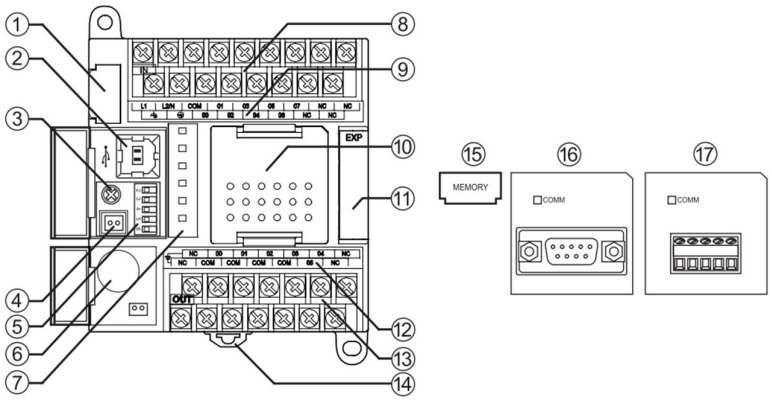


Figura 4-2 – Imagem do *Getting Starded Guide* para o SYSMAC CP1L [28]

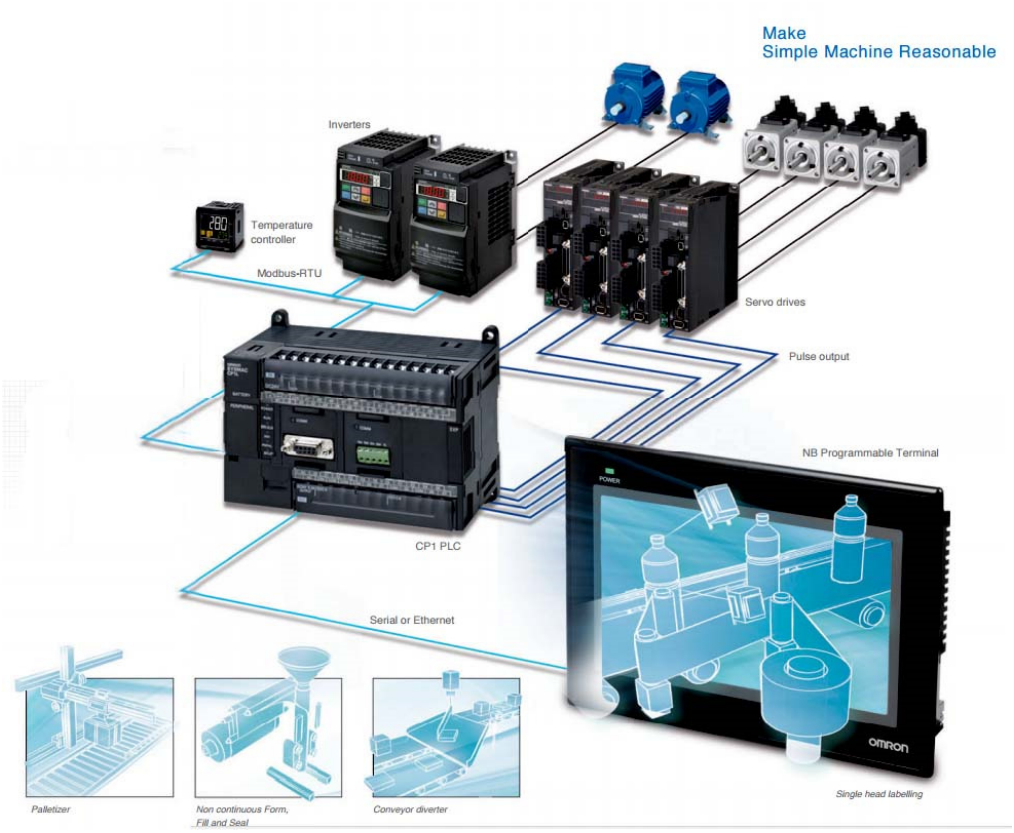


Figura 4-3 – Ilustração das possibilidades dos PLCs CP1 da Omron

4.2.1 Software programação CX-Programmer

O CX-Programmer é o *software* da Omron para programação de todas as séries de PLCs. Pode-se utilizar blocos de funções *standard* em texto estruturado em conformidade com IEC 61131-3 ou a linguagem ladder convencional, como linguagens de programação.

Inclui ferramentas de simulação dos PLCs para testar alguma funcionalidade antes de exportar por USB, ou outra ligação série, para o autómato.

A função de texto Estruturado (ST) (Figura 4-4) permite mais facilmente realizar processamento numérico ou comparações lógicas.

```
For loop := 0 TO 100 DO
      Total := Total + Val[loop];
End_For;

IF Contact26 = TRUE THEN
      (* Calculate Average *)
      Average := Total / 100;
END_IF;
```

Figura 4-4 – Texto Estruturado no CX-Programmer

4.3 Consola Omron® NB5Q-TW01B

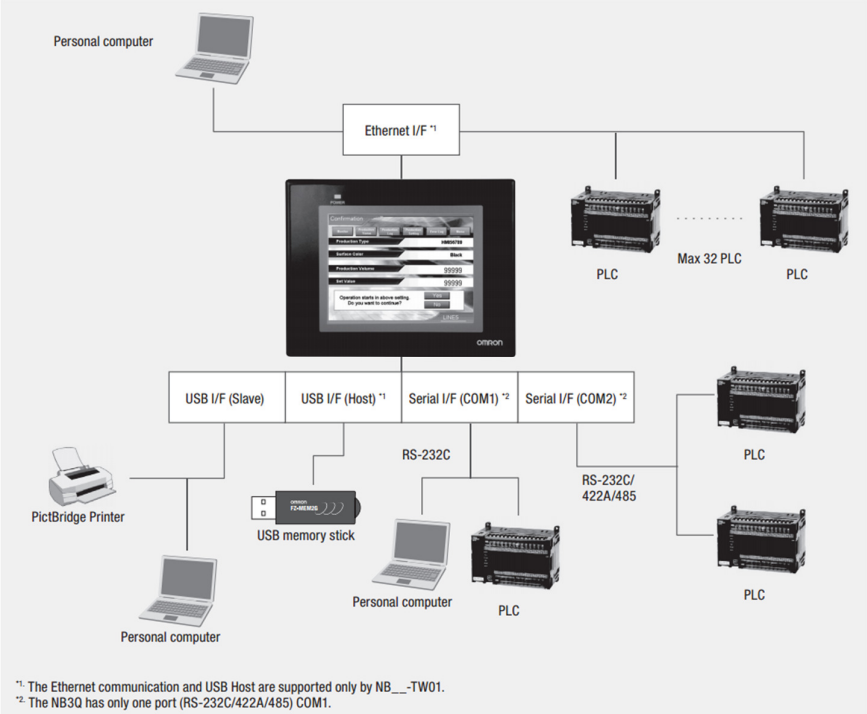


Figura 4-5 – Possibilidades de ligação com consola NB5

A consola utilizada, como se pode verificar na figura 4-5, apresenta bastantes possibilidades de ligação. Para além das ligações utilizando a porta RJ45 (*MODBUS* TCP e *Ethernet*) existem ainda as ligações série onde é possível ligar uma impressora, um computador e/ou um PLC.

4.3.1 Software NB-Designer

Para a programação da consola é necessário instalar o software NB-Designer. A ligação para programação pode ser feita diretamente utilizando a porta USB, a porta série ou remotamente através da porta *Ethernet* (Figura 4-5).

Para podermos aceder à consola é necessário configurar a mesma conforme as ligações que estão, ou vão ser implementadas (Figura 4-6).

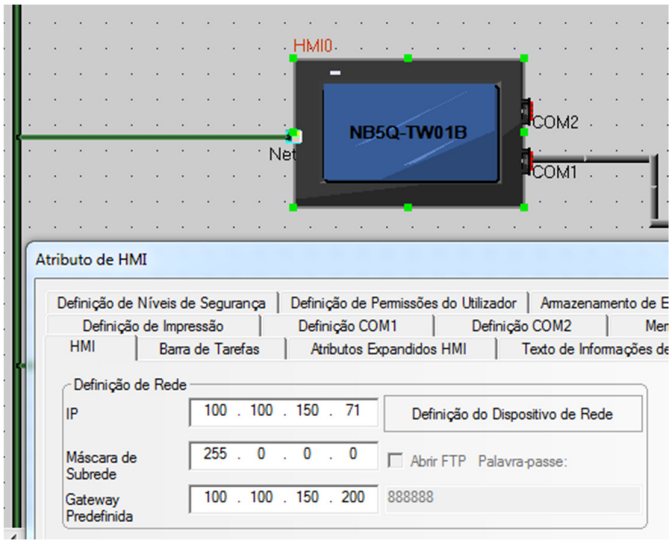


Figura 4-6 – Configuração Inicial Consola

4.4 *MODBUS* TCP

MODBUS é um protocolo de mensagens do nível de aplicação, posicionado no nível 7 do modelo OSI [29]. Proporciona comunicações tipo cliente/servidor a dispositivos ligados em diferentes tipos de barramentos ou redes, como se pode verificar na figura 4-7.

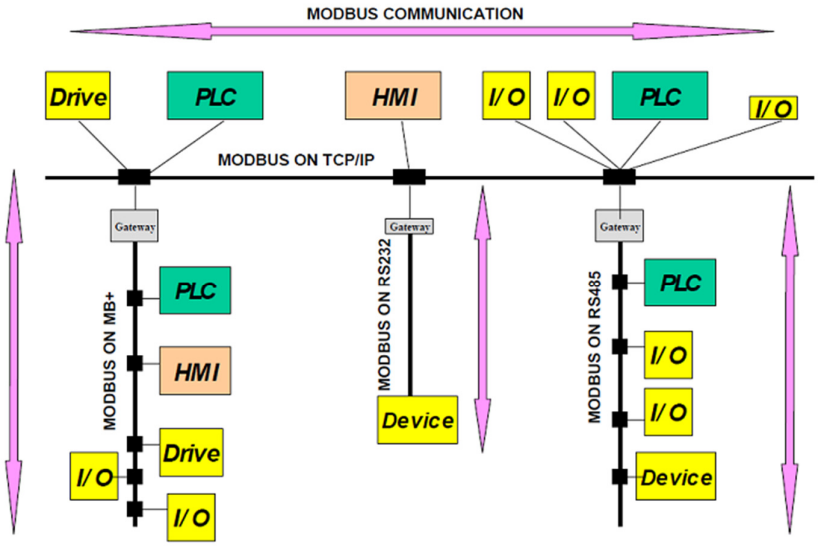


Figura 4-7 – Arquitetura de uma rede *MODBUS* onde se podem interligar redes diferentes [30].

É um protocolo tipo pedido/resposta (Figura 4-8) e oferece serviços especificados por códigos de funções. A comunidade da *Internet* pode aceder ao *MODBUS* pela porta de sistema 502 na pilha TCP/IP [30].

Sockets são os blocos base de comunicação. Em *MODBUS* TCP a comunicação é executada através de *sockets* [31]. Para verificar a implementação de *sockets* neste projecto verificar o capítulo 5.2 Servidor *MODBUS* TCP (*Slave*).

Numa rede *MODBUS* TCP/IP o pacote de dados apresenta algumas particularidades relativamente a outras redes.

Na figura 4-9 pode-se verificar a particularidade da trama pela inclusão, no cabeçalho da mensagem, de um conjunto de bytes que identificam a mensagem e o protocolo; o número de bytes da mensagem e a unidade de onde é proveniente. Este cabeçalho, com o comprimento de 7 bytes, tem o nome de *MODBUS Application Protocol* (MBAP).

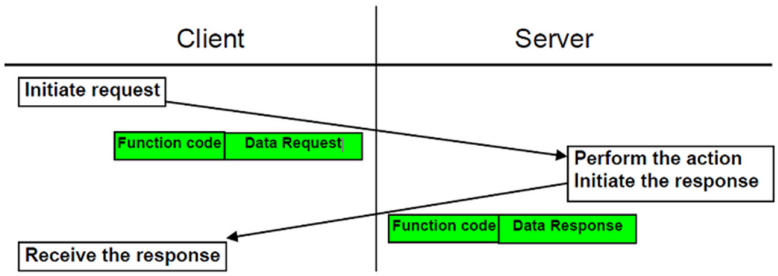


Figura 4-8 – Exemplo de uma transação em *MODBUS* [30]

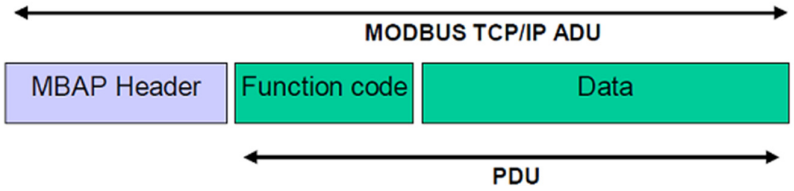


Figura 4-9 – Trama MODBUS em TCP/IP [31]

4.5 Computador com Sistema Operativo Windows® 7

Qualquer computador ou *tablet* que tenha instalado a versão 2016 do Access ou o Access 2016 *Runtime* pode usar a aplicação de gestão criada. Na versão final da aplicação foi colocado o conjunto de tabelas num computador da rede a os formulários (*front-end*) são utilizados nos computadores dos utilizadores no departamento de manutenção.

4.5.1 Microsoft Access® 2016

A 13 de Novembro de 1992 foi lançada a versão 1.0 do Microsoft Access proveniente do projeto com código Cirrus.

O Programa Microsoft Access faz parte do pacote Office da Microsoft mas pode ser vendido separadamente. É um sistema de gestão de Base de Dados (DBMS) que combina a *Microsoft Jet Database Engine* com uma *interface* gráfica e ferramentas de desenvolvimento de *software* [32].

O Access pode importar ou ligar-se diretamente a dados guardados em outras aplicações ou Base de Dados.

Trata-se de uma ferramenta fácil de utilizar mas que perdeu muito por não acompanhar a expansão da nuvem para acesso à informação.

4.5.1.1 Visual Basic for Applications (VBA)

A implementação do Visual Basic (VB) da Microsoft nas aplicações Office tem o nome de Visual *Basic for Applications* (VBA) [33].

Quem desenvolve aplicações em Access utiliza macros de vez em quando. Apesar das macros proporcionarem uma forma rápida e fácil de automatizar a aplicação, escrever módulos em VBA revelou-se a melhor forma de criar a aplicação [22]. O VBA permite criar ciclos e ramificar a informação, enquanto que com as macros a flexibilidade é muito menor.

Nas últimas versões do Access as macros ganharam mais funcionalidades, como o gestor de erros, mas são sempre menos flexíveis.

4.6 Sumário

O presente capítulo apresentou os equipamentos, protocolos e ferramentas de programação utilizadas para construir o sistema que se apresenta na presente monografia.

Desde os equipamentos profissionais, aos equipamentos educacionais e às ferramentas de desenvolvimento, o leitor poderá averiguar as diferentes tecnologias envolvidas caso pretenda construir um sistema semelhante, quer em ambiente profissional quer em ambiente académico.

5 Desenvolvimento de *software*

Neste Capítulo aborda-se o desenvolvimento do software necessário ao funcionamento completo do sistema de Gestão de Manutenção. No presente caso, o sistema de Gestão de Manutenção é constituído por uma base de dados em Microsoft Access, um servidor MySQL instalado numa unidade Raspberry PI® e por uma unidade industrial de aquisição de dados, suportada por tecnologia do fabricante Omron acoplada a uma máquina de corte de tubos – as unidades do sistema foram descritas nos Capítulos anteriores.

Cada uma dessas unidades comporta *software* desenvolvido especificamente para a recolha, análise e introdução dos dados da máquina e do trabalho desenvolvido pelo Departamento de Manutenção. A figura 5-1 ilustra os vários blocos de *software* que fazem parte do projeto e está relacionada com o *hardware* que se pode observar na página 11. Nos primeiros dois blocos (bloco 1 e 2) foram criadas as adaptações aos programas no PLC e na consola NB com as versões definitivas nos ficheiros mencionados. Os blocos 3 e 4 permitem guardar os dados da máquina na base de dados MySQL. No bloco 5 encontra-se a base do desenvolvimento de todo o projeto. É nele que toda a informação é analisada e que é iniciado o processo de manutenção de um equipamento.

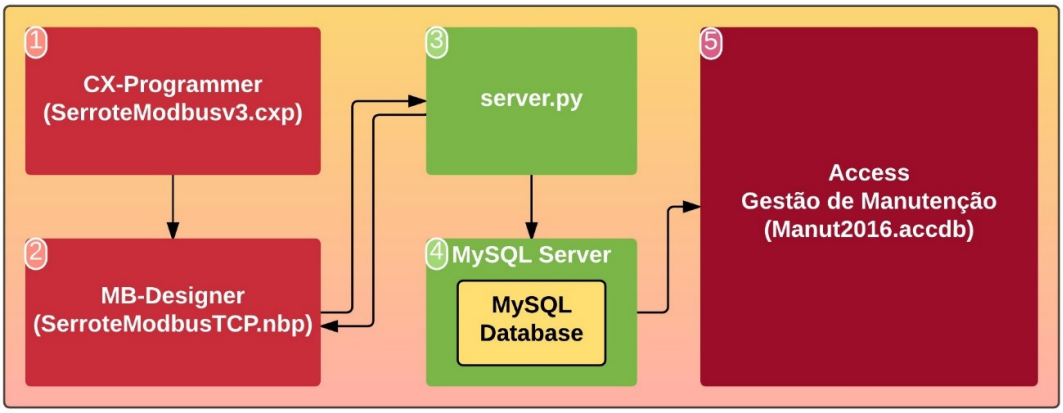


Figura 5-1 – Diagrama de blocos com os vários elementos do *software* criado [34]

O desenvolvimento de *software* da aplicação em Microsoft Access será discutida em detalhe na secção 0 face ao seu enfoque basilar neste trabalho. No presente Capítulo será então dado ênfase ao desenvolvimento da aplicação instalada no Raspberry PI, nomeadamente desde o desenvolvimento dos ficheiros *bash* necessários ao arranque automático do servidor MODBUS, passando pelo programa servidor em Python, à base de Dados MySQL para guardar os dados da máquina até ao software dos equipamentos Omron.

Assim:

- as seções 5.1 e 5.2 estão dedicadas ao *software* desenvolvido para o Raspberry PI®;
- a seção 5.3 está dedicada ao *software* da aplicação em Microsoft Access - ferramenta gestora de base de dados, os seus formulários e relatórios.

5.1 Arranque do servidor MODBUS TCP (Raspberry Pi®)

O objetivo de recolher dados de equipamentos e máquinas é uma questão importante. Com a necessidade de independência da intervenção humana na recolha desses dados, um aspeto

importante, no Raspberry PI®, é o arranque automático de todos os serviços quando é fornecida energia ao equipamento. Desta forma, os sistemas de recolha de dados deverão ser automáticos e autónomos por forma a estarem preparados para quebras de energia e também registar quebras de comunicação.

Na presente implementação, sempre que o sistema de recolha reinicia ou há falha de energia, o Raspberry PI® reinicia-se automaticamente, restabelecendo a comunicação com os equipamentos Omron instalados na máquina de corte de tubos. Exemplifica-se de seguida dois exemplos de *scripts* de inicialização, sendo os restantes serviços um aspeto integrante do sistema operativo do Raspberry PI® quando instalado pelo administrador do sistema operativo.

No projeto foi criado um *script* com o código presente no anexo i – código para o modbus server (modbusserver.sh) [35]. Este foi colocado na diretoria `/etc/init.d` seguindo o procedimento para que o Raspberry Pi o reconheça como um programa a correr no arranque (*boot*), nomeadamente tornar os ficheiros executáveis (server.py, modbusserver.sh e modbusserver). Isso é conseguido na linha de comandos, com `chmod 755 nomedoficheiro` nas respetivas diretorias; colocar a primeira linha do ficheiro server.py com o endereço do interpretador Python pretendido `(#!/usr/local/bin/python3)`; para além disso é necessário, mais uma vez na linha de comandos, escrever o comando `sudo update-rc.d modbusserver.sh defaults` que adiciona uma ligação simbólica na diretoria /etc/rc?.d para que o *script* corra nas ocasiões definidas no sistema operativo como *default*.

```
1   DIR=/home/pi/
2   DAEMON=$DIR/modbusserver
3   DAEMON_NAME=modbusserver

4   DAEMON_OPTS=""

5   DAEMON_USER=root

6   PIDFILE=/var/run/$DAEMON_NAME.pid

7   . /lib/lsb/init-functions

8   do_start(){
        a   log_daemon_msg "Starting system $DAEMON_NAME deamon"
        b   start-stop-daemon --start --background --pidfile $PIDFILE --make-pidfile -
            -user $DAEMON_USER --chuid $DAEMON_USER --startas $DAEMON -- $DAEMON_OPTS
        c   log_end_msg $?
9   }
10  do_stop () {
        a   log_daemon_msg "Stopping system $DAEMON_NAME daemon"
        b   start-stop-daemon --stop --pidfile $PIDFILE --retry 10
        c   log_end_msg $?
11  }
```

Código 5-1 – *Shell script* de arranque de outro *script* modbusserver

O código 5-1, que se encontra na diretoria `/etc/init.d`, é parte do *shell script* de arranque modbusserver.sh. Na linha 1, indica-se a localização do ficheiro a correr (`/home/pi/`), na linha 2 o nome do ficheiro (`modbusserver`) e na linha 3 é indicado o nome que o sistema deve atribuir ao processo residente (DAEMON) [36] para correr o serviço. Na linha 6 é indicado o ficheiro onde é guardado o PID (*Process Identifier*), quando o serviço é iniciado [37].

```
1 #!/bin/bash
2 # Arranque do Servidor Modbus TCP

3 clear
4 echo "Starting Modbus Server"
5 sudo python3 /home/pi/ModbusServer/server.py
```

Código 5-2 – Shell script para arranque do servidor em Python (`/home/pi/modbusserver`).

Quando o sistema operativo arranca irá iniciar o processo para que o ficheiro `modbusserver` seja executado. Este (Código 5-2) coloca o servidor MODBUS em funcionamento com o comando `sudo python3 /home/pi/ModbusServer/server.py`.

De seguida passa-se à descrição do funcionamento deste servidor.

5.2 Servidor MODBUS TCP (*Slave*)

O servidor ModBus TCP foi programado em Python 3 (Python 3.5). Durante o desenvolvimento foram criadas algumas funções para: receber as comunicações na porta 502; verificar se a estrutura do pacote está de acordo com as descrições técnicas do protocolo [31]; verificarr pedidos de leitura ou escrita e envio dos dados para o servidor SQL (na presente implementação, envio de dados pelo equipamento Omron instalado na máquina de corte de tubos).

O livro "*MODBUS Messaging on TCP/IP Implementation Guide*" [31], no ponto 4.4.2 (MODBUS Server) orienta na implementação de um programa servidor. Com base nesta bibliografia foi criado o fluxograma da figura 5-2, cuja cor verde das caixas, ilustra a parte deste protocolo que está efetivamente implementada nesta solução. Este fluxograma documenta de forma exata a implementação efetuada em código python.

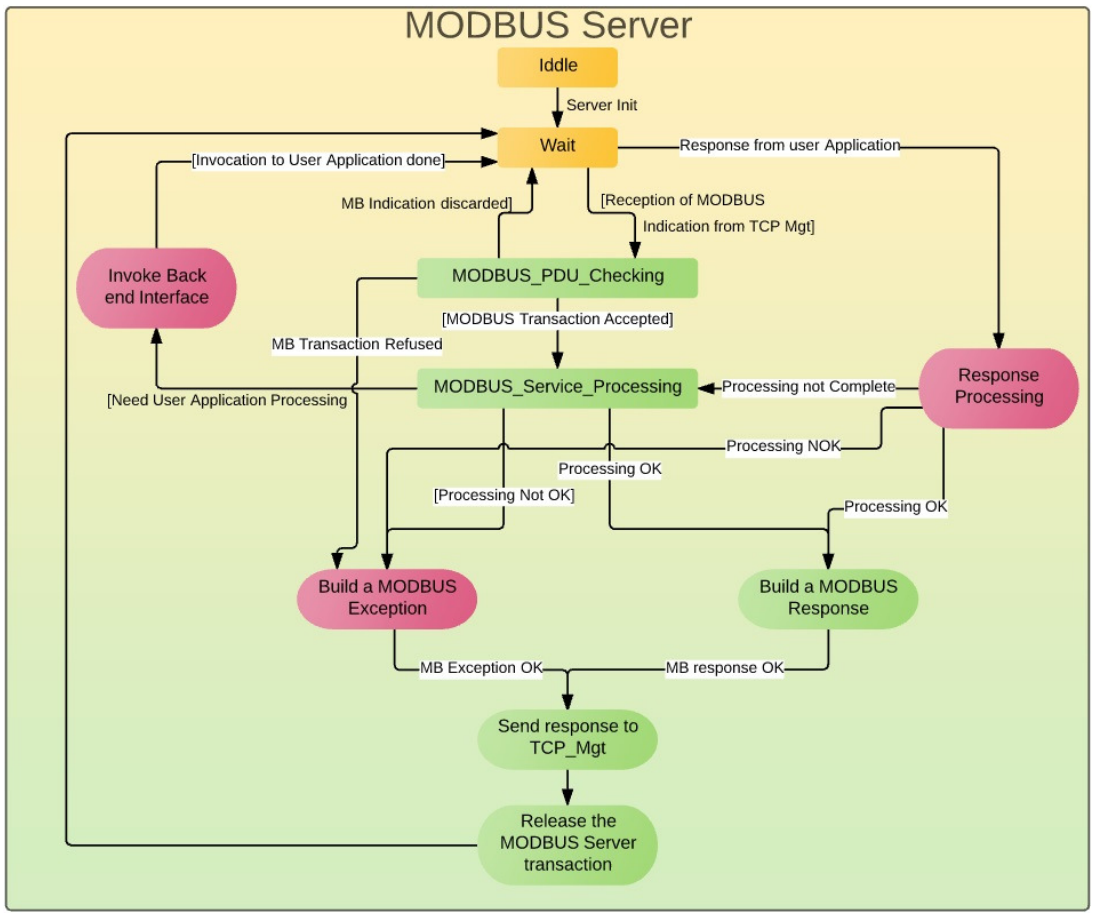


Figura 5-2 – Diagrama de atividade no servidor MODBUS [31]

Os módulos mais relevantes utilizados (Código 5-3) permitem:

- a utilização de *sockets* para a ligação TCP;

- gerir várias ligações com o servidor de forma concorrente tendo-se utilizado para isso o módulo *asyncio;*
- analisar a estrutura de dados recebidos com o módulo *struct*;
- trabalhar com a base de dados MySQL.

```
1  #!/usr/local/bin/python3
2  # Ficheiro server.py servidor MOdbusTCP na porta 502
3
4  from socket import *
5  import asyncio, struct,time
6  import pymysql as sql
7
8  # Endereco do MySQL Server
9  EnderecoServidor = '127.0.0.1'
```

Código 5-3 – Cabeçalho do ficheiro server.py. Módulos importados.

O módulo *asyncio* gere um conjunto de tarefas que ocorrem de forma concorrente. Para fazer essa gestão é criada a tarefa a ser executada, que depois é colocada num ciclo infinito (`loop.run_forever()` código 5-4).

```
1  # Loop necessário para utilizar o módulo asyncio em Python 3.5
2  loop = asyncio.get_event_loop()
3
4  # Arranque da task modbus_server e escutar na porta 502
5  # Criar a tarefa que abre a co-rotina modbus_server (tuple)
6  loop.create_task(modbus_Server(('',502)))
7
8  #corre o ciclo de tarefas para sempre
9  loop.run_forever()
```

Código 5-4 – Main de server.py (servidor MODBUS TCP)

Na linha 6 do código 5-4 é adicionada a tarefa para aguardar pela recepção de dados na porta 502 (ModBus) e na linha 9 é executada a função do módulo que coloca a lista de tarefas a correr indefinidamente de forma concorrente.

No código 5-5, no ciclo `while True` (infinito), o programa suspende a sua execução na expressão `await`. O módulo *asyncio* passará à execução da tarefa seguinte. Como no programa apenas foi adicionada na lista uma tarefa, o programa aguarda que haja um novo cliente (aguarda que seja estabelecida uma ligação *socket* pelo cliente MODBUS). Quando há um cliente ativo para comunicação, os dados recebidos serão avaliados, quer em quantidade de bytes, quer na verificação da estrutura de *bytes* para que esteja de acordo com o protocolo MODBUS TCP (Código 5-5).

```
# Analisar o pacote recebido por forma a verificar se corresponde a um pacote
ModBusTCP
async def modbus_handler(client):
    with client:
        while True:
            data = await loop.sock_recv(client, 10000)
            #print("Dados Recebidos",data)
            if len(data) == 12:
                ModbusRequest(data)
            #sendDadosDB(data)
            if not data:
                client.close()
                break
            print(client)
            await loop.sock_sendall(client,data)
    print("Connection closed")
```

Código 5-5 – Gestor dos pacotes MODBUS. Avaliação da quantidade de bytes recebidos

```
# Criar o socket server para receber ligações na tuple 'address'
async def modbus_Server(address):
    sock = socket(AF_INET, SOCK_STREAM)
    sock.setsockopt(SOL_SOCKET,SO_REUSEADDR,1)
    sock.bind(address)
    sock.listen(5)
    sock.setblocking(False)
    while True:
        client, addr = await loop.sock_accept(sock)
        print ("Connection with:", addr )
        loop.create_task(modbus_handler(client))
```

Código 5-6 – Tarefa para escutar ligações na porta 502 do computador (Raspberry Pi® 3)

A avaliação da quantidade de *bytes* recebidos na mensagem permite filtrar apenas as mensagens pretendidas, para que os dados corretos sejam passados à função de análise da sua estrutura (*ModbusRequest()*). Ao aumentar o número de *bytes* a função dará erro e não será executada corretamente.

Após a correta receção dos dados e o devido processamento, estes serão enviados como resposta ao cliente que abriu a sessão.

Para guardar dados da máquina, não há necessidade de processar as respostas para envio, nem enviar exceções para conhecimento que os dados não foram comunicados. Os dados que pretendemos recolher, são armazenados no autómato e enviados sempre que houver comunicação.

```
def ModbusRequest(data):

    try:
        #print(data)
        #print (len(data))
        dados = struct.unpack('!HHHBBHH', data)
        TransactionID = dados[0]
        Protocolo = dados[1]
        MensagemLengh = dados[2]
        UnitIdentifier = dados[3]
        MBAP = dados[0:4]
        #verificar se a mensagem é válida
        reply = MODBUS_PDU_Checking(MBAP)
```

Código 5-7 – Decomposição dos bytes de informação pelos respetivos nomes

Os dados recebidos são depois analisados e separados pelos seus elementos constituintes. No código 5-7 é ilustrada a forma como esta função foi implementada. A variável dados utiliza o módulo struct e a função unpack para obter os dados da trama de bytes (`struct.unpack(fmt, buffer)`) [38]. No primeiro argumento da função, é colocada uma string que indica que tipo de sequência de bytes se está a analisar (`'!HHHBBHH'`). Assim, o caratere `!` significa que a ordem de bytes corresponde a dados de uma rede, o H é um integer com 2 bytes e o B é um integer com 1 byte. A estrutura de dados corresponde ao que foi analisado no capítulo 4.4 *MODBUS* TCP.

O conjunto de bytes correspondentes ao MBAP, é enviado para a função `MODBUS_PDU_Checking()` (Código 5-8) que verifica se a segunda posição do vetor tem o valor 00. Se for validado, fica identificado tratar-se de uma comunicação por MODBUS.

```
def MODBUS_PDU_Checking (MBAP):
    #Verificar se o protocolo está corretamente identificado
    if MBAP[1] == 0:
        return 1
    else:
        return 0
```

Código 5-8 – Verificação do MBAP

Após a validação da mensagem recebida, obtém-se o número da função MODBUS e o endereço do registo. Trata-se da posição de memória, definida na consola, e que permite identificar que classe de dados estamos a receber. Na tabela 5-1 indica-se que dados são recebidos da máquina e o endereço que foi utilizado para receber cada tipo de dado. No exemplo, na memória do PLC na *word* D502 é registado o tempo em que a serra está ligada. Segundo informação da tabela, de cada vez que é pressionado o botão ligado à entrada 1.03 (*Start*), o valor do registo D502 é enviado para o servidor no endereço MODBUS 00.

Descrição	Trigger no PLC	Memória no PLC	Objeto na Consola	Endereço na Consola	Função MODBUS	Endereço MODBUS
Botão Start Pressionado	200.04	1.03	DT0	0X - 1	00	00
Número de Barras Carregadas	100.05	D610	DT1	4X - 7	06	06
Tempo de Serra Ligada	1.03	D502	DT2	4x - 1	06	00
Número de tubos descarregados	200.05	D614	DT3	4X -9	06	08
Tubos cortados Por Hora	200.05	D618	DT4	4X-11	06	10
Tubos carregados por Hora	200.05	D624	DT5	4X-13	06	12

Tabela 5-1 – Tabela de endereçamento para os dados recebidos da máquina

```
cur.execute ("INSERT INTO tblLog (Mensagem, MBAP, IDSlave, MBFunction,
PrimeiroEndereco, Dados) VALUES( '{:0>7}', '{:0>4}' ,'{:0>2}',
'{:0>2}','{:0>2}','{:0>10}');".format(msg,MBAP,Slave,Function,FirstRegi
ster,Data))

        db.commit()
```

Código 5-9 – Execução do comando MySQL para inserir dados na tabela tblLog

O servidor envia para a tabela *tblLog* a mensagem completa recebida bem como os dados discriminados (Código 5-9).

Caso haja erro na inserção de dados na tabela do servidor MySQL, apenas o arranque do servidor na consola SSH permitirá visualizar o tipo de erro que está a acorrer.

5.3 Gestão da Base de Dados Microsoft Access®

A aplicação em Access desenvolvida tem servido de base à organização do trabalho da Manutenção na Miralago. Após algumas alterações administrativas dentro da empresa, ficou a ser uma ferramenta essencial na organização do serviço. Têm sido introduzidas algumas funcionalidades, que surgem de necessidades dentro da organização, e certamente que outras serão desenvolvidas a partir daqui.

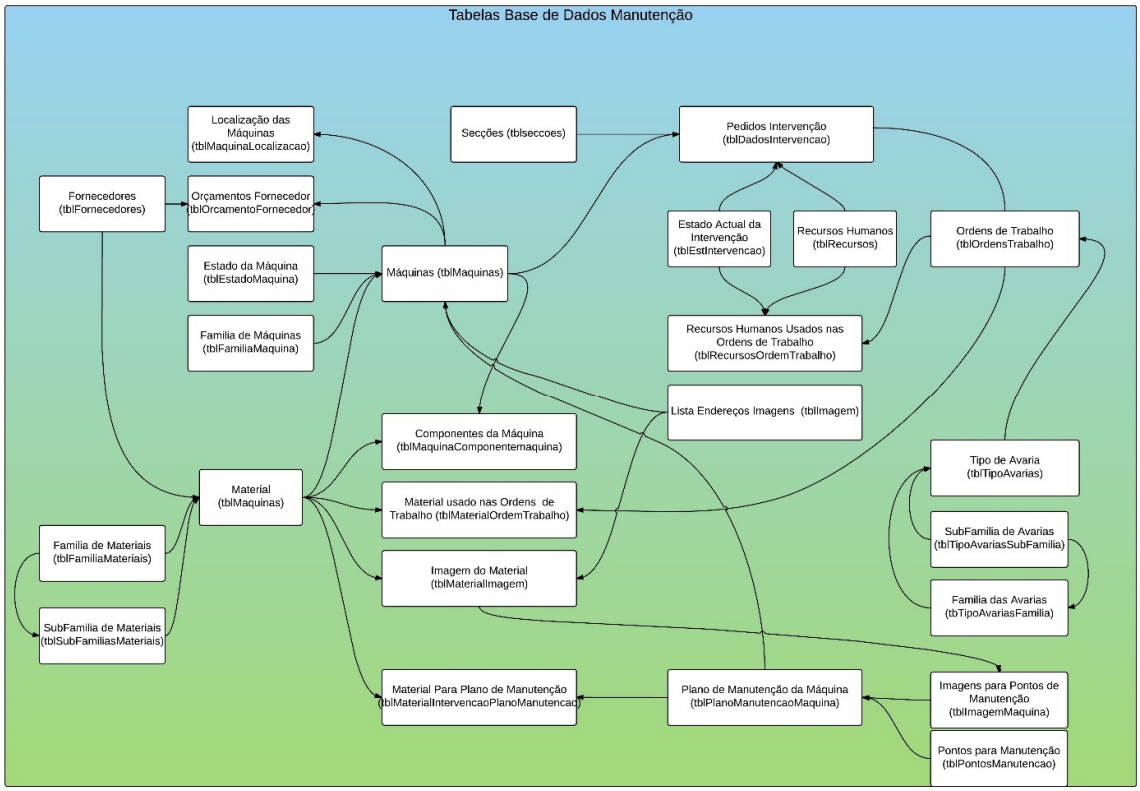


Figura 5-3 – Organização de tabelas da base de dados.

A organização das tabelas centra-se nas Máquinas, ramificando para os Pedidos de Intervenção (PI) que geram Ordens de Trabalho (OT). A figura 5-3 ilustra a forma como todas as tabelas estão relacionadas entre si. Quando as linhas apresentam uma seta considera-se que a relação é de 1 para infinito (seta) e quando não apresentam qualquer seta a relação é de 1 para 1.

Passando a explicar como foi desenvolvida a relação de tabelas, pode considerar-se que todas as máquinas pertencem a uma família (Prensas, Fresadoras, Tornos,…) e que todas têm um estado atual (Em Serviço, Fora de Serviço mas no Imobilizado ou Fora de Serviço e Fora do Imobilizado). Têm também, pelo menos, um Plano de Manutenção, um conjunto de imagens associadas e vários materiais que dela fazem parte (válvulas, motores, inversores, óleos,...). Por outro lado, elas passam por várias localizações dentro da fábrica, têm vários orçamentos associados, uma lista de componentes, que por sua vez, fazem parte da lista de materiais.

Cada secção (solda, prensas, montagem,…) pode fazer vários PIs e pode haver vários para uma só máquina. Cada PI passa por um estado atual (Em Agendamento, Aguarda Material,…) até ao momento em que é executado, e é solicitado por um Recurso Humano da empresa.

Para cada OT há um PI associado e é realizado por vários Recursos Humanos pertencentes, normalmente, ao Departamento de Manutenção. Com a execução da OT é necessário fazer a classificação do tipo de avaria identificada para posterior análise.

Na figura 5-4 é apresentada a relação entre os formulários que permitem ao utilizador da aplicação introduzir dados, atualizá-los e fazer análise de toda a informação introduzida nas tabelas.

Na secção 5.3.1 irá ser analisado em pormenor os formulários criados e alguns aspetos que foram tidos em consideração na sua elaboração.

Passa-se de seguida a fazer uma breve apresentação desses formulários tendo em atenção os pontos principais.

O primeiro formulário (Início), que surge ao executar o ficheiro da aplicação, enquadra e estabelece a ligação com todos os outros formulários direta ou indiretamente. Ao abrir o formulário, o utilizador pode visualizar a estatística associada às intervenções pendentes de execução e a sua distribuição pelos vários estados (Em Agendamento, Aguardar Material, …). Está dividido em várias áreas consideradas importantes durante o trabalho no Departamento.

Os Pedidos de Intervenção, onde é possível adicionar, preenchendo depois os dados necessários; pesquisar por número, texto ou máquina, por forma a obter uma lista de casos que se poderão enquadrar na pesquisa e, a partir daí, selecionar o pedido que preencha os requisitos.

Com os pedidos de intervenção criados pode-se optar por criar a OT associada. Mais uma vez no formulário "Início" pode-se pesquisar por número de OT e aí introduzir os dados fornecidos como material utilizado, tipo de avaria e tempo de execução. Existe ainda a possibilidade de imprimir as OTs limpas e alguma análise de ocupação dos recursos humanos intervenientes nas OTs.

Na área Material é possível adicionar um determinado artigo que se pretende comprar, ou fornecedor de onde se pretende um orçamento ou ainda enviar um e-mail à logística para proceder à encomenda, e-mail este que pode ser consultado mais tarde noutro formulário e reenviado. Em cada um dos artigos de material pode-se adicionar uma imagem, o orçamento para futura consulta ou a respetiva informação técnica. O formulário para pesquisa de artigos, as listas de material utilizado por mês bem como o inventário por armazém são também acessíveis na área relativa ao Material.

Como já dito anteriormente, a lista de máquinas é a base desta aplicação. A ela está dedicado um formulário, acedido na área Máquinas, que apresenta várias informações. Caso haja intervenções previamente executadas e, portanto, com a respetiva OT terminada, surgem alguns indicadores como o *Mean Time Between Failures* (MTBF), *Mean Time To Repair* (MTTR) e *Mean Waiting Time* (MWT) da máquina. Como é evidente existe a possibilidade de após a pesquisa de máquina por número, enviar e-mail para abrir número de máquina pelo departamento financeiro, enviar e-mail para solicitar guia de transporte para reparação de máquina, adicionar uma máquina ou o respetivo Plano de Manutenção.

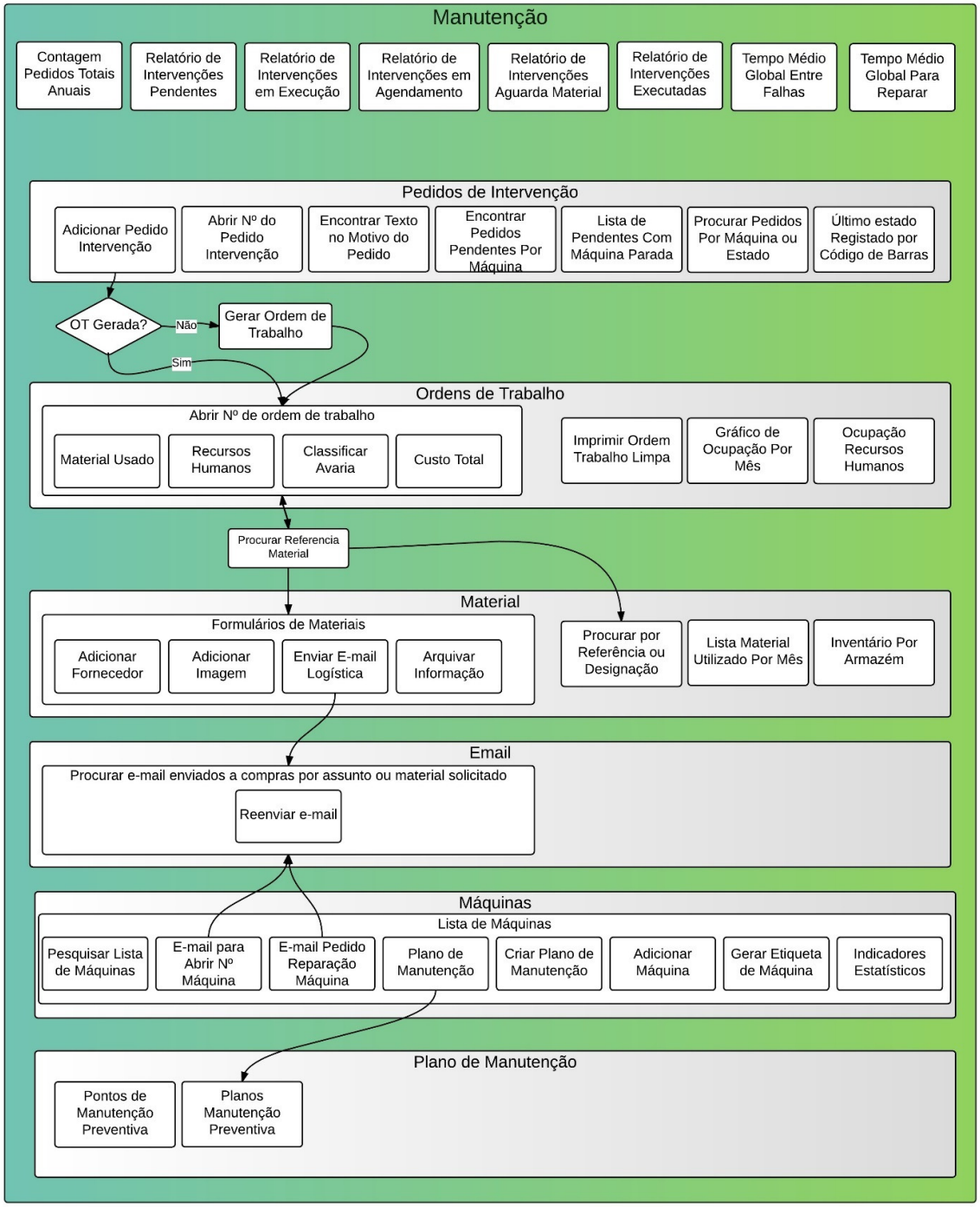


Figura 5-4 – Organização e distribuição dos formulários

A área relativa ao Plano de Manutenção Preventiva compreende dois formulários. A criação de pontos de manutenção genéricos, ou seja, que podem ser aplicados a várias máquinas (*p. ex.,* Verificar Nível de Óleo) e a criação do plano de manutenção para uma determinada máquina (pode ser acedido diretamente no formulário da máquina). Neste plano, é adicionada a informação pertinente para uma correta elaboração do plano de manutenção como, a periodicidade da execução, uma imagem alusiva à ação ou local da ação e o material a utilizar caso seja necessário. Também foi adicionado um campo para colocar o tempo planeado para a ação por forma a verificar desvios.

5.3.1 Organização dos Formulários

Quando se abre o ficheiro Access da aplicação é apresentado o panorama geral do estado das intervenções por ano (Figura 5-5).

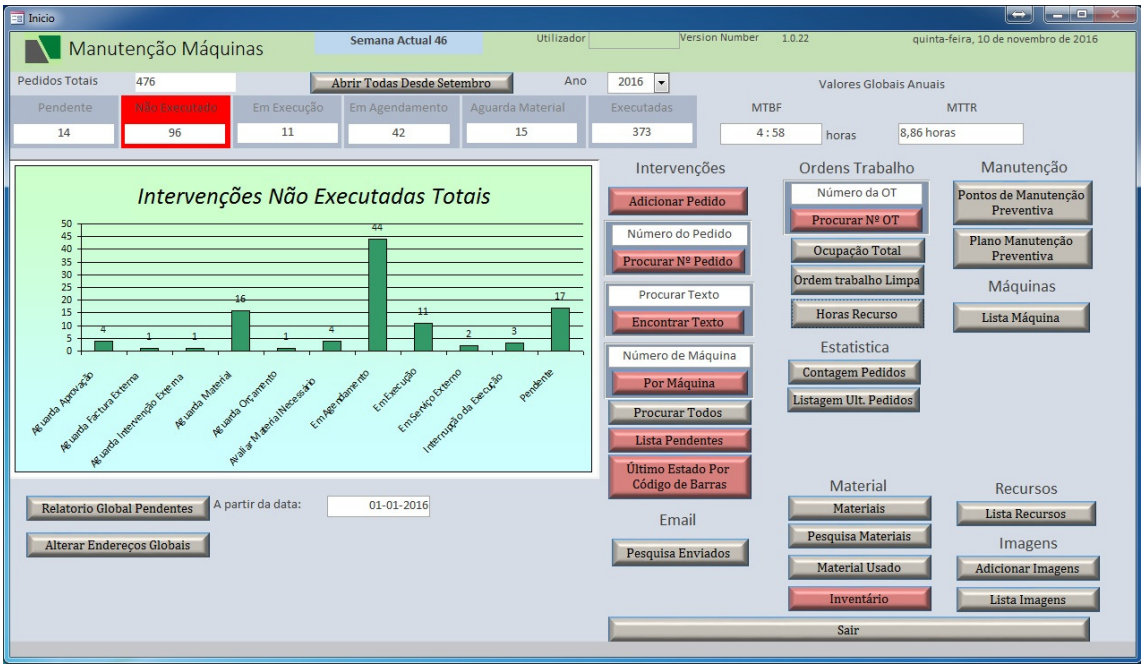


Figura 5-5 – *Dashboard* do Departamento de Manutenção

O gráfico visualizado na figura 5-5 apresenta a distribuição das intervenções solicitadas ao sistema e que carecem de execução. Trata-se de todas as intervenções e não apenas as relativas ao ano selecionado na parte superior do formulário. Pode verificar-se, que a maioria das solicitações estão distribuídas por três áreas. Ou carecem de análise/aprovação (pendentes), ou estão em agendamento ou aguardam material.

Para criar o gráfico, é necessário adicionar o objeto "*chart* " da lista de objetos predefinidos do Access. Os dados devem ser apresentados de forma a que a informação fique clara e objetiva. Para tal, realizam-se as consultas (*querys*) à base de dados por forma a concentrar os dados no número de campos necessários à construção do gráfico. No caso apresentado, foi feita a consulta presente no código 5-10.

```
SELECT Count(tblDadosIntervencao.ID) AS Totais, tblDadosIntervencao.EstadoActual
FROM tblDadosIntervencao
GROUP BY tblDadosIntervencao.EstadoActual
HAVING (((tblDadosIntervencao.EstadoActual)<>"executado"));
```

Código 5-10 – Consulta para dados do gráfico de distribuição de intervenções não executadas

Ao abrir a aplicação o utilizador deve ficar com uma ideia do panorama geral das intervenções pendentes no Departamento de Manutenção.

Para além do gráfico há ainda, neste formulário, um conjunto de caixas e botões que permitem aceder às principais funções implementadas.

Os relatórios das intervenções a realizar, onde é possível visualizar uma listagem de todas as intervenções por estado atual (Figura 5-6) surgem nas caixas da parte superior do formulário.

Figura 5-6 – Caixas de quantidade de intervenções não terminadas.

Ao clicar, por exemplo, na caixa "Aguardar Material" surge uma listagem, organizada por número de máquina (Figura 5-7), e que apresenta todas as intervenções cuja conclusão depende de material que ainda não existe em armazém.

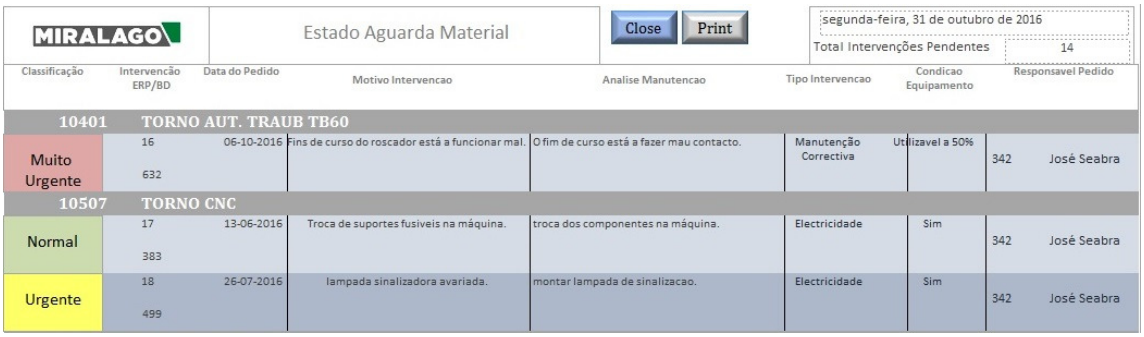


Figura 5-7 – Relatório de intervenções que aguardam material

Para obter a informação da base de dados foi utilizada a linguagem VBA incluída na aplicação Office Access. Ao clicar no identificador da caixa "Aguarda Material" é desencadeado um evento "On Click" presente no código 5-11. A sub-rotina corre uma função chamada `ContagemEstados`, que recebe como primeiro argumento o estado que se pretende consultar e como segundo argumento a totalidade das máquinas (`"all"`) ou apenas uma máquina como base para a consulta.

```
Private Sub lblAguardaMaterial_Click()
    ContagemEstados "Aguarda Material", "all"
End Sub
```

Código 5-11 – Evento ao clicar no identificador "Aguarda Material" no *dashboard*

Dentro da função, uma primeira parte verifica o ano que se pretende pesquisar (Código 5-12). Para tal utilizou-se duas funções Access VBA `DateSerial` e `CDate`. A primeira função permite obter o valor da data com base em três inteiros e a segunda formata o inteiro obtido numa data com o formato dd/mm/yyyy (dia – mês- ano). Na linguagem SQL é necessário utilizar a formatação de data utilizada nos países anglo-saxónicos (mm-dd-yyyy) daí que foi utilizada a função `Format` em que o segundo argumento permite formatar a data de acordo com o pretendido.

```
1   If strMaquina = "all" Then

2   'Especificar a data do pedido
3   strDataPedido = CDate(DateSerial(CInt(Me.cboAno), Month(Date), Day(Date)))

4   'Selecção do primeiro e ultimo dia do ano em análise
5   FirstDayYear = DateSerial(Year(strDataPedido), 1, 1)
6   LastDayYear = DateSerial(Year(strDataPedido), 12, 31)

7   'Estabelecimento do intervalo de tempo para realizar a consulta
8   strMinDate = Format(CDate(FirstDayYear), "mm-dd-yyyy")
9   strMaxDate = Format(CDate(LastDayYear),"mm-dd-yyyy")

10  strTask = "AND ((tblDadosIntervencao.DataPedido) > #" & strMinDate & "#) AND
    ((tblDadosIntervencao.DataPedido) < #" & strMaxDate & "#) AND
    ((tblDadosIntervencao.AnularApagar) = False))"
```

Código 5-12 –Gerar data do pedido e datas para o ano em análise

A informação que irá ser colocada no relatório tem por base o Código 5-13 que seleciona alguns campos das tabelas *tblDadosIntervencao*, *tblMaquinas* e *tblRecursos* e verifica onde ocorre o estado *"Aguarda Material"* entre as datas *1-1-2016* e *31-12-2016*.

```
SELECT tblDadosIntervencao.Maquina, tblMaquinas.Designacao,
tblDadosIntervencao.IntervencaoSistema, tblDadosIntervencao.DataPedido AS Data,
tblDadosIntervencao.MotivoIntervencao, tblDadosIntervencao.AnaliseManutencao,
tblDadosIntervencao.TipoIntervencao, tblDadosIntervencao.CondicaoEquipamento,
tblDadosIntervencao.AnularApagar, tblRecursos.IDPHC AS Responsavel,
tblRecursos.Nome, tblDadosIntervencao.Urgencia,
tblDadosIntervencao.Classificacao, tblDadosIntervencao.ID
FROM tblRecursos INNER JOIN (tblMaquinas INNER JOIN tblDadosIntervencao ON
tblMaquinas.NumeroMaquina = tblDadosIntervencao.Maquina) ON tblRecursos.ID =
tblDadosIntervencao.ResponsavelPedido
WHERE (((tblDadosIntervencao.EstadoActual) = 'Aguarda Material') AND
((tblDadosIntervencao.DataPedido) > #01-01-2016#) AND
((tblDadosIntervencao.DataPedido) < #12-31-2016#) AND
((tblDadosIntervencao.AnularApagar) = False))
ORDER BY tblDadosIntervencao.Maquina;
```

Código 5-13 – Código SQL para selecionar dados para relatório

Com os dados recolhidos é necessário organizá-los no relatório. O procedimento para criar o relatório começa no menu *Create* no campo *Reports* onde é utilizado *Blank Report*. São então adicionadas as caixas de texto que irão corresponder aos campos da consulta.

Para que a informação fique organizada por máquina é possível agrupar os campos (Figura 5-8). Como se pode observar na figura 5-7 do relatório, as máquinas foram utilizadas para que as intervenções pudessem ficar agrupadas.

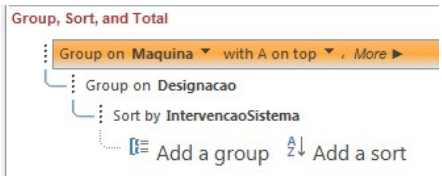


Figura 5-8 – Informação agrupada no relatório de Manutenção

A grande maioria das ações desenvolvidas pela manutenção, começam no "Pedido de Intervenção". O impresso apresentado no anexo v – impresso de registo das intervenções da manutenção, foi totalmente produzido no Access.

A lista da figura 5-9 apresenta o conjunto de operações, relacionadas com os Pedidos de Intervenção, que é possível executar a partir do *dashboard*.

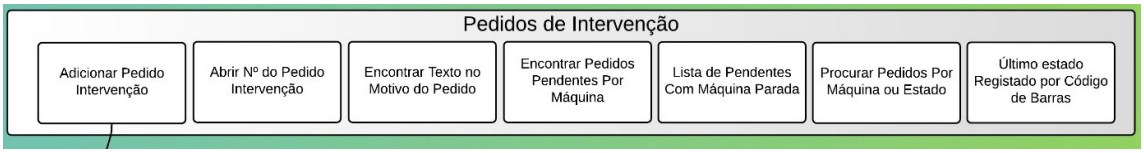


Figura 5-9 – Lista de Formulários Relacionados com Pedidos de Intervenção

Ao clicar no botão 

, surge o formulário apresentado na figura 5-10, com o número "ID" da intervenção e uma data pré-preenchida. Este formulário tem ligação com a tabela `tblPedidosIntervencao`.

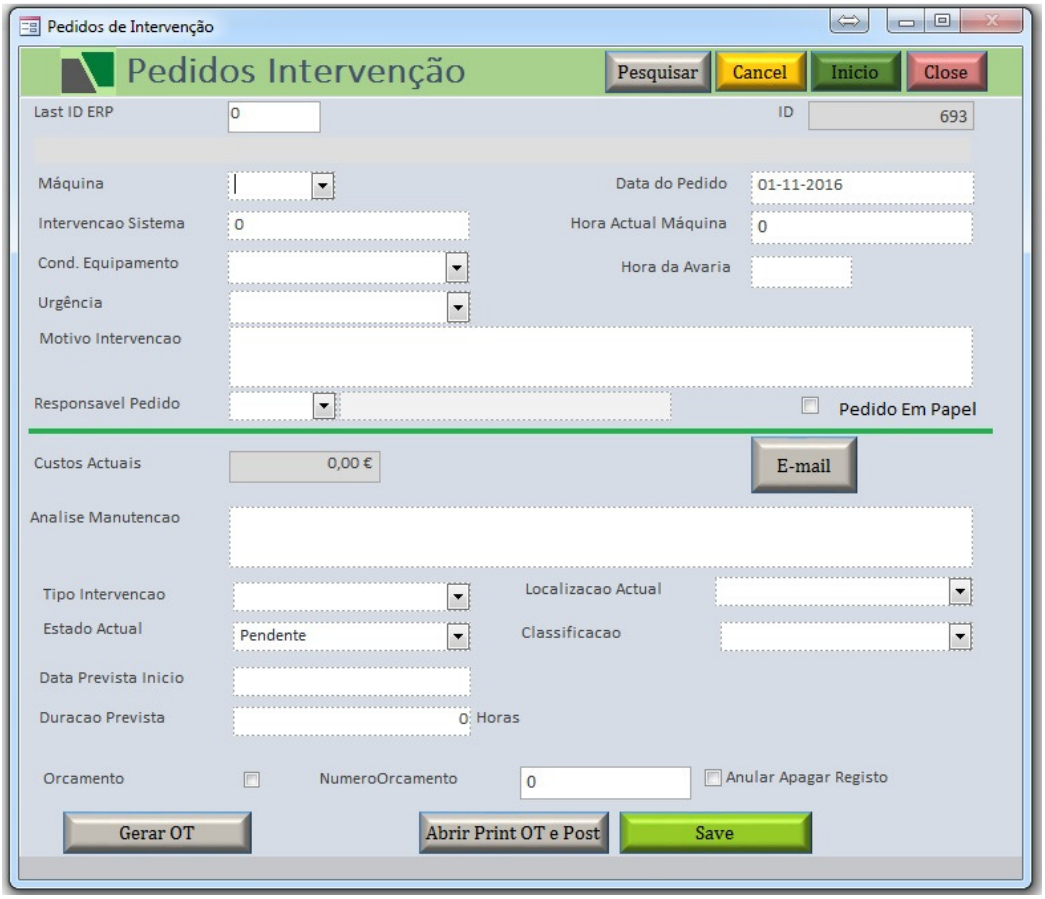


Figura 5-10 – Formulário "Pedido de Intervenção"

No início, a base de dados foi pensada no sentido de completar lacunas importantes no *software* de gestão de manutenção existente na empresa. A meio do ano de 2016, o sistema informático da empresa foi descontinuado e deixou de fazer sentido introduzir mais dados. O risco de que os dados, lá introduzidos, não pudessem ser acedidos passou a ser real e foi necessário fazer algumas adaptações nesta base de dados.

O valor colocado no campo "Intervenção Sistema" (Figura 5-10) é o número atribuído pelo sistema informático da empresa à intervenção aberta. Esse número era colocado neste campo e validado, juntamente com o número de máquina, no módulo BaseDeDados por forma a verificar se a relação entre o número da intervenção e a máquina já estavam registados. Além de, na organização da base de dados, este conjunto ser chave primária, procurou-se comunicar com o utilizador que a intervenção já estava registada. A função pública `EncontrarRegistoIntervencao`, dentro do

módulo BaseDeDados (Anexo VI – VBA de verificação de número de intervenção) executa a consulta SQL com o Código 5-14.

```
SELECT tblDadosIntervencao.Maquina, tblDadosIntervencao.IntervencaoSistema
FROM tblDadosIntervencao
WHERE(((tblDadosIntervencao.Maquina)='10101')
AND((tblDadosIntervencao.IntervencaoSistema)=23));
```

Código 5-14 – Linguagem SQL para recolha de dados para máquina e intervenção do sistema da empresa

A quantidade de resultados é utilizada como forma de verificar se a relação já existe na tabela *tblDadosIntervencao*. Se o resultado da consulta for superior a 0 (zero) isso indica que a relação já existe e a função retorna 0, caso seja encontrado algum registo a função retorna 1 e uma mensagem surge informando que a informação da relação máquina-registo já foi criada (Código 5-15).

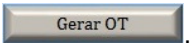
```
Private Sub Maquina_BeforeUpdate(Cancel As Integer)
Dim bolValidar As Boolean
Dim strState As String

'Validar se já existe o registo que se está a tentar introduzir
bolValidar = BaseDeDados.EncontrarRegistoIntervencao(Me.Maquina,
Me.IntervencaoSistema)

'Verificar ultima intervenção no sistema informático da Miralago
strState = BaseDeDados.EncontrarLastERPNumber(Me.Maquina)

Me.txtLastERP = strState

'Caso já exista o registo no sistema informa e anula alteração
If bolValidar = True Then
    a. MsgBox "Relação máquina número de registo já criado"
    b. MsgBox "Irá ser incrementado o registo anterior"
    c. Me.IntervencaoSistema.Value = strState + 1
    d. Cancel = True
End If
End Sub
```

Código 5-15 – Evento BeforeUpdate para o campo máquina

Com a descontinuação do sistema informático da Miralago, tornou-se necessário criar uma função que gerasse automaticamente os números que o sistema da empresa gerava anteriormente. Dentro do módulo *BaseDeDados* criou-se a função pública *EncontrarLastERPNumber* que, com base na consulta com o código 5-16, procura o número da última intervenção registada.

```
SELECT tblDadosIntervencao.Maquina,
Max(tblDadosIntervencao.IntervencaoSistema) AS MaxOfIntervencaoSistema
FROM tblDadosIntervencao
GROUP BY tblDadosIntervencao.Maquina
HAVING (((tblDadosIntervencao.Maquina) ='10101'))
ORDER BY tblDadosIntervencao.Maquina,
Last(tblDadosIntervencao.IntervencaoSistema) DESC;
```

Código 5-16 – Consulta SQL do número mais pequeno da intervenção máquina "10101"

O valor obtido é utilizado no evento *Maquina_BeforeUpdate* (Código 5-15) que o coloca no campo "Last ID ERP" (Figura 5-10). Caso se verifique que a relação máquina-registo já existe, soma um (1) ao valor encontrado, e coloca-o no campo "Intervenção Sistema", referido anteriormente, informando o utilizador de tal ocorrência. Depois de preencher os campos acima da linha verde (Figura 5-11) é possível gerar a respetiva Ordem de Trabalho clicando no botão 

. O evento *On*

Click, deste botão, irá executar o código (Anexo VII – Adição de Novas Ordens de Trabalho) necessário para que seja criada uma nova entrada na tabela de Ordens de Trabalho (tblOrdensTrabalho).

Há várias formas, em Access, de introduzir dados nas tabelas. Adicionar manualmente; usando macros; ou VBA. Neste caso foi utilizada uma aproximação em VBA que utiliza um objeto tipo `Database` e corre o método `Execute` com a linguagem SQL presente no Código 5-17.

```
INSERT INTO tblOrdensTrabalho (ID_Intervencao )
SELECT tblDadosIntervencao.ID
FROM tblDadosIntervencao LEFT JOIN tblOrdensTrabalho ON
tblDadosIntervencao.ID = tblOrdensTrabalho.ID_Intervencao
WHERE (((tblDadosIntervencao.ID)= 633));
```

Código 5-17 - SQL para inserir dados nas tabelas

O fluxo de informação entre quem solicita uma intervenção e quem tem a responsabilidade de a avaliar e dar resposta foi tido em conta nesta aplicação. Trata-se de uma relação entre o cliente (solicitador) e o fornecedor (Departamento de Manutenção).

No preenchimento do formulário "Pedidos Intervenção", na atualização do "Estado Atual" (Figura 5-11), é desencadeado um evento que irá registar a alteração do estado na base de dados e permite ao utilizador enviar um e-mail, informando quem solicitou a intervenção, do estado da mesma.

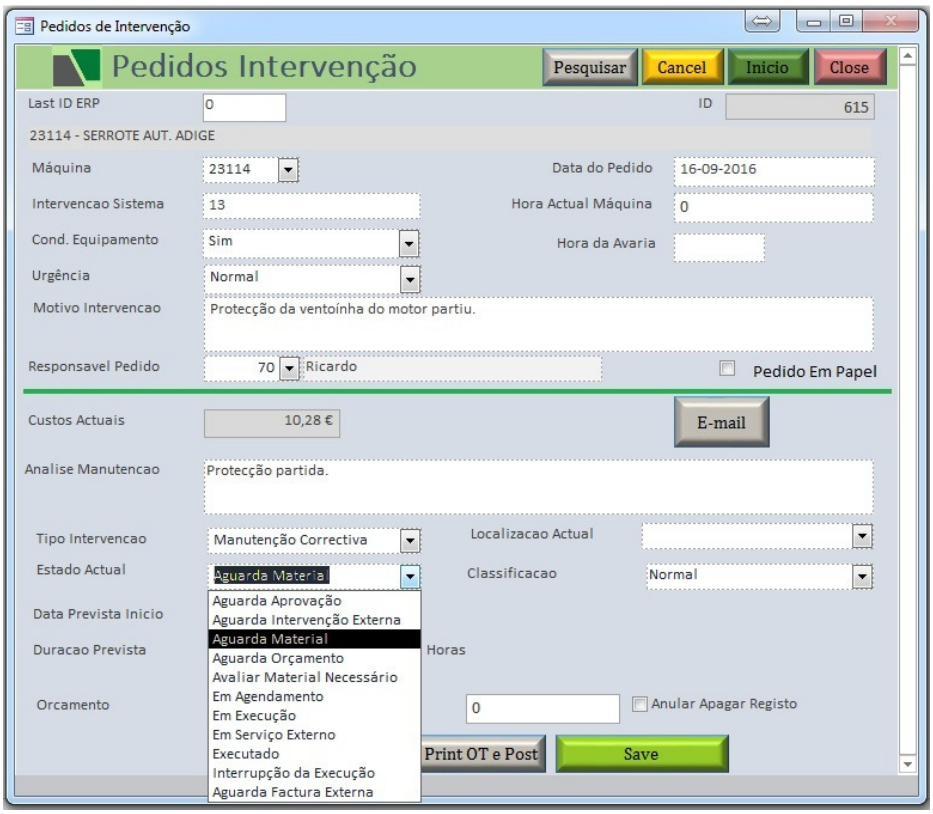


Figura 5-11 – Formulário de pedido de intervenção parcialmente preenchida.

Com o registo do tempo em que foi registada a alteração, do estado da intervenção, no sistema será possível criar um indicador de desvio entre o registo das alterações e os dados presentes nas Ordens de Trabalho registados pelos Técnicos de Manutenção.

O objetivo do módulo `Logging` é introduzir funções de recolha de dados das ações pertinentes executadas nos formulários. Quando o utilizador altera o estado, a função `WriteAlteracaoEstado`

do módulo *Logging* é executada (Código 5-18) com os seguintes argumentos: ID da intervenção, estado atual (alterado), número da intervenção no sistema informático da empresa, máquina. Como se pode verificar, o código utilizado permite adicionar dados a uma tabela, com VBA (diferente de *DataBase.Execute* usado anteriormente). A utilização do objecto *Recordset* facilita a execução da linguagem SQL que é utilizada para adicionar dados às tabelas. No entanto, torna a aplicação mais lenta e, por vezes pode dar problemas quando estamos a falar de tabelas interligadas.

```
1  Public Sub WriteAlteracaoEstado(IDIntervencao As Long, EstadoActual As
   String, IDIntervencaoSistema As Long, _
   NumeroMaquina As String)

2  Dim db As Database
3  Dim rs As Recordset

4  Set db = CurrentDb
5  Set rs = db.OpenRecordset("tblRegistoAlteracaoestado", dbOpenDynaset)

6  rs.AddNew
7  rs("IDIntervencao") = IDIntervencao
8  rs("EstadoActual") = EstadoActual
9  rs("IDIntervencaoSistema") = IDIntervencaoSistema
10 rs("NumeroMaquina") = NumeroMaquina
11 rs.Update

12 rs.Close
13 Set rs = Nothing
14 db.Close

15 End Sub
```

Código 5-18 – Função do módulo *Logging* para registo na tabela tblRegistoAlteracaoestado

De qualquer forma não seria difícil passar esta função para a linguagem SQL o que tornaria a execução mais rápida. Em muitos casos, que irão ser abordados noutras secções, essa foi a opção. Este aparente zig-zag no desenvolvimento da aplicação deriva do facto de o processo de criação ser acompanhado de um processo de aprendizagem da utilização do próprio Access.

```
strAssunto = "Estado da Intervencao " & Me.IntervencaoSistema & " na máquina
" & Me.Maquina
    strCorpoEmail = "A Intervenção " & Me.IntervencaoSistema & " da máquina
" & Chr(34) & Me.txtDesignacao & Chr(34) & vbCrLf & _
    "Passou a " & Chr(34) & Me.EstadoActual & Chr(34)

(…)

strResposta = MsgBox("Quer enviar email ao responsável pelo pedido?",
vbYesNo)

    If strResposta = 6 Then
        Logging.emailEstado strAssunto, strCorpoEmail, Me.ResponsavelPedido
    ElseIf strResposta = 7 Then
        MsgBox "Envio de mensagem cancelado"
    End If
```

Código 5-19 – Opção de enviar e-mail ao Responsável pelo Pedido

Para o método *OpenRecordset*, é necessário fornecer o primeiro argumento, nome da tabela de onde se pretendem obter os registos ("*tblRegistoAlteracaoestado*"). Todos os restantes argumentos (*Type, Options, LockEdit*) são opcionais. Como se pretende, num futuro próximo, que haja a separação das tabelas dos formulários, criando um *front* e um *backoffice*, é necessário utilizar o tipo *Dynaset* que cria um conjunto de registos locais para cada utilizador. Este aspecto terá que ser tido em conta mais tarde. A adição de registos com o *Dynaset* não atualiza os dados para outros

utilizadores, num primeiro momento. Outro utilizador que aceda aos dados, antes de eles terem sido enviados para as tabelas, não terá acesso aos dados atualizados. O método `AddNew`, cria e adiciona um registo e o método `Update` grava a adição de dados ao registo. Com a execução da ação, é recomendado fechar o *Recorset*, limpá-lo de memória (`rs=Nothing`) e fechar a base de dados (`db.Close`).

```
1  Public Sub emailEstado(strSubject As String, strBody As String,
   strResponsavel As String)

2  'Inicializar objectos para o envio de email
3  Dim oApp As Object, oEmail As Object

4  'Criar objectos respectivos para a aplicação e o email
5  Set oApp = CreateObject("Outlook.Application")
6  Set oEmail = oApp.CreateItem(0)

7  strEmail = Nz(DLookup("[Email]", "tblRecursos", "ID = " & strResponsavel),
   "")
8  If strEmail = "" Then
      a. MsgBox " Não há registo de email para o responsável pelo pedido"
      b. Exit Sub
9  End If

10 With oEmail
      a. .To = strEmail
      b. .Subject = strSubject
      c. .Body = strBody

      d. If Not IsNull(.To) And Not IsNull(.Subject) And Not IsNull(.Body)
         Then
            i. .Send
           ii. MsgBox "Email Enviado"
          iii. Logging.WriteLog "Email sent - To: " & strEmail _
                        a. & " Subject: " & strSubject _
                        b. & " Body: " & strBody
      e. Else
            i. MsgBox " Por favor preencha todos os campos necessários"
      f. End If
11 End With
12 End Sub
```

Código 5-20 – Função para envio de e-mail de alteração de estado da intervenção

Com o código 5-19, se a resposta à questão for afirmativa, a função `emailEstado` (Código 5-20) é executada com os argumentos: assunto; um corpo de texto; e o número do responsável.

Para o envio de e-mail utilizando o Access, existem algumas hipóteses. A hipótese mais simples será utilizar o cliente de e-mail do computador onde a aplicação Access está a ser executada. Para tal, pode utilizar-se o método *DoCmd.SendObject com os argumentos To, Subject, MessageText e EditMessage* (se pretendemos abrir a mensagem no cliente para edição antes do envio). Pode-se também utilizar a biblioteca do *Outlook* e os seus objetos no VBA. Para aceder a esta biblioteca é necessário aceder ao Menu "*Tools*" –> "*References*" e escolher a biblioteca "*Microsoft Outlook 16.0 Object Library*" (Figura 5-12). No caso do código 5-20 foi utilizada uma alternativa à adição de referências para o compilador correr no início e alocar memória. Foi criado um objeto com o nome `Outlook.Application` e o objeto necessário para o envio do E-mail (`oEmail`) e que, a partir deste ponto, funciona como se fosse carregada a respetiva biblioteca. A vantagem será que este objeto funcionará com qualquer versão do *Outlook* instalado na máquina que está a executar o programa.

Após ter-se criado o objeto para a aplicação, é necessário criar outro para utilizar dentro da aplicação. Com `oEmail = oApp.CreateItem(0)` cria-se um item de email dentro da aplicação. O `0` significa *olMailItem,* segundo a tabela presente na página do *Office – Development Center* [39].

Também utilizado neste código, a função *DLookup*, que surge também noutros pontos do código e em caixas de texto nos formulários, que permite recolher dados de um campo particular das tabelas. Esta função apresenta três argumentos (*expr, domain, criteria*) [40]: o argumento "expr" é a expressão que identifica o campo (nome do campo em parenteses rectos []) que se pretende consultar; *domain* onde é referido o conjunto de registos de onde se pretende consultar, pode ser uma tabela ou um *query*; o terceiro argumento define o critério a utilizar na consulta. Será equivalente ao WHERE numa declaração SQL.

`strEmail` é uma *string* de dados. Se o resultado de *Dlookup* for *Null*, ou seja, caso o responsável pelo pedido não tenha o seu e-mail registado na base de dados, causa um problema (*Null* não é uma *string* " "). Para resolver este problema utilizou-se a função `Nz` que executa exatamente o que se pretende. Cria uma *string*, quando o resultado do primeiro argumento é *Null.*

A partir deste ponto o envio de e-mail depende das propriedades *.To* (para quem); *.Subject* (assunto); *.Body* (corpo do email); *.Send* (enviar sem editar) ou .Display (abrir janela Outlook com e-mail e ser enviado, para editar)

Uma outra abordagem, para o envio de e-mail, que foi utilizada no envio para a logística, é utilizar a biblioteca "*Microsoft CDO for Windows 2000 Library*". CDO significa *Collaboration Data Objects* e dela fazem parte todos os objetos necessários para enviar um e-mail sem utilizar o cliente pré-definido. A principal vantagem é mesmo essa e a quase certeza que a aplicação, ao correr em ambiente Windows, terá instalada esta biblioteca pois remontam ao Windows NT. Sobre este envio mais pormenores serão dados a abordar o formulário `frmMaterial`. (Anexo XX – Envio de e-mails para Logística)

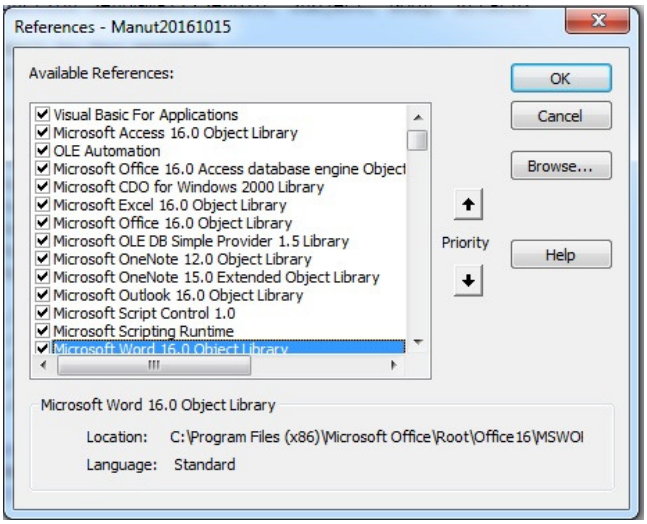


Figura 5-12 – Escolha de bibliotecas de outras aplicações.

```
1  Private Sub Form_AfterUpdate()
2  Logging.WriteLog ("Intervenção " & Me.ID & " Maquina " & Me.Maquina &
   "Estado Actual " & Me.EstadoActual)
3  End Sub
```

Código 5-21 – Guardar dados após sair do formulário

Após a introdução dos dados disponíveis no formulário "Pedido Intervenção", o utilizador pode fechá-lo. Esta ação desencadeia, entre outros eventos, a execução do evento "*AfterUpdate*" do formulário (Código 5-21).

Este evento executa a função `WriteLog` do módulo `Logging` que irá permitir guardar, em ficheiro texto, dados relativos à intervenção com informação da data em que foi executado.

Para que o Access possa interagir com o sistema de ficheiros, é necessário usar o objeto com essas características. Foi utilizado o objeto do tipo *FileSystemObject* `FSO` que pertence à biblioteca *Microsoft Scripting Runtime* (Figura 5-12). A adição desta biblioteca segue o mesmo procedimento das referidas anteriormente.

É utilizado o objecto `CurrentProject` cuja propriedade `Path` indica o endereço, no sistema de ficheiros, da diretoria onde o ficheiro Access está a ser executado. Com a informação fornecida por este objeto, é criada uma *string* com o endereço onde se pretende que seja gravado o ficheiro *log*. Caso este endereço ainda não exista, é utilizado o método `CreateFolder` do `FSO` para o criar.

O nome do ficheiro contém dados das funções `Year`, `Month` e `Day` cujo argumento `Date` contempla a data do sistema.

```
1  Dim FSO As New FileSystemObject
2
3  strPath = CurrentProject.Path & "\LogFile"

4  If Not Dir(strPath, vbDirectory) <> vbNullString Then
    a. FSO.CreateFolder (strPath)
5  End If

6  fileName = strPath & "\LogFile" & Year(Date) & Month(Date) & Day(Date) &
   ".txt"
```

Código 5-22 – Criação de diretoria e ficheiro texto para guardar dados do pedido de intervenção

Com a definição do nome, é necessário utilizar o objeto para manipular o ficheiro com extensão txt. O objeto tipo `TextStream` (`txtStream`), que juntamento com `FSO` e o método `OpenTextFile` permite a criação de um ficheiro de texto sequencial o que é ótimo na criação de um *logfile.*

Com o método `OpenTextFile`: é aberto o ficheiro `fileName`; informado o objetivo para aceder ao ficheiro `ForAppending`; caso o ficheiro não exista deve ser criado; e o formato do ficheiro é o pré-definido pelo sistema `TristateUseDefault` (Código 5-23).

```
1 Public Sub WriteLog(message As String)

2 Dim txtStream As TextStream

3 Set txtStream = FSO.OpenTextFile(fileName, ForAppending, True,
  TristateUseDefault)

4 message = Now() & ": " & message

5 txtStream.WriteLine message
6 txtStream.Close

7 Set txtStream = Nothing
```

Código 5-23 – Criação da mensagem a colocar no ficheiro Log

Ao fechar o formulário de “Pedidos Intervenção” é novamente reposto o formulário inicial.

Para além de informar, os Responsáveis pelos Pedidos, sobre a resposta do departamento de manutenção, é necessário que o departamento promova o preenchimento das respetivas Ordens de Trabalho na base de dados.

A figura 5-13 ilustra o acesso, a partir do formulário inicial, aos dados das OTs e a alguns indicadores dos trabalhos já executados ou em execução

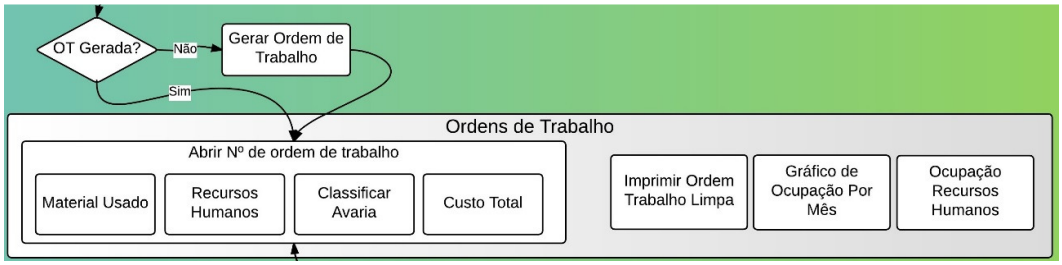
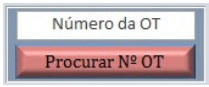


Figura 5-13 – Ordens de Trabalho

Ao colocar um número de OT na caixa 

, inicia-se o processo de verificação se o número da OT existe, usando a função *FindOneOT* no módulo *BaseDeDados*.

O código 5-24 segue a mesma aproximação do utilizado no

anexo vi – vba de verificação de número de intervenção.

```
1   strTask = "SELECT * FROM tblOrdensTrabalho WHERE (ID_OT=" & strFind & ")"

    a.  Set db = CurrentDb
    b.  Set rs = db.OpenRecordset(strTask)

    c.  With rs
    d.  i = 0
    e.  Do Until .EOF
          i.  i = i + 1
         ii.  .MoveNext
    f.  Loop
```

Código 5-24 – Verificação da existência de determinado registo de número de OT

É criada a expressão em SQL que procura por determinado número de OT na tabela *tblOrdensTrabalho*. Depois é utilizado um ciclo *Do Until*, que executa a contagem de todos os registos encontrados que correspondem ao critério. Caso nenhum seja encontrado, *i* é zero, se for mais que zero é enviada informação ao evento *On Click,* do botão "Procurar Nº OT" (Anexo X – Procura de Ordens de Trabalho), que existe um registo válido (*FindOneOT = "OK"*).

```
1 Case "OK"

2 DoCmd.Minimize
3 DoCmd.OpenForm "frmOrdensTrabalho", , , "[ID_OT]=" & strNumeroOT, , , "inicio"
4 DoCmd.MoveSize 300, 300

5 End Select
```

Código 5-25 – Abrir formulário da Ordem de Trabalho pesquisada

Com a informação de que o número existe, é necessário que o formulário a abrir seja o correto.

Primeiro o formulário inicial é minimizado (*DoCmd.Minimize*) depois é executado o comando com o método para abrir o formulário (*DoCmd.OpenForm*). Este método foi utilizado para abrir todos os formulários desta aplicação Access. Aproveita-se para aprofundar os argumentos que foram utilizados neste trabalho: o primeiro argumento do método contém uma *string* com o nome do formulário que se encontra presente na lista de formulários da aplicação; no segundo argumento pode-se definir como abrir o formulário, se para edição, datasheet,… (*acNormal*); no terceiro argumento pode-se utilizar uma *string* com o nome do *query,* definido na lista de *querys* da aplicação; o quarto argumento é utilizado como a expressão WHERE de SQL mas sem escrever WHERE (*"[ID_OT]=" &*

`strNumeroOT`); no quinto pode-se definir o nível de acesso ao formulário a abrir (adicionar, editar, alterar propriedades, ler)(`acFormPropertySetting`); no sexto define-se o tipo de janela como pretendemos abrir o formulário (`acWindowNormal`); e no sétimo argumento, pode-se utilizar uma *string* com dados para ser utilizado pelo formulário que entretanto se vai abrir.

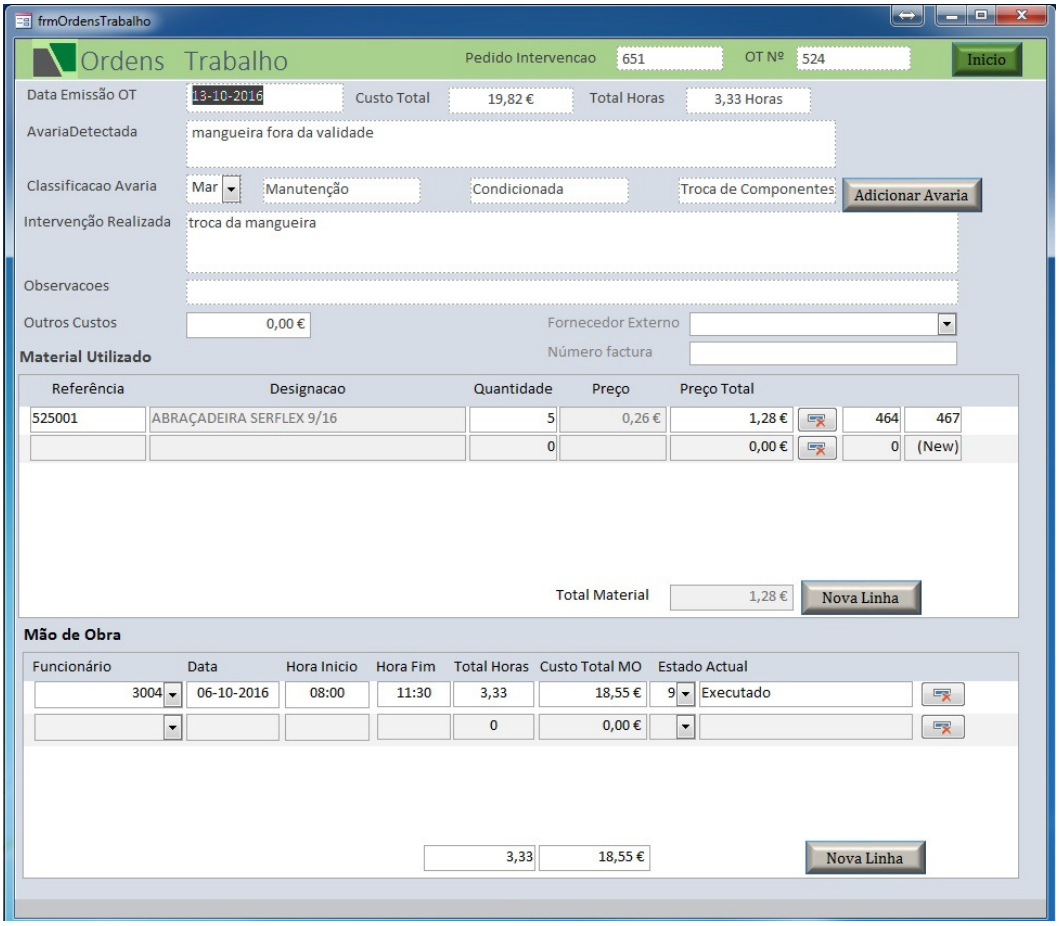


Figura 5-14 – Formulário Ordens Trabalho

Em Access, cada formulário é tratado como um objeto, portanto, cada um tem métodos e propriedades associadas. A propriedade que define os registos a utilizar no formulário tem o nome de `RecordSource`.

Com a utilização do quarto argumento do método `OpenForm`, é necessário que o campo `ID_OT` (Código 5-25) esteja presente no *RecordSource* (pode ser uma tabela ou um *query*) do formulário que se pretende abrir.

O evento *Form.Load ()* ocorre quando o formulário é aberto e o primeiro registo é apresentado, já o *Form.Open ()* ocorre quando o formulário é aberto, mas antes do primeiro registo ser apresentado e pode cancelar a abertura do formulário [41]. Esta será a altura para confirmar algumas condições importantes para o formulário abrir. No caso do formulário `frmOrdensTrabalho` é necessário ter em atenção que é constituído por subformulários que abrem antes do formulário principal. O subformulário `frmMaterialOrdemTrabalho` é aberto primeiro e será neste que o evento *Form.Open()* irá ocorrer.

Neste caso não foram criadas condições particulares de abertura. Apenas o *Form.Load()*, do formulário `frmOrdensTrabalho`, utiliza os dados de *OpenArgs*, enviados no sétimo argumento de `DoCmd.OpenForm`, para gravar o formulário de origem, ou seja, o formulário que deu ordem a este para abrir. Esta solução permite que, quando o formulário é fechado, regresse ao formulário de origem.

```
ElseIf OpenArgs = "inicio" Then
FormOrigem = "inicio"
Me.cmdSaveClose.Visible = False
```

Código 5-26 . Evento *Form_Load*() do formulário Ordens Trabalho

No código 5-26 verifica-se que `FormOrigem` (variável privada do formulário) é igual aos dados fornecidos por `OpenArgs` e indica o formulário de origem. Com essa informação, há um botão que é omitido usando a propriedade `Visible` do objeto do formulário `cmdSaveClose`.

Pretende-se agora salientar algumas funcionalidades presentes no subformulário `frmMaterialOrdemTrabalho` onde são introduzidas as referências de materiais utilizados nas Ordens de Trabalho. Este formulário surge identificado em "Ordens Trabalho" como "Material utilizado" Figura 5-14

As funcionalidades existentes, quando o utilizador coloca uma referência na caixa de texto "Referência", procuram validar a sua pré-existência nos registos da tabela `tblMaterial`. O funcionamento dessa validação segue a mesma lógica de exemplos anteriores e encontra-se no Anexo XI – Gerar gráfico em Excel com os dados do tipo de manutenção.

Deseja-se apenas salientar que, quando a referência não está registada ela pode ser adicionada diretamente na caixa de texto e seguir para o formulário `frmMaterial` relativo à introdução de dados para esse material. O utilizador, neste caso, deve ter presente alguma informação pertinente como o preço, o desconto e o fornecedor, para que os campos possam ser preenchidos com os dados que serão necessários para o custo na "Ordem Trabalho".

Ao introduzir uma referência registada no impresso "Pedido Intervenção" (Anexo V – Impresso de Registo das Intervenções da Manutenção), é executado o evento `txtRefInternaMiralago_beforeUpdate(Cancel As Integer)` que, para além de fazer a avaliação referida anteriormente, questiona o utilizador se pretende adicionar a referência, entretanto escrita na caixa de texto.

```
'Avaliar o resultado da pesquisa de referência
If strIDMaterial = "False" Then
  a.   lResposta = MsgBox("A referencia " & strReferenciaMiralago & " não
    existe. " & vbCrLf & " Pretente cria-la?", vbYesNo)

  b.   If lResposta = vbYes Then
   i. DoCmd.OpenForm "frmMaterial", , , , , , "OT;" & strReferenciaMiralago
  c.   ElseIf lResposta = vbNo Then
   i. Cancel = True
  d.   End If

'Caso a referencia exista os dados são colocados no campo respectivo
ElseIf Not IsNull(strIDMaterial) Then
  a. Me.IDMaterial.Value = BaseDeDados.FuncIDMaterial(strReferenciaMiralago)
```

Código 5-27 – Avaliação *BeforeUpdate* após pesquisa de existência de referência

Caso a resposta do utilizador seja no sentido de introduzir a referência, como se pode verificar no código 5-27 (`lResposta = vbYes`), é executado o método usado anteriormente para abrir o formulário `frmMaterial`. Neste caso, o sétimo argumento deste método (`DoCmd.OpenForm`) é utilizado para enviar para o formulário a abrir, dados do formulário de origem (`OT`), como anteriormente, mas também a referência que foi introduzida (`strReferenciaMiralago`). O separador utilizado foi o `;` (ponto-e-virgula).

Como já abordado anteriormente, a abertura de um formulário é uma sucessão de eventos.

O evento `Form_Open()`, focado no código 5-28 (Anexo XIV – Evento de abertura do formulário frmMaterial) separa *OpenArgs* em duas *strings* usando a função `Split`. Esta apresenta quatro argumentos (string, separador, limite, comparar): o argumento "*string*" constitui a base que será utilizada para avaliação; o segundo argumento ("separador") refere-se ao caracter que se pretende utilizar para separar a string (está pré-definido o espaço " " como delimitador); no argumento "limite" pode-se definir o número de partes em que queremos dividir a *string*; e o quarto argumento pode ser utilizado, no caso de caracteres de texto, numa mensagem, que é necessário usar como separador. Um exemplo, será a utilização de letras maiúsculas para separar uma *string* com letras minúsculas iguais ("A" e "a").

```
1 strOpenArgs = Split(OpenArgs, ";")
  'Receber dados para adicionar nova referência recebida de outro formulário
2 FormOrigem = strOpenArgs(0)

3 If FormOrigem = "inicio" Then
  a.   DoCmd.GoToRecord , , acLast

4 ElseIf FormOrigem = "OT" Then

    'Caso sejam enviados argumentos para a abertura da ref. é aberto um novo
    registo
  a.   DoCmd.GoToRecord , , acNewRec
    'A nova referencia
  b.   strReferenciaMiralago = strOpenArgs(1)

    'A nova referência é escrita em material e colocada em textbox bloqueada
  c.   Me.txtRefInternaMiralago = strReferenciaMiralago
  d.   Me.txtRefInternaMiralago.Locked = True

5 End If
```

Código 5-28 – Dados de *OpenArgs* são utilizados para abrir nova referência

Com esta separação, o vetor de duas posições contém na primeira (`strOpenArgs(0)`) a informação do formulário de origem ("`OT`") e na segunda, a referência a utilizar na caixa de texto `txtRefInternaMiralago`. Após o preenchimento deste campo, a caixa de texto é bloqueada utilizando a propriedade `Locked` com valor `True`. O utilizador terá então que completar a informação do artigo e ao fechar ele estará disponível no formulário de origem (`frmMaterialOrdemTrabalho`).

```
1 Dim intID As Integer

2 intID = Me.ID.Value
3 DoCmd.Close acForm, "frmMaterial"

  'Após o fecho do formulário verificar que formulário desencadeou a abertura
  deste e desencadear as acções de acordo com tal
4 Select Case FormOrigem
5 Case "OT"
  a.   Form_frmOrdensTrabalho.SetFocus
  b.   Form_frmMaterialOrdemTrabalho.IDMaterial.Value = intID
  c. DoCmd.Restore
```

Código 5-29 – Envio de dados para o campo do formulário frmMaterialOrdemTrabalho

A estratégia utilizada para preencher o campo do formulário, tem por base que o formulário de origem (`frmMaterialOrdemtrabalho`) está aberto no registo que se pretende preencher. Neste caso, o campo `IDMaterial`, que é um objeto dentro do formulário, tem a propriedade `Value`, cujo valor é o `ID`, referente ao registo da referência do artigo, entretanto criado no formulário `frmMaterial`. Será com base neste preenchimento que a designação e o preço do artigo será obtido. É utilizada a função `Dlookup`, já analisada, para obter estes valores para os campos respetivos.

O utilizador pode introduzir a referência diretamente ou procurar a referência numa lista. Tal opção ocorre com o duplo clique na caixa de texto onde se pretende colocar a referência do artigo. Ao executar esta ação, o evento *txtRefInternaMiralago_DblClick (Cancel As Integer)* (Código 5-30) é desencadeado.

```
1  Private Sub txtRefInternaMiralago_DblClick(Cancel As Integer)
2  Me.txtRefInternaMiralago = " "

3  DoCmd.OpenForm "frmFindmaterial", , , , , , "MaterialOT;" &
   Me.ID_OrdemTrabalho & ";" & Me.ID

4  End Sub
```

Código 5-30 – Ao executar duplo clique na caixa de texto *txtRefInternaMiralago* é desencadeado este evento

Como anteriormente, a abertura do formulário utiliza o método *DoCmd.OpenForm* e no argumento *OpenArgs* foram introduzidos dados relativos ao formulário de origem, ao número da Ordem de Trabalho e o ID do registo, na tabela *tblMaterialOrdemTrabalho* (Figura 5-3 – Organização de tabelas da base de dados. pág. 28), onde se encontram os dados de todo o material utilizado nas várias ordens de trabalho.

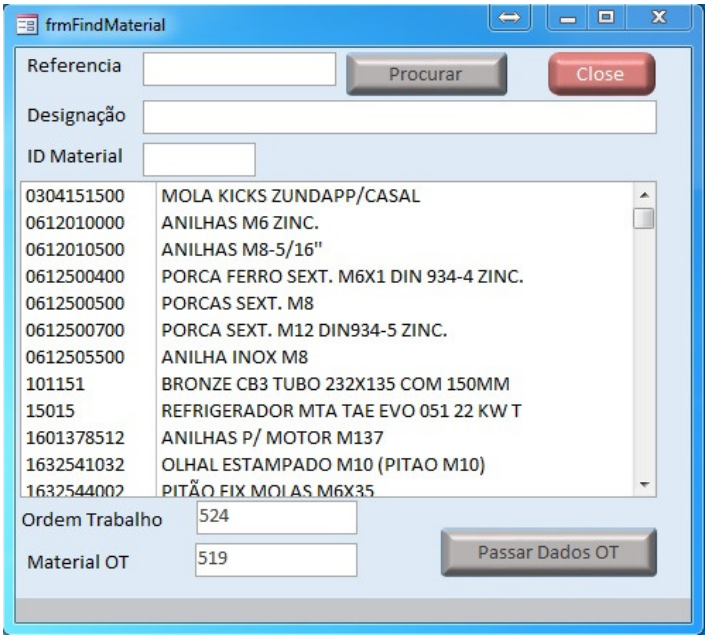


Figura 5-15 – Formulário para encontrar referências de artigos

Ao clicar no botão "Procurar" é verificado que caixas de texto ("Referencia", "Designação") têm algum texto para ser pesquisado. Se ambas forem preenchidas é executado o SQL indicado no código 5-31, que verifica todas as referências que começam por "56" e, juntamente, todas as designações que contenham "ca", da tabela *tblMaterial* (ver figura 5-16)

```
SELECT tblMaterial.ID, tblMaterial.RefInternaMiralago,
tblMaterial.DesignacaoMiralago
FROM tblMaterial
WHERE (((tblMaterial.RefInternaMiralago) Like '56*') AND
((tblMaterial.DesignacaoMiralago) Like '*Ca*'))
ORDER BY tblMaterial.RefInternaMiralago;
```

Código 5-31 – SQL para pesquisa da referência de artigos

O resultado desta pesquisa surge na figura 5-16 na forma de uma lista com duas colunas visíveis. Na primeira coluna está a referência que corresponde à pesquisa (começa por "56") e na segunda coluna a designação respetiva que contem "ca".

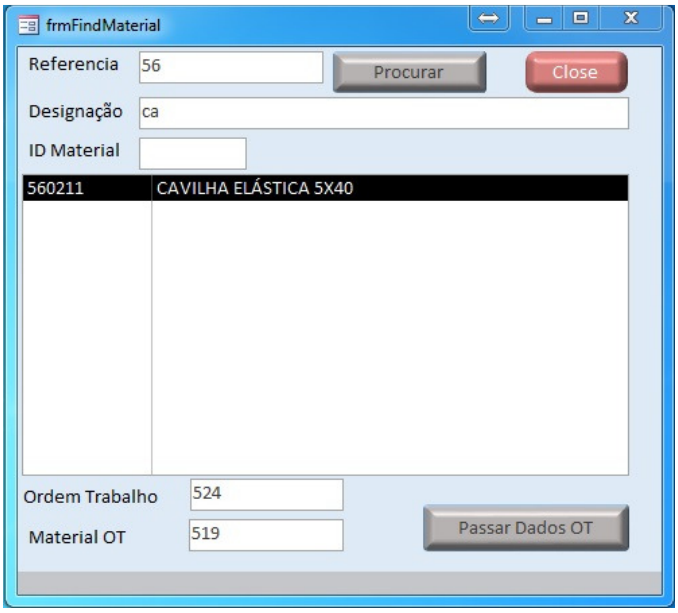


Figura 5-16 – Resultado da pesquisa executada no formulário *frmFindMaterial*

No código 5-32 encontra-se ilustrada a forma como os dados são introduzidos na lista. O código SQL utilizado para fazer a pesquisa é copiado para a propriedade `RowSource` da lista.

```
1 Else
  a.   strTask = "SELECT tblMaterial.ID, tblMaterial.RefInternaMiralago,
    tblMaterial.DesignacaoMiralago " _
    & "FROM tblMaterial " _
    & "WHERE (((tblMaterial.RefInternaMiralago) Like '" & Me.txtRefMiralago &
    "*') AND ((tblMaterial.DesignacaoMiralago) Like '*" & Me.txtDesignacao &
    "*')) " _
    & "ORDER BY tblMaterial.RefInternaMiralago;"
2 End If

3 If Me.txtRefMiralago.Value = " " Then Me.txtRefMiralago = Null

4 Me.ListRefMiralago.RowSource = strTask
```

Código 5-32 – Preenchimento da lista no formulário *frmFindMaterial*

O utilizador, deve fazer duplo clique no artigo que pretende, o que desencadeia mais um evento `ListaRefMiralago_DblClick(Cancel As Integer)` (Código 5-33). Este preenche as caixas de texto com os dados da linha que foi clicada.

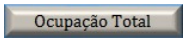
```
1 Private Sub ListRefMiralago_DblClick(Cancel As Integer)

2 Me.txtRefMiralago = Me.ListRefMiralago.Column(1)
3 Me.txtDesignacao = Me.ListRefMiralago.Column(2)
4 Me.txtIDMaterial = Me.ListRefMiralago.Column(0)

5 End Sub
```

Código 5-33 – Preenchimento dos dados da consulta nas caixas de texto respetivas

Com a confirmação dos dados do artigo é necessário colocá-lo na lista de material da "Ordem Trabalho". A forma como os dados foram exportados para o formulário é similar ao abordado anteriormente, e portanto, não irá ser aprofundado novamente.

Relativamente aos formulários associados à introdução de artigos no formulário "Material" irá ser feita uma análise mais aprofundada mais à frente.

Voltando ao formulário "Inicio" pode-se analisar alguns indicadores, entretanto criados, com base nos dados introduzidos nas "Ordens Trabalho" e nos "Pedidos de Intervenção". A distribuição de horas de trabalho por tipo de intervenção (Correctiva, Condicionada, Preventiva,…) e a distribuição de horas trabalhadas por Recurso Humano em cada mês. O objetivo deste último está relacionado com falhas no registo dos tempos de intervenção, resultado da forma manual como é executado.

A 

resulta do registo feito no campo `TipoIntervencao` na tabela `tblDadosIntervenção`, cujo formulário e campo está apresentado na figura 5-17

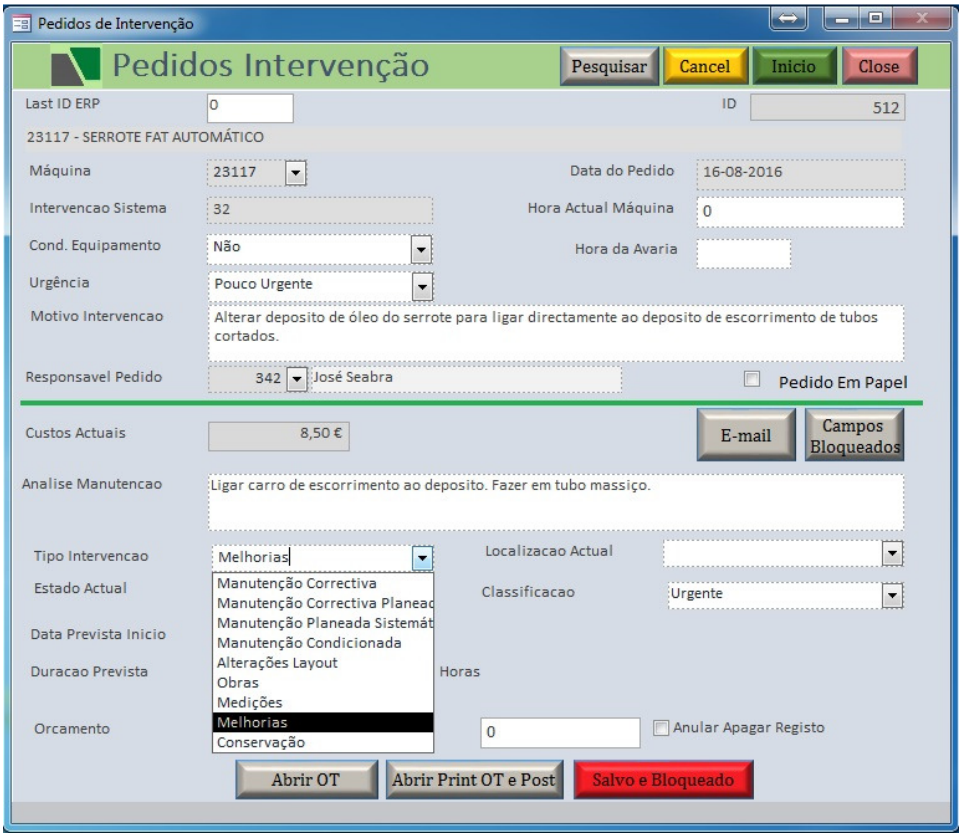


Figura 5-17 – Escolha do tipo de intervenção é feito pelo Departamento de Manutenção

O registo do tipo de intervenção pode ocorrer na altura em que o departamento faz a avaliação da intervenção a realizar, ou pode ser alterada mais tarde fruto de uma reavaliação.

A introdução de gráficos já foi abordada anteriormente pelo que aqui irá ser apresentada uma outra forma de obtenção de dados das tabelas usando SQL.

Com o formulário `frmOcupacaoTotal` (Figura 5-20) é apresentada a totalidade de horas registadas durante o mês na execução dos serviços e o tipo de manutenção que lhe foi atribuída. Ao clicar no botão "Actualizar" é desencadeado o evento `cmdActualizar_Click()` que executa o SQL de acordo com o preenchimento das caixas de texto do formulário.

O SQL presente no código 5-34 permite obter o conjunto de registos com as horas registadas por dia e por intervenção realizada. O resultado desta listagem é o conjunto de horas totais registadas por dia em determinada "Ordem Trabalho".

```
--qryHorasRecursosOT Horas dos recursos por ordem de trabalho
SELECT tblOrdensTrabalho.ID_Intervencao, tblRecursosOrdemTrabalho.ID_Sistema,
tblRecursosOrdemTrabalho.Data, tblRecursosOrdemTrabalho.HoraInicio,
tblRecursosOrdemTrabalho.HoraFim, Round(([HoraFim]-[HoraInicio])*24,2) AS
TotalHoras
FROM tblOrdensTrabalho INNER JOIN tblRecursosOrdemTrabalho ON
tblOrdensTrabalho.ID_OT = tblRecursosOrdemTrabalho.ID_OrdemTrabalho;
```

Código 5-34 – SQL para obtenção das horas registadas nas Ordens de Trabalho

Para a construção deste *query*, foi utilizada a *interface* gráfica do Access. No menu Create -> Query Design  adiciona-se tabelas, ou outros *querys*, que pretendemos utilizar na consulta.

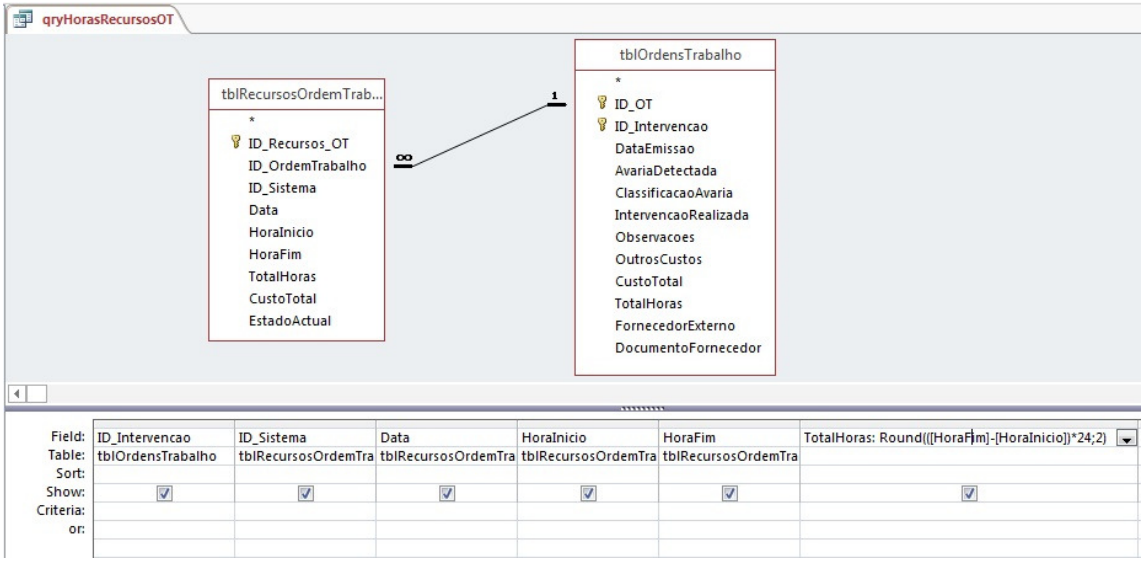


Figura 5-18 – Desenho de SELECT query usando Access

Para gerar o SQL do código 5-34 foram adicionadas as tabelas `tblRecursosordemTrabalho` e `tblOrdemTrabalho` como se pode verificar na figura 5-18. Na parte inferior da figura, observa-se várias colunas, onde são adicionados os campos das duas tabelas e na última coluna é feita a operação para obtenção das horas de trabalho (`TotalHoras:Round(([HoraFim]-[HoraInicio])*24;2)`). Da função `Round` resulta o arredondamento com o número de casas decimais colocado no segundo argumento (`2`), do valor colocado no primeiro argumento `([HoraFim]-[HoraInicio])*24`.

Para além deste tipo de *querys* é também possível desenhar outros tipos, utilizando este método. Pode-se criar tabelas, adicionar dados, atualizar e apagar dados.

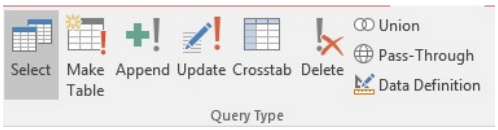


Figura 5-19 – Tipo de *querys* disponíveis com Access

O conjunto de dados obtido com este *query* é depois conciliado com a tabela *tblDadosIntervencao* e o campo *TipoIntervencao* entre datas (Código 5-35). As horas obtidas são somadas, se a elas estiverem associadas determinado tipo de intervenção (manutenção).

```
--Distribuição do total de horas por tipo de manutenção entre datas
SELECT tblDadosIntervencao.TipoIntervencao, Sum(qryHorasRecursosOT.TotalHoras)
AS SumOfTotalHoras
FROM tblDadosIntervencao INNER JOIN (qryHorasRecursosOT INNER JOIN
tblOrdensTrabalho ON qryHorasRecursosOT.ID_Intervencao =
tblOrdensTrabalho.ID_Intervencao) ON tblDadosIntervencao.ID =
tblOrdensTrabalho.ID_Intervencao
WHERE (((qryHorasRecursosOT.Data) > #10-01-2016#) AND
((qryHorasRecursosOT.Data) < #10-31-2016#) )
GROUP BY tblDadosIntervencao.TipoIntervencao ORDER BY
Sum(qryHorasRecursosOT.TotalHoras);
```

Código 5-35 – Informação do tipo de intervenção ordenada por soma total de horas

No mesmo formulário foi também adicionado uma outra lista para incluir o valor total de horas registadas. A utilização de um objeto tipo lista, e não caixa de texto, está relacionada com a origem dos dados. As caixas de texto não têm a propriedade *RowSource* disponível, pelo que a utilização de uma lista resolveu este problema sem ser necessário adicionar mais código.

```
--Soma total de horas entre datas
SELECT Round(Sum([qryHorasRecursosOT].[TotalHoras]),2) AS SumOfTotalHoras
FROM tblDadosIntervencao INNER JOIN (qryHorasRecursosOT INNER JOIN
tblOrdensTrabalho ON qryHorasRecursosOT.ID_Intervencao =
tblOrdensTrabalho.ID_Intervencao) ON tblDadosIntervencao.ID =
tblOrdensTrabalho.ID_Intervencao
WHERE (((qryHorasRecursosOT.Data)>#10-01-2016#) AND
((qryHorasRecursosOT.Data)<#10-31-2016#));
```

Código 5-36 – Soma total de horas para o mês em análise

```
1 'criar objecto para o book e a folha no documento excel
2 Set xlBook = xlApp.Workbooks.Add
3 Set xlSheet = xlBook.Worksheets(1)

4 'Na folha colocar os dados nas células indicadas
5 With xlSheet
  a.   .Range("B1").Value = "Tipo Manutenção em " & Nz(Me.txtMes.Column(1),
    "????")

  b.   i = 2
  c.   Do While Not rs.EOF
    i. .Range("A" & i).Value = Nz(rs!TipoIntervencao, "???")
   ii. .Range("B" & i).Value = Nz(rs!SumOfTotalHoras, 0)

  iii. i = i + 1
  d.   rs.MoveNext
  e. Loop
```

Código 5-37 – Criação dos objetos Excel e preenchimento dos dados

Os dados disponibilizados neste gráfico permitem apresentar à Administração o tipo de manutenção que o departamento tem vindo a desenvolver. Levanta depois a questão, se é esta a manutenção que se pretende, ou deveremos evoluir para um tipo de manutenção diferente. Por exemplo, com base no gráfico da figura 5-20 poderemos concluir que no mês de Outubro, à data de recolha de dados, nenhuma intervenção preventiva tinha sido executada e que 40% das intervenções foram corretivas de urgência, ou seja, não planeadas.

A análise de dados fora do ambiente mais restrito da base de dados para o utilizador, é certamente uma mais valia. Foi adicionada a hipótese de exportação dos dados, e respetivo gráfico, para um documento Excel (Anexo XI – Gerar gráfico em Excel com os dados do tipo de manutenção). O utilizador pode então usar os dados, copiados para o Excel, e formatar de forma mais atrativa o gráfico caso o entenda. Na figura 5-21 é apresentado o gráfico gerado automaticamente ao pressionar o botão "Excel" no `frmOcupacaoTotal`.

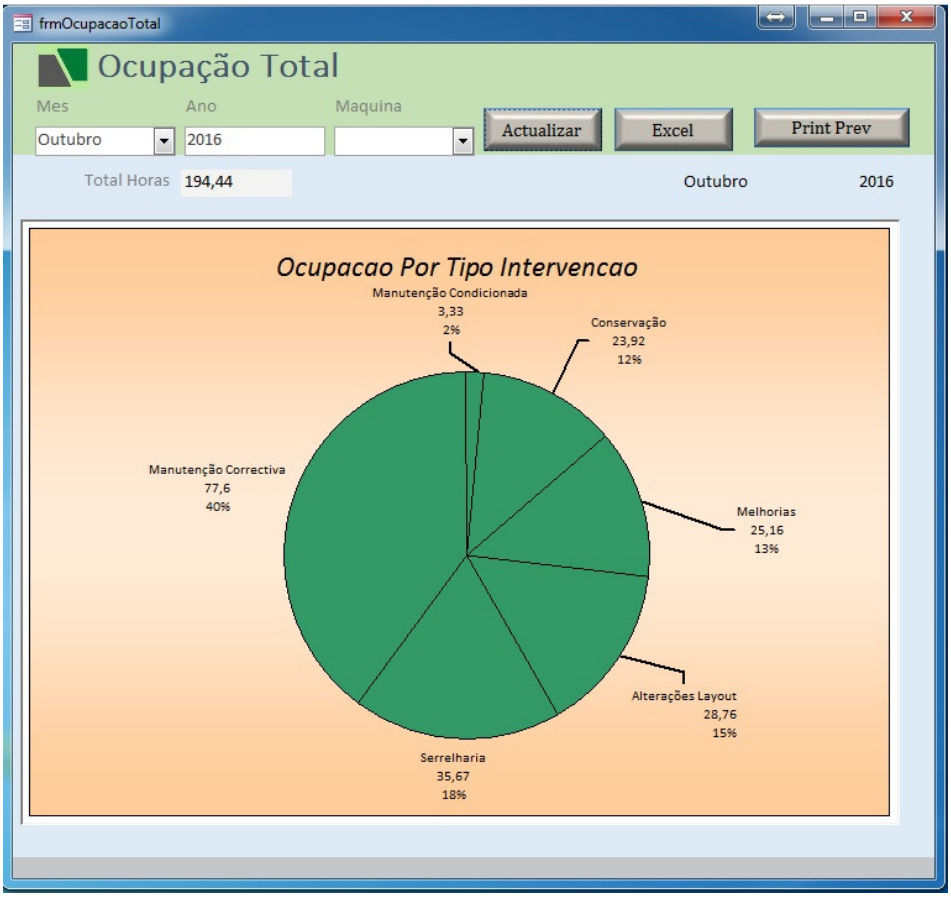


Figura 5-20 – Formulário de distribuição temporal mensal por tipo de intervenção

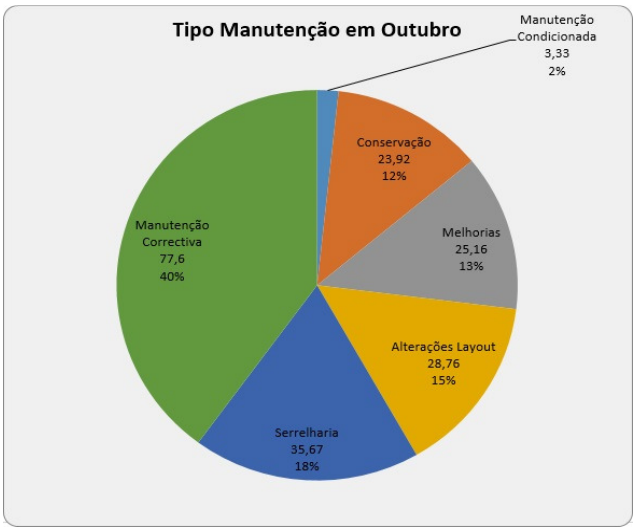


Figura 5-21 – Gráfico no Excel apresentado, ao ser pressionado o botão "Excel" em `frmOcupacaoTotal`

Como em exemplos anteriores, é necessário referenciar a biblioteca "Microsoft Excel 16.0 Object Library" (ver Figura 5-12 – Escolha de bibliotecas de outras aplicações. página 39) por forma a utilizar os seus objectos e o "intellisense" (conclusão de código inteligente) presente no ambiente VBA, que facilita, na escrita do código, e facilita o acesso a algumas funções disponíveis sem necessidade de pesquisas mais aprofundadas.

Os dados obtidos com o *query* (código 5-35) são copiados para uma folha Excel, usando o código 5-37. A criação do gráfico utiliza as propriedades do objeto *Chart* [42] para definir o tipo de gráfico (*ChartType*) como *xlPie*, se tem ou não legenda (*HasLegend=False*) , a origem dos dados (*SetSourceData*) e a formatação do gráfico nas suas várias vertentes (Código 5-38).

Dos métodos do objeto *Chart*, definidos em VBA, destaca-se a *ApplyDataLabels* que contém vários argumentos (*Type*, *LegendKey*, *AutoText*, *HasLeaderLines*, *ShowseriesName*, *ShowCategoryName*, *ShowValue*, *ShowPercentage*, *ShowBubbleSize*, *Separator*) que permitem colocar um conjunto de informação no gráfico para facilitar a sua leitura e aumentar a informação disponibilizada.

```
1   With .Chart

    a.  .ChartType = xlPie

    b.  .HasTitle = True
    c.  '.ChartTitle = "Ocupação Tipo Manutenção"

    d.  .HasLegend = False
    e.  '.Legend.Position = xlLegendPositionBottom


    f.  With .ChartTitle.Font
          i.  .Name = "Calibri"
         ii.  .FontStyle = "Bold"
        iii.  .Size = 16

    g.  End With

    h.  .SetSourceData Source:=xlSheet.Range("A1:B" & i - 1)
    i.  .ChartArea.Interior.Color = RGB(240, 240, 240)
    j.  .ApplyDataLabels , , , , , True, , True, , Chr(13)

2   End With 'chart
```

Código 5-38 – Construção do gráfico em VBA

No argumento *Type* há seis (6) tipos de dados que podem ser parametrizados como marcadores dos campos do gráfico. Como era pretendido que os valores inteiros fossem mostrados nos campos do gráfico (*xlDataLabelsShowLabel*), esse argumento foi deixado em branco porque é assumido por omissão. O argumento *ShowCategoryName* foi utilizado para que o nome do tipo de manutenção pudesse surgir nos campos do gráfico. Para que fosse mostrada a percentagem nos campos, havia duas hipóteses: no argumento *Type* (*xlDataLabelsShowLabelAndPercent*) ou colocar o argumento *ShowPercentage=True*. Esta última hipótese acabou por ser a que foi seguida. Como por omissão o separador de dados é o "*;*" utilizou-se o último argumento deste método, *Separator*, para o alterar para o *carrige return,* usando *Chr(13)* que utiliza a função *Chr(). O* argumento é um número decimal da tabela ASCII (*American Standard Code for Information Interchange*) a que corresponde o caracter ou acção definida na referida tabela.

Outro formulário que apresenta dados relativos aos recursos, junta as horas trabalhadas por dia, com as intervenções onde essas horas foram registadas. É uma forma de sensibilizar os recursos humanos para a importância dos registos nas Ordens de Trabalho, pois as horas registadas não correspondem ao período efetivo de trabalho com desvios que chegam aos 50%.

Com este objetivo em mente, foi adicionada uma listagem de todas as intervenções do período selecionado e respetivo gráfico. Acede-se diretamente à intervenção que se pretende analisar, clicando duas vezes sobre o número desta.

Neste caso os procedimentos para obtenção dos dados foram similares aos mencionados para o formulário anterior e apenas se gostaria de destacar a função criada dentro do módulo *Horario*. Esta função (*DiasUteis*) recebe as datas de início e finalização, colocadas no formulário "Ordens trabalho" e determina as horas dos dias úteis. Neste momento, não adiciona feriados e férias dos recursos humanos.

DiasUteis utiliza a função *DateDiff* de Access. Como o próprio nome indica, esta função permite calcular uma diferença entre duas datas, em semanas, anos ou dias. Contém cinco argumentos (intervalo, data1, data2, primeiro dia da semana (*firstdayofweek*), primeira semana do ano (*firstweekofyear*)). O argumento intervalo (*interval*) é a unidade do intervalo de tempo que se pretende utilizar para calcular a diferença entre data1 e data2; os argumentos dois e três serão as datas final e inicial; o quarto argumento (*firstdayofweek*) tem pré-definido o domingo, mas pode ser utilizado outro dia; o quinto argumento também é opcional e assume que a primeira semana será aquela do primeiro de janeiro.

Outra função utilizada é *DatePart* de onde se obtém, entre outros dados, o dia da semana ou a semana do ano em que a data considerada se encontra. No caso da função *DiasUteis*, foi necessária para determinar que dias da semana eram as datas inicial e final. Caso a data inicial seja um domingo, como não é considerada na verificação do intervalo, terá que se adicionar um dia aos dias de fim-de-semana. O mesmo se passa com a última data se esta for um sábado.

Com este resultado é possível determinar a taxa de ocupação dos funcionários em horas, partindo do princípio que eles trabalham 40 horas por semana (8 horas por dia).

```
1  'Calcular o numero de dias inclusive (+1 para adicionar a data de inicio)
2  varDias = DateDiff(Interval:="d", date1:=startDate, date2:=endDate) + 1

3  ' Calcular o número de dias de fds completos com 2 dias de FDS
4  varDiasFDS = DateDiff("ww", startDate, endDate) * 2

5  'Se a data inicial for um domingo adicionar um dia
6  If DatePart("w", startDate) = vbSunday Then varDiasFDS = varDiasFDS + 1

7  'Se a data final for um sábado adicionar um dia
8  If DatePart("w", endDate) = vbSaturday Then varDiasFDS = varDiasFDS + 1

9  'Calcular o número de dias uteis.
10 DiasUteis = (varDias - varDiasFDS)
```

Código 5-39 – Determinação de dias úteis incluindo as datas definidas

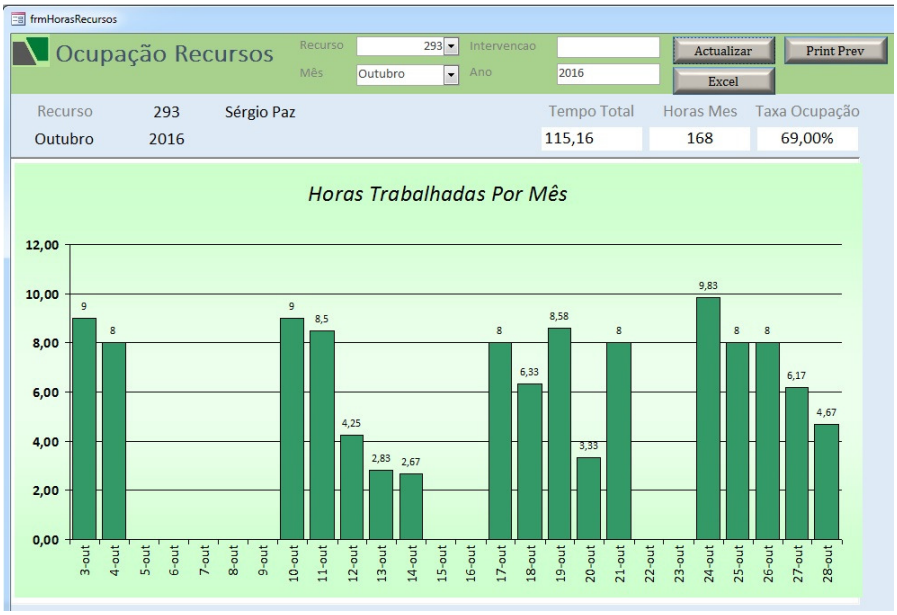


Figura 5-22 – Ocupação do recurso 293 no mês de Outubro 2016

Na figura 5-22 observa-se o gráfico gerado com base nos dados recolhidos. Na parte superior do gráfico, encontra-se informação da soma total de horas (Tempo Total), horas com base na função `DiasUteis` (Horas Mês) e a taxa de ocupação do tempo disponível (69%).

Também aqui foi adicionada a possibilidade de enviar os dados para um ficheiro Excel. Ao clicar no botão "Excel" surge a informação da quantidade de recursos humanos que estão registados no período considerado e o gráfico em Excel, com a distribuição das horas laborais por dia para cada recurso (Figura 5-23).

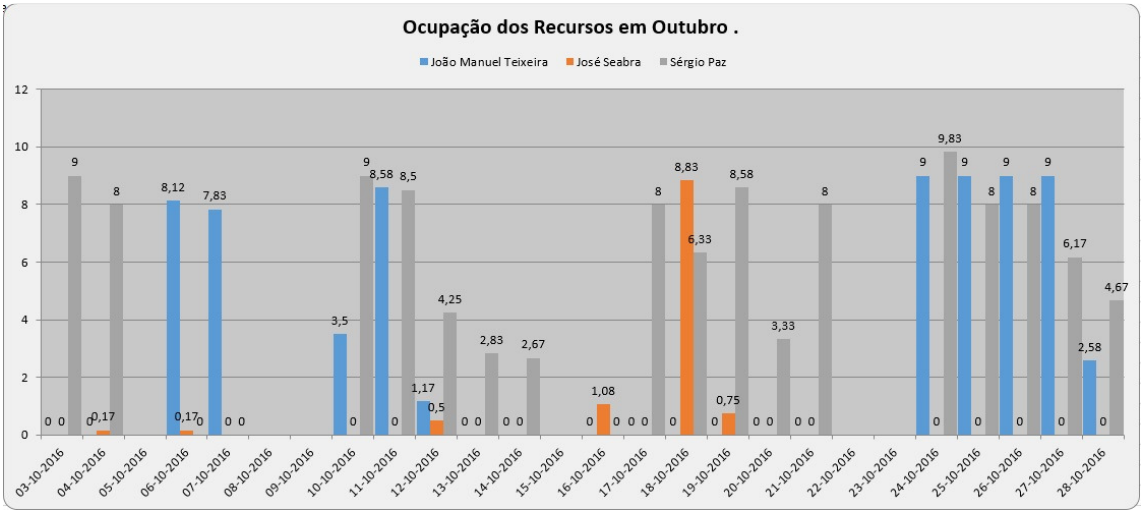


Figura 5-23 – Gráfico em Excel com os registos de trabalho dos vários recursos no mês de Outubro

Para organizar os dados necessários para construir o gráfico da figura acima foi necessário recorrer a outro tipo de query no Access. Recebe o nome de Crosstab e encontra-se no Menu-Design dos Querys  . Funciona como a função *PivotTable* do Excel onde é possível agrupar os dados de várias formas conforme a conveniência na apresentação dos resultados.

Na figura 5-24 ilustra-se a obtenção de informação a partir do total de horas registadas por recurso, que estão registadas na tabela `tblRecursoOrdemTrabalho`. A tabela fica então organizada

por data (`Row Heading`), recurso (`Column Heading`) e o valor das horas (`TotalHoras`) que é colocado como dado da tabela (`Value`). O resultado é a distribuição de horas pelos vários dias do período considerado.

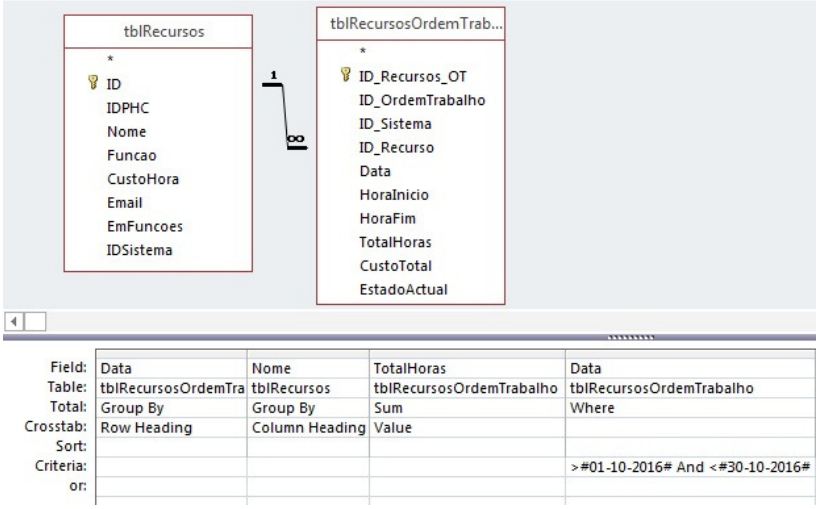


Figura 5-24 – *Query* para organização das horas por recursos humanos e data

Ao clicar no botão "Excel", o evento `cmdExcel_Click` (Anexo XII – Gerar gráfico com mais do que uma série de dados em VBA) utiliza os dados do *query* do código 5-40, cria os objetos necessários para que seja aberto um gráfico em Excel para o utilizador (Figura 5-23).

```
--Query para organizar os dados das tabelas por data/recurso
TRANSFORM Sum(tblRecursosOrdemTrabalho.TotalHoras) AS SumOfTotalHoras
SELECT tblRecursosOrdemTrabalho.Data
FROM tblRecursos INNER JOIN tblRecursosOrdemTrabalho ON tblRecursos.ID =
tblRecursosOrdemTrabalho.ID_Recurso
WHERE (((tblRecursosOrdemTrabalho.Data)>#10/1/2016# And
(tblRecursosOrdemTrabalho.Data)<#10/30/2016#))
GROUP BY tblRecursosOrdemTrabalho.Data
PIVOT tblRecursos.Nome;
```

Código 5-40 – *Query* usando a função *Crosstab* do Access para organizar os dados por data/recurso humano

A declaração `TRANSFORM` [43] surge antes da função SQL que agrupa os dados a colocar na tabela (neste caso a soma das horas de um recurso em determinado dia). De seguida, surge a declaração `SELECT` onde surgem os dados a colocar no cabeçalho das linhas (intervalo de datas) e por último, e no caso do Access sempre associado a `TRANSFORM`, deve-se colocar a declaração `PIVOT` com o campo a utilizar no cabeçalho das colunas (neste caso o nome do recurso).

Usando esta técnica é possível resumir um conjunto de dados e apresentá-los de forma muito mais inteligível. Neste caso pode-se obter a informação do conjunto de recursos humanos alocados à manutenção num determinado período, que dias trabalharam para a manutenção, quanto tempo, em que trabalhos e quem está a fazer um registo correto.

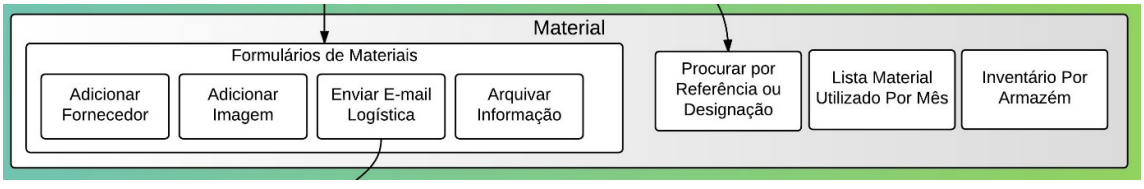


Figura 5-25 – Formulários associados com material no Departamento de Manutenção

Os formulários associados a materiais são utilizados para adicionar novas referências de artigos, adicionar fornecedores e manter uma lista de inventário atualizada. A aplicação, no entanto, não faz gestão de *stocks*, querendo com isto dizer que, à saída de material dada nas Ordens de Trabalho não corresponde uma quebra no stock.

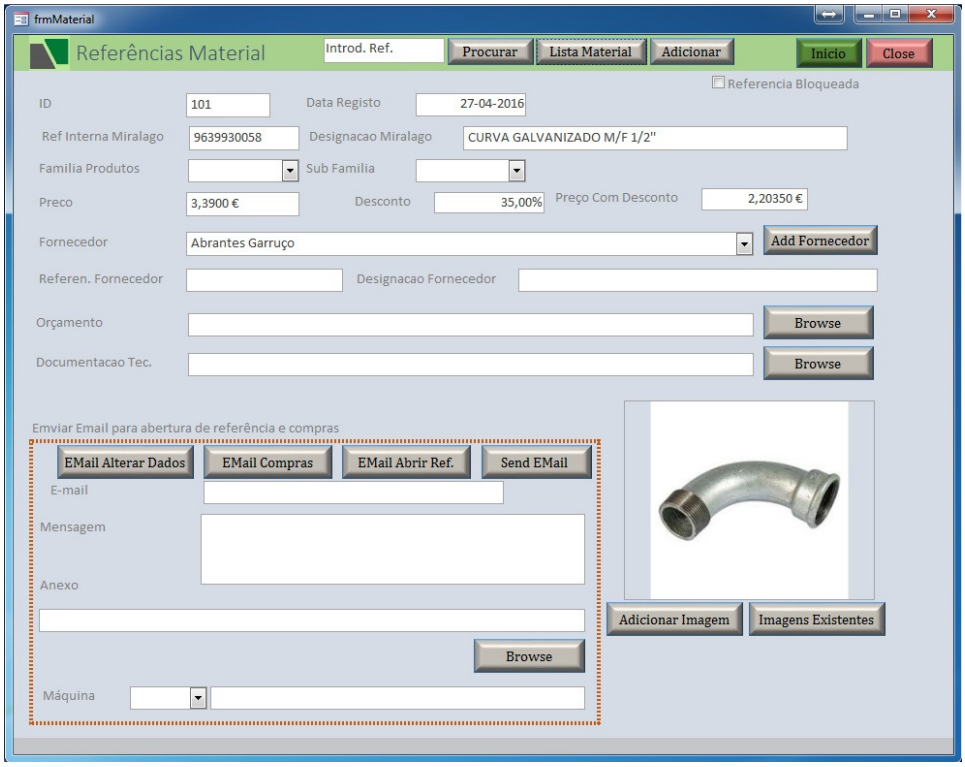


Figura 5-26 – Formulário de adição de observação de características de materiais

Como se pode verificar, neste formulário surge uma imagem. Porque as bases de dados em Access não podem ultrapassar os 2 *Gbytes*, optou-se por colocar no registo os endereços do sistema de ficheiros das imagens (bem como de todos os outros anexos) colocando-as em pastas do computador onde a aplicação está a ser executada. Aqui começam as limitações desta base de dados. Ao ser utilizada noutros computadores, para os utilizadores terem acesso às imagens, é necessário instalar a pasta de imagens indicadas nos registos. Ou seja, se um utilizador introduz uma imagem no formulário, outro utilizador não terá acesso a ela, se o endereço colocado na caixa de texto não corresponder a um endereço que o computador desse utilizador possa aceder.

Uma das formas para colocar as imagens diretamente nas tabelas em Access, é utilizar um campo para anexos (*attachment*). Com este campo definido na tabela, é possível utilizá-lo num controlo no formulário e adicionar vários ficheiros ao campo (sem quebrar as normas das base de dados pois o Access trata da criação das tabelas necessárias). Poderá ser uma melhoria futura, no caso de se manter a base de dados ligada apenas a tabelas de Access. Neste caso seria possível, para além de adicionar a imagem do artigo poder-se-ia adicionar o *datasheet* ou a informação de segurança.

No formulário da figura 5-26, após a introdução do "Fornecedor", os botões "Adicionar Imagem" e "Imagens Existentes" ficam visíveis.

O botão 

promove a abertura do formulário da figura 5-27 onde se procura uma imagem para adicionar ao conjunto de imagens registadas. Como se pode verificar, na caixa de texto "Link Imagem", o endereço surge claramente, e torna-se importante, usando este método, que haja organização das pastas onde se colocam as imagens.

Ao clicar no botão “Encontrar Imagem” o evento `cmdAdicionarImagem_Click()` é desencadeado o que por sua vez executa a função `procurarJPG()` do módulo `ficheiro`

Para encontrar o ficheiro, é aberta uma janela (caixa de diálogo) para que o utilizador escolha o ficheiro pretendido. É utilizada a propriedade `FileDialog` do objeto de `Application` por forma a que seja aberta a janela. Esta propriedade aceita um argumento(`dialogType`) que define que tipo de janela se pretende abrir. Ao escolher `msoFileDialogFilePicker` abre-se a janela para escolher o ficheiro.

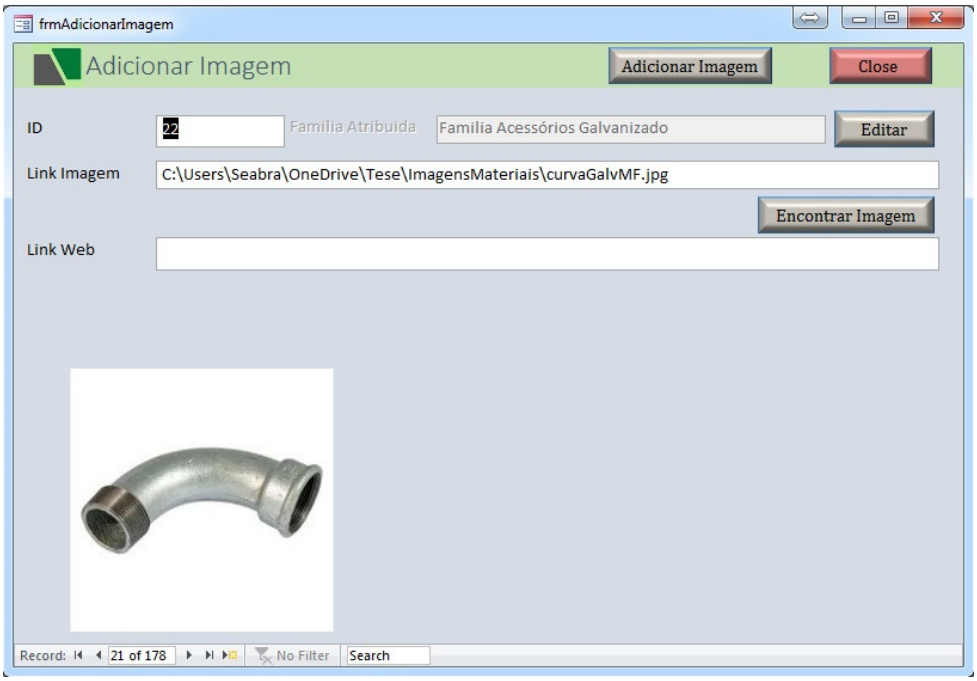


Figura 5-27 – Formulário de adição de imagens

```
1 Public Function procurarJPG() As String
2 'Procurar ficheiro usando a criação de objecto
3 Dim objFile As Object
4 Dim strOut As String

5 strOut = vbNullString

6 'Set up de file dialog objecto
7 Set objFile = Application.FileDialog(msoFileDialogFilePicker)

8 With objFile
  a.   ' Não permite a adição de mais do que um ficheiro
  b.   .AllowMultiSelect = False

  c.   'Titulo da janela
  d.   .Title = "Selecione um ficheiro JPG"

  e.   'Clear out the current filter, and add our own
  f.   .Filters.Clear

  g.   .Filters.Add "JPEG Files", "*.jpg"

9 If .Show = True Then
  a.   strOut = .SelectedItems(1)
10     End If

11     End With

12     Set objFile = Nothing

13     procurarJPG = strOut

14     End Function
```

Código 5-41 – Procurar ficheiro com extensão .jpg [44]

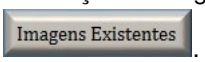
Para que o utilizador apenas possa escolher ficheiros jpg, é necessário utilizar algumas propriedades de *FileDialog*. *AllowMultiSelect* determina se o utilizador pode selecionar vários ficheiros, o que neste caso é falso (*False*);*Title* determina o título a colocar na caixa de diálogo (*"Selecione um Ficheiro JPG"*); *Filters* é utilizado para que o utilizador apenas possa escolher ficheiros com a extensão definida em *Filters.Add*.

Quando o utilizador escolhe um ficheiro clicando em "OK", o método *FileDialog.Show* apresenta o valor *True*. Para guardar a informação do ficheiro escolhido, a propriedade *SelectedItem* guarda os endereços dos ficheiros escolhidos. Neste caso, como se pretende apenas um endereço, foi colocado o argumento (1), para informar o sistema que apenas deve guardar o endereço de um ficheiro.

Como se pode constatar no código 5-42, função *ficheiro.procurarJPG* devolve uma *string* (*strEscolha*) com o endereço do ficheiro, que é colocada na caixa de texto "Link Imagem" (*txtLinkImagem*). A imagem do formulário é também atualizada, na linha *Me.imgMaterial.Picture*, com o mesmo endereço. A propriedade *Picture* permite a visualização da imagem no controlo.

```
1  Private Sub cmdAdicionarImagem_Click()
2  Dim strEscolha As String
3  strEscolha = ficheiro.procurarJPG

4  If Len(strEscolha) > 0 Then
   a. Me.txtLinkImagem = strEscolha
   b. Me.imgMaterial.Picture = Me.txtLinkImagem
5  End If
6  End Sub
```

Código 5-42 – Evento para escolha de imagem JPG do sistema de ficheiros

Voltando ao formulário "Referências Material", e com o endereço da imagem adicionado ao registo *tblImagem.LinkImagem* , podemos clicar no botão 

. Não é obrigatório adicionar uma imagem antes, mas depreendeu-se que ao criar um novo artigo, o utilizador deverá introduzir os dados e, um deles, deverá ser uma imagem associada. Isso não invalida o facto de se colocar uma imagem pré-existente na lista de imagens.

Para visualizar as imagens existentes, foi utilizado um outro tipo de formulário. Em Access recebe o nome de "*Split Form*" e trata-se de um formulário, que surge ao utilizador, juntamente com a lista de registos desse formulário em formato "*Datasheet*" (tabela).

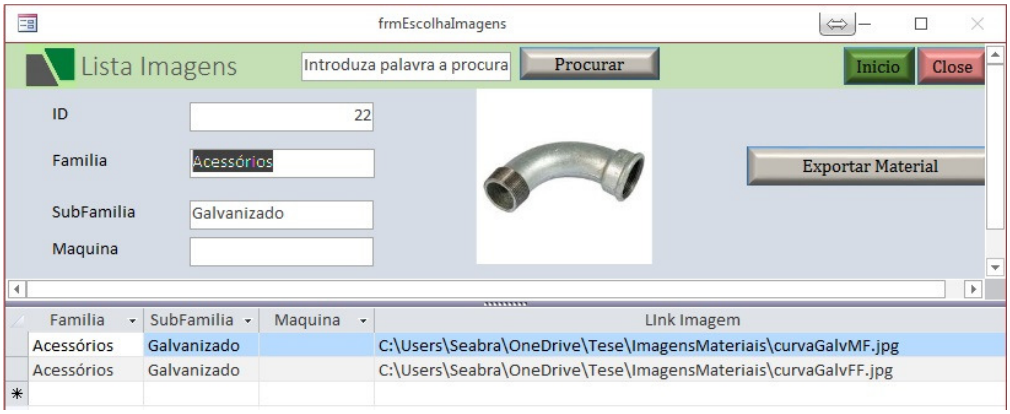
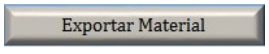


Figura 5-28 – Formulário de Escolha de imagens previamente carregadas na tabela tblImagens

Na figura 5-28 observa-se o "*Split Form*" de pesquisa de imagens. A pesquisa das imagens dá importância ao nome atribuído aos ficheiros, como se pode observar no *query* de pesquisa ilustrado no código 5-43. O nome atribuído ao ficheiro, na altura de o gravar na directoria "ImagensMateriais", deve seguir algumas regras. O nome deve conter, pelo menos, uma designação genérica do que está na imagem. No SQL do código seguinte, verifica-se que o utilizador procura todas as curvas registadas no nome dos ficheiros. Não interessa o material de que elas são feitas desde que sejam curvas.

```
SELECT tblImagem.ID, tblImagem.LinkImagem, tblImagem.LinkWeb,
tblImagem.Familia, tblImagem.SubFamilia, tblImagem.Maquina
FROM tblImagem
WHERE (((tblImagem.LinkImagem) Like '*curva*'));
```

Código 5-43 – Pesquisa de texto dentro do endereço de imagem

Ao clicar no botão 

a imagem é associada ao material, na tabela *tblMaterialImagem* usando a função *AdicionarImagemMaterial* no método *BaseDeDados*. O método utilizado para adicionar valores às tabelas já foi abordado e encontra-se no Anexo XIX – Associar Imagem a Material

O envio de 

utiliza a mesma estratégia já abordada anteriormente. A criação de objetos, para utilizar o cliente de e-mail instalado no computador onde se executa a aplicação.

A mensagem de e-mail enviada para a secção de compras, deve conter uma imagem do artigo que se pretende adquirir (Figura 5-29).

Figura 5-29 – Exemplo de mensagem enviada para a secção de compras

Para que seja possível, no Outlook, que esta imagem apareça no destinatário, foi criado um e-mail em HTML. As imagens, não podem ser adicionadas à mensagem diretamente. Têm de ser enviadas como anexo para que possam ser visualizadas no destino. Outra hipótese, e que funcionaria também para o caso dos servidores Gmail, seria colocar a imagem num endereço Web público. Não foi essa a estratégia, e foi necessário utilizar funções do Outlook que permitissem desenvolver o que se pretende.

Para que a imagem evidenciada na figura 5-29 surja na mensagem, é necessário identificar o nome do ficheiro (`terminalOlhalCobre.jpg`) separando-o do endereço completo no sistema de ficheiros. Para tal, foi utilizada a função `Split` com separação pela barra invertida "\" como se pode verificar no código 5-44. Posteriormente é contado o número de elementos do vetor criado, usando a função `UBound`. Esta pode receber dois argumentos, sendo o segundo (*Rank*), opcional, e permite identificar um determinado elemento dentro do vetor. No primeiro argumento da função (*Array*), indica-se o vetor, cuja dimensão se pretende avaliar.

O resultado será uma *string* com a informação da última posição do vector (`strImageFile`) que corresponde ao nome do ficheiro.

```
1 If IsNull(Me.imgImagemProduto.Picture) Then
  a.   strImagePath = ""
2 Else
  a.   strImagePath = Me.imgImagemProduto.Picture
  b.   SplitFile = Split(strImagePath, "\")
  c.   iArraySize = UBound(SplitFile)
  d.   strImageFile = SplitFile(iArraySize)
3 End If
```

Código 5-44 – Obtenção do nome do ficheiro da imagem a ser enviada por e-mail

Para adicionar a imagem como anexo foi utilizado o método do Outlook `Attachments.Add` que aceita quatro argumentos [45]: o primeiro é a origem do ficheiro (*Source*), que no caso é o endereço do sistema de ficheiros da imagem; em *Type* (segundo argumento) indica-se o tipo de anexo, que no

caso é *olByValue,* onde se informa que é uma cópia do ficheiro original e pode ser acedido mesmo que o ficheiro original tenha sido apagado; a *Position* é utilizado em formato *Rich Text* e determina a posição onde o ficheiro deve aparecer (ao ser colocado o valor zero, o anexo fica escondido); o quarto argumento é o *DisplayName* e também apenas se aplica em formato *Rich Text* pois quando se trata do formato HTML o nome do anexo é o do ficheiro utilizado.

```
1   If FSO.FileExists(strImagePath) Then
    a.  .Attachments.Add strImagePath, 1, 0

    b.  .HTMLBody = .HTMLBody & "<br><br>" & Me.txtBox & "</font><br>" _
        & "<hr><br><br>" & strBody & "</font><p>Fornecedor: " & strFornecedor &
        "</font></span>" _
        & "<hr><img src='cid:" & strImageFile & "'" _
        & "alt='Imagem Não Disponivel'" _
        & "with='500' height='200'><br>"

    c.  strBodyGmail = "<br><br>" & Me.txtBox & "</font><br>" _
        & "<hr><br><br>" & strBody & "</font><p>Fornecedor: " & strFornecedor &
        "</font></span>"

2   Else
    a.  .HTMLBody = "<br><br>" & Me.txtBox & "</font><br>" _
        & "<hr><br>" & strBody & "</font><p>Fornecedor: " & strFornecedor &
        "</font></span>"
3   End If
```

Código 5-45 – Construção da mensagem a enviar à secção das compras

Após a introdução do endereço da imagem, como anexo da mensagem, é feita a referência a ela dentro do código HTML com a tag (identificador) `<img>` com o argumento `src` (source).

De acordo com a RFC2392 [46] uma das formas para relacionar as imagens incluídas nas mensagens, ou outro ficheiro, usando código HTML, é utilizando *Content-ID Uniform Resorce Locator* (`cid:'URL'`). O código HTML faz então referência ao nome do ficheiro `strImageFile`, obtido anteriormente, dentro da mensagem a enviar, no argumento `src` da *tag* `<img>`.

No código 5-45 também é gerado o corpo da mensagem para enviar pelo servidor Gmail (`strBodyGmail`). O processo de adição de imagens ao corpo da mensagem para este servidor, segue outros requisitos, alguns deles relacionados com a segurança e localização da imagem a ser incluída. A empresa, recentemente, passou a utilizar o servidor *Gmail* pelo que haveria vantagens em o utilizar para o envio de mensagens.

Como referido anteriormente, o envio de mensagens de e-mail, usando diretamente o servidor, pode ser conseguido utilizando a biblioteca CDO (*Collaboration Data Objects)* que foi desenhada para criar ou manipular mensagens enviadas na Internet. CDO para Windows 2000, suporta o envio de mensagens com protocolo SMTP (*Simple Mail Transfer Protocol*) e NNTP (*Network News Transfer Protocol*) pelo que basta o acesso a um servidor SMTP (`smtp.gmail.com`) para que seja possível o envio de mensagens [47].

```
1   Public Function SendGmail(SendTo, Subject, Body, Attach)

2   Dim Mail As New CDO.message
3   Dim Config   As CDO.Configuration

4   Set Config = Mail.Configuration
5   Config(cdoSendUsingMethod) = cdoSendUsingPort
6   Config(cdoSMTPServer) = "smtp.gmail.com"
7   Config(cdoSMTPAuthenticate) = cdoBasic
8   Config(cdoSMTPUseSSL) = True
9   Config(cdoSendUserName) = "jose.seabra@miralago.pt"
10  Config(cdoSendPassword) = "10254078"
11  Config.Fields.Update

12  Mail.To = SendTo
13  Mail.CC = "julio.chio@miralago.pt"
14  Mail.From = Config(cdoSendUserName)
15  Mail.Subject = Subject
16  Mail.HTMLBody = Body

17  If Not Attach = "" Or IsNull(Attach) Then
    a.  Mail.AddAttachment Attach
18  End If

19  On Error Resume Next

20  Mail.Send

21  If Err.Number <> 0 Then
    a.  MsgBox Err.Description, vbCritical, "There was an error"
22  End If

23  End Function
```

Código 5-46 – Envio de mensagens de E-mail utilizando diretamente o servidor externo

A biblioteca CDO apresenta muitas opções para o envio de mensagens. Está implementado o envio, com as configurações mínimas, para que funcione com o servidor SMTP Gmail. Faz-se de seguida uma análise ao código 5-46. Primeiro inicializam-se dois objetos, um para as mensagens (*Mail*) e outro para as configurações (*Config*) do servidor. *CDO.message* apresenta diversas propriedades de configuração da mensagem a enviar, tal como os endereços de envio (*.To* e *.CC*), o assunto (*.Subject*) e o corpo (*.HTMLBody*). O objeto para a configuração apresenta diversas propriedades (campos) que são preenchidas de acordo com as necessidades. O *cdoSendUsingMethod* deve ser configurado por forma a definir se o servidor SMTP se encontra local ou remotamente. Como o servidor é remoto, é necessário utilizar o valor dois (2) ou *cdoSendUsingPort*; em *cdoSMTPServer* coloca-se o endereço do servidor (*smtp.gmail.com*); em *cdoSMTPAuthenticate* indica-se como são enviados os dados do utilizador, no caso em texto (“clear-text”) e portanto, com a atribuição *cdoBasic* ou o valor 1 (um); no campo *cdoSMTPUseSSL* atribui-se o valor *True* pois irá utilizar-se *Secure Socket Layer* (SSL) para o envio das mensagens; os campos *cdoSendUserName* e *cdoSendPassword* são preenchidos com o texto das credenciais do utilizador [48].

Com os valores introduzidos, é necessário criar o objeto de configuração com a linha *Config.Fields.Update.*

Foi ainda implementado um outro serviço, para o caso de haver atrasos na resposta da secção de compras e que implementa algumas soluções atrás apresentadas.

A figura 5-30 apresenta o formulário que ilustra a informação essencial que é guardada aquando do envio dos e-mails na tabela *tblLogEnvioEmails*.

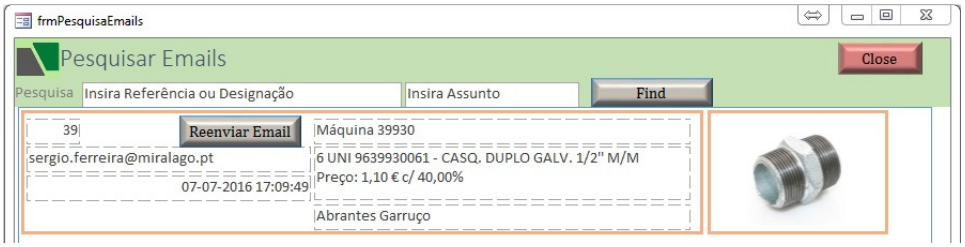


Figura 5-30 – Pesquisa de E-mails enviados

Para o reenvio de emails é utilizada a conta Gmail, que apesar de não incluir a imagem, que terá sido enviada anteriormente, ao não utilizar o cliente SMTP do computador (Outlook) executará mais rapidamente. Neste formulário as pesquisas podem ser feitas por assunto, normalmente máquina, referência de material ou a própria designação. Os querys utilizados utilizam a mesma metodologia utilizada noutros formulários de pesquisa analisados anteriormente.

Uma questão que tem sido analisada ultimamente pelo Departamento de Manutenção e que carece de claras melhorias é o inventário de material a utilizar só pela manutenção de máquinas, equipamentos e edifícios.

Ao clicar em 

surge o formulário que junta e melhora algumas funcionalidades até aqui implementadas (Figura 5-31).

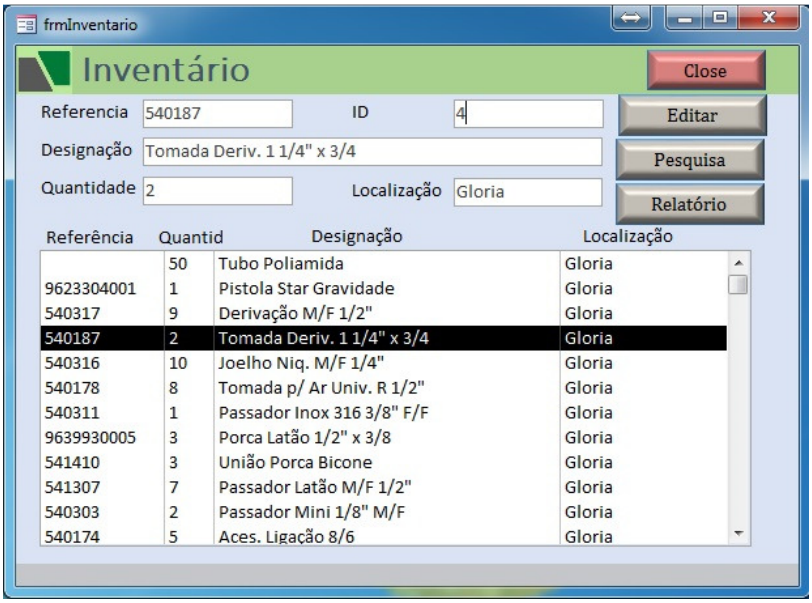


Figura 5-31 – Formulário para gestão do inventário

Não há ligação dos dados da tabela `tblInventarioGeral` com a tabela `tblMaterial`. Nunca foi objetivo desta aplicação fazer gestão de *stocks.* Na parte inferior da figura 5-31 encontra-se uma listagem, recolhida nos armazéns, do material existente. Por vezes, não é possível determinar a referência do material, pelo que o espaço fica em branco para posterior introdução. Todos os campos se encontram bloqueados até que seja clicado sobre o botão 

onde surge outro formulário, onde é possível editar o registo. Na figura 5-32 verifica-se que a caixa de texto "Referência" não apresenta qualquer valor. Ao introduzir um valor para servir de referência, este fica gravado na tabela `tblInventarioGeral` através do comando `DoCmd.Save` como se pode verificar no código 5-47.

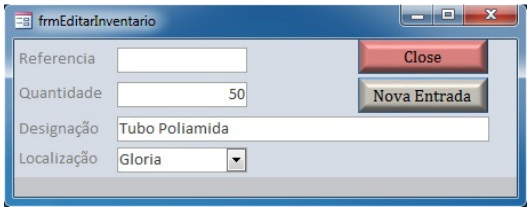


Figura 5-32 – Formulário para editar artigos do inventário

```
1  Private Sub cmdClose_Click()
2  DoCmd.Save acForm, "frmEditarInventario"
3  DoCmd.Close
4  Form_frmInventario.SetFocus
5  DoCmd.Restore
6  DoCmd.MoveSize 1000, 1000
7  End Sub
```

Código 5-47 – Evento "cmdClose_Click" promove a gravação dos valores de todo o formulário na tabela

O formulário 

(*frmMaquinas*) concentra a informação disponível sobre as máquinas da empresa independentemente do estado em que se encontrem.

Figura 5-33 – Formulários acessíveis na listagem de máquinas

As máquinas podem não estar "Em Serviço", mas o seu histórico e plano de manutenção ficam guardados.

Para além de uma pesquisa individual, em que o utilizador coloca um número a pesquisar na caixa de texto "Pesquisa" (Figura 5-34), há o acesso a um formulário de pesquisa da lista completa de máquinas presentes no sistema ("Pesquisa Na Lista"). Usando a estratégia apresentada anteriormente pode-se enviar um e-mail para abrir uma nova referência de máquina ou fazer um pedido de reparação. Estes e-mails também serão gravados para posterior pesquisa caso seja necessário, podendo ser reenviados.

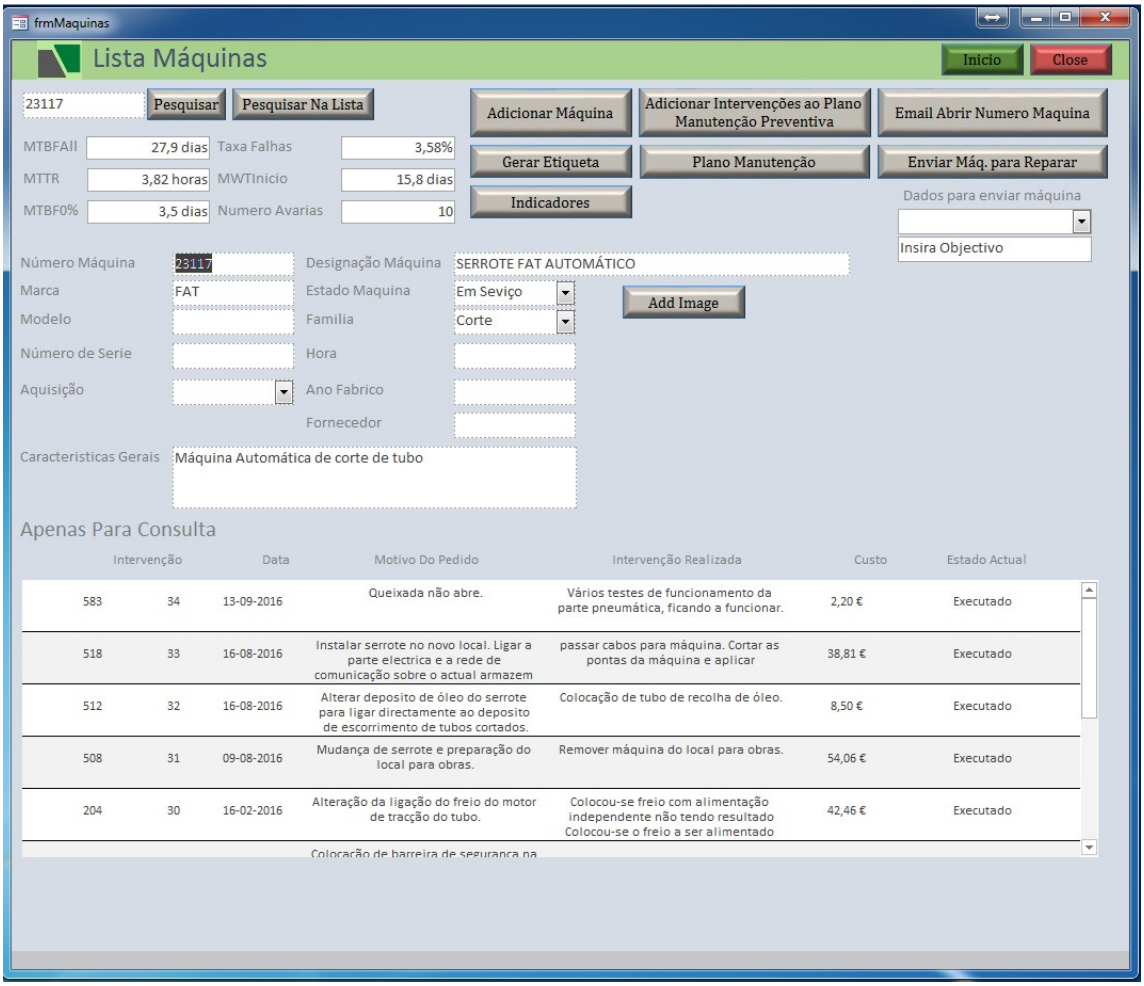


Figura 5-34 – Formulário com informação das Máquinas

Na imagem da figura 5-34 é apresentado o formulário da máquina "23117- SERROTE FAT AUTOMÁTICO". No canto superior direito (Figura 5-35), encontram-se alguns indicadores da máquina como o MTBF, MTTR e o MWT.

O cálculo dos indicadores gerais segue os conceitos apresentados por Pinto (2013) [6].

Foram considerados dois tipos de falha para o indicador MTBF. O MTBFAll em que a falha ocorreu num componente individual da máquina e não impede o seu funcionamento. Serão todas as intervenções executadas em componentes da máquina que não impedem o seu funcionamento. O MTBF0% decorre de falhas que originaram a impossibilidade do funcionamento da máquina.

Figura 5-35 – Quadro indicadores máquina 23117 (1-11-2016)

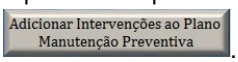
O módulo `IndicadorMaquina` contém as funções, `TempoMedioEntreFalhas`, `TempoMedioDeReparacao`, `TempoMedioEsperaParaInicio` e `NumeroAvarias`.

A função `TempoMedioEntreFalhas (Maquina, Ano)` (Anexo XXI – Indicador Tempo Médio Entre Intervenções) (Código 5-48) contém o argumento `Maquina` que aceita uma *string* com o número da máquina que se pretende analisar e uma *string* `"all"` para consultar todas as máquinas. O argumento `Ano` aceita o valor `0` para analisar apenas uma máquina ou o valor `-1` para obter dados das falhas catastróficas (MTBF0%).

```
1 With rs
  a.   NumeroDeRegistos = .RecordCount
  b.   .MoveFirst
  c.   For i = 1 To NumeroDeRegistos - 1
    i. DataAnterior = .Fields("DataPedido").Value
   ii. .MoveNext
  iii. NovaData = .Fields("DataPedido").Value
   iv. SomaDatas = SomaDatas + (NovaData - DataAnterior)
  d.   Next i

  e.   MTBF = SomaDatas / NumeroDeRegistos
  f.   TempoMedioEntreFalhas = Round(MTBF, 2)
2 End With
```

Código 5-48 – Obtenção de número de registos com base nos critérios e somatório da diferença de datas

Os restantes indicadores apresentados seguem a mesma lógica de obtenção e cálculo. Gerar o *query* para consulta dos dados de acordo com parâmetros específicos. Usando o conjunto de registos gerados passa-se à parte da contagem de acordo com o indicador que se pretende obter.

No formulário "Lista Máquinas" cria-se o plano de manutenção para a máquina. O primeiro passo será adicionar pontos de intervenção no plano clicando no botão 

.

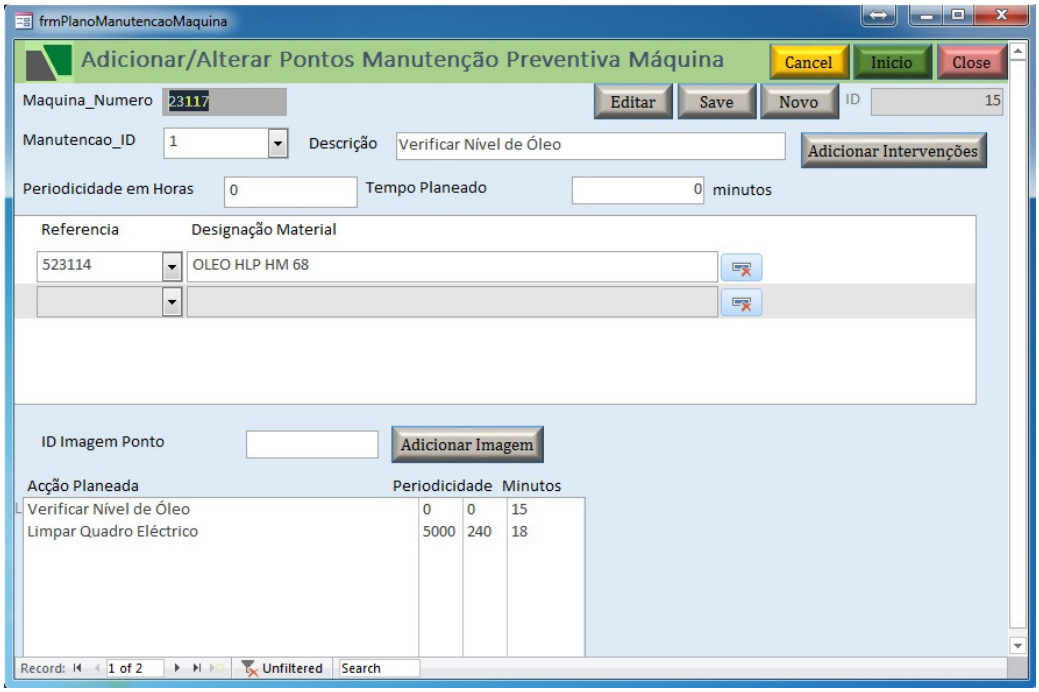


Figura 5-36 – Formulário de adição de pontos de manutenção

Surge o formulário, presente na figura 5-36, com a listagem dos pontos de manutenção já definidos na parte inferior e, na parte superior, encontra-se o primeiro ponto com a identificação da operação, a periodicidade, o tempo planeado para a operação e o material a utilizar.

Cada máquina pode apresentar procedimentos de manutenção distintos, mas a ação que decorre da necessidade de manutenção, apresenta pontos em comum que podem ser definidos ao clicar no botão “Adicionar Intervenções”. Neste formulário faz-se a adição de pontos de manutenção que apresentam uma ação comum. Na figura 5-37 é apresentada uma listagem de “Intervenções a realizar” que são muito genéricas. A título de exemplo, a verificação de nível de óleo varia de máquina para máquina, podendo estar visível ou ser necessário indicar “Cuidados a ter” para proceder à sua verificação.

Figura 5-37 – Listagem de pontos de manutenção**Error! Bookmark not defined.**

Os formulários do Plano de Manutenção para as máquinas utilizam funções de VBA já identificadas e explicadas anteriormente. São utilizados os procedimentos já atrás referidos para apresentar um relatório, onde constam as intervenções a realizar na máquina que fazem parte do plano de manutenção.

Os dados colocados nas tabelas do MySQL do Raspberry Pi® são acedidos ao clicar no botão Dados Recolhidos Na Máquina. Assim como em exemplos anteriores, este evento faz uma consulta à tabela ligada *tblLog* de acordo com o código 5-49.

```
--recolher dados do slave "01" na função modbus "06" e cuja data é superior a
1-08-2016
SELECT tblLog.IDSlave AS Maquina, tblLog.PrimeiroEndereco AS Funcao,
tblLog.MBFunction, Max(tblLog.Dados) AS Dados, Max(tblLog.Data) AS Data
FROM tblLog
GROUP BY tblLog.IDSlave, tblLog.PrimeiroEndereco, tblLog.MBFunction
HAVING (((tblLog.IDSlave)="01") AND ((tblLog.MBFunction)="06") AND
((Max(tblLog.Data))>#8/1/2016#))
ORDER BY tblLog.PrimeiroEndereco;
```

Código 5-49 – Consulta à base de dados para obtenção de informação de funcionamento da máquina

O conjunto de registos irão ser utilizados para preencher o relatório da figura 5-39. Foi criada uma nova tabela (*tblListaFuncoesModbus*) para colocar os endereços utilizados na comunicação MODBUS e os dados que constam nesses endereços.

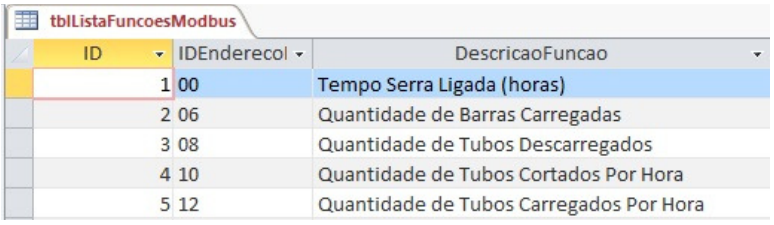


Figura 5-38 – Tabela com a lista de endereços MODBUS e os dados associados

Os dados desta tabela são verificados, utilizando a função *Dlookup* na caixa de texto indicativa do tipo de dados.

Figura 5-39 – Relatório com os dados recolhidos da máquina

5.4 Sumário

Neste capítulo foi descrito todo o funcionamento do servidor MODBUS para a comunicação com o Serrote FAT. Depois de configurada a consola OMRON, com os endereços de memória que pretendemos utilizar, no objeto para transmissão de dados, estes são analisados pelo servidor e reencaminhados para a base de dados *MySQL*. O *software* foi desenvolvido em Python3 sendo o módulo mais relevante aquele que permite enviar código SQL para gravar dados nas tabelas.

Este *software* não foi testado com mais do que uma máquina pelo que o módulo *asyncio* não está completamente ensaiado.

O conjunto de dados recolhidos é enviado para a aplicação de gestão de base de dados Microsoft Access. O acesso aos dados é feito a partir de formulários definidos para cada conjunto de informação que se pretende guardar ou recolher. Há informação sobre todas as intervenções a executar e executadas para cada uma das máquinas presentes no sistema. Para que seja possível obter alguns indicadores de manutenção foram adicionadas várias funcionalidades à aplicação. A recolha de dados dos recursos humanos e a utilização de materiais na execução das intervenções são tratados de forma prioritária. Numa organização onde o número de intervenções corretivas é elevado, a recolha de dados relativos à manutenção facilita a sensibilização da gestão de topo para a necessidade de mudanças. Será sempre necessário analisar o contexto da empresa e em particular o departamento de Produção. Este aspeto foge ao âmbito deste projeto. Caso o equipamento seja pouco utilizado a manutenção corretiva é a escolha mais acertada analisando os custos e a rentabilidade da manutenção ao longo do tempo.

6 Conclusões e Trabalho Futuro

Este capítulo apresenta as conclusões que se podem retirar do projeto realizado e apresenta sugestões para trabalho futuro na plataforma desenvolvida.

6.1 Conclusões

Este trabalho teve como objetivo fundamental a melhoria, e a demonstração das vantagens, no desenvolvimento de uma base de dados, das atividades dentro do Departamento de Manutenção da Miralago S.A.. A Manutenção sempre foi vista como solucionadora dos problemas das máquinas e equipamentos de uma forma geral. A sua ação estava dependente do Departamento de Produção e apenas satisfazia os interesses desta. A eficiência das ações de manutenção era "analisada" com base em registos manuais inseridos, primeiro no *dossier* de máquina e depois no ERP instalado na empresa em 2010.

O panorama da manutenção dentro da empresa nunca foi do agrado do autor. Era projeto do autor procurar melhorias que facilitassem o seu trabalho e ao mesmo tempo demostrassem claramente o que estava a acontecer dentro do departamento. O trabalho deveria ser registado e não apenas o resultado deste. Os dados do trabalho deveriam ser analisados para procurar ineficiências e diminuir o número de intervenções não planeadas.

Neste momento, a aplicação desenvolvida, está a ser utilizada todos os dias dentro da empresa Miralago. Neste aspeto o Sr. Júlio Chio, meu colega no departamento, tem utilizado a aplicação para a gestão diária das ordens de trabalho, a verificação de desvios nos registos, feito pelos técnicos no terreno e solicitar material ao departamento de compras.

Todos os dias se tem verificado que a aplicação facilita o trabalho dentro do departamento e, mais importante, permite obter informação sobre as atividades desenvolvidas de uma forma que facilita o processo de decisão.

O capítulo anterior evidencia isso mesmo com a emissão de relatórios de documentos em Excel sobre o trabalho desenvolvido

Agora é possível apresentar à Administração os tipos de manutenção mais frequentes na empresa. Os indicadores gerais para qualquer Departamento de Manutenção, independente do seu valor real, (Tempo Médio de Reparação e Tempo Médio Entre Falhas) podem também ser apresentados, bem como os custos das intervenções. No entanto, o indicador de disponibilidade das máquinas não foi possível determinar. Para que isso viesse a acontecer, é necessário que a aplicação tenha uma ligação mais próxima com o Sistema Informático da empresa, na medida em que o tempo de espera pela concretização de encomendas (Aguardar Material) não pode ser determinado de forma fidedigna. Não se conseguiu determinar corretamente o Tempo Médio de Espera (MWT); no entanto, o tempo de espera para início da intervenção, no caso de intervenções corretivas, é determinado.

6.2 Melhorias no *Hardware*

A utilização do *hardware* existente foi uma das razões para o sucesso deste projeto. Trata-se de uma forma barata de implementar um sistema de recolha de dados das máquinas sem ser necessário fazer investimentos em sistemas de processamento.

Há outras variáveis da máquina que poderiam ser recolhidas, como o consumo do motor da serra, a recolha de dados de vibração do disco, variação de pressão sobre o tubo a cortar (pinças) ou o caudal de óleo de corte utilizado.

Com o impulso dado por várias multinacionais como Intel, Google, Ericsson, etc., ao *IoT* (*Internet of Things*) e a Indústria 4.0, a recolha de dados nas máquinas apenas pode melhorar. As empresas de distribuição de produtos eletrónicos dedicam-lhe páginas com informação e novos produtos [49].

A recolha dos dados atrás referidos poderia ser executada com algum investimento. As vantagens da maior recolha de dados será um melhor planeamento de intervenções necessárias por forma a manter a máquina em perfeito estado de funcionamento. Mesmo as manutenções corretivas podem passar a ser planeadas.

A utilização do Raspberry Pi foi uma opção lógica pelo preço e facilidade de implementação. Ao expandir a rede será necessário um computador com maior capacidade para gerir as ligações ao seu servidor MySQL. Este computador pode depois ser utilizado numa máquina, juntamente com alguns sensores, para fazer recolha de dados.

6.3 Melhorias no Software

Ao longo do processo de desenvolvimento o autor foi notando algumas limitações do *software* Access para gestão de base de dados. A utilização de *links* para imagens nos campos onde essas são utilizadas, em vez da utilização das imagens dentro da base de dados, limita a possibilidade de utilizar a aplicação por vários utilizadores em locais onde o acesso a uma rede comum é mais limitada. Outra hipótese será a utilização de um servidor remoto, com acesso para os utilizadores da aplicação, com os documentos utilizados (imagens, pdf, …).

Os documentos de manutenção gerados, nomeadamente o relativo à manutenção preventiva, devem ser melhorados na sua apresentação. Para além disso, não foi implementada uma solução que produzisse as Ordens de Trabalho relativas às manutenções preventivas, enquadradas no plano de manutenção anual.

6.4 Trabalhos Futuros

Alargar a recolha de dados a um maior número de máquinas será uma condição para que esta aplicação venha a vingar. Da pesquisa realizada, existem boas soluções de *software* para a manutenção na indústria. Em algumas é possível recolher dados da instalação e das máquinas em tempo real, juntando esses dados ao plano de manutenção, pode ser utilizado para correção e/ou alteração de Planos de Manutenção. Com as limitações de tesouraria das empresas, normalmente estes ficam para o fim da lista de compras. Na Miralago S.A. se juntarmos o facto de as máquinas, na sua maioria, pertencerem a uma geração anterior a computadores, a justificação do investimento será ainda mais difícil.

A recolha de um maior número de variáveis de funcionamento da máquina também será um aspeto a implementar. Fazendo uso de sensores tradicionais ligados aos PLCs existentes ou utilizando um mini-PC com sensores remotos, as hipóteses são muitas e variadas mesmo utilizando o MODBUS TCP ou Ethernet.

Bibliografia

[1] Banco de Portugal, “Análise Setorial da Industria Metalomecânica - estudo Central de Balanços,” Imprensa Nacional Casa da Moeda, Lisboa, 2015.

[2] Governo de Portugal, “Decreto-Lei nº 372/2007 de 6 de Novembro,” *Diário da República,* vol. 1ª Serie, pp. 8080-8084, 2007.

[3] Enterprise Europe Network, “Definição Europeia de PME,” 20 Maio 2003. [Online]. Available: http://www.enterpriseeuropenetwork.pt/info/polserv/pol%C3%ADticas/Paginas/p1.aspx. [Acedido em 16 Novembro 2016].

[4] Miralago S.A., “Miralago,” 2016. [Online]. Available: http://www.miralago.pt/i/0/pt/company/#company. [Acedido em 16 Novembro 2016].

[5] S. Franklin, “Redefining maintenance - Delivering Reliability,” em *maintenance Engineering Handbook - Seventh Edition*, McGraw-Hill Companies, Inc., 2008, pp. 1.3-1.8.

[6] J. P. Pinto, Manutenção Lean, Lisboa: LIBEL - Edições Técnicas, Lda., 2013.

[7] Capterra, "Top Maintenance management Software products," 2016. [Online]. Available: http://www.capterra.com/maintenance-management-software/. [Accessed 16 Setembro 2016].

[8] ManwinWin Software, “ManWinWin Software - Visão Geral,” Navaltik Mnagement, 2013. [Online]. Available: http://www.manwinwin.com/PT/manwinwin/manwinwin.htm. [Acedido em 01 Dezembro 2016].

[9] CentralGest, “Gestão da Manutenção,” CentralGest - Produção de Software, S.A., 2016. [Online]. Available: http://www.centralgest.com/software/manutencao/gestao-da-manutencao. [Acedido em 16 Novembro 2016].

[10] PHC - Software, S.A., “PHC CS gestão de manutenção manufactor,” PHC - Software, S.A., 2016. [Online]. Available: http://www.phc.pt/portal/programs/estview.aspx?ref=ManGestaoManut#.WEMQZbKLRpg. [Acedido em 01 Dezembro 2016].

[11] Engecompany Engenharia de Sistemas, “Demonstrativo Industrial,” Engecompany Engenharia de Sistemas, [Online]. Available: http://engeman.com.br/pt-br/demonstrativos/industrial/. [Acedido em 01 Dezembro 2016].

[12] Infor, Inc., “The Power of Infor EAM,” Infor, Inc., 2016. [Online]. Available: http://www.infor.com/solutions/eam/. [Acedido em 01 Dezembro 2016].

[13] IBM, “Maximo Asset Management,” IBM, [Online]. Available: http://www-03.ibm.com/software/products/pt/maximoassetmanagement. [Acedido em 01 Dezembro 2016].

[14] Websites, IT Works, “IMS - Innovate Maintenance Systems - Maintenance Pro,” Innovative Maintenance Systems, [Online]. Available: http://www.mtcpro.com/maintenance-pro.htm. [Acedido em 01 Dezembro 2016].

[15] Técnica Aplicada Internacional S.A. de C.V., “MP Software - Folhetos,” [Online]. Available: http://www.mpsoftware.com.mx/software_manutencao/mp_cmms.html. [Acedido em 12 Fevereiro 2016].

[16] Manager Plus, “ManagerPlus - Desktop,” Manager Plus, 2016. [Online]. Available: http://www.managerplus.com/maintenance-software/desktop. [Acedido em 01 Dezembro 2016].

[17] SMGlobal Inc., “Fastmaint CMMS Software - maintenance Management Software,” SMGlobal Inc., 2016. [Online]. Available: http://www.smglobal.com/fastmaint/. [Acedido em 01 Dezembro 2016].

[18] Sourceforge, “Firebird,” 02 12 2016. [Online]. Available: https://sourceforge.net/projects/firebird/. [Acedido em 02 Dezembro 2016].

[19] Rede Industrial, “Apresentação SIGMA,” 2016. [Online]. Available: http://www.centralsigma.com.br/index.php?option=com_content&view=article&id=265&Itemid=316. [Acedido em 02 Dezembro 2016].

[20] CWorks Systems Inc., “The Easy Route to Maintenance Reports CMMS,” CWoks, 2016. [Online]. Available: http://www.cworkssystems.com/cworks-easy.html. [Acedido em 02 Dezembro 2016].

[21] Glintt, “gmac.2,” Glintt, 2013. [Online]. Available: http://sites.glintt.com/glinttconsulting/gmac2/. [Acedido em 01 Dezembro 2016].

[22] M. Alexander e D. Kusleika, Access 2016 Bible - The Comprehensive Tutorial Resource, Indianapolis, Indiana: John Wiley & Sons, Inc., 2016.

[23] Omron Corporation, *NB-Series Programmable Terminals - Setup Manual,* 2013.

[24] Modbus.org, “Modbus Technical Resources,” Modbus Organization, Inc., 2016. [Online]. Available: http://www.modbus.org/tech.php. [Acedido em 14 Fevereiro 2016].

[25] Raspberry Pi Foundation, “Raspberry Pi - Teach, Learn, and Make with Raspberry Pi,” 2016. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Acedido em 02 Janeiro 2016].

[26] Oracle Corporation, “MySQL,” Oracle Corporation, 2016. [Online]. Available: http://www.mysql.com/. [Acedido em 23 Janeiro 2016].

[27] Microsoft Cooperation, “Microsoft Access 2016 Runtime,” Microsoft Cooperation, 2016. [Online]. Available: https://www.microsoft.com/pt-PT/download/details.aspx?id=50040. [Acedido em 04 Fevereiro 2016].

[28] Omron, *Getting Started Guide para SYSMAC CP1L.*

[29] ModBus.Org, “Modbus Protocol,” Modbus Organization, Inc., 2016. [Online]. Available: http://www.modbus.org/specs.php. [Acedido em 02 Fevereiro 2016].

[30] ModBus. Org, *ModBus Application protocol Specification V1.1b3,* 2012.

[31] ModBus.Org, *ModBus Messaging on TCP/IP - Implementation Guide,* 2002.

[32] Wikipedia, “Microsoft Access,” 24 Setembro 2015. [Online]. Available: https://pt.wikipedia.org/wiki/Microsoft_Access. [Acedido em 02 Novembro 2015].

[33] Wikipédia, “Visual Basic for Applications,” 23 Agosto 2015. [Online]. Available: https://pt.wikipedia.org/wiki/Visual_Basic_for_Applications. [Acedido em 02 Janeiro 2016].

[34] Lucidchart, “Lucidchart,” Lucid Software Inc., 2016. [Online]. Available: https://www.lucidchart.com/. [Acedido em 10 Novembro 2016].

[35] Stephen, “Getting a python script to run in the background (as service) on boot,” 21 Julho 2013. [Online]. Available: http://blog.scphillips.com/posts/2013/07/getting-a-python-script-to-run-in-the-background-as-a-service-on-boot/. [Acedido em 12 Fevereiro 2016].

[36] Wikipédia, “Daemon,” 7 Setembro 2016. [Online]. Available: https://pt.wikipedia.org/wiki/Daemon_(computa%C3%A7%C3%A3o). [Acedido em 16 Novembro 2016].

[37] StackExchange, “Unix & Linux - What is the meaning/purpose of *.pid files in /var/run,” 12 Setembro 2015. [Online]. Available: http://unix.stackexchange.com/questions/229304/what-is-the-meaning-purpose-of-pid-files-in-var-run. [Acedido em 02 Novembro 2016].

[38] Python Software Foundation, “7.1. struct - Interpret bytes as packed binary data,” 2016. [Online]. Available: https://docs.python.org/3.5/library/struct.html. [Acedido em 14 Fevereiro 2016].

[39] Microsoft Cooperation, "Office - Dev Center - OlItemType Enumeration (Outlook)," Microsoft Cooperation, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/office/ff869291.aspx. [Accessed 01 Novembro 2016].

[40] Microsoft Cooperation, “Microsoft - DLookup Function,” Microsoft Cooperation, [Online]. Available: https://support.office.com/en-us/article/DLookup-Function-8896cb03-e31f-45d1-86db-bed10dca5937. [Acedido em 01 Novembro 2016].

[41] Microsoft Cooperation, “Microsoft - Developer Network - Form.Open Event,” Microsoft Cooperation, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb214851(v=office.12).aspx. [Acedido em 01 Novembro 2016].

[42] Microsoft Cooperation, “Office Dev center - Chart Object (Excel),” Microsoft Cooperation, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/office/ff194426.aspx. [Acedido em 27 Novembro 2016].

[43] Microsoft Cooperation, “Microsoft Development Network TRANSFORM Statement (Microsoft Access SQL),” Microsoft Cooperation, [Online]. Available: https://msdn.microsoft.com/en-us/library/bb177905(v=office.12).aspx. [Acedido em 27 Novembro 2016].

[44] Microsoft Cooperation, “Office - Dev Center - Application.FileDialog Property (Access),” Microsoft Cooperation, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/office/ff196794.aspx. [Acedido em 01 Novembro 2016].

[45] Microsoft Cooperation, “Office - Dev Center - Attachments.Add.Method (Outlook),” Microsoft Cooperation, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/office/ff869553.aspx. [Acedido em 01 Novembro 2016].

[46] The Internet Engineering Task Force (IETF(r)), “Content-ID and Message-ID Uniform Resorce Locators,” 08 1998. [Online]. Available: https://tools.ietf.org/html/rfc2392. [Acedido em 12 Novembro 2016].

[47] Microsoft Cooperation, “Microsoft - Developer Network - About CDO for Windows 2000,” Microsoft Cooperation, 08 06 2004. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms526577(v=exchg.10).aspx. [Acedido em 01 Novembro 2016].

[48] Microsoft Cooperation, “Microsoft - Developer Network - http://schemas.microsoft.com/cdo/configuration,” Microsoft Cooperation, 08 Junho 2004. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms526318(v=exchg.10).aspx. [Acedido em 12 Novembro 2016].

[49] Amidata S.A., “Internet das Coisas,” 2016. [Online]. Available: http://pt.rs-online.com/web/generalDisplay.html?id=i/iot-internet-of-things&redirect-relevancy-data=636F3D3126696E3D4931384E5461786F6E6F6D794272616E64266C753D7074266D6D3D6D61746368616C6C26706D3D5E5B5C707B4C7D2D2F5D2B2426706F3D313626736E3D592673723D5265646. [Acedido em 01 Dezembro 2016].

[50] Omron Industrial Automation, “Programmable Terminals NB Series,” 2016. [Online]. Available: https://www.ia.omron.com/products/family/3110/specification.html. [Acedido em 20 Setembro 2016].

[51] R. Assis, Apoio à Decisão em Manutenção na Gestão de Ativos Físicos - 2ª Edição, Lisboa: LIDEL - Edições Técnicas, Lda., 2014.

[52] RS Amidata, “Raspberry Pi 3 Model B,” 2016. [Online]. Available: http://pt.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/8968660/.

[53] Libelium Comunicationes Distribuidas S.L., “libelium,” 2016. [Online]. Available: http://www.libelium.com/. [Acedido em 01 12 2016].

[54] Microsoft Cooperation, “How to send HTML formated mail using CDO for Windows 2000 and a remote SMTP service,” 19 Junho 2014. [Online]. Available: https://support.microsoft.com/en-us/kb/286431. [Acedido em 12 Novembro 2016].

[55] Microsoft Corporation, “Office Dev Center - Understanding Named Arguments and Optional Arguments,” Microsoft Cooperation, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/office/gg251503.aspx. [Acedido em 01 Novembro 2016].

[56] M. Garrels, “Bash guide for Beginners,” 27 Dezembro 2008. [Online]. Available: http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html. [Acedido em 01 Novembro 2016].

[57] ModMyPi, “Tutorial:How to run a program from boot,” ModMyPi LTD, 30 Outubro 2013. [Online]. Available: https://www.modmypi.com/blog/tutorial-how-to-run-a-program-from-boot. [Acedido em 15 Agosto 2016].

[58] Oracle Corporation, “MySQL 5.5 Reference Manual,” Oracle Corporation, 2016. [Online]. Available: http://dev.mysql.com/doc/refman/5.5/en/. [Acedido em 14 Janeiro 2016].

[59] Python Software Foundation, “Python,” Python Software Foundation, 2016. [Online]. Available: https://www.python.org/. [Acedido em 26 Fevereiro 2016].

[60] “Raspberry Pi - Run Program at start-up,” Martin O'Hanlon, 10 Junho 2012. [Online]. Available: http://www.stuffaboutcode.com/2012/06/raspberry-pi-run-program-at-start-up.html. [Acedido em 02 Janeiro 2016].

[61] Youtube.com, “Keynote David Beazley - Topics of Interest (Python Asyncio),” pythonbrasil, 21 Dezembro 2015. [Online]. Available: https://youtu.be/ZzfHjytDceU. [Acedido em 02 Janeiro 2016].

[62] Youtube.com, “David Beazley: Python Concurrency from the Ground Up:LIVE! - Pycon 2015,” Pycon 2015, 11 Abril 2015. [Online]. Available: https://www.youtube.com/watch?v=MCs5OvhV9S4. [Acedido em 02 Janeiro 2016].

Anexos

Anexo I – Código para o MODBUS Server (modbusserver.sh)

```
#!/bin/sh

### BEGIN INIT INFO
#Provides:           modbusserver
#Required-Start:     $remote_fs $syslog
#Required-Stop:      $remote_fs $syslog
#Default_Start:      2 3 4 5
#Default-Stop:       0 1 6
#Short-Description:  Servidor para ModBus TCP
#Description:        Servidor ModBus para equipamentos na Miralago S.A.
  ### END INIT INFO

  DIR=/home/pi/
  DAEMON=$DIR/modbusserver
  DAEMON_NAME=modbusserver

  DAEMON_OPTS=""

  DAEMON_USER=root

  PIDFILE=/var/run/$DAEMON_NAME.pid

  . /lib/lsb/init-functions

  do_start(){
  log_daemon_msg "Starting system $DAEMON_NAME deamon"
  start-stop-daemon --start --background --pidfile $PIDFILE --make-pidfile --
 user $DAEMON_USER --chuid $DAEMON_USER --startas $DAEMON -- $DAEMON_OPTS
  log_end_msg $?
  }
  do_stop () {
  log_daemon_msg "Stopping system $DAEMON_NAME daemon"
  start-stop-daemon --stop --pidfile $PIDFILE   --retry 10
  log_end_msg $?
  }

  case "$1" in
  start|stop)
   a do_${1}
   b ;;
  restart|reload|force-reload)
   a do_stop
   b do_start
   c ;;
  status)
   a status_of_proc "$DAEMON_NAME" "$DAEMON" && exit 0 || exit $?
   b ;;
  *)
   a echo "Usage: /etc/init.d/$DAEMON_NAME {start|stop|restart|status}"
   b exit 1
   c ;;

  esac
  exit 0
```

Código A-1 – Código do ficheiro colocado para fazer o arranque do programa

Anexo II – Servidor ModBus em Python

```
1   #!/usr/local/bin/python3
2   # Ficheiro server.py servidor Modbus TCP na porta 502

3   from socket import *
4   import asyncio, struct,time
5   import pymysql as sql
6   # Endereco do MySQL Server
7   EnderecoServidor = '127.0.0.1'
8   #
9   UnidadeCliente = 1

10  # Loop necessario para utilizar o modulo asyncio do python3
11  loop = asyncio.get_event_loop()

12  # Criar o socket server para receber ligacoes
13  async def modbus_Server(address):
    a   sock = socket(AF_INET, SOCK_STREAM)
    b   sock.setsockopt(SOL_SOCKET,SO_REUSEADDR,1)
    c   sock.bind(address)
    d   sock.listen(5)
    e   sock.setblocking(False)
    f   while True:
          i  client, addr = await loop.sock_accept(sock)
         ii  print ("Connection with:", addr)
        iii  loop.create_task(modbus_handler(client))

14  # Analisar o pacote recebido por forma a verificar se corresponde a um
    pacote ModBusTCP
15  async def modbus_handler(client):
16
    a   with client:
          i  while True:
               1  data = await loop.sock_recv(client, 10000)
               2  #print("Dados Recebidos",data)
               3  if len(data) == 12:
               4  ModbusRequest(data)
               5  #sendDadosDB(data)
               6  if not data:
                    a  client.close()
                    b  break
               7  print(client)
               8  await loop.sock_sendall(client,data)
    b   print("Connection closed")
```

```
1 # Separar o pacote de dados nos seus componentes conforme documentação técnica
2 def ModbusRequest(data):

3 try:
  a #print(data)
  b #print (len(data))
  c dados = struct.unpack('!HHHBBHH', data)
  d TransactionID = dados[0]
  e Protocolo = dados[1]
  f MensagemLengh = dados[2]
  g UnitIdentifier = dados[3]
  h MBAP = dados[0:4]
  i #verificar se a mensagem é válida
  j reply = MODBUS_PDU_Checking(MBAP)
  k #se a mensagem estiver OK então processa o resto dos dados
  l if reply == 1:

     i #print(''.join(map(str,dados)))
    ii #print (MBAP)
   iii #print(UnitIdentifier)
    iv FunctionCode = dados[4]
     v EnderecoFirstRegister = dados[5]

    vi sql_log(''.join(map(str, dados)), ''.join(map(str, MBAP)),
       UnitIdentifier,FunctionCode, EnderecoFirstRegister,dados[6])

   vii # verificar função solicitada
  viii #mensagem de leitura
    ix if FunctionCode == 3:
        1 NumeroRegistos = dados[6]
        2 mensagem_leitura = {'TransactionID': TransactionID,
                  'MensagemLengh': MensagemLengh,
                  'UnitIdentifier': UnitIdentifier,
                  'FunctionCode': FunctionCode,
                  'EnderecoFirstRegister': EnderecoFirstRegister,
                  'NumeroRegistos': NumeroRegistos}
        3 MODBUS_SERVICE_Processing(mensagem_leitura)
        4 # mensagam_recebida = dict(dados)

     x #mensagem de escrita
    xi elif FunctionCode == 6:
        5 valor = dados[6]
        6 mensagem_escrita = {'TransactionID': TransactionID,
                  'MensagemLengh': MensagemLengh,
                  'UnitIdentifier': UnitIdentifier,
                  'FunctionCode': FunctionCode,
                  'EnderecoFirstRegister': EnderecoFirstRegister,
                  'Dados': valor}
        7 MODBUS_SERVICE_Processing(mensagem_escrita)
   xii else:
        1 print('Função não suportada')

  m elif reply ==0:
     i  print ('Erro na mensagem')

4 except struct.error as error:
  a print(error)
```

```
1   #verificar se MBAP é zero como indicado na documentação técnica
2   def MODBUS_PDU_Checking (MBAP):
      a   #Verificar se o protocolo está correctamente identificado
      b   if MBAP[1] == 0:
      c   return 1
      d   else:
      e   return 0

3   #processamento da mensagem
4   def MODBUS_SERVICE_Processing(msg):
      f   print ('Função : ' , msg['FunctionCode'])
      g   if msg['UnitIdentifier'] == 1:
           i  maquinaCliente = '23117'
          ii  if msg['FunctionCode'] == 6:
                1  print('Dados: ' ,msg['Dados'])
                2  print ('Endereço registo: ', msg['EnderecoFirstRegister'])
                3  if msg['EnderecoFirstRegister'] == 0:
                     a  #O primeiro endereco trata da horas de trabalho do motor
                     b  Horas_Motor(msg['Dados'],maquinaCliente)
                4  elif msg['EnderecoFirstRegister'] == 4:
                     a  Tempo_Automatico(msg['Dados'],maquinaCliente)

         iii  elif msg['FunctionCode'] == 3:
                1  print('Número de registos solicitados: ',
                   msg['NumeroRegistos'])
          iv  #print('A mensagem é ', msg)
           v  msg.clear()
      h   else:
          i   print('Unidade Desconhecida')

5   #registo de tempo de horas motor
6   def Horas_Motor(dados,maquina):
      a   print('Horas Funcionamento motor %s na máquina %s' %(dados,maquina))
      b   #sql_dados(dados)

7   def Tempo_Automatico(dados, maquina):
      a   print('Tempo Automatico é de %s na máquina %s' %(dados,maquina))

8   # envio de dados da mesnagem ModBus na tabela tbllog
9   def sql_log(msg,MBAP,Slave,Function, FirstRegister, Data):
      a   db = sql.connect(host= EnderecoServidor, user='seabra',
          passwd='junior45', db='miralago')
      b    = db.cursor()
      c   #print(msg ,MBAP, Slave, Function, FirstRegister, Data)

      d   try:
           i  cur.execute ("INSERT INTO tblLog
              (Mensagem,MBAP,IDSlave,MBFunction,PrimeiroEndereco,Dados) VALUES"
          1   " ( '{:0>7}', '{:0>4}' ,'{:0>2}',
              '{:0>2}','{:0>2}','{:0>10}');".format(msg,MBAP,Slave,Function,First
              Register,Data))
          ii  print (cur._last_executed)
         iii  #print('Dados enviados correctamente')
          iv  db.commit()

      e   except cur.DatabaseError as Dberror:
           i  print (Dberror)
          ii  print(cur._last_executed)
      f   #print('Erro no envio de dados')
      g   except:
           i  (cur._last_executed)
          ii  print('Erro no envio de dados')

      h   db.close()

10  # Arranque da task modbus_server e escutar na porta 502
11  loop.create_task(modbus_Server(('',502)))
12  loop.run_forever()
```

Código A-2 – Servidor MODBUS

Anexo III – Descrição do MBAP Header

Campos	Comprimento	Descrição	Cliente	Servidor
Transaction Identifier	2 bytes	Identificação da transação pedido/resposta do MODBUS	Inicializado pelo cliente	Recopiado pelo servidor do pedido recebido
Protocol Identifier	2 bytes	0 = protocolo MODBUS	Inicializado pelo cliente	Recopiado pelo servidor do pedido recebido
Length	2 bytes	Número de bytes seguintes	Inicializado pelo cliente (pedido)	Inicializado pelo servidor (resposta)
Unit Identifier	1 byte	Identificação de escravo remoto ligado numa linha série ou outros barramentos	Inicializado pelo cliente	Recopiado pelo servidor do pedido recebido

A header tem 7 bytes de comprimento:

- *Transaction Identifier* – É utilizado para emparelhamento de transação, o servidor MODBUS copia em resposta estes bytes do pedido.
- Protocol Identifier – É utilizado para multiplexagem intra-sistemas. O protocolo MODBUS é identificado como 0 (zero).
- *Length* – O campo *length* é a contagem de bytes dos campos seguintes da mensagem, inclui o Unit Identifier e os campos de dados.
- Unit Identifier – Este campo é utilizado para roteamento intra-sistemas. É tipicamente utilizado para comunicar com linhas escravas série MODBUS ou MODBUS+ através de uma gateway entre a rede Ethernet TCP-IP e a linha série MODBUS. Este campo é iniciado pelo cliente MODBUS no pedido e deve ser devolvido com o mesmo valor na resposta do servidor.

Anexo IV – Formulários da Aplicação

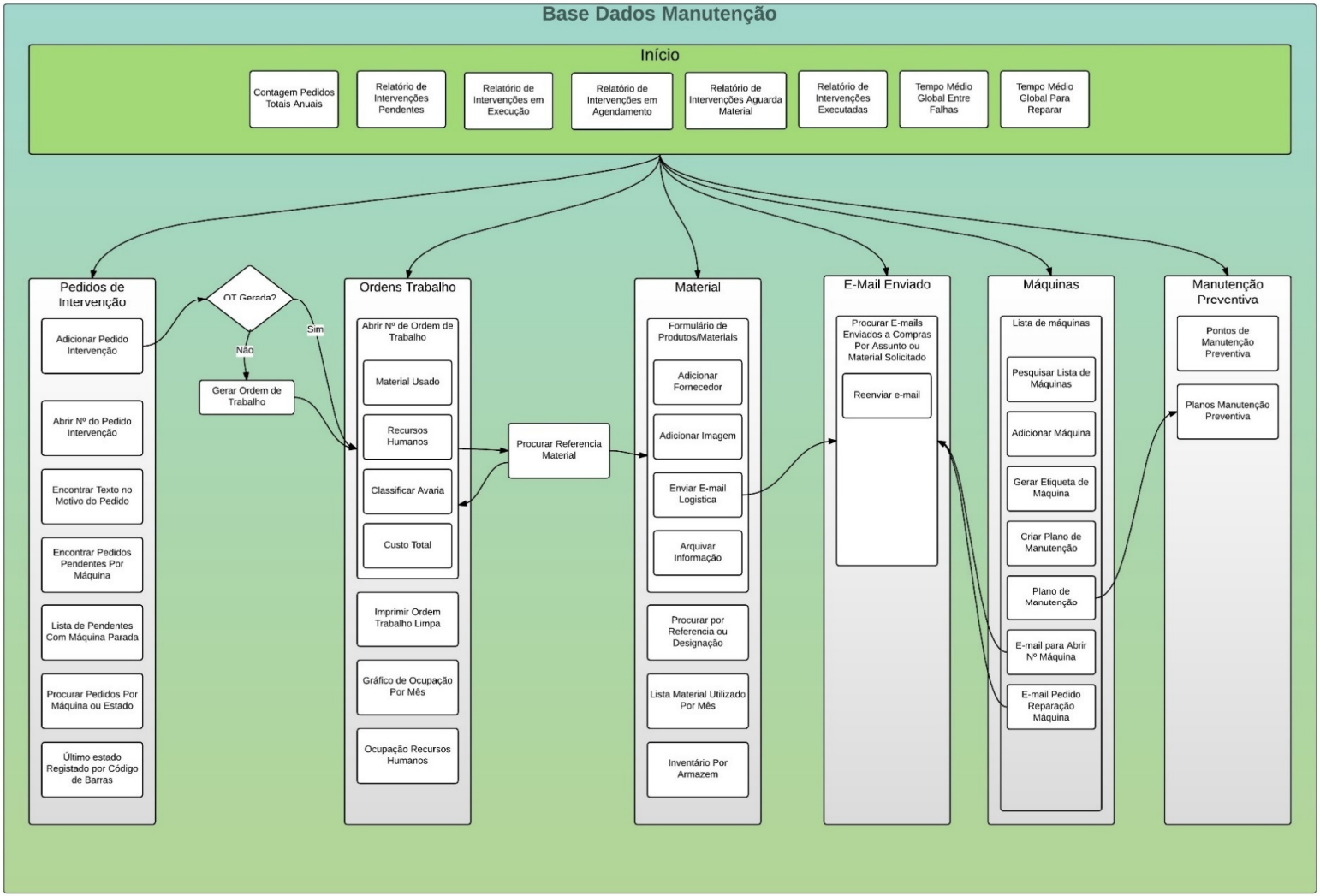
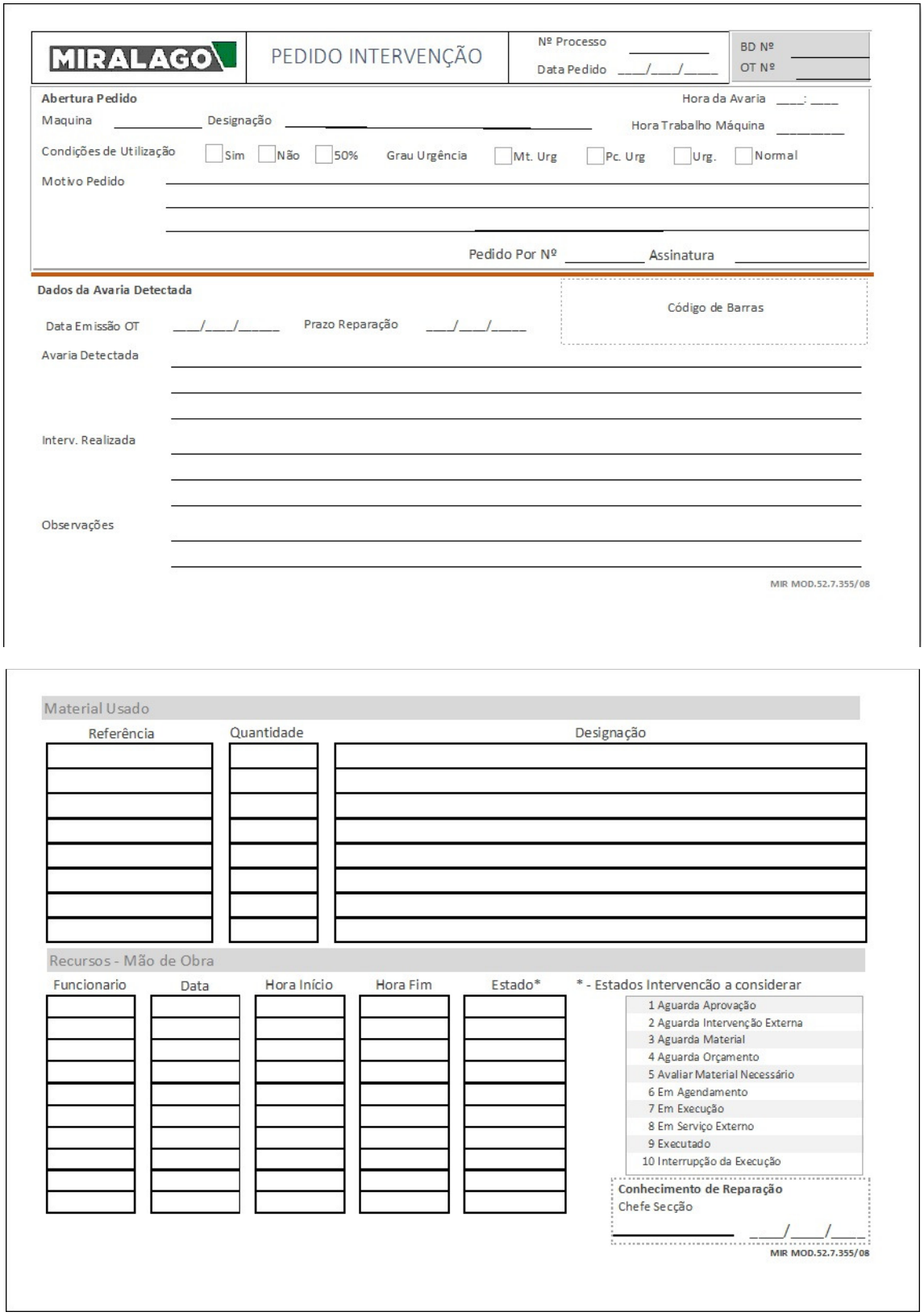


Figura A-1 – Organização dos formulários da aplicação de Gestão de Manutenção

Anexo V – Impresso de Registo das Intervenções da Manutenção



MIRALAGO

PEDIDO INTERVENÇÃO

Nº Processo _____
Data Pedido ___/___/____

BD Nº _____
OT Nº _____

Abertura Pedido

Hora da Avaria ___:___

Maquina _____ Designação _____ Hora Trabalho Máquina _____

Condições de Utilização ☐ Sim ☐ Não ☐ 50% Grau Urgência ☐ Mt. Urg ☐ Pc. Urg ☐ Urg. ☐ Normal

Motivo Pedido _____

Pedido Por Nº _____ Assinatura _____

Dados da Avaria Detectada

Código de Barras

Data Emissão OT ___/___/_____ Prazo Reparação ___/___/____

Avaria Detectada _____

Interv. Realizada _____

Observações _____

MIR MOD.52.7.355/08

Material Usado

Referência	Quantidade	Designação

Recursos - Mão de Obra

Funcionario	Data	Hora Início	Hora Fim	Estado*

* - Estados Intervencão a considerar

1 Aguarda Aprovação
2 Aguarda Intervenção Externa
3 Aguarda Material
4 Aguarda Orçamento
5 Avaliar Material Necessário
6 Em Agendamento
7 Em Execução
8 Em Serviço Externo
9 Executado
10 Interrupção da Execução

Conhecimento de Reparação
Chefe Secção
_____ ___/___/____

MIR MOD.52.7.355/08

Figura A-2 – Impresso "Pedido de Intervenção"

Anexo VI – VBA de verificação de número de intervenção

```
1 Public Function EncontrarRegistoIntervencao(strMaquina As String, intIntervencao
  As Integer) As Boolean

2 Dim strTask As String
3 Dim db As DAO.Database
4 Dim rs As DAO.Recordset
5 Dim i As Integer

6 On Error GoTo errorHandler

7 If IsNull(strMaquina) Or IsNull(intIntervencao) Then

    a.      'A função não recebeu argumentos
    b.      EncontrarRegistoIntervencao = 1
    c.      Exit Function
8 Else
    a.      'Se a função recebeu argumentos verifica se são válidos na base de dados
      com 5 dígitos
    b.      If Len(strMaquina) = 5 Then

      i. 'No caso se a máquina ser válida procura a intervencao respectiva
     ii. strTask = "SELECT tblDadosIntervencao.Maquina,
         tblDadosIntervencao.IntervencaoSistema " _
    iii. & "FROM tblDadosIntervencao " _
     iv. & "WHERE (((tblDadosIntervencao.Maquina)='" & strMaquina & "') AND
         ((tblDadosIntervencao.IntervencaoSistema)=" & intIntervencao & "));"
      v. Set db = CurrentDb
     vi. Set rs = db.OpenRecordset(strTask)
    vii. With rs
              a.  i = 0
              b.  Do Until .EOF
                i. i = i + 1
              c.  .MoveNext
              d.  Loop
   viii. End With

     ix. If i = 0 Then
      x. 'Não existe registo dessa intervenção pode ser aberta
     xi. EncontrarRegistoIntervencao = 0
    xii. Exit Function

   xiii. Else
    xiv. EncontrarRegistoIntervencao = 1
     xv. End If

    xvi. rs.Close
   xvii. Set rs = Nothing
  xviii. db.Close

    c.      Else
      i. 'Se a máquina não for válida pede para executar novamente
     ii. MsgBox "A máquina não é válida. Corrija o número de máquina"
    iii. EncontrarRegistoIntervencao = 1
     iv. Exit Function
    d.      End If
9 End If
10   Exit Function

11   errorHandler:
12   MsgBox "Ocorreu um erro a executar a pesquisa de dados. Tente novamente"

13   End Function
```

Código A-3 – Código completo para verificação da existência de relação entre máquina e intervenção

Anexo VII – Adição de Novas Ordens de Trabalho

```
1 Private Sub cmdGerarOT_Click()

2 'Em caso de erro executar rotina
3 On Error GoTo Err_execute

4 Dim respostaInformacao As Integer
5 Dim db As DAO.Database
6 Dim LastOrdemTrabalho As Long
7 Dim confirmarOT As String


8 Set db = CurrentDb

9 LastOrdemTrabalho = DMax("ID_OT", "tblOrdensTrabalho") + 1

10   respostaInformacao = MsgBox("Dados a carregar" & vbCrLf & "Máquina " &
  Me.Maquina & _
          1.      vbCrLf & "Intervenção N° " & Me.IntervencaoSistema.Value & vbCrLf
            & _
          2.      "Ordem de Trabalho a gerar N° " & LastOrdemTrabalho, vbOKCancel,
            "Carregar Dados")

11   'Em caso de carregar OK
12   If respostaInformacao = 1 Then

    a.      Me.Refresh
    b.      'Adicionar dados á tabela de OTs
    c.      db.Execute "INSERT INTO tblOrdensTrabalho (ID_Intervencao )SELECT
      tblDadosIntervencao.ID FROM tblDadosIntervencao LEFT JOIN tblOrdensTrabalho ON
      tblDadosIntervencao.ID = tblOrdensTrabalho.ID_Intervencao WHERE
      (((tblDadosIntervencao.ID)= " & Me.ID & "));", dbFailOnError

    d.      confirmarOT = BaseDeDados.FindOneOT(Str(LastOrdemTrabalho))

    e.      If confirmarOT = "1" Then
       i. MsgBox "Ordem de trabalho não foi criada."
      ii. Exit Sub

    f.      ElseIf confirmarOT = "OK" Then
       i. MsgBox "Introdução bem sucedida"
      ii. 'Me.txtFind.SetFocus
     iii. Me.AnaliseManutencao.SetFocus
      iv. Me.cmdGerarOT.Visible = False
       v. Me.cmdAbrirOT.Visible = True

    g.      End If

13   'Em caso de carregar cancel
14   ElseIf respostaInformacao = 2 Then

    a.      MsgBox "Introdução de dados cancelada."

15   End If

16   db.Close

17   Exit Sub

18   Err_execute:
19   MsgBox ("Erro a inserir dados ou dados já inseridos")
20   End Sub
```

Código A-4 – Criação de nova OT com base no pedido de intervenção.

Anexo VIII – Verificação da existência de registo de OT

```
1 Public Function FindOneOT(strFind As String) As String
2 Dim strTask As String
3 Dim db As DAO.Database
4 Dim rs As DAO.Recordset
5 Dim i As Integer

6 If IsNull(strFind) Or strFind = "" Then
    a.      'MsgBox "Introduza um número intervenção", vbOKOnly, "Necessário
      preencher"
    b.      FindOneOT = "0"
    c.      Exit Function

7 Else
    a.      strTask = "SELECT * FROM tblOrdensTrabalho WHERE (ID_OT=" & strFind &
      ")"

    b.      Set db = CurrentDb
    c.      Set rs = db.OpenRecordset(strTask)

    d.      With rs
    e.      i = 0
    f.      Do Until .EOF
      i. i = i + 1
     ii. .MoveNext
    g.      Loop

    h.      End With

    i.      If i = 0 Then
      i. 'MsgBox "Não existe registo dessa intervenção"
     ii. FindOneOT = "1"
    iii. Exit Function

    j.      Else
      i. FindOneOT = "OK"
    k.      End If

8 rs.Close
9 Set rs = Nothing
10   db.Close

11   End If

12   End Function
```

Código A-5 – Função para encontrar número da OT na tabela tblOrdensTrabalho

Anexo IX – Escrever LogFile em formato .txt

```
1 Public Sub WriteLog(message As String)
2 Dim fileName As String, strPath As String
3 Dim txtStream As TextStream
4 Dim FSO As New FileSystemObject

5 strPath = CurrentProject.Path & "\LogFile"

6 If Not Dir(strPath, vbDirectory) <> vbNullString Then
    a.      FSO.CreateFolder (strPath)
7 End If

8 fileName = strPath & "\LogFile" & Year(Date) & Month(Date) & Day(Date) & ".txt"

9 Set txtStream = FSO.OpenTextFile(fileName, ForAppending, True,
  TristateUseDefault)


10   message = Now() & ": " & message

11   txtStream.WriteLine message
12   txtStream.Close

13   Set txtStream = Nothing

14   End Sub
```

Código A-6 – Criar diretoria e ficheiro de registo de eventos nos pedidos de intervenção

Anexo X – Procura de Ordens de Trabalho

```
1 Private Sub cmdProcurarOT_Click()
2 Dim strResposta As String, strTask As String, strNumeroOT As String

3 On Error GoTo errorHandler

4 If IsNull(Me.txtProcurarOT) Or Me.txtProcurarOT = " " Then
    a.      MsgBox "Deve introduzir dados"
5 Else
    a.      strNumeroOT = Me.txtProcurarOT

    b.      strResposta = BaseDeDados.FindOneOT(strNumeroOT)

    c.      Select Case strResposta

    d.      Case "1"
      i. MsgBox "Não existe registo dessa Ordem de Trabalho"
     ii. Me.txtProcurarOT.SetFocus
    iii. Me.txtProcurarOT = ""
    e.      Case "OK"
      i. strTask = "SELECT * FROM tblOrdensTrabalho WHERE (ID_OT=" & strNumeroOT &
         ")"

     ii. DoCmd.Minimize
    iii. DoCmd.OpenForm "frmOrdensTrabalho", , , , , , "inicio"
     iv. DoCmd.MoveSize 300, 300

      v. Form_frmOrdensTrabalho.RecordSource = strTask
    f.      End Select

6 End If

7 Exit Sub

8 errorHandler:
9 MsgBox "Dados incorrectos"
10   Me.txtProcurarOT.SetFocus
11   End Sub
```

Código A-7 – Procura de Ordens de Trabalho e abertura de formulário respectivo

Anexo XI – Gerar gráfico em Excel com os dados do tipo de manutenção

```
'Gerar documento para ler em excel com os dados do gráfico
Private Sub cmdExcel_Click()
    Dim xlApp As Excel.Application
    Dim xlBook As Excel.Workbook
    Dim xlSheet As Excel.Worksheet

    Dim db As DAO.Database
    Dim rs As DAO.Recordset
    Dim i As Integer
    Dim xlChart As Excel.ChartObject
    Dim rng As Range

    DoCmd.Hourglass (True)

    If IsNull(strTask) Or strTask = "" Then
        DoCmd.Hourglass False
       Exit Sub
    End If

    Set db = CurrentDb
    Set rs = db.OpenRecordset(strTask, dbOpenSnapshot)

    If rs.RecordCount = 0 Then
        MsgBox " Não há dados para colocar no excel", vbInformation + vbOKOnly, "Sem
dados"
        Exit Sub
    End If
    'Criar objecto da aplicação excel
    Set xlApp = Excel.Application

    xlApp.Visible = False

    'criar objecto para o book e a folha no documento excel
    Set xlBook = xlApp.Workbooks.Add
    Set xlSheet = xlBook.Worksheets(1)

    'Na folha colocar os dados nas células indicadas
    With xlSheet
        .Range("B1").Value = "Tipo Manutenção em " & Nz(Me.txtMes.Column(1), "????")

        i = 2
        Do While Not rs.EOF
            .Range("A" & i).Value = Nz(rs!TipoIntervencao, "???")
            .Range("B" & i).Value = Nz(rs!SumOfTotalHoras, 0)

            i = i + 1
        rs.MoveNext
        Loop
        'Criar objecto do gráfico na folha excel
        Set xlChart = .ChartObjects.Add(200, 20, 438, 373)

        'Configurar o gráfico
        With xlChart
            .RoundedCorners = True
```

```
            With .Chart

                .ChartType = xlPie

                .HasTitle = True
                '.ChartTitle = "Ocupação Tipo Manutenção"

                .HasLegend = False
                '.Legend.Position = xlLegendPositionBottom


                With .ChartTitle.Font
                    .Name = "Calibri"
                    .FontStyle = "Bold"
                    .Size = 16

                End With

                .SetSourceData Source:=xlSheet.Range("A1:B" & i - 1)
                .ChartArea.Interior.Color = RGB(240, 240, 240)
                .ApplyDataLabels , , , , , True, , True, , Chr(13)

            End With 'chart

        End With 'end xlChart

    End With

    DoCmd.Hourglass False
    xlApp.Visible = True
    rs.Close
    Set rs = Nothing
    db.Close

End Sub
```

Código A-8 – Criação de ficheiro Excel com o gráfico de dados do tipo de manutenção

Anexo XII – Gerar gráfico com mais do que uma série de dados em VBA

```
Private Sub cmdExcel_Click()
   Dim xlApp As Excel.Application
    Dim xlBook As Excel.Workbook
    Dim xlSheet As Excel.Worksheet

    Dim db As DAO.Database
    Dim rs As DAO.Recordset
    Dim rs1 As DAO.Recordset
    Dim i As Integer, y As Integer, iCampos As Integer, a As Integer
    Dim xlChart As Excel.ChartObject
    Dim xlChart2 As Excel.ChartObject
    Dim rng As Range
    Dim strDebug As String

    DoCmd.Hourglass (True)

    If IsNull(strTask5) Or strTask5 = "" Then
        DoCmd.Hourglass False
       Exit Sub
    End If

    Set db = CurrentDb
    Set rs = db.OpenRecordset(strTask5, dbOpenSnapshot)
    Set rs1 = db.OpenRecordset(strTask6, dbOpenSnapshot)


    If rs.RecordCount = 0 Or rs1.RecordCount = 0 Then
        MsgBox " Não há dados para colocar no excel", vbInformation + vbOKOnly, "Sem
dados"
        Exit Sub
    End If
    iCampos = rs.Fields.Count - 1

    Set xlApp = Excel.Application

    xlApp.Visible = False

    Set xlBook = xlApp.Workbooks.Add
    Set xlSheet = xlBook.Worksheets(1)

    With xlSheet
        .Range("B1").Value = "Ocupação do Recursos em " & Nz(Me.cboMes.Column(1),
"????")
        i = 2

        'Obter o nome dos campos
        .Range("A" & i).Value = "Data"
        For a = 1 To iCampos
            .Range(Chr(65 + a) & i).Value = rs(a).Name
        Next a

        MsgBox " Há " & iCampos & " recursos registados no mês"

        'Obter os dados dos registos para os campos
        i = 3
        Do While Not rs.EOF
            .Range("A" & i).Value = Nz(rs!Data, "???")
            For a = 1 To iCampos
                .Range(Chr(65 + a) & i).Value = Nz(rs(a), 0)
            Next a

            i = i + 1
            rs.MoveNext
        Loop
```

```
         'Determinar a área e localização do gráfico
        Set xlChart = .ChartObjects.Add(20, 20, 838, 373)

        'Determinar o tipo de gráfico e formata-lo
        With xlChart
            .RoundedCorners = True

            With .Chart

                .ChartType = xlColumnClustered

                .HasTitle = True
                .ChartTitle.Text = "Ocupação dos Recursos em " & Me.cboMes.Column(1) &
" ."


                .HasLegend = True
                .Legend.Position = xlLegendPositionTop


                With .ChartTitle.Font
                    .Name = "Calibri"
                    .FontStyle = "Bold"
                    .Size = 16

                End With
                .SetSourceData Range("A3:" & Chr(65 + iCampos) & i - 1)
                .ChartArea.Interior.Color = RGB(240, 240, 240)

                For a = 1 To iCampos
                    .SeriesCollection(a).Name = Range(Chr(65 + a) & "2")
                Next a

                .ApplyDataLabels
                '.SeriesCollection(1).Interior.Color = RGB(125, 165, 213)
                .PlotArea.Format.Fill.ForeColor.RGB = RGB(200, 200, 200)
                .ChartGroups(1).GapWidth = 10

            End With 'chart

        End With 'end xlChart
```

Código A-9 – VBA para criar gráfico de barras com mais do que uma série de dados

```
        With xlChart2

            .RoundedCorners = True

            With .Chart

                .ChartType = xlColumnClustered

                .HasTitle = True
                '.ChartTitle = .Range("B" & y).Value
                '.ChartTitle.Text = .Range("B" & y).Value

                .HasLegend = False
                '.Legend.Position = xlLegendPositionBottom



                'End With

                .SetSourceData Source:=xlSheet.Range("A" & y & ":B" & i - 1)
                .ChartArea.Interior.Color = RGB(240, 240, 240)
                .ApplyDataLabels
                '.SeriesCollection(1).Interior.Color = RGB(125, 165, 213)
                .ChartGroups(1).GapWidth = 10
                .PlotArea.Format.Fill.ForeColor.RGB = RGB(220, 220, 220)
                With .SeriesCollection(1).Format.Fill
                    .TwoColorGradient msoGradientHorizon, 1
                    .ForeColor.RGB = RGB(217, 60, 43)
                    .BackColor.RGB = RGB(228, 198, 182)

                End With

            End With 'chart

        End With 'end xlChart
    End With

    DoCmd.Hourglass (False)
    xlApp.Visible = True
    rs.Close
    Set rs = Nothing
    db.Close

End Sub
```

Código A-10 – VBA para criar segundo gráfico de colunas com a distribuição de horas por intervenção

Anexo XIII – Procurar ID de material com base na referência interna da empresa usando a classe *Recordset*

```
1 Public Function FuncIDMaterial(strRefMaterialMiralago As String) As String
2 Dim db As DAO.Database
3 Dim rsMaterialOT As DAO.Recordset

4 Set db = CurrentDb
5 Set rsMaterialOT = db.OpenRecordset("tblMaterial", dbOpenDynaset)

6 'Procurar se referencia existe
7 rsMaterialOT.FindFirst "RefInternaMiralago = '" & strRefMaterialMiralago & "'"

8 If rsMaterialOT.NoMatch Then
  a.   'Resposta caso o ID não exista
  b.   FuncIDMaterial = "False"
9 Else
  a.   'A resposta contem o valor do ID do material para introduzir no formulário
  b.   FuncIDMaterial = rsMaterialOT!ID
10     End If

11     rsMaterialOT.Close
12     Set rsMaterialOT = Nothing
13     db.Close

14     End Function
```

Código A-11 – Procura ID na tabela tblMaterial que corresponde à referência na Miralago

Anexo XIV – Evento de abertura do formulário frmMaterial

```
1 Private Sub Form_Open(Cancel As Integer)
2 Dim strReferenciaMiralago As String, strOpenArgs() As String

3 If IsNull(OpenArgs) Then
  a.   'desbloquear a referencia a criar
  b.   Me.txtRefInternaMiralago.Locked = False
4 Else

  a.   strOpenArgs = Split(OpenArgs, ";")
  b.   'Receber dados para adicionar nova referência recebida de outro formulário
  c.   FormOrigem = strOpenArgs(0)
  d.   If FormOrigem = "inicio" Then
   i. DoCmd.GoToRecord , , acLast

  e.   ElseIf FormOrigem = "OT" Then

   i. 'Caso sejam enviados argumentos para a abertura da ref. é aberto um novo registo
  ii. DoCmd.GoToRecord , , acNewRec
 iii. 'A nova referencia
  iv. strReferenciaMiralago = strOpenArgs(1)

   v. 'Usar variavel dentro da classe para obter informação do formulário de onde é
      proveniente a abertura
  vi. FormOrigem = "OT"

 vii. 'A nova referência é escrita em material e colocada em textbox bloqueada
viii. Me.txtRefInternaMiralago = strReferenciaMiralago
  ix. Me.txtRefInternaMiralago.Locked = True

  f.   End If
5 End If
6 End Sub
```

Código A-12 – Evento *Form_Open()* no formulário frmMaterial

Anexo XV – Processo de Fecho do Formulário frmMaterial

```
1   Private Sub cmdClose_Click()
2   Dim intID As Integer

3   intID = Me.ID.Value
4   DoCmd.Close acForm, "frmMaterial"

    'Após o fecho do formulário verificar que formulário desencadeou a abertura deste e
    desencadear as acções de acordo com tal
5   Select Case FormOrigem
6   Case "OT"
     a. Form_frmOrdensTrabalho.SetFocus
     b. Form_frmMaterialOrdemTrabalho.IDMaterial.Value = intID
     c. DoCmd.Restore
7   Case "PesquisaMaterial"
     a. Form_frmInicio.SetFocus
     b. DoCmd.Restore
8   Case "Inicio"
     a. Form_frmInicio.SetFocus
     b. DoCmd.Restore
9   Case "subPesquisaMateriais"
     a. Form_frmInicio.SetFocus
     b. DoCmd.Restore
10  Case Else
     a. Form_frmInicio.SetFocus
     b. DoCmd.Restore
11  End Select

12  End Sub
```

Código A-13 – Evento cmdClose_Click() no formulário frmMaterial

Anexo XVI – Comando para pesquisar referências de artigos

```
1 Private Sub cmdFind_Click()
2 Dim strTask As String

3 If IsNull(Me.txtRefMiralago) Then
  a.   strTask = "SELECT tblMaterial.ID, tblMaterial.RefInternaMiralago,
    tblMaterial.DesignacaoMiralago " _
    & "FROM tblMaterial " _
    & "WHERE ((tblMaterial.DesignacaoMiralago) Like '*" & Me.txtDesignacao & "*') " _
    & "ORDER BY tblMaterial.RefInternaMiralago;"

4 ElseIf IsNull(Me.txtDesignacao) Then
  a.   strTask = "SELECT tblMaterial.ID, tblMaterial.RefInternaMiralago,
    tblMaterial.DesignacaoMiralago " _
    & "FROM tblMaterial " _
    & "WHERE (((tblMaterial.RefInternaMiralago) Like '" & Me.txtRefMiralago & "*')) "
    _
    & "ORDER BY tblMaterial.RefInternaMiralago;"

5 ElseIf IsNull(Me.txtDesignacao) And IsNull(Me.txtRefMiralago) Then:
  a.   strTask = "SELECT tblMaterial.ID, tblMaterial.RefInternaMiralago,
    tblMaterial.DesignacaoMiralago " _
    & "FROM tblMaterial " _
    & "WHERE (((tblMaterial.RefInternaMiralago) Like '@')) " _
    & "ORDER BY tblMaterial.RefInternaMiralago;"
6 Else
  a.   strTask = "SELECT tblMaterial.ID, tblMaterial.RefInternaMiralago,
    tblMaterial.DesignacaoMiralago " _
    & "FROM tblMaterial " _
    & "WHERE (((tblMaterial.RefInternaMiralago) Like '" & Me.txtRefMiralago & "*') AND
    ((tblMaterial.DesignacaoMiralago) Like '*" & Me.txtDesignacao & "*')) " _
          & "ORDER BY tblMaterial.RefInternaMiralago;"
7 End If

8 If Me.txtRefMiralago.Value = " " Then Me.txtRefMiralago = Null

9 Me.ListRefMiralago.RowSource = strTask

10     End Sub
```

Código A-14 – Preenchimento das caixas de textos e listas para pesquisa de artigos

Anexo XVII – Determinar o número de dias úteis

```
'Retornar o número de dias uteis sem fins de semana retorna -1 em caso de erro
Public Function DiasUteis(startDate As Date, endDate As Date) As Integer
    Dim varDias As Variant, varDiasFDS As Variant

    On Error GoTo DiasUteis_Error

    ' Temporary storage for datetime.
    Dim tempDate As Date

    'Se a data de fim for menor que a data de inicio, trocar datas
    If endDate < startDate Then
        temDate = startDate
        startDate = endDate
        endDate = tempDate
    End If

    'Calcular o numero de dias inclusive (+1 para adicionar a data de inicio)
    varDias = DateDiff(Interval:="d", date1:=startDate, date2:=endDate) + 1

    ' Calcular o número de dias de fds completos com 2 dias de FDS
    varDiasFDS = DateDiff("ww", startDate, endDate) * 2

    'Se a data inicial for um domingo adicionar um dia
    If DatePart("w", startDate) = vbSunday Then varDiasFDS = varDiasFDS + 1

    'Se a data final se for um sábado adicionar um dia
    If DatePart("w", endDate) = vbSaturday Then varDiasFDS = varDiasFDS + 1

    'Calcular o número de dias uteis.
    DiasUteis = (varDias - varDiasFDS)

DiasUteis_Exit:
    Exit Function

DiasUteis_Error:
    DiasUteis = -1
    MsgBox "Erro " & Err.Number & ": " & Err.Description, vbCritical, "Dias Uteis"
    Resume DiasUteis_Exit
End Function
```

Código A-15 – Determinar o número de dias uteis entre datas

Anexo XVIII – Encontrar imagens já carregadas na base de dados

```
'--------------------------------------------------------------
' cmdFind_Click
'
'--------------------------------------------------------------
Private Sub cmdFind_Click()

On Error GoTo cmdFind_Click_Err

Me.RecordSource = "SELECT tblImagem.ID, tblImagem.LinkImagem, tblImagem.LinkWeb,
 tblImagem.Familia, tblImagem.SubFamilia, tblImagem.Maquina " _
 & "FROM tblImagem " _
 & "WHERE (((tblImagem.LinkImagem) Like '*" & Me.txtFind & "*'));"

Me.txtFind.SetFocus
Exit Sub

cmdFind_Click_Exit:
Exit Sub

cmdFind_Click_Err:
MsgBox Error$
    Resume cmdFind_Click_Exit

    End Sub
```

Código A-16 – Pesquisa de texto no endereço de imagens da tabela tblImagens

Anexo XIX – Associar Imagem a Material

```
1  Private Sub cmdExportarMaterial_Click()
2  Dim strLinkImagem As String, strIDImagem As String

3  'Guardar valores pertinentes do formulário antes de o fechar
4  strIDImagem = Me.txtID
5  strLinkImagem = Me.txtLinkImagem

6  'Fechar Formulário frmEscolhaImagens
7  DoCmd.Close acForm, "frmEscolhaImagens"

8  'Confirma que o endereço de origem é material antes de prosseguir
9  If FormOrigem = "material" Then
    a. BaseDeDados.AdicionarImagemMaterial strIDImagem, strIDMaterial
    b. Form_frmMaterial.SetFocus
    c. DoCmd.Restore
    d. Form_frmMaterial.imgImagemProduto.Picture = strLinkImagem
10 End If
```

Código A-17 - Associar imagem a determinado material

```
1 Public Function AdicionarImagemMaterial(strIDImagem As String, strIDMaterial As
  String) As Boolean

2 Dim db As DAO.Database
3 Dim rs As DAO.Recordset
4 Dim ImagemRecord As Long

5 On Error GoTo ErrorHandler

6 Set db = CurrentDb
7 Set rs = db.OpenRecordset("tblMaterialImagem", dbOpenDynaset)

8 rs.FindFirst "Material = " & strIDMaterial

9 'Adicionar imagem á tabela de materiais
10      If rs.NoMatch Then
  a.    rs.AddNew
  b.    rs("Material") = strIDMaterial
  c.    rs("ImagemMaterial") = strIDImagem
  d.    rs.Update

11      Else
  a.    rs.Edit
  b.    rs("ImagemMaterial") = strIDImagem
  c.    rs.Update

12      End If

13      rs.Bookmark = rs.LastModified
14      ImagemRecord = rs("ID")

15      rs.Close
16      Set rs = Nothing
17      db.Close

18      MsgBox "ID da Imagem adicionada " & ImagemRecord & " ao material " &
  strIDMaterial

19      Exit Function

20      errorHandler:
21      MsgBox "Erro a carregar dados na tabela ImagemMaterial"
22      End Function
```

Código A-18 – Método para adicionar ID da Imagem ao ID do Material

Anexo XX – Envio de e-mails para Logística

```
1 Private Sub SendEmail(strBody As String, strSubject As String, strEmail As String)

2 Dim strSubjectGmail As String, strBodyGmail As String, strSendToGmail As String,
  strAttachGmail As String
3 Dim SplitFile() As String
4 Dim strImageFile As String, strFornecedor As String
5 Dim iArraySize As Integer

6 Dim strResposta As String

7 Dim FSO As New FileSystemObject
8 Dim folderPath As String, fileName As String, strImagePath As String, TempFilePath
  As String

9 'Inicializar objectos para o envio de email
10      Dim oApp As Object
11      Dim oEmail As Object

12      'Recolher caminho da imagem no computador de origem com base na informação
13      'preente no formulário ref. Material
14      If IsNull(Me.imgImagemProduto.Picture) Then
  a.    strImagePath = ""
15      Else
  a.    strImagePath = Me.imgImagemProduto.Picture
  b.    SplitFile = Split(strImagePath, "\")
  c.    iArraySize = UBound(SplitFile)
  d.    strImageFile = SplitFile(iArraySize)

16      End If

17      'Designação do fornecedor
18      strFornecedor = Me.cboFornecedor.Column(1)

19      'O ficheiro seleccionado existe?
20      If FSO.FileExists(Me.txtAnexo.Value) Then
  a.    'Caminho para a directoria e caso não exista cria a directoria a copia o
    ficheiro
  b.    folderPath = CurrentProject.Path & "\anexos"

 i. 'Caso não exista a directoria deve cria-la
  c.    If Not FSO.FolderExists(folderPath) Then FSO.CreateFolder folderPath

  d.    'Copiar o ficheiro para a directoria
  e.    FSO.CopyFile Nz(Me.txtAnexo.Value, ""), folderPath & "\", True

  f.    'Ficar apenas com o nome do ficheiro
  g.    fileName = FSO.GetFileName(Nz(Me.txtAnexo.Value, ""))

  h.    'Usar o ficheiro entretanto copiado
  i.    oEmail.Attachments.Add folderPath & "\" & fileName

  j.    strAttachGmail = folderPath & "\" & fileName
21      Else
  a.    strAttachGmail = ""
22      End If

23      (Continua)….
```

Código A-19 – Primeira parte de código para envio de e-mail e respetivo ficheiro anexo

```
1   'Criar objectos respectivos para a aplicação e o email
2   Set oApp = CreateObject("Outlook.Application")
3   Set oEmail = oApp.CreateItem(0)

4   With oEmail
    a.  If Not Me.txtEmail = "" Then
        i.  .To = gblEmailCompras
    b.  Else
        i.  .To = strEmail
    c.  End If

    d.  .CC = gblEmailConhecimento
    e.  .Subject = strSubject

    f.  'Adicionar imagem como anexo
    g.  If FSO.FileExists(strImagePath) Then
         i.  .Attachments.Add strImagePath, 1, 0

        ii.  .HTMLBody = .HTMLBody & "<br><br>" & Me.txtBox & "</font><br>" _
             & "<hr><br><br>" & strBody & "</font><p>Fornecedor: " & strFornecedor &
             "</font></span>" _
             & "<hr><img src='cid:" & strImageFile & "'" _
             & "alt='Imagem Não Disponivel'" _
             & "with='500' height='200'><br>"

       iii.  strBodyGmail = "<br><br>" & Me.txtBox & "</font><br>" _
             & "<hr><br><br>" & strBody & "</font><p>Fornecedor: " & strFornecedor &
             "</font></span>"

    h.  Else
         i.  .HTMLBody = "<br><br>" & Me.txtBox & "</font><br>" _
             & "<hr><br>" & strBody & "</font><p>Fornecedor: " & strFornecedor &
             "</font></span>"
    i.  End If

    j.  strSendToGmail = .To
    k.  strSubjectGmail = strSubject

    l.  modEnviarMail.SendGmail strSendToGmail, strSubjectGmail, strBodyGmail,
        strAttachGmail

    m.  If Not IsNull(.To) And Not IsNull(.Subject) And Not IsNull(.Body) Then
         i.  .Display

        ii.  '.Send
       iii.  MsgBox "Email Enviado"
        iv.  Me.txtEmail = ""
         v.  Me.txtBox = ""
        vi.  Me.txtAnexo = ""
       vii.  Me.cboListaMaquinas = ""
      viii.  Me.txtDesignacaoMaquina = ""

        ix.  'Fazer log dos dados enviados em ficheiro e em tabela

         x.  Logging.WriteLog "Email sent - To: " & strEmail _
         a.  & " Subject: " & strSubject _
         b.  & " Body: " & strBody _
         c.  & " Fornecedor: " & strFornecedor
        xi.  Logging.WriteLogToTable strEmail, strSubject, strBody, strFornecedor,
             strImagePath

    n.  Else
         i.  MsgBox " Por favor preeencha todos os campos necessários"
    o.  End If

5   End With
6   End Sub
```

Código A-20 – Envio de e-mail usando cliente do sistema e usando cliente externo (Gmail).

```
1 Public Function SendGmail(SendTo, Subject, Body, Attach)

2 Dim Mail As New message
3 Dim Config   As Configuration

4 Set Config = Mail.Configuration
5 Config(cdoSendUsingMethod) = cdoSendUsingPort
6 Config(cdoSMTPServer) = "smtp.gmail.com"
7 Config(cdoSMTPAuthenticate) = cdoBasic
8 Config(cdoSMTPUseSSL) = True
9 Config(cdoSendUserName) = "jose.seabra@miralago.pt"
10      Config(cdoSendPassword) = "10254078"
11      Config.Fields.Update

12      Mail.To = SendTo
13      Mail.CC = "julio.chio@miralago.pt"
14      Mail.From = Config(cdoSendUserName)
15      Mail.Subject = Subject
16      Mail.HTMLBody = Body

17      If Not Attach = "" Or IsNull(Attach) Then
  a.    Mail.AddAttachment Attach
18      End If

19      On Error Resume Next

20      Mail.Send

21      If Err.Number <> 0 Then
  a.    MsgBox Err.Description, vbCritical, "There was an error"
22      End If

23      End Function
```

Código A-21 – Envio de E-mail usando CDO e Gmail

Anexo XXI – Indicador Tempo Médio Entre Intervenções

```
1 Public Function TempoMedioEntreFalhas(Maquina As String, Ano As Integer) As String
2 Dim db As DAO.Database
3 Dim rs As DAO.Recordset
4 Dim NumeroDeRegistos As Double, MTBF As Double, SomaDatas As Double
5 Dim DataAnterior As Date, NovaData As Date
6 Dim i As Integer

7 Dim strDataPedido, strMinDate, strMaxDate, strTask1 As String
8 Dim FirstDayYear As Single, LastDayYear As Single

9 ' obtenção da data de hoje do sistema em numero série
10     strDataPedido = CDate(DateSerial((Ano), Month(Date), Day(Date)))

11     'Selecção do primeiro e ultimo dia do ano em análise
12     FirstDayYear = DateSerial(Year(strDataPedido), 1, 1)
13     LastDayYear = DateSerial(Year(strDataPedido), 12, 31)

14     'Tarefa a executar para separar por anos os dados iniciais
15     strMinDate = CDate(FirstDayYear)
16     'strMinDate = Format(strMinDate, "MM-dd-YYYY")

17     'data máxima do ano (fim do ano)
18         strMaxDate = CDate(LastDayYear)
```

Código A-22 - Inicialização de variáveis e obtenção de datas para avaliação para o MTBF

```
1 'caso seja colocado o ano 0 a função apenas considera a máquina que se pretende
  analisar todas as datas
2 If ano = 0 Then
  a.   strTask1 = "SELECT tblDadosIntervencao.ID, tblDadosIntervencao.Maquina, " _
    & "tblDadosIntervencao.DataPedido, tblDadosIntervencao.EstadoActual " _
    & "FROM tblDadosIntervencao " _
    & "WHERE (((tblDadosIntervencao.Maquina)='" & Maquina & "') AND
    ((tblDadosIntervencao.EstadoActual)= 'Executado')) " _
    & "ORDER BY tblDadosIntervencao.DataPedido;"

3 'Caso de pretenda analisar todas as máquinas num determinado ano
4 ElseIf (Maquina = "all" And ano > 2013) Then
  a.   strTask1 = "SELECT tblDadosIntervencao.ID, tblDadosIntervencao.Maquina, " _
    & "tblDadosIntervencao.DataPedido, tblDadosIntervencao.EstadoActual " _
    & "FROM tblDadosIntervencao " _
    & "WHERE (((tblDadosIntervencao.Datapedido) > #" & strMinDate & "# AND
    (tblDadosIntervencao.Datapedido) < #" _
    & strMaxDate & "#) AND (tblDadosIntervencao.EstadoActual)= 'Executado') " _
    & "ORDER BY tblDadosIntervencao.DataPedido;"
5 ElseIf (ano = -1) Then
  a.   strTask1 = "SELECT tblDadosIntervencao.ID, tblDadosIntervencao.Maquina,
    tblDadosIntervencao.DataPedido, tblDadosIntervencao.CondicaoEquipamento,
    tblDadosIntervencao.EstadoActual " _
    & "FROM tblDadosIntervencao " _
    & "WHERE (((tblDadosIntervencao.Maquina)='" & Maquina & "') AND
    ((tblDadosIntervencao.CondicaoEquipamento)='não')AND
    ((tblDadosIntervencao.EstadoActual)= 'Executado')) " _
    & "ORDER BY tblDadosIntervencao.DataPedido;"
6 End If

7 Set db = CurrentDb
8 Set rs=db.OpenRecordset(strTask1)

9 If rs.RecordCount = 0 Then
  a.   TempoMedioEntreFalhas = 0
  b.   Exit Function
10     End If

11     With rs
  a.   '.MoveLast
  b.   NumeroDeRegistos = .RecordCount
  c.   .MoveFirst
  d.   For i = 1 To NumeroDeRegistos - 1
    i. DataAnterior = .Fields("DataPedido").Value
   ii. .MoveNext

  iii. NovaData = .Fields("DataPedido").Value
   iv. SomaDatas = SomaDatas + (NovaData - DataAnterior)

  e.   Next i

  f.   MTBF = SomaDatas / NumeroDeRegistos

  g.   TempoMedioEntreFalhas = Round(MTBF, 2)

12     End With
13     rs.Close
14     Set rs = Nothing

15     db.Close

16     End Function
```

Código A-23 - *Querys* para obtenção de informação da base de dados e aplicação da fórmula utilizada para MTBF