Universidade do Porto
FEUP Faculdade de Engenharia

Licenciatura em Engenharia Electrotécnica e de Computadores

Ramo de Automação, Produção e Electrónica Industrial

Sistemas de Informação Empresarial

5º Ano – 1ºSemestre
2005 - 2006

Sistema de controlo de Produção para a empresa *SoMetal*

Versão 1.2

Proposta elaborada por:

ee04255	**Ana Martins**
ee00011	**André Carvalhosa**
ee04217	**António Nogueira**
ee01234	**Catarina Santiago**
ee04249	**Gabriel Silva**
ee04250	**João Felgueiras**
ee04216	**Ricardo Cardoso**

0. Controlo do Documental

0.1 Identificação do Documento

Tipo de documento	Relatório final do projecto
Estado do documento	Aprovado
Âmbito	Desenvolvimento de um Sistema de informação empresarial
Autor	Grupo C
Nome do ficheiro	Relatório_final_projecto_V_1_2.pdf

0.2 Controlo de Versões

Edição/Revisão	Data	Justificação
1.0	17/11/2005	Criação do documento
1.1	25/11/2005	Alteração dos requisitos do sistema
1.2	23/12/2005	Término do projecto e a sua documentação

Índice

Índice de figuras

Índice de tabelas

1. Introdução

1.1. Propósito do documento

Este documento visa a especificação de um sistema de controlo de produção para apoio aos gestores da empresa SoMetal que indica, em tempo útil, informações sobre rendimentos e estado operacional dos recursos produtivos.

1.2. Abreviações

BD – Base de dados
SIE - Sistema de informação empresarial

1.3. Âmbito

O sistema recolhe dados ao nível do chão de fábrica, apresentando, através de tratamento e armazenamento de informação, o estado das máquinas do sistema de produção e indicadores estatísticos sobre rendimentos de máquinas, operadores e tipos de peças.

Este sistema automático de recolha e tratamento de dados tem como objectivo a melhoria da eficiência operacional, de modo a que se torne possível, não só a enumeração das causas de perda de rendimento, mas também quantificar a contribuição de cada uma das causas no rendimento geral da produção. O sistema permite ainda o acesso a informação detalhada de cada deficiência de produtividade, informação do tipo molde, tipo de peça, máquina, operador ou mesmo tempo de paragem.

1.4. Antevisão

No capítulo seguinte será efectuada uma descrição geral do sistema a desenvolver, especificando as interfaces e classes de utilizadores do sistema, assim como a interface com outros hardwares, interfaces de comunicações e as funcionalidades do mesmo.

No terceiro capítulo é efectuada uma descrição geral da arquitectura do sistema, identificando os principais blocos do SIE a desenvolver.

No quarto capítulo serão descritas as potencialidades do sistema evidenciando os aspectos que permitem detectar o problema da elevada taxa de não conformidades da SoMetal.

No quinto capítulo, o código desenvolvido nas aplicações será comentado tentando sempre que possível recorrer a diagramas para descrever o fluxo do mesmo.

No último capítulo será feita uma autocrítica ao trabalho realizado.

2. Descrição geral

2.1. Perspectiva do produto

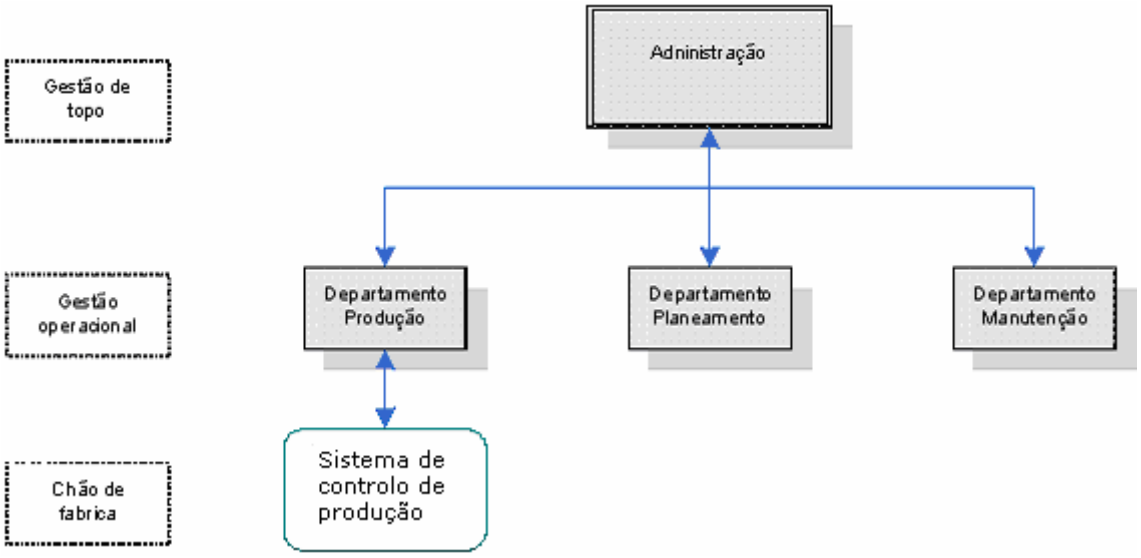


Fig. 1 - Perspectiva do produto

O sistema apresentado deverá ser implementado sobre o sistema de gestão da produção existente na empresa permitindo a interacção com o sistema de manutenção, sistema de planeamento, gestão da qualidade e a administração.

2.2. Interface com utilizador

As interfaces com os diferentes utilizadores do sistema serão especificadas no ponto 5.2 (Aplicações web) do documento.

2.3. Interface com hardware

O sistema a desenvolver contempla interfaces entre diferentes tipos de hardware. A recolha dos dados provenientes do chão de fábrica será efectuada com o auxílio de um autómato programável.

As operações que geram os sinais são simuladas recorrendo a um kit. Existirão ainda terminais do tipo PC que permitirão ao operador introduzir informações que o sistema peça e ao gestor consultar os dados de produção.

2.4. Interfaces de comunicação

A comunicação entre o autómato e o PC será feita com o protocolo Hostlink sobre uma rede RS232. A comunicação entre os terminais e o sistema será através de uma rede Ethernet.

2.5. Funções do produto

O sistema fornece informação sobre o estado das máquinas assim como causas de paragem e cadência de produção actual. Em relação aos indicadores estatísticos, o sistema fornece rendimentos para equipamentos, operadores e tipos de peça. É ainda disponibilizada a opção de calcular rendimentos cruzados, máquina/tipo de peça ou máquina/operador.

2.6. Características dos utilizadores

Os utilizadores do sistema a desenvolver serão os gestores e operadores da empresa SoMetal.

Os gestores da empresa deverão ter formação académica na área de gestão e conhecimentos de informática que permitam tirar o maior partido das funcionalidades do sistema.

Os operadores deverão ter uma formação específica na sua área de actuação para que a sua acção seja eficiente.

3. Especificação do sistema de controlo de produção

Este ponto especifica o sistema de informação empresarial a desenvolver, descrevendo as funções dos módulos que compõem o mesmo.

3.1. Arquitectura Geral do Sistema

Como se pode ver o sistema pode ser dividido em três módulos que representam campos distintos de aplicação. Assim, existe um módulo que é responsável por recolher os dados ao nível do chão de fábrica, outro que constitui o interface com os utilizadores e o último responsável por tratar os dados fornecidos pelo autómato e armazená-los na base de dados.

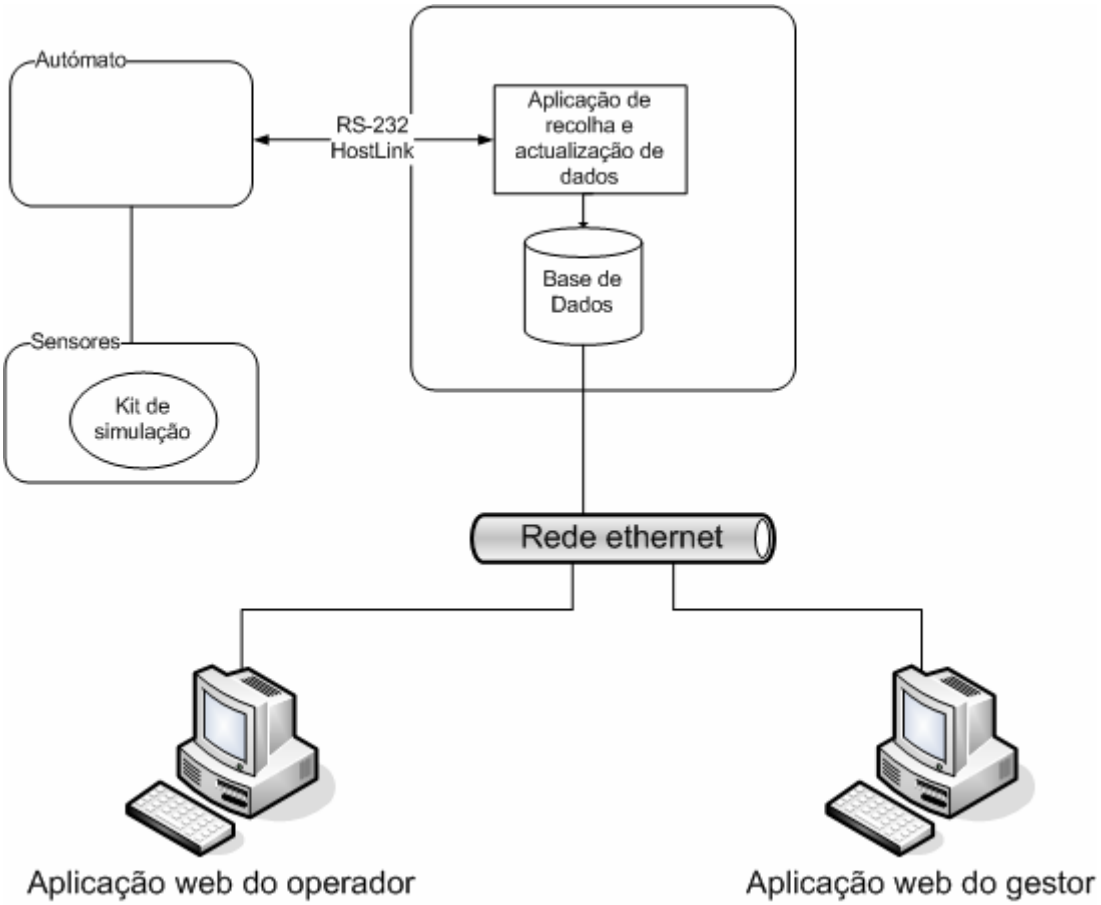


Fig. 2 - Arquitectura geral do sistema

4. Potencialidades do sistema desenvolvido

O SIE desenvolvido permite efectuar um diagnóstico cuidado da(s) potenciais fonte(s) de anomalias do processo de produção, na medida em que fornece o conjunto de dados necessários à sua análise.

Através da monitorização do número de peças produzidas, do tempo de funcionamento, do tempo de paragem, tempo de avaria, do tipo de molde utilizado e do operador, o sistema desenvolvido permite à gestão, através de pesquisa cruzada na base de dados, consultar os seguintes indicadores:

- O tempo de funcionamento e de paragem de uma dada máquina;
- A causa de paragem de uma dada máquina;
- O tempo e número de utilizações do molde;
- O número de peças produzidas por cada molde;
- O número e tipo de peças produzidas por cada maquina;
- A cadência associada a uma dada máquina e a uma dada ordem de trabalho;
- O operador associado a uma dada máquina e o seu “rendimento”;
- O operador associado à produção de um dado tipo de peça e o seu “rendimento”;

A pesquisa de dados cruzada implementada por este sistema permite desta forma detectar se o problema da elevada taxa de não conformidades da SoMetal está associado a anomalias de uma dada máquina, peça, operador ou molde.

Os requisitos iniciais foram no sentido de monitorizar as máquinas que a gestão da SoMetal considera como principais fontes de problemas associados à elevada taxa de não conformidades. No entanto, a extensão deste sistema a todo o processo produtivo é facilmente conseguida através da replicação e integração do código para todas as máquinas.

Asseguramos pelas razões acima indicadas que estão reunidas as condições para detectar as principais fontes da elevada taxa de não conformidades da SoMetal.

5. Aplicações desenvolvidas

5.1. Aplicação do autómato

Através do protocolo HostLink o PC escreve e lê o conteúdo de diversas posições de memória. No início do ciclo o autómato copia as variáveis fornecidas pelo PC, que são escritas em DATA MEMORY'S (D), para as WORD'S (W). O programa processa sobre as mesmas e no fim do ciclo copias as W para as D, para poderem ser acedidas pelo PC.

As variáveis recebidas, descrições e endereços podem ser visualizados na seguinte tabela:

Variável	**Descrição**	**Valor**
D30.0 D30.1	Novo molde máquina 1 Novo molde máquina 2	0 – Sem alteração 1 – Novo molde
D31	3 * Tempo de ciclo nominal da máquina 1 em segundos	
D32	3 * Tempo de ciclo nominal da máquina 2 em segundos	

Tabela 1 - Variáveis recebidas pelo autómato

Antes de ser dada uma nova ordem de produção é necessário colocar o novo tempo de ciclo nominal (D31 e/ou D32) e só depois activar as flags de novo molde (D30.0 e/ou D30.1), para que o tempo seja actualizado.

As variáveis fornecidas pelo autómato, endereços e respectivas descrições são:

Variável	Descrição	Valor
D0.0 D0.1	Estado da máquina 1 Estado da máquina 2	0 – OFF 1 – ON
D0.2 D0.3	Estado da máquina 1 Estado da máquina 2	0 – Avariada 1 – Em funcionamento
D0.[13..12] D0.[15..14]	Tipo de peça máquina 1 Tipo de peça máquina 2	00 - A 01 - B 10 - C 11 - D
D1	Número total de ciclos máquina 1[1]	
D2	Número total de ciclos máquina 2[1]	
D3	Tempo de funcionamento da máquina 1 em minutos[2]	
D3	Tempo de funcionamento da máquina 2 em minutos[2]	

[1] É colocada a zero sempre que é lançada nova ordem.
[2] É colocado a zero sempre que a respectiva máquina pára.
[2] É colocada a zero sempre que a respectiva máquina retoma o seu funcionamento

D5	Tempo de paragem da máquina 1 em minutos[3]	
D6	Tempo de paragem da máquina 2 em minutos[3]	
D7	Tempo total de funcionamento da máquina 1 em minutos[1]	
D8	Tempo total de funcionamento da máquina 2 em minutos[1]	
D9	Numero de peças produzidas pela máquina 1[1]	
D10	Numero de peças produzidas pela máquina 2[1]	

Tabela 2 - Variáveis fornecidas pelo autómato

Ao ser dada uma nova ordem de produção todas as variáveis anteriores são colocadas a zero pelo autómato.

Embora o autómato forneça os tempos com uma resolução de minutos, internamente, por uma questão de precisão, todos os tempos utilizados são em segundos.

Na transição ascendente do sinal da cadência é iniciada a contagem do tempo de ciclo actual, sendo esta comparada com 3 * o valor do tempo de ciclo nominal (D31 e D32). Quando o tempo de ciclo actual é superior a 3 * o valor do tempo de ciclo nominal o autómato assinala a máquina como avariada.

Se a máquina estiver a meio de um ciclo e for dada uma nova ordem de fabrico, o novo valor do tempo de ciclo nominal e a inicialização de todas as variáveis dessa máquina, apenas é efectuado na próxima transição ascendente, para evitar inconsistências nos dados fornecidos.

[3] É colocada a zero sempre que a respectiva máquina retorna ao seu funcionamento.

5.2. Base de dados

A construção do modelo da base de dados partiu de um conjunto de tabelas que o grupo pensou como estritamente necessárias visto o seu grau de singularidade e especificidade, como:

- Peça
- Lote
- Molde
- Causa de Paragem
- Máquina
- Utilizador

Além da informação geral de cada uma destas tabelas, como descrição e código, há a realçar alguns campos específicos que foram adicionados tendo em conta as necessidades da nossa solução.

Como tal, nas tabelas “Molde” e ”Máquina” estão presentes informação do tipo: total de utilizações e de tempo de funcionamento, útil para a apresentação de indicadores do processo produtivo.

Na tabela “Peça” foi adicionado o tempo nominal de ciclo e a quantidade máxima de cada lote, da respectiva peça, permitindo o correcto cálculo de cadências de produção e a correcta divisão de ordem de fabrico em lotes.

Cada operador foi vinculado a uma máquina específica através do campo código de máquina da tabela “Utilizador”, que no caso de gestor se encontra vazio. Esta informação permite ao sistema reconhecer automaticamente a máquina através do login de um operador. Este método de reconhecimento seria complementado, no caso de implementação real, com a informação do próprio posto, que poderia estar também ele vinculado a uma só máquina.

De seguida e tendo em conta as soluções encontradas para o problema proposto foram criadas algumas tabelas adicionais:

- Ordem de Fabrico
- Ordem de Trabalho
- Históricos

No ponto de vista do grupo faz sentido, seguindo uma lógica corrente na indústria, criar ordens de fabrico e consequentemente serem gerados os lotes necessários para satisfazer o número de produtos pretendidos. Assim optou-se por adicionar a tabela “Ordem de Fabrico”.

Por outro lado foi criada uma tabela “Ordem de Trabalho”, com o objectivo de centralizar toda a informação necessária ao processo de produção, como máquina e operador, entre outras.

Por último, sendo necessário o armazenamento de informação relativa a uma ordem de trabalho, do tipo tempo, estado e cadência de operação, foi implementada uma tabela “Históricos”, que permite uma visão do passado e do desenrolar da produção.

A par da informação do tipo “passado” é necessário um outro tipo de conhecimento do processo de produção, ao qual chamamos de “tempo real” e adicionamos como campos na tabela “Máquina”, em que está definido o estado e a cadência real da respectiva máquina.

Ao contrário do apresentado no guião anterior optou-se por não utilizar vistas dado que a própria estrutura da base de dados permite a centralização de informação na tabela “Ordem Trabalho”, o que reduz a complexidade das *querys* a efectuar à base de dados, objectivo último da implementação de vistas.

Modelo da base de dados:

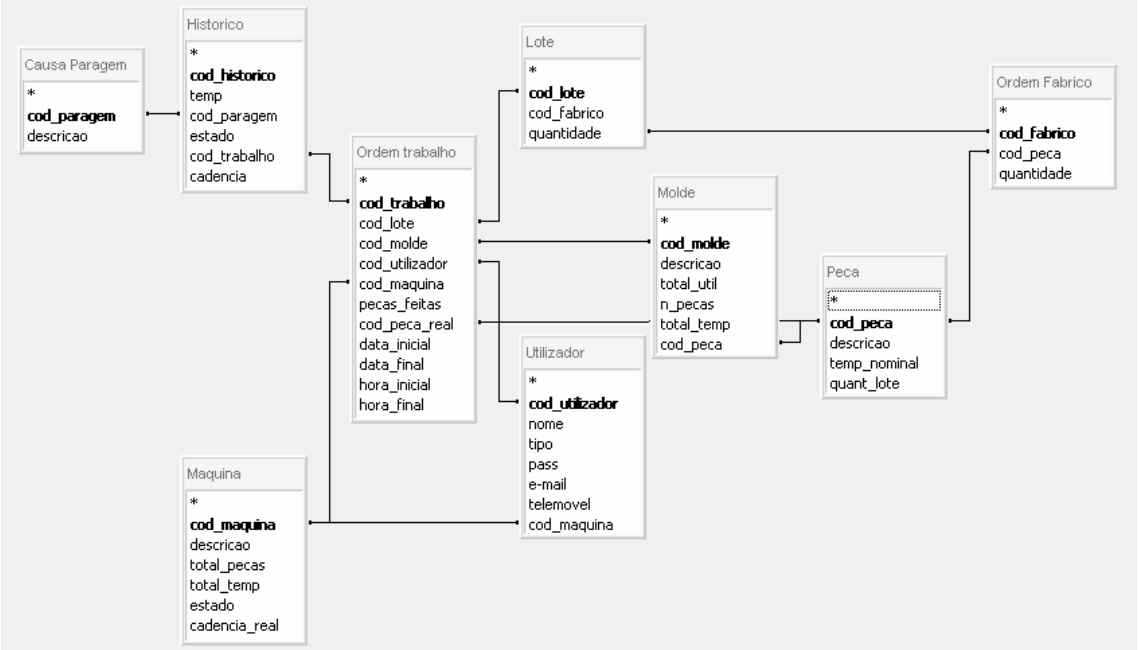


Fig. 3 - Modelo da base de dados

Restrições:

Além das validações de informação da própria interface, realizada por *Javascript* do tipo valores não nulos e positivos (caso necessário), foram implementadas medidas de segurança na própria base de dados, que garantem a integridade dos vários campos de informação.

Como tal foi definido o tipo de informação de cada campo e sempre que necessário restrições do tipo *not null*.

Por outro lado, a definição de chaves primárias e estrangeiras, implementam por si só um conjunto de restrições. Assim as chaves primárias são únicas e não nulas (campos a negrito do modelo

apresentado), enquanto que as chaves estrangeiras (ligações entre tabelas) definem, entre outros, o modo de propagação de informação em caso de alterações. Neste caso optou-se por não propagar alterações, porque se, por um lado não são permitidas alterações nos campos definidos como chaves primárias, por outro é permitida a alteração de informação dos campos definidos como chave estrangeira desde que a informação exista na tabela referenciada.

5.3. Aplicações web

As aplicações web foram desenvolvidas tendo em conta o nível de funcionalidade oferecido para as duas formas de utilizador: gestor e operador. A aplicação começa com uma página de login, a qual fará distinção entre utilizadores, através do campo “tipo” da tabela “utilizador”. Caso o utilizador seja um gestor será apresentada uma página diferente da do operador, que oferece um maior leque de funcionalidades que serão apresentadas de seguida.

Gestor

A aplicação de gestor apresenta várias funcionalidades entre elas, a apresentação do cronograma, sinóptico e vários indicadores estatísticos. O gestor tem ainda a opção de adicionar novas ordens de fabrico que serão convertidas em lotes de forma transparente.

Após ser inserida uma nova ordem de fabrico, recorrendo à tabela “peça” e ao campo quantidade de lote, a ordem será dividida em lotes com a quantidade estabelecida que serão inseridos na tabela “lote”, os quais podem ser consultados detalhadamente, na página de visualização de ordens de trabalho.

O gestor pode ainda fazer várias alterações e inserções de diversas informações do sistema, como: máquinas, utilizadores, moldes, causas de paragem e peças. Estas alterações prendem-se com actualizações que podem ser feitas nas tabelas acima referidas. Por exemplo, podemos fazer um update a um determinado molde, inserindo ou eliminando elementos da tabela, visto estes poderem ter sido submetidos a arranjo ou ajuste.

Todas as informações inseridas são protegidas por JavaScript sendo aceites se válidas. Existem campos como o número de peças em que a quantidade terá de ser positiva e não nula.

Cronograma

Para a visualização do funcionamento de cada uma das máquinas ao longo de um período, pensou-se fazer esta funcionalidade recorrendo a um gráfico, o qual apresentaria a cadência por máquina ao longo de um período.

O gráfico de apresentação poderia ter sido implementado de várias maneiras, recorrendo a tabelas, completando o conteúdo de cada célula, recorrendo à compilação de livrarias no servidor que apoiariam a realização de gráficos mais complexos e atractivos visualmente. No entanto e devido à escassez de tempo, tomou-se opção por aquilo que seria mais viável. Após ter sido feito um estudo sobre a forma de implementar os gráficos, verificou-se que se poderia fazer os mesmos recorrendo a uma livraria já existente no servidor, sendo esta a livraria GD.

Achou-se por bem recorrer a um gráfico que apresentasse a evolução do rendimento da máquina para cada ordem de trabalho. Assim, o gestor teria a opção de escolher entre as duas máquinas que se encontram com ordens de trabalho atribuídas. Após a escolha da máquina o gestor selecciona a ordem de fabrico que pretende observar e é gerado o respectivo cronograma. Devemos notar que cada ordem de trabalho, ou cada lote só estará na linha no máximo durante 24 horas, daí se ter decidido dar um espaço temporal no gráfico de 24 horas. No entanto para se incluir o gráfico na mesma página web, foi necessário encontrar um certo equilíbrio entre a percepção e dimensões do gráfico. Daí ter-se efectuado uma divisão do espaço temporal em meias horas, com uma dimensão de 720 pixeis. Para o rendimento, também foi escolhida uma dimensão de 720 pixeis, no entanto, optou-se apenas por dividir em dez parcelas, correspondendo a cada uma 10% de rendimento.

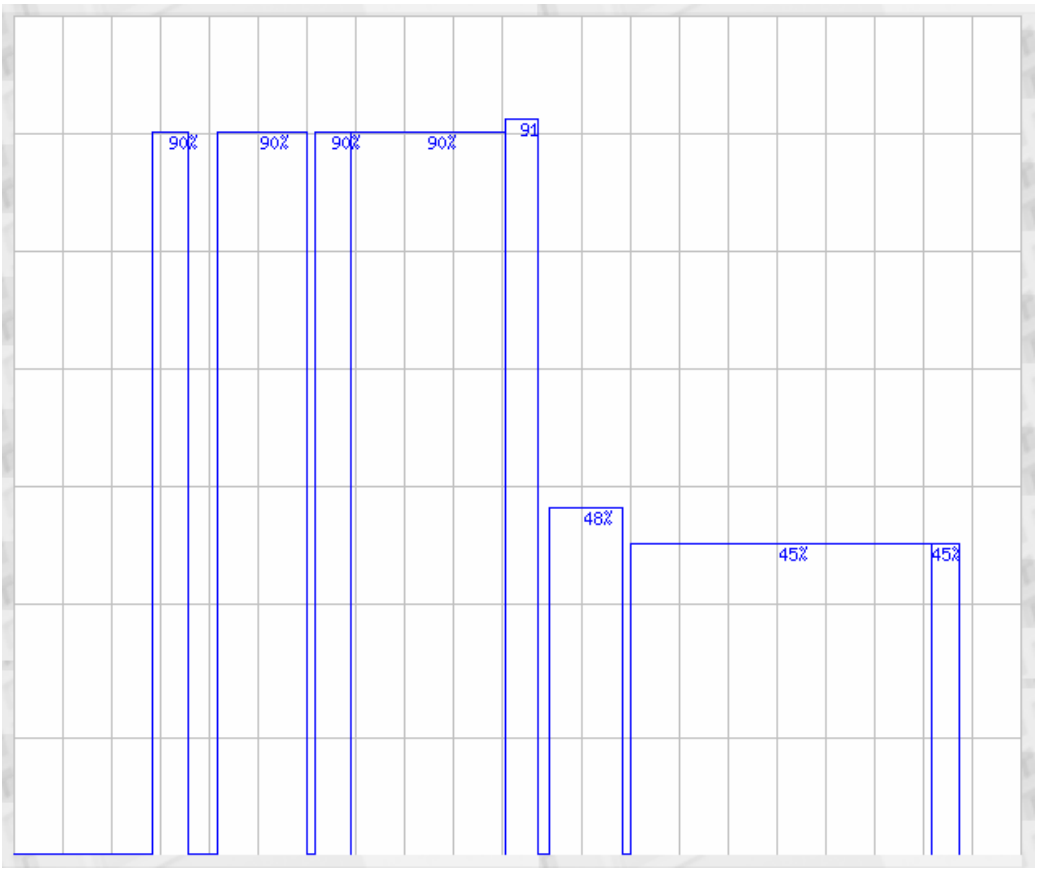


Fig. 4 - Aspecto do cronograma

Através de querys feitas à BD, focando-se especialmente na tabela de histórico, podemos desenhar as linhas que correspondem ao rendimento médio da máquina na última meia-hora. É impressa um linha por cada vez que a cadência muda, assim como o seu valor no gráfico, por cima da linha correspondente. No caso da máquina se encontrar parada ou avariada e o tempo exceder 5 minutos, é impresso o seu estado. Neste último caso, é impresso na página o tempo que a máquina se encontrou parada ou avariada, e ainda o código da mesma paragem. Isto é feito recorrendo a querys feitas à BD nomeadamente à tabela histórico.

Sinóptico

O sinóptico é uma apresentação visual do estado das máquinas em tempo real, que apresenta também a cadência da máquina. O sinóptico é representado, mostrando a máquina, a sua descrição, uma imagem que varia consoante o valor da cadência real e o valor obtido da tabela máquina.

Decidimos, para uma boa percepção, que a visualização da cadência através de uma imagem poderia ser interessante, para isso fizemos um balizamento de valores e atribuímos cores diferentes consoante o valor da cadência.

Caso a máquina esteja parada ou avariada, aparecerá um sinal vermelho, caso a cadência se encontre abaixo dos 50% o sinal apresenta uma cor alaranjada, entre 50% e 80% o sinal apresenta um valor amarelado. Para cadências acima dos 80% o sinal apresenta um tom esverdeado.

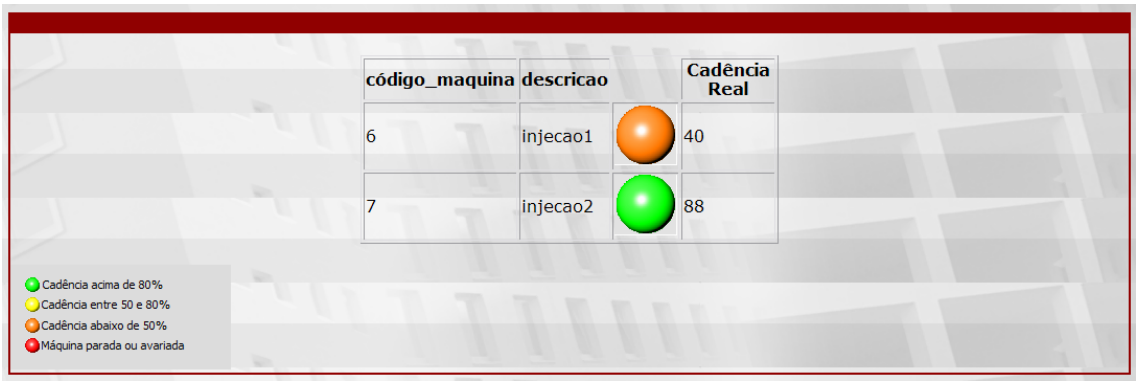


Fig. 5 - Sinóptico

Indicadores

Os indicadores são uma forma de retirar informação sobre o processo produtivo e foram implementados dois tipos de indicadores. Um dos tipos designado por indicadores estáticos e o outro de indicadores estatísticos. Esta pesquisa é feita recorrendo às tabelas máquina e molde na BD.

Os indicadores estáticos apresentam informações sobre tempos de utilização e números de utilização de máquinas e de moldes.

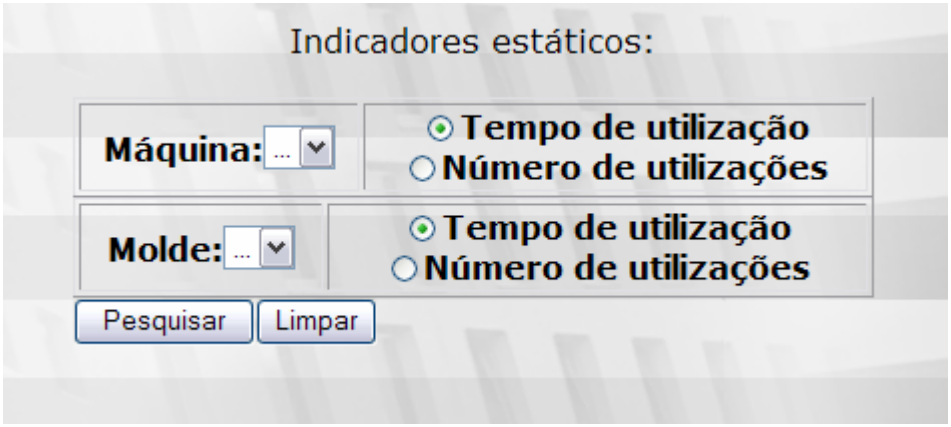


Fig. 6 - Indicadores estáticos

Quanto aos indicadores estatísticos, estes apresentam uma informação mais complexa. Enquanto o primeiro quadro de pesquisa (em baixo) mostra cadências produtivas de máquinas, operadores, peças e moldes, o segundo quadro, trata de informação cruzada entre máquina/operador, máquina/molde e operador/peça. Neste último caso é feita uma pesquisa entre várias tabelas das quais retiramos informação para apresentação e cálculo da cadência para cada uma das situações.

As quantidades totais de peças produzidas são calculadas através da soma de cada quantidade de peças da tabela “Ordem de Trabalho” enquanto que os tempos de operação pela soma dos tempos da tabela “Históricos”.

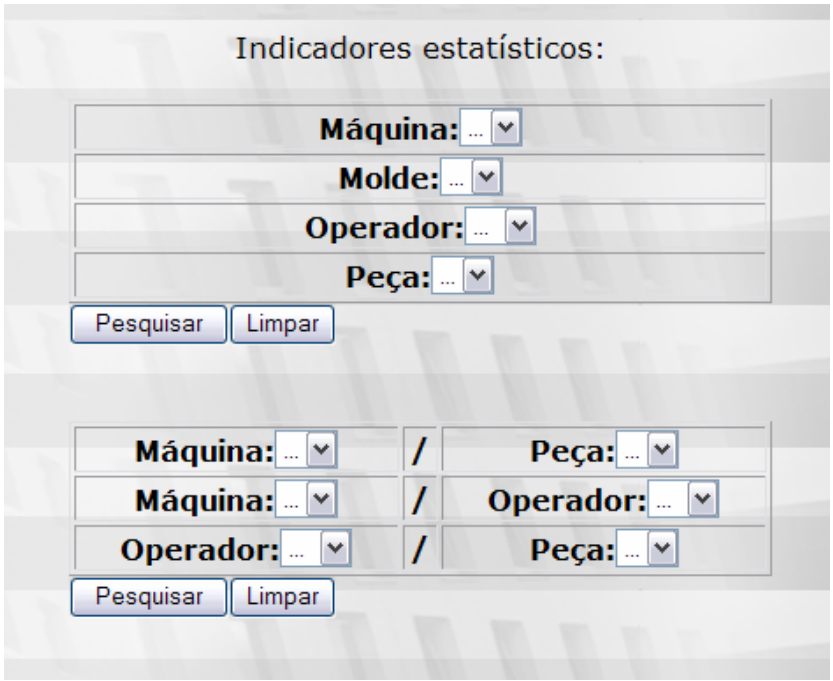


Fig. 7 - Indicadores estatísticos

Esta última pesquisa cruzada é feita através do recurso às tabelas máquina, peça, operador, histórico e ordem de trabalho.

Máquina	Descrição	Peça	Descrição	Rendimento
6	injecao1	2	porca	19.477611940299peças/m

Voltar

Fig. 8 - Exemplo de pesquisa cruzada

Operador

A aplicação ao nível do operador foi criada tendo em conta a simplicidade e as funcionalidades disponíveis, tentando ser intuitiva e prática na sua utilização.

O colaborador quando chega ao seu posto de trabalho regista-se (inserindo o nome e a password). De seguida, o sistema verifica a sua existência e as suas permissões (tabela "Utilizador"). Caso se trate de um operador encaminha-o para a página de operador onde o mesmo pode verificar o seu ranking pelo número de peças produzidas (somatório de todas as ordens de trabalho por ele realizadas) e escolher um lote em espera para iniciar a sua ordem de trabalho.

Para cada lote o operador tem a opção de escolher um dos moldes que produzem aquela peça. Para cada peça podem existir um ou vários moldes, sendo só disponibilizados os moldes compatíveis com a peça a produzir.

Depois de escolhido o molde o sistema insere uma nova ordem de trabalho dando o seu início com a marcação da data e hora inicial, ficando pronto para contabilizar as peças, informar o operador da respectiva cadência e sinalizar as paragens, assim como sinalizar uma escolha errada de peça.

Caso as paragens sejam superiores a 5 minutos a aplicação pede ao utilizador o respectivo motivo de paragem, apresentando pequenos formulários onde o operador escolhe o motivo da paragem. Os formulários desaparecem depois da paragem ter uma causa atribuída.

A aplicação também alerta o operador se a peça do lote escolhido for diferente da peça que está a ser produzida na realidade.

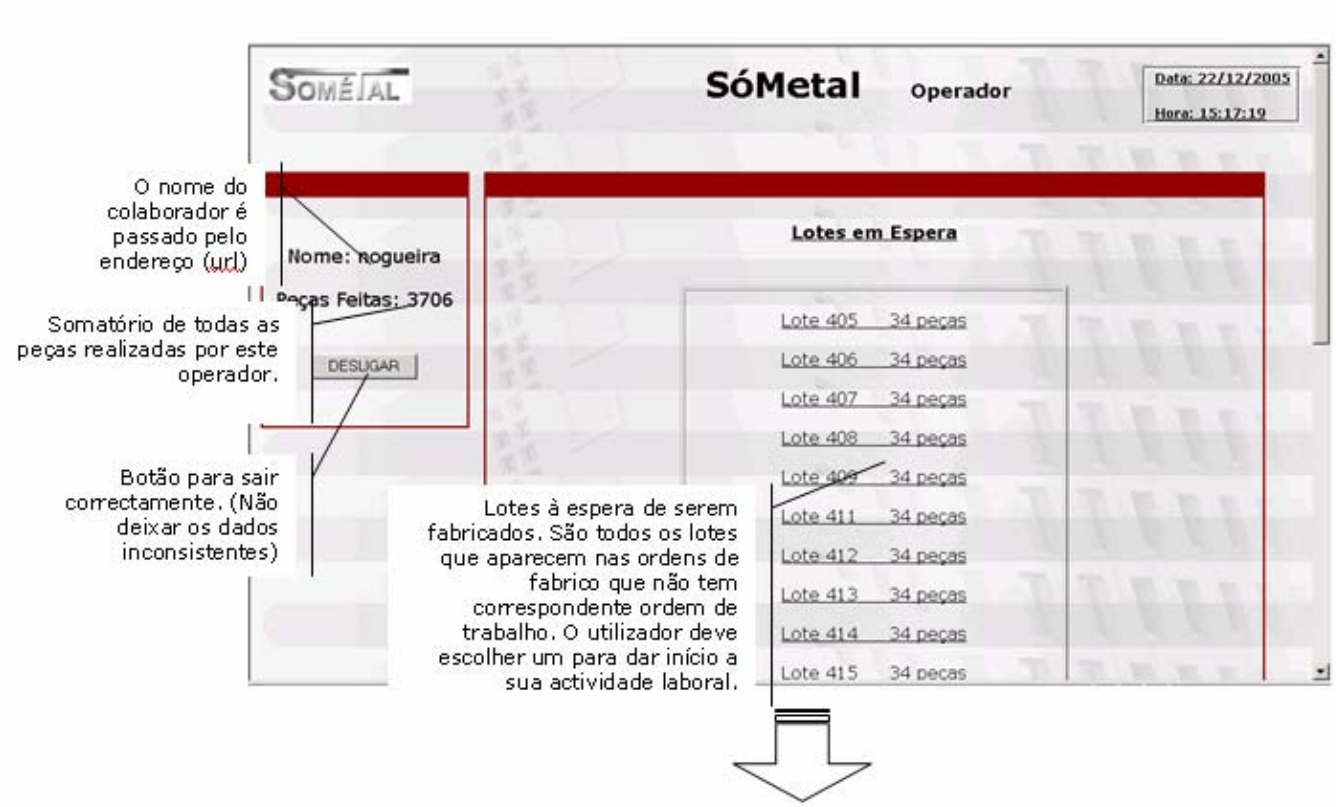
Para terminar o lote o operador selecciona *terminar lote,* sendo este fechado na base de dados, inserindo a data e hora final na ordem de trabalho.

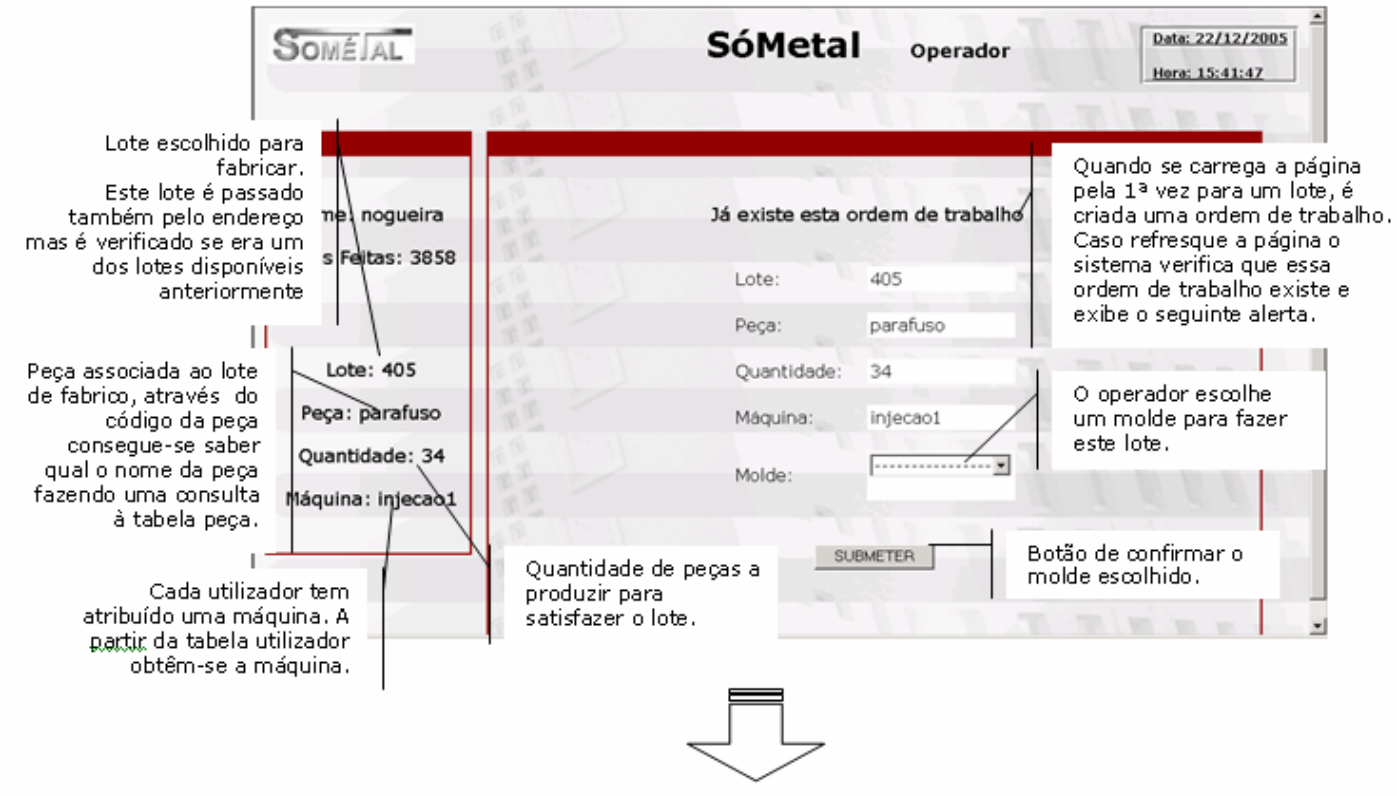
Todas as páginas web do operador são refrescadas a cada 60 segundos.

Restrições

Para a página web do operador funcionar correctamente o operador não pode alterar o *url* nem fechar o programa a meio de um lote de fabrico. Para tal teria de se ocultar todas as barras de ferramentas inclusive a barra de título.

Utilizaram-se algumas protecções contra este problema, como a verificação de alguns dados do *url*, validando a existência do lote e a verificação, no *login*, da existência de alguma ordem de trabalho iniciada e não concluída para o utilizador, o que implica que o utilizador entrará automaticamente nessa ordem de trabalho.


SóMetal
Operador
Data: 22/12/2005
Hora: 15:17:19
O nome do colaborador é passado pelo endereço (url)
Nome: nogueira
Peças Feitas: 3706
Somatório de todas as peças realizadas por este operador.
DESUGAR
Botão para sair correctamente. (Não deixar os dados inconsistentes)
Lotes em Espera
Lote 405 34 peças
Lote 406 34 peças
Lote 407 34 peças
Lote 408 34 peças
Lote 409 34 peças
Lote 411 34 peças
Lote 412 34 peças
Lote 413 34 peças
Lote 414 34 peças
Lote 415 34 peças
Lotes à espera de serem fabricados. São todos os lotes que aparecem nas ordens de fabrico que não tem correspondente ordem de trabalho. O utilizador deve escolher um para dar início a sua actividade laboral.


SóMetal
Operador
Data: 22/12/2005
Hora: 15:41:47
Lote escolhido para fabricar. Este lote é passado também pelo endereço mas é verificado se era um dos lotes disponíveis anteriormente
Já existe esta ordem de trabalho
Lote: 405
Peça: parafuso
Quantidade: 34
Máquina: injecao1
Molde:
Quando se carrega a página pela 1ª vez para um lote, é criada uma ordem de trabalho. Caso refresque a página o sistema verifica que essa ordem de trabalho existe e exibe o seguinte alerta.
Peça associada ao lote de fabrico, através do código da peça consegue-se saber qual o nome da peça fazendo uma consulta à tabela peça.
Lote: 405
Peça: parafuso
Quantidade: 34
Máquina: injecao1
O operador escolhe um molde para fazer este lote.
Cada utilizador tem atribuído uma máquina. A partir da tabela utilizador obtêm-se a máquina.
Quantidade de peças a produzir para satisfazer o lote.
SUBMETER
Botão de confirmar o molde escolhido.

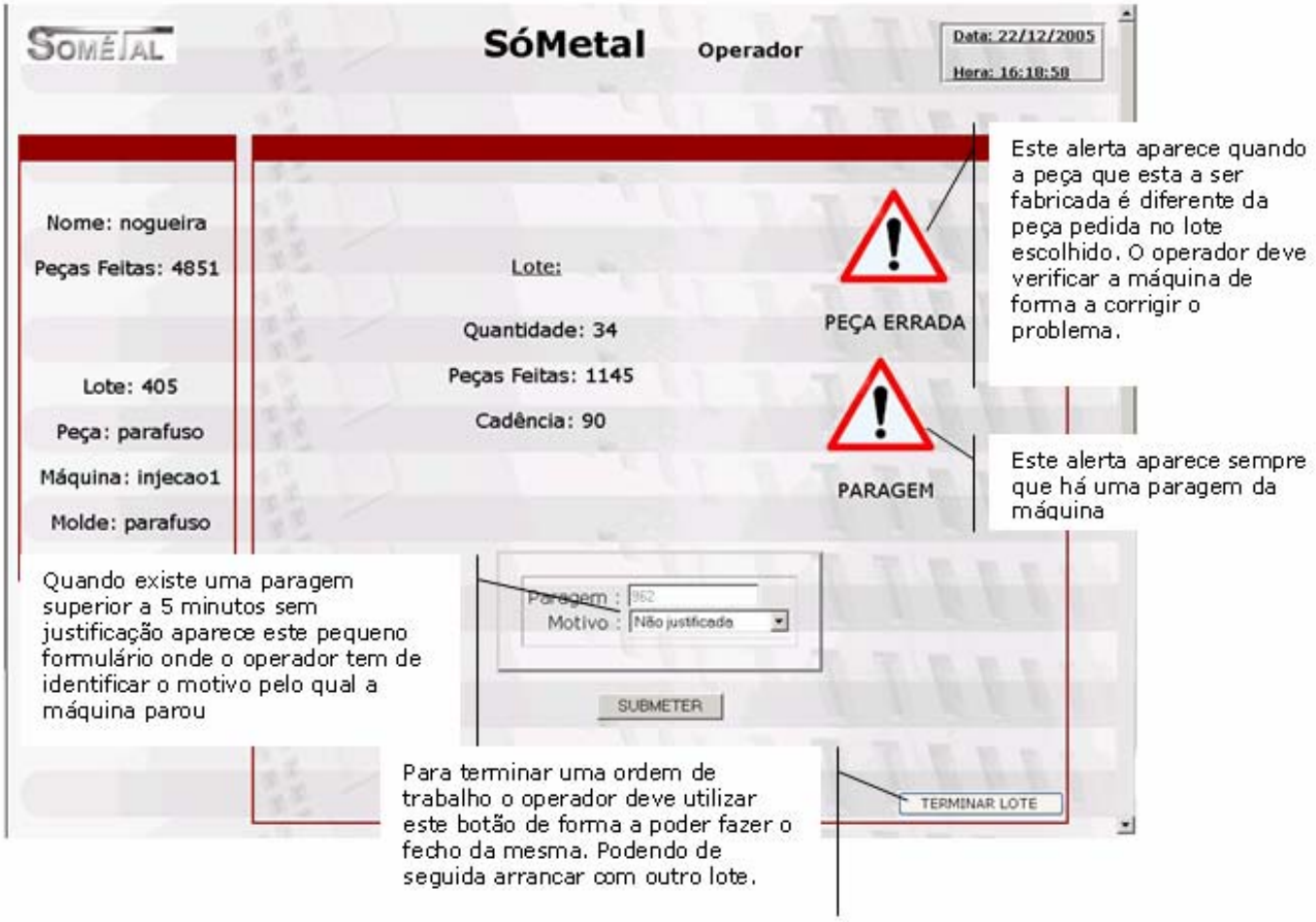


Fig. 9 - Layout da interface com o operador

5.4. Aplicação Lazarus

Uma vez que o cálculo da cadência é feito por esta aplicação, convém salientar que não utilizamos o tempo nominal de produção especificado no guião, pois devido ao período do sinal disponibilizado pelo kit a cadência varia entre 0 e 2000%.

Optamos então por medir o tempo nominal de produção de uma peça. Com o auxílio de um cronómetro, medimos o número de peças produzidas num dado espaço de tempo.

Introdução

Nesta parte do trabalho é feita a actualização, na Base de Dados, dos dados provenientes do kit e tratados pelo autómato, sendo também feita a actualização no autómato de dados provenientes da BD.

A aplicação foi desenvolvida em Lazarus, para a comunicação com a Base de Dados foi necessário instalar o componente Zeos (disponível em: http://sourceforge.net/projects/zeoslib).

Introdução ao componente Zeos do Lazarus

Para fazer uma aplicação com este componente é necessário proceder da seguinte maneira, arrastando para a form:

- **ZConnection** 
  - Colocar User, Password, Host, Port e Protocol nas propriedades da Zquery (No caso presente: sie5, sie5, paranhos.fe.up.pt, 5432 e postgresql-7.4)
  - Colocar Connected a True.

- **ZQuery**  (não confundir com ZReadOnlyQuery).
  - Colocar a Connection para a ZConnection activa (ZConnection1).
  - Colocar na propriedade SQL por exemplo **SELECT * FROM peca**;
  - Colocar Active a True.

- **DataSource**  do tab **[Data Access]**.

o Colocar a propriedade DataSet para a ZQuery activa (Zquery1).

- **DBGrid**  do tab **[Data Controls]**

o Colocar Datasource para a DataSource activa. (Datasource1)

o Se tudo correu bem o resultado da Query aparece nesta tabela.

Existem duas formas de efectuar uma Zquery à base de dados:

- Na propriedade SQL da ZQuery
- Por código

Abaixo indicado está um exemplo dos comandos indispensáveis para pesquisa na base de dados:

```
ZQuery1.Fields.Clear; //limpa eventual lixo de outra query
ZQuery1.SQL.text:='Select * from peca;'; //é indicada a query que se pretende realizar
ZQuery1.Open; //a query é enviada para a Base de dados
ZQuery1.First; //a query é feita na Base de Dados
ZQuery1.Close;
//Para ir buscar o resultado de cada campo da query devolvido pela base de dados utiliza-
//se:
ZQuery1.Fields[i].AsString //em que i representa o índice do campo que se pretende
manipular
```

5.4.1. Desenvolvimento da aplicação para a troca de dados Autómato-BD

Na tabela seguinte são apresentadas as variáveis que são recolhidas na BD e as variáveis da BD que precisam de Update.

Nota: o código desenvolvido para a máquina 1 é idêntico ao da máquina 2, por isso só se refere a máquina 1 e suas variáveis.

Variáveis requeridas à BD	Significado	Variáveis actualizadas na BD – Tabela da BD	Significado
temp_nominal	Indica o tempo de ciclo nominal para uma dada peça	Total_pecas – MAQUINA	Indica o número total de peças produzidas na máquina
Cod_trabalho	Indica o código de trabalho de determinada ordem de trabalho para uma determinada máquina	Total_tempo - MAQUINA	Indica o tempo desde que a máquina está a produzir
Cod_molde	Indica o código do molde actual presente na máquina	Estado – MAQUINA	Pode tomar três estados: Parada, em Funcionamento e Avariada
Cod_maquina	Indica o código da máquina, cujas propriedades se pretendem manipular	Cadencia_real - MAQUINA	Indica a cadência da máquina actualizada nos últimos cinco minutos
		Total_util – MOLDE	Indica o nº total de utilizações do molde
		n_pecas – MOLDE	Indica o nº total de peças produzidas nesse molde
		Total_temp - MOLDE	Indica o tempo total desde que o molde começou a ser usado
		Peca_real – ORDEM_TRABALHO	Indica a peça que realmente está a ser produzida numa dada máquina
		Pecas_feitas – ORDEM_ TRABALHO	Contém o número de peças que foram efectuadas para uma dada ordem de trabalho

		Tempo – HISTÓRICO	Contém o tempo que a máquina permanece num dado estado
		Cadencia - HISTÓRICO	Contém a média da cadência da última meia hora, caso a máquina esteja a funcionar à mais de 30 minutos, ou 0 caso contrário.

Tabela 3 - Variáveis actualizadas e acedidas na base de dados

Máquina de estados da actualização da base de dados

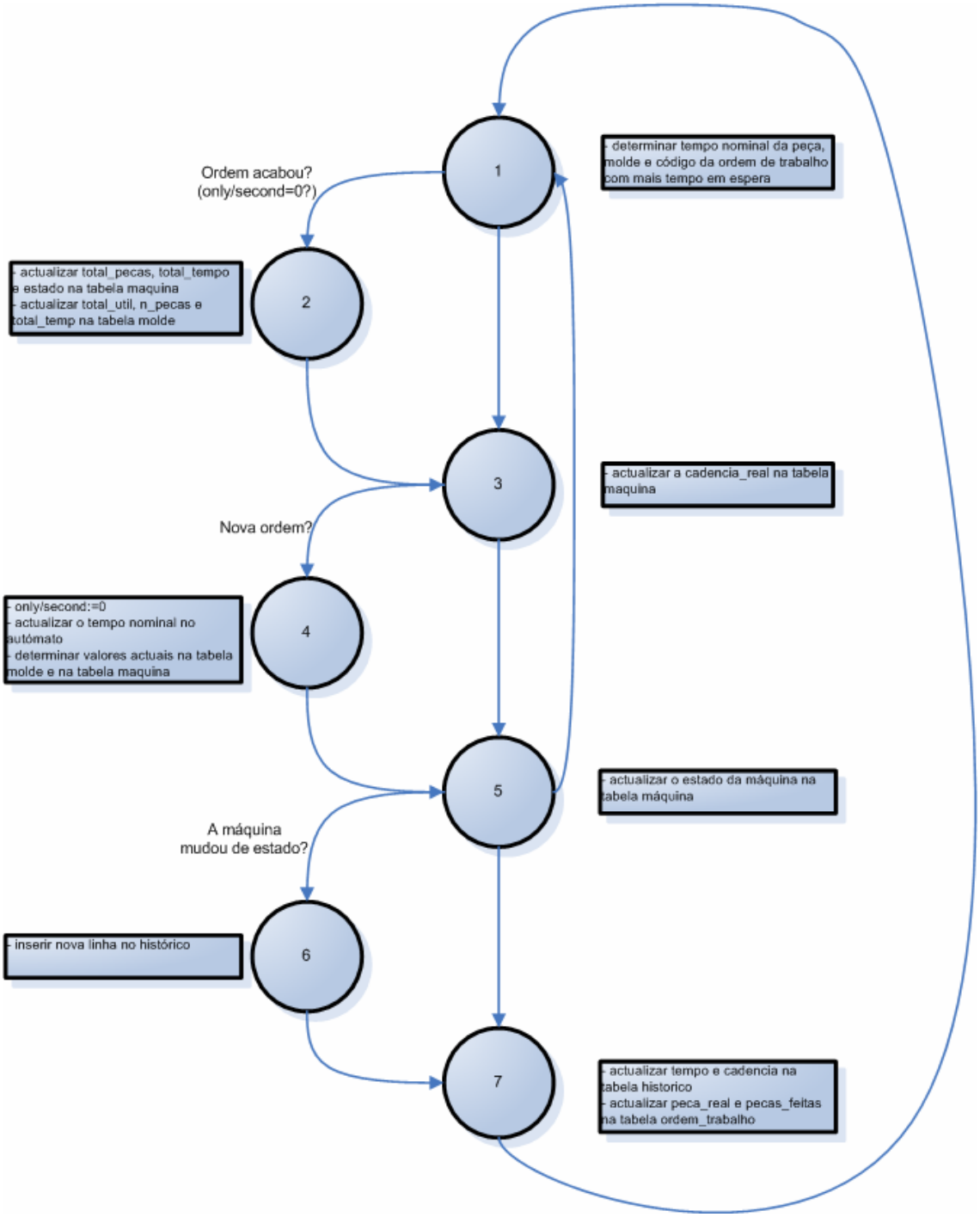


Fig. 10 - Máquina de estados da actualização da Base de dados

Querys feitas na aplicação

1- É feita uma Query à BD para obter o tempo nominal da peça que a máquina 1 está a produzir através de:

```
ZQuery1.SQL.text:='Select temp_nominal from peca where cod_peca='+inttostr(peca_1)+';';
```

1.1- O tempo nominal obtém-se fazendo:

```
temp_nom_m1:=strtoint(ZQuery1.Fields[0].AsString);
```

```
memo3.append('seleccionar temp_nominal');
ZQuery1.SQL.text:='Select temp_nominal from peca where cod_peca='+inttostr(peca_1)+';';
ZQuery1.Open;
ZQuery1.First;
edit18.text:=inttostr(temp_nom_m1);

if ZQuery1.Fields[0].AsString<>'' then
begin
temp_nom_m1:=strtoint(ZQuery1.Fields[0].AsString);
end;
```

2- Query para obter a ordem de trabalho com maior tempo de espera para a máquina 1

```
ZQuery1.SQL.text:='Select min(cod_trabalho), cod_molde from ordem_trabalho where
cod_maquina=6 and data_final is null group by cod_molde;';
```

2.1 - Obtém-se o código de trabalho para a máquina 1 e o código do molde que está na máquina 1 da query anterior através de:

```
cod_trabalho_m1:=strtoint(ZQuery1.Fields[0].AsString);

cod_molde_m1:=strtoint(ZQuery1.Fields[1].AsString);

numero_ordem_1:= cod_trabalho_m1;
```

2.2 - Se a query devolver os campos vazios, é feito o reset ao código de trabalho e ao código do molde, para que a aplicação entenda que não há ordens pendentes e não actualize nenhum dos campos da base de dados:

```
if (ZQuery1.Fields[0].AsString='') or (ZQuery1.Fields[1].AsString='') then
    begin
            numero_ordem_1:=0;
            cod_trabalho_m1:=0;
     end;
```

```
ZQuery1.Fields.Clear;
   ZQuery1.SQL.text:='Select min(cod_trabalho), cod_molde from ordem_trabalho where cod_maquina=6 and
data_final is null group by cod_molde;';
   ZQuery1.Open;
   ZQuery1.First;
//----------------------------Fim do ciclo para determinar a ordem com maior tempo de espera  para a maquina 1


   if (ZQuery1.Fields[0].AsString<>'') and ((ZQuery1.Fields[1].AsString<>'')) then
     begin
         cod_trabalho_m1:=strtoint(ZQuery1.Fields[0].AsString);
         cod_molde_m1:=strtoint(ZQuery1.Fields[1].AsString);

         numero_ordem_1:= cod_trabalho_m1; //esta variavel está a ser repetida!!!! rectificar isso

         edit13.text:='molde_1: '+ inttostr(cod_molde_m1);
     end;

      if (ZQuery1.Fields[0].AsString='') or (ZQuery1.Fields[1].AsString='') then
      begin
          numero_ordem_1:=0;
          cod_trabalho_m1:=0;
          memo3.append('ei não há ordens de trabalho pendentes');
```

2.3 - Detecção que uma ordem de trabalho da máquina 1 terminou

- Se a query anterior devolver o campo cod_trabalho vazio quer dizer que a ordem de trabalho actual terminou. Actualizam-se as seguintes variáveis:

  numero_ordem_1:=numero_ordem_1_ant;

  cod_trabalho_m1:=0;

- Faz-se o update à tabela máquina do número total de peças produzidas pela máquina 1, do tempo total de produção e do estado actual da máquina 1:

```
ZQuery1.SQL.text:='update maquina SET
total_pecas='+inttostr(num_ciclos_1+total_pecas_maq1)+',
total_tempo='+inttostr(temp_tot_1+total_tempo_maq1)+',
estado='+chr(39)+estado_1+chr(39)+'where cod_maquina=6;';
```

- Faz-se o update à tabela molde do número total de utilizações do molde, do número total de peças produzidas, do número total de utilizações e do tempo que o molde foi utilizado:

```
 ZQuery1.SQL.text:='update molde SET
total_util='+inttostr(num_ciclos_1+total_util_molde1)+',
n_pecas='+inttostr(total_pecas_molde1+num_ciclos_1)+',
total_temp='+inttostr(total_temp_molde1+temp_tot_1) +' where
cod_molde='+inttostr(cod_molde_m1)+';'
```

```
if only=0 then //so actualiza uma vez no fim de cada ordem
   begin

       if ZQuery1.Fields[0].AsString='' then   //faz isto quando a data final não é nula
       begin
           numero_ordem_1:=numero_ordem_1_ant;
           cod_trabalho_m1:=0;
           memo3.append('acabei a ordem');

           if numero_ordem_1<>0 then
           begin
               //-------------------------------------------------para actualizar a máquina 1
               memo3.append('Vou updatar a máquina');
               ZQuery1.SQL.text:='update maquina SET total_pecas='+inttostr(num_ciclos_1+total_pecas_maq1)+',
total_tempo='+inttostr(temp_tot_1+total_tempo_maq1)+', estado='+chr(39)+estado_1+chr(39)+'where cod_maquina=6;';
               ZQuery1.Open;
               ZQuery1.First;
               memo3.append('Num é:   '+ inttostr(pecas_prod_1+total_pecas_maq1));
               //-------------------------------------------------------para actualizar o molde

               memo3.append('Vou updatar o molde');
               ZQuery1.SQL.text:='update molde SET total_util='+inttostr(num_ciclos_1+total_util_molde1)+',
n_pecas='+inttostr(total_pecas_molde1+num_ciclos_1)+', total_temp='+inttostr(total_temp_molde1+temp_tot_1) +' where
cod_molde='+inttostr(cod_molde_m1)+';';
               ZQuery1.Open;
               ZQuery1.First;

           end;
           only:=1
       end;
   end;
```

3- Actualização da Cadência Real da Máquina 1 na BD:

```
ZQuery1.SQL.text:='update maquina SET cadencia_real='+inttostr(cad_real_1)+' where cod_maquina=6;';
```

```
memo3.append('actualizar cadencia real_1'+inttostr(cad_real_1));
   ZQuery1.SQL.text:='update maquina SET cadencia_real='+inttostr(cad_real_1)+' where cod_maquina=6;';
   ZQuery1.Open;
   ZQuery1.First;
```

4- Detecção de uma nova ordem de fabrico (novo molde) na máquina1

Quando começa uma nova ordem a variável numero_ordem_1 é actualizada ficando diferente da variável auxiliar numero_ordem_1_ant, detectando-se assim a introdução de um novo molde

É feito um pedido à BD do código do molde introduzido para a ordem de trabalho actual:

```
ZQuery1.SQL.text:='Select cod_molde from ordem_trabalho where
cod_trabalho='+inttostr(numero_ordem_1)+';';
```

O tempo nominal da máquina 1 é actualizado:

```
temp_nom_m1:=temp_nom_m1*3;
```

O temp_nom_m1 é múltiplicado por 3, e depois fornecido ao autómato para que este detecte durações de ciclo superiores temp_nom_m1 sinalizando uma avaria.

É indicado ao autómato que começou uma nova ordem de trabalho através de:

novo_molde(1,inttostr(temp_nom_m1));

```
if numero_ordem_1<>numero_ordem_1_ant then
   begin

   only:=0;

   memo3.append('ei mudei para a ordem de trabalho '+inttostr(numero_ordem_1));

      ZQuery1.SQL.text:='Select cod_molde from ordem_trabalho where cod_trabalho='+inttostr(numero_ordem_1)+';';
      ZQuery1.Open;
      ZQuery1.First;

      if ZQuery1.Fields[0].AsString<>'' then
      begin
          cod_molde_m1:=ZQuery1.Fields[0].AsInteger;
      end;

      memo3.append('nova ordem c tempo nominal '+inttostr(temp_nom_m1));
      memo3.append('vou mandar reiniciar a contagem do automato do ricardo');
      temp_nom_m1:=temp_nom_m1*3;

      novo_molde(1,inttostr(temp_nom_m1)); //envio de trama para o automato a avisar que entrou nova ordem em
produção na maq1

      index_1:=1; //foi iniciado um novo molde como tal o vector da cadência tem que ser reiniciado
```

4.1- Pedido à BD dos valores iniciais do número de peças e tempo de utilização do molde quando é detectada uma nova ordem de fabrico na máquina 1

```
ZQuery1.SQL.text:='Select total_util,n_pecas, total_temp from molde where cod_molde='+inttostr(cod_molde_m1)+';';
      ZQuery1.Open;
      ZQuery1.First;

      if ZQuery1.Fields[0].AsString<>'' then
      begin
          total_util_molde1:=ZQuery1.Fields[0].AsInteger;
          total_pecas_molde1:=ZQuery1.Fields[1].AsInteger;
          total_temp_molde1:=ZQuery1.Fields[2].AsInteger;
      end;

      if ZQuery1.Fields[0].AsString='' then
      begin
          total_util_molde1:=0;
          total_pecas_molde1:=0;
          total_temp_molde1:=0;
      end;
```

4.2- Pedido à BD dos valores iniciais do número de utilizações do molde e tempo de utilização da máquina quando é detectada uma nova ordem de fabrico na máquina 1

```
ZQuery1.SQL.text:='Select total_tempo,total_pecas from maquina where cod_maquina=6;';
      ZQuery1.Open;
      ZQuery1.First;

      if ZQuery1.Fields[0].AsString<>'' then
      begin
          total_tempo_maq1:=ZQuery1.Fields[0].AsInteger;
          total_pecas_maq1:=ZQuery1.Fields[1].AsInteger;
      end;

      memo3.append('total_temp_m1 '+inttostr(total_tempo_maq1));
  end;

   ZQuery1.SQL.text:='update maquina SET estado='+chr(39)+estado_1+chr(39)+' where cod_maquina=6;';
   ZQuery1.Open;
   ZQuery1.First;

   numero_ordem_1_ant:=numero_ordem_1;  //actualiza o numero de ordem para a maquina 1

 //-------------------------máquina 1 ----------------------------------------------------------------------------

   edit24.text:='Cod_trab: '+inttostr(cod_trabalho_m1);

   if cod_trabalho_m1<>0 then begin   //implica que existem ordens de trabalho para serem iniciadas
```

5-Quando a máquina muda de estado a tabela histórico é actualizada

```
if estado_1<>estado_antigo_1 then
    begin

      if estado_1='funcionamento' then
      begin
          memo3.append('Vou inserir no histórico funcionando');
          ZQuery1.SQL.text:='insert into historico(tempo,estado,cod_trabalho,cadencia) values
('+inttostr(temp_fun_1)+','+chr(39)+estado_1+char(39)+','+ inttostr(cod_trabalho_m1)+','+ inttostr(cad_1) + ');';
          ZQuery1.Open;
          ZQuery1.First;
      end

      else
      begin
          memo3.append('Limpei o vector de cadência');
          index_1:=1;
          memo3.append('Vou inserir no histórico parado');
          ZQuery1.SQL.text:='insert into historico(tempo,estado,cod_trabalho,cadencia) values
('+inttostr(temp_par_1)+','+chr(39)+estado_1+char(39)+','+ inttostr(cod_trabalho_m1)+',0);';
          ZQuery1.Open;
          ZQuery1.First;
      end;
    end;

    estado_antigo_1:=estado_1;

    //para se determinar qual é a última linha do histórico para poder ser actualizada
    ZQuery1.SQL.text:='select max(H.cod_historico) from historico as H, ordem_trabalho as O Where
O.cod_trabalho=H.cod_trabalho and O.cod_maquina=6;';
    ZQuery1.Open;
    ZQuery1.First;
    if ZQuery1.Fields[0].AsString<>'' then num_max_1:= strtoint(ZQuery1.Fields[0].AsString);
    if ZQuery1.Fields[0].AsString='' then num_max_1:=0;

    // é actualizada a ordem de trabalho
    memo3.append('Vou updatar a ordem de trabalho');
    ZQuery1.SQL.text:='update ordem_trabalho SET peca_real='+inttostr(peca_1)+',
pecas_feitas='+inttostr(num_ciclos_1)+' where cod_trabalho='+inttostr(cod_trabalho_m1)+';';
    ZQuery1.Open;
    ZQuery1.First;

    // o histórico vai sendo actualizado
    if estado_1='funcionamento' then
    begin
        memo3.append('Vou updatar o historico');
        ZQuery1.SQL.text:='update historico SET tempo='+inttostr(temp_fun_1)+', cadencia='+inttostr(cad_1) +' where
cod_historico='+inttostr(num_max_1)+';';
        ZQuery1.Open;
        ZQuery1.First;
    end

    else
    begin
        memo3.append('Vou updatar o historico');
        ZQuery1.SQL.text:='update historico SET tempo='+inttostr(temp_par_1)+', cadencia=0 where
cod_historico='+inttostr(num_max_1)+';';
        ZQuery1.Open;
        ZQuery1.First;
    end;

    end;        //este end acaba o que é feito caso haja uma ordem de trabalhao para a maq 1
```

Variáveis utilizadas

Variável do Lazarus	*Descrição*
estado_1 estado_2	Dependendo do estado da máquina pode tomar os seguintes valores: - ‘parada’, se a máquina não se encontra a trabalhar; - ‘avariada’, se não é detectado o sinal de ciclo de operação; - ‘funcionamento’, se a máquina está o operar normalmente
peca_1 peca_2	Contém o tipo de peça que está a ser laborada: - 1 : peça do tipo A; - 2 : peça do tipo B; - 3 : peça do tipo C; - 4 : peça do tipo D.
num_ciclos_1 num_ciclos_2	Contém o número de ciclos que a máquina efectuou desde que é iniciado um novo molde.
temp_fun_1 temp_fun_2	Contém o tempo que a máquina está a funcionar. Este tempo é inicializado sempre que a máquina pára, avaria ou é introduzido um novo lote.
temp_par_1 temp_par_2	Contém o tempo que a máquina permanece parada, este tempo é inicializado sempre que a máquina entra novamente em funcionamento.
temp_tot_1 temp_tot_2	Contém o tempo que a máquina está a funcionar, este tempo apenas é inicializado quando é introduzido um novo molde, como tal, sempre que a máquina pára ou se avaria, o tempo fica estagnado, continuando a contagem logo que esta retome o funcionamento.
pecas_prod_1 pecas_prod_2	Contém o número de peças que são produzidas num ciclo. Sempre que a máquina pára esta variável é inicializada.
vec_cadencia_1[i] vec_cadencia_2[i]	È um vector com 6 posições, em que cada posição contém o número de peças produzidas nos último(s) 1(5) minuto(s).
cad_1 cad_2	Contém o valor, em percentagem, da relação entre a cadência real e a cadência nominal.

Tabela 4 - Variáveis utilizadas pela aplicação lazarus

5.4.2. Desenvolvimento da aplicação para comunicação com o autómato

function TComunica.convert_to_dec(numero:**string**):integer

Esta função recebe uma string com um valor em hexadecimal (valor este que é recebido na trama que vem do autómato) e converte-o num valor decimal, para tal faz uso dos códigos ASCII dos caracteres.

```
function TComunica.convert_to_dec(numero:string):integer;
var
  i, num_aux, temperatura: integer;
  testar: set of 'A'..'F';
begin
  testar:=['A'..'F'];
  temperatura:=0;
  for i:=1 to length(numero) do begin
     if numero[i] in testar then begin
       num_aux:=((ord(numero[i]))-55);
     end;
     if not (numero[i] in testar) then num_aux:=((ord(numero[i]))-48);

     temperatura:=temperatura*16+num_aux;
  end;
    Result:=temperatura;
end;
```

function TComunica.calc_FCS(frame:**string**): **string**;

Esta função é responsável por calcular o FCS, ou seja, o Frame Check Sequence, para tal faz um **xor** entre todos os caracteres que compõem a trama.

```
function TComunica.calc_FCS(frame:string): string;
var
   FCS:integer;
   i:integer;
begin
   FCS:=0;
   for i:=1 to length(frame) do
       FCS:=FCS xor ord(frame[i]);
   Result :=IntTohex(FCS,2);
end;
```

function TComunica.create_message(comando:**string**; pos_mem:**string**; string_message:**string**):**string**;

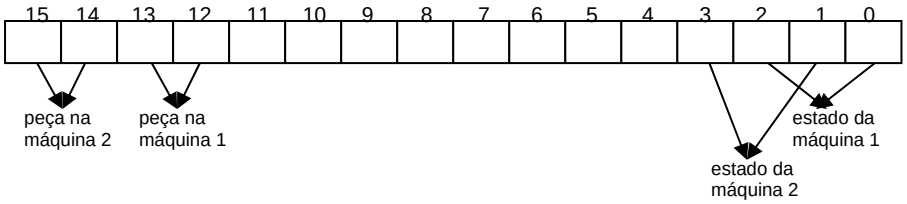
Esta função é responsável por compilar toda a trama a ser enviada ao autómato, para tal recebe três parâmetros: o primeiro indica o tipo de operação que se pretende efectuar, por exemplo leitura 'RD' ou escrita 'WD'; o segundo indica a primeira posição de memória a que se pretende aceder e o terceiro contém os parâmetros

característicos da função utilizada (se for leitura indica o número de posições que se pretende ler, se for escrita contém os valores que se pretende escrever). Aos parâmetros característicos de cada trama a função acrescenta o cabeçalho '@00', o FCS e a terminação '*[CR][4]' A trama retornada pode então ser enviada para o autómato pela porta série.

```
function TComunica.create_message(comando:string; pos_mem:string; string_message:string):string;
var
    temp:string;
begin
        temp:='@00'+comando+pos_mem+string_message;
        temp:=temp+calc_FCS(temp)+'*'+char(13);

        Result:=temp;
end;
```

Esta função é responsável por determinar o estado de cada uma das máquinas, bem como o tipo de peças que estão a laborar, sendo necessário efectuar máscaras para seleccionar os bits, segundo o esquema que a seguir se apresenta:




```
procedure TComunica.processa_estado(estado_actual:integer);
var
   confirma, auxiliar:integer;
begin
    auxiliar:=5;
    confirma:= auxiliar and estado_actual;
    //determinar estado da máquina 1
    if (confirma=4)then estado_1:='parada';
    if confirma=5 then estado_1:='funcionamento';
    if confirma=0 then estado_1:='avariada';
    //determinar estado da máquina 2
    auxiliar:=10;
    confirma:= auxiliar and  estado_actual;
    if confirma=8 then estado_2:='parada';
    if confirma=10 then estado_2:='funcionamento';
    if confirma=0 then estado_2:='avariada';
    //determinar o tipo de peça que está em 1
    auxiliar:=12288;
    confirma:= auxiliar and  estado_actual;
    confirma:=confirma shr 12;
    case confirma of
      0: peca_1:=1;
      1: peca_1:=2;
      2: peca_1:=3;
      3: peca_1:=4;
    end;
    //determinar o tipo de peça que está em 2
    auxiliar:=49152;
    confirma:= auxiliar and  estado_actual;
    confirma:=confirma shr 14;
    case confirma of
      0: peca_2:=1;
      1: peca_2:=2;
      2: peca_2:=3;
      3: peca_2:=4;
    end;
end;
```

procedure TComunica.processa_trama(trama_proc:**string**);

Este procedimento determina se a trama foi recebida correctamente e se é uma trama de leitura ou escrita. Assim, inicialmente verifica se a trama é destinada ao autómato através da leitura do endereço de destino.

Se for uma ***trama de escrita*** verifica se a escrita foi efectuada com sucesso através da análise do código de erro, bem como verifica o FCS da trama recebida.

Se for uma ***trama de leitura***, para além de verificar o código de erro e o FCS, extrai ainda os valores das variáveis que se encontram no autómato (ver tabela 2).

```
 procedure TComunica.processa_trama(trama_proc:string);
var
  auxiliar, trama, FCS_recebido, enviar_temp:string;
  pointer, size:integer;

begin

   size:=length(trama_proc);
   trama:=copy(trama_proc,1, (size-4));
   auxiliar:=(trama_proc[4]+trama_proc[5]);
   if (trama_proc[2]+trama_proc[3])<>'00' then begin
      Memo1.Text:='A trama recebida não é destinada ao autómato'; end;
   if (trama_proc[2]+trama_proc[3])='00' then begin
       if auxiliar='RD' then begin
             FCS_recebido:=trama_proc[size-3]+trama_proc[size-2];
             if (trama_proc[6]+trama_proc[7])<>'00' then begin
               memo1.text:='Ocorreu erro na recepção da trama'
             end else if FCS_recebido<>calc_fcs(trama) then begin
               memo1.append('Erro: diferente FCS');   end
             else begin

                //determinar o estado das máquinas e o tipo de peça
                enviar_temp:='';
                for pointer:=8 to 11 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                estado_maq_aux:=convert_to_dec(enviar_temp);

                //determinar num_ciclos da máquina 1
                enviar_temp:='';
                for pointer:=12 to 15 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                num_ciclos_1:=convert_to_dec(enviar_temp);

                //determinar num_ciclos da máquina 2
                enviar_temp:='';
                for pointer:=16 to 19 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                num_ciclos_2:=convert_to_dec(enviar_temp);

                //determinar tempo de funcionamento da máquina 1
                enviar_temp:='';
                for pointer:=20 to 23 do
                   enviar_temp:=enviar_temp+trama_proc[pointer];
                temp_fun_1:=convert_to_dec(enviar_temp);

               //determinar tempo de funcionamento da máquina 2
                enviar_temp:='';
                for pointer:=24 to 27 do
                   enviar_temp:=enviar_temp+trama_proc[pointer];
                temp_fun_2:=convert_to_dec(enviar_temp);
```

```
                //determinar tempo de paragem da máquina 1
                enviar_temp:='';
                for pointer:=28 to 31 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                temp_par_1:=convert_to_dec(enviar_temp);

                //determinar tempo de paragem da máquina 2
                enviar_temp:='';
                for pointer:=32 to 35 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                temp_par_2:=convert_to_dec(enviar_temp);

                //determinar total de funcionamento da máquina 1
                enviar_temp:='';
                for pointer:=36 to 39 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                temp_tot_1:=convert_to_dec(enviar_temp);

                //determinar total de funcionamento da máquina 2
                enviar_temp:='';
                 for pointer:=40 to 43 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                temp_tot_2:=convert_to_dec(enviar_temp);

                //determinar o número de peças produzidas no ciclo para a máquina 1
                enviar_temp:='';
                 for pointer:=44 to 47 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                pecas_prod_1:=convert_to_dec(enviar_temp);

                //determinar o número de peças produzidas no ciclo para a máquina 2
                 enviar_temp:='';
                 for pointer:=48 to 51 do
                     enviar_temp:=enviar_temp+trama_proc[pointer];
                pecas_prod_2:=convert_to_dec(enviar_temp);

             end;
        end else if auxiliar='WD' then begin
             FCS_recebido:=trama_proc[8]+trama_proc[9];
             if (trama_proc[6]+trama_proc[7])<>'00' then begin
               memo1.text:='Ocorreu erro no envio do Set Value'; end
             else if FCS_recebido<>calc_fcs(trama) then begin
               memo1.text:='Erro: diferente FCS'; end;
       end;
    end;
      processa_estado(estado_maq_aux);
end;
```

Esta função poderia ter sido optimizada substituindo o código que se encontra dentro do **if** que determina os vários parâmetros provenientes do autómato pelo seguinte:

```
                        enviar_temp:='';
               for pointer:=8 to 11 do
                    enviar_temp:=enviar_temp+trama_proc[pointer];
               estado_maq_aux:=convert_to_dec(enviar_temp);

              j:=1;
             for pointer:=12 to 51 do begin
                     enviar_temp:='';
                    for i:=1 to 4 do
                      enviar_temp:=enviar_temp+trama_proc[pointer];
                    variaveis[j]:= convert_to_dec(enviar_temp);
                    j:=j+1;
             end;
```

Note-se contudo que este código se tornaria menos legível, na medida em que as variáveis seriam posições de um vector, como tal, noutras funções não se faria referência a um nome que ilustrasse o conteúdo da variável.

procedure TComunica.maq_estados(caracter:char);

Esta função é responsável pela construção de tramas que cheguem à porta série. O processo é exemplificado através da máquina de estados seguinte:

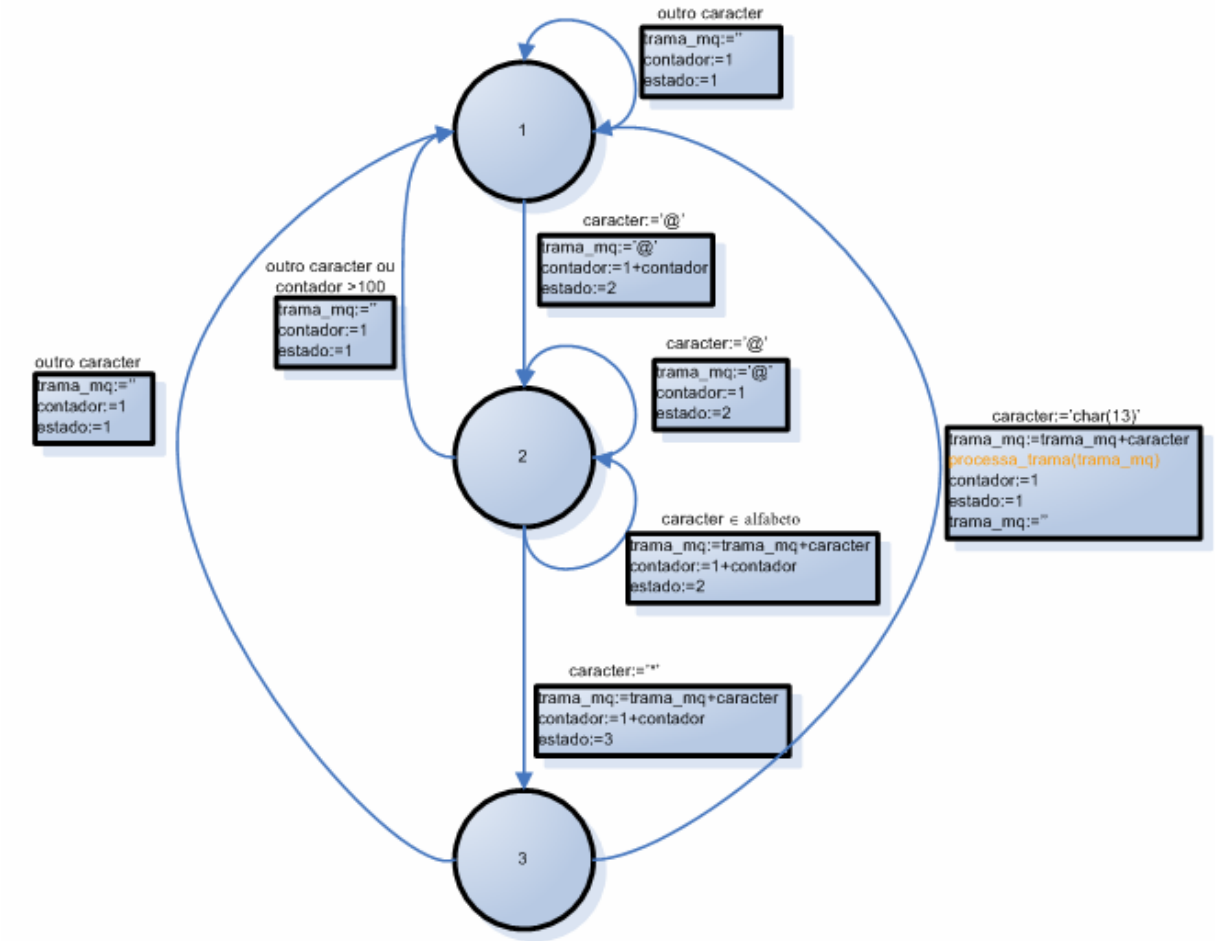


Fig. 11 - Máquina de estados da comunicação com o autómato

```
procedure TComunica.maq_estados(caracter:char);
var
   alfabeto: set of '0'..'z';
begin
alfabeto:=['a'..'f', 'A'..'F', '0'..'9', 'R', 'W', 'T', 'S'];
     case estado of
     1: begin
            if caracter='@' then begin
               trama_mq:='@';
               contador:= contador+1;
               estado:=2;

            end else begin
                trama_mq:='';
                contador:=1;
            end;
      end;

     2: begin
            if caracter='@' then begin
               estado:=2;
               contador:=1;
               trama_mq:='@';
            end
             else if caracter='*' then begin
                trama_mq:=trama_mq+caracter;
                estado:=3;
                contador:=contador+1;
            end
```

```
            else if NOT (caracter in alfabeto) then begin
                estado:=1;
                contador:=1;
                trama_mq:='';
            end
            else if contador>100 then begin
                estado:=1;
                contador:=1;
                trama_mq:='';
            end else begin
                estado:=2;
                contador:=contador+1;
                trama_mq:=trama_mq+caracter;
            end;
     end;
  3: begin
            if caracter<>chr(13) then begin
               estado:=1;
               contador:=1;
               trama_mq:='';
            end else begin
                contador:=contador+1;
                trama_mq:=trama_mq+caracter;
                processa_trama(trama_mq);
                estado:=1;
                contador:=1;
                trama_mq:='';
           end;
     end;
end;
end;
```

procedure TComunica.Timer1Timer(Sender: TObject);

O temporizador foi definido para 5 segundos, assim, de 5 em 5 segundos é enviada uma trama para o autómato para se determinar os valores que se pretendem. É também neste procedimento que é actualizada a BD. No entanto reparamos que se utilizarmos o sistema operativo Microsoft Windows, é necessário temporariamente, parar o temporizador para que não existam conflitos entre a interrupção gerada pelo timmer e o acesso à BD. Com Linux isso já não acontece.

Por fim, é calculada a cadência com que cada uma das máquinas está a operar. Note-se que os valores usados para o teste foram de actualização da cadência de minuto a minuto e visualização de cadência após 6 minutos, ou seja, usou-se um factor de redução de 5 (5min → 1min, 30min → 6min). Desta forma, sempre que passe 1 minuto (5 minutos) é inserido um novo valor no buffer circular que contém, em cada posição, o número de peças efectuadas no(s) último(s) 1 (5) minuto(s). Posteriormente e, logo que o tempo de funcionamento da máquina seja superior a 6 (30) minutos, o rendimento é actualizado com uma periodicidade de 1(5) minuto(s).

Note-se que para se determinar o rendimento é necessário determinar a relação entre a cadência real e a cadência nominal:

- $\text{cadência nominal} = \frac{3600}{\text{tempo nominal}} (peças / hora)$

- $\text{cadência real} = 2 \times \sum_{1}^{6} vec\_cadência[i](peças / hora)$ -> para intervalos de 5 minutos

- $\text{cadência real} = \frac{60}{12} \times 2 \times \sum_{1}^{6} vec\_cadência[i](peças / hora)$ -> para intervalos de 1 minuto

```
procedure TComunica.Timer1Timer(Sender: TObject);
var
    pos_mem_leitura, num_pos, leitura:string;
    aux_1, aux_2:double;
    i, teste: integer;
begin
    aux_1:=0;
    aux_2:=0;
    tempo:=tempo+5;
    tempo_insert:=tempo_insert+5;
    pos_mem_leitura:='0000';
    num_pos:='0011';
    leitura:=create_message('RD', pos_mem_leitura, num_pos);
    Com.WriteData(leitura);
    Timer1.Enabled:=False; //nota foi acrescentado por conflito de acessos entre porta serie e base
dados
    envia_bd;
    Timer1.Enabled:=True; //nota
    if tempo_insert>=tempo_insert_global then begin     // de 5 em 5 minutos o vector que contém a
cadência é actualizado

      teste:=num_ciclos_1;
      vec_cadencia_1[index_1]:=round(teste-num_pec_1_antes);
      cad_real_1:=round((teste-num_pec_1_antes)*temp_nom_m1);
      num_pec_1_antes:=teste;

      if index_1=6 then begin index_1:=1; end else begin index_1:=index_1+1; end;

      teste:=num_ciclos_2;
      vec_cadencia_2[index_2]:=round(teste-num_pec_2_antes);
      cad_real_2:=round((teste-num_pec_2_antes)*temp_nom_m2);
      num_pec_2_antes:=teste;
      if index_2=6 then begin index_2:=1; end else begin index_2:=index_2+1; end;
    //-------------------------------------------------------------------------
    //calcula-se a cadência da máquina, para tal é necessário um buffer circular
    //que vai armazenando os valores da cadência de 5 em 5 min
    // Também de 5 em 5 minutos é necessário efectuar o cálculo da cadência, logo
    // faz-se a soma de todas as posições do vector
              if temp_fun_1*60>=360 then begin            //uma vez passada a 1/2 hora a cadência tem que
ser actualizada no histórico de 5 em 5 minutos
                 aux_1:=0;
                 for i:=1 to 6 do
                    aux_1:=aux_1+vec_cadencia_1[i];
                 aux_1:=aux_1*2;                  //para passar de 1/2 hora para 1h
                 cad_1:=round(0.8*100*(aux_1/(720/temp_nom_m1))); // alterar os 720 para 3600
              end;

             if temp_fun_1*60<360 then begin cad_1:=0; end; //uma vez que ainda não passou 1/2 hora a
cadência aparece com valor nulo

              if temp_fun_2*60>=360 then begin            //não esquecer de alterar aqui o tempo
                aux_2:=0;
                for i:=1 to 6 do
                     aux_2:=aux_2+vec_cadencia_2[i];

                aux_2:=aux_2*2;
                cad_2:=round(0.8*100*(aux_2/(720/temp_nom_m2)));
              end;

              if temp_fun_2*60<360 then begin cad_2:=0; end;

     tempo_insert:=0;//para começar a contagem de 5 minutos
     end;

end;
```

Procedure
TComunica.novo_molde(maq:integer;temp_nominal:string);

Este procedimento é invocado sempre que é iniciada uma nova ordem de fabrico, o que implica um novo tempo nominal para a peça e a inicialização do número de peças produzidas, bem como do tempo de funcionamento do molde introduzido.

```
procedure TComunica.novo_molde(maq:integer; temp_nominal:string);
var
   escrita, pos_inicial, valor_escreve: string;
   escrita_nominal, pos_init, temp_aux:string;

begin

  pos_inicial:='0030';
  if maq=1 then valor_escreve:='0001';
  if maq=2 then valor_escreve:='0002';
  escrita:=create_message('WD', pos_inicial, valor_escreve);
  Com.WriteData(escrita);

  sleep(500);// para garantir que o autómato recebe o novo tempo antes de ser iniciada uma nova ordem

 temp_aux:=inttohex(strtoint(temp_nominal),4);

  if maq=1 then pos_init:='0031';
  if maq=2 then pos_init:='0032';
  escrita_nominal:=create_message('WD', pos_init, temp_aux);
  Com.WriteData(escrita_nominal);

end;
```

procedure TComunica.E_ler_com(Sender: TObject);

Este procedimento é invocado sempre que chega um ou mais caracteres à porta série, ou seja, esta rotina corresponde à rotina de atendimento de interrupção da porta série. Após a recepção dos caracteres é chamada o procedimento que está encarregue de os integrar nas tramas.

```
procedure TComunica.E_ler_com(Sender: TObject);
var
   trama_leitura:string;
   percorrer:integer;

begin
  trama_leitura:=Com.ReadData();
  for percorrer:=1 to length(trama_leitura) do
      maq_estados(trama_leitura[percorrer]);
end;
```

6. Autocrítica

Como é inerente a qualquer trabalho realizado, existem sempre pontos onde se poderia fazer melhor, no entanto, consideramos que uma das decisões que gerou maior “discussão” foi o uso ou não de vistas, optando-se por não recorrer ao uso das mesmas. Isto porque, como referido no ponto 5.2, do ponto de vista da base de dados não foi considerado relevante, dado que a tabela “Ordem de trabalho” permite por si só uma centralização suficiente de informação.

No entanto do ponto de vista da aplicação Lazarus, devido à quantidade de querys efectuadas pela aplicação, concluiu-se que a existência de vistas e das respectivas regras permitiria uma optimização dos recursos por parte da aplicação e mesmo uma divisão de capacidade de cálculo com o Sistema de Gestão de Base de Dados.