Instituto Superior de Engenharia do Porto
Departamento de Engenharia Electrotécnica
Rua Dr. António Bernardino de Almeida 431, 4200-072 Porto

Aplicação Web para Tratamento Estatístico de Dados de Teste

Relatório da Disciplina de Tese/Dissertação do Mestrado em Engenharia Electrotécnica e Computadores - Área de Especialização de Telecomunicações

Nuno Miguel Ferreira Pacheco

Orientadores Isep: Prof.ª Benedita Malheiro
Prof. Manuel Gonçalves Soares
Orientador Preh: Eng.º José Costa

Ano Lectivo: 2007-2008

Resumo

Este relatório apresenta o trabalho realizado no âmbito da Tese/Dissertação do Mestrado em Engenharia Electrotécnica e Computadores - Ramo de Telecomunicações. Este projecto resultou da identificação de uma necessidade do Departamento de Engenharia de *Software* e Testes da empresa Preh Portugal e tem por objectivo criar uma solução/metodologia que permita o tratamento, armazenamento e processamento estatístico dos dados gerados pelo processo de teste de placas electrónicas.

A solução a adoptar deverá permitir armazenar informação e aceder às bases de dados SQL de uma forma simples e segura, possibilitando a sua actualização e manutenção assim como a avaliação de resultados mediante o cálculo de parâmetros diversos, a geração de relatórios e o lançamento semi-automático de pedidos de intervenção. Este sistema será instalado no referido departamento, ficando acessível a todos os seus membros.

Este projecto foi dividido em quatro partes principais:

- Escolha da arquitectura mais indicada para desenvolver a solução, pesquisa das tecnologias mais adequadas e a escolha do ambiente de desenvolvimento;
- Criação de uma aplicação que extrai os dados dos ficheiros de texto gerado pelas máquinas de teste, os converta para um formato de representação apropriado e os armazene em bases de dados MySQL;
- Configuração de um servidor de aplicações, desenvolvimento da aplicação *web*, depuração e teste;
- Avaliação da nova metodologia na empresa.

Foi adoptado um modelo que se integra facilmente no cenário actual da empresa onde: *(i)* todos os utilizadores tem um computador ligado à rede e utilizam

o navegador *Internet Explorer 6* constituindo a camada de interface com o utilizador; *(ii)* a instalação de um servidor de bases de dados MySQL irá permitir armazenar a informação do processo de teste; *(iii)* a instalação de um servidor de aplicações (camada de negócio) do tipo código aberto, irá processar e disponibilizar a informação requerida pelos utilizadores. O servidor de Aplicações seleccionado foi o Apache Tomcat 6 que disponibiliza as tecnologias *Java Server Pages* (JSP), *Javaserver Faces* (JSF) e *JavaServer Tag Library* (JSTL).

Abstract

This report presents the work carried within the scope of the Thesis/Dissertation module of the Master Degree in Electrical Engineering and Computers - Specialization in Telecommunications. It was based on an identified need of the Engineering Department of Preh Portugal and its objective is to develop a methodology to convert, store and process the data generated by the electronic boards testing machines, as well as generate reports and launch semi-automatically intervention requests.

This information should be easily accessed through a web application for historical and statistical processing and report generation for all authorized members of the Department.

This project was divided in four main parts:

- Selection of the most appropriate architecture, research of the available technologies and choice of the development environment (IDE);
- Development of a dedicated application that gathers, converts and saves the data from the text files in a MySQL server database;
- Configuration of an application server, development of the web application, debugging and testing.
- Evaluation of the impact of the conceived methodology.

The solution should integrate easily with the current scenario of the department where: *(i)* the users have their computers connected to a network and use the Internet Explorer 6 browser (interface layer); *(ii)* a MySQL server can be installed to store the information that will be processed by the application (data access layer available), and *(iii)* an open-source application server can be easily installed, (business layer).

Conteúdo

Lista de Figuras

Lista de Excertos de Código

Lista de Pedidos SQL

Glossário

Abreviatura	Descrição	Definição
ISEP	Instituto Superior de Engenharia do Porto	página xiii
SMD	*Surface Mount Devices*	página 1
ICT	*In Circuit Test*	página 1
MDL	*Measurement Data Log*	página 2
EOL	*End of Line Tests*	página 1
HTTP	*HyperText Transfer Protocol*	página 3
TCP/IP	*Transmission Control Protocol/Internet Protocol*	página 5
ARPA	*Advanced Research Projects Agency*	página 5
NCP	*Network Control Program*	página 5
FTP	*File Transfer Protocol*	página 5
CERN	*Centre Européenne de Recherche Nucléaire*	página 5
HTML	*HyperText Markup Language*	página 6
XML	*eXtensible Markup Language*	página 6
SGML	*Standard Generalized Markup Language*	página 6
GML	*Generalized Markup Language*	página 6
ISO	*International Organization for Standardization*	página 6
WWW	*World Wide Web*	página 6
CSS	*Cascading Style Sheets*	página 8
DOM	*Document Object Model*	página 8
DHTML	*Dynamic HyperText Markup Language*	página 8
E4X	*ECMAScript for XML*	página 8
WML	*Wireless Markup Language*	página 8
ECMA	*European association for standardizing information and communication systems.*	página 9
AJAX	*Asynchronous Javascript And* XML	página 9
XHTML	*eXtensible HyperText Markup Language*	página 9
WAP	*Wireless Application Protocol*	página 9
CGI	*Common Gateway Interface*	página 11
API	*Application Programming Interface*	página 12
JSP	*JavaServer Pages*	página 12
WS	*Web Service*	página 12

Abreviatura	Descrição	Definição
JSF	*JavaServer Faces*	página 13
IIS	*Internet Information Services*	página 14
ODBC	*Open Data Base Connectivity*	página 14
RPC	*Remote Procedure Call*	página 14
URI	*Universal Resource Identifier*	página 14
WSDL	*Web Services Description Language*	página 15
UDDI	*Universal Description Discovery and Integration*	página 15
SOAP	*Simple Object Access Protocol*	página 15
SOA	*Service-Oriented Architecture*	página 15
REST	*Representational State Transfer*	página 15
RSS	*Really Simple Syndication*	página 18
JVM	*Java Virtual Machine*	página 19
PDF	*Portable Document Format*	página 20
SDK	*Software Development Kit*	página 20
HTTPS	*Secure* HTTP	página 21
JEE	*Java Enterprise Edition*	página 21
NNTP	*Network News Transfer Protocol*	página 21
SGBD	Sistema de Gestão de Bases de Dados	página 22
SQL	*Structured Query Language*	página 22
JDBC	*Java DataBase Connectivity*	página 22
IDE	*Integrated Development Environment*	página 23
UI	*User Interface*	página 23
JSTL	*JavaServer Pages Standard Tag Library*	página 24
VWP	*Visual Web Pack*	página 24
SMTP	*Simple Mail Transfer Protocol*	página 29
IMAP	*Internet Message Access Protocol*	página 29
POP3	*Post Office Protocol v.3*	página 29
SOP	*Start Of Production*	página 87

Agradecimentos

Gostaria de agradecer, em primeiro lugar, à minha mulher Sónia e à minha filha Carolina pela enorme paciência, aos meus orientadores no ISEP, Prof.ª Benedita Malheiro e Prof. Manuel Gonçalves Soares, ao meu orientador na Preh, Eng.º José Costa, aos colegas Eng.º Armindo Rocha, Eng.º Rui Bessa e aos restantes elementos do Departamento de Engenharia da Preh, aos meus pais, ao meu irmão, aos meus sogros, aos amigos e colegas do ISEP e a todos que de algum modo influenciaram e ajudaram na elaboração deste projecto.

Capítulo 1

Introdução

A Preh Portugal é uma empresa de origem Alemã que se dedica à produção de produtos electrónicos para a indústria automóvel. A sua produção é de larga escala e fornece directamente as principais marcas de automóveis.

A sua sede localiza-se em Bad Neustadt, na Alemanha, onde se centra o desenvolvimento dos produtos. Em Portugal, a empresa desenvolve os processos de fabrico e dispõe de varias áreas tais como injecção plástica, pintura, montagem de *Surface Mount Devices* (SMD), soldadura, montagem, gravação a laser, testes eléctricos e funcionais. Entre os produtos fabricados, destacam-se os painéis para comando de climatização, sistemas de navegação e sistemas de controlo de iluminação.

Todos os produtos são testados, existindo dois grupos principais de testes: os testes de componentes - *In Circuit Tests* (ICT) - e os testes finais - *End of Line Tests* (EOL). Os primeiros são responsáveis por testar as placas electrónicas à saída da colocação dos componentes SMD, detectando não só a presença dos componentes, mas também o seu valor, polaridade e funcionamento. Os EOL são os últimos da cadeia de produção e têm como missão o teste completo do produto, incluindo verificações eléctricas, teste de protocolos de comunicação, a inspecção automática da simbologia e outros pormenores de *design*, a sua programação, validação, etiquetagem e embalagem. É durante este processo que é armazenado um conjunto vasto de informação em ficheiros de texto e em bases de dados, que ficam disponíveis para uma posterior consulta.

1.1 Problemas e Motivação

Na primeira fase de teste, durante os testes ICT, os resultados são armazenados em ficheiros de texto designados *Measurement Data Log* (MDL). Pode ser gerado um ficheiro por cada placa electrónica testada ou, em casos de cadências de teste elevadas, ser gerado um ficheiro por cada dez placas, podendo a placa ter desde cerca de dez até cerca de mil componentes a verificar. Em determinados casos podem ser gerados mais de mil ficheiros diários com menos de 4 kB[1], que facilmente enchem os discos rígidos dos computadores dos dispositivos, tendo de ser guardados regularmente noutros locais.

Este conjunto enorme de informação, que fica guardado em ficheiros, levanta alguns problemas: Como consultar rapidamente a informação com base em critérios específicos? Como verificar a evolução de um determinado parâmetro durante um dado intervalo de tempo? Como ter algum controlo sobre o processo de teste? Como gerar relatórios específicos? Como agilizar o processo de desencadeamento de intervenções correctivas?

A possibilidade de procurar nos milhares de ficheiros de texto uma dada informação é incomportável. Torna-se necessário guardar a informação numa base de dados e criar um meio de aceder aos dados de forma fácil e eficaz.

Para fazer face a este problema, nasceu a ideia de desenvolver uma metodologia que torne a informação de teste acessível a todos os utilizadores autorizados, recorrendo a uma ferramenta existente em todos os computadores - o *browser* ou navegador - que permita processar estatisticamente a informação desejada, gerar relatórios e desencadear acções correctivas.

1.2 Objectivos

A aplicação a conceber destina-se aos elementos do Departamento de Engenharia de Testes e tem por objectivo permitir o acesso à informação de teste armazenada, a detecção de falhas e a geração semi-automática de pedidos de intervenção. Deve constituir-se como uma ferramenta capaz de agilizar o processo de detecção e posterior correcção de falhas. A aplicação deverá ter um funcionamento simples e eficaz. As principais acções a disponibilizar aos utilizadores são:

- Selecção de produtos a consultar;
- Selecção de tipos de dados a consultar dentro de um determinado produto;
- Visualização de resultados;

[1] O sistema de ficheiros utilizado é o NTFS sendo o tamanho de um *cluster* de 4 kB. Qualquer ficheiro com um tamanho inferior reserva esse espaço no disco.

- Detecção de falhas por análise estatística;
- Geração de relatórios;
- Desencadeamento do processo correctivo das falhas;
- Gestão de utilizadores dos dispositivos de teste.

Os requisitos para a implementação desta ferramenta são simples e não têm custos associados: passa pela utilização da rede de dados existente, de um servidor de aplicações HTTP e do estabelecimento de ligações ao servidor MySQL. A nível de *hardware*, apenas será necessário um PC para a configuração do servidor de aplicações HTTP. Todos os restantes recursos serão apenas relacionados com *software*, tendo-se a preocupação de recorrer a *software* livre ou pré-existente na empresa.

O servidor terá de ser registado num domínio da rede, sendo-lhe atribuído um nome, para assim estar acessível a qualquer utilizador do departamento.

1.3 Organização da Tese

A estrutura desta Tese compreende nove capítulos. O Capítulo 1 contém uma pequena introdução assim como os objectivos propostos. O Capítulo 2 apresenta uma breve descrição histórica e descreve algumas das tecnologias utilizadas no desenvolvimento de aplicações *web*. O Capítulo 3 refere-se às tecnologias para o desenvolvimento de aplicações, tais como a plataforma Java, os IDE, os servidores de base de dados e os servidores de aplicações. No Capítulo 4 é descrita a configuração efectuada no ambiente de desenvolvimento para a realização deste projecto, nomeadamente em relação ao servidor MySQL, ao IDE NetBeans e ao servidor de aplicações Apache Tomcat. No Capítulo 5 é apresentada com pormenor a aplicação Java desenvolvida de conversão de ficheiros de texto MDL e armazenamento dos dados no servidor MySQL. O Capítulo 6 descreve as funcionalidades e características de cada página da aplicação *web*. No Capítulo 7 fornecem-se as instruções para instalar a aplicação num servidor Apache Tomcat externo ao IDE. No Capítulo 8 apresenta-se uma análise do impacto da adopção desta nova metodologia de trabalho na empresa. Por último, o Capítulo 9 apresenta as conclusões e propõe algumas optimizações.

Capítulo 2

Estado da Arte

Neste capítulo descrevem-se algumas das tecnologias que permitem tornar as páginas web mais interactivas e interessantes. Apresenta-se um breve resumo da evolução da Internet e destacam-se algumas das linguagens mais utilizadas na World Wide Web.

2.1 Nota Histórica

A Internet aparece em 1969 após ter sido proposta em 1962 e criada pela *Advanced Research Projects Agency* (ARPA). Era então conhecida por *Arpanet* e era constituída por alguns computadores localizados nas sedes de desenvolvimento do projecto. O protocolo utilizado, do tipo *host-to-host*, era chamado *Network Control Program* (NCP). Em 1972, já existia bastante fluxo de informação entre computadores de algumas Universidades e Institutos de Tecnologia nos Estados Unidos da América (EUA). Estavam em desenvolvimento os protocolos de *e-mail*, *newsgroups*, *Telnet* e *File Transfer Protocol* (FTP). No início da década de 80 aparece o Transmission Control Protocol (TCP/IP) que substituiu o NCP. A *Arpanet* transforma-se então na *Internet*.

Em 1989 aparece a *World Wide Web*, criada por especialistas de computadores no CERN - Organização Europeia para Investigação Nuclear, utilizando o protocolo *HyperText Transfer Protocol* (HTTP) que permitia a publicação e divulgação de trabalhos entre a comunidade científica. É criado o primeiro *browser*, chamado *WorldWideWeb*, seguindo-se-lhe, em 1993, o NCSA *Mosaic* e, em 1994, o *Netscape*.

Enquanto o desenvolvimento tecnológico progredia e permitia às redes de comunicação uma maior capacidade na transmissão de dados, surgiram as aplicações *web* para fazer face aos requisitos crescentes dos utilizadores.

2.2 Tecnologias Web

2.2.1 Linguagens de Anotação

As linguagens de anotação destinam-se, por si só, à criação de páginas estáticas. Estas páginas são servidas pelos servidores HTTP aos navegadores cliente onde serão interpretadas e apresentadas ao utilizador. No entanto, à medida que os computadores ligados à WWW se começaram a vulgarizar, surgiu a necessidade de incluir mais funcionalidades a este tipo de serviço. Apareceram, assim, as tecnologias que permitem criar páginas de conteúdo dinâmico.

2.2.1.1 HyperText Markup Language

A *HyperText Markup Language* (HTML) é sem dúvida a linguagem mais popular da WWW. Apareceu juntamente com a World Wide Web em 1990 e, desde então, pouco se alterou na sua essência. É aqui que o conceito de *tag*, (etiqueta/anotação) se tornou popular. Este conceito provém de uma linguagem já existente e utilizada na Organização Europeia para Investigação Nuclear, o SGML - *Standard Generalized Markup Language*[1]. A utilização de etiquetas ou anotações permite especificar a formatação de secções, subsecções, parágrafos, listas ou tabelas de um modo que, quando uma página é interpretada pelo navegador, o conteúdo é apresentado ao utilizador final com a formatação especificada.

Esta linguagem, como se verá mais à frente, é o formato de saída de uma grande variedade de tecnologias de geração de páginas de conteúdo dinâmico.

2.2.1.2 Extensible Markup Language

A *eXtensible Markup Language* (XML) permite, de uma maneira prática e flexível, formatar informação para ser trocada de forma automática entre aplicações. Utiliza também etiquetas, mas não se limita a um conjunto pré-definido. Podem ser criadas etiquetas específicas, formando assim uma estrutura que pode ser interpretada por qualquer aplicação. A diferença face ao HTML é que o XML não se limita a especificar o formato de visualização, mas pretende descrever a estrutura da informação contida no ficheiro. As etiquetas e campos de dados são definidos

[1]Definida no Standard ISO 8879:1986, é baseada na linguagem GML (Generalized Markup Language) criada pela IBM na década de 60.

pelo utilizador, podendo um ficheiro servir para armazenar dados, ser uma fonte de dados ou para ser exibido, dependendo da aplicação que o utilizar.

2.2.2 Tecnologias Web do Lado do Cliente

2.2.2.1 JavaScript

Esta tecnologia foi desenvolvida pela NetScape para se tornar um padrão para a WWW. Trata-se de uma linguagem de *scripting* (baseada em guiões) que consiste em pequenos programas, geralmente embebidos no código HTML, que têm como objectivo introduzir alguma interactividade nas páginas *web*. Sendo uma linguagem de *scripting*, utiliza uma programação bastante simples e não requer licença para desenvolvimento. É uma boa ferramenta para pessoas que não têm conhecimento de linguagens de programação, tais como os *web designers*, pois a sua sintaxe é bastante simples e fácil de implementar. Com o *JavaScript* podemos facilmente ter campos de texto dinâmicos, efectuar validações do lado do cliente, criar *cookies* e até provocar estragos consideráveis no computador do cliente ao criar *scripts* nocivos ou mal intencionados.

Excerto de Código 2.1 Exemplo de código escrito em *JavaScript*

```
1  <html>
2    <head>
3    </head>
4    <body>
5      <script type="text/javascript">
6        window.alert("Olá Mundo!");
7      </script>
8    </body>
9  </html>
```

2.2.2.2 VBScript

A linguagem *Microsoft Visual Basic Scripting*, mais conhecida por VBScript, aparece por volta de 1994, como um derivado do *Visual Basic* que se encontra agregado ao *Internet Explorer 3.0*. Teve como objectivo tornar as páginas *web* mais activas, de modo a aumentar substancialmente a troca de informação e a dividir o processamento entre a plataforma do utilizador e do servidor. Mantém as características que o *Visual Basic* tinha para o desenvolvimento de aplicações orientadas à WWW. Nos ambientes Microsoft não necessita de instalação, nem de *software* de compilação. No entanto, pode também ser utilizado fora das

aplicações *web* pois, em ambiente Windows, estas aplicações são reconhecidas como executáveis (semelhantes a *batch files* mas mais poderosas).

Tal como o *JavaScript*, o *VBScript* é uma linguagem baseada em guiões. Enquanto o *VBScript* foi inicialmente criado para desenvolver aplicações do lado do servidor, o *JavaScript* foi concebido para executar do lado do cliente. Ambas as linguagens de *scripting* têm o mesmo objectivo, não se podendo falar de melhor ou pior linguagem. Operam de modo semelhante e, embora o *VBScript* possa ser uma linguagem mais simples de aprender, o *JavaScript* é mais parecido com o C++. A sintaxe é diferente como podemos verificar no excerto de código 2.2:

Excerto de Código 2.2 Exemplo de código escrito em *VBScript*

```
1  <html>
2    <head>
3    </head>
4    <body>
5      <script type="VBScript">
6        MsgBox("Olá Mundo!")
7      </script>
8    </body>
9  </html>
```

2.2.2.3 Dynamic Hypertext Markup Language

A *Dynamic HyperText Markup Language* (DHTML) não constitui um padrão, mas sim uma união das tecnologias HTML, *JavaScript* e CSS associadas a um *Document Object Model* (DOM). Com a DHTML é possível alterar as etiquetas HTML e as suas propriedades em função de um evento externo. Por exemplo, mostrar uma determinada informação até que o utilizador clique numa hiperligação ou a movimentação de imagens e texto dentro da página. Existem, no entanto, comportamentos específicos para navegadores diferentes como o Netscape e Internet Explorer. No caso do Netscape é possível enviar tipos de letra de texto codificadas na página para o texto ter sempre o aspecto que o autor deseja. No caso do Internet Explorer, podemos efectuar ligações a bases de dados no servidor através de um *ActiveX*.

2.2.2.4 E4X

O E4X, ou mais precisamente, o ECMAScript para XML, junta as funcionalidades do XML ao ECMAScript[2]. Devido à grande utilização e importância do XML tanto na *web* como em tecnologias móveis (*Wireless Markup Language* - WML),

[2]O ECMAScript é mais conhecido pelos seus dialectos Javascript ou JScript.

tornou-se útil ter uma linguagem que não dependa apenas do Modelo de Objectos de Documentos (DOM). A sua utilização não é simples e é praticamente reservada a programadores. O E4X resolve estes problemas implementando tarefas de programação simples. Para aqueles já habituados ao ECMAScript, a semântica e a sintaxe não apresentam nada de novo. O E4X está especificado no padrão ECMA-357.

2.2.2.5 Wireless Markup Language Script

O *Wireless Markup Language Script* (WMLScript) é a linguagem de *scripting* da *Wireless Markup Language* (WML). A WML é baseada em XML e é a linguagem de anotação utilizada por dispositivos móveis que implementam o *Wireless Application Protocol* (WAP). O WMLScript é também baseado no ECMAScript. A sua principal diferença reside no facto de não ser embebido no documento WML, mas ser colocado num ficheiro separado. Os URL referem-se aos *scripts* dentro do documento WML. Esta tecnologia conta com bibliotecas que contêm funções para o programador.

2.2.2.6 Asynchronous JavaScript and XML

O *Asynchronous JavaScript and XML* (AJAX) não é uma tecnologia nova, mas sim um método de utilização conjunta de tecnologias existentes. Destas tecnologias destacam-se o HTML ou XHTML, o *Cascading Style Sheets* (CSS), *JavaScript*, DOM, XML, XSLT e o objecto `XMLHttpRequest` do JavaScript.

A utilização do AJAX tem um objectivo claro: criar páginas *web* mais rápidas em que alterações parciais não obrigam ao reenvio da totalidade da página por parte do servidor da interface. Quando se necessita apenas de uma pequena informação, por exemplo, obter o valor de um preço, não há necessidade de retornar toda a informação da página juntamente com a informação pretendida. Com o AJAX consegue-se uma maior rapidez nas respostas e um aumento significativo do conforto na navegação. O AJAX é assíncrono porque, enquanto espera por uma resposta do servidor, a página permanece acessível ao utilizador (ver Figuras 2.1 e 2.2).

Neste contexto, o papel do *JavaScript* é fazer pedidos ao servidor e processar as respostas. Poderá ter de interagir com o DOM para a actualização dos campos desejados. A informação recebida do servidor é empacotada num ficheiro XML, sendo facilmente tratada pelo *JavaScript*.

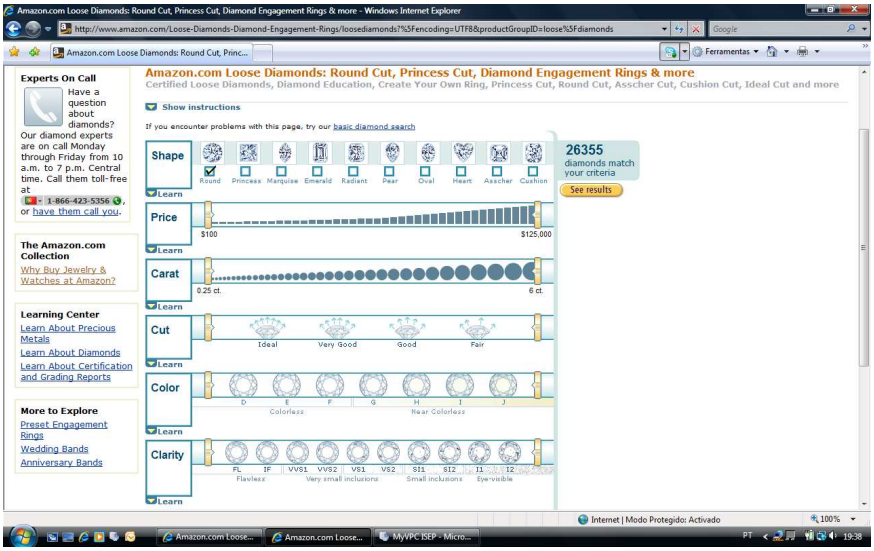

Figura 2.1: Sítio da Amazon Loose Diamonds

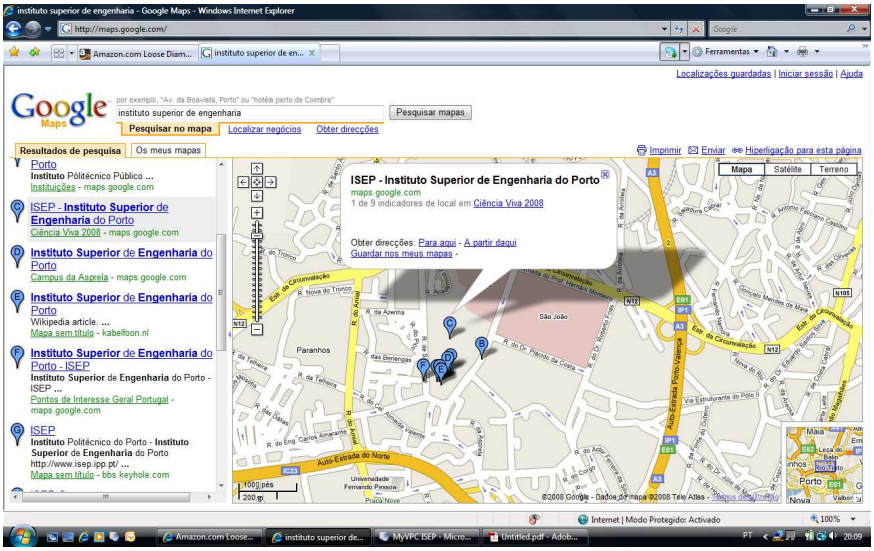

Figura 2.2: Sítio do Google Maps

2.2.3 Tecnologias Web do Lado do Servidor

2.2.3.1 Server Side Includes

As *Server Side Includes* (SSI) são directivas colocadas directamente nas páginas HTML que são avaliadas no servidor quando as páginas são requisitadas por um cliente. Permitem adicionar conteúdo dinâmico a uma página HTML já existente

sem necessidade de compilação. O servidor deve estar configurado para permitir a execução de SSI. Quando é invocada uma página com SSI, o servidor executa o código especificado e devolve a página ao utilizador.

As SSI são uma boa forma de adicionar pequenas porções de código a páginas *web*. Para grandes quantidades de conteúdo dinâmico, esta tecnologia deixa de ser eficaz.

Para a utilização de SSI, os ficheiros HTML têm de ser guardados com a extensão **.shtml*.

2.2.3.2 Common Gateway Interface

Uma das primeiras formas de implementar páginas *web* dinâmicas foi através da *Common Gateway Interface* (CGI). A CGI é uma especificação que permite executar programas no servidor, criando uma interface entre este e aplicações externas. A introdução desta tecnologia permitiu a escrita e leitura de informação de ficheiros ou bases de dados, no âmbito do atendimento de pedidos do cliente, e a devolução do resultado sob a forma de código HTML.

As aplicações CGI podem ser escritas em várias linguagens de programação, sendo as mais comuns o Perl e o C. O Perl, por ser independente do sistema operativo e orientado ao processamento de texto, é muito popular.

A especificação CGI torna-se, no entanto, pouco eficiente quanto são necessárias funções complexas e extensas. Não é possível a reutilização de código com CGI.

Excerto de Código 2.3 Exemplo de código escrito em *Perl*

```
1   #!/usr/bin/perl
2
3   print "Content-type: text/html\r\n\r\n";
4   print "<HTML>\n";
5   print "<HEAD><TITLE>Olá Mundo!</TITLE></HEAD>\n";
6   print "<BODY>\n";
7   print "<H2>Olá Mundo!</H2>\n";
8   print "</BODY>\n";
9   print "</HTML>\n";
10
11  exit (0);
```

2.2.3.3 Servlets

Os *servlets* têm por objectivo criar páginas *web* dinâmicas. Permitem criar ligações entre o navegador do cliente e uma série de aplicações e serviços que, por si só, não estão preparados para trocar informação. E, mesmo que estivessem, não seria conveniente permitir a interacção directa por razões de segurança. Os *servlets* desempenham este papel, agindo como intermediários entre o cliente e as aplicações e serviços *web* (WS).

A tecnologia é disponibilizada através de uma *Application Programming Interface* (API) de desenvolvimento. Trata-se de programas escritos em Java, residentes num servidor *web* que recebem pedidos HTTP de clientes, processam-nos e devolvem um resultado sob a forma de documentos HTML.

A arquitectura típica da tecnologia *servlet* é composta por três camadas distintas: *(i)* o cliente final, *(ii)* a camada de negócio constituída por um servidor *web* ou de aplicações, e *(iii)* os servidores de bases de dados ou outros serviços necessários.

Os *servlets* face às aplicações CGI apresentam as seguintes vantagens:

- O *servlet* corre num só processo até que a aplicação termine sendo cada pedido executado num novo *thread*;
- Os *servlets* são suportados na maioria dos servidores *web*, tornando-se independentes da plataforma utilizada;
- Sendo aplicações Java, são independentes do sistema operativo e mais robustos no tratamento de erros do que as linguagens Perl ou C.

2.2.3.4 JavaServer Pages

Enquanto podemos afirmar que os *servlets* são programas em Java que geram páginas HTML, as JSP são páginas escritas em HTML com excertos de código Java embebidos. O código Java incluído na página é executado no servidor quando o utilizador faz um pedido e é devolvida uma resposta sob a forma de um documento HTML. É também possível incluir *scripts* nas páginas JSP, complementando algumas acções específicas.

As JSP foram criadas para simplificar a construção das páginas, sendo a especificação destas derivada da especificação dos *servlets*. Com JSP podemos alterar os conteúdos da página sem necessidade de recompilação, o que não acontece com os *servlets*. É uma tecnologia mais prática e adequada para aqueles que não têm muita experiência de programação como, por exemplo, os *web designers*. As JSP são, assim, mais orientadas para a apresentação enquanto que os *servlets* são mais orientados ao processamento e a tarefas mais complexas.

No entanto, é possível, consoante a aplicação, utilizar apenas JSP, *servlets* ou conjugar ambos, obtendo aplicações poderosas tanto no âmbito da lógica de negócio (processamento de informação, acessos a serviços e bases de dados), como no âmbito da apresentação dinâmica da página ao utilizador final.

2.2.3.5 JavaServer Faces

Ao longo dos anos, o Java tornou-se numa das linguagens de programação mais utilizadas para o desenvolvimento de aplicações *web*. As tecnologias JSP e *servlet* tiveram um papel fundamental porque tornaram as aplicações mais poderosas e robustas. Mas, com o aumento da complexidade das aplicações, perde-se cada vez mais tempo com o desenvolvimento da interface gráfica. Para dar resposta a esta questão surgiu o *framework JavaServer Faces* que fornece objectos gráficos adicionais para a construção da interface gráfica com o utilizador - *Graphical User Interface* (GUI). Todas as implementações de JSF são suportadas pela tecnologia JSP na camada de apresentação. No entanto, as *JavaServer Faces* permitem utilizar outras tecnologias de apresentação além do JSP, como, por exemplo, a utilização de programação Java. As JSF são desenvolvidas de acordo com o *Java Community Process* (JCP) e pretendem transformar-se num padrão. A arquitectura das JSF permitem aos programadores especializar as classes dos objectos disponíveis para gerar as suas próprias bibliotecas de anotações. Esta tecnologia inclui alguns elementos principais:

- Um conjunto de *Application Programming Interfaces* (API) para representar os objectos GUI, a sua gestão, o tratamento de eventos, validação de dados, acessibilidade e suporte.
- Bibliotecas de anotações JSP para definir a *interface* das JSF na página JSP.

Com este modelo de programação simples e bem definido, esta tecnologia permite introduzir vários objectos GUI numa aplicação por simples arrasto para a página. Faz ligações a fontes de dados e associa os eventos dos objectos do cliente à aplicação no servidor. É destinada a diversos níveis de programadores, desde *web designers* a programadores que fazem a acesso a bases de dados, gestão de eventos e desenvolvem lógica de negócio.

2.2.3.6 Active Server Pages

As *Active Server Pages* (ASP) são uma tecnologia da Microsoft muito semelhante à JSP. Executa código embebido em HTML e o seu funcionamento é em tudo

similar ao do JSP. Permite a inclusão de objectos *ActiveX* para a execução de tarefas complexas. Na sua versão mais recente, o ASP.NET recorre ao *framework* .NET e conta com as linguagens C#, VisualBasic.NET e JScript.NET.

Embora as suas potencialidades sejam enormes e extensíveis com *software* de terceiros, tem a limitação de só correr em ambientes Microsoft e no servidor HTTP IIS *(Internet Information Server)*.

2.2.3.7 PHP:Hypertext Preprocessor

O PHP:*Hypertext Preprocessor* (PHP) é uma linguagem baseada em guiões desenhada para incluir conteúdo dinâmico nas páginas *web*. A sua interpretação e execução é feita no servidor. Esta tecnologia aparece em regime de *open source*, tornando-a bastante atractiva e utilizada. Trata-se de uma boa alternativa às tecnologias JSP e ASP.NET para aqueles que não têm acesso a formação. É embebida directamente no código HTML e as versões mais recentes são mais orientadas a objectos. A sua sintaxe é muito semelhante à do Perl e C.

O PHP é perfeitamente compatível com vários sistemas operativos, trabalha nos servidores HTTP da Apache e da Microsoft e suporta um grande número de bases de dados (MySQL, Oracle, Generic ODBC, *etc.*).

2.3 Serviços Web

Uma forma das aplicações comunicarem automaticamente entre si na Internet é através de serviços *web*. Os serviços *web* são componentes de *software* identificados por um *Universal Resource Identifier* (URI) cuja interface e ligação é descrita através de XML de forma automática. Estas definições podem ser descobertas por outras aplicações ou programas e permitem a interacção e troca automática de informação sob a forma de documentos XML.

Por exemplo, um portal *web* de informação que mostra aos seus utilizadores o valor das acções de bolsas nacionais e internacionais, faz pedidos aos vários servidores que disponibilizam essa informação no formato de serviços *web*.

Os serviços *web* suportam *Remote Procedure Calls* (RPC) que mostram quais os parâmetros de entrada e saída que o serviço *web* suporta para o utilizador poder invocar os seus métodos e procedimentos. Os serviços *web* podem ser dos seguintes tipos:

- *Remote Procedure Call* (**RPC**) - A interacção é efectuada através da invocação remota de funções ou métodos.

- *Service Oriented Architecture* (**SOA**) - A interacção é realizada através da troca de mensagens (*message oriented services*).
- *Representational State Transfer* (**REST**) - A interacção emula os protocolos do tipo HTTP, permitindo a interacção apenas através de um conjunto de operações padrão bem conhecidas (*e.g.*, *GET*, *PUT*, *DELETE*). O foco reside na interacção entre recursos e não na troca de mensagens ou na invocação de operações/métodos.

Para este sistema funcionar correctamente, foram criadas tecnologias específicas para os serviços *web*, suportando a interoperabilidade entre diferentes plataformas e sistemas operativos.

Os componentes principais de um serviço *web* são o *Simple Object Access Protocol* (SOAP), *Web Service Description Language* (WSDL) e a especificação *Universal Description, Discovery and Integration* (UDDI).

2.3.1 Simple Object Access Protocol

O *Simple Object Access Protocol* (SOAP) é o protocolo de nível de aplicação utilizado pelos serviços *web* para enviar mensagens que contêm documentos XML, recorrendo a várias tecnologias tais como *Simple Mail Transfer Protocol* (SMTP), *HyperText Transfer Protocol* (HTTP) e *File Transfer Protocol* (FTP). Estas características são importantes pois, desta forma, ultrapassa algumas restrições de segurança como as *firewalls*. Geralmente, as mensagens SOAP são trocadas entre clientes e serviços, utilizando pedidos e respostas HTTP. O SOAP pode também ser usado para trocar documentos XML completos ou invocar RPC.

A estrutura de uma mensagem SOAP consiste num envelope que contém um cabeçalho e um corpo. Estes componentes são delimitados por etiquetas XML chamadas respectivamente *Envelope*, *Header* e *Body*.

O *Envelope* deve existir em todas as mensagens. É o elemento raiz do documento XML. Pode conter declarações de *namespaces* e outros atributos relativos à forma como os dados são representados no documento XML (*encoding style*). O *Header* é um cabeçalho opcional, mas, se existir, deve ser o primeiro elemento do *Envelope*. Transporta informações adicionais como, por exemplo, se a mensagem deve ser processada por um nó intermediário (a mensagem pode passar por diversos pontos intermediários durante o seu percurso, até alcançar o destino final).

O *Body* é obrigatório e contém a informação a ser transportada para o destino final. Dentro do *Body* é possível adicionar o elemento opcional *Fault*, usado para transportar mensagens de estado e erros retornados pelos nós intermediários ao processarem a mensagem.

2.3.2 Web Services Description Language

A *Web Services Description Language* (WSDL) é uma linguagem destinada a descrever a interface do serviço, as suas ligações e os protocolos envolvidos. A WSDL especifica três tipos de elementos:

(i) **Definições** - Descrevem em XML os tipos dos dados e das mensagens;
(ii) **Operações** - As acções associadas às mensagens suportadas pelo serviço podem ser dos seguintes tipos:

- *One-Way*: A mensagem enviada não requer resposta;
- *Request/Response*: A mensagem enviada requer uma mensagem de resposta;
- *Solicit Response*: Solicitação de uma mensagem de resposta;
- *Notification*: Envio de uma mensagem de notificação.

(iii) **Ligações ao serviço** - Fazem a associação de um conjunto de operações a um porto associando um endereço de rede.

2.3.3 Universal Description, Discovery and Integration

A especificação *Universal Description, Discovery and Integration* (UDDI) fornece um serviço de registo para serviços *web*, suportando a descrição e descoberta de organizações, negócios, fornecedores de serviços, as listas de serviços que estes disponibilizam e a forma de lhes aceder. O serviço de registos UDDI é muito parecido com a utilização de uma lista telefónica para encontrar uma empresa. Existem três principais directórios do registo empresarial do UDDI:

- *White Pages* - Informações básicas como o nome da empresa, contactos e endereços;
- *Yellow Pages* - Informações sobre categorias empresariais, organizadas por classificações padrão;
- *Green Pages* - Detalhes técnicos relativos aos serviços disponibilizados.

O processo de utilização de um serviço *web* é simples (ver Figura 2.3): o fornecedor do serviço *web* afixa a descrição do seu serviço em WSDL no directório descritor de serviços UDDI. O utilizador envia mensagens para o directório UDDI para localizar o serviço que necessita e saber como efectuar uma ligação. Recebe o ficheiro WSDL de descrição do serviço com toda a informação necessária para interagir com o serviço. O utilizador, baseado na descrição obtida, envia pedidos ao serviço e este retorna as respostas adequadas.

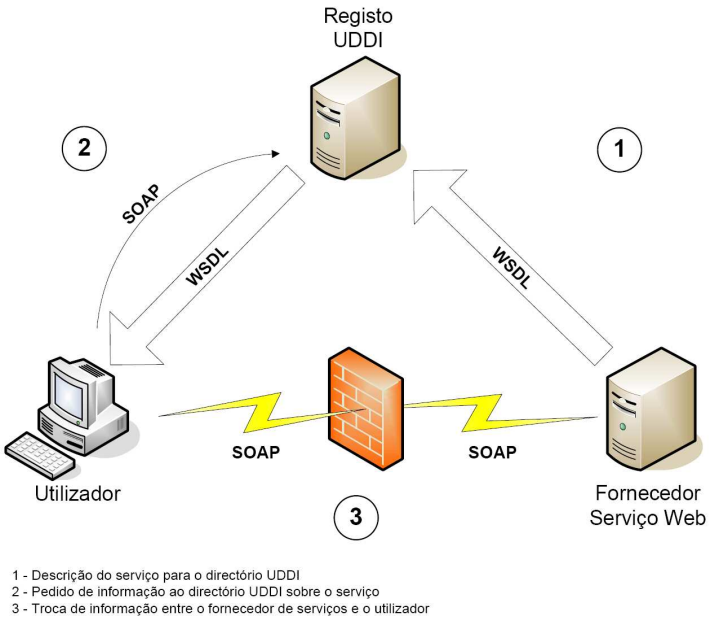

Figura 2.3: Descrição da utilização de um Serviço *web*

2.4 Web 2.0

A *web* de segunda geração, como também é conhecida, aparece como conceito por volta de 2004. A *web* passa a ser vista com uma grande plataforma através da qual os utilizadores usam aplicações *web*, trocam informação e são encorajados a contribuir com conteúdos para as aplicações. Existe uma noção de partilha de conteúdo, informação, processamento e lógica de negócio.

Esta nova visão foi impulsionada pelo crescente aparecimento de serviços e aplicações *web*, gratuitos ou não com funcionalidades tão boas ou superiores a programas instalados nos computadores clientes. Estes serviços recorrem a tecnologias independentes das plataformas utilizadas, desenhadas especificamente para a *web* e com melhoramentos substanciais das interfaces gráficas. As páginas *web*, além do seu próprio conteúdo, começam a ter conteúdo externo, proveniente de outros sítios (calendários, informações do tempo, da bolsa, marcações de viagens, hotéis, *etc.*), podendo estes ser avaliados pela sua frequência de utilização e melhorados por colaboração do utilizador ou por meio de algoritmos de "inteligência artificial".

As características que suportam esta nova visão da *web* são baseadas nos seguintes factos:

- Estima-se que cerca de um bilião de pessoas possuem acesso à Internet em todo o mundo;

- Os acessos de faixa larga permitem uma grande troca de informação;
- A mobilidade dos equipamentos com acesso à rede é crescente, sendo talvez superior às ligações fixas (PC de secretária);
- Uma grande quantidade de utilizadores mantém as suas ligações *on-line* durante grandes períodos de tempo;
- A utilização de serviços *web* tornou-se vulgar e essencial ao mundo empresarial;
- A criação de *Blogs*, *Podcasting* e utilização de RSS cresceu muito;
- A utilização cada vez mais frequente das tecnologias como AJAX, Flash e acessos a bases de dados remotas torna mais rica a interacção;
- A troca de dados formatados em XML permite a interacção automática;
- A utilização do SOAP.

São estas características que criaram as condições para o aparecimentos desta nova era, sendo-lhe atribuído a designação de 2.0 (não como nova versão de uma especificação ou *software*, mas porque é um novo conceito da *web*).

Web 1.0	**Web 2.0**
Refere-se a Leitura	Refere-se a Escrita
Refere-se a Empresas	Refere-se a Comunidades
Modelo Cliente - Servidor	Modelo entre pares
Baseada em HTML	Baseada em XML
Utiliza *Home Pages*	Utiliza *Blog*
Utiliza Portais *web*	Utiliza RSS
Propriedade de conteúdos	Partilha de conteúdos
Utiliza Formulários *web*	Utiliza Aplicações *web*
Custos *Hardware*	Custos Largura Faixa

2.5 Conclusão

Neste trabalho irão ser utilizadas as linguagens de anotação HTML, XML, a linguagem baseada em guiões JavaScript e as tecnologias do lado do servidor dos *servlets*, JSP e JSF.

Capítulo 3

Tecnologias de Desenvolvimento

Este capítulo apresenta algumas das tecnologias existentes para o desenvolvimento de aplicações Web, nomeadamente as utilizadas neste projecto. É feita uma introdução à linguagem de programação utilizada - Java, uma referência a servidores de aplicações e servidores de bases de dados e, por último, a ambientes de desenvolvimento.

3.1 Java

A linguagem de programação Java é uma linguagem de alto nível criada pela Sun Microsystems que pode ser caracterizada pelos seguintes aspectos:

- Orientada a objectos;
- Independente da plataforma utilizada;
- Portável;
- Multitarefa.

O processo de desenvolvimento de um programa escrito em Java é o seguinte: o código fonte é escrito em ficheiros de texto e gravado utilizando a extensão `*.java`. Estes ficheiros são compilados pelo compilador de Java `javac` resultando ficheiros `*.class`. Os ficheiros `class` não contêm código máquina para ser executado pelo processador, mas sim código intermédio (*bytecodes*) para ser lido e interpretado pela Máquina Virtual de Java ou *Java Virtual Machine* (JVM).

A JVM é uma máquina de computação abstracta com um registo de instruções que manipula a memória do sistema em tempo real. O seu objectivo é a portabilidade e o suporte em múltiplas plataformas dos programas Java. Aparece com as versões *Java HotSpot Client* VM, para plataformas cuja utilização seja maioritariamente em aplicações cliente, e a *Java HotSpot Server* VM, para plataformas onde se pretende acelerar o tempo de execução em prejuízo da utilização de memória.

Sendo independente das plataformas físicas, como os sistemas operativos, os ficheiros `.class` são interpretados pela JVM em múltiplos sistemas operativos como o *Microsoft Windows*, *Solaris*, *Linux* ou *Mac OS*.

A Máquina Virtual de Java pertence à plataforma da linguagem Java que, mais do que uma linguagem de programação, consiste numa plataforma baseada em *software*. A plataforma Java é constituída por dois componentes:

- Máquina Virtual de Java (JVM);
- *Application Programming Interfaces* (API) de Java.

As API de Java são uma colecção de componentes de *software*, constituindo o núcleo da linguagem de programação, onde se incluem bibliotecas de classes e interfaces que permitem o desenvolvimento de aplicações com funcionalidades de rede, segurança, acesso a bases de dados, processamento de XML, *etc.*

A Sun disponibiliza esta tecnologia através de dois formatos: *(i)* a *Java Platform Standard Edition Runtime Environment* (JRE) que contém as bibliotecas, a Máquina Virtual de Java e outros componentes necessários para executar aplicações *stand-alone* e *applets*; *(ii)* a *Java Platform Standard Edition Software Development Kit* (SDK) que permite a criação de aplicações e *applets* pois inclui o JRE e as ferramentas de desenvolvimento tais como compiladores e depuradores. Ambos podem ser obtidos de forma gratuita através do sítio da Sun em: *http://java.sun.com/javase/downloads/index.jsp*.

3.2 Servidores de Aplicações

3.2.1 Apache Tomcat

O *Apache Tomcat* é um servidor de aplicações ou contentor *web* disponibilizado em regime de *open-source* e desenvolvido pela *Apache Software Foundation*. Um servidor HTTP é um programa que disponibiliza recursos estáticos (documentos HTML, PDF, *etc.*) em resposta a pedidos de um *browser*. Um servidor de aplicações não se limita a fornecer recursos estáticos. Pelo contrário, destina-se a executar programas e devolver páginas de conteúdo dinâmico. O Tomcat é um

excelente servidor de aplicações pois implementa as tecnologias *Java Servlet* e *JavaServer Pages* da *Sun* e inclui um minisservidor HTTP. Tem grandes vantagens porque, para além de ser gratuito, é suportado por um grande número de plataformas.

As versões disponibilizadas pelo *Apache Tomcat* acompanham as versões da tecnologia *Java*, estando neste momento na versão 6.x que corresponde à tecnologia *Java Servlet* 2.5 e *JavaServer Pages* 2.1. Pode ser feito o seu *download* no sítio oficial da Apache: *http://tomcat.apache.org/download-60.cgi*.

3.2.2 IIS Internet Information Services

Este programa, propriedade da Microsoft, fornece um conjunto de serviços para servidores que utilizem a plataforma Microsoft Windows. É o segundo servidor *web* mais utilizado, depois do Apache. Actualmente, este servidor suporta os serviços de FTP, SMTP, NNTP e HTTP/HTTPS. O desenvolvimento de aplicações é suportado pela tecnologia ASP.NET que se encontram inserida no *framework* .NET, também propriedade da Microsoft.

3.2.3 Sun Java System Application Server

O *Sun Java System Application Server* (SJSAS) é uma plataforma desenhada para fornecer aplicações e serviços orientados para o desenvolvimento do lado do servidor assim como para o desenvolvimento de arquitecturas orientadas a serviços - *Service Oriented Architectures* (SOA) - e Web 2.0. Foi desenvolvido pela *Sun* e faz parte do pacote de *software Java Enterprise Edition* (JEE). Vem incluído no *Java Development Kit* e suporta os ambientes de desenvolvimento integrados *NetBeans* e *Eclipse*.

3.3 Servidores de Bases de Dados

Os servidores de bases de dados são essenciais à criação de aplicações *web*. Estas necessitam de, rapidamente, aceder à informação necessária para o funcionamento da lógica de negócio. A utilização destes servidores tornou-se bastante comum, sendo ferramentas capazes de gerir grandes quantidades de informação. A *Structured Query Language* (SQL) é a linguagem utilizada para interagir com os servidores de bases de dados mais comuns e populares, tornando estes servidores bastante eficazes e versáteis.

Antes de referir os servidores mais utilizados é importante destacar o método de ligação entre a aplicação *web* e estes servidores. Visto que a plataforma de desenvolvimento deste projecto se baseará na tecnologia Java, esta ligação será feita através do *Java Database Connectivity* (JDBC). O JDBC é uma API para estabelecer a conectividade entre uma aplicação Java e uma grande variedade servidores de bases de dados.

Esta API contém dois tipos de interfaces: uma que faz a ligação do lado da aplicação e a outra que faz a ligação ao controlador da base de dados. Neste último, a ligação pode ser efectuada de diversas formas: *(i)* directamente ao servidor de bases de dados; *(ii)* através de um controlador de um fornecedor externo desenvolvido de acordo com a especificação do JDBC; e *(iii)* recorrendo a uma *"bridge"* `JDBC-ODBC` que interliga as duas especificações. Deste modo, esta API é uma das mais utilizadas e suportadas por plataformas e sistema operativos.

3.3.1 MySQL

Desenvolvido pela MySQL AB (adquirida em Janeiro de 2008 pela Sun Microsystems), o MySQL é um sistema de gestão de bases de dados relacional (SGBD) baseado na linguagem *Structured Query Language* (SQL). A *Structured Query Language* é uma linguagem utilizada para armazenar, consultar e actualizar informação numa base de dados relacional.[1] Tem a grande vantagem de se apresentar em código aberto e ser suportado na maioria dos sistemas operativos assim como ser compatível com varias linguagens de programação (Delphi, Java, C/C++, Python, Perl, PHP e Ruby) através dos controladores ODBC, JDBC e .NET.

3.3.2 Microsoft SQL Server

Trata-se de um servidor de bases de dados que implementa a linguagem *Structured Query Language*. Bastante robusto e fiável, é restrito a ambientes Microsoft,

[1] Uma base de dados relacional é uma colecção de dados persistentes armazenados numa ou mais tabelas, a partir das quais se encontram acessíveis de diferentes maneiras, sem ser necessária a sua reorganização.

disponibilizando um controlador ODBC. As linguagens de programação da Microsoft Visual Basic.NET e C#.NET permitem a sua integração directa, sendo bastante simplificada através do pacote de desenvolvimento *Visual Studio 2005*. Outras linguagens de programação também se podem ligar facilmente, como o Java, através da API JDBC. É uma solução bastante utilizada em ambientes empresariais dado o seu desempenho.

3.4 Ambientes de Desenvolvimento

A utilização de sistemas para desenvolvimento de *software* tornou-se vulgar, existindo uma oferta grande de ambientes integrados de desenvolvimento - *Integrated Development Environment* (IDE) - que simplificam o processo de programação e melhoram substancialmente o resultado final. Tradicionalmente, o código era escrito em ficheiros de texto e compilado numa linha de comandos. Neste momento, os ambientes de desenvolvimento integrado para além de integrarem as ferramentas de compilação, interpretação e depuração, apresentam interfaces com o utilizador - *User Interface* (UI) - de grande qualidade, fornecem assistência à verificação da correcta sintaxe da linguagem, completando a escrita de código automaticamente (*auto complete*), permitem a depuração do código linha a linha, fornecem ajudas interactivas com procura na *web*, integração de acessos a bases de dados e são expansíveis com actualizações e componentes externos. Os IDE não permitem criar apenas aplicações mas também definir projectos que constituem aplicações mais complexas.

3.4.1 IDE NetBeans

O NetBeans IDE é um ambiente de desenvolvimento integrado que é disponibilizado em regime de *open source*. Permite o desenvolvimento de vários tipos de aplicações (orientadas à *web*, aplicações *stand-alone*, aplicações para dispositivos móveis, *etc.*). As linguagens utilizadas passam pelo Java, C/C++ e Ruby. Este IDE é suportado pelas plataformas Windows, Linux, Solaris e Mac OS. Integra acessos a bases de dados, ligação com servidores de aplicações (como o Tomcat) e um monitor HTTP que permite monitorizar todo o processo de pedido e resposta, *cookies*, *beans*, *etc.*

Entre as suas capacidades destaca-se o desenvolvimento de aplicações e serviços *web* com o *Java Enterprise Edition* (JEE), aplicações para sistemas móveis, especificação de projectos com *Unified Markup Language* (UML)[2], desenvolvimento e teste de arquitecturas orientadas a serviços *web* (SOA).

[2]O UML permite gerar código fonte através de diagramas.

Na versão 5.5 é possível adicionar um pacote de componentes extra, o *Visual Web Pack* (VWP). Contém elementos importantes para trabalhar com AJAX, *JavaServer Faces*, *Struts*, *JSP Standard Tag Libraries* (JSTL), Java *Servlets* e ligações a fontes de dados. Tem uma funcionalidade muito útil, um editor de *queries* para trabalhar com SQL. O IDE e o *addon* Visual Web Pack podem ser obtidos em: *http://download.netbeans.org/netbeans/6.0/final/*. A última versão, a 6.0, não necessita deste *addon* pois já vem incluído por omissão.

3.4.2 Microsoft Visual Studio 2005

É o IDE criado pela Microsoft para desenvolvimento de aplicações e inclui várias ferramentas de desenvolvimento. Utiliza o *framework* .NET cujo objectivo é servir de plataforma para o desenvolvimento de aplicações com linguagens de programação diferentes. Permite que, independentemente da linguagem ou da plataforma utilizada e desde que o *framework* esteja instalado na máquina de destino, a aplicação possa ser executada.

Esta versão apresenta as ferramentas ASP.NET para desenvolvimento de aplicações *web* e as linguagens *Visual Basic 2005*, inclui a capacidade de execução multitarefa, suporta linguagens como o C#; VisualJ#, o Visual C++, e muitas outras ferramentas para aplicações específicas.

3.4.3 JavaStudio Creator

Foi desenvolvido para a plataforma Java e destina-se em particular às aplicações e portais (*portlets*) *web*. Suporta as tecnologias AJAX, *JavaServer Faces*, *JavaServer Pages* e JDBC. Inclui o servidor de aplicações *Java System Application Server*. Apresenta uma grande colecção de objectos para o desenvolvimento do UI, desde simples campos de texto até objectos mais complexos. Não é um *software open-source*.

3.5 Conclusão

O projecto será realizado no ambiente integrado de desenvolvimento NetBeans, recorrendo à linguagem Java e às tecnologias de desenvolvimento de aplicações associadas, designadamente às disponibilizadas através do servidor de aplicações Apache Tomcat, e ao servidor de bases de dados MySQL.

Capítulo 4

Configuração do Ambiente de Desenvolvimento

Este capítulo descreve os passos da configuração do ambiente de desenvolvimento. Foram escolhidos os componentes (i) servidor de bases de dados MySQL Community Server 5.0.26, (ii) o IDE NetBeans 6.01 para a criação da aplicação e (iii) o servidor de aplicações Apache Tomcat 6.0.14.

4.1 Configuração do Servidor de Bases de Dados MySQL

Antes da instalação de qualquer outro programa ou plataforma, a instalação do servidor de bases de dados deverá ser efectuada, visto que este não depende de programas terceiros ou ambientes de desenvolvimento, mas sim o oposto.

Para a configuração deste servidor foram instaladas as ferramentas:

1. MySQL Server 5.0.26
2. MySQL Tools for 5.0
3. MySQL JDBC Connector Driver 5.1.5

Todas estas ferramentas são *open source* e podem ser obtidas no sítio do MySQL: *http://dev.mysql.com/downloads/mysql/5.0.html*. O MySQL Tools inclui o MySQL Administrator, o MySQL Query Browser e outras ferramentas que permitem configurar o servidor facilmente com uma interface gráfica e intuitiva. Deste modo, foram criadas as seguintes configurações de acesso ao servidor:

Tabela 4.1: Configuração de acesso total ao *MySQL Administrator*:

`Connection: MySQLConnection`
`ServerHost: localhost`
`Port: 3306`
`Username: root`
`Password: rootroot`

4.2 Pré-requisitos para o IDE

Antes da configuração do IDE, é necessária a instalação da plataforma Java - *Java Standard Edition Development Kit* (J2SE JDK). Pode ser efectuada durante a instalação do NetBeans porque vem incluído no pacote fornecido. No entanto, se for necessária uma versão diferente, esta pode ser obtida no sítio da Sun: *http://java.sun.com/javase/downloads/index.jsp*.

A localização do JDK deverá ser acessível a todas as aplicações que dela necessitem: Assim, deve-se adicionar a sua localização às variáveis de ambiente do *Windows*, permitindo a execução do compilador e aplicações Java, independentemente da localização dos ficheiros a executar.

No campo de variáveis do sistema, na variável `"Path"` foi adicionada a localização da plataforma Java: `C:\ProgramFiles\Java\jdk1.6.0_03;.;.`

4.3 Configuração do NetBeans IDE 6.01

Este IDE é instalado facilmente e já inclui os servidores de aplicações *Apache Tomcat 6.0.14* e o *GlassFish V2*. O pacote de instalação pode ser obtido no sítio da NetBeans *http://download.netbeans.org/netbeans/6.0/final/* ou requisitando um CD livre de encargos.

Após a instalação, é recomendada a actualização automática do IDE. Existem também inúmeros *plugins* disponíveis que facilmente se activam ou desactivam. Uma grande vantagem face à versão anterior - a 5.5 - é que, para se trabalhar com *JavaServer Faces*, já não é necessário instalar o *Visual Web Pack*, pois encontra-se incluído por omissão no IDE.

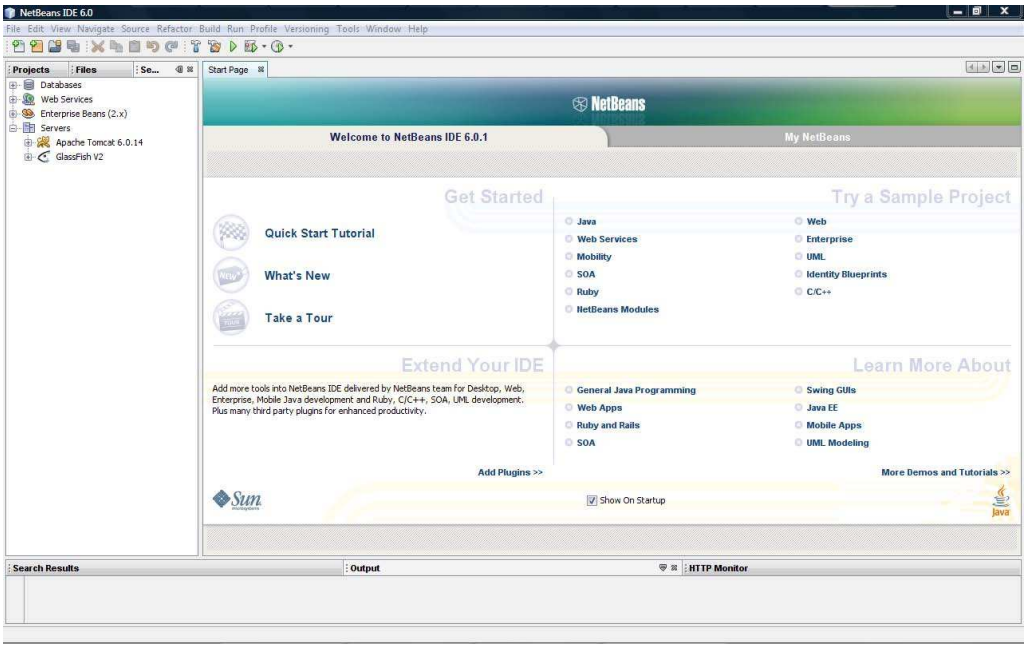

Figura 4.1: Página de início do NetBeans 6.0

4.3.1 Ligação ao Servidor MySQL

Uma das principais acções para o desenvolvimento deste projecto foi a ligação à base de dados. Optou-se pela ligação através do controlador *mysql-connector-java-5.1.6*. Este pacote já vem incluído no NetBeans, podendo-se executar a aplicação dentro do contexto do desenvolvimento das aplicações. Fora deste contexto, é necessária a sua inclusão nos caminhos conhecidos de acesso às classes do projecto. No caso deste projecto, deve-se seguir o seguinte procedimento:

1. Indicar a localização do ficheiro *mysql-connector-java-5.1.6.jar*. No separador dos projectos, clicar com a tecla do lado direito do rato em primeiro lugar em `Libraries`, depois em `Add JAR/Folder`.

2. Configuração da ligação ao MySQL (Figura 4.3). Só assim é possível executar comandos a partir do IDE e associar os dados das tabelas directamente a objectos como os disponibilizados pelo *JavaServer Faces*.

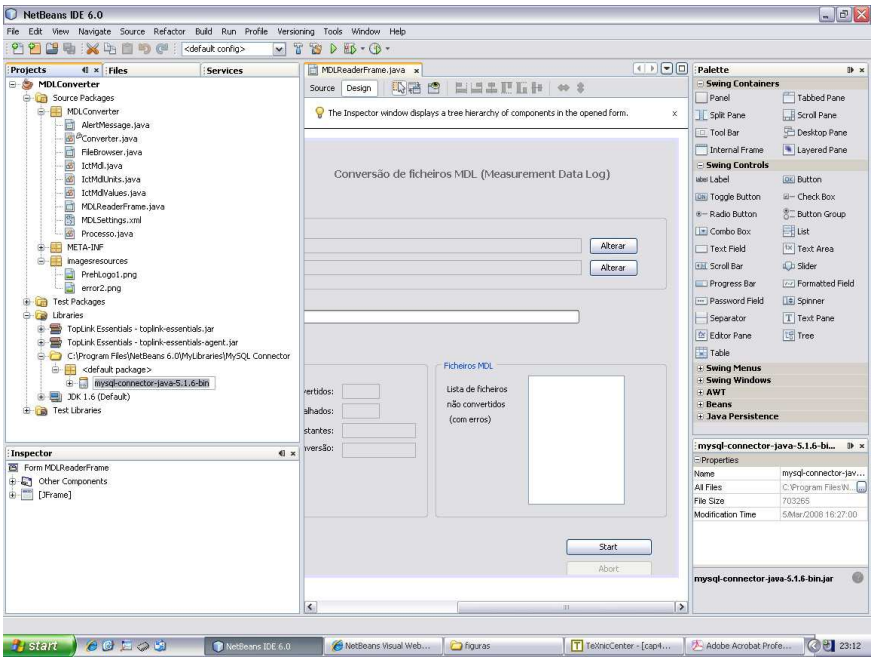

Figura 4.2: Instalação do controlador Java *JDBC* para utilização do MySQL

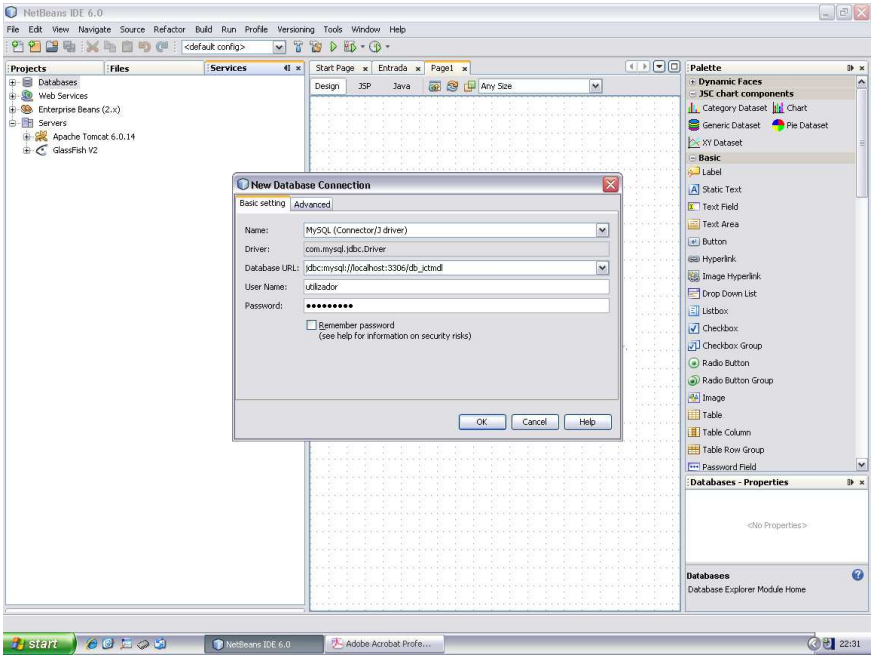

Figura 4.3: Configuração da ligação à base de dados

4.3.2 Instalação de Bibliotecas Adicionais

Para o desenvolvimento deste projecto foram escolhidas funcionalidades que não existem no NetBeans. Assim, recorreu-se a pacotes de *software open source* que, como veremos com mais detalhe no capítulo 6, são necessários à apresentação e funcionamento da aplicação *web*. Estes incluem objectos para a criação de gráficos, de mensagens de correio electrónico e de documentos HTML e *Portable Document Format* (PDF). Neste capítulo apenas se fará referência à sua instalação e configuração no IDE.

São quatro as bibliotecas adicionais utilizadas neste projecto:

- *JFreeChart 1.0.9*
- *JCommon 1.0.12*
- *JavaMail API 1.4.1*
- *JasperReports 2.0.5 project*

4.3.2.1 JFreeChart

O JFreeChart é um pacote de ferramentas que permite criar gráficos muito facilmente a partir de tabelas ou vectores de dados. Inclui também métodos para cálculo estatístico que permitem atingir resultados complexos mais rapidamente. Muitos destes métodos estão incluídos na biblioteca *JCommon*[1]. O seu *download*, documentação e alguns exemplos podem ser encontrados no sítio da *JFree* no endereço *http://www.jfree.org/jfreechart/*.

Para adicionar a biblioteca ao IDE, tornando-a disponível para todos os projectos, deve-se utilizar o *Library Manager* (`Menu > Tools > Libraries`). De seguida, clicar em `New Library` e atribuir um nome. Com o separador `Classpath` seleccionado, adicionar o ficheiro *jfreechart-1.0.9.jar* através de `Add JAR/Folder`. Premir OK para terminar.(Figura 4.4). O mesmo procedimento é utilizado para a biblioteca *jcommon-1.0.12.jar* (Figura 4.5).

4.3.2.2 API JavaMail

A API *JavaMail* permite criar aplicações que necessitem de enviar e receber mensagens de correio electrónico. São suportados os protocolos IMAP, POP3 e SMTP. A utilização desta biblioteca tem como finalidade enviar notificações por correio electrónico que incluem os relatórios de teste gerados pela aplicação.

[1] A biblioteca *JCommon* é utilizada em conjunto com outros projectos da JFree, tal como o JFreeReport

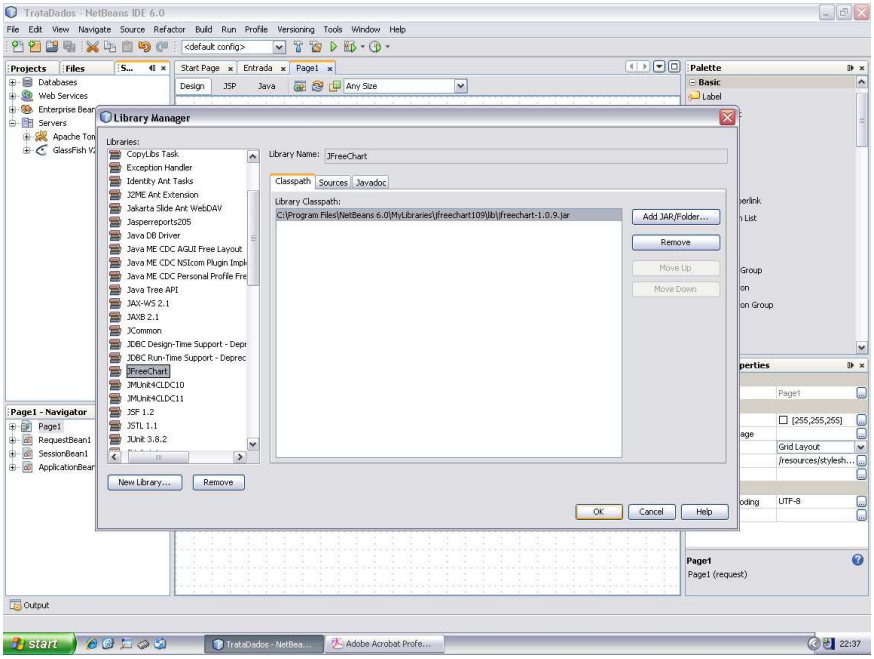

Figura 4.4: Instalação da biblioteca JFreeChart

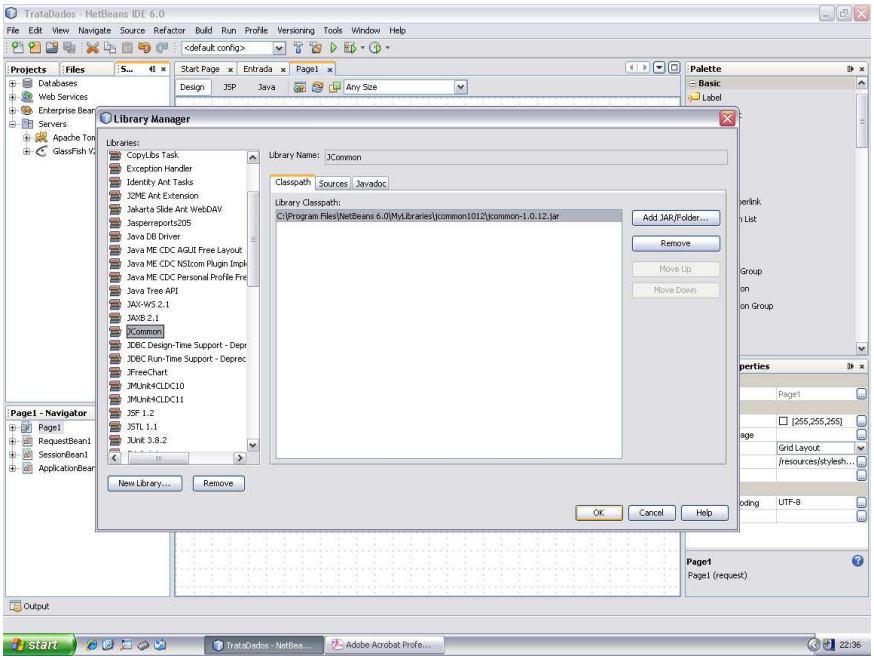

Figura 4.5: Instalação da biblioteca JCommon

4.3.2.3 JasperReports

O JasperReports é uma ferramenta que permite a criação de relatórios em formatos HTML e PDF com dados e imagens. A instalação da biblioteca JasperReports apresenta um pouco mais de complexidade. Em vez de apenas um ficheiro, é necessário incluir pelo menos seis.

Deve-se começar por extrair o pacote *JasperReports-2.0.5-project.zip*. Seguindo um procedimento semelhante aos das bibliotecas anteriores, devemos seleccionar os ficheiros seguintes e premir OK.

- *jasperreports-2.0.5.jar*
- *commons-beanutils-1.7.jar*
- *commons-collections-2.1.jar*
- *commons-digester-1.7.jar*
- *commons-logging-1.0.2.jar*
- *Jitext-1.3.1.jar*

A publicação do relatório em PDF passa pela construção de um modelo ou *template* que servirá de suporte para os dados e imagens a publicar. Este modelo tem um formato XML, embora apresente uma extensão diferente: *jrxml*.

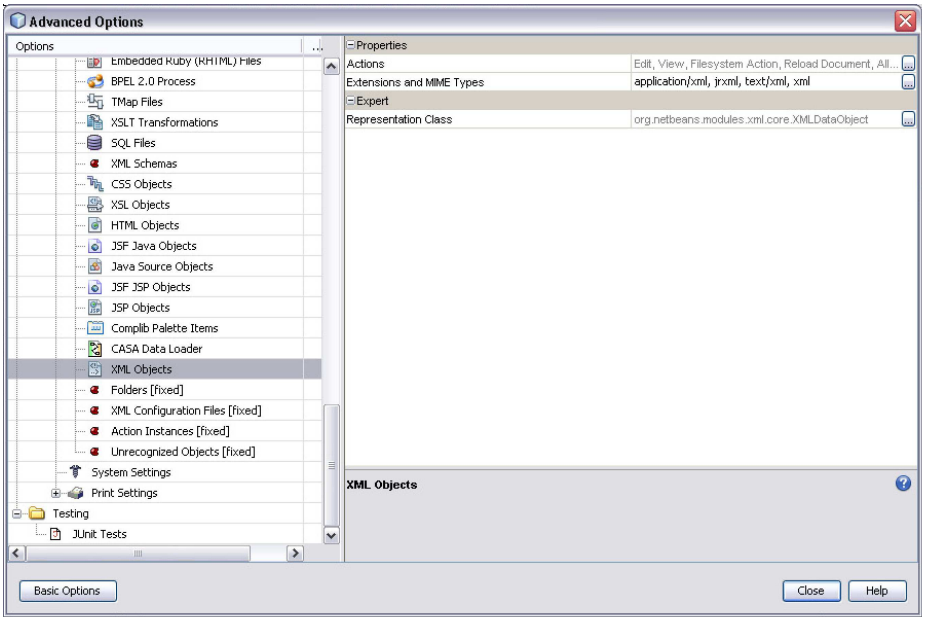

Figura 4.6: A extensão *jrxml* passa a ser reconhecida como um ficheiro XML

O IDE não reconhece esta extensão como sendo um ficheiro XML. No entanto, temos de o configurar para tal, seleccionando `Options/Advanced Options` (Figura 4.6), seguido de `Object Types > XML Objects`. No campo `Extension and MIME Types` adicionar `jrxml`. Premir `Close` para terminar.

Para a construção do suporte do relatório existe uma aplicação bastante útil que permite a criação de relatórios com uma apresentação bastante sofisticada: o iReport (Figura 4.7). É também um *software open source* que pode ser obtido no sítio da Jasperforge: *http://www.jasperforge.org*. Com este programa é fácil fazer um modelo para os relatórios, configurar as entradas de dados com ou sem ligação a um servidor SQL e pré-visualizar a saída. Este programa também está disponível como *plug-in* do NetBeans.

A seguir, explica-se o processo de criação deste modelo e a sua inclusão no projecto de modo a ser compilado e executado pelo IDE.

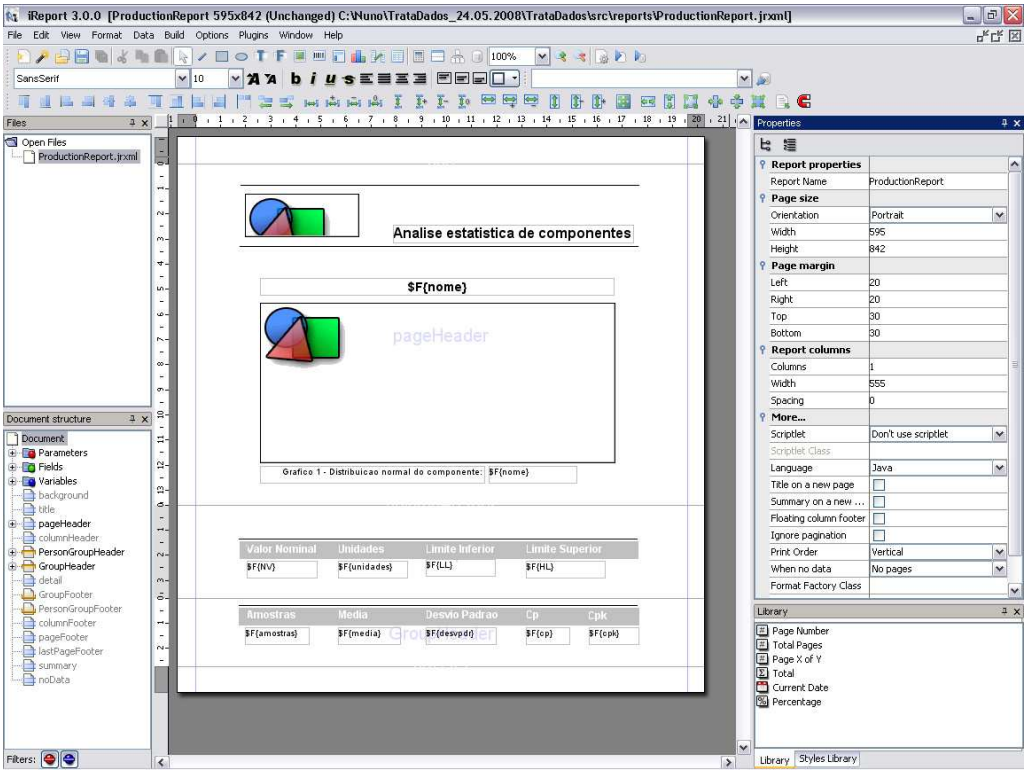

Figura 4.7: Ambiente do iReport

O ficheiro criado neste programa tem de ser incluído numa pasta de recursos do projecto. Para isso, é necessário:

1. No separador `Files`, clicar com o botão direito do rato na pasta `src` seguida de `New > Folder` para criar uma nova pasta. Dar um nome à pasta (*e.g.*, *reports*).

2. Na pasta do projecto, seleccionar `Properties` com o botão direito do rato. No separador `Categories`, seleccionar `Sources`, fazer `Add Folder` e escolher a pasta que foi criada no passo anterior. Na *label*, colocar *JasperReports Definitions*. Premir *Enter* e fazer OK.

3. O ficheiro criado no iReport pode ser copiado para a pasta: `C:\<DirectórioProjecto>\src\reports`.

4. No separador `Files`, dentro da pasta geral do projecto, abrir o ficheiro `build.xml` e adicionar o código do excerto 4.1 no final deste ficheiro.

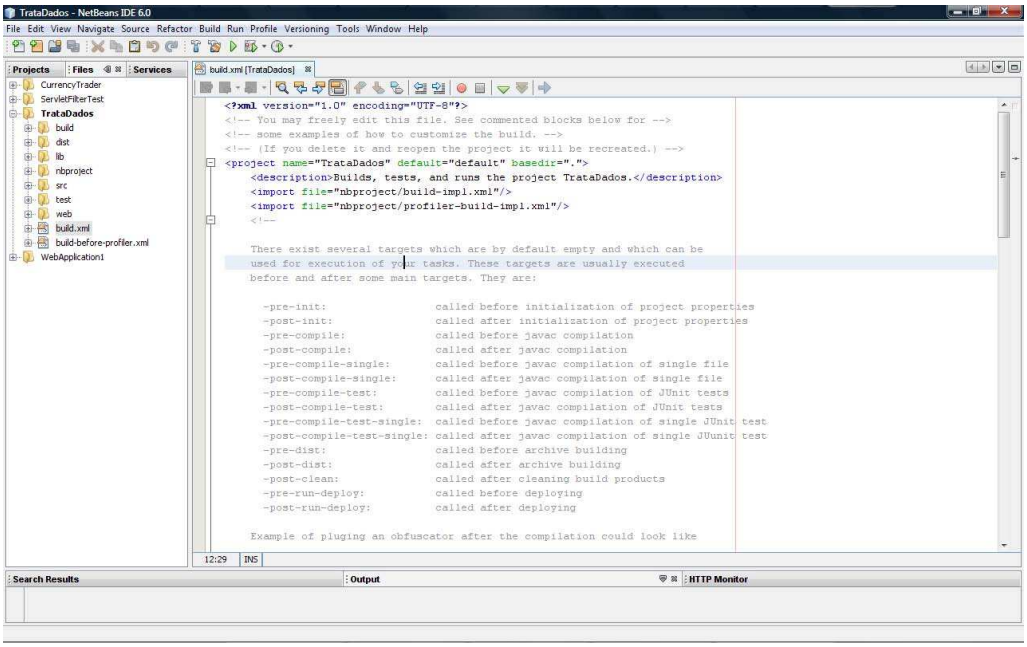

Figura 4.8: Edição do ficheiro *buid.xml*

Excerto de Código 4.1 *Script* para compilar os ficheiros *jrxml*

```
<!-- Propriedade que indica a localização dos relatórios -->
  <property name="jrc.home"
   value="C:\Programas\MyLibraries\jasperreports-2.0.5"/>
<!-- Localização das classes usadas para a compilação -->
  <path id="jrc.classpath">
    <fileset dir="${jrc.home}/dist" includes="*.jar"/>
    <fileset dir="${jrc.home}/lib" includes="*.jar"/>
  </path>
<!-- Tarefa Ant que compila as definições dos relatórios -->
  <taskdef name="jrc"
    classname="net.sf.jasperreports.ant.JRAntCompileTask">
    <classpath refid="jrc.classpath"/>
  </taskdef>
  <target name="-post-compile"
        description="Compile all Jasper Reports Definitions">
    <!-- Criação do directório de saída -->
    <mkdir dir="${build.web.dir}/WEB-INF/reports"/>
    <!-- Compile report definition -->
    <jrc srcdir="src/reports"
              destdir="${build.web.dir}/WEB-INF/reports">
        <classpath refid="jrc.classpath"/>
        <include name="*.jrxml"/>
    </jrc>
    <delete>
        <fileset dir="${build.web.dir}/WEB-INF/classes"
                 includes="*.jrxml"/>
    </delete>
  </target>
```

Para incluir imagens no relatório é necessário configurar um *servlet*. Quando é pedida uma imagem, a aplicação servidora invoca o *servlet* de criação de imagens. Para configurar este *servlet* há que, no ficheiro `WEB-INF > web.xml`, clicar em `Servlets` seguido de `Add Servlet Element`. Adicionar os parâmetros da tabela e premir OK.

`Servlet Name: ImageServlet`
`Servlet Class: net.sf.jasperreports.j2ee.servlets.ImageServlet`
`URL Patterns: /image`

Finalmente, adiciona-se o código responsável pela lógica de criação dos relatórios. Este tem de ser inserido no *Application Bean* (programas 4.2 e 4.3).

Excerto de Código 4.2 Lógica funcional da criação dos relatórios - parte 1

```
1 private static final String PREFIX = "/WEB-INF/reports/";
2 private static final String SUFFIX = ".jasper";
3 private static final String[] VALID_TYPES =
4 { "text/html",            // Representação em HTML
5   "application/pdf",      // Representação em PDF
6 };
7
8 public void jasperReport(String name, String type, ResultSet data){
9     jasperReport(name, type, data, new HashMap());}
10
11 public void jasperReport(String name, String type, ResultSet data, Map params){
12     boolean found = false;
13     for (int i = 0; i < VALID_TYPES.length; i++){
14       if (VALID_TYPES[i].equals(type)){
15         found = true;
16         break;
17       }
18     }
19     if (!found) {
20       throw new IllegalArgumentException("Invalid report type '"
21                 + type
22                 + "' requested");}
23
24     ExternalContext econtext = getExternalContext();
25     InputStream stream = econtext.getResourceAsStream(PREFIX + name
26                 + SUFFIX);
27     if (stream == null){
28       throw new IllegalArgumentException("Unknown report name '"
29                 + name
30                 + "' requested");}
31     try{
32        data.beforeFirst();}
33     catch (Exception e){
34       throw new FacesException(e);}
35
36     JRResultSetDataSource ds = new JRResultSetDataSource(data);
37     JasperPrint jasperPrint = null;
38     try{
39       jasperPrint = JasperFillManager.fillReport(stream, params, ds);}
40     catch (RuntimeException e){
41       throw e;}
42     catch (Exception e){
43       throw new FacesException(e);}
44     finally{
45       try{
46          stream.close();}
47       catch (IOException e){;}
48      }
49 }
```

Excerto de Código 4.3 Lógica funcional da criação dos relatórios - parte 2

```
1     JRExporter exporter = null;
2     HttpServletResponse response = (HttpServletResponse)
3     econtext.getResponse();
4     FacesContext fcontext = FacesContext.getCurrentInstance();
5     try {
6         response.setContentType(type);
7         if ("application/pdf".equals(type)) {
8           exporter = new JRPdfExporter();
9           exporter.setParameter(JRExporterParameter.JASPER_PRINT,
10            jasperPrint);
11          exporter.setParameter(JRExporterParameter.OUTPUT_STREAM,
12            response.getOutputStream());
13        } else if ("text/html".equals(type)) {
14          exporter = new JRHtmlExporter();
15          exporter.setParameter(JRExporterParameter.JASPER_PRINT,
16            jasperPrint);
17          exporter.setParameter(JRExporterParameter.OUTPUT_WRITER,
18            response.getWriter());
19          HttpServletRequest request =
20            (HttpServletRequest)
21            fcontext.getExternalContext().getRequest();
22          request.getSession().setAttribute(
23            ImageServlet.DEFAULT_JASPER_PRINT_SESSION_ATTRIBUTE,
24            jasperPrint);
25          exporter.setParameter(
26            JRHtmlExporterParameter.IMAGES_MAP, new HashMap());
27          exporter.setParameter(JRHtmlExporterParameter.IMAGES_URI,
28            request.getContextPath() + "/image?image=");
29        }
30    } catch (RuntimeException e) {
31        throw e;
32    } catch (Exception e) {
33        throw new FacesException(e);
34    }
35
36    try {
37        exporter.exportReport();
38    } catch (RuntimeException e) {
39        throw e;
40    } catch (Exception e) {
41        throw new FacesException(e);
42    }
43
44    fcontext.responseComplete();
45 }
```

4.4 Configuração do Servidor Apache Tomcat

Apesar do NetBeans incluir o servidor *Apache Tomcat*, é necessário instalar este servidor de aplicações na máquina que disponibilizará a aplicação aos utilizadores, pois estaremos fora do contexto do IDE. A versão utilizada foi a 6.0.16 e a sua instalação é simples. Durante a instalação é configurado o porto HTTP que, por omissão, é o 8080. Existe um monitor do Apache que arranca automaticamente, permitindo controlar o estado do servidor de aplicações. A gestão das aplicações faz-se através do *Tomcat Manager* (na sua página local, disponibilizada no porto 8080), depois de uma autenticação. A configuração presente no servidor Apache é a seguinte:

Tabela 4.2: Configurações de acesso ao *Apache Tomcat*:

`ServerHost: localhost`
`HTTP Port: 8080`
`Username: admin`
`Password: adminadmin`

Uma vez configurado o servidor de aplicações, é necessário adicionar a localização das API de Java disponibilizadas pelo Tomcat às variáveis de ambiente. Caso não esteja definida a variável `"CLASSPATH"`deverá ser criada no campo de variáveis do sistema. O seu conteúdo deverá incluir:

```
C:\(...)\ApacheTomcat6.0.14\lib\servlet-api.jar;
C:\(...)\ApacheTomcat6.0.14\lib\jsp-api.jar;
C:\(...)\ApacheTomcat6.0.14\lib\mysql-connector-java-5.1.5-bin.jar;
```

A última linha refere-se ao controlador JDBC para ligação ao servidor MySQL que, neste caso, foi colocado junto com as API do Tomcat. Estas API incluem:

- *servlet-api.jar - Biblioteca para desenvolvimento de servlets*
- *jsp-api.jar - Biblioteca para desenvolvimento de páginas JSP*

O servidor é iniciado através do menu de configuração (Figura 4.9). Após o seu arranque, é possível invocar-se a página *web* de entrada que dá acesso às aplicações e às suas definições. O endereço da página inicial é *http://localhost:8080/* (Figura 4.10).

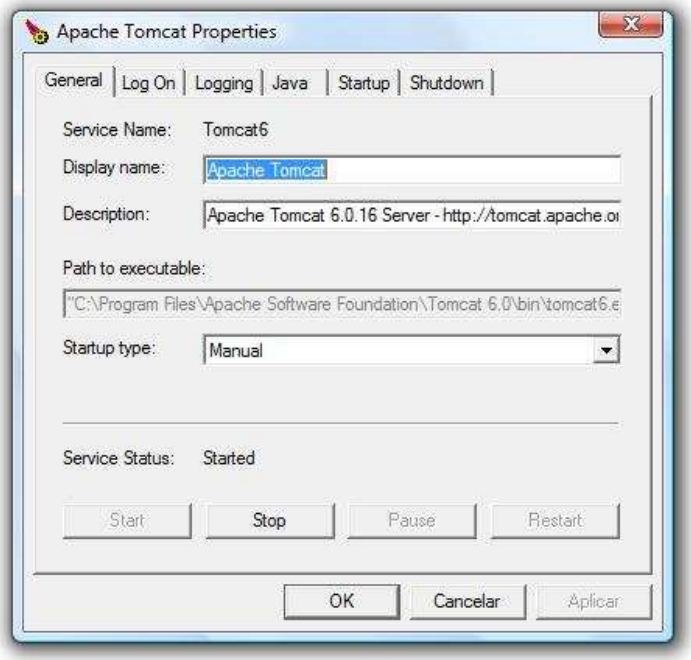

Figura 4.9: Menu de configuração do Apache Tomcat

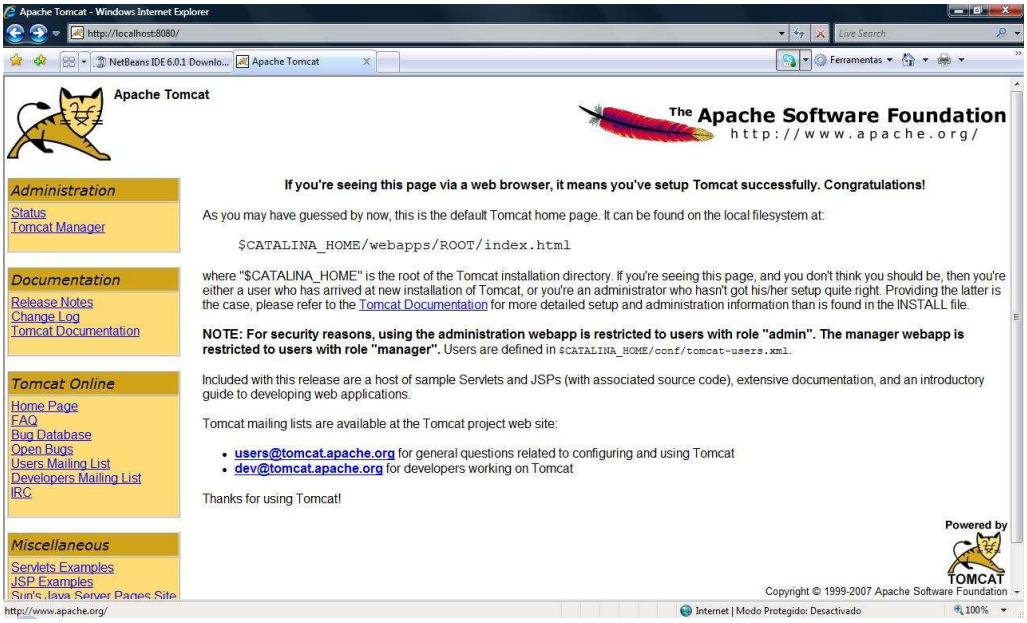

Figura 4.10: Página de entrada do Apache Tomcat

4.5 Conclusão

Após a realização dos procedimentos de configuração do ambiente de desenvolvimento descritos, é possível iniciar o desenvolvimento da aplicação que se detalha nos dois capítulos seguintes.

Capítulo 5

Conversão de Ficheiros MDL

Esta aplicação tem como objectivo recolher a informação guardada em ficheiros de texto, processá-la e, por último, armazená-la em bases de dados. Estes ficheiros, que têm a extensão MDL (Measurement Data Log), contêm os dados gerados durante o processo de teste dos produtos e componentes.

5.1 Estrutura dos Ficheiros

Foi necessário desenvolver uma aplicação que converta os ficheiros de texto gerados pelos dispositivos de teste ICT e guarde a informação resultante numa base de dados MySQL. É uma aplicação *stand-alone* para ser executada em cada máquina de teste, que lê os ficheiros, extrai e processa a informação desejada, guarda-a no servidor MySQL e apaga os ficheiros originais. Estes ficheiros de texto apresentam uma formatação delimitada por etiquetas, o que simplifica o seu tratamento. Existem dois campos distintos: um cabeçalho, com informações relativas ao nome do produto e resultado final do teste, e um campo de conteúdo ou unidade testada, com informações relativas a todos os componentes testados (Figura 5.1).

5.1.1 Cabeçalho - Header

O cabeçalho é delimitado por duas etiquetas: *SHeader* e *EHeader*. A informação contida entre estas duas etiquetas diz respeito ao produto e ao seu programa de teste. O cabeçalho faz referência aos seguintes parâmetros:

- **TS** - Nome do Operador (utilizador)
- **BTyp** - Identificação do produto
- **TPrg** - Identificação do programa de teste
- **TPTime** - Data de criação do programa de teste
- **STime** - Início do processo de teste
- **ETime** - Fim do processo de teste
- **OSts** - Resultado do processo de teste

Apesar de existirem outros elementos no cabeçalho, estes não são utilizados pelo sistema de teste.

5.1.2 Unidade Testada - Unit

Este campo é delimitado por duas etiquetas: *SUnit* e *EUnit*. Contém todos os dados relativos a um dos componentes testados. Por cada componente testado existe um linha do tipo que se apresenta abaixo:

`{D014, F, 320 mV, 305.45 mV, 275.2 mV, 352 mV, PASS}`[1]

- **D014** - Nome do componente (Díodo 14)
- **F** - Medida da tensão directa do díodo (*Forward Voltage Measurement*)
- **320 mV** - Tensão nominal
- **305.45 mV** - Tensão medida
- **275.2 mV** - Limite inferior de tensão admissível
- **352 mV** - Limite superior de tensão admissível
- **PASS** - Resultado do teste do componente

[1] A notação utilizada nestes ficheiros é a de língua Inglesa.

```
STest
  SHeader
    BID NONE
    TS 6018
    FIX NONE
    BTyp 048-0001
    Ver  0
    ECO 0
    TPrg REGELBG
    TPTime 08/12/2006 09:38:54
    Lot NONE
    Stime 09/01/2007 15:48:20
    Etime 09/01/2007 15:48:29
    OSts FAIL
  EHeader
  SUnit
    UName REGELBG
    {P002, 1, 2 KOhm, 2.36063 KOhm, 1.4 KOhm, 2.6 KOhm, PASS}
    {MB2S D1, 1 F, 660 mV, 684.875 mV, 594 mV, 726 mV, PASS}
    {MB2S D1, 1 R, 1.5 V, 1.4248 V, 1.35 V, 1.65 V, PASS}
    {MB2S D1, 2 F, 660 mV, 682.373 mV, 594 mV, 726 mV, PASS}
    {MB2S D1, 2 R, 1.5 V, 1.50488 V, 1.35 V, 1.65 V, PASS}
    {MB2S D1, 3 F, 660 mV, 907.288 mV, 594 mV, 726 mV, FAIL}
    {MB2S D1, 3 R, 1.5 V, 1.51233 V, 1.35 V, 1.65 V, PASS}
    {MB2S D1, 4 F, 660 mV, 909.79 mV, 594 mV, 726 mV, FAIL}
    {MB2S D1, 4 R, 1.5 V, 1.51978 V, 1.35 V, 1.65 V, PASS}
    {D005, 1/F, 700 mV, 708.618 mV, 630 mV, 770 mV, PASS}
    {D005, 1/R, 1.5 V, 1.51733 V, 1.35 V, 1.65 V, PASS}
    {D005, 2/F, 700 mV, 702.332 mV, 630 mV, 770 mV, PASS}
    {D005, 2/R, 1.5 V, 1.50989 V, 1.35 V, 1.65 V, PASS}
    {D002, F, 2 V, 1.82227 V, 1.8 V, 2.2 V, PASS}
    {D002, R, 2.5 V, 2.51465 V, 2.25 V, 2.75 V, PASS}
    {T002, BE, 42 KOhm, 41.271 KOhm, 33.6 KOhm, 50.4 KOhm, PASS}
    {R003, 1, 1.37 KOhm, 1.3495 KOhm, 1.233 KOhm, 1.507 KOhm, PASS}
    {R017, 1, 47 KOhm, 45.659 KOhm, 39.95 KOhm, 54.05 KOhm, PASS}
    {R018, 1, 47 KOhm, 46.147 KOhm, 39.95 KOhm, 54.05 KOhm, PASS}
    {R019, 1, 10 KOhm, 9.891 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R020, 1, 430 Ohm, MAX_OVL, 365.5 Ohm, 494.5 Ohm, FAIL}
    {R021, 1, 430 Ohm, 404.375 Ohm, 365.5 Ohm, 494.5 Ohm, PASS}
    {R022, 1, 430 Ohm, 442 Ohm, 365.5 Ohm, 494.5 Ohm, PASS}
    {R023, 1, 430 Ohm, 403.516 Ohm, 365.5 Ohm, 494.5 Ohm, PASS}
    {R027, 1, 10 KOhm, 9.72 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R028, 1, 10 KOhm, 9.663 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R030, 1, 2.2 KOhm, 2.21038 KOhm, 1.87 KOhm, 2.53 KOhm, PASS}
    {R033, 1, 10 KOhm, 9.7105 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R034, 1, 10 KOhm, 9.695 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R036, 1, 10 KOhm, 9.772 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R041, 1, 1 KOhm, 0.966156 KOhm, 0.85 KOhm, 1.15 KOhm, PASS}
    {R042, 1, 2.7 KOhm, 2.68575 KOhm, 2.295 KOhm, 3.105 KOhm, PASS}
    {R043, 1, 10 KOhm, 9.845 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R050, 1, 5.1 KOhm, 4.9595 KOhm, 4.335 KOhm, 5.865 KOhm, PASS}
    {R054, 1, 10 KOhm, 9.772 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R055, 1, 1 KOhm, 0.965156 KOhm, 0.85 KOhm, 1.15 KOhm, PASS}
    {R056, 1, 2.7 KOhm, 2.71575 KOhm, 2.295 KOhm, 3.105 KOhm, PASS}
    {R067, 1, 10 KOhm, 9.845 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R058, 1, 5.1 KOhm, 4.8495 KOhm, 4.335 KOhm, 5.865 KOhm, PASS}
    {R060, 1, 680 Ohm, 663.469 Ohm, 578 Ohm, 782 Ohm, PASS}
    {C013, 1, 10 nF, 10.8207 nF, 7.5 nF, 12.5 nF, PASS}
    {C016, 1, 10 nF, 10.5356 nF, 7.9 nF, 12.1 nF, PASS}
    {C017, 1, 1 nF, 1.08126 nF, 0.75 nF, 1.25 nF, PASS}
    {C018, 1, 1 nF, 1.09491 nF, 0.75 nF, 1.25 nF, PASS}
    {C019, 1, 1 nF, 1.04743 nF, 0.65 nF, 1.35 nF, PASS}
    {C020, 1, 1 nF, 1.06023 nF, 0.8 nF, 1.2 nF, PASS}
    {C021, 1, 1 nF, 0.994792 nF, 0.75 nF, 1.25 nF, PASS}
    {C024, 1, 10 nF, 10.7461 nF, 7.5 nF, 12.5 nF, PASS}
    {Q2_C1, 1, 200 pF, 184.4 pF, 140 pF, 260 pF, PASS}
    {Q2_C2, 1, 200 pF, 173.458 pF, 140 pF, 260 pF, PASS}
    {C011//C010, 1, 220 uF, 220.82 uF, 176 uF, 264 uF, PASS}
    {R028//C015, RES, 10 KOhm, 9.4015 KOhm, 8.5 KOhm, 11.5 KOhm, PASS}
    {R028//C015, CAP, 10 nF, 11.0736 nF, 7.9 nF, 12.1 nF, PASS}
  EUnit
ETest
```

Figura 5.1: Exemplo de relatório de Teste *(Measurement Data Log)*

5.2 Arquitectura e Funcionamento da Aplicação

A aplicação de conversão de ficheiros de teste irá residir em cada uma das máquinas de teste, resultando numa arquitectura do tipo apresentado na Figura 5.2. Esta aplicação verifica o estado do ficheiro e extrai os dados importantes para o servidor MySQL.

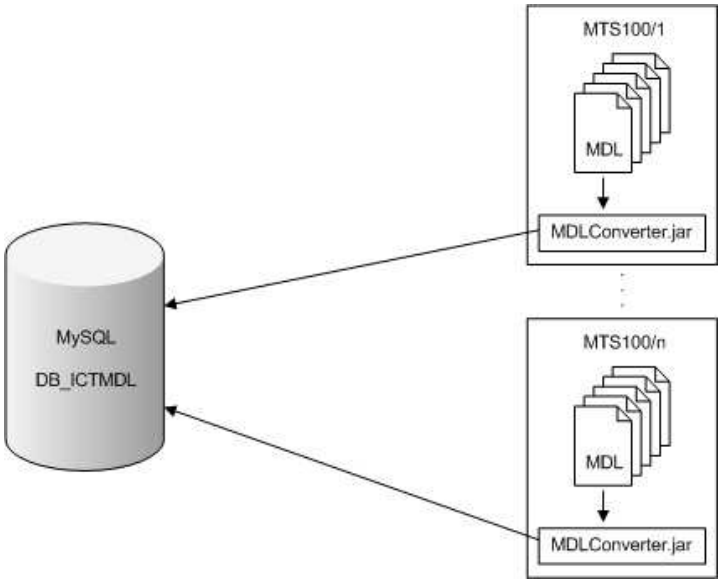

Figura 5.2: Arquitectura utilizada para a conversão de ficheiros de teste

A conversão é efectuada em cinco passos principais:

1. Verificação da integridade e forma do ficheiro de texto;
2. Identificação do sistema de teste;
3. Extracção dos dados do cabeçalho e inserção na base de dados;
4. Extracção dos dados dos componentes e inserção na base de dados;
5. Caso a operação seja bem sucedida, procede à eliminação do ficheiro original.

O ficheiro MDL deverá estar bem formado e ser íntegro. No caso de existirem erros de forma, o ficheiro MDL é considerado um ficheiro inválido, não podendo ser convertido.

Depois desta confirmação, é retirada do cabeçalho a seguinte informação: nome do produto, tempo de início e de finalização do teste, identificação do dispositivo e o resultado final. Estes dados são inseridos numa tabela que guarda apenas os cabeçalhos (tabela de ficheiros `ICT_MDL`), ou seja, cada entrada nesta tabela representa um ficheiro convertido. O índice de entrada é devolvido ao programa, pois será necessário para relacionar os valores dos componentes testados.

Do corpo do ficheiro são retirados os parâmetros relativos a cada um dos componentes. Estes, juntamente com o índice que os relaciona com o cabeçalho, são inseridos numa outra tabela (tabela de componentes `ICT_MDL_VALUES`) representando uma entrada. Depois de todos os componentes serem inseridos, e de não ter existido nenhum erro, seja este de interpretação do ficheiro, seja na ligação à base de dados, o ficheiro é eliminado.

Para a interpretação das linhas de texto é utilizado um *parser* do Java, o *scanner*. Este pertence à biblioteca `java.utils.scanner`. Este método é utilizado para extrair todos os dados necessários.

A aplicação desenvolvida é formada pelas classes:

- `MDLReaderFrame.class` para a interface com o utilizador;
- `AlertMessage.class` que indica ao utilizador eventuais mensagens de erro;
- `FileBrowser.class` para a escolha da pasta de origem de ficheiros MDL;
- `Processo.class` que é a classe principal da aplicação;
- `IctMdl.class` gerada pelo NetBeans para acesso à tabela `ICT_MDL`;
- `IctMdlUnits.class` gerada pelo NetBeans para acesso à tabela `ICT_MDL_UNITS`;
- `IctMdlValues.class` gerada pelo NetBeans para acesso à tabela `ICT_MDL_VALUES`.

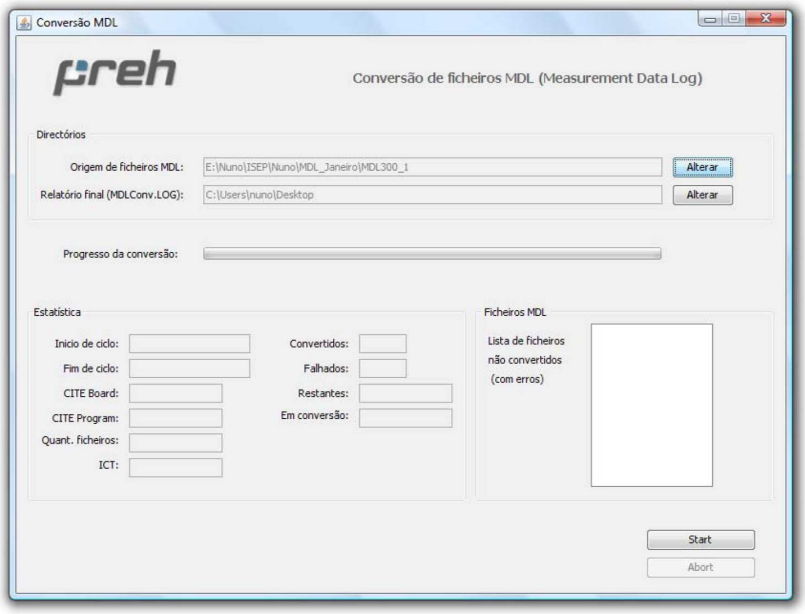

Figura 5.3: Aplicação para a conversão dos ficheiros

A classe `Processo.class` contém toda a lógica para a conversão e armazenamento dos dados no servidor MySQL. Os seus métodos podem ser resumidos de acordo com o excerto de código 5.1.

Esta aplicação será colocada em cada um dos dispositivos de teste. Adicionalmente, no directório onde são guardados os ficheiros MDL resultantes do processo de teste, será colocado um pequeno ficheiro XML que inclui um identificador do dispositivo em questão.

Esta aplicação é desencadeada manualmente ou agendada para execução periódica nas tarefas do *Windows*. Note-se que, em regime normal, estes ficheiros são gerados por amostragem, resultando um por cada dez peças testadas.

O programa disponibiliza uma opção para seleccionar o caminho do directório dos ficheiros. Este caminho é guardado num ficheiro `MDLSettings.xml`. A aplicação gera um pequeno relatório de execução para, no fim do processo de conversão, o utilizador ser notificado de possíveis erros. Durante a execução do programa podemos ver uma barra de progresso representando a evolução do processamento, o nome do produto, o nome do ficheiro em conversão e uma lista de ficheiros não convertidos, entre outros dados.

Nos testes efectuados com mil ficheiros, o tempo de conversão foi pouco mais de 5 min. Num ambiente normal de produção poderão existir em média cerca de 100 a 200 ficheiros em cada máquina por dia e num total de onze máquinas. Esta colecta de dados será feita apenas de acordo com as necessidades de análise de

Excerto de Código 5.1 Classe principal da aplicação de conversão de ficheiros MDL

```
1  public class Processo extends Observable implements Runnable
2  {
3    List lstFileList;
4    short srtICTid;
5    String TestProgram = "";
6    String BoardType = "";
7    String StartTime = "";
8    String EndTime = "";
9    String TestResult = "";
10   String MDLName = "";
11   String CompMDLName = "";
12   private final String SQLClass = "com.mysql.jdbc.Driver";
13   private final String SQLDriver = "jdbc:mysql://localhost:3306/db_ictmdl";
14   private final String SQLUsername = "root";
15   private final String SQLPass = "tevd231";
16
17   public Processo(Observer observador){
18     addObserver(observador);
19   }
20
21   public void run(){
22   // Sequência do programa (...)
23   }
24
25   // Verificação da formatação do ficheiro:
26   private boolean processFileTags() throws FileNotFoundException {(...)}
27   // Processamento de cabeçalho:
28   private boolean processHeader() throws FileNotFoundException {(...)}
29   // Processamento de uma linha do cabeçalho:
30   private String processHeaderLine(String aLine, String Identifier) {(...)}
31   // Processamento dos componentes:
32   private boolean processBody() throws FileNotFoundException {(...)}
33   // Processamento de uma linha da secção de componentes:
34   private boolean processBodyLine(String aLine) {(...)}
35   // Devolve a lista de ficheiros no directório seleccionado:
36   private boolean ListFiles() {(...)}
37   // Devolve a unidade de medida de um componente:
38   private short ListDBUnits(String strUnit) {(...)}
39   // Devolve os tipos de medida de um componente:
40   private short ListDBTestTypes(String strUnit) {(...)}
41   // Lê os directórios dos ficheiros e Log final:
42   public boolean ReadXML() {(...)}
43   // Devolve a identificação do dispositivo de teste:
44   public boolean ReadICT_ID() {(...)}
45   // Executa uma Stored Proc para guardar o cabeçalho:
46   public boolean ExecStoredProcHeaderData() {(...)}
47   // Guarda os componentes na base de dados:
48   private boolean SaveComponentData() {(...)}
49 }
```

um ou mais produtos e não de uma forma permanente. Embora os dados sejam armazenados num servidor MySQL, rapidamente o espaço necessário pode atingir proporções elevadas. Na simulação efectuada para este trabalho, foram inseridos na base de dados cerca de 23 000 ficheiros contendo cerca de 900 000 componentes testados no mês de Janeiro. Este volume de informação ocupa cerca de 100 MB no servidor MySQL.

5.3 Estrutura da Base de Dados

A base de dados `DB_ICTMDL` que foi criada no MySQL consiste em seis tabelas para o armazenamento de informação e um procedimento (*stored procedure*). A estrutura desta base de dados e o relacionamento entre as tabelas podem ser vistos na figura 5.4, sendo o objectivo de cada elemento descrito a seguir:

- `UTILIZADORES` - Utilizadores autorizados;
- `ICT_MDL` - Entradas de ficheiros MDL (produtos testados);
- `ICT_MDL_VALUES` - Componentes testados;
- `ICT_MDL_TYPES` - Tipos de medição efectuada sobre o componente;
- `ICT_MDL_UNITS` - Unidades de medida dos componentes;
- `ICT_MDL_ICTS` - Dispositivos de teste ICT.

Todas as transacções para a base de dados são efectuadas com pedidos (*queries*) nativos a partir da aplicação *Java*. Nesta altura surgiu uma questão importante: para inserir um produto na tabela e saber qual o seu índice de entrada seria necessário executar dois pedidos. Apesar de, em circunstancias normais, este ser reduzido, se outro dispositivo de teste inserir um produto na tabela antes do primeiro ter requerido o seu índice, então os componentes que este inserir a seguir serão relacionados com um produto diferente. Daqui resultou a necessidade de criar um procedimento residente no servidor que permita receber os dados da aplicação e retornar o índice de entrada. O procedimento que insere os campos relativos aos produtos e retorna o índice de entrada é apresentado na Query 5.1.

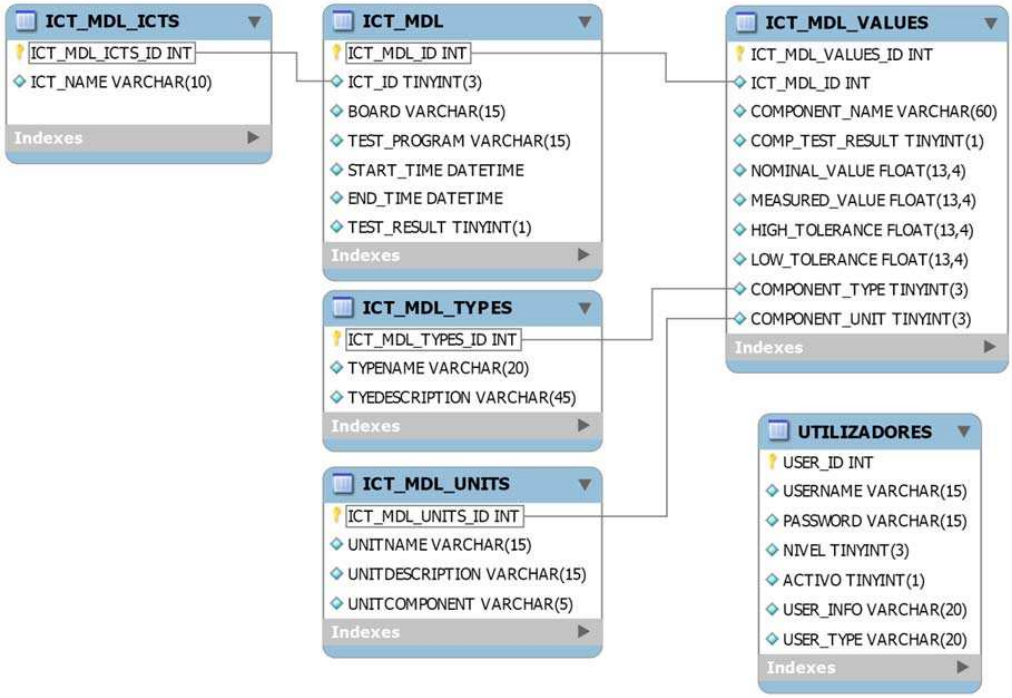

Figura 5.4: Estrutura da base de dados DB_ICTMDL

***Query* 5.1** *Stored procedure* SP_INSERTMDL

```
1  SP_INSERTMDL (OUT _MdlID INT, _IctID TINYINT(3), _Board VARCHAR(15),
2    _TProgram VARCHAR(15), _Inicio DATETIME, _Fim DATETIME, _Resultado BIT)
3  BEGIN
4   INSERT INTO db_ictmdl.ict_mdl
5   VALUES (_IctID, _Board, _TProgram, _Inicio, _Fim, _Resultado);
6   SET _MdlID = LAST_INSERT_ID();
7  END
```

As variáveis de entrada deste procedimento são respectivamente, a identificação do dispositivo (`_IctID`), o nome do produto (`_Board`), o nome do programa de teste (`_TProgram`), o instante de início do teste (`_Inicio`), o instante de finalização do teste (`_Fim`) e o resultado final (`_Resultado`). A variável de saída aparece no início do procedimento: `_MdlID`.

5.4 Conclusão

A conversão e o armazenamento em bases de dados dos ficheiros MDL gerados pelas máquinas de teste é efectuada de forma transparente e automática para os utilizadores do sistema, através de uma aplicação Java dedicada que reside e executa em cada máquina de teste.

Capítulo 6

Aplicação Web

Neste capítulo descreve-se a aplicação web que faculta a consulta dos elementos guardados na base de dados e permite efectuar análises estatísticas, relatórios e desencadear pedidos de intervenção.

6.1 Descrição

A aplicação *web* de processamento dos dados de teste ICT tem por objectivo permitir a consulta e visualização da informação relacionada com os testes ICT efectuados aos produtos, identificar os produtos com falhas, obter valores estatísticos, gráficos, relatórios personalizados e gerar pedidos de intervenção através de notificações por correio electrónico. Toda a informação utilizada pela aplicação Web está armazenada no servidor MySQL, tendo sido inserida automaticamente pela aplicação Java descrita no Capítulo 5.

Para alcançar este objectivo foi desenvolvida uma aplicação *web* designada `TrataDados` com a arquitectura da figura 6.1.

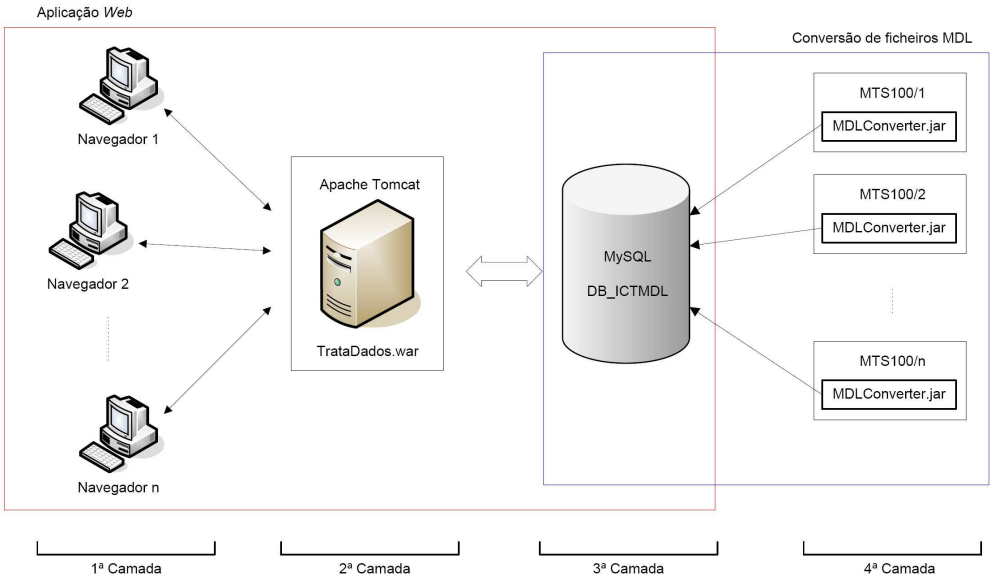

Figura 6.1: Arquitectura global da aplicação desenvolvida

Esta aplicação *web* possui cinco páginas para utilização geral, e uma página reservada ao administrador da aplicação para gestão dos utilizadores. As páginas que formam esta aplicação são:

- Página de acesso - `login.jsp` - permite a autenticação dos utilizadores registados. Existem apenas dois tipos de acesso: o acesso normal e o acesso de administração;

- Página de entrada - `entrada.jsp` - permite ao utilizador optar pelos diferentes tipos de consulta;

- Página de visualização estatística - `distribuicao.jsp` - fornece informação estatística (média, desvio padrão, gráfico com uma distribuição normal, Cp[1], Cpk e quantidades) de um componente específico, mediante uma selecção. É possível gerar relatórios no formato PDF com os resultados obtidos e enviar notificações por correio electrónico;

- Página de testes efectuados - `produzidos.jsp` - permite visualizar todos os testes efectuados aos produtos e os seus resultados através de uma listagem e de um gráfico de barras;

[1]Cp e Cpk são índices de capabilidade do processo de fabrico, permitindo verificar se um determinado processo está ou não dentro do limite especificado.

- Página de produtos com falhas - `analisefalhas.jsp` - permite, através de uma tabela e da selecção de um produto, visualizar, por componente, as falhas identificadas;

- Página de gestão de utilizadores - `utilizadores.jsp` - está reservada apenas a utilizadores com privilégios de administração, permite inserir, remover e alterar utilizadores.

6.2 Criação de um Novo Projecto

Decidiu-se utilizar os componentes gráficos *JavaServer Faces* do pacote *Visual Web* para o desenvolvimento da GUI deste trabalho. As JSF são bastante eficazes para a construção de páginas com uma GUI elaborada assim como permitem associar elementos de bases de dados a objectos, arrastando-os com o rato. Este *framework* foi escolhido logo no início do projecto. Entre outras opções, estão disponíveis para o desenvolvimento da interface gráfica as *Visual Web JavaServer Faces*, *JavaServer Faces* e *Struts* (Figuras 6.2, 6.3 e 6.4).

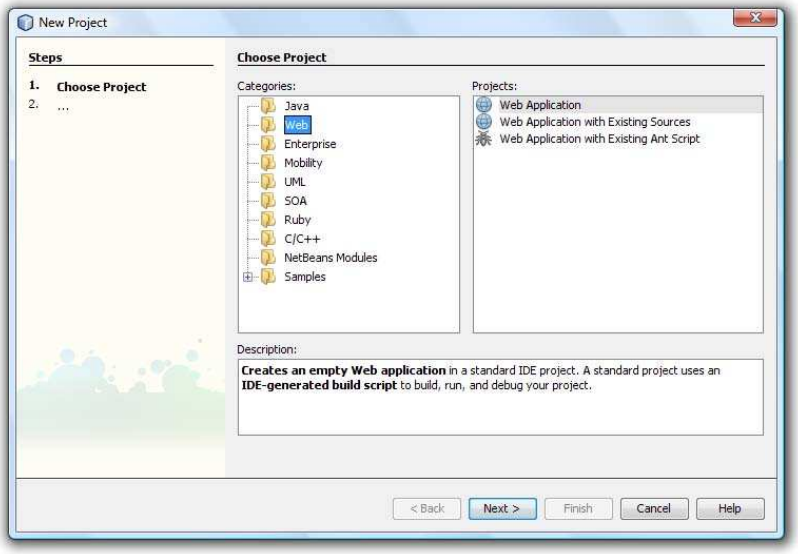

Figura 6.2: Escolha de um novo projecto

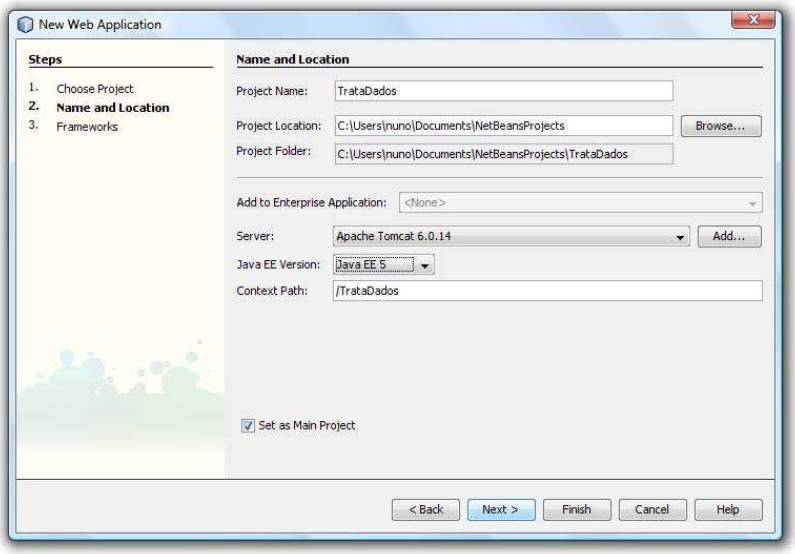

Figura 6.3: Escolha de um nome e do servidor

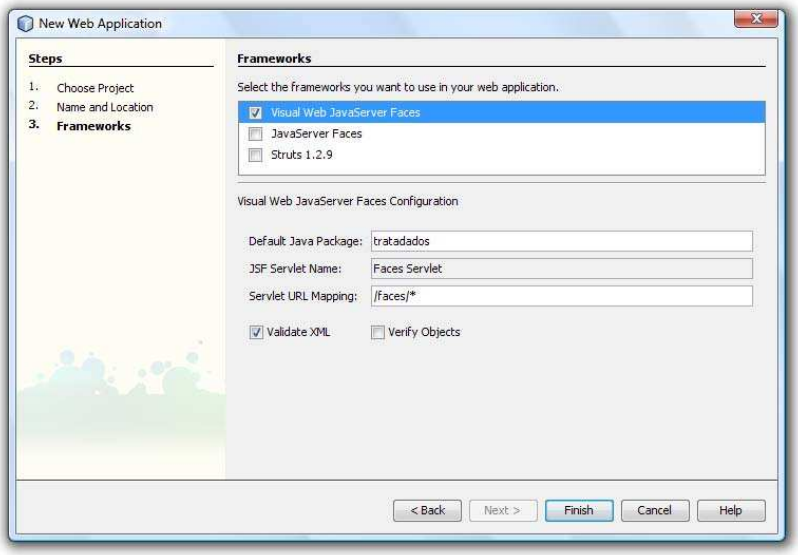

Figura 6.4: Escolha dos *frameworks*

Após a criação do projecto, o NetBeans apresenta três separadores distintos: *(i)* `Design` dedicado à construção da interface (colocação dos objectos e criação do *layout* das páginas); *(ii)* `JSP` destinado ao desenvolvimento do código JSP com as etiquetas referentes aos objectos UI utilizados e *(iii)* `Java` reservado à definição do

código fonte de desenvolvimento da lógica da aplicação. A construção das páginas é desencadeada pela criação da interface gráfica. Todos os objectos necessários são colocados e dispostos de modo a criar o aspecto final desejado.

Ao adicionar uma nova página, é criado automaticamente um conjunto de elementos. No separador `Navigator` pode-se visualizar a estrutura HTML da página contendo o cabeçalho (*header*) e o corpo (*body*). No corpo aparece, por omissão, um formulário (*form*) que é possível alterar. Podem-se ainda adicionar outros formulários, se for necessário.

Também são criados três dos elementos mais importantes da aplicação *web*: *(i)* o Javabean da aplicação - `ApplicationBean` - cujo contexto abrange todo o funcionamento da aplicação, incluindo as sessões de todos os utilizadores (a sua utilização deve ser relacionada com aspectos de gestão e supervisão da aplicação); *(ii)* o Javabean de sessão - `SessionBean` - que diz respeito a uma sessão específica, *i.e.*, a informação contida neste é específica de um utilizador, e *(iii)* o Javabean para as transacções entre páginas - `RequestBean` - cujo propósito é permitir a troca de informação entre elementos de páginas consecutivas.

Os JavaBeans tornam-se essenciais para o armazenamento temporário de informação durante a navegação na aplicação. Podemos assegurar assim que ao movermos-nos de página em página dentro da aplicação, uma quantidade relevante de informação permanece disponível.

Para a implementação de métodos e propriedades nos *JavaBeans*, basta clicar duas vezes sobre eles e editar o código fonte.

6.3 Página de Autenticação

Esta página permite controlar a acesso dos utilizadores à aplicação. Existem três campos de entrada: o campo de texto `USERNAME`, o campo protegido de texto `PASSWORD` e um botão para submeter os dados inseridos.

Foram também definidos dois elementos comuns a todas as páginas: o cabeçalho e rodapé das páginas. Estes elementos são conhecidos por *page fragments* e podem ser incluídos em qualquer local de qualquer página.

Os dois campos de texto são obrigatórios, *i.e.*, estes objectos são marcados como de preenchimento obrigatório.

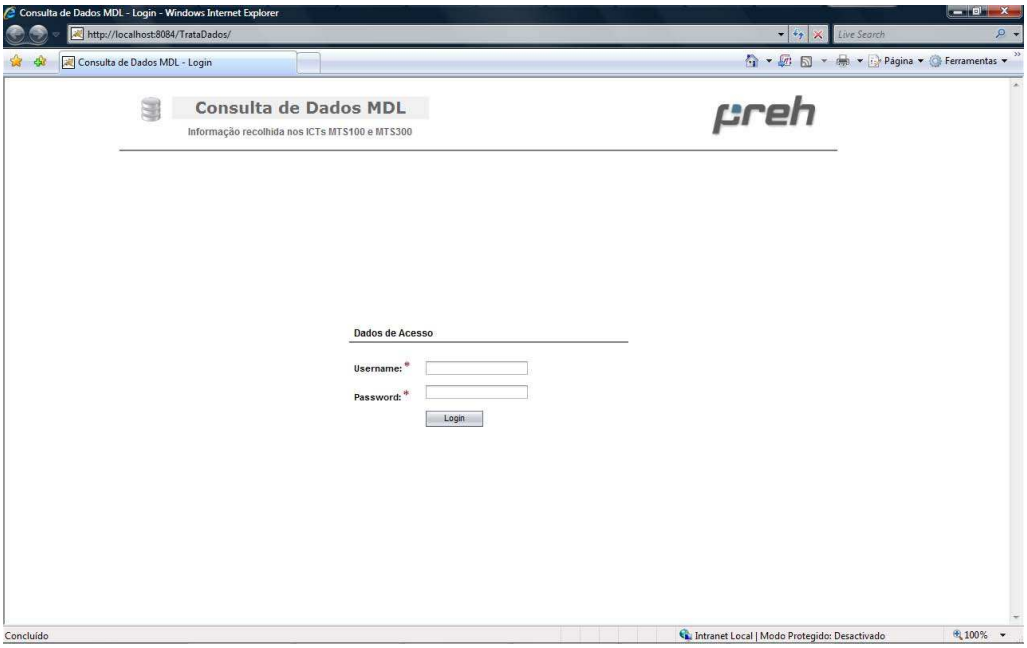

Figura 6.5: Página de Autenticação

6.3.1 Ligação à Tabela de Utilizadores

Ao inserir os dados de acesso, será necessário compará-los com os da tabela `UTILIZADORES` da base de dados. O NetBeans utiliza dois objectos para uma utilização simplificada de bases de dados: o *DataProvider* e o *RowSet*. Para a sua configuração procede-se do seguinte modo:

- No separador `Services` expande-se `Databases` e, de seguida, expande-se a ligação à base de dados `DB_ICTMDL` (ver Capítulo 4, Figura 4.3). Arrasta-se a tabela `UTILIZADORES` para uma zona qualquer da página no separador *Design*.
- Na janela `Navigator`, no canto inferior esquerdo (se não estiver visível, seleccionar `Window>Navigating>Navigator` ou `Ctrl + 7`), aparece um *DataProvider*[2] chamado `utilizadoresDataProvider` inserido no âmbito da página e, no *SessionBean*, um *RowSet*[3] chamado de `utilizadoresRowSet`.

O `utilizadoresRowSet` é o resultado da consulta à tabela `UTILIZADORES` que, por omissão, retorna todos os campos da tabela. É importante notar que este

[2]O *DataProvider* fornece uma interface para aceder a diferentes tipos de dados.
[3]O *RowSet* é responsável pela ligação à base de dados, da execução dos pedidos e gestão dos resultados.

RowSet fica carregado com todos os dados da tabela. Se se utilizar o processo descrito atrás sobre tabelas que contenham muita informação, o IDE pode ficar muito lento ou até mesmo parar o seu funcionamento. Para esses casos deve-se inserir um *DataProvider* e um *Rowset* arrastando-os do menu de objectos JSF e configurá-los manualmente.

6.3.2 Autenticação do utilizador

O método utilizado para efectuar a pesquisa do utilizador na página de autenticação envolve uma pequena alteração ao *RowSet*. Adicionou-se à sua *query* um critério de consulta que depende do nome de utilizador inserido. Este apenas será fornecido quando o utilizador premir o botão de *Login*, submetendo os seus dados (Figura 6.6).

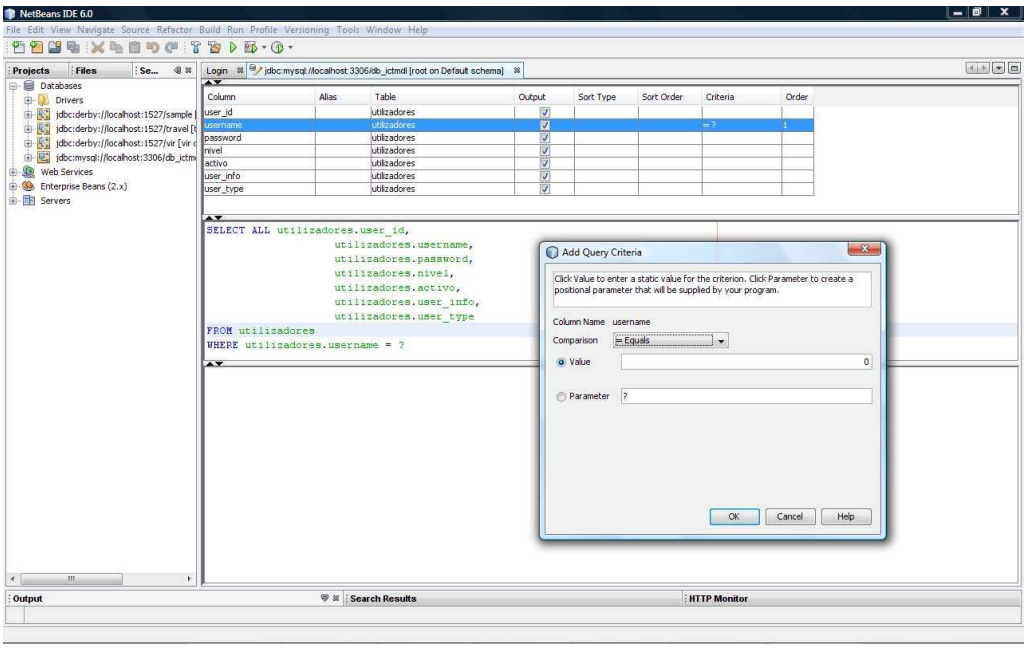

Figura 6.6: Configuração do *RowSet* para receber o nome do utilizador

O nome de utilizador submetido é pesquisado na tabela e, caso exista, a *password* introduzida é comparada com a registada na base de dados (excerto de código 6.1).

Excerto de Código 6.1 Pesquisa de um utilizador para a autenticação

```
1   getSessionBean1().getUtilizadoresRowSet().setObject(1, txtUser.getText());
2   utilizadoresDataProvider.refresh();
3   utilizadoresDataProvider.getCachedRowSet();
4
5   if (utilizadoresDataProvider.getRowCount() == 0){
6   stxtUser1.setText("O Nome de utilizador não é válido");
7   return "";
8   }
9   if (utilizadoresDataProvider.getRowCount() > 1){
10  stxtUser1.setText("Erro, mais do que um utilizador encontrado");
11  return "";
12  }
13
14  String strUser = (String) utilizadoresDataProvider.getValue("utilizadores.username");
15  String strPass = (String) utilizadoresDataProvider.getValue("utilizadores.password");
16  String strDesc = (String) utilizadoresDataProvider.getValue("utilizadores.user_info");
```

Se o utilizador estiver registado, é criada uma nova sessão e, nesta, um novo atributo que a relaciona com o nome do utilizador. Foi também criada um propriedade no *SessionBean* que identifica o tipo de utilizador para a aplicação saber, ao longo da navegação, quais os seus privilégios de acesso (excerto de código 6.2).

Excerto de Código 6.2 Criação da sessão após a autenticação de um utilizador

```
1   if (txtPsw.getText().equals(strPass) && txtUser.getText().equals(strUser)){
2
3   //Criar uma nova sessão:
4   FacesContext fc = FacesContext.getCurrentInstance();
5   session = (HttpSession) fc.getExternalContext().getSession(false);
6
7   if (intUsrLevel > 2){
8       getSessionBean1().setUserTypeBean("ADMIN");
9       strAux = "AdminLogin";
10      } else {
11      getSessionBean1().setUserTypeBean("USER");
12      strAux = "UserLogin";
13      }
14
15  session.setAttribute(strAux, txtUser.getText());
16  (...)
17  return strAux;
18  } else {
19  stxtUser1.setText("Nome de utilizador ou palavra chave inválidos!");
20  return "";
21  }
22  } catch (Exception ex){
23  stxtUser1.setText(ex.getCause().toString());
```

6.3.3 Terminação da Sessão

Para terminar a sessão o utilizador pode premir o botão de `Sair`. O método invocado é o `LogoutUser()` (excerto de código 6.3) e reside no `SessionBean`. A sessão é invalidada e o utilizador é redireccionado para a página de Autenticação.

Excerto de Código 6.3 Método para terminar a sessão

```
1  public void LogoutUser(){
2    this.setUserBean("");
3    this.setUserTypeBean("");
4    try{
5      FacesContext fc = javax.faces.context.FacesContext.getCurrentInstance();
6      ((HttpSession) fc.getExternalContext().getSession(false)).invalidate();
7      } catch (Exception e){
8      e.printStackTrace();
9      }
10   try{
11     getFacesContext().getExternalContext().redirect("Login.jsp");
12     } catch (Exception e){
13     e.printStackTrace();
14     }
15 }
```

6.3.4 Verificação da Validade da Sessão

A duração de uma sessão é trinta minutos. Se se tentar navegar na aplicação após este período, a aplicação remete o utilizador para a página de Autenticação. A verificação da validade da sessão é feita através de uma classe especial do tipo *filter*. Esta classe foi criada e configurada de acordo com o seguinte procedimento[4]:

1. Para a criação desta classe, clica-se com o botão direito do rato sobre o Projecto e escolher `New > Other`. No quadro que se abre, em `Categories`, escolher `Web` e em `File Types` escolher `Filter`. Prime-se `Next` para continuar;

2. Ao abrir-se o quadro `New Filter`, tem de se fornecer um nome para a classe em `Class Name` (neste caso o nome atribuído foi *"SessionCheckFilter"*). Em `Location` escolhe-se `Source Packages` e em `Packages` selecciona-se a aplicação (`tratadados`). Para continuar prime-se `Next`;

3. No quadro que se abre, selecciona-se o filtro criado e prime-se `edit`. Abre-se um pequeno quadro em que se deve seleccionar `Servlet` e escolher a

[4]Este procedimento está descrito no sítio da NetBeans no endereço:
http://www.netbeans.org/kb/55/sessionredirect.html

opção `Faces Servlet`. Prime-se `OK` para fechar o quadro. No quadro inicial prime-se `Finish` para terminar.

Por último, é necessário editar o código da classe `SessionCheckFilter`, substituindo todo o código existente (criado por omissão) pelo código que está descrito no excerto 6.4.

Excerto de Código 6.4 Verificação da validade da sessão actual

```
1 public class SessionCheckFilter implements Filter {
2
3   private static int firstRequest = 0;
4
5   public void doFilter(ServletRequest request, ServletResponse response,
6     FilterChain chain) throws IOException, ServletException {
7     HttpServletRequest hreq = (HttpServletRequest) request;
8     HttpServletResponse hres = (HttpServletResponse) response;
9     HttpSession session = hreq.getSession();
10    if (session.isNew()) {
11      if (firstRequest == 0) {
12        firstRequest++;
13      } else {
14        hres.sendRedirect("Login.jsp?session=logout");
15        return;
16      }
17    }
18    chain.doFilter(request, response);
19  }
20
21  public void init(FilterConfig filterConfig) throws ServletException {
22  }
23
24  public void destroy() {
25  }
26 }
```

6.4 Página de Entrada

Após a autenticação com sucesso do utilizador, é efectuado o redireccionamento para a página de entrada. No caso de ter privilégios de administração, é disponibilizada uma hiperligação para a página de gestão de utilizadores.

Esta página apenas contém ligações para acesso às restantes páginas da aplicação e não contém qualquer tipo de processamento lógico ou ligação a tabelas de dados.

Figura 6.7: Página de Entrada (*Home Page*)

Os elementos JSF que constituem esta página são:

- Dois objectos do tipo *GridPanel* que irão conter todos os restantes componentes, criando uma disposição em formato livre;
- Várias hiperligações para aceder às páginas seguintes, associadas a imagens ou texto;

A utilização dos objectos de posicionamento (*layout*) - os *GridPanel* - permitem dispor os elementos que neles são inseridos de uma forma fácil. As suas posições relativas são definidas de um modo transparente visto que estes se podem mover ou situar em qualquer ponto da página utilizando o rato ou o cursor do teclado.

No canto superior esquerdo, logo por baixo do título da aplicação, é indicada a localização actual da navegação. No lado direito aparecem as boas-vindas ao utilizador, seguido de um botão para terminar a sessão (*Sair*).

Na barra azul de cabeçalho, no lado esquerdo, aparece a ligação para a página de gestão de utilizadores e no lado direito existe uma barra de acesso rápido com as ligações para as restantes páginas. Estas características são comuns a todas as páginas.

Finalmente, no corpo da página estão dispostas três imagens que permitem efectuar a ligação às três páginas de consulta de dados. A descrição associada

a cada uma das imagens informa o utilizador sobre o tipo de consulta que lhe é oferecido em cada opção.

As hiperligações disponíveis dão acesso às seguintes páginas: *(i)* `Relatórios de Produção`, *(ii)* `Análise de Falhas` e *(iii)* `Informação Estatística`.

6.5 Página de Testes Efectuados

Esta página tem como objectivo mostrar todos os testes efectuados aos produtos e os seus resultados. Deve ser vista como a primeira página de consulta pois o utilizador encontra aqui, sem grandes detalhes, a informação sobre todos os produtos existentes na base de dados e as suas taxas de rejeição.

Para este tipo de abordagem decidiu-se utilizar um gráfico de barras verticais que mostra todos os produtos testados indicando as respectivas quantidades aprovadas e rejeitadas e, para o complementar, uma tabela contendo os valores exactos e as percentagens correspondentes.

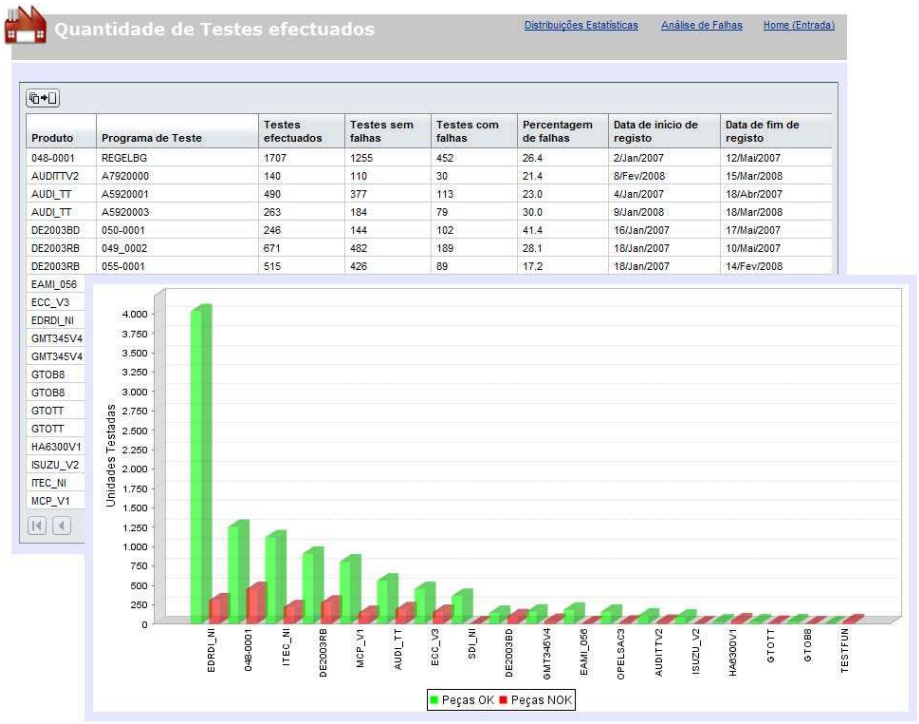

Figura 6.8: Aspecto geral da página de Testes efectuados aos produtos

O papel do utilizador nesta página é passivo. O gráfico é preenchido através do método `PlotBarChart()` que está localizado dentro do método `init()` da página.

6.5.1 Gráfico de Barras

O gráfico de barras utilizado pertence à biblioteca de código livre `JFreeChart` e permite mostrar os resultados através da utilização de um `DataSet` pertencente à mesma biblioteca.

Para o processo de preenchimento do gráfico é utilizado um `ResultSet` auxiliar e uma ligação à base de dados. A classe `java.sql.ResultSet` não é compatível com a classe `org.jfree.data.category.DefaultCategoryDataset`. Assim, o pedido ao servidor é executado preenchendo o objecto `ResultSet` que, de seguida, através de um pequeno ciclo, preenche um objecto `DataSet`. Uma vez terminado este ciclo, os dados contidos no `DataSet` são inseridos no gráfico através da propriedade `chartRelation.setDatasource(Dataset)`. Este objecto tem uma propriedade que especifica o tipo de gráfico a criar. Tem o nome de `type` e contém cerca de vinte e sete formatos. Isto é feito no quadro `Properties` do NetBeans com o gráfico seleccionado.

Excerto de Código 6.5 Inserção dos dados na gráfico de barras JFreeChart

```
1  private void PlotBarChart(){
2    Connection connection;
3    ResultSet RSet;
4    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
5    try{
6      Class.forName(SQLClass);
7      connection = DriverManager.getConnection(SQLDriver, SQLUsername, SQLPass);
8      StringBuffer sbSQL = new StringBuffer(BarChartSQLQuery);
9      Statement s = connection.createStatement();
10     RSet = s.executeQuery(sbSQL.toString());
11     try{
12       while (RSet.next()){
13         int intTot = RSet.getInt("Total");
14         int intFail = RSet.getInt("Falhas");
15         int intPercent = RSet.getInt("Percent");
16         int intOK = intTot - intFail;
17         dataset.addValue(intOK, "Peças OK", RSet.getString("Nome"));
18         dataset.addValue(intFail, "Peças NOK", RSet.getString("Nome"));
19         }
20       chartRelation.setYlabel("Unidades Testadas");
21       chartRelation.setDatasource(dataset);
22       } catch (Exception e){
23       e.printStackTrace();
24       }
25     connection.close();
26     } catch (SQLException e){
27     e.printStackTrace();
28     }
29 }
```

A *query* `BarChartSQLQuery` (ver o conteúdo em *Query* 6.1) utiliza a informação da tabela `ICT_MDL` e surge na linha 8 do Excerto de código 6.5.

***Query* 6.1** Valores para o gráfico de produtos testados

```
1  SELECT
2    A.Nome, A.Total, B.Falhas, TRUNCATE((B.Falhas*100/A.Total),1) AS Percent
3  FROM
4    (SELECT board AS Nome, COUNT(board) AS Total
5     FROM ict_mdl
6     GROUP BY board) AS A
7  LEFT JOIN
8    (SELECT DISTINCT board AS Nome, COUNT(board) AS Falhas
9     FROM ict_mdl
10    WHERE test_result=0
11    GROUP BY board) AS B
12 ON A.Nome=B.Nome
13 ORDER BY Total DESC
```

6.5.2 Tabela de Dados

Esta tabela pertence ao conjunto de objectos *JavaServer Faces*. A utilização deste componente recorre a um `Dataprovider` e a um `RowSet`, sendo este último associado directamente à tabela. Para efectuar a sua associação (*binding*) deve-se utilizar o mesmo procedimento da página de Autenticação, mas, desta vez, arrastando a tabela `ICT_MDL`. Aparece dentro do âmbito da página um `DataProvider` chamado de `ict_mdlDataProvider` e, dentro do `SessionBean`, aparece um `RowSet` identificado com o nome `ict_mdlRowSet`.

Para editar a *query* a enviar ao servidor MySQL basta clicar com o botão direito do rato sobre `ict_mdlRowSet` e seleccionar `Edit SQL Statement`. No campo de comandos SQL substituir o conteúdo existente pela *Query* 6.2[5].

A associação da tabela de dados ao `RowSet` é feita da seguinte maneira:

1. Clicar com o botão direito do rato em cima da tabela e seleccionar `Bind to Data`.

2. Seleccionar a *Drop-down list* com o nome `Get Data from:` e escolher a fonte de dados `ict_mdlDataProvider`.

3. Seleccionar os campos que aparecerem na lista do lado esquerdo para a do lado direito e confirmar com OK.

[5]Esta *query*, assim como todas as restantes neste projecto, foram previamente testadas no *MySQL Query Browser*.

Para personalizar a tabela deve-se clicar novamente com o botão direito do rato em cima da tabela e escolher *Table Layout*. Pode-se alterar os nomes que aparecem nas colunas, o número de linhas a aparecer na tabela, as opções de selecção, *etc*. A tabela deverá aparecer já com as colunas escolhidas.

Qualquer modificação que se realize sobre a *query* do `ict_mdlRowSet` afectará o `ict_mdlDataProvider` e, consequentemente, a tabela à qual está associado.

***Query* 6.2** Comando do *ict_mdlRowSet* que preenche a tabela tblGeral

```
1  SELECT
2     A.Nome AS Nome, A.Total AS Testadas, (A.Total - B.Falhas) AS UnitsOK,
3     B.Falhas AS UnitsNOK, TRUNCATE((B.Falhas*100/A.Total),1) AS Percent
4  FROM
5    (SELECT board AS Nome, COUNT(board) AS Total
6     FROM ict_mdl
7     GROUP BY board) AS A
8  LEFT JOIN
9    (SELECT DISTINCT board AS Nome, COUNT(board) AS Falhas
10    FROM ict_mdl
11    WHERE test_result=0
12    GROUP BY board) AS B
13 ON A.Nome=B.Nome
14 ORDER BY Total DESC
```

6.6 Página de Análise de Falhas

Esta página apresenta ao utilizador todos os componentes de um determinado produto que falharam durante o teste. As falhas destes componentes são a causa da rejeição do produto, sendo importante identificar os componentes mais críticos e planear o melhor método para a correcção do problema. Enquanto muitas vezes aparecem componentes com defeitos variados, outras vezes as falhas resultam de problemas no dispositivo de teste ou de metodologias de medição não optimizadas. Os elementos constituintes desta página são:

- Uma tabela com as colunas: *(i)* `Nome do componente`, *(ii)* `Descrição`, *(iii)* `Unidades testadas`, *(iv)* `Falhas`, *(v)* `Percentagem de falhas` e *(vi)* uma coluna com um botão para aceder aos detalhes do componente. Tal como a tabela da página de produtos testados, esta tabela também está associada a um `DataProvider`.

- Duas listas *Drop-Down*, uma delas para selecção do dispositivo de teste e, outra, para a selecção do produto. A primeira lista está configurada como *Auto Submit*, *i.e.*, ao ser seleccionada submete o seu valor. Isto é possível através da utilização de um evento, o `valueChange`. Para o activar, basta

clicar com o botão direito do rato e seleccionar `Auto-Submit on Change`. De seguida, ao fazer duplo clique sobre a lista é criado o método correspondente no separador de código Java, sendo possível definir o seu conteúdo. Neste caso, o método actualiza a lista de produtos com os produtos que foram testados naquele dispositivo.

- Dois elementos do tipo `Calendar` que permitem restringir a pesquisa a um intervalo de tempo específico.

- Um botão para submeter a selecção e mostrar os resultados.

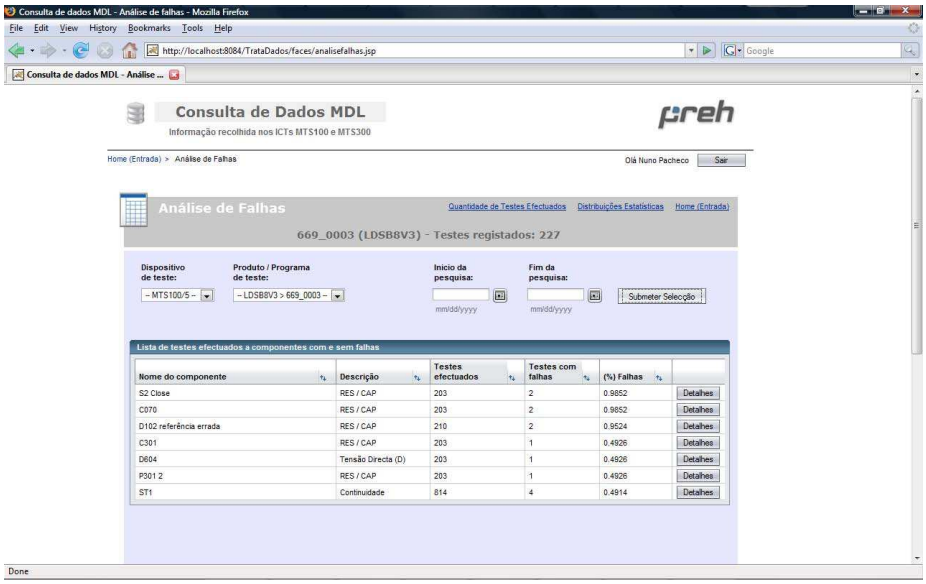

Figura 6.9: Página de análise de componentes rejeitados

6.6.1 Funcionamento

O utilizador deverá, em primeiro lugar, escolher um dispositivo de teste. Se desejar consultar um determinado produto independentemente do dispositivo, pode escolher a opção `<Todos>`. A lista de produtos é actualizada automaticamente. O pedido à base de dados que permite preencher a lista de produtos pode ser visto na *Query* 6.3.

***Query* 6.3** Preenchimento da lista de produtos

```
1  SELECT DISTINCT
2    BOARD FROM ict_mdl
3  INNER JOIN
4    ict_mdl_icts
5  ON
6    ict_mdl.ict_id = ict_mdl_icts.ict_mdl_ict_id
7  WHERE
8    ict_mdl_icts.ict_name='MTS100/1'
9  GROUP BY BOARD
10 ORDER BY BOARD
```

O utilizador tem a possibilidade de escolher um período de tempo para restringir a sua consulta. Deverá, neste caso, escolher uma data de início e de fim. Ambas têm de ser preenchidas ou, no caso de uma consulta global, deverá deixar ambas em branco.

Depois de feitas as escolhas a submeter, a tabela é preenchida com todos os componentes que falharam no teste. Esta revela o nome do componente, a sua descrição (tipo de componente), a quantidade total de componentes testados, a quantidade com falhas e a percentagem de falhas. Estes valores podem ser limitados pelo intervalo de tempo escolhido e pelo dispositivo de teste escolhido.

Componentes com falhas

Nome Componente	Descrição	Unid. Testadas	Falhas	(%) Falhas	
voltage Uref	RES / CAP	1161	13	1.1197	Detalhes
P1	RES / CAP	1193	6	0.5029	Detalhes
C1	RES / CAP	1193	3	0.2515	Detalhes
STATUS temperatur meas	RES / CAP	1158	2	0.1727	Detalhes
D7	Tensão Inversa (D)	1190	2	0.1681	Detalhes
C6	RES / CAP	1191	2	0.1679	Detalhes
C3	RES / CAP	1191	1	0.0840	Detalhes
D9	Tensão Inversa (D)	1190	1	0.0840	Detalhes
C7	RES / CAP	1191	1	0.0840	Detalhes
D12	Tensão Inversa (D)	1190	1	0.0840	Detalhes
C9	RES / CAP	1192	1	0.0839	Detalhes
C131	RES / CAP	1192	1	0.0839	Detalhes
C51	RES / CAP	1193	1	0.0838	Detalhes
R14	RES / CAP	1193	1	0.0838	Detalhes
C11	RES / CAP	1193	1	0.0838	Detalhes
C501	RES / CAP	1193	1	0.0838	Detalhes
R19	RES / CAP	1193	1	0.0838	Detalhes
C8	RES / CAP	1193	1	0.0838	Detalhes
C12	RES / CAP	1193	1	0.0838	Detalhes

Figura 6.10: Tabela de componentes testados com falhas

O botão "Detalhes" reencaminha a navegação para a página de distribuições normais que será descrita mais à frente neste capítulo. Interessa, no entanto, mencionar que os detalhes permitem obter informação acerca da média dos valores medidos, do desvio padrão e de outros parâmetros de controlo. O utilizador pode ter interesse em analisar o comportamento de um determinado componente e consegue fazê-lo através desta ligação sem ter de memorizar qual o componente, produto ou dispositivo em questão.

Esta tabela está associada ao DataProvider ict_mdlDataProviderAnalise e ao RowSet ict_mdlRowSetAnalise. O pedido ao servidor MySQL que está configurado por omissão é *dummy* ou falso. Como para esta tabela os dados dependem das escolhas feitas pelo utilizador, quando a página carrega, a tabela é actualizada com um pedido que não devolve qualquer tipo de informação, mas respeita todos os seu campos e tipos de dados.

O comando executado para preencher a tabela é um pouco extenso (*Query* 6.4) e, dependendo da quantidade de informação a pesquisar na base de dados, pode tornar-se um pouco moroso. Neste caso, existe a necessidade de cruzar dados das tabelas de componentes (ICT_MDL_VALUES), de produtos (ICT_MDL) e de tipos de medição (ICT_MDL_TYPES).

***Query* 6.4** Comando para preencher a tabela de análise de falhas

```
1  SELECT
2    T.component_name, R.ict_mdl_types_id AS myType, R.TypeDescription,
3    T.Total, D.Defects, (100*D.Defects/T.Total) AS Per100
4  FROM
5    (SELECT component_name, component_type, COUNT(ICT_MDL_ID) AS Total
6     FROM ict_mdl_values
7     WHERE ict_mdl_id IN
8       (SELECT ict_mdl_id
9        FROM ict_mdl
10       WHERE board = 'produto'
11       AND start_time BETWEEN '2008-01-01 00:00:00' AND '2008-02-01 00:00:00'
12       AND ict_id = '1')
13    GROUP BY component_name, component_type) AS T
14 RIGHT JOIN
15   (SELECT component_name, component_type, COUNT(ICT_MDL_ID) AS Defects
16    FROM ict_mdl_values
17    WHERE comp_test_result=0 AND ict_mdl_id IN
18      (SELECT ict_mdl_id
19       FROM ict_mdl
20       WHERE board = 'produto'
21       AND start_time BETWEEN '2008-01-01 00:00:00' AND '2008-02-01 00:00:00'
22       ANDict_id = '1')
23    GROUP BY component_name, component_type) AS D
24 ON T.component_name=D.component_name AND T.component_type=D.component_type
25 LEFT JOIN
26   ict_mdl_types R
27 ON T.component_type=R.ict_mdl_types_id
28 ORDER BY Per100 DESC
```

6.7 Página de Análise Estatística

Nesta página são calculados e disponibilizados ao utilizador os parâmetros estatísticos relevantes para a análise do processo de teste dos produtos. Estão disponíveis para selecção os dispositivos de teste, os produtos, os componentes e as diferentes medidas efectuadas[6].

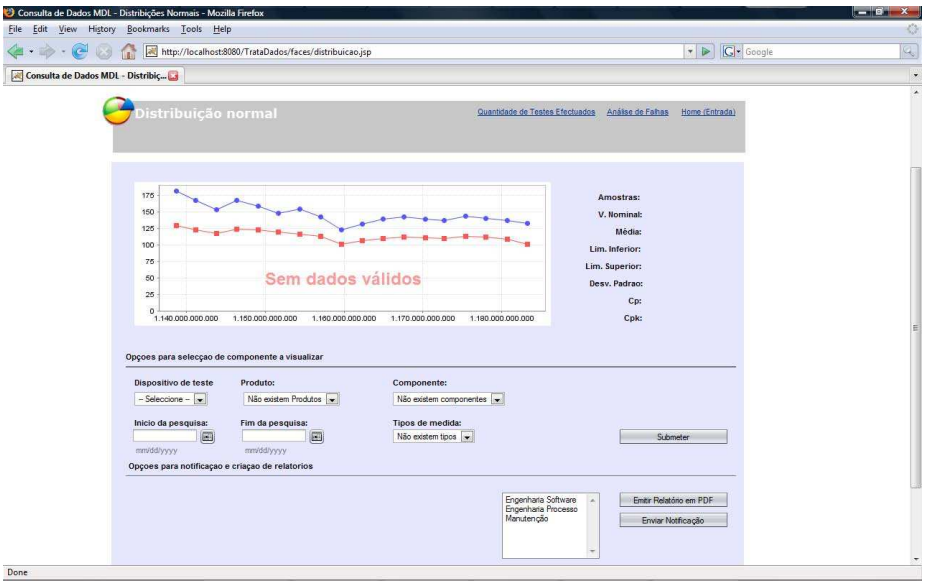

Figura 6.11: Página de análise estatística de componentes

Para a construção desta página foram utilizados os seguintes objectos:

- Um gráfico da biblioteca `JFreeChart` cujo objectivo é mostrar a distribuição normal dos valores medidos. Este gráfico é idêntico ao utilizado na página de testes efectuados. Para traçar curvas ou distribuições, a propriedade `type` foi configurada para `XY Line`;
- Quatro listas *drop-down* das quais três submetem o seu valor automaticamente quando seleccionadas, utilizando o evento `valueChange`;
- Um botão para a criação de um relatório em PDF com os resultados da análise escolhida, utilizando a biblioteca *JasperReports*;

[6]Poderá existir mais do uma abordagem ao mesmo componente durante o teste de um produto, *e.g.*, no caso dos díodos, são efectuadas duas medições: a sua tensão directa e inversa.

- Um botão para o envio de notificações por *email* que incluem os relatórios gerados;
- Um botão para a submissão das escolhas efectuadas.

6.7.1 Funcionamento

Tal como na página anterior de análise de falhas, o utilizador da página de análise estatística tem que escolher um dispositivo de teste (ou todos), escolher um produto e, posteriormente, escolher o componente a analisar. As listas, quando são seleccionadas, comportam-se da seguinte maneira:

1. A lista de dispositivos invoca o método `drpICT_processValueChange` para preencher a lista de produtos;
2. A lista de produtos recorre ao método `drpProduto_processValueChange` para actualizar a lista de componentes;
3. A lista de componentes invoca o método `drpComponente_processValueChange` para actualizar a lista de tipos de medidas.

Uma vez efectuada a selecção desejada, quando se prime o botão `"Submeter"`, é construída uma *query* que é enviada à base de dados, sendo retornados no `ResultSet` e para cada registo os seguintes campos:

- `nome` - Nome do componente;
- `NV` - Valor nominal;
- `unidades` - Unidade de medida;
- `LL` - Limite de tolerância inferior;
- `HL` - Limite de tolerância superior;
- `media` - Valor médio;
- `desvpdr` - Desvio padrão amostral;
- `amostras` - Número de amostras.

Os valores da média e desvio padrão são calculados no servidor MySQL através das funções `AVG` (média) e `STDDEV_SAMP` (desvio padrão) - ver *Query* 6.5.

Todos estes valores são mostrados em *labels* situadas à direita do gráfico. Para a criação do gráfico apenas são utilizados os valores da média, do desvio padrão e dos limites de tolerância superior e inferior.

***Query* 6.5** Comando que retorna parâmetros estatísticos

```
1  SELECT
2    V.component_name AS Nome, TRUNCATE(V.nominal_value,2) AS NV,
3    U.unitname AS Unidades, TRUNCATE(V.low_tolerance,2) AS LL,
4    TRUNCATE(V.high_tolerance,2) AS HL, TRUNCATE(AVG(V.measured_value),4) AS MEDIA,
5    TRUNCATE(STDDEV_SAMP(V.measured_value),4) AS DESVPDR, COUNT(V.measured_value) AS Amostras,
6  FROM ict_mdl_values V
7  INNER JOIN ict_mdl M
8  ON V.ict_mdl_id=M.ict_mdl_id
9  INNER JOIN ict_mdl_units U
10 ON V.component_unit=U.ict_mdl_units_id
11 INNER JOIN ict_mdl_types Q
12 ON V.component_type = Q.ict_mdl_types_id
13 WHERE board = 'produto'
14 AND ict_id = '1'
15 AND component_name = 'C6'
16 AND component_type = 'Tipo'
17 AND start_time BETWEEN '2008-01-01 00:00:00' AND '2008-02-01 00:00:00'
18 GROUP BY V.nominal_value
```

6.7.2 Gráfico de Distribuições Normais

As classes e interfaces necessárias para visualizar a distribuição normal no gráfico são:

- A interface `org.jfree.data.function.Function2D` que é implementada pelas classes:
  - `Function2D`:
  - `NormalDistributionFunction2D`;
  - `PowerFunction2D`.
- A interface `org.jfree.data.function.NormalDistributionFunction2D`;
- A interface `org.jfree.data.general.DatasetUtilities`.

A interface `Function2D` permite criar uma função matemática que, neste caso, será associada a uma nova instância da classe `NormalDistributionFunction2D`. Esta instância recebe como argumentos os valores médio e desvio padrão e permite preencher um `DataSet` para criar o gráfico correspondente.

Excerto de Código 6.6 Método para criar a distribuição normal no gráfico

```
1  (...)
2  Function2D Norm = new NormalDistributionFunction2D(media, desvpadr);
3  XYDataset DSet = DatasetUtilities.sampleFunction2D(Norm, Start, End, NSamp, "Normal");
4  chartNormalDistr.setDatasource(DSet);
5  (...)
```

O objecto `"Norm"` (ver excerto de código 6.6) representa a função de distribuição normal criada com os valores médio e desvio padrão especificados. Os parâmetros `"Start"` e `"End"` referem-se aos limites inferior e superior do eixo dos X, utilizando-se aqui o intervalo de tolerância do componente.

O parâmetro `"NSamp"` refere-se ao número de amostras ou componentes medidos.

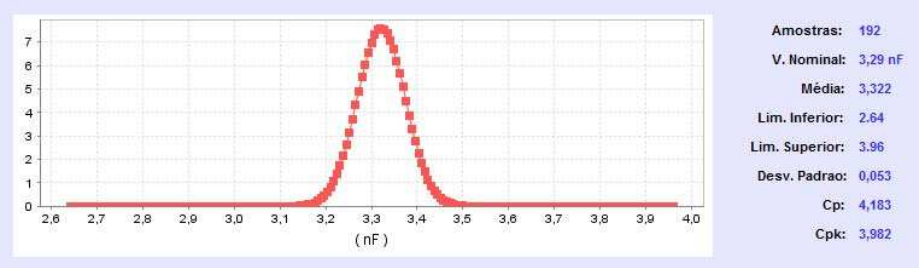

Figura 6.12: Distribuição normal dos valores de um componente

Existem dois parâmetros que aparecem à direita do gráfico chamados Cp e Cpk. O primeiro, conhecido por Capabilidade do Processo, é definido como o intervalo da tolerância especificada sobre 6 vezes o desvio padrão (Equação 6.1). Este índice é independente do desvio do valor médio em relação ao centro do gráfico. O Cpk, ou Índice de Capabilidade de Processo (Equação 6.2), dá-nos informação sobre o desvio que o valor médio apresenta em relação ao ponto central do intervalo dos limites superior e inferior. Este são parâmetros para controlo estatístico de processo, bastante úteis para a respectiva análise do desempenho.

$$Cp = \frac{(L_s - L_i)}{6\sigma} \qquad \text{(Equação 6.1)}$$

$$Cpk = \min\left(\frac{L_s - \overline{x}}{3\sigma}, \frac{\overline{x} - L_i}{3\sigma}\right) \qquad \text{(Equação 6.2)}$$

6.7.3 Criação de Relatórios em PDF

Como já foi referido no Capítulo 4, esta aplicação permite gerar relatórios no formato PDF. A Figura 6.13 mostra um relatório gerado. Este relatório contém toda a informação gerada pela página, permitindo guardar o resultado num documento PDF para posterior consulta ou simples arquivo.

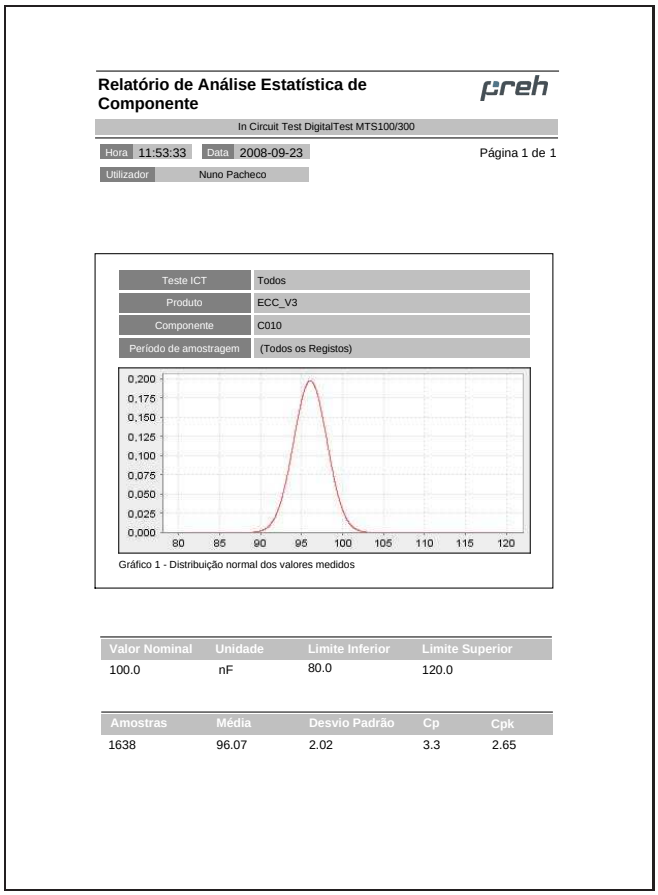

Figura 6.13: Relatório PDF de análise de um componente

Durante a construção do método de criação do relatório, deparou-se com a dificuldade da conversão do gráfico numa imagem. Na documentação disponibilizada no sítio da JFree, o procedimento utilizado passa pela declaração local de um novo gráfico que não é colocado no formulário da página. Este novo gráfico tem uma iniciação semelhante à do original e permite a utilização do método `createBufferedImage` para produzir uma imagem. O gráfico original que é um objecto JSF e pertence à classe `chart` do pacote `net.sf.jschart.component` não disponibiliza este método de conversão. Os objectos que suportam este método de conversão pertencem à classe `JFreeChart` do pacote `org.jfree.chart`.

O método utilizado para gerar o relatório - `GeneratePDFReport()` (Excerto de código 6.7) - utiliza o objecto `Map` para conter as imagens a enviar. Assim, através do método criado no *ApplicationBean* - `jasperReport()` - são especificados: *(i)* o nome do modelo do relatório, *(ii)* o tipo de relatório (PDF ou texto/HTML), *(iii)* os valores estatísticos a incluir (média, desvio padrão, *etc.*), e *(iv)* as imagens. O relatório abrir-se-á numa nova janela do navegador (se este estiver configurado

para abrir este tipo de documento) ou questionará o utilizador se o deseja abrir com outro programa ou guardar, sendo este, no entanto, sempre guardado pela aplicação.

Excerto de Código 6.7 Método *GeneratePDFReport()*

```
1  private void GeneratePDFReport(){
2
3    Map fillParams = new HashMap();
4    try{
5      XYDataset DSet = getSessionBean1().getDSet();
6      JFreeChart myChart = ChartFactory.createXYLineChart(
7            "", "", "", DSet, PlotOrientation.VERTICAL, false, false, false);
8      fillParams.put("LogoURL", myChart.createBufferedImage(400, 180));
9      fillParams.put(
10           "PrehLogo", getExternalContext().getResource("/resources/LogoPREH2.gif"));
11     getApplicationBean1().jasperReport(
12           "ProductionReport", "application/pdf", getSessionBean1().getRs(), fillParams);
13     } catch (Exception e){
14     log("Exception generating report", e);
15     error("Exception generating report: " + e);
16     }
17 }
```

6.7.4 Envio de Notificações por Correio Electrónico

Após a criação do relatório, é possível enviar uma notificação por *email*. Esta notificação transporta a informação recolhida sobre os equipamentos de teste, produtos e componentes seleccionados na página de análise estatística e, na qual, se gerou um relatório. Este relatório é incluído como anexo na mensagem de *email* a enviar.

Para enviar uma notificação é necessário gerar um relatório e seleccionar os destinatários da notificação. Existe uma pequena *listbox* onde se encontram os contactos dos departamentos que poderão ser intervenientes em acções correctivas sobre o processo de teste. É possível seleccionar mais do que um contacto. Após o envio da notificação, uma pequena mensagem surge na página, informando o utilizador sobre o sucesso do processo de envio.

Os principais objectos a utilizar da API *JavaMail* são os seguintes:

- Objecto `MimeMessage` - especifica o modelo de uma mensagem de *email* e contém toda a informação relativa ao corpo da mensagem, aos anexos, ao remetente e aos destinatários;
- Objecto `mimeBodyPart` - define o corpo da mensagem e anexos;
- Objecto `Transport` - responsável pelo estabelecimento da ligação ao servidor e pelo envio da mensagem.

O método responsável pelo envio do correio electrónico é apresentado no Excerto de Código 6.8.

Excerto de Código 6.8 Método *SendEmail()*

```
1  private void SendEmail() throws NoSuchProviderException, MessagingException {
2    if (getSessionBean1().getReportName() != null) {
3      if (!getSessionBean1().getReportName().equals("")) {
4        String[] Addresses = (String[]) multiSelectListbox1.getSelectedValues();
5        if (Addresses.length == 0) {
6          notifyResultMsg.setText("Deve seleccionar pelo menos um destinatário!");
7          return;
8        }
9        boolean debug = false;
10       Properties props = new Properties();
11       props.setProperty("mail.transport.protocol", "smtp");
12       props.setProperty("mail.host", "smtp.ipp.pt");
13       props.setProperty("mail.user", "1000231");
14       props.setProperty("mail.password", "xxxxx");
15
16       Session mailSession = Session.getDefaultInstance(props, null);
17       mailSession.setDebug(debug);
18
19       Transport transport = mailSession.getTransport();
20       MimeMessage message = new MimeMessage(mailSession);
21       message.setSubject("Notificação de falha em Teste ICT");
22
23       MimeBodyPart textPart = new MimeBodyPart();
24       textPart.setContent("  Corpo da mensagem .. ");
25       MimeBodyPart attachFilePart = new MimeBodyPart();
26       FileDataSource fds = new FileDataSource(ReportPath + Reportname.pdf");
27       attachFilePart.setDataHandler(new DataHandler(fds));
28       attachFilePart.setFileName(fds.getName());
29
30       Multipart mp = new MimeMultipart();
31       mp.addBodyPart(textPart);
32       mp.addBodyPart(attachFilePart);
33       message.setContent(mp);
34       message.setFrom( new InternetAddress("1000231@isep.ipp.pt"));
35       for (int I = 0; I < Addresses.length; I++) {
36         message.addRecipient(Message.RecipientType.TO, new InternetAddress(Addresses[I]));
37       }
38       transport.connect();
39       transport.sendMessage(message, message.getRecipients(Message.RecipientType.TO));
40       transport.close();
41       notifyResultMsg.setText("Notificação enviada com sucesso!");
42     }
43   }
44 }
```

6.8 Página de Gestão de Utilizadores

Esta página é reservada ao administrador da aplicação. O seu objectivo é gerir os utilizadores da aplicação.

Para a construção desta página utilizaram-se dois objectos: uma tabela que lista os utilizadores autorizados e um conjunto de dois separadores que permitem: *(i)* adicionar um novo utilizador; *(ii)* alterar as definições de um utilizador existente; e *(iii)* remover um utilizador.

Figura 6.14: Página para gestão de utilizadores

A tabela está ligada a um `RowSet` - `manageRowSet` - que é carregada automaticamente em conjunto com a página. Foi adicionada uma coluna extra com componentes do tipo `radiobutton` cuja finalidade é seleccionar uma linha, ou seja, seleccionar um utilizador. No entanto, este objecto, por si só, não permite seleccionar a linha respectiva da tabela pois não está configurado para ser submetido automaticamente. Foi necessário adicionar um pequeno guião (*script*) associado ao evento `onClick` do `radioButton` (excerto de código 6.9).

Excerto de Código 6.9 *radioButton* com submissão automática

```
webui.suntheme.common.timeoutSubmitForm(this.form,
        'table1:tableRowGroup1:tableColumn5:radioButton1');
```

Este método permite submeter à página qual dos elementos da linha da tabela foi escolhido e, assim, transferir os dados do utilizador para o separador de edição.

6.8.1 Adicionar novo Utilizador

Os campos disponibilizados no separador para a inserção de um novo utilizador são:

- *Username* (máximo vinte caracteres);
- *Password* (máximo vinte caracteres);
- Breve descrição (máximo quarenta e cinco caracteres);
- Nível de acesso;
- Selecção de utilizador (activado ou desactivado).

Figura 6.15: Separador de inserção de um novo utilizador

Todos os campos são de preenchimento obrigatório. No caso da *password*, esta tem de ser introduzida duas vezes para garantir que não houve enganos.

Aquando da submissão, é verificada a existência do *username* na base de dados. Se já existir, é pedido para ser introduzido um nome diferente. O método utilizado para efectuar esta verificação chama-se `CheckUserAvailability(String`

`User)` e retorna `true` ou `false` consoante existe ou não disponibilidade (excerto de código 6.10) para criar esse *username*.

Excerto de Código 6.10 Método para verificação da existência de um utilizador

```
1  public boolean CheckUserAvailability(String strUsr)
2    {
3    manageDataProvider.cursorFirst();
4    RowKey R = manageDataProvider.findFirst("username", strUsr);
5    if (R == null)
6      return true;
7    return false;
8    }
```

Caso se trate de um novo utilizador, os seus dados são inseridos e a tabela é actualizada, contemplando já o novo registo. Se o utilizador inserido tiver sido marcado como desactivado, este não poderá aceder à aplicação. Este campo torna mais simples a gestão de um grupo grande de utilizadores, podendo a inserção de todos os membros do grupo ser efectuada de uma só vez e, posteriormente, filtrarem-se os que se pretendem inibir.

O método que insere o utilizador na base de dados está descrito no excerto de código 6.11.

Excerto de Código 6.11 Método para adicionar um novo utilizador

```
1  public boolean SavetoDB(){
2    try{
3      if (manageDataProvider.canAppendRow()){
4        RowKey R = manageDataProvider.appendRow();
5        if (R == null) return false;
6        manageDataProvider.setCursorRow(R);
7        manageDataProvider.setValue("utilizadores.username", txtNewUser ...
8        manageDataProvider.setValue("utilizadores.password", pswNewPass ...
9        manageDataProvider.setValue("utilizadores.user_info", txtDescription ...
10       manageDataProvider.setValue("utilizadores.activo", "1");
11       manageDataProvider.setValue("utilizadores.user_type", "ADMIN");
12       manageDataProvider.commitChanges();
13       manageDataProvider.refresh();
14       }
15     } catch (Exception e){
16     return false;
17     }
18   return true;
19   }
```

6.8.2 Alteração de Dados de um Utilizador

Os dados do utilizador podem ser alterados em qualquer altura, nomeadamente a *password*, o nível de acesso, a descrição e a sua activação ou desactivação. Tal

como foi descrito acima, ao seleccionar uma linha da tabela, estes campos são disponibilizados no separador de edição.

Figura 6.16: Separador de edição de um utilizador existente

Existem quatro campos de texto para inserir os novos dados. Também aqui todos os campos têm de ser preenchidos e, se o novo nome de utilizador for diferente, também será verificada a sua disponibilidade.

Excerto de Código 6.12 Método para alteração dos dados de um utilizador

```
1  public boolean UpdateUserInDB( ... ){
2    try{
3      manageDataProvider.cursorFirst();
4      RowKey R = manageDataProvider.findFirst("username", ActualUsername);
5      if (R != null){
6        manageDataProvider.setCursorRow(R);
7        manageDataProvider.setValue("username", NewUsername);
8        manageDataProvider.setValue("password", NewPassword);
9        manageDataProvider.setValue("activo", Activo);
10       manageDataProvider.setValue("user_info", NewDescription);
11       manageDataProvider.setValue("user_type", UserType);
12       manageDataProvider.commitChanges();
13       manageDataProvider.refresh();
14       return true;
15       }
16     return false;
17     } catch (Exception e){
18     return false;
19     }
20   }
```

No mesmo separador é disponibilizado um botão para a remoção do utilizador seleccionado. No entanto, para prevenir que se pressione este botão por acidente, foi adicionado um pequeno *script* que abre uma pequena janela com uma mensagem de aviso (*alertbox*), onde o administrador deverá confirmar a operação.

Este *script* foi adicionado ao método que é executado quando é disparado o evento `onClick` do botão `Remover`. Para isto basta seleccionar o botão e, na tabela de propriedades do objecto no IDE, clicar no campo `JavaScript > onClick` e adicionar o *script* representado no Excerto de Código 6.13.

Excerto de Código 6.13 *Script* para confirmação da remoção de um utilizador

```
return confirm('Tem a certeza que deseja remover este Utilizador?')
```

Figura 6.17: Resultado da execução do *script* 6.13

6.9 Conclusão

A aplicação global que foi desenvolvida comporta a aplicação *web* `TrataDados` e a aplicação de conversão e armazenamento dos dados de teste `MDLConverter`. As funcionalidades implementadas na aplicação *web* realizam as tarefas de processamento da informação gerada durante o processo de teste ICT, detecção e análise de falhas de componentes, criação de relatórios e desencadeamento de acções correctivas através dos navegadores dos colaboradores do Departamento de *Software* e Testes da Preh.

Capítulo 7

Instalação da Aplicação no Servidor Apache Tomcat

Este capítulo descreve sucintamente o processo de compilação e instalação da aplicação web num servidor de aplicações Apache Tomcat externo ao IDE NetBeans.

7.1 Compilação da Aplicação

Para exportar a aplicação para o servidor de aplicações, o NetBeans permite compilar todos os ficheiros de código fonte e todas as bibliotecas utilizadas para um ficheiro de saída do tipo *Java Archive* (`jar`) ou *Web Application Archive* (`war`). No caso da aplicação de conversão dos ficheiros de texto, o ficheiro de saída resultante é do tipo `jar` e, no caso da aplicação *web*, é do tipo `war`.

O processo de criação destes ficheiros de saída chama-se `Build`. Deve-se, no entanto, verificar, caso exista mais do que uma aplicação em desenvolvimento, que está seleccionada a que se pretende compilar. Esta deverá estar com um tipo de letra mais forte ou a negrito. Se não estiver, deve-se clicar com o botão direito do rato sobre a aplicação e escolher `Set as Main Project`. De seguida, basta clicar no botão `Build` ou premir `F11`. O processo demora alguns segundos e, na janela de `Output`, são mostrados os passos efectuados e o respectivo resultado.

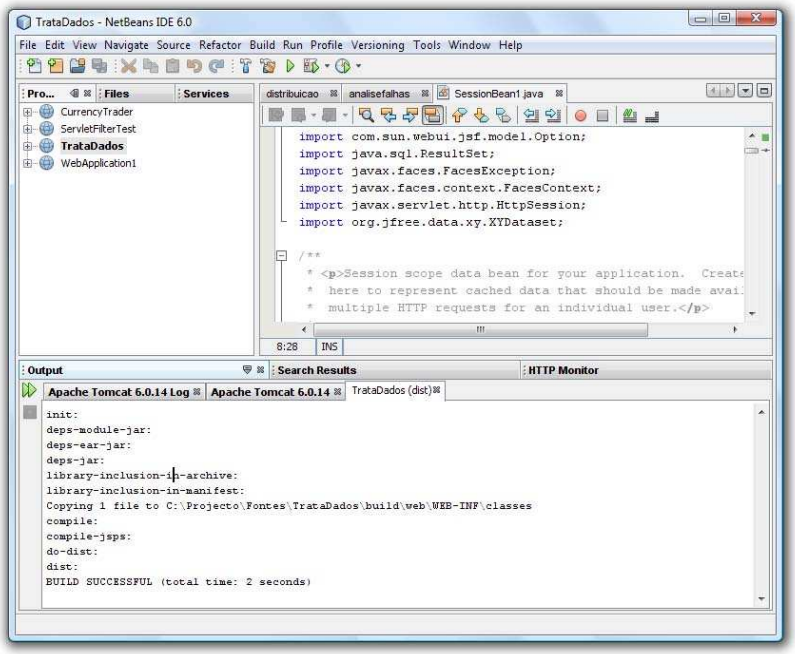

Figura 7.1: Compilação da aplicação *web*

7.2 Configuração da Aplicação no Apache Tomcat

O NetBeans coloca no directório de trabalho da aplicação dentro da pasta com o nome `dist` o ficheiro `war` resultante.

O modo mais simples para se configurar a aplicação no Apache Tomcat é o seguinte:

- Verificar que o servidor Apache Tomcat está desligado;
- Copiar o ficheiro `war` para o directório das aplicações do Apache Tomcat. Dependendo do directório de instalação, esta pasta localiza-se em `C:\(...)\Apache_Software_Foundation\Tomcat_6.0.16\webapps`;
- Iniciar o servidor.

Ao iniciar, o servidor detecta a existência do novo ficheiro `war` e desencadeia a instalação - *deployment* - da aplicação, *i.e.*, descomprime o ficheiro e cria a estrutura de directórios e ficheiros para o funcionamento da aplicação. Para aceder à aplicação basta escrever o endereço: `http//localhost:8080/TrataDados`.

Capítulo 8

Avaliação da Aplicação em Ambiente Produtivo

Este capítulo apresenta algumas das vantagens resultantes da utilização da aplicação desenvolvida no ambiente de produção.

Interessa, em primeiro lugar, esclarecer um facto importante antes de referir os benefícios resultantes da adopção desta ferramenta. A informação que é gerada sob a forma de ficheiros de texto no processo de teste ICT não inclui qualquer tipo de identificação em relação às placas electrónicas testadas. Significa isto que as peças que falham no processo de teste, depois de serem recuperadas e novamente testadas, são contabilizadas como peças novas por não existir um modo de as identificar. O mesmo acontece com peças que são testadas repetidamente.

De facto, se forem produzidas 100 peças e, destas, cinco não passarem no teste, obter-se-ia uma taxa de rejeição (*falloff*) de 5 %. Contudo, estas ao serem recuperadas, novamente testadas e finalmente aprovadas no processo de teste, resultaria um *falloff* de 4,7 %. Obviamente, neste cenário, não faz sentido falar de *falloff* porque as cinco peças inicialmente testadas e posteriormente recuperadas foram duplamente contabilizadas. Este facto deve ser sempre levado em conta pelo utilizador quando interpretar os dados apresentados por esta aplicação. Assim, todas as quantidades indicadas deverão ser interpretadas como quantidades de **testes** efectuados e não quantidades de **peças** testadas.

No entanto, a quantidade de peças que entra novamente nas linhas de teste ICT é muito inferior à quantidade que é testada no seu fluxo de processo normal. É possível aproximar-se a quantidade mostrada pela aplicação com a quantidade real de peças testadas, desde que contraposta com outros dados, sejam estes

provenientes de um sistema de rastreio criado para o efeito ou de outros Departamentos como os de Produção, Planeamento e Qualidade.

De seguida, vão ser analisadas duas situações onde a ferramenta desenvolvida contribui para a detecção e resolução atempada de problemas: *(i)* durante o funcionamento normal da linha de produção; e *(ii)* na fase de arranque de produção de um novo produto.

8.1 Normal Funcionamento da Linha de Produção

Através da consulta de alguns dados armazenados, identificaram-se os produtos que foram rejeitados na fase de teste e, através da página de relatórios de teste, foram seleccionados aqueles que apresentaram percentagens de rejeição mais elevadas. O estudo subsequente centrou-se nestes produtos para verificar qual o tipo de intervenção a efectuar. O produto escolhido está apresentado no Quadro 8.1.

```
Dispositivo de Teste .............: Todos
Produto ..........................: DACMOD~1
Programa de Teste ................: V6550002
Período de registo ...............: 02.03.2008 a 21.03.2008
Testes efectuados ................: 399
```

Quadro 8.1: Produto alvo de análise

De seguida navegou-se para a página de análise de falhas seleccionando os seguintes critérios de pesquisa: *(i)* mostrar todos os dispositivos de teste, *(ii)* seleccionar o produto `DACMOD~1 V6550002` e *(iii)* não limitar o período de registo das amostras. Os resultados obtidos são apresentados na Figura 8.1. No Quadro 8.2 mostram-se os componentes deste que mais falharam.

```
1º componente com mais falhas ....: 100 % (Signal Frequency)
2º componente com mais falhas ....: 100 % (CAN Communication Fail)
3º componente com mais falhas ....: 100 % (CAN - READ SERIAL NUM NOK)
4º componente com mais falhas ....: 100 % (Timeout-no SignalFrequency)
5º componente com mais falhas ....: 8,27 % (C090)
6º componente com mais falhas ....: 6,39 % (ST1_1,2,8,13)
7º componente com mais falhas ....: 4,29 % (T012-EB)
```

Quadro 8.2: Componentes com mais falhas

Lista de testes efectuados a componentes com e sem falhas

Nome do componente	Descrição	Testes efectuados	Testes com falhas	(%) Falhas	
CAN - READ SERIAL NUM NOK	RES / CAP	3	3	100.0000	Detalhes
Signal Frequency	RES / CAP	10	10	100.0000	Detalhes
Timeout - no SignalFrequency	RES / CAP	1	1	100.0000	Detalhes
CAN Communication Fail	RES / CAP	63	63	100.0000	Detalhes
C090	RES / CAP	399	33	8.2707	Detalhes
ST1_1,2,8,13	Continuidade	391	25	6.3939	Detalhes
T012-EB	RES / CAP	396	17	4.2929	Detalhes
T012-CB	RES / CAP	396	11	2.7778	Detalhes
D052	Tensão Directa (D)	398	10	2.5126	Detalhes
Codigo Hardware R171,R173 e R175	Continuidade	500	10	2.0000	Detalhes
T011-BE	RES / CAP	396	4	1.0101	Detalhes
T101	Tensão UBC (T)	230	2	0.8696	Detalhes
PCB Errado	Continuidade	1173	10	0.8525	Detalhes
D101	Tensão Directa (D)	396	3	0.7576	Detalhes
C003	RES / CAP	397	2	0.5038	Detalhes
R405	RES / CAP	384	1	0.2604	Detalhes
ST2_6 14 ST1_7,11	Continuidade	390	1	0.2564	Detalhes
ST1_3,5,7,11,14	Continuidade	391	1	0.2558	Detalhes
R402	RES / CAP	394	1	0.2538	Detalhes
R131//C130	Paralelo CAP	395	1	0.2532	Detalhes
R003	RES / CAP	396	1	0.2525	Detalhes
C013	RES / CAP	396	1	0.2525	Detalhes

Page: 1 of 2 Go

Figura 8.1: Tabela de componentes do produto DACMOD1 V6550002.

Verifica-se que os quatro primeiros elementos apresentam uma taxa de falhas de 100 %. Neste caso específico não se trata de medições a componentes mas sim de resultados de testes efectuados a um protocolo de comunicação estabelecido entre o dispositivo de teste e a peça. Estes dados apenas aparecem registados no caso da comunicação não se estabelecer e, daí, apresentarem um rejeição de 100 %.

O primeiro componente da lista cujos valores são medidos é o condensador C90 que apresenta uma taxa de falhas de 8,27 %.

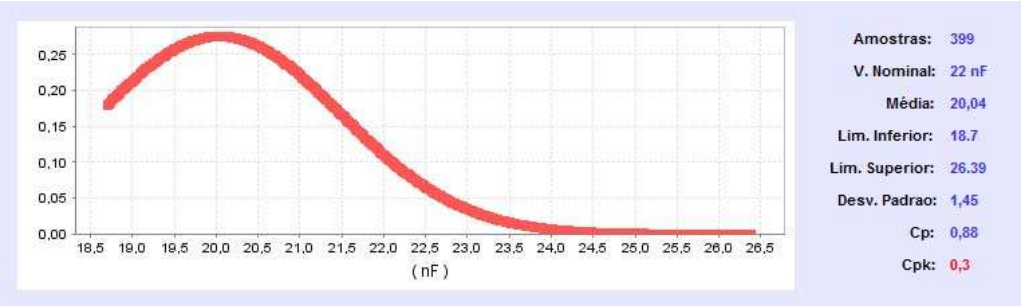

Figura 8.2: Distribuição normal dos valores medidos do componente C90

Ao clicar em `Detalhes`, na linha correspondente a este componente, é apresentada a distribuição normal dos seus valores medidos (Figura 8.2). A distribuição revela uma dispersão de valores elevada e uma média bastante descentrada em relação aos limites de tolerância especificados[1], levando a concluir que o seu valor medido é inconstante. A partir deste resultado podem ser conjecturadas pelo menos três hipóteses:

1. As peças poderão, de facto, apresentar estas diferenças nos valores medidos. No caso de taxas de rejeição mais elevadas, deverá ser desencadeada pelo Departamento de Qualidade uma inspecção ao produto em laboratório para confirmar se o problema reside no componente ou na placa electrónica e proceder à paragem da produção se assim achar necessário (embora seja um cenário possível, é bastante improvável que as peças apresentem valores tão variados e dispersos entre si);

2. A metodologia utilizada para a medição do valor do componente poderá ser deficiente, *i.e.*, este método poderá necessitar de uma optimização, sendo necessário para isso actuar no programa de teste. Este tipo de problema deverá ser inspeccionado pelo Departamento de Engenharia de *Software*;

3. O problema poderá estar relacionado com *hardware*, *e.g.*, as agulhas de contacto podem estar deficientes ou danificadas ou a interface do *adapter*[2] de teste com o dispositivo pode não estar nas melhores condições. É necessário solicitar uma intervenção por parte do Departamento de Manutenção.

Na linha de produção, foram acompanhados os testes ao produto em questão e constatou-se que a medição efectuada ao componente se revelava bastante instável e, por vezes, aproximava-se bastante dos limites de tolerância especificados. Para proceder a uma intervenção correctiva, interrompeu-se a linha de teste. Depois de algumas correcções, foi possível tornar a medição do componente estável, não se verificando qualquer outro problema. Foram inspeccionadas as medições sobre outros componentes durante a sequência normal do teste para despiste do mesmo tipo de problema. No final, a linha de teste foi novamente activada.

Para a avaliação dos resultados, os ficheiros gerados nos dias seguintes são mantidos. Ao fim de alguns dias e após terem sido testadas 110 placas electróni-

[1] Este componente já de si apresenta um valor nominal descentrado em relação aos seus limites de tolerância. Isto deve-se ao facto de, neste caso específico, a ausência deste componente resultar num valor medido muito próximo do seu limite inferior, tendo este sido aumentado para garantir uma melhor detecção.

[2] Um *adapter*, ou consola de teste, é um dispositivo passivo que serve de receptáculo das unidades a testar. Inclui uma cama de agulhas para o contacto de pontos específicos existentes nas placas electrónicas e uma interface que lhe permite ser integrado no dispositivo de teste. Cada consola de teste é desenhada e construída para um produto específico, sendo esta amovível para dar lugar a outra caso seja necessário.

cas do mesmo tipo, os resultados obtidos mostram que existe uma substancial evolução no desempenho do processo de teste deste componente (Figura 8.3).

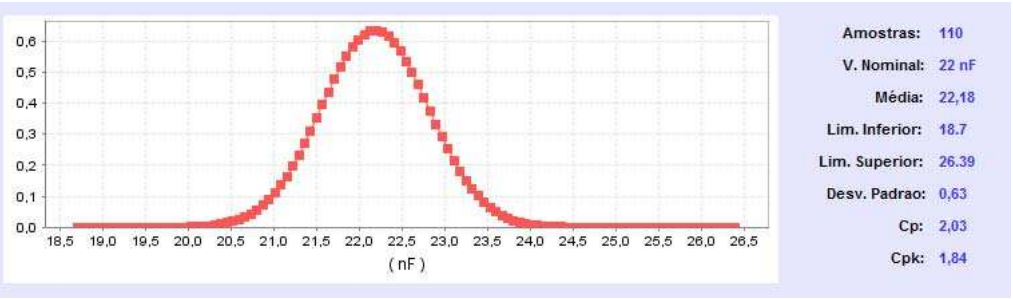

Figura 8.3: Valores medidos do componente C90 após medidas correctivas

Este caso prático ilustra a forma como esta ferramenta pode tornar mais simples e eficaz a detecção e a resolução de problemas na medição dos componentes dos produtos que estão em produção, contribuindo para a redução dos custos de produção.

8.2 Arranque de Produção de Novos Produtos

Outra situação em que esta nova ferramenta se revela bastante útil para o Departamento de *Software* e Testes é na fase de colocação em funcionamento de um novo *adapter* e respectivo programa de teste. Tradicionalmente, durante o desenvolvimento do programa e construção do *adapter*, não existem amostras suficientes do novo produto para um correcto aperfeiçoamento do processo de teste. O programa e o *adapter*, depois de concluídos, são assim verificados e testados utilizando as quatro ou cinco placas electrónicas disponíveis no momento e, após serem aprovados pelo Departamento de Qualidade de Processo, aguardam pelo arranque oficial de produção - *Start Of Production* (SOP).

Para mais rapidamente se determinar o desempenho destes novos meios de teste, aquando do arranque de produção, o programa de teste é configurado para gerar um ficheiro MDL por cada unidade testada, sendo estes convertidos e armazenados no servidor MySQL. Assim que exista um número significativo de testes efectuados (que são dependentes das quantidades requisitadas às linhas), os dados podem ser analisados, podendo então ser extraídas informações importantes:

1. Na página de Análise de falhas podem ser identificados os componentes que provocaram a rejeição das peças e a sua percentagem;

2. Na página da distribuição normal é possível confirmar: *(i)* se os valores limite de tolerância atribuídos estão correctamente especificados; *(ii)* se a estabilidade das medições através dos valores médio e desvio padrão e, por conseguinte, verificar o desempenho e a fiabilidade dos meios de teste.

Qualquer correcção ou optimização a implementar pode, assim, ser efectuada durante o período de arranque da produção em que as quantidades a produzir são mais baixas e o impacto da paragem das linhas de produção é mais reduzido.

8.3 Conclusão

A contribuição desta ferramenta na detecção e resolução de problemas nos componentes dos produtos em produção é inquestionável. Esta contribuição ocorre quer no momento crítico do arranque de produção de novos produtos, quer no normal decorrer da produção, permitindo diminuir os tempos de paragem das linhas de produção e, consequentemente, os custos de produção.

Capítulo 9

Conclusões

Este capítulo apresenta o balanço do trabalho realizado face aos objectivos propostos. São também identificadas algumas optimizações e melhorias a ter em conta no futuro.

9.1 Considerações Finais

No âmbito desta tese desenvolveu-se uma metodologia que permite detectar falhas e desencadear acções correctivas sobre o processo produtivo da empresa PREH Portugal. Esta metodologia levou à concepção de duas aplicações: *(i)* uma aplicação *stand-alone* de conversão, processamento e armazenamento dos dados produzidos durante o processo de teste ICT; e *(ii)* uma aplicação *web* que permite aos membros do Departamento de *Software* e Testes detectar de forma expedita os produtos com eventuais problemas e desencadear imediatamente acções correctivas.

O desenvolvimento destas duas aplicações permitiu atingir os objectivos principais deste trabalho: *(i)* foi criado um método para o armazenamento dos dados gerados em ficheiros de texto num servidor de bases de dados MySQL, organizando assim a informação que até então estava dispersa e permitindo uma consulta expedita; *(ii)* foi implementada uma aplicação que processa estes dados mostrando aos utilizadores com algum pormenor os resultados estatísticos do processo de teste ICT; *(iii)* foram utilizadas exclusivamente ferramentas de *software open source*; *(iv)* não foram realizados investimentos adicionais ou prejudicados outros projectos em curso; *(iv)* foi agilizado o processo de detecção de falhas e de desencadeamento de intervenções correctivas.

9.2 Desenvolvimentos Futuros

Existem algumas optimizações a considerar neste trabalho. Em primeiro lugar, sugere-se uma alteração da estrutura da base de dados. Neste caso específico dos dados do processo de teste, poderão existir milhões de entradas na tabela de componentes, o que requer bastante espaço em disco e que pode vir a tornar lenta a pesquisa dos componentes. No sentido de minimizar esta questão propõem-se as seguintes alterações à base de dados:

Alterações na base de dados:

- Tabela de componentes `ICT_MDL_VALUES`: O campo `COMPONENT_NAME` poderá ser substituído por um identificador que estará relacionado com os nomes de componentes numa outra tabela específica. (A maioria dos componentes tem o mesmo nome de produto para produto). Neste caso o espaço em disco ocupado pelos dados no servidor será menor;
- Tabela de produtos `ICT_MDL`: Os campos `TEST_PROGRAM` (programa de teste) e `BOARD` (nome do produto) poderão ser substituídos por identificadores. Estes identificadores estarão relacionados com os nomes dos produtos e os nomes dos programas de teste localizados noutras tabelas, reduzindo o espaço ocupado em disco;
- Utilizar uma transacção SQL para a inserção dos dados dos ficheiros de teste, tornando-se mais robusta e não permitir eventuais inconsistências por falha da ligação de dados.

Outras sugestões de melhoria incluem:

- A inserção automática de produtos novos a desenvolver na aplicação de conversão de ficheiros. No caso do produto a converter não existir na base de dados, será adicionado à tabela de produtos automaticamente;
- A automatização do processo de detecção de falhas e de desencadeamento das intervenções correctivas;
- A longo prazo será também possível recorrer a técnicas *data mining* sobre os dados armazenados com o intuito de detectar padrões que possam indicar oportunidades de melhoria do processo produtivo;
- A inclusão de um módulo de predição que permita prever problemas e desencadear operações preventivas.

Bibliografia

[1] Pedro Coelho: "Programação com JSP - Curso Completo", FCA Editora de Informática, 2005, ISBN 972-722-389-3.

[2] Benedita Malheiro. "RIAP - Redes Inteligentes e Aplicações" (2006). Disponível em http://ave.dee.isep.ipp.pt/~bene/RIAP.

[3] Adam Myatt. "Pro NetBeans IDE 5.5 Enterprise Edition", Apress, 2007, ISBN-13 978-1-59059788-0.

[4] Marty Hall, Larry Brown. "Core Servlets and JavaServer Pages", Sun Microsystems, 2000.

[5] Hans Bergsten. "JavaServerFaces", O'Reilly, 2004, ISBN 0-596-00539-3.

[6] Ryan Asleson, Nathaniel T. Schutta. "Foundations of Ajax", Apress, 2006, ISBN 1-59059-582-3.

[7] Steve Holzner. "Ajax for Dummies", Wiley Publishing, Inc., 2006, ISBN 0-471-78597-0.

[8] Kim Topley. "Java WebServices in a Nutshell", O'Reilly, 2003, ISBN 0-596-00399-4.

[9] Wikipedia."Wikipedia" (2008). Disponível em http://www.wikipedia.com.

[10] NetBeans. "Documentation, training and Support" (2008). Disponível em http://www.netbeans.org/kb/index.html.

[11] NetBeans. "Generating Reports and PDFs From a Web Application" (2007). Disponível em http://www.netbeans.org/kb/55/vwp-reports.html.

[12] Onur Derin. "A Tutorial on Reporting in Java" (2005). Disponível em http://sgsoft.itpub.net/get/28147/A Tutorial on JasperReports, iReport and JFreeChart.pdf.

[13] NetBeans. "Redirection When Session Times Out" (2007). Disponível em http://www.netbeans.org/kb/55/sessionredirect.html.

[14] Don Winton. "Process Capability Studies" (1999). Disponível em http://elsmar.com/pdf_files/CPK.pdf.

[15] MySQL. "MySQL 5.0 Reference Manual".
Disponível em http://dev.mysql.com/doc/refman/5.0/en/.

[16] Insider Scoop From the Tutorial Divas. "Table Component Sample Visual Web Project for NetBeans IDE 6.0" (2007). Disponível em http://blogs.sun.com/divas/entry/table_component_sample_visual_web.

[17] JavaServer Faces Tutorials.
Disponível em http://www.jsftutorials.net/.

[18] "Introduction to Profiling Java Applications in NetBeans IDE".
Disponível em http://www.netbeans.org/kb/60/java/profiler-intro.html.

[19] Copacetic - Blog archive "Web 1.0 vs Web 2.0". Disponível em http://joedrumgoole.com/blog/2006/05/29/web-20-vs-web-10/.