FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Sistema de Informação para Produção

Tiago José Gonçalves Cascão

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Mário Jorge Rodrigues de Sousa

Co-orientador: Eng. Nelson da Costa Silva

24 de Julho de 2014

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"Sistema de Informação para Produção"

foi aprovada em provas realizadas em 21-07-2014

o júri

Presidente Professor Doutor Paulo José Lopes Machado Portugal
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Luís Lino Ferreira
Professor Adjunto do Departamento de Engenharia Eletrotécnica da Instituto Superior de Engenharia do Porto

Professor Doutor Mário Jorge Rodrigues de Sousa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Tiago José Gonçalves Cascão

Faculdade de Engenharia da Universidade do Porto

Resumo

O presente trabalho foi realizado após proposta da empresa Nibble - Engenharia Lda. para o desenvolvimento de uma plataforma Web que permitisse o registo dos processos de produção de vários produtos. Dada a já existência de um sistema implementado na empresa, optou-se por centrar atenções na melhoria do mesmo, aproveitando as suas funcionalidades mais importantes.

Desta forma, seriam, não só, cumpridos os critérios de qualidade impostos, como também haveria uma melhoria do sistema de controlo e monitorização da produção da empresa.

As vantagens passam por isso pela obtenção de um processo de produção mais rápido e qualificado, ao qual acresce o aumento da qualidade do método de trabalho. Além disso, é possível haver uma maior rastreabilidade dos produtos, sendo obtidos dados mais exatos e precisos sobre a sua composição, o que torna mais eficiente o processo quer de produção quer de reparação.

Inicialmente, foi realizado um estudo do processo de produção da Nibble, tendo sido também analisado e documentado o sistema informático. Ainda antes da execução do trabalho, foi feito, conjuntamente com a empresa, o levantamento de requisitos a implementar.

Após estas etapas, procedeu-se à implementação das novas funcionalidades a adicionar ao sistema e das melhorias a que este foi sujeito, tendo, por fim, sido testadas a propósito de verificar a exequibilidade e a utilidade das mesmas.

Considero que este trabalho constituiu uma mais-valia quer para mim quer para a empresa, havendo, no entanto, possibilidade de, no futuro, o sistema já utilizado ser alvo de mais aperfeiçoamentos, detetados já durante o desenvolvimento deste projeto e descritos neste documento.

Abstract

This project was carried out after a proposal of the company "Nibble – Engineering Lda". For the development of a Web platform that allowed the registration of the production's processes of several products. There was already a system implanted in the company, so we choose to focus our attention in the improvement of this system, appropriating their most important features.

In this way, there would be, fulfilled, not only the criterion of quality imposed, but also would be an improvement of the control system and the motoring of the company's production.

The advantages are the obtainment of a faster and qualified production process, to which is added the increase of the quality of the work method. Moreover, it is possible to have a greater traceability of the products, being obtained more accurate and precise data on their composition, what makes the process of the production and of the repair more efficient.

Initially, it was executed a study of the process of Nibble's production, the computer system was analyzed and documented too. Before the execution of the project, it was done, together with the company, the survey of the requisites to implement.

After these steps, we proceeded to the implementation of new features to add to the system and its improvements, having finally been tested in order to verify the practicability and usefulness of the same.

I consider that this project is an added value both for me and for the company, having, however, the possibility, in the future, the system already used be targeted for more improvements, already detected during the development of this project and described in this document.

Agradecimentos

O desenvolvimento desta Tese foi, para mim, um grande desafio. Ao trabalho diário que tive, tenho de adicionar todo o apoio que me foi dado e que contribuiu para que, por fim, surgisse este projeto tal como o apresento. Assim, pela importância que de uma ou outra forma tiveram durante estes últimos meses, deixo aqui o meu agradecimento.

Ao Prof. Mário Sousa, pelos conselhos e apoio prestados na elaboração deste projeto;

Ao Engº Nelson Silva, pela disponibilidade e ajuda que me prestou em todos os momentos;

À empresa Nibble e a todos os seus colaboradores, pelo modo como me receberam e pelo bom ambiente criado ao longo destes meses;

Aos meus pais, por todas as condições que me proporcionaram e apoio incondicional dado ao longo deste percurso académico;

À minha restante família, pelo incentivo e ânimo que me transmitiram durante estes 5 anos;

À Mafalda, um especial agradecimento por todo o companheirismo e dedicação que teve, mesmo nos momentos menos bons e em que parecia impossível alcançar os objetivos. É com extrema alegria que partilho com ela mais este importante momento na minha vida;

Aos meus colegas e amigos, por todo o companheirismo e apoio;

A todos vós, um sentido obrigado!

Tiago Cascão

"If money is your hope for independence you will never have it.
The only real security that a man will have in this world
is a reserve of knowledge, experience, and ability."

Henry Ford

Conteúdo

Lista de Figuras

Abreviaturas e Símbolos

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
BD	Base de Dados
CSS	Cascading Style Sheets
FW	Firmware
GPS	Global Positioning System
GSM	Global System for Mobile
HW	Hardware
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
LED	Light Emitting Diode
MVC	Model-View-Controller
NASA	National Aeronautics and Space Administration
PCB	Printed Circuit Board
PDF	Portable Document Format
PHP	Hypertext Preprocessor
POP3	Post Office Protocol 3
SGBD	Sistemas de Gerenciamento de Base de Dados
SI	Sistema de Informação
SQL	Structured Query Language
SW	Software
WWW	World Wide Web
XML	eXtensible Markup Language
XLS	MS Excel file extension

Capítulo 1

Introdução

A elaboração deste trabalho foi proposta pela empresa Nibble – Engenharia Lda, no âmbito da disciplina Dissertação do 5º ano do Mestrado Integrado em Engenharia Eletrotécnica e Computadores da Faculdade de Engenharia da Universidade do Porto.

A Nibble – Engenharia Lda é uma empresa portuguesa sediada na Trofa, que está presente no mercado desde 2004 e é marca registada desde 2009, tendo, mais recentemente em 2013, obtido o registo de marca internacional. Deste modo, a Nibble projeta-se para um mercado mais amplo e no qual se pretende implementar e ganhar notoriedade, através dos projetos desenvolvidos pelos seus colaboradores.

Esta empresa, tem como principal objetivo o fornecimento de um conjunto de produtos e serviços de qualidade nas áreas da "Segurança e Domótica", "Gps e Comunicações Móveis" e "Iluminação LED e Eletrónica", sendo uma referência no mercado nacional. Esta empresa distingue-se, também, pela especialização na "Gestão e Desenvolvimento de Projetos de Engenharia Eletrónica", atuando desde a ideia inicial até ao produto final, passando pelo desenvolvimento e controlo de todo o processo de Gestão do Produto.

A "Consultadoria e Projeto"é uma grande aposta da empresa e constitui uma vantagem competitiva no mercado, adquirida pela já longa experiência no ramo, que já deu frutos de relevo quer nacional quer internacional. Esta área permite ao cliente conceber uma solução para uma ideia inicialmente pensada pelo mesmo e executada em conjunto com os colaboradores da Nibble.

1.1 Âmbito do Projeto

O sistema de produção da Nibble abrange dois tipos de processos, a saber o de fabrico e o de reparação. O primeiro contempla as etapas de testes aos componentes, para o cumprimento de critérios de qualidade, necessários à atividade de qualquer empresa, montagem dos mesmos e posterior confirmação da sua correta execução. Já o segundo, é importante essencialmente para a reparação de produtos com anomalias detetadas pelos clientes. Dado o relevo de ambos os processos e a impossibilidade de os dissociar, este projeto surge como um suporte bastante útil aos mesmos. Considera-se essencial ter um sistema de informação, seja manual ou automatizado, onde

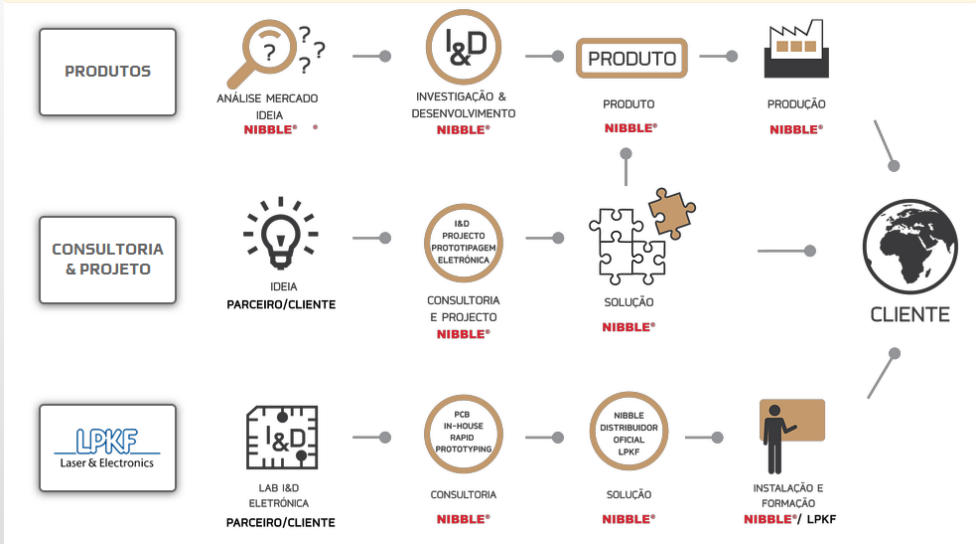

Figura 1.1: Produtos e Serviços Nibble

fique registado e haja um simples e eficaz acesso a todas as tarefas realizadas durante estas fases. Começando pelos testes a produtos, documentos enviados e recebidos dos clientes, abrangendo também produtos que estejam prontos a ser montados, componentes utilizados em cada produto e não esquecendo a capacidade de utilizar, gerir e distribuir dados, com a finalidade de melhorar o controlo, planeamento e análise dos processos da empresa. De referir ainda que, apesar de já existir um sistema de informação na empresa, a presença de algumas limitações constitui um estímulo e uma maior motivação para este trabalho, para que no final estas já estejam colmatadas e a empresa possa usufruir de um sistema de maior qualidade.

Neste caso em particular, o sistema consiste numa plataforma web que permite o registo dos processos de produção para vários produtos, nomeadamente uma Firewall (Figura 1.2 - sistema de deteção de incêndios), um comunicador GSM (Figura 1.3 - dispositivo de controlo e monitorização do que acontece à distância, por exemplo em casa ou no escritório) e ainda uma Sirene de Alarme Exterior (Figura 1.4 - responsável por gerar sinais de alerta sonoros contra incêndios ou intrusões).

Figura 1.2: Firewall

Figura 1.3: GSM

Figura 1.4: Sirene

Todos os componentes destes produtos, como os módulos de GSM, as PCBs, entre outros são submetidos a testes antes de ficarem aptos para montagem, de modo a que o produto final não tenha problemas quando for entregue aos distribuidores, que os irão por fim vender aos clientes

finais. No processo de montagem, os componentes são agregados de forma a constituir o produto final e nessa altura é fundamental ter um sistema que permita ter acesso ao estado atual do nível de produção (se está pronto para montagem ou se está em fase de testes), rastrear os componentes utilizados e gerir todo o seu histórico a nível de produção, transportes/encomendas e ainda de possíveis processos de reparações.

1.2 A Empresa

Este capítulo pretende explicar o modo de funcionamento dos dois sistemas utilizados pela empresa, a saber o sistema de produção e o sistema de informação para produção. O primeiro sistema engloba a elaboração física dos produtos e será explicado o modo como a empresa trabalha nas suas etapas, as quais incluem as suas reparações. Já o segundo sistema funciona como apoio à produção e permite informatizar todo o processo de produção, de forma a que quando for necessário reparar um produto ou obter alguma informação sobre o mesmo seja mais fácil e eficaz. Nessa parte do capítulo, serão explicadas as suas potencialidades antes da implementação deste projeto.

1.2.1 Sistema de Produção

No sistema de produção da empresa, podem ser identificados dois processos, o processo de fabrico, onde estão agregadas todas etapas de testes aos componentes vindos dos fornecedores, montagem e testes à etapa de montagem, e o processo de reparação dos produtos acabados, que passa pela recolha dos produtos entregues e a sua inclusão do sistema, testes às falhas referidas pelo cliente, etapa de reparação dessas falhas e entrega novamente aos clientes.

O processo de fabrico começa na entrega dos vários elementos, *printed circuit boards* (PCBs), fontes de alimentação, caixas exteriores, entre outros, por parte dos fornecedores para ser possível a construção dos produtos finais, isto é, da Firewall, do Comunicador GSM e da Sirene. De seguida, é efetuado um conjunto de testes unitários aos componentes mais importantes, como a fonte de alimentação, o módulo GSM e as PCBs, para os viabilizar para a zona de montagem. A estas operações tem de ser atribuído um número de serviço, que inclui o operador responsável (ex. ABC1234 - número de serviço 1234, em que o operador responsável foi o ABC).

A fase seguinte passa por acoplar e interligar os vários componentes para constituir o produto final. Concluída esta fase, é necessário simular o ambiente real e testar o produto, de forma a certificar que o produto funciona conforme o previsto. Quando estiver tudo testado é necessário colocar os produtos nas embalagens de comercialização juntamente com os componentes adicionais (como sejam os parafusos, os fusíveis e as resistências), os manuais e o certificado de garantia e por fim é realizada a selagem da embalagem.

Neste momento, o produto está pronto a receber um número de série que irá ser necessário para intervenções futuras no mesmo. Em seguida, é criado um documento de saída que contém informações sobre os produtos expedidos, bem como os respetivos números de série. Aqui, pode-se dar por terminado o processo de fabrico.

Para facilitar o entendimento do processo, este encontra-se representado através do seguinte diagrama de fluxo.

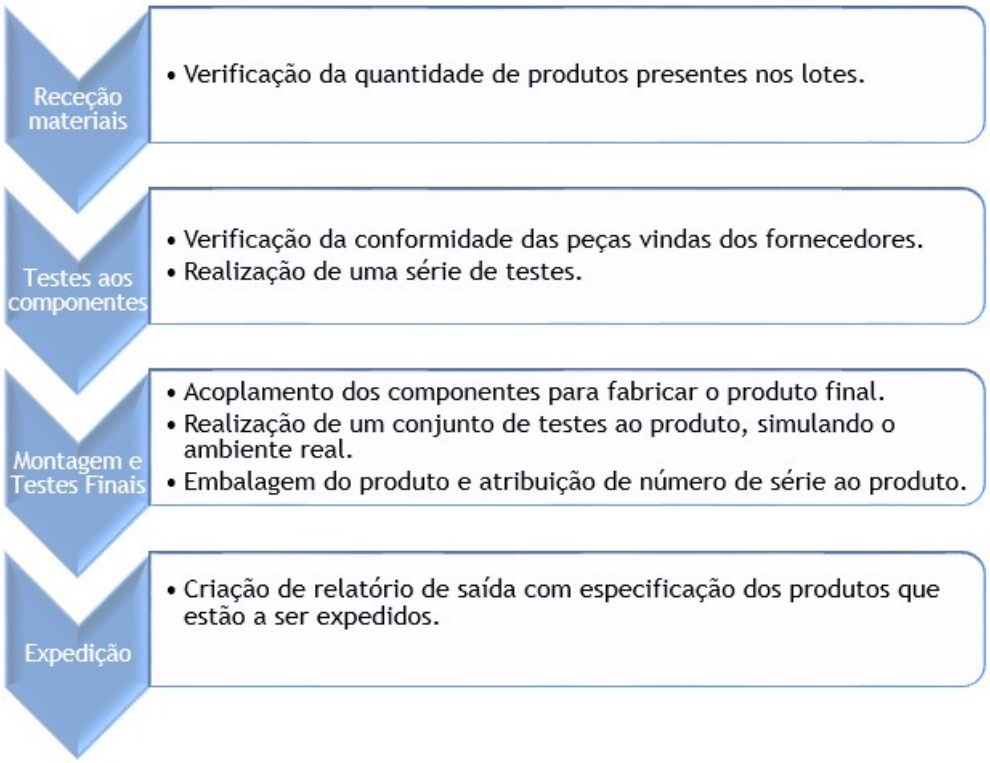

Figura 1.5: Fluxo do processo de fabrico

O outro processo é aquele que engloba a reparação dos produtos acabados, ou seja, se houver alguma anomalia com o produto, o cliente devolve o produto para reparação com a respetiva guia, onde é especificado o número de série do produto e o seu problema. Quando recebido, é introduzido no sistema e irá ser posteriormente reparado pelo técnico, associado a um novo número de serviço. Aqui, é confirmado que os números de série dos componentes correspondem aos iniciais e verificado se esse produto está coberto por garantia. Se tudo estiver dentro das conformidades, o produto é reparado, com novo número de serviço e o produto volta a ser expedido, junto de um documento de saída, para o cliente.

Assim como foi feito para o processo anterior, podemos apresentar o processo de reparação através do seguinte diagrama de fluxo.

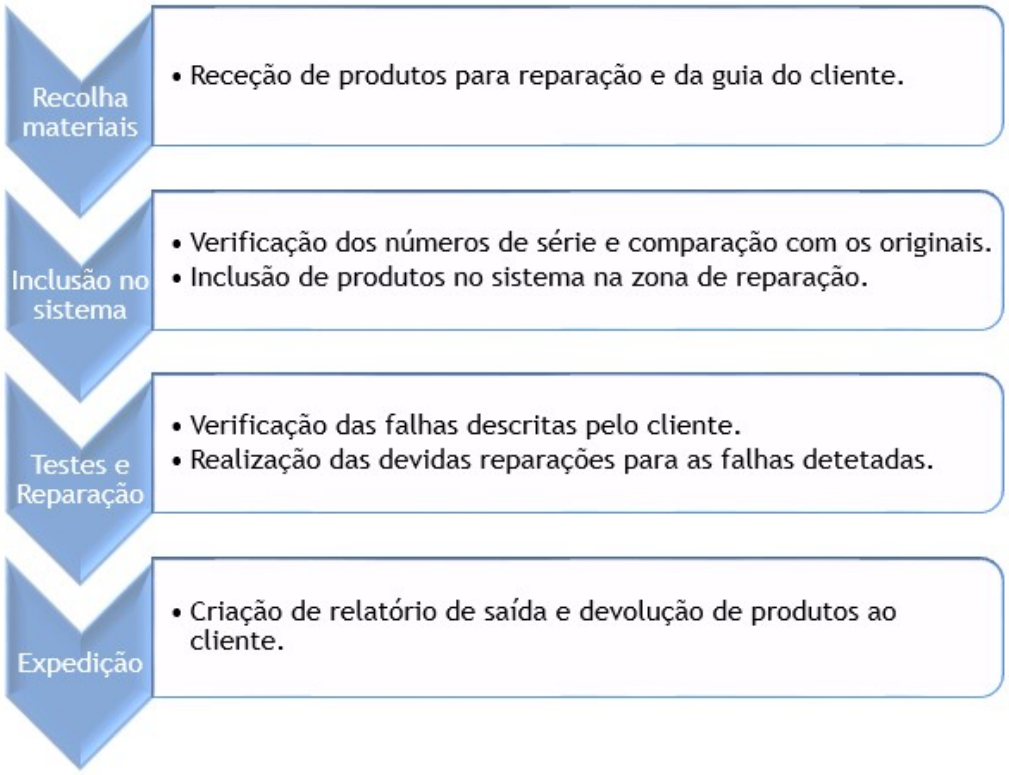

Figura 1.6: Fluxo do processo de reparação

1.2.2 Sistema de Informação para Produção

A plataforma existente é capaz de, para cada produto, apresentar o registo de diversas informações, como sejam os testes efetuados aos componentes principais (sendo, no entanto, apenas possível aceder a um componente por produto), o seu número de série, a data em que foi produzido, qual a versão de firmware e hardware que possui e ainda se a garantia continua válida. É possível também agrupar os produtos em listas, que englobam envios e/ou receções com números de guia e/ou fatura, moradas, datas e nomes relevantes. Todos os dados introduzidos são associados a um utilizador responsável, através de um número de serviço. Por seu turno, a mercadoria devolvida para reparação é inserida manualmente no sistema aquando da receção dos produtos. Este sistema permite ainda fazer pesquisas através do número de série do produto, obtendo assim acesso aos testes, reparações e relatórios feitos ao produto. Assim sendo, pode-se indicar como principais formas de utilização do sistema a realização de testes aos produtos, a montagem dos mesmos e a emissão e receção de guias de material.

No entanto, este sistema apresenta algumas limitações. Com a expansão da empresa e o aumento da quantidade de produtos presentes na base de dados, o sistema demora mais tempo a responder, nomeadamente nos processos de testes aos produtos. Diminui, deste modo, a sua qualidade e usabilidade, pois pretende-se uma resposta rápida do sistema, para que este seja um complemento importante para a empresa e não uma fonte de atrasos e perdas de tempo no que diz

respeito à sua produtividade.

1.2.2.1 Base de Dados

A Base de Dados (BD) do sistema é constituída por 21 tabelas, a salientar a tabela “Product” (onde são guardados os produtos existentes), “Test Category” (onde estão incluídos todos os testes a fazer a produtos), “Test Result” (onde constam os resultados dos testes efetuados) e ainda o “Repair Report” e “Test Report Entry” (onde estão presentes os relatórios de teste e de reparação dos produtos, que se pretende obter quando se realiza a consulta do histórico do produto na empresa) e ainda por 31 vistas sobre tabelas, que servem para gerar tabelas virtuais com junções de outras tabelas, de modo a que a informação a utilizar nas diferentes solicitações ao servidor esteja toda na mesma forma de apresentação.

Estas tabelas e vistas podem ser visualizadas e manipuladas através do “mysqlWorkbench”, que se trata de uma ferramenta que permite gerir todas as tabelas e vistas de uma forma mais intuitiva e fácil, como pode ser visto na figura seguinte.

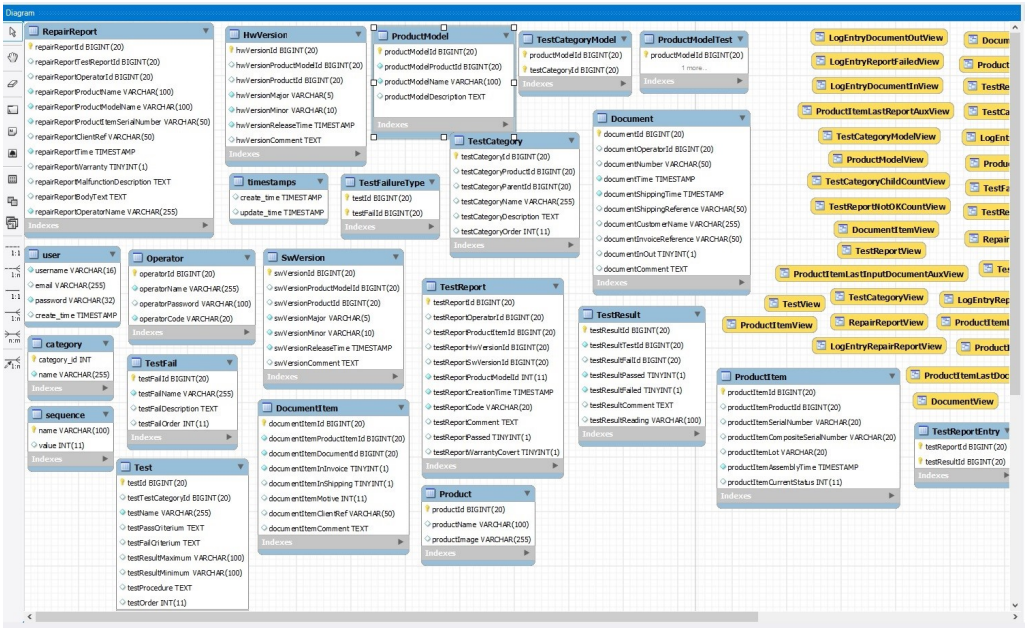

Figura 1.7: Visão sobre as tabelas presentes no sistema

1.2.2.2 Interface

Quanto à interface do sistema, no separador "Produto"podemos selecionar o produto em que queremos trabalhar (Firewall, Comunicador GSM ou Sirene), como pode ser visto na figura 1.8

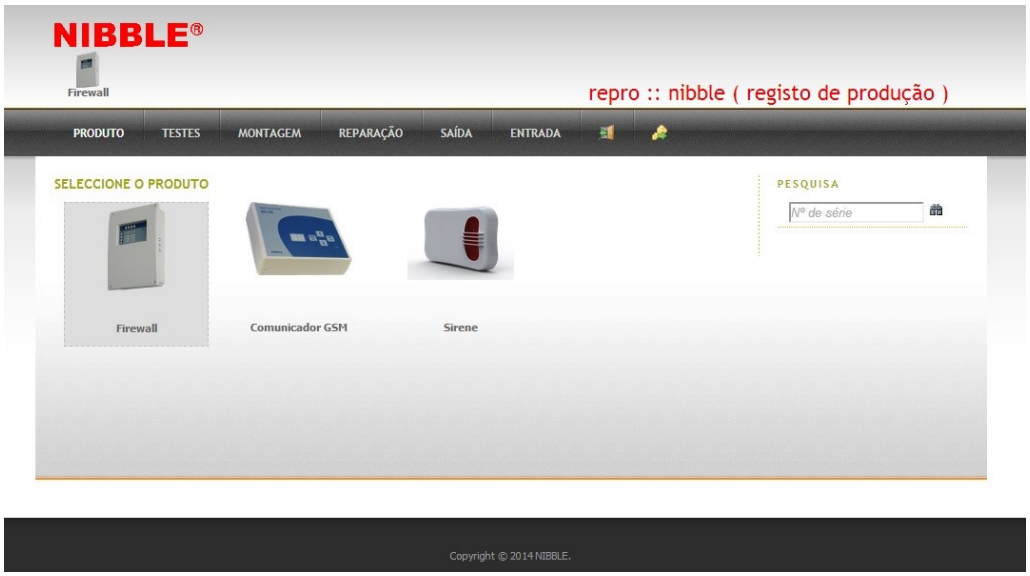

Figura 1.8: Interface para a escolha do produto

Depois de selecionado o produto, podemos aceder à página de testes, à lista de produtos a reparar, à lista de componentes prontos para a montagem e ainda dar entrada e saída de materiais.

Figura 1.9: Exemplo de uma página de testes

Na figura 1.9, podemos ver um exemplo de uma página de testes para uma Firewall. Depois do técnico preencher todos os campos do relatório, submete o botão “Guardar” e este é armazenado.

Figura 1.10: Exemplo da página de montagem

A figura 1.10 é ilustrativa da página de montagem, onde aparecem os vários produtos que estão prontos para serem submetidos a montagem e os que já o foram, com os números de série dos componentes (que surgem ocultados, pois são dados confidenciais da empresa), o lote, data de montagem e ainda um número de serviço, que, como foi abordado anteriormente, é obrigatório.

Figura 1.11: Exemplo da página de entrada de produtos para reparação

Na figura 1.11, é possível ver um exemplo da entrada de produtos para reparação, efetuada por parte do cliente. Este processo é acompanhado do nome do cliente, número de guia, o motivo pelo qual o material está a ser enviado, qual o seu número de série e ainda se está incluído no prazo da garantia (“G”).

1.3 Estrutura da Dissertação

Para além da Introdução, na qual são apresentados os sistemas com os quais tive oportunidade de lidar durante o desenvolvimento deste projeto e nos quais está incluído tanto o sistema de produção da empresa, como o sistema de informação para produção previamente implementado na Nibble – Engenharia Lda, esta dissertação contém ainda mais seis capítulos.

No capítulo 2, é abordado o padrão de desenvolvimento a aplicar na reformulação da arquitetura do Sistema de Informação da Empresa e são sucintamente explicadas as tecnologias e linguagens de programação usadas na realização deste projeto e que irão permitir uma melhor compreensão do restante trabalho.

No 3º capítulo, está presente o levantamento de requisitos que foi realizado em conjunto com a empresa, com o intuito de diminuir as limitações do sistema e assim melhorar a sua performance. A cada requisito está associada uma breve descrição, os testes necessários para a sua validação e ainda o modo em que cada um poderá contribuir para o aperfeiçoamento do sistema e consequente enriquecimento do trabalho desenvolvido pela empresa.

No capítulo 4, está desenvolvida a arquitetura do Sistema de Informação, que inicialmente necessitou de um estudo aprofundado, posteriormente completado com a sua documentação. Desta forma, foi possível colmatar a falta de informação pré-existente acerca da arquitetura do sistema. Dentro deste capítulo, primeiramente será apresentada uma arquitetura de alto nível e em seguida o modelo *Model-View-Controller* (MVC), no qual está incluído o modelo entidade-relação.

No capítulo 5, é apresentado o modo de desenvolvimento e de implementação do trabalho realizado no sistema, através da explicação do processo baseada em diagramas demonstrativos das sequências de execução dos *scripts*.

No capítulo 6, são apresentados os testes realizados para a validação dos requisitos implementados. Nele estão incluídos *printscreens* dos mesmos, acompanhados da explicação da utilização dos dados em questão e do modo em que os testes são considerados válidos.

No último capítulo, estão presentes as conclusões, bem como ideias para o trabalho a desenvolver no futuro, de forma a aumentar, ainda mais, a qualidade do sistema.

Capítulo 2

Estado da Arte

Para a elaboração desta melhoria no Sistema de Informação (SI) em questão, foi necessário implementar algumas mudanças ao nível da arquitetura de desenvolvimento do mesmo, sendo que o padrão de arquitetura que mais se aproxima do utilizado até então é o *Model-View-Controller* (MVC). Assim irá ser feita, em seguida, uma breve abordagem para um melhor entendimento das suas caraterísticas e funcionamento.

Foi também fundamental realizar um estudo sobre as tecnologias que estavam a ser utilizadas, pois apesar de algumas delas serem conhecidas, a sua prática não tinha sido muito explorada, pelo que foi necessário aprofundar o conhecimento do seu funcionamento. Assim sendo, serão feitas breves abordagens às diferentes tecnologias utilizadas, o que irá facilitar o alcance do objetivo do trabalho.

2.1 Padrão de Arquitetura MVC

O MVC é um padrão de arquitetura muito utilizado em inúmeras aplicações, das quais faz parte a criação de aplicações Web e de várias *frameworks* para o desenvolvimento web, em PHP ou outras linguagens. Este padrão de arquitetura separa o código fonte das aplicações em três grandes grupos, definindo as interações entre elas:

Modelo

Contém não só todo o código que permite o acesso à base de dados, como também os dados presentes na mesma. No Modelo, mantém-se a máxima abstração da complexidade da base de dados e criam-se, apenas, funções para receber, inserir, atualizar ou apagar informação das tabelas.

Os fatores a ter em conta no Modelo serão o tipo de BD utilizado e as tabelas que devem ser criadas, bem como as suas relações.

As funções de chamadas à base de dados devem ser mantidas no mesmo ficheiro, fato que irá permitir invocá-las em qualquer parte do código. De seguida, o Modelo fica encarregue de as processar, de acordo com um conjunto de regras que contém.

Visão

Responsável por codificar e manter a apresentação final da aplicação vista pelo utilizador. Isto é, na Visão deve estar incluído todo o código HTML, CSS, JS e outros que têm como função gerar a página idealizada pelo utilizador. Na prática, a Visão não se restringe à criação de páginas web, uma vez que também é possível criar outra saída para enviar ao utilizador em formatos ou linguagens diferentes, como sejam o JSON ou XML. Aqui é também incluído o Browser que permite ao utilizador ver o que foi produzido pela Visão e ainda a interação com a aplicação.

Controlador

Este pode ser visto como a parte mais importante da aplicação, dado que funciona como ligação entre o Modelo, a Visão e qualquer outro recurso a ser processado pelo servidor, a fim de gerar a página web.

Como foi dito, uma das muitas aplicações onde se utiliza o MVC é o desenvolvimento de aplicações para Web. Neste caso, a Visão normalmente corresponde a páginas HTML, o Controlador contém o código que valida os inputs do utilizador e que gera os dados dinâmicos para incluir no HTML e o Modelo inclui o conteúdo geralmente armazenado em bases de dados ou arquivos XML.

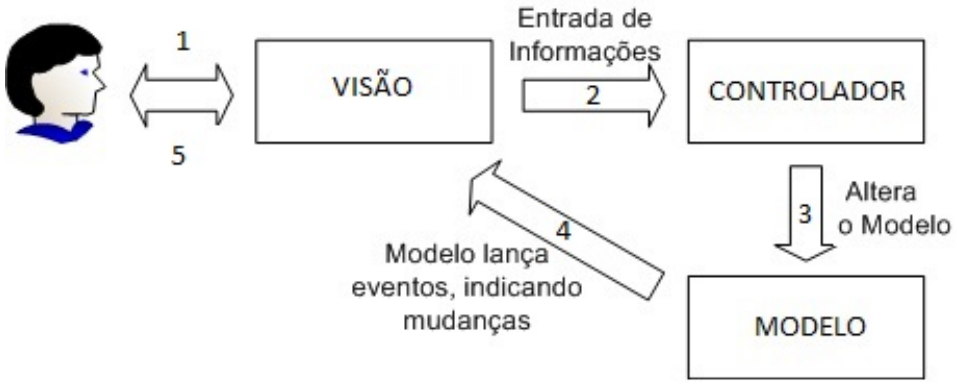

Figura 2.1: Fluxo MVC

O fluxo MVC pode ser resumido a:

1. O utilizador interage com o *Browser*, como por exemplo através do preenchimento de um formulário;

2. O Controlador manipula o evento da interface do utilizador, através de uma rotina pré-definida;

3. O Controlador comunica com o Modelo atualizando-o, de acordo com a informação inserida pelo utilizador, como seja a atualização dos dados de um dado produto;

4. A Visão obtém os dados necessários do Modelo e gera uma nova Visão, não tendo o Modelo um conhecimento direto do que se passa na Visão;

5. A Visão espera por próximas interações, o que irá iniciar um novo ciclo.

Estudo das Tecnologias Utilizadas

2.2 MySQL

O MySQL é um sistema de gestão de bases de dados (SGBD) relacionais, um dos mais populares a nível mundial, sendo utilizado por grupos como a NASA, a Nokia, entre outros. Isto significa que pode conter grandes quantidades de dados em tabelas separadas e não numa só tabela. Este sistema dá uma maior flexibilidade e rapidez ao sistema e tem como grande vantagem o fato de ser muito veloz e de fácil utilização, dada a sua arquitetura. Este sistema é o único SGBD que faz uma separação entre o servidor base e o motor de busca, o que permite um acesso extremamente rápido ou muito controlado ao disco, conforme a situação.

Um SGBD permite criar e gerir bases de dados, seja apenas uma simples BD ou uma BD relacional, que se podem definir como:

Base de dados : coleção de dados estruturados numa única tabela. Pode ser simplesmente uma lista de compras;

Base de dados relacional : armazena dados em tabelas separadas e não apenas numa tabela. Este tipo de BD aumenta a velocidade e a flexibilidade do sistema.

Uma base de dados é constituída por entidades (grupo de elementos, como por exemplo “Alunos”, sobre os quais é necessário guardar informação), atributos (elementos ou propriedades que permitem caracterizar uma entidade, como por exemplo “Disciplinas”) e ainda relações (utilizadas para relacionar as entidades, correspondendo normalmente a um verbo, como por exemplo “Inscrito”).

Os tipos de relação possíveis neste tipo de modelo são:

- ***Relação 1..1 (lê-se "relação um para um")*** - indica que as tabelas têm uma relação inequívoca entre si, isto é, as chaves primárias das entidades envolvidas são iguais (mesmo nome e tipo). É possível escolher a tabela que vai receber a chave estrangeira;

- ***Relação 1..n (lê-se "um para muitos")*** - a chave primária da tabela que tem o lado 1 vai para a tabela do lado n. No lado n, esta é chamada de chave estrangeira;

- ***Relação n..n (lê-se "muitos para muitos")*** - quando as tabelas têm entre si uma relação “n..n”, é necessário criar uma nova tabela com as chaves primárias das tabelas envolvidas, ficando assim uma chave composta, isto é, formada por diversos campos-chave de outras tabelas.

Um servidor como o MySQL disponibiliza uma interface, para que seus clientes possam adicionar, alterar ou consultar dados previamente armazenados. Nas bases de dados relacionais, a interface é constituída pelas APIs ou drivers do SGBD, que executam comandos na linguagem Structured Query Language (SQL).

É possível também a junção de informação de várias tabelas através das vistas (que funcionam como tabelas virtuais, não armazenando, portanto, fisicamente os dados). Para gerir todo este processo, a solução utilizada neste sistema foi o "mysqlWorkbench", uma ferramenta que permite gerir todas as tabelas e vistas de forma mais intuitiva e fácil e também ter uma visão mais organizada e detalhada da arquitetura da BD. A linguagem de programação utilizada para a criação das vistas (que não é mais que uma pesquisa na BD) é também o SQL. Esta linguagem destaca-se pela sua simplicidade e facilidade de uso, uma vez que é de acessível aprendizagem e a sua finalidade é selecionar a informação desejada para mostrar ao utilizador do sistema.

Figura 2.2: Logotipo MySQL

2.3 SQL

O SQL é uma linguagem de consulta a BD, em que não se especifica como, nem em que ordem são executados os processos que irão fornecer os resultados pretendidos. O SGBD é responsável por escolher o melhor procedimento a executar para que os resultados sejam obtidos o mais eficientemente possível. Trata-se de uma linguagem de interrogação de BD relacionais que permite a um gestor a realização de tarefas cruciais em base de dados como criar/alterar/eliminar tabelas e ainda inserir/consultar/atualizar/apagar os dados presentes nas mesmas. Esta linguagem é constituída por palavras-chave, que permitem a execução das tarefas referidas. Primeiramente, para criação/alteração/eliminação de tabelas da base de dados temos comandos como:

- ***Create Table***, cria uma tabela;
- ***Alter Table***, altera uma tabela;
- ***Drop Table***, elimina uma tabela.

Em seguida, as principais tarefas de manipulação de dados nas tabelas referidas atrás podem ser divididas em:

- Tarefas de atualização de dados, que podem ser executadas através de comandos como:
  - ***Insert***, adiciona dados a uma tabela;

  - ***Update***, altera dados de uma tabela ;
  - ***Delete***, elimina dados de uma tabela.

- Tarefas de consulta que são realizadas como o comando *"Select"*, que tem como função recolher a informação que cumpre todos os critérios da interrogação;

As tarefas de interrogação do tipo *"Select"*, as quais têm o nome de *"Query"* seguem uma sintaxe que é assente em seis clausulas, embora apenas duas sejam obrigatórias, o *"Select"* e o *"From"*. Qualquer uma das outras acrescenta critérios a cumprir.

A sintaxe de uma *query* resume-se a:

```
SELECT [ALL/DISTINCT] <COLUMN EXPRESSIONS>
FROM <TABLES NAMES AND JOIN SPECIFICATIONS>
WHERE <CONDITIONS>
GROUP BY <GROUPING EXPRESSIONS>
HAVING <GROUP SELECTION CONDITIONS>
ORDER BY <SORTING EXPRESSIONS>
```

Figura 2.3: Sintaxe de uma *query* SQL

onde,

- ***Select***, indica o nome das colunas a selecionar para a tabela resultado;
- ***From***, indica o nome das tabelas de onde se extrai as informações ;
- ***Where***, indicam-se condições de seleção e são eliminadas todas as linhas da tabela obtida do *"From"* que não cumpram as condições ;
- ***Group By***, agrupa os resultados baseando-se em determinadas colunas a fim de todas as linhas desse grupo tenham o mesmo valor para as colunas que são passadas como argumento;
- ***Having***, funciona da mesma forma que a condição *"Where"* e tem como função filtrar a informação obtida com o *"Group By"*.

Outra das funcionalidades do SQL que será muito útil neste sistema é o fato de permitir construir vistas, *"Views"*. Tratam-se de tabelas virtuais que obtêm direta ou indiretamente informações de tabelas base, através de *"queries"* que são expressadas através do comando *"Create View"*. Este tipo de tabela pode ser atualizável se for apenas uma visão sobre uma tabela base, caso contrário a tabela está apenas disponível para leitura.

2.4 HTTP

O HyperText Transfer Protocol (HTTP) é um protocolo de comunicação utilizado para sistemas de informação. Este é a base da comunicação de dados da WWW e é responsável pelo tratamento de pedidos e respostas entre cliente e servidor.

O HTTP surgiu pela necessidade de enviar informação, quer texto, quer imagens, pela internet e, para que essa distribuição fosse possível, foi necessário criar um padrão de comunicação. Assim sendo, passou a ser utilizado para a comunicação entre computadores na internet e para especificar o modo de realização das transações entre clientes e servidores.

Esta comunicação ocorre em duas etapas, sendo a primeira aquela em que o cliente envia o pedido (HTTP *Request*) e a segunda a que o servidor processa o pedido e envia uma resposta (HTTP *Response*). O HTTP *Request* é enviado pelo cliente e inclui:

1. Uma linha que especifica o tipo de documento, o método aplicado e a versão;
2. Os campos do cabeçalho, que são facultativos, mas que permitem ao servidor saber qual o *browser* ou o sistema operativo a utilizar.
3. O corpo do pedido que também é opcional, sendo nele inseridos os dados submetidos pelo cliente.

O HTTP *Response* é enviado pelo servidor ao cliente e terá de conter:

1. O estado no qual deverá estar indicada a versão do protocolo utilizada;
2. O estado do pedido;
3. O significado do estado, podendo estar apresentado sob a forma de texto, como por exemplo “OK” ou “NOT OK”;
4. Os campos do cabeçalho para o cliente ser avisado de informações extras que não estão incluídas no estado;
5. O corpo da resposta que representa o ficheiro que o cliente pediu.

2.4.1 Cookies

O *cookie* é um grupo de dados que o servidor troca com o *browser* e que é colocado num ficheiro de texto criado no computador do utilizador. A informação que fica contida no *cookie* pode ser encriptada para garantir a sua privacidade e segurança.

O *cookie* é enviado pelo servidor Web para um *browser* como um cabeçalho HTTP. Este cabeçalho é reenviado pelo cliente sem alterações de cada vez que acede ao servidor, sendo assim possível a sua utilização para autenticação em alguns sites, sem que seja preciso repetir a palavra-passe, registar preferências, como por exemplo a cor de fundo da página do cliente, entre outras possibilidades que possam estar armazenadas num ficheiro de texto.

Os *Cookies* não se tratam de ficheiros executáveis, daí não constituírem perigo para o cliente, como os vírus. No entanto, os processos utilizados pelo *browser* para manipulá-los podem ser usados como *spyware*.

A maioria dos *browsers* modernos permite que os utilizadores decidam se pretendem aceitar cookies, bem como o prazo para mantê-los, mas rejeitar *cookies* torna alguns sites inutilizáveis.

2.5 PHP

O Hypertext Preprocessor (PHP) é uma linguagem de *scripts* que é utilizada principalmente para desenvolver páginas Web dinâmicas, podendo ser incorporada no HyperText Markup Language (HTML).

Esta é uma linguagem interpretada pelo que não precisa de ser compilada, necessitando apenas de um interpretador para a sua execução.

A grande diferença dos *scripts* em PHP e programas em C, ou outra linguagem do género, é que não é necessário escrever comandos que disponibilizem o HTML, uma vez que o PHP é inserido no código HTML entre *tags* que permitem ao interpretador saber, facilmente, que aquele código é PHP (<?php ... ?>).

O PHP permite recolher dados de formulários HTML que foram inseridos pelo utilizador, gerar conteúdo dinâmico das páginas, enviar ou receber *cookies*, efetuar estatísticas, gerar gráficos ou ficheiros PDF e ainda animações. Outro ponto forte do PHP consiste na possibilidade de efetuar ligações à base de dados que irá permitir, consoante inputs dos utilizadores, obter respostas imediatas em formas de páginas HTML.

Esta linguagem é executada, predominantemente, no lado do servidor. Isto significa que o cliente recebe os resultados da execução do *script* sem aceder ao código fonte, ao contrário do que acontece no JavaScript (JS).

Resumidamente, o dinamismo que é pretendido na criação destas páginas é obtido através de um cliente Web (*browser*), servidor Web (ex. Apache), SGBD (ex. MySQL) e uma ferramenta que é capaz de fazer interagir estes servidores (ex. PHP).

O processo, para que o *script* armazenado no servidor Web consiga aceder e manusear os dados contidos na BD, resume-se a:

1. O utilizador introduz os dados no formulário HTML do *browser*;
2. Os dados são enviados para o servidor;
3. O servidor verifica as ligações à base de dados e confirma a ligação;
4. Os dados são processados no servidor de acordo com as instruções do *script*;
5. O servidor envia a página HTML resultante da execução do PHP ao *browser*.

Esta tecnologia está a ser cada vez mais utilizada, pois é muito prática e flexível, funcionando ainda em praticamente todos os servidores web, bem como em sistemas operativos, sendo todos

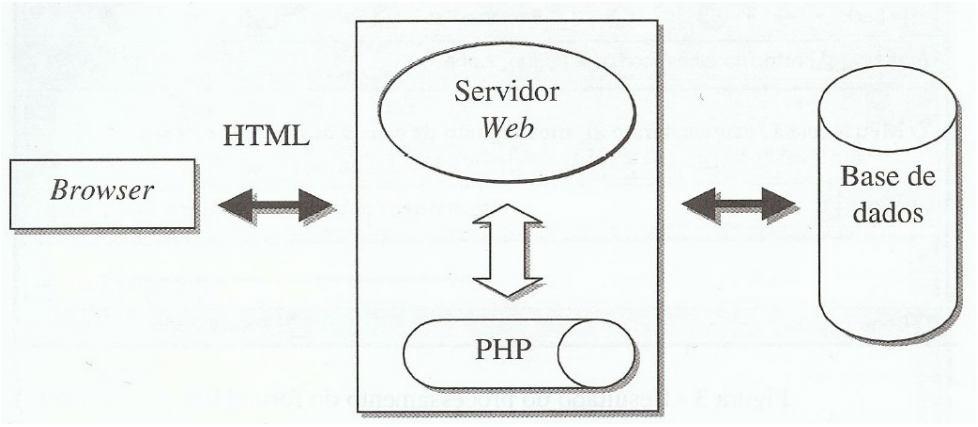

Figura 2.4: Funcionamento PHP

estes fatores que agradam a grande parte dos programadores web na criação de páginas dinâmicas. Outra caraterística fundamental é a sua capacidade de suportar uma grande variedade de bases de dados, como MYSQL e Oracle e ainda de permitir a utilização de protocolos como HTTP, POP3, entre outros.

Figura 2.5: Logotipo PHP

2.6 HTML

O HTML é uma linguagem de marcação baseada em marcas (*'tags'*), através das quais é formatado o conteúdo das páginas.

Inicialmente, o HTML tinha regras de sintaxe mais flexíveis, o que facilitava a publicação de conteúdos na Web. Hoje em dia essa sintaxe é muito mais rígida, o que leva a um código mais preciso e mais complexo. Por isso mesmo, ao longo do tempo, a utilização de ferramentas para criação de páginas de HTML aumentou, assim como a tendência em tornar a sintaxe cada vez mais rígida para diminuir as faltas de compatibilidades nos *browsers* e tornar o código cada vez mais padronizado, sem que a facilidade de criação de páginas Web fosse afetada.

Os documentos em HTML são ficheiros de texto simples que podem ser criados e editados em qualquer editor de texto comum, como o bloco de notas. No entanto, para ajudar na criação deste tipo de páginas, existem no mercado editores de texto orientados para a sua construção que, com recursos sofisticados, facilitam a repetição de tarefas, inserção de objetos, elaboração de tabelas e outros recursos.

Todos os ficheiros HTML são compostos por um cabeçalho (*header*) e por um corpo (*body*). Neste tipo de linguagem cada *tag* corresponde a um comando e está contida entre os caracteres "<" e ">". Qualquer outro caractere que não se encontre entre "<" e ">" é tratado como texto e inserido diretamente na página.

Os comandos ficam ativos desde que são abertos até serem fechados através do caractere "/".

O HTML dispõe dos comandos de formatação de texto, tal como os que se encontram nos processadores de texto, como por exemplo:

- . . . - introduz um parágrafo;
-
 - introduz uma quebra de linha;
- ... - itálico;
- ... - negrito;
- <u> ... </u> - sublinhado.

As funcionalidades do HTML são muito diversificadas e outra das competências é introduzir, dentro de um documento, ligações (*links*) para outros documentos. Estas são introduzidas através da marca <a> ... .

Como elementos fundamentais do HTML devem-se apontar as tabelas e as divisões que podem ser utilizadas com finalidades distintas, isto é, as divisões servem para ajudar a definir a estrutura da página Web, definindo o layout dos vários elementos, como textos e figuras. As tabelas também podem ter esta finalidade e ainda a de apresentar os dados na forma tabular, o que facilita a compreensão dos mesmos por parte dos utilizadores da aplicação.

Figura 2.6: Logotipo HTML

2.6.1 CSS

O CSS (Cascading Style Sheets) foi o nome atribuído às folhas de estilo. Estas definem um conjunto de regras que determinam a forma em que o *browser* deve apresentar os documentos HTML.

Através das CSS, é possível separar o conteúdo do documento (HTML) e o estilo da apresentação (CSS).

As regras de estilo são formadas por um elemento (por exemplo um elemento HTML como *body*, p ou table) e um estilo a ser aplicado a esse mesmo elemento.

Através do CSS, é possível definir um grande conjunto de propriedades, como por exemplo o tipo e/ou tamanho de letra a utilizar quando a *tag* do elemento for invocada (font-family, font-size, font-weight), as cores do texto e do respetivo fundo e possíveis imagens a utilizar como fundo (color, background-color, background-image). É igualmente possível definir o alinhamento do texto (text-align), as margens e as bordas para tabelas (margin, border), além de uma imensa quantidade de propriedades que o CSS nos permite utilizar. As folhas de estilos, normalmente, são definidas em ficheiros externos aos ficheiros HTML, o que permite utilizar as mesmas definições de estilo em diferentes páginas. Desta forma, é possível mudar completamente o estilo de uma aplicação, que poderá ser constituída por dezenas ou centenas de páginas, modificando apenas um único ficheiro. As folhas de estilo também podem estar contidas nos cabeçalhos dos documentos HTML, através do elemento *"STYLE"*, ou até mesmo incluídas no interior de qualquer elemento do corpo *body*.

Figura 2.7: Logotipo Css

2.7 JavaScript

O JavaScript (JS) é uma linguagem de programação utilizada para o desenvolvimento de *scripts* do lado do cliente (*client-side*) de aplicações Web, o que permite que haja mais interatividade com o utilizador.

A execução do código JS juntamente com a página HTML é efetuada pelo *browser*. O JS é frequentemente utilizado para validar os dados introduzidos por utilizadores num formulário, ou seja, os dados inseridos são analisados, sendo posteriormente possível concluir se cumprem todas as condições do campo que está a ser preenchido. No caso de os dados não cumprirem as condições, o preenchimento do formulário não prosseguirá até que os erros sejam corretamente corrigidos.

Esta funcionalidade é importante, principalmente quando os servidores estão disponíveis para muitas páginas por minuto, uma vez que deixam de precisar de validar os dados do formulário e reenviar a página com uma notificação de erro ao utilizador.

Além da validação dos dados de formulários, o JS permite detetar eventos, ou seja, quando o utilizador carrega num determinado elemento, como seja um botão ou uma opção de uma lista, ao qual estão associadas ações como "onclick", "onchange", entre outras. O JS permite ainda atuar sobre os valores e/ou atributos dos elementos HTML da páginas Web, expandir divisões ocultas ou compactadas, carregar documentos, entre muitas outras possibilidades.

Esta linguagem é baseada em objetos, embora não seja orientada a objetos, dado que permite declarar objetos com as suas próprias propriedades e métodos, mas não podem ser utilizados mecanismos de herança e especialização que são tipicamente utilizados em linguagens orientadas a objetos. As propriedades dos objetos podem ser obtidas através do nome, de um número de ordem (índice) ou de uma notação mista, por exemplo (item['nome']).

Este sistema em particular é usado, por exemplo, para ampliar a ficha de testes, selecionar produtos e expandir as suas informações, sem recarregar todo o conteúdo da página, o que melhora significativamente a performance e o tempo de resposta do sistema.

Figura 2.8: Logotipo JavaScript

2.7.1 Ajax

O Ajax (Asynchronous Javascript And XML) é uma técnica de desenvolvimento que possibilita que as páginas Web sejam atualizadas de forma assíncrona com um pequeno número de trocas de dados com o servidor. Isto significa que é possível atualizar partes de uma página Web, sem ter que recarregar a página inteira, ao contrário das páginas Web que não usam esta tecnologia.

Quando ocorre um evento, é criado um objeto XMLHttpRequest, que se trata do componente fundamental do Ajax e que é enviado para o servidor. O servidor analisa o pedido, processa e cria a resposta, enviando-a para o *browser*. A informação recebida pode ser processada em background e usada para, dinamicamente, atualizar elementos numa página Web, sem necessidade de recarregar toda a página. O funcionamento desta ferramenta está ilustrado na figura abaixo apresentada.

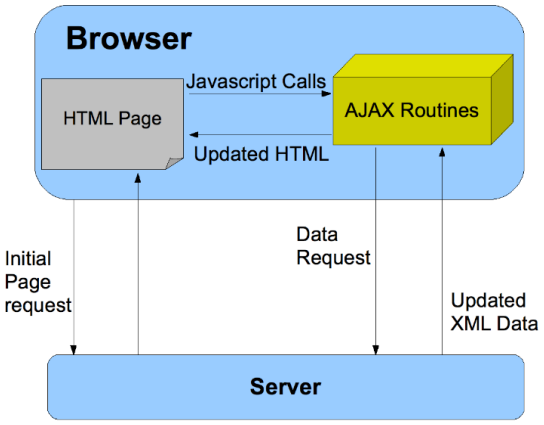

Figura 2.9: Funcionamento Ajax

O AJAX é assíncrono, pois o *browser* não vai esperar pela informação devolvida pelo servidor, mas vai processá-la somente quando esta for enviada pelo servidor. Esta característica é importante, dado que a aplicação Web não fica em estado de espera, aguardando o retorno da informação do servidor.

Figura 2.10: Logotipo Ajax

2.7.2 JQuery

O jQuery é uma biblioteca JS muito completa e que permite aos programadores adicionar elementos dinâmicos e interativos às suas páginas, atenuar as diferenças de processamento dos vários *browsers* e reduzir o tempo de desenvolvimento das aplicações, aumentando assim a produtividade.

É um projeto open-source, que possibilita uma maneira diferente de escrever código em JS e que se torna num instrumento essencial para fornecer uma grande compatibilidade entre os *browsers*. Tem ainda a capacidade de simplificar a forma como são percorridas as páginas HTML, manipulados os eventos e o acréscimo de interações AJAX numa página Web. Nesta biblioteca, é possível encontrar seletores de elementos HTML, funções para manipular os eventos associados ao HTML, efeitos e animações. Através dos seletores, é possível modificar elementos HTML como um grupo ou como um único elemento. As funções de tratamento permitem executar um código que surge através de um evento, com instruções como *"onclick"* ou *"onchange"* e que fazem com que, após um clique sobre uma área da página ou numa mudança do input de uma lista, corra um *script* que vai ser responsável, por exemplo pela a atualização uma tabela.

Também permite, de forma fácil, alterar o CSS da página de acordo com eventos, como por exemplo a mudança de cor de uma determinada área após um clique sobre a mesma, sendo possível enumerar muitas outras vantagens.

Perante tudo isto acima descrito, são facilmente percetíveis as inúmeras vantagens que esta biblioteca oferece aos programadores na criação de páginas Web.

Figura 2.11: Logotipo JQuery

Capítulo 3

Levantamento de Requisitos

Para a melhoria deste SI, um dos pontos essenciais foi o levantamento de requisitos, os quais irão permitir que o produto final se encontre de acordo com o idealizado pela empresa. Depois de algumas trocas de impressões, de forma a compreender o que era pretendido, e após analisar o SI já existente, foi possível elaborar uma lista de requisitos fundamentais. Estando esta etapa concluída, foi mais fácil fazer uma calendarização e organização de prioridades de implementação para se atingir um SI mais completo e robusto que cubra as necessidades de produção da Nibble – Engenharia Lda. Assim sendo, esses requisitos foram divididos em "Melhorias ao Sistema" (M, que são aspetos que já estão implementados e apenas sofrerão aperfeiçoamentos) e "Novas Funcionalidades" (NF, aspetos que a empresa considera importantes ter no SI, depois da experiência que teve até então com o SI atual). Cada requisito apresenta um id, que não terá em conta a prioridade de implementação, um título, uma descrição que inclui um resumo do que se pretende com esse requisito e ainda os testes, onde estão especificados os conjuntos de testes que foram realizados a fim de verificar se tal requisito foi, ou não, cumprido. A cada um segue-se também uma breve explicação do modo de contribuição da implementação do requisito para a melhoria do SI e do sistema de produção e como tal para o rendimento da empresa.

1 (NF)	Realizar Estatísticas de Falhas
Descrição Permitir saber quais os produtos e modelos de produtos que avariam mais frequentemente, quais os testes onde se verificam mais falhas e quais as falhas que ocorrem com maior frequência. Estas informações serão apresentadas numa tabela através da Interface, estando disponíveis para *download* sob a forma de ficheiro ".xls". Esta funcionalidade é essencial para que se possam implementar medidas que evitem a continuidade das avarias.	
Teste Inserir falhas propositadas em cada produto, modelos e testes diferentes e verificar se a tabela disponibilizada apresenta as informações corretas. Efetuar o *download* e confirmar que o ficheiro ".xls" contém as falhas introduzidas.	

A correção de avarias dos produtos é essencial para a melhoria do desempenho de qualquer empresa, sendo, por isso, de extrema importância para a Nibble - Engenharia Lda. que haja uma rápida deteção dos erros mais frequentes ao nível da produção, o que facilita, naturalmente, a sua correção. Desta forma, esta nova funcionalidade a acrescentar ao SI já usado pela empresa, vai permitir aperfeiçoar a produção e melhorar o seu rendimento.

2 (NF)	Associar mais que um Componente a um Produto
Descrição Permitir que todos os componentes essenciais estejam associados ao produto. Para cada modelo de produto, estará pré-definido um conjunto de extras que poderão ser adicionados a esse produto no formulário de montagem. Na lista de produtos apresentada na página de montagem estará presente o componente padrão do produto em questão, definido pelo técnico como o mais importante, que poderá ser alterado. Ainda nesta lista, poderão ser adicionados novos extras ao produto. Atualmente, apenas um componente por produto pode ser adicionado.	
Teste Associar os vários constituintes a cada um dos produtos de diferentes formas. Em primeiro lugar, adicionar os componentes ao produto através do formulário de montagem, que através do número do serviço irá obter o modelo do produto que foi testado e apresentar a listagem pré-definida dos extras do modelo em questão. Em segundo lugar, adicionar apenas um componente neste formulário que irá figurar na lista de produtos montados. Em seguida, adicionar novos extras e mudar o padrão já definido, verificando que as informações ficaram atualizadas.	

A associação de todos os componentes ao produto permite uma mais correta análise do produto em questão. É essencial alargar o número de componentes atualmente associados ao produto, de forma a que quando seja efetuada uma pesquisa acerca do mesmo se obtenham todas as informações necessárias. Assim, esta nova funcionalidade vai melhorar a listagem de componentes de um dado produto, o que é muito importante no caso de necessidade de conhecimento de detalhes do produto (como seja a realização de prévias alterações aos vários componentes). Deste modo, a empresa beneficia da existência de dados mais completos, o que, naturalmente, melhora todo o processo.

2.1 (NF)	Definir Tipos de Componentes para cada Modelo
Descrição Facilitar a associação de componentes aos produtos, estando pré-definidos os extras que cada modelo pode suportar. A associação é feita através dos ids do tipo de extra e do modelo de produto.	
Teste Após pré-definir diferentes tipos de extras para o mesmo produto mas modelos diferentes, verificar que as opções de novos extras a adicionar a cada produto variam consoante o modelo do produto em questão. Na barra presente na página de montagem, selecionar diferentes serviços (com modelos de produtos diferentes) e confirmar que os extras apresentados correspondem à informação da BD. No caso de se optar pela adição a um item já montado, comprovar que os extras correspondem ao modelo do item selecionado.	

Com a implementação do requisito abordado em 2, surgiu a oportunidade de pré-definir uma lista com os extras a associar aos respetivos modelos, tornando assim mais fácil a realização da associação de componentes a produtos. Desta forma, para cada modelo de produto, é possível adicionar variados componentes já presentes na base de dados, fazendo deste um processo mais célere e completo.

3 (NF)	Associar uma Lista de Falhas aos Testes
Descrição Permitir que o técnico possa adicionar novos tipos de falhas aos diferentes testes realizados. A lista criada fica associada ao respetivo teste e ao produto, podendo ser utilizada daí em diante nos processos de testes realizados e consultada nos processos de reparação.	
Teste Na página dos testes ao produto, inserir algumas falhas em diferentes testes de um dado produto e, em seguida, confirmar que a lista de falhas está corretamente associada quer ao teste quer ao produto; Verificar que após a submissão de um relatório de testes, a falha selecionada para um determinado teste ficou associada ao mesmo; No processo de reparação, abrir o relatório de teste efetuado e verificar que a falha está corretamente identificada.	

A lista de falhas é importante para a sua posterior análise e correção, devendo, por isso, ser atualizada logo aquando do processo de testes, o que permite, imediatamente, uma poupança de tempo e ganho de qualidade. Da mesma forma, no processo de reparação, esta funcionalidade permite o acesso imediato à falha ocorrida em determinado teste. Assim, fica garantida a realização

de uma lista de falhas mais correta e associada a um processo significativamente mais rápido.

4 (NF)	Adicionar Novos Tipos de Produtos através da Interface
Descrição Atualmente o sistema é usado para a Firewall, o Comunicador GSM e a Sirene. Esta funcionalidade irá permitir a um administrador adicionar novos produtos através do preenchimento de um formulário executado na interface, não necessitando de recorrer diretamente à BD.	
Teste Inserir no sistema um produto novo através do formulário adequado e confirmar que as suas informações estão presentes na BD.	

Esta nova funcionalidade permite um mais fácil manuseamento do SI, sendo possível a qualquer técnico adicionar novos produtos. A linguagem da base de dados não é de simples entendimento, não sendo, por isso, acessível a todos. No entanto, com a implementação desta funcionalidade, poderá ser possível, através de uma linguagem básica e de etapas facilmente realizadas, melhorar a performance da empresa, diminuindo o tempo até então despendido na introdução de novos produtos.

5 (NF)	Adicionar Versões de Firmware (FW) e Hardware (HW) através da Interface
Descrição Neste sistema, para adicionar novas versões de FW e HW é necessário fazê-lo diretamente na base de dados. Concluído o projeto, deverá ser possível adicionar novas versões de FW e HW através do preenchimento de um formulário executado na interface e como tal sem ter de recorrer diretamente à BD.	
Teste Inserir no sistema novas versões de FW e HW através do formulário adequado e confirmar que estas foram introduzidas corretamente na BD, estando disponíveis para a utilização.	

Analogamente ao anteriormente descrito, através desta nova funcionalidade, as novas versões de Firmware e Hardware poderão ser introduzidas de uma forma consideravelmente mais simples e rápida, não sendo necessária a sua realização por um profissional especializado. Tudo isto aumenta a eficácia e a celeridade do processo, constituindo óbvias vantagens para a empresa.

6 (NF)	Realizar Pesquisa por Número de Serviço
Descrição Permitir realizar pesquisas através do número de serviço	
Teste Introduzir um número de serviço e confirmar que é apresentada toda a informação das ações feitas associadas a esse número de serviço.	

A pesquisa através do número de serviço permitirá um mais simples e célere acesso a toda a informação do serviço previamente realizado, presente sob a forma de relatório de testes ou relatório de reparação, o que agiliza todo o processo e melhora, significativamente, a sua qualidade.

7 (NF)	Introduzir Privilégios nos Utilizadores
Descrição Diferenciar os tipos de utilizadores do sistema, estando os administradores habilitados a aceder a tarefas como sejam a consulta das estatísticas realizadas e a adição de novos produtos e novas versões de FW e HW. Os técnicos têm um acesso limitado a tarefas importantes para a produção.	
Teste Fazer login como técnico e verificar que não há acesso ao *"backoffice"* do sistema. Fazer login como administrador e verificar que é possível realizar ações de *"backoffice"*.	

Esta funcionalidade permite fazer a distinção entre os dois tipos de utilizadores do SI. Desta forma, os administradores podem, a partir de agora, ter acesso diferencial a tarefas e informações importantes, distinguindo-se assim dos técnicos.

Assim, os técnicos têm acesso aos dados e tarefas importantes para a produção, como por exemplo a realização dos testes, processos de montagem e reparação e a gestão da lista de falhas dos testes realizados. As informações mais específicas, como os resultados das estatísticas realizadas e a introdução de novos produtos e novas versões de hardware e software, ficam somente acessíveis a quem estiver designado como administrador.

8 (M)	Melhorar Tempo de Resposta do Sistema de Informação
Descrição Diminuir o tempo de resposta de cerca de 6seg (que se verifica atualmente) para igual ou inferior a 2seg em páginas cruciais à utilização do sistema como a de processos de teste e de montagem.	
Teste Navegar entre as várias páginas do sistema acima referidas e conferir que o tempo de resposta do SI está dentro do que foi definido inicialmente (cerca de 2seg).	

A implementação desta melhoria no SI é bastante importante, uma vez que permite aos técnicos acelerar os processos de testes e reparações e ainda de inserção de produtos montados.

A partir de agora, com a diminuição do tempo de resposta do SI, está melhorada a sua usabilidade e é acrescentada qualidade aos processos realizados.

Este aperfeiçoamento constitui uma grande mais-valia para a empresa, pois dará origem a um processo mais rápido e eficaz.

9 (M)	Introduzir Paginação no Histórico
Descrição Neste momento, no menu do histórico, apenas é possível consultar os últimos 10 números de serviço. Pretende-se que seja possível, através de uma paginação, consultar mais resultados, isto é, pelo menos 50 resultados.	
Teste No menu do histórico verificar que existem os últimos 50 números de serviços e também que é possível consultar detalhes do que foi realizado nos mesmos.	

A implementação da paginação do histórico facilita o acesso aos serviços que foram realizados mais recentemente. Anteriormente, eram disponibilizados apenas os últimos dez serviços, o que limitava, um pouco, o acesso aos relatórios de teste recentes. Tal aperfeiçoamento garante também uma maior comodidade para os técnicos e contribui para um trabalho mais rápido e eficiente.

Capítulo 4

Arquitetura do Sistema de Informação

Neste capítulo será apresentado o estudo da arquitetura e as respetivas mudanças que foram realizadas neste Sistema de Informação (SI) ao longo deste projeto, dado que não existia documentação que permitisse obter uma visão sobre a sua estrutura. De referir que a arquitetura inicial sofreu pequenas alterações, pois existiam partes da aplicação que eram de difícil compreensão e não seguiam uma estrutura muito bem definida. O padrão de arquitetura utilizado para a reformulação desta arquitetura foi o *Model-View-Controller* (MVC). De seguida será apresentada a arquitetura reformulada desta aplicação e serão apresentadas as principais diferenças existentes em relação a arquitetura inicial.

4.1 Visão de Alto Nível do SI

Depois de uma breve análise desta aplicação, é possível apresentar uma visão de alto nível, representada na Fig.4.1, que ilustra de um modo geral os grandes blocos desta aplicação. Isto é, uma BD onde é guardada toda a informação utilizada, como tipos de produtos, produtos montados, operadores, relatórios de teste e guias de transações de produtos, uma aplicação Web responsável por manusear e processar os dados contidos na BD e ainda uma interface Web onde serão apresentados ao utilizador os resultados do processamento dos dados.

Figura 4.1: Visão Alto Nível da SI

4.2 Modelo MVC

Depois de reformulado, podemos afirmar que este SI segue o padrão de arquitetura MVC. Em seguida, serão explicados como são compostos os blocos deste padrão para o SI em estudo e por fim irá ser explicado tanto o funcionamento como as diferenças existentes entre a arquitetura atual e a inicial.

4.2.1 Modelo

Como foi referido anteriormente, o modelo contém não só a BD, como também o conjunto de funções/métodos que permitem o acesso aos dados. Comecemos então por estudar a estrutura da BD atual.

Base de Dados

Neste trabalho, é muito importante manter uma base de dados bem estruturada e organizada, uma vez que se a informação não estiver bem organizada, a sua utilização torna-se muito complicada e demorada. Nesta BD da aplicação são armazenados os tipos e modelos de produtos e operadores existentes, a composição de cada item (produto montado), todos os testes com a sua respetiva categoria que podem ser feitos aos produtos, os relatórios realizados e respetivos resultados, assim como relatórios de reparação e documentos de entrada e saída de material.

Esta informação está armazenada em diferentes tabelas. Essas tabelas estão apresentadas na Figura 4.2 que apresenta as tabelas existentes e a sua relação, bem como a cardinalidade da mesma.

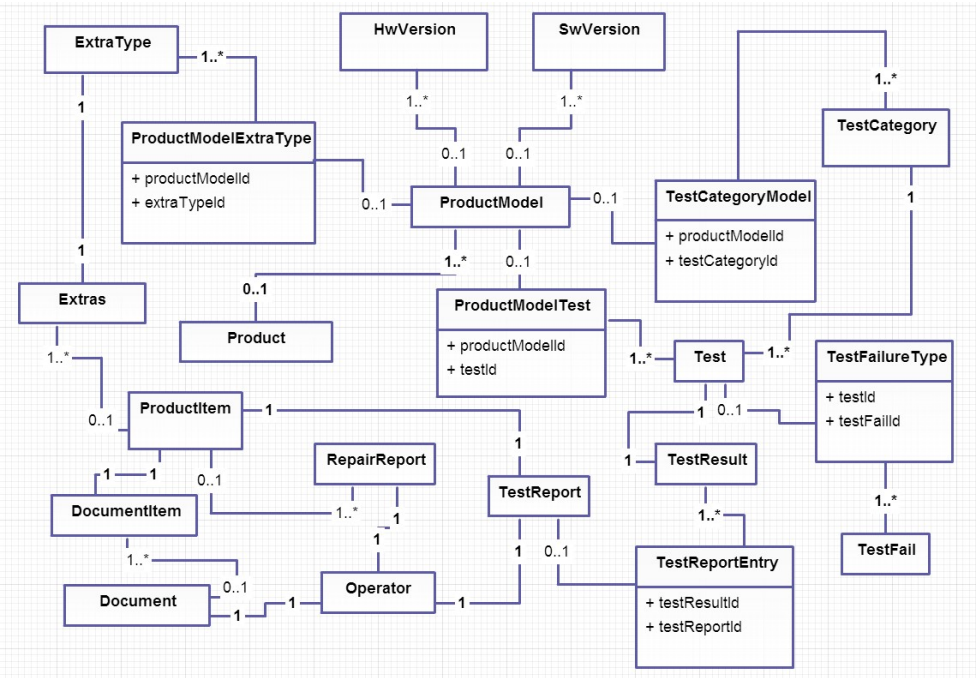

Figura 4.2: Modelo Relacional da BD

Fazendo uma breve abordagem ao conteúdo das tabelas presentes na BD é de salientar:

- 'Operator': armazena os dados relativos a cada utilizador do sistema;
- 'ProductModel': contém todos os modelos possíveis para os diferentes tipos de produtos. Estes tipos de produtos estão presentes na tabela 'Product';
- 'ProductItem': corresponde a cada produto físico já montado. Um productItem já tem associado, entre outras coisas, um número de série;
- 'Extras': armazena todos os componentes e respetiva informação, como o número de série e o tipo de componente a que pertence, que são adicionados aos diferentes productItem. O tipo de componente está guardado na tabela 'ExtraType';
- 'TestReport': guarda a informação referente a cada relatório de testes submetido, como seja qual o modelo do produto, a versão de SW e HW utilizada, o operador responsável e o resultado (positivo ou negativo) do teste;
- 'TestResult': contém o resultado de cada teste realizado em cada relatório. Em caso de falha, identifica o id da falha associada. A informação sobre as falhas, como o nome e descrição, encontra-se na tabela 'TestFail';
- 'Test': estão presentes todos os testes possíveis de fazer aos produtos. Aqui, pode-se encontrar o nome do teste, os critérios para validar ou não o teste realizado, a descrição e ainda a categoria de testes a que pertence. Por sua vez, estas categorias permanecem na tabela 'TestCategory', na qual será possível agrupar os testes em várias categorias e para diferentes tipos e modelos de produtos;
- 'RepairReport': guarda a informação do item que vai ser reparado, o cliente que o enviou para reparação e depois, caso seja submetido novo relatório de testes (depois de reparado), guarda também o relatório que ficou associado a esta reparação e o operador responsável pelo mesmo;
- 'Document' e 'DocumentItem': são tabelas que contêm informação sobre o estado atual dos itens, ou seja, informam se estes já foram enviados, em que guia foram enviados, para que cliente ou se já retornaram à empresa para reparação, entre outras tantas informações. Cada documento tem de ter um operador associado.

Acesso aos Dados

Além da BD, o modelo contém também as funções para aceder à mesma. Estas funções estão contidas em classes, que possuem métodos *getters* e *setters* que permitem aceder aos dados presentes em cada tabela. Cada tabela tem uma classe associada formando deste modo uma camada de abstração da BD.

O diagrama de classes do sistema pode ser visto na Figura 4.3, onde se pode constatar que existe uma classe "MySQLDbObject" que funciona como uma classe mãe e que é responsável pelas funções de ligação e término de ligação com a BD e de validação da sintaxe das *queries*. Dessa classe deriva a classe DbObject que faz a abstração das funções de inserção, atualização e eliminação de dados para qualquer objeto, contém os métodos para obter o nome da tabela, vista ou chave-primária e ainda dos métodos Get e Post, sendo criada uma função 'readRequest()' que permite obter um parâmetro enviado através destes métodos, utilizando "readRequest('parâmetro')". No nível abaixo surgem quase todas as tabelas do sistema que já possuem atributos privados, que correspondem aos campos da BD e permitem a criação dos métodos *getters* e *setters* para manipulação dos mesmos. Assim sendo, quando na aplicação forem utilizados objetos de uma dada classe é possível através destes métodos obtê-los.

No último nível do diagrama de classes surgem as classes de 'SwVersion' e 'HwVersion', pois derivam de uma classe 'Version', dado que são muito idênticas. Estas classes apresentam diferenças ao nível da tabela que representam.

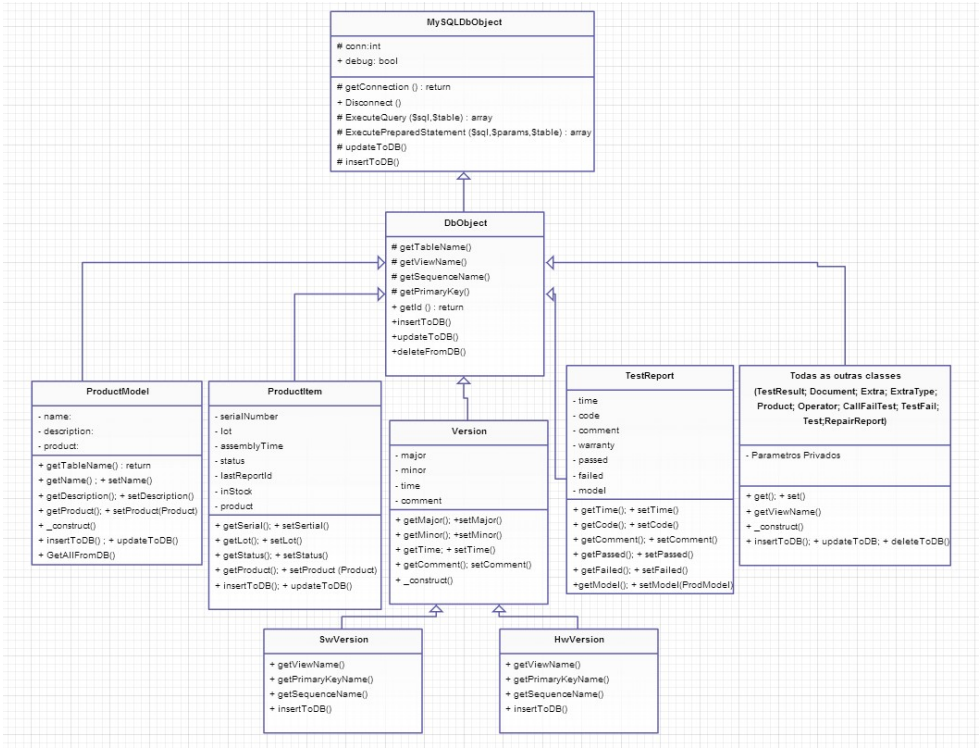

Figura 4.3: Diagrama de Classes

4.2.2 Controlador

Neste sistema existem 2 tipos de controladores. O primeiro tipo representa os controladores mais comuns nos sistemas que contêm formulários. Estes controladores recebem os dados introduzidos nos formulários através da função readRequest(), sendo em seguida verificado se todos os inputs que são necessários para a rotina a executar foram recolhidos com sucesso, caso contrário devolve uma mensagem de erro. Por último executa a rotina pré-definida, que no caso destes controladores passa por criar um objeto da classe, cujos dados vão ser manipulados na BD, utilizar os métodos *setters* para atribuir os novos valor e no fim invocar a função de inserção ou atualização da tabela.

Para a melhor compreensão, este funcionamento está representado na Figura 4.4.

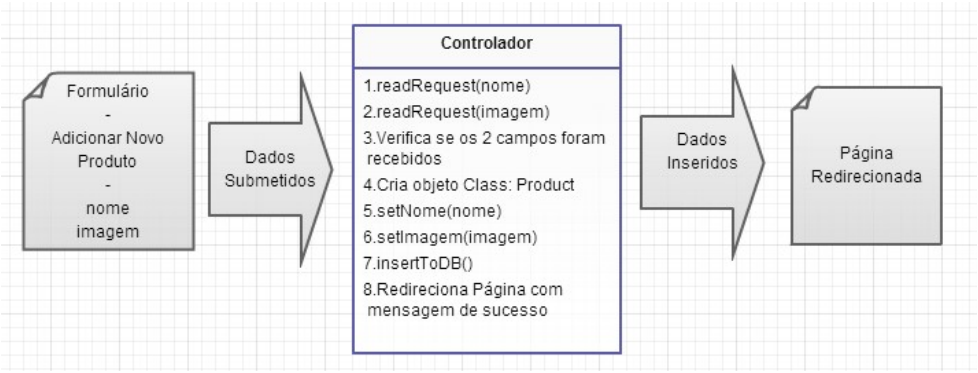

Figura 4.4: Controlador Tipo 1

O outro tipo de controladores presentes são os que permitem recolher os dados que vão ser utilizados pela visão que irá ser apresentada mais abaixo. Assim sendo, depois de solicitada uma página pelo utilizador, o controlador executa as funções adequadas de cada classe para obter os dados do modelo que irão ser armazenados em variáveis ou vetores que vão ser enviados para a visão, como é possível verificar na Figura 4.5.

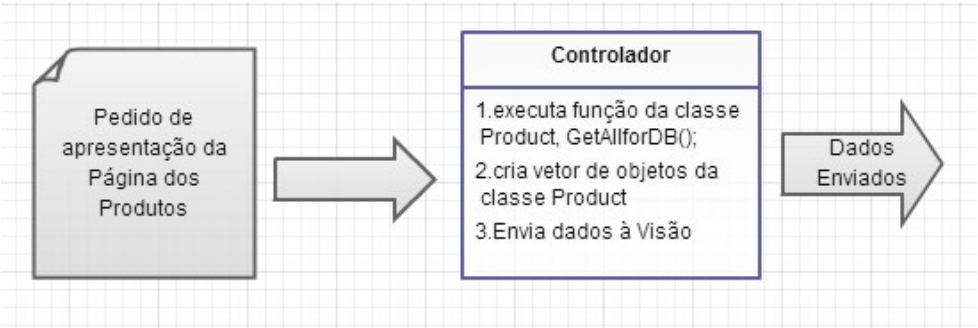

Figura 4.5: Controlador Tipo 2

4.2.3 Visão

Quanto à Visão, esta é formada por um conjunto de ficheiros que são responsáveis por criar as interfaces para o utilizador da aplicação, que serão reproduzidas no browser. Estes ficheiros contém o código HTML da página a gerar e o código JS que é executado no caso de haver interações com alguns dados HTML. Estes ficheiros possuem também ciclos PHP para apresentar os dados presentes nas variáveis ou vetores que o controlador envia. Seguindo o exemplo apresentado no esquema da Figura 4.5 usada na demonstração do controlador, a Visão recebe o vetor de objetos da classe 'Product', executa um ciclo para percorrer o vetor utilizando PHP e apresenta sob a forma de tabela os dados relativos a cada produto. Isto, pode ser verificado através do seguinte esquema.

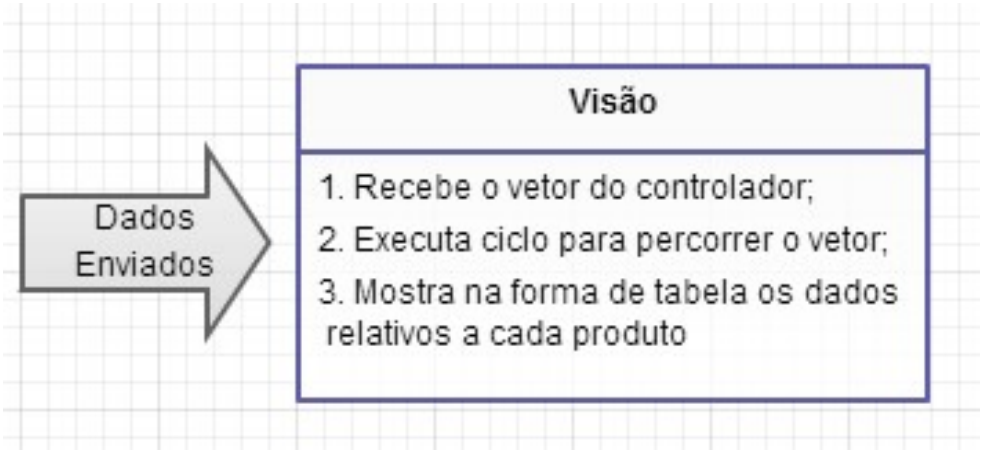

Figura 4.6: Visão

Neste momento, é possível perceber, de forma genérica, como funciona o MVC nesta aplicação.

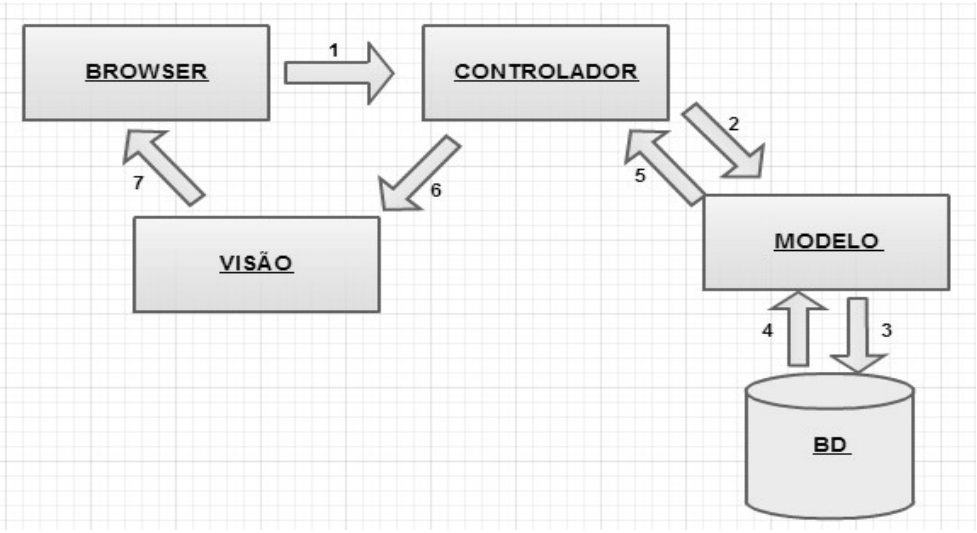

Figura 4.7: Fluxo MVC do Sistema

0. O utilizador interage com o *Browser*, como por exemplo através do preenchimento de um formulário;
1. O *browser* envia o pedido ao Controlador;
2. O Controlador manipula o evento da Visão do utilizador, através de uma rotina pré-definida, e comunica com o Modelo, podendo atualizá-lo de acordo com a informação inserida pelo utilizador, como seja a atualização dos dados de um dado produto;
3. O Modelo executa as trocas na BD;
4. São enviados ao Modelo os dados atualizados;
5. O Controlador executa as funções presentes nas classes e guarda os dados;
6. A Visão recebe os dados do Controlador e gera uma nova Visão;
7. O *Browser* apresenta a Visão criada e aguarda novas interações para reiniciar o ciclo;

4.3 Principais Diferenças na Arquitetura

As principais diferenças existentes entre a arquitetura do SI atual e a do SI inicial prendem-se com o fato de agora haver uma separação bem marcada entre o Controlador e a Visão, tanto em termos de código como de diretório de ficheiros. Assim, cada ficheiro responsável pela criação de uma interface para o utilizador tem na pasta "*controllers*"um ficheiro com o mesmo nome, para que a associação entre os dois seja mais acessível e simplifique uma posterior utilização do SI. Outra diferença a realçar é de que ao nível da BD existe um modelo relacional devidamente interligado e com a devida cardinalidade das relações entre entidades. Desta forma, e contrariamente ao que se verificava no início deste projeto, o entendimento das relações existentes entre as tabelas da BD tornou-se significativamente mais simples, o que facilita a utilização da BD em todo o SI.

Capítulo 5

Implementação dos Requisitos

Neste capítulo será abordado o modo como os requisitos levantados no inicio do trabalho foram implementados no SI existente na Empresa.

1. Realizar Estatísticas de Falhas

Foi criada uma página "stats.php" à qual é possível aceder através da página de administrador, tendo esta última sido criada aquando do cumprimento do requisito "Introduzir Privilégios nos Utilizadores". Na página de estatísticas, foi também criada uma caixa de seleção, na qual foram introduzidas as quatro opções de estatísticas (Fig 5.1) explicadas mais adiante, e à qual está associada uma função JS que é executada quando ocorre um evento "onChange", ou seja, a opção for trocada, e que tem como parâmetro o valor da opção.

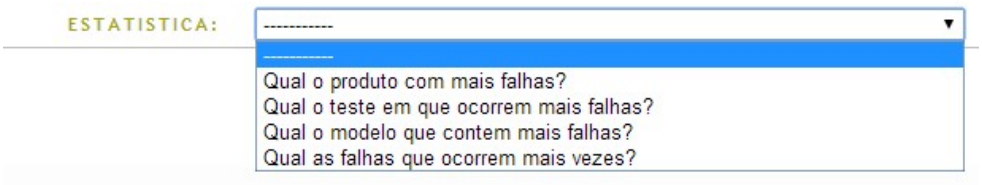

Figura 5.1: Opções de Estatísticas

No momento em que a opção é selecionada, essa função recebe o valor associado à opção escolhida e identifica a opção escolhida, interrompendo o *script* e enviando um objeto "XMLHttpRequest". Através deste, abre-se um ficheiro em background, sendo passados os parâmetros necessários para a sua execução, e aguarda-se uma resposta. Quando esta resposta estiver disponível, o servidor envia um objeto "XMLHttpResponse" que, neste caso, contém o código HTML que irá permitir apresentar a tabela correspondente à estatística pedida. Neste caso, o *script* que ocorre em *background*, recebe como parâmetro a opção escolhida e executa a função associada a cada estatística, obtendo uma resposta para enviar ao *browser*.

A solução escolhida para organizar o código passou pela criação de uma função que fosse acumulando numa variável o código HTML que irá dar origem à tabela final com os respetivos dados e no final apresentasse essa variável, recorrendo ao comando *'echo'*.

Esses dados resultam da execução de funções, que utilizam a informação contida nas tabelas "TestFail", "ProductModel" e "TestResult", pelo que foi necessário agregá-las, através da criação da vista "StatFailView".

Com a criação da vista concluída, iniciou-se a abordagem a cada opção de estatística individualmente.

Para a opção "Qual o produto com mais falhas?" foi criada uma função que agrupa os dados presentes na vista, através do id do produto, e da qual é retirada a informação do nome do produto e ainda uma contagem do número de tipos de falhas diferentes existentes em cada produto. Esta informação é guardada na forma de vetor, associando em cada posição o produto ao número total de falhas.

Através da função "array_ column()" é possível extrair de um vetor a informação dos campos, utilizando para isso o nome da coluna selecionada na *query*, isto é, executando $result1 = array_ column($re,'coluna'). Sendo $re o vetor que contém os resultados, obtém-se um vetor 'result1', o qual conterá os dados da coluna do vetor 're'. Por fim, quando obtiver as várias colunas pretendidas, através de um ciclo "foreach" é possível obter os dados para expor na tabela, pois a informação contida na posição 0 do vetor result1 tem correspondência à informação da posição 0 do vetor result2. Por exemplo, se na posição 0 result1 contém "Firewall", na posição 0 do vetor result2 terá o total de falhas associadas à Firewall. Na tabela, ainda é mostrada qual percentagem correspondente ao número de falhas de cada produto atendendo, para isso, ao número total de falhas. Este número total é obtido através de uma contagem de diferentes tipos de falhas presentes na tabela 'TestFail'.

Para a opção "Qual o teste em que ocorrem mais falhas?" todo o processo ocorre similarmente em relação ao explicado anteriormente, sendo apenas mudada a função a executar para a obtenção dos dados. Para este caso, obtém-se o teste correspondente, o nome do produto e a soma das falhas que serão agrupadas, tendo em conta o id do teste.

Para a opção "Qual as falhas que ocorrem mais vezes?" é igualmente utilizado o mesmo processo, sendo o agrupamento de dados feito através do nome da falha.

Para a opção "Qual o modelo que contém mais falhas?" foi criada uma nova vista "ModelStatFailView" que agrega a informação da tabela das falhas 'TestFailureType', responsável pela associação da falha ao teste e modelo do produto, sendo de seguida utilizada a mesma lógica, mudando apenas a variável de agregação, que neste caso é o id do modelo do produto.

Para completar este requisito, foi também adicionado um *script* que recebe a informação de qual a opção selecionada pelo utilizador e executa a função correspondente, copiando a variável que a função retorna e na qual está contido o HTML da tabela. Esta variável contém também os cabeçalhos (*headers*) que permitem ao *browser* interpretar aquela página como um ficheiro Excel, fazendo download do mesmo.

2. Associar mais que um Componente a um Produto

Anteriormente, no SI, estava incluída a associação de apenas um componente para cada item, sendo que esse componente se encontrava guardado num campo da tabela "ProductItem", representado pelo seu número de série. Assim, e para que fosse possível associar a cada produto mais que um componente, foi inicialmente necessário definir os tipos de componentes disponíveis. Essa informação fica guardada na tabela "ExtraType" e engloba o id, o nome e uma descrição do mesmo. Para que haja a associação de cada extra a um item, foi criada a tabela "Extras", que contém o id do componente, o id do tipo de componente (para se saber qual o tipo de componente que estamos a associar), o id do item (para ser percetível qual o item que está a receber o componente em questão), um identificador ou um número de série, que é único no SI, e ainda um campo *"default"*, que é definido pelo utilizador e é essencial na indicação de qual dos componentes é o padrão de cada item, podendo este padrão ser alterado. Esta informação é muito relevante aquando da listagem dos produtos montados, na qual aparecerá o item e o componente associado (o padrão). Os restantes que poderão ser adicionados surgem aquando da solicitação dos detalhes do produto, como será explicado em seguida.

Após a realização das mudanças ao nível da base de dados, no código foi necessário criar as classes "ExtraType" e "Extras" e os respetivos métodos *'get'* e *'set'*, para ser possível obter e editar os valores dos dados, criar os construtores de cada classe e ainda as respetivas funções de "insertDB" e "updateDB". Posto isto, na página "details.php", na qual se encontra o formulário que inclui os detalhes do produto (o número de série do produto, os vários componentes que o compõe, a devida identificação do padrão e ainda o lote no qual está incluído o dado item), quando o utilizador clica no ícon da lupa presente na listagem dos produtos montados são apresentados os detalhes do item. Estes surgem, sob a forma de formulário, numa janela *popup*, isto é, uma janela que passa para primeiro plano na interface, criada através da função presente na biblioteca JQuery "('# div').dialog()" e que é responsável por dimensionar a janela e configurar as ações dos botões 'Ok' e 'Cancelar' presentes na mesma. Foi definido que o 'Cancelar' é responsável pelo fecho da janela e o 'OK' realiza a submissão dos dados nos inputs do formulário. O formulário é invocado através de outra função JQuery, "('# div').load()" que irá carregar o script 'details.php'.

Depois de executado o *script* são apresentados vários inputs no formulário, que podem ser editáveis ou apenas alvo de consulta. Os campos apresentados são o número de série do produto, os vários extras que o compõe, a devida identificação do padrão e ainda o lote no qual está incluído o dado item.

No *script* 'details.php' é utilizado como parâmetro o id do item selecionado. Com este id, é invocada a função "GetById()" da classe "ProductItem", que cria o objeto desta classe com os respetivos parâmetros obtidos da BD. Destes, são utilizados o número de série do produto e o lote do qual advém, sendo estes dados apresentados nos respetivos inputs. Em seguida, são executadas duas funções da classe "Extra", a primeira que permite a obtenção do número de extras associados ao item em questão e a segunda que obtém um vetor de objetos da classe "Extra" que estão associados ao item. Os nomes dos tipos de componentes são adquiridos através do objeto

"ExtraType" associado ao parâmetro "extra_ extraTypeld", presente na tabela "Extra". De referir ainda que os inputs dos extras são dinâmicos, isto é, surgem de acordo com o número obtido na função "GetCoutforProductItem()". Estes aparecem dentro de um ciclo que itera tantas vezes quantos extras existirem no item. Os ids dos inputs associados aos números de série dos extras são obtidos sobre a forma de vetor, a fim de obter um número diferente de inputs, consoante o item e a informação do extra padrão, que é recebida também na forma de vetor e na qual é indicado, na posição [0], o id do componente no qual o *radio option* está selecionado, ou seja, se o componente selecionado tiver o id igual a 3, o vetor default[] terá na posição [0] o valor 3. São enviados ainda dois campos do tipo *hidden*, ou seja, que não aparecem na interface, estando num contida a informação do número de componentes e no outro o id do componente presente em cada posição do vetor.

Com tudo isto acima explicado, o formulário surge na forma:

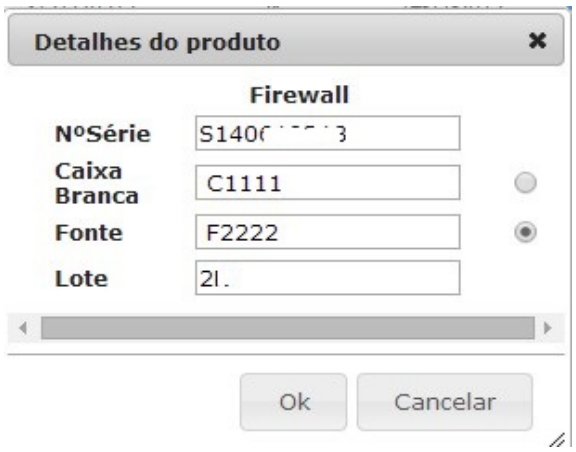

Figura 5.2: Exemplo de Detalhes de um Item

Foi ainda criada a ação que permite receber e manipular os dados introduzidos no formulário e que ocorre se os campos forem alterados e o botão "OK" for submetido. A ação deste formulário está contida no "update.php" na pasta "ProductItem", que primeiramente recebe os parâmetros do formulário. Os parâmetros do número de série do produto e do lote são recebidos e, através dos métodos "setSerialNumber" e "setLot", são alterados no construtor, que depois é invocado na função "updateDB()", presente na mesma classe. Feita esta atualização é executado um ciclo com um número de componentes que foi passado como *hidden*, que a cada iteração atualiza o número identificativo do mesmo e compara a posição do valor do vetor extrald[], também enviado como *hidden*, com o número presente na primeira posição do vetor "default[]". Caso sejam iguais, o parâmetro *'default'* desse item é alterado para 1, caso contrário o valor é 0. Esta condição permite alterar o componente padrão escolhido, pois como só um pode ser escolhido, temos, não só de pôr o escolhido a 1, como também o que estava escolhido anteriormente a 0.

Ainda neste requisito, foi alterada a listagem de produtos montados. Onde anteriormente aparecia o campo presente na tabela "productItem" como "productItemComposite", uma vez que

era só um, é agora executada uma função que verifica qual é o extra padrão de cada item e o apresenta na respetiva coluna.

2.1. Definir Tipos de Componentes para cada Modelo

Para pré-definir a lista de componentes que poderão ser associados a cada modelo de produto, foi criada na BD uma tabela "ProductModelExtraType" que é composta pelo id do tipo de componente e do modelo do produto. Neste momento, os dados desta tabela terão de ser inseridos diretamente na BD.

Foi criada ainda uma vista "ProductModelExtraTypeView", que associa a informação existente nas tabelas dos tipos de extras, dos extras e do modelo do produto e permite, de forma fácil, saber quantos e quais os tipos de componentes que é possível associar a um item, assim como obter tanto o número identificativo do componente como qual dos componentes foi escolhido como padrão.

3. Associar uma Lista de Falhas aos Testes

Ao nível da BD já existia previamente uma tabela para as falhas "TestFail", contudo de nada servia, uma vez que ainda não era utilizada no SI. Existia também uma tabela "TestFailureType", na qual estão contidas as chaves primárias, correspondentes aos ids, das tabelas "Test", "TestFail" e "ProductModel". Para a tabela "TestFailureType", foi criada uma classe "CallFailTest", onde é declarado o seu construtor, e uma função para inserir os valores na BD.

Esta funcionalidade foi adicionada à página de testes, através do símbolo '+' presente a seguir a cada teste apresentado. Cada teste tem também uma caixa de seleção, na qual estão apresentadas as falhas correspondentes ao mesmo.

Para adicionar uma nova falha, foi criado um formulário HTML, utilizando o mesmo método de criação de formulários da associação de produtos referida anteriormente, isto é, utilizando a função JS invocada através do evento "onClick".

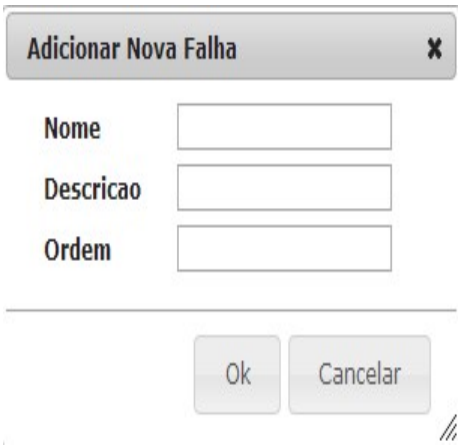

Figura 5.3: Formulário para Adicionar Nova Falha

Neste formulário, é pedido um nome para a falha e uma descrição da mesma, que possibilita, por exemplo, identificar qual componente associado aquela falha para posteriormente obter a informação sobre o componente que falha mais vezes, mas também pode servir apenas para descrever os sinais necessários para assumir aquela falha. É igualmente necessário atribuir um número de ordem, que será utilizado para a ordenação da lista de falhas apresentadas.

Depois de submetidos os dados introduzidos, a ação faz a leitura dos mesmos e ainda recebe o id do teste em questão, assim como o modelo do produto, ambos enviados como inputs do tipo *hidden*. Em seguida, é criado um objeto da classe "TestFail" e, através dos métodos set, são atribuídos os valores aos parâmetros corretos. Por fim, é invocada a função responsável pela inserção na BD.

Depois de inserida a falha, é criado um objeto da classe "CallFailTest" que utiliza o id da falha que acabou de ser inserida, o modelo do produto e o id do teste utilizados como parâmetros *hidden*, sendo estes dados inseridos também na BD. Só depois se dá por concluída a inserção da falha.

O passo seguinte foi a criação de uma vista que permitisse associar e obter, numa única tabela, a informação sobre os testes que contêm uma ou mais falhas, quais os parâmetros dessa falha e ainda qual o modelo do produto a que pertence esse tipo de falhas. Esta vista será utilizada na função responsável pela pesquisa na BD das falhas que existem associadas ao id do teste e do modelo do produto, que são enviados como parâmetros da função. Estes parâmetros são utilizados nas cláusulas "Where" da *query*.

Por último, quando o relatório é submetido, é enviado, para a ação de inserção do relatório na BD, um vetor com os ids das falhas assinaladas em cada teste, o que irá permitir carregar as falhas selecionadas quando o relatório for solicitado.

4. Adicionar Novos Tipos de Produtos através da Interface

Na página "product.php", responsável pela produção da interface do separador "Produto" presente no menu, foi adicionado um ícon que após ser clicado, abre uma janela *popup*, que é criada com o mesmo método das anteriores. O formulário que é carregado através da função "('# div').load()", responsável pelo conteúdo HTML está presente no ficheiro 'addProduct.php'. Neste ficheiro, foram criados os dois inputs necessários para a elaboração de um novo produto, a saber o nome do produto e o nome da imagem do produto com a respetiva extensão, que deverá estar contida na pasta de imagens presente no diretório de pastas do SI.

Depois de ser submetido o formulário, foi ainda criada uma ação de "insert" que irá receber os dados inseridos pelo utilizador e elaborar um objeto da classe "Product" ("new Product"), utilizando os métodos "setName($value)" e "setIcon($value)", onde $value é substituído pelos dados inseridos.

No final, regressa à página inicial, apresentando a mensagem "Produto Inserido com Sucesso" no caso de todo este processo ter sido bem sucedido ou "Falha a Inserir Produto", no caso da ocorrência de algum erro.

5. Adicionar Versões de Firmware (FW) e Hardware (HW) através da Interface

Quando é feito o acesso à página de administrador, são expostas duas opções: aceder à página de estatísticas, como já foi anteriormente explicado, ou selecionar a opção "Adicionar Novas Versões de HW/FW". Selecionanda essa opção dispara um evento JS "onClick" que executa uma função, responsável por mostrar uma divisão que está oculta quando a página é carregada, através da função da biblioteca JQuery "('# div').show(). Esta divisão contém uma caixa de seleção (criada através do HTML) onde é possível escolher se se pretende adicionar uma nova versão de HW ou de SW. Esta escolha é realizada utilizando o evento "onChange". Depois da seleção são disponibilizados os formulários respetivos, caso a intenção seja adicionar HW,

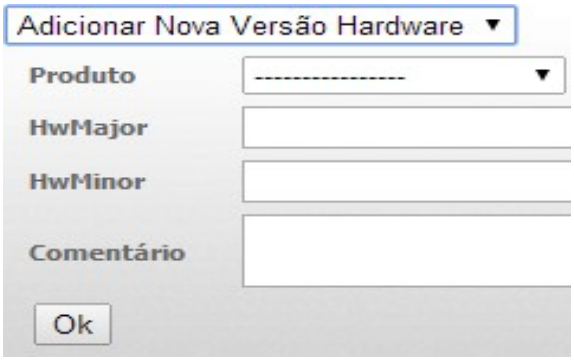

Figura 5.4: Formulário para Adicionar Nova Versão de Hardware

Ou caso seja adicionar SW,

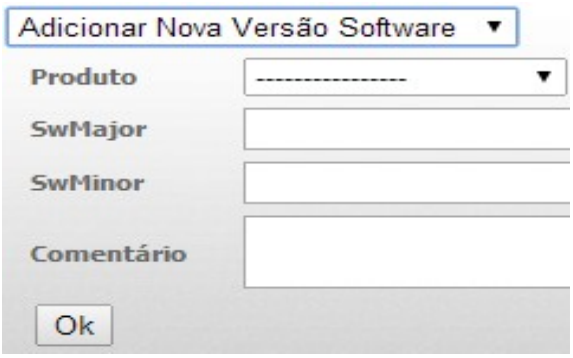

Figura 5.5: Formulário para Adicionar Nova Versão de Software

O primeiro passo é escolher para qual produto se pretende adicionar a nova versão. Esta seleção é feita através de uma caixa de seleção, onde as opções são obtidas com a execução da função "GetAllfromDB" presente na classe "Product", que devolve um vetor de objetos desta classe com os dados presentes na BD. Depois de escolhido o produto, é utilizado, outra vez, o evento "onChange" que executa a função "showModels(value)" que recebe como parâmetro o

id do produto escolhido. Esta função utiliza o Ajax e é capaz de intercetar o pedido da página e executar um script em background no servidor, de modo assíncrono, que depois retorna o conteúdo alterado ou acrescentado na divisão da página que foi definida. Esta tecnologia torna possível atualizar/editar divisões da página Web, sem que esta necessite de ser recarregada por completo. O *script* executado em background está presente no ficheiro "findProductModels.php", que recebe o id do produto selecionado. Com esse id, é criado o objeto da classe "Product" que irá ser utilizado como parâmetro na função "GetAllFromDB()", responsável por extrair da BD todos os modelos disponíveis para o produto selecionado. De seguida, é apenas necessário finalizar a inserção dos dados de *major* e *minor*, em que o *minor* corresponde à família de versões de uma versão *major*, isto é, no caso da versão 3.24, o "3" corresponde ao *major* e o "24" ao *minor*. É ainda possível adicionar um comentário à versão, como por exemplo informar quais as modificações que surgem com aquela atualização.

Por último, foram criados os *scripts* de insert para os dois casos, onde são recolhidos os dados inseridos, sendo de seguida criado o objeto da classe respetiva e efetuada a inserção na BD.

6. Realizar Pesquisa por Número de Serviço

Para esta implementação, foi necessária a reutilização da função "GetAllFromDB" da classe "TestReport", uma vez que o objetivo é obter um relatório de testes tendo por base uma pesquisa resultante da recolha de dados inseridos pelo utilizador. Foi, por isso, criado um input HTML na página de testes, onde é possível inserir um número de serviço para o qual queiramos ter acesso ao seu respetivo relatório. Depois de submetido, esse dado é passado à função que irá adicionar uma condição "Where" à *query*, restringindo o campo "testReportCode" ao código inserido pelo utilizador. É então criado um objeto da classe "TestReport" com todos os dados do relatório resultante do serviço.

Na interface aparece, em forma de tabela, o número de serviço que foi introduzido, qual a data em que foi realizado e ainda o número de série do produto testado. Existe ainda a possibilidade de abrir esse mesmo relatório e ter acesso aos resultados dos testes efetuados.

7. Introduzir Privilégios nos Utilizadores

Para fazer a distinção dos utilizadores do SI, foi adicionado à tabela que contém a informação do operador um novo campo: "operatorAdmin". Este poderá ter os valores "0", caso o utilizador seja um técnico, e "1", para o caso de ser um administrador. Por defeito, quando é criado um novo utilizador este é considerado como técnico.

De seguida, na classe "Operator" foram adicionados os respetivos métodos *'get'* e *'set'* deste parâmetro, tendo-se modificado também o construtor do objeto "Operator" e as funções de inserir e modificar os dados da base de dados.

No *script* "header.php", que produz a interface de entrada no SI, e que, por isso, contém o código que apresenta o menu, foi acrescentada a condição de se o utilizador autenticado no SI,

cujos dados do seu objeto estão guardados numa variável de sessão, tiver parâmetro "operatorAdmin" igual ao valor 1, é apresentado o separador "Administração", no qual é possível o acesso à página de estatísticas e aos menus de adição novas versões de hardware e software. Outra opção que também surgirá neste caso é a possibilidade de adicionar novos produtos, como foi descrito no requisito "Adicionar Novos Produtos".

8. Melhorar Tempo de Resposta do Sistema de Informação

Inicialmente, começou-se por tentar perceber qual o fator que estava a originar um aumento do tempo de resposta do SI.

Para a página de testes foi detetado que a causa do atraso se devia, principalmente, à inclusão do ficheiro "incProductTestLinks.php". Este é responsável pela criação do histórico de serviços anteriores e apresentação dos inputs de pesquisa, quer por número de série quer por número de serviço, no final do código, uma vez que o *script* é interrompido de forma a incluir todo o código que está nesse ficheiro, só depois estando concluído o carregamento da página.

Assim sendo, foi implementada uma divisão nas páginas, onde se pretende que o conteúdo deste ficheiro apareça com o nome de *'links'*, e de seguida, através da função .load() da biblioteca JQuery, é realizado o carregamento dessa divisão quando a página que realmente se pretende estiver completamente carregada. Desta forma, esta divisão é carregada de modo assíncrono.

Esta implementação é de melhor compreensão seguindo o esquema da Fig. 5.6.

Quanto à página de montagem, que se tratava de um dos principais *'bottleneck'* do SI, pois cada vez que era inserido um produto a página era recarregada por completo e a cada ciclo estava a criar a tabela de produtos montados, o que se tornava muito pesado para o SI. Assim sendo, começou-se por tornar o *script* de criação da tabela num *script* independente e assim que é inserido um novo produto montado, a tabela é carregada usando o Ajax, ou seja, a página com a barra de montagem é devolvida quase instantaneamente e a tabela é recarregada de forma assíncrona. Isto é, caso o utilizador volte a inserir um novo produto antes da tabela ter sido recarregada, a execução do carregamento é cancelada, é inserido o produto e a página é carregada já com o novo produto e assim sucessivamente.

Do mesmo modo, esta implementação está representada no esquema da Fig.5.7.

Outro aspeto que foi revisto e melhorado foi o tempo de execução das *queries* que são responsáveis pela criação das vistas que dão origem aos dados necessários para carregar as páginas em questão. Este processo foi capaz de uma ligeira melhoria do tempo de resposta do SI.

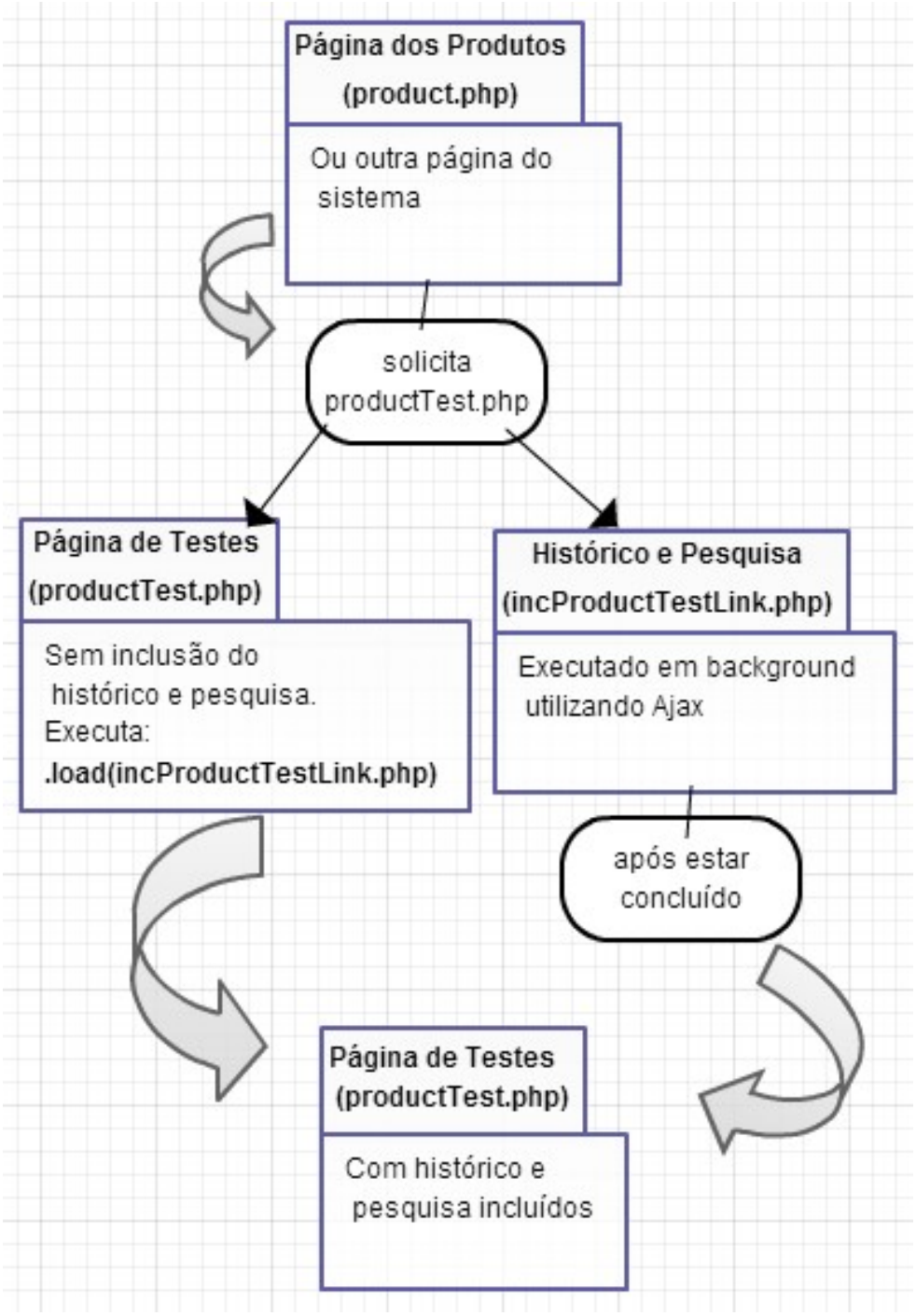

Figura 5.6: Produção da Página de Testes

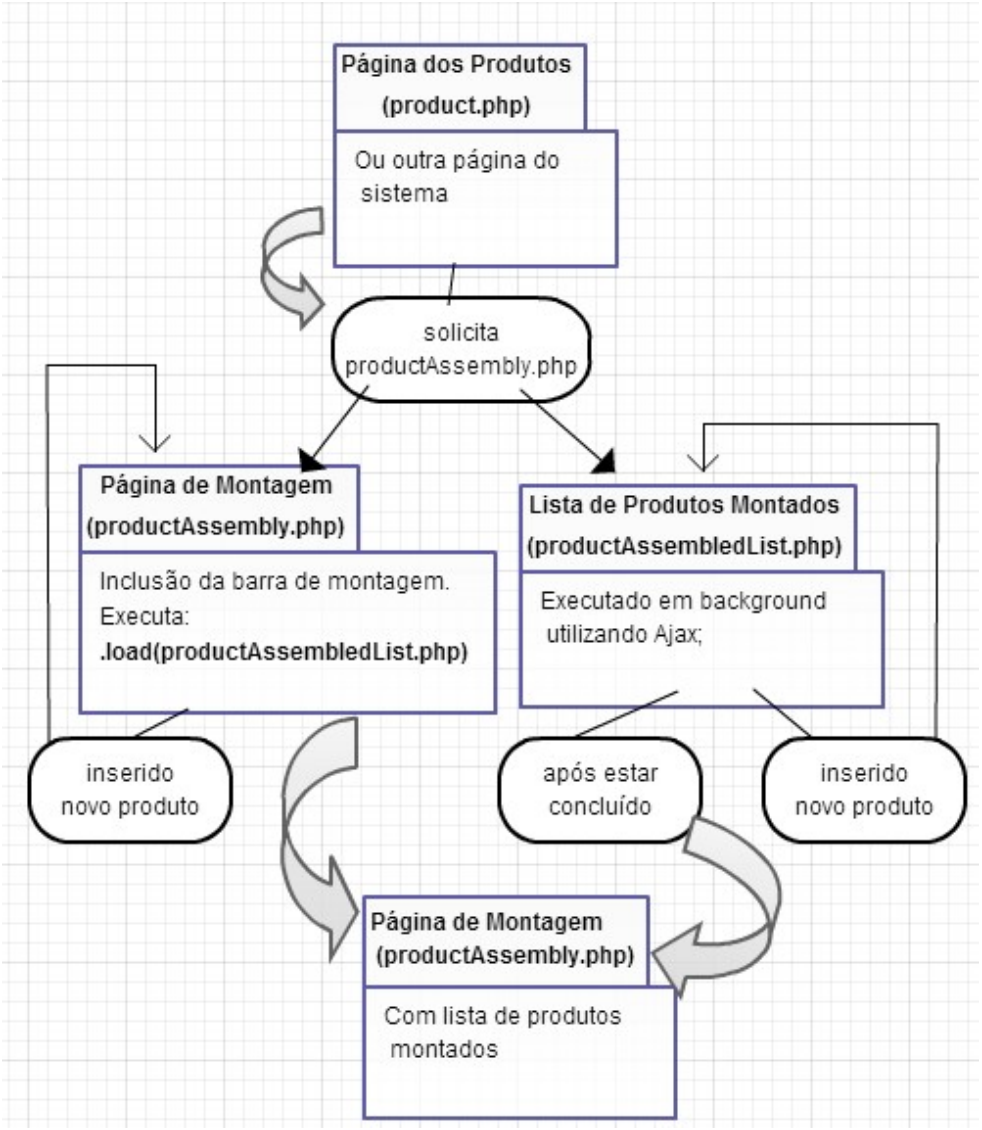

Figura 5.7: Produção da Página de Montagem

9. Introduzir Paginação no Histórico

Para o cumprimento deste requisito, foi necessário criar uma função na classe "TestReport", responsável por criar objetos que contêm todas as informações dos relatórios de testes realizados. A função criada foi "GetAllFromDB", ou seja, uma função que permite criar um vetor de objetos da classe "TestReport", de acordo com um conjunto de critérios. Para permitir uma seleção de resultados da execução da função, esta recebe, além de filtros de pesquisa (como o operador ou produto, que neste caso são parâmetros nulos, dado que o que interessa são os últimos relatórios efetuados), filtros para especificar o intervalo de resultados pretendidos. Estes parâmetros serão utilizados no operador "LIMIT" na *query* SQL executada, sendo responsáveis por definir esse

intervalo. Esta função recebe dois valores separados pelo caratere (“,”), por exemplo (x,y), em que o ‘x’ indica a posição inicial do registo a consultar e ‘y’ o número de registos a mostrar. Assim sendo, foi definido que se pretendiam visualizar 10 relatórios por cada página do histórico, pelo que na primeira página a função recebe (0,10), isto é, começa no registo 0 e mostra os 10 seguintes. Após serem solicitados os próximos resultados, a variável x passa para o valor que resulta da multiplicação do número da página pelo número de valores a apresentar, ou seja, (1x10) e o y continua a ser o número de relatórios a mostrar. Este processo repete-se assim sucessivamente, até ao último relatório presente no vetor de resultados da função.

Capítulo 6

Testes e Validações

Neste capítulo serão abordados os testes realizados ao sistema e os seus resultados. Mediante os resultados obtidos, os requisitos serão ou não validados.

1. Realizar Estatísticas de Falhas

Na página "Estatísticas" foi criada através de uma caixa de seleção, na qual estavam incluídas as opções de estatísticas disponíveis, tal como já foi abordado no capítulo da Implementação.

Para comprovar a funcionalidade de cada uma dessas opções, serão apresentados os conteúdos da BD que permitem obter as estatísticas pedidas e depois comprovar que a tabela os consegue reproduzir corretamente.

Para o primeiro caso, "Qual o produto com mais falhas?", podemos verificar, através da imagem seguinte, que existem 9 falhas na BD em que 5 pertencem à Firewall, 3 ao Comunicador GSM e 1 à Sirene.

productName	testFailName
Sirene	StepUp
Firewall	Buzzer_1
Firewall	Falha230AC
Firewall	Prog1
Firewall	Saida24_1
Firewall	Saida24_2
Comunicador GSM	Chamada
Comunicador GSM	Digital1
Comunicador GSM	Ring

Figura 6.1: BD- Falhas associadas aos Tipos de Produtos

Solicitando esta estatística, podemos analisar a tabela da figura 6.2, que como se pode comprovar, traduz os valores esperados.

Produto	Nr Falhas	% Falhas totais
Firewall	5	55.555555555556
Comunicador GSM	3	33.333333333333
Sirene	1	11.111111111111

Figura 6.2: Estatística da opção 1

Para a segunda opção, "Qual o teste em que ocorrem mais falhas?", é possível verificar na BD o conjunto de testes submetidos que contêm falhas.

testName	productName	total
Saída auxiliar de 24VDC	Firewall	2
Chamada de teste	Comunicador GSM	2
Falha 230AC	Firewall	1
Step-up (70V)	Sirene	1
Modo programação	Firewall	1
BUZZER	Firewall	1
Ring	Comunicador GSM	1
Entrada digital nº 1	Comunicador GSM	1

Figura 6.3: BD- Falhas associadas aos Testes

A tabela 6.4 produzida também o comprova.

Nome do Teste	Produto	Falhas totais	% Falhas totais
Sa&#65533;da auxiliar de 24VDC	Firewall	2	20
Chamada de teste	Comunicador GSM	2	20
Ring	Comunicador GSM	1	10
Entrada digital n&#65533; 1	Comunicador GSM	1	10
Falha 230AC	Firewall	1	10
Step-up (70V)	Sirene	1	10
Modo programa&#65533;&#65533;o	Firewall	1	10
BUZZER	Firewall	1	10

Figura 6.4: Estatística da opção 2

Na estatística "Qual o modelo que contém mais falhas?", analogamente à anterior, é verificado de entre os relatórios submetidos, quais os que contêm falhas e em seguida constatado qual o seu modelo de produto, como é possível observar:

productModelName	testFailName	total
Firewall 12	Prog1	5
GC-15	Chamada	3
GC-22	Digital1	1
Sirene Arc	StepUp	1

Figura 6.5: BD- Falhas associadas ao Modelo do Produto

Solicitando a tabela, estes dados são comprovados.

Produto	Nr Falhas	% Falhas totais
Firewall 12	5	50
GC-15	3	30
GC-22	1	10
Sirene Arc	1	10

Figura 6.6: Estatística da opção 3

Por último, na opção "Quais as falhas que ocorrem mais vezes?" é recolhido o número de vezes que cada falha é reportada nos relatórios já submetidos.

testFailName	total
Saida24_1	6
Chamada	1
Buzzer_1	1
StepUp	1
Ring	1

Figura 6.7: BD- Falhas mais recorrentes

Mais uma vez, é possível fazer corresponder os dados esperados à tabela.

Nome Falha	Nr Falhas	% Falhas totais
Saida24_1	6	60
Chamada	1	10
Buzzer_1	1	10
StepUp	1	10
Ring	1	10

Figura 6.8: Estatística da opção 4

Para validar definitivamente este requisito, carregando sobre o ícon do excel em qualquer das opções, é realizado o download de uma folha de cálculo com a respetiva tabela que pode ser armazenada pelo utilizador.

	A	B	C	D
1	**Nome do Teste**	**Produto**	**Falhas totais**	**% Falhas totais**
2	Chamada de teste	Comunicador GSM	2	20
3	Saída auxiliar de 24VDC	Firewall	2	20
4	Step-up (70V)	Sirene	1	10
5	Modo programação	Firewall	1	10
6	BUZZER	Firewall	1	10
7	Ring	Comunicador GSM	1	10
8	Entrada digital nº 1	Comunicador GSM	1	10
9	Falha 230AC	Firewall	1	10

Figura 6.9: Exemplo ficheiro Excel (Opção 2

Resultado:

2. Associar mais que um Componente a um Produto

Anteriormente no sistema, após o processo de montagem do produto, eram armazenados na tabela 'ProductItem' os dados relativos ao seu número de série e a apenas um extra que compunha o produto. Este extra estava definido, para os utilizadores, como fonte para a Firewall, módulo GSM para o comunicador e bateria para a Sirene, mas para o sistema não havia qualquer distinção entre eles, como se pode verificar na figura 6.10.

productItemId	productItemProductId	productItemSerialNumber	productItemCompositeSerialNumber
1	1	S1006[illegible]	EA97[illegible]
2	1	S10[illegible]	EA971[illegible]
3	1	S10060[illegible]	EA971[illegible]
4	1	S10060[illegible]	EA9719[illegible]
5	1	S10060[illegible]	EA8B0[illegible]
7	1	S1006[illegible]	EA9719[illegible]
8	1	S100[illegible]	EA9719[illegible]
9	1	S100[illegible]	EA8B08[illegible]

Figura 6.10: Anterior Associação dos Extras

Na nova implementação, foi criada a tabela 'Extra' que associa vários extras a cada produto.

Atualmente, existem 2 hipóteses de adicionar componentes extras a um dado produto. A primeira hipótese é através da barra da página de montagem.

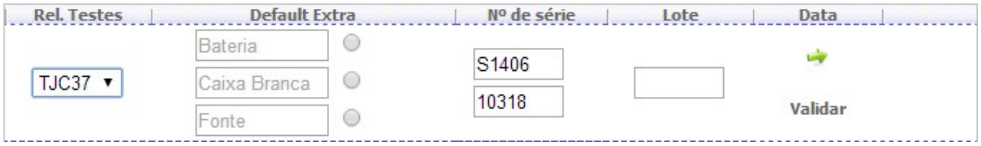

Figura 6.11: Formulário para Montagem de Produto

Por exemplo, no caso acima apresentado, o objetivo é montar o produto associado ao serviço 'TJC37', que se trata de uma Firewall 2K, havendo três possibilidades de extras a adicionar.

No primeiro caso, vamos adicionar a fonte e a caixa e definir a fonte como padrão a ser apresentado na lista de produtos montados.

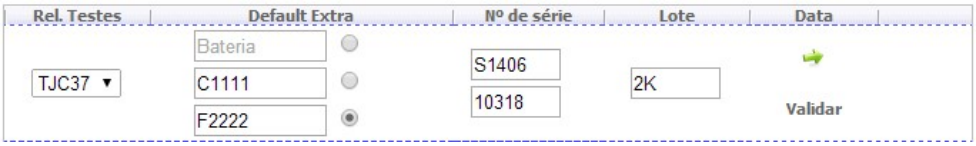

Figura 6.12: Exemplo de Montagem de Produtos com Extras

Depois de submetido o botão 'Validar', é preciso verificar se as informações ficaram corretamente armazenadas.

Assim sendo, verificamos que na lista de produtos montados aparece o produto devidamente identificado e com o padrão corretamente inserido.

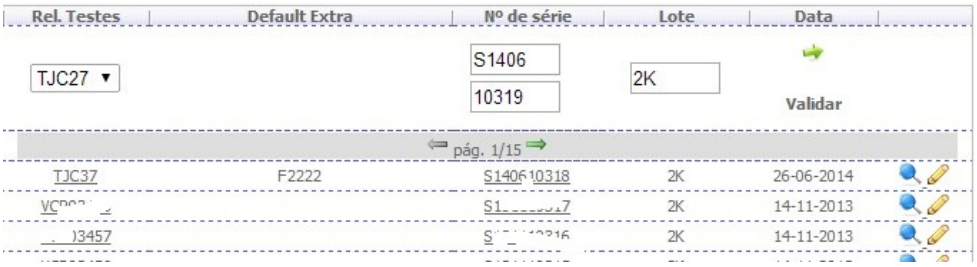

Figura 6.13: Lista de Produtos Montados com o Item inserido

Em seguida, selecionando a lupa que está presente na tabela ao nível da linha do item em questão, é possível verificar os detalhes do produto.

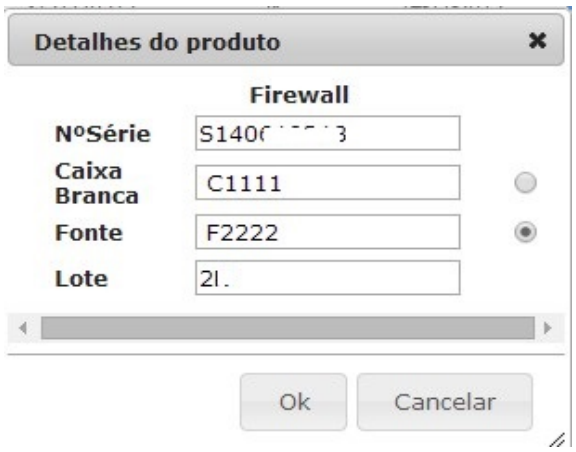

Figura 6.14: Detalhes do Item adicionado

Também através da BD, se pode confirmar a correta associação.

extraTypeName	extraIdentifier	extraDefault	productItemSerialNumber
Caixa Branca	C1111	0	S1406
Fonte	F2222	1	S140

Figura 6.15: Detalhes do Item adicionado na BD

O segundo método para adicionar componentes, consiste na seleção do lápis presente ao lado da lupa. Neste formulário, é apresentada uma caixa de seleção com os extras disponíveis e em seguida é pedido o número de série do extra. Ao mesmo produto, será adicionado o único que não foi adicionado antes, a Bateria, e depois confirmar-se-á que também foi bem associada.

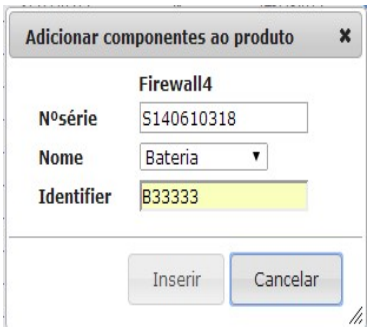

Figura 6.16: Adicionar Detalhe

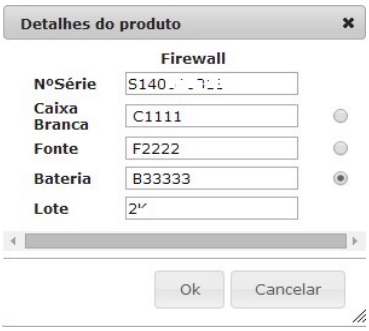

Figura 6.17: Confirmação Detalhe

É importante reparar que quando foram de novo solicitados os detalhes do produto, o extra padrão foi alterado para a bateria. Depois de selecionado o 'Ok', foi alterado o padrão, tal como se pode verificar na figura seguinte.

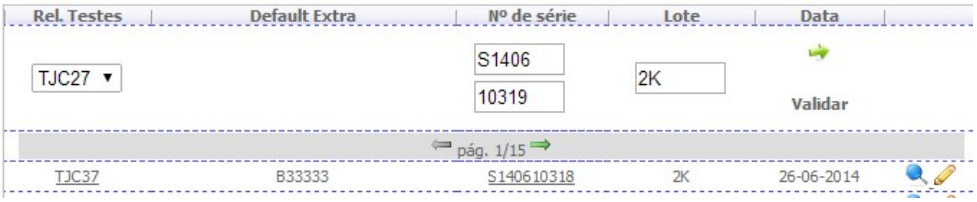

Figura 6.18: Alteração do Extra definido como Padrão

O mesmo se verificou ao nível da BD.

extraTypeName	extraIdentifier	extraDefault	productItemSerialNumber
Caixa Branca	C1111	0	S14061 1
Fonte	F2222	0	S1406
Bateria	B33333	1	S1406 1

Figura 6.19: BD- Extras associados ao Produto Montado

Resultado:

2.1. Definir Tipos de Componentes para cada Modelo

Para começar a validação deste requisito, foram inseridos diretamente na BD, 5 possíveis tipos de extras que poderão ser agregados aos produtos através da tabela 'ExtraType'. Em seguida, foram associados ao modelo do produto, através da tabela 'ProductModelExtraType'. A associação realizada foi a seguinte:

extraTypeId	extraTypeName	productModelName
1	Bateria	Firewall 2K
3	Caixa Branca	Firewall 2K
5	Fonte	Firewall 2K
1	Bateria	Firewall 12K
4	Caixa Preta	Firewall 12K
1	Bateria	GC-15
2	ModuloGSM	GC-15
3	Caixa Branca	Sirene Arc

Figura 6.20: BD- Tipos de Extras para cada Modelo do Produto

Depois foram realizados vários serviços a modelos diferentes, incluindo os que constam na lista acima apresentada.

testReportCode	testReportProductModelId	ProductModelName
TJC27	7	Firewall 12
TJC28	7	Firewall 12
TJC32	7	Firewall 12
TJC37	4	Firewall 2K
TJC22	8	Firewall 12K
TJC24	20	GC-15
TJC26	22	GC-20
TJC28	30	Sirene Arc

Figura 6.21: BD-Modelo do Produto de cada Serviço

Em seguida, será apresentado o modo de processamento da montagem de um produto. Através da barra que está presente na página de montagem, é selecionado o serviço que foi realizado para testar o produto que irá ser montado. Depois de selecionado, o sistema terá de verificar a que modelo de produto corresponde o serviço e de apresentar a possível lista de extras a juntar ao produto.

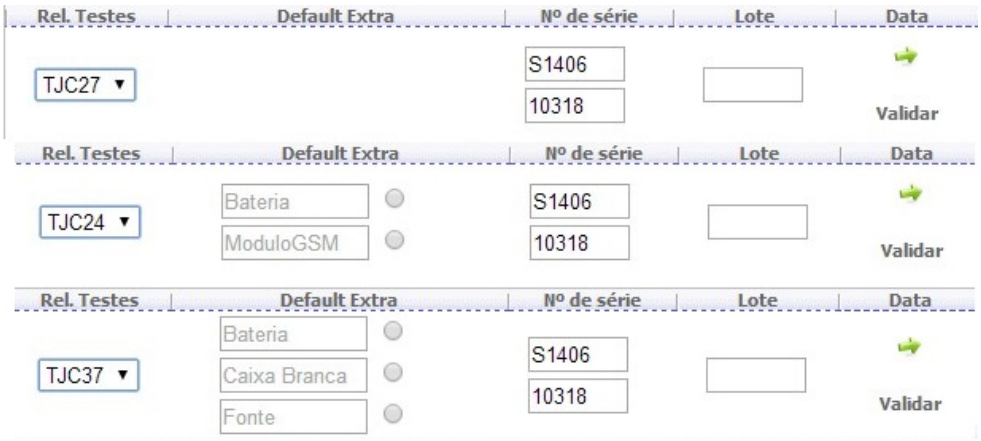

Figura 6.22: Extras a associar ao Modelo do Produto

Como é possível comprovar quando foi selecionado o serviço 'TJC27', que correspondia a uma Firewall 12 que não continha qualquer extra disponível, não é apresentado qualquer input no campo 'Default Extra'.

Por outro lado, é possível verificar que no caso do serviço 'TJC24', que incidia sobre um GC-15 e já continha 2 possíveis extras, são apresentados os 2 inputs. No último caso, é apresentado um modelo que contém 3 possíveis extras e também, tal e qual o segundo, aparecem os respetivos inputs. Para o produto ser considerado montado, basta que contenha um extra, ou seja, apenas um dos inputs seja preenchido como foi visto no teste do requisito 2.

Resultado:

3. Associar uma Lista de Falhas aos Testes

Como foi descrito nos testes a realizar para validar este requisito, começou-se por inserir várias falhas a vários produtos. Essas falhas foram inseridas utilizando o formulário que permite a adição de uma nova falha presente após selecionar o símbolo '+' na linha de cada teste. Um desses exemplos, que resultou da adição de uma falha no teste "Saída auxiliar de 24VDC" na categoria "Sem alimentação Externa" da Firewall está apresentado de seguida

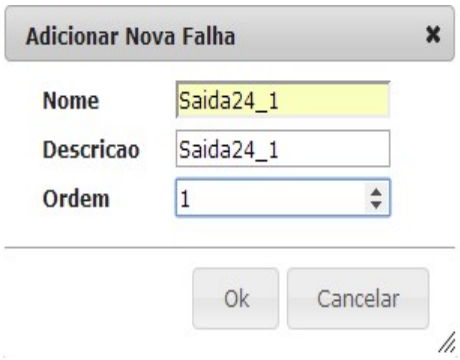

Figura 6.23: Exemplo de Adição de uma Falha

Depois de inseridas algumas falhas, como é possível observar na Figura 6.24, foi verificado na BD que as falhas inseridas têm a devida associação ao teste e ao produto que foram submetidos.

productName	testName	testFailName	testFailOrder
Firewall	BUZZER	Buzzer_1	1
Comunicador GSM	Chamada de teste	Chamada	1
Comunicador GSM	Entrada digital nº 1	Digital1	1
Firewall	Saída auxiliar de 24VDC	Saida24_1	1
Firewall	Saída auxiliar de 24VDC	Saida24_2	2

Figura 6.24: BD - Falhas inseridas

Em seguida deverá ser possível adicionar estas falhas nos próximos testes realizados de acordo com o produto em questão. Assim sendo, o exemplo que é apresentado na Figura 6.25 retrata um teste realizado a uma Firewall, posteriormente à adição das falhas e é esperado tal como se pode observar na Figura 6.24, que o teste "Saída auxiliar de 24VDC" tenha 2 falhas associadas e o teste "BUZZER" tenha 1 falha. Os restantes testes não têm possíveis falhas a adicionar.

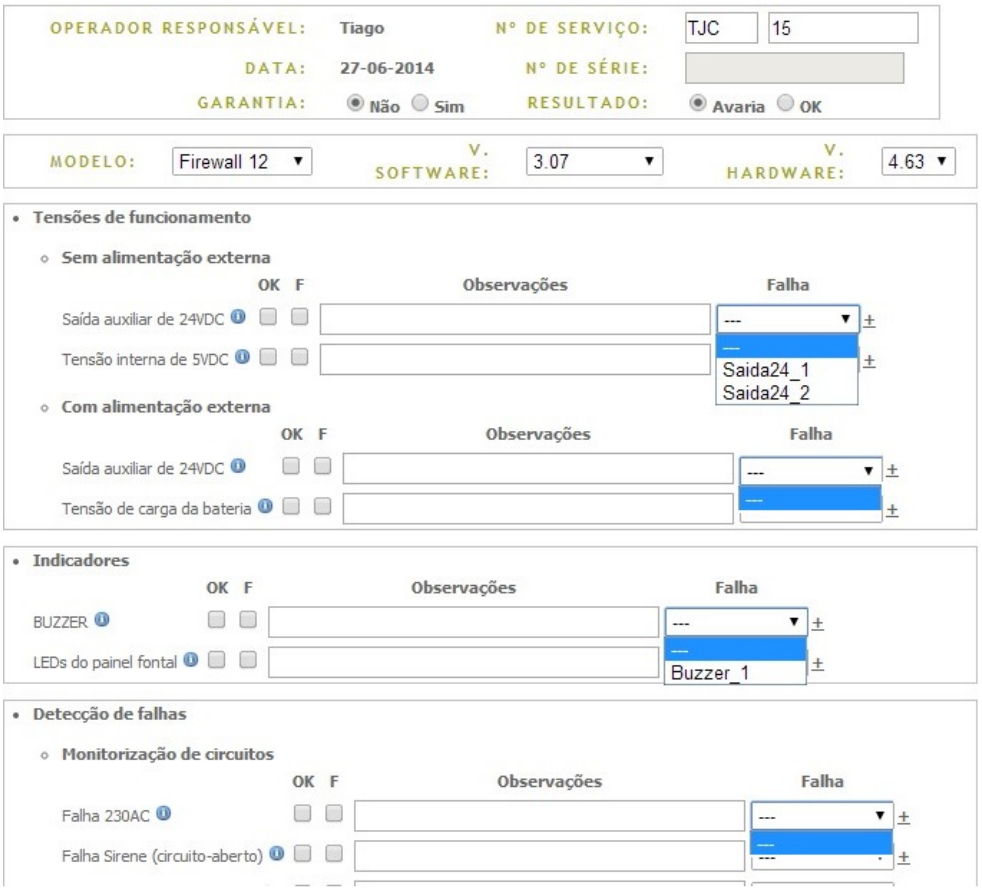

Figura 6.25: Página de Testes com Falhas existentes

O próximo passo é o armazenamento das falhas aquando a submissão do relatório. Utilizando o caso da figura 6.25, foi submetida a falha “Saída24_ 1” e a falha “Buzzer_ 1” no relatório do serviço ‘TJC15’. Na figura 6.26, irá ser apresentado a alteração da BD após submeter o mesmo.

testName	testReportCode	testResultFailId	testFailName
Saída auxiliar de 24VDC	TJC15	1	Saida24_1
Tensão interna de 5VDC	TJC15	NULL	NULL
Saída auxiliar de 24VDC	TJC15	NULL	NULL
Tensão de carga da bateria	TJC15	NULL	NULL
BUZZER	TJC15	4	Buzzer_1
LEDs do painel fontal	TJC15	NULL	NULL
Falha 230AC	TJC15	NULL	NULL
Falha Sirene (circuito-aberto)	TJC15	NULL	NULL
Falha Sirene (curto-circuito)	TJC15	NULL	NULL
Falha Terra (fuga a GND)	TJC15	NULL	NULL
Falha Terra (fuga a VCC)	TJC15	NULL	NULL
Repouso	TJC15	NULL	NULL
Fusível da saída auxiliar	TJC15	NULL	NULL

Figura 6.26: BD - Resultado do relatório efetuado

Ou seja, para os vários testes realizados no serviço 'TJC' apenas foram detetados erros nos testes "Saída auxiliar de 24VDC" e "BUZZER", como seria de esperar.

Para concluir os testes a esta implementação, foi solicitado o relatório do serviço 'TJC15' presente no histórico e foi verificado que as falhas que haviam ocorrido eram expostas automaticamente.

Figura 6.27: Revisão do relatório com as devidas falhas associadas

Resultado:

4. Adicionar Novos Tipos de Produtos através da Interface

Para a validação deste requisito, procedeu-se à inserção dos dados do produto (nome e imagem) na BD, utilizando para isso o formulário que surge em *popup*, através da seleção do *link* "Novo Produto", disponível no separador "Produto" da aplicação.

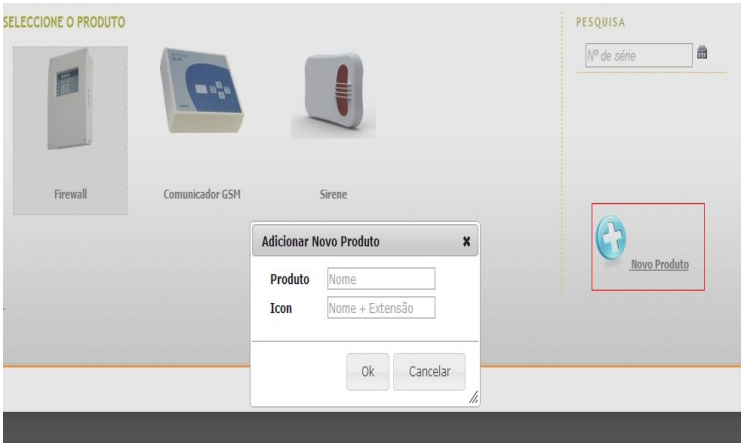

Figura 6.28: Formulário para adicionar novo Tipo de Produtos

Neste caso, o produto adicionado para teste vai ser uma luminária. Após ter sido colocada a imagem da luminária (luminária.jpg) na pasta das imagens no diretório do sistema, foram introduzidos os valores no formulário.

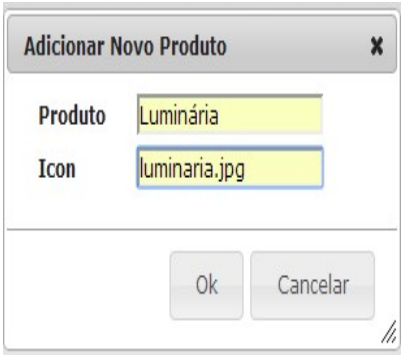

Figura 6.29: Adição de Novo Tipo de Produto

Depois de selecionar o botão 'Ok' é possível verificar que tanto na BD, como na página de produtos, já aparece a informação do novo produto e daí pode-se concluir que o produto foi inserido com sucesso.

Figura 6.30: Página 'Produtos' com Novo Tipo de Produto inserido

E na base de dados:

productId	productName	productImage
1	Firewall	firewall.png
2	Comunicador GSM	GC25-w125.png
3	Sirene	sirene.png
4	Luminária	luminaria.jpg

Figura 6.31: BD - Tipos de Produtos existentes

Resultado:

5. Adicionar Versões de Firmware(FW) e Hardware(HW) através da Interface

Através do formulário que é disponibilizado depois de clicar no link "Adicionar Novo HW SW" e selecionar as opções "Adicionar Nova Versão de Hardware" e "Adicionar Nova Versão de Software", foram inseridas as novas versões de Hw (5.01) e Sw(4.1) para a Firewall com o modelo 12K. Esses exemplos ficam expostos de seguida

Figura 6.32: Adição de Nova Versão de Hw

Figura 6.33: Adição de Nova Versão de Sw

Para saber se foram corretamente inseridas e se estão prontas a ser utilizadas, será inserido um relatório com as novas versões e confirmado se o mesmo ficou armazenado na BD com os valores das versões corretos.

Assim sendo, o relatório do serviço 'TJC21' foi submetido com os valores das versões de 5.01 para o Hw e 4.1 para o Sw.

Figura 6.34: Dados do relatório do serviço 'TJC21'

Como se pode comprovar é isso que acontece.

testReportCode	swversionmajor	swversionminor	hwversionmajor	hwversionminor
TJC21	4	1	5	01

Figura 6.35: BD- Confirmar Introdução das Versões

Resultado:

6. Realizar Pesquisa por Número de Serviço

Através do input abaixo apresentado, é possível realizar uma pesquisa por um determinado número de serviço. Para comprovar este fato, foi introduzido o número de serviço 'TJC37', como é possível verificar na Figura 6.36.

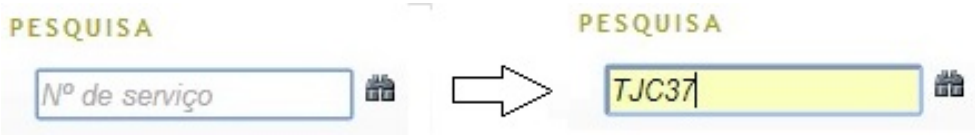

Figura 6.36: Pesquisa por Número de Serviço

Em seguida, é dada ordem para a pesquisa e é apresentada a informação do número de serviço pretendido.

Figura 6.37: Resultado da Pesquisa

Através do símbolo de informações que aparece na última coluna da tabela, é apresentado o relatório previamente realizado para o número de serviço inserido.

Figura 6.38: Revisão do Relatório Pesquisado

Resultado:

7. Introduzir Privilégios nos Utilizadores

Em primeiro lugar, definiu-se a variável 'operatorAdmin', da tabela 'Operator', que contém as informações sobre o utilizador, sendo que o valor '0' indica que o mesmo não é administrador, logo não terá acesso à página de administração, como é possível verificar na Fig.6.39. Foi realizada uma impressão do objeto 'Operator' que está autenticado no sistema e realçada a variável em questão. Também foi realçado, no menu, o espaço onde estaria disponível a página de administração, caso o utilizador fosse administrador.

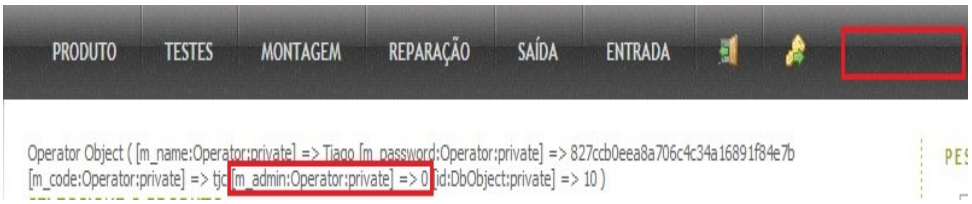

Figura 6.39: Menu de Utilizador: Técnico

Para o segundo teste, definiu-se a variável 'operatorAdmin' com o valor '1', valor esse que torna o utilizador num administrador. Assim sendo, já tem acesso à página de administração.

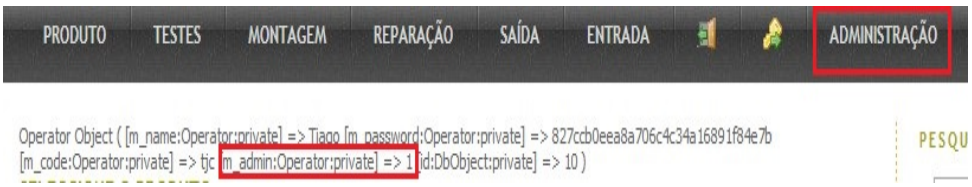

Figura 6.40: Menu de Utilizador: Administrador

Resultado:

8. Melhorar Tempo de Resposta do Sistema

Para testar este requisito foi utilizada a função "microtime()" que devolve o tempo que a página demora a ser processada. De referir que os testes foram efetuados com os dados presentes na BD até à data do inicio do trabalho mais os dados introduzidos para os testes, isto é, 9910 produtos montados e 10247 relatórios de teste. Deste modo, é possível estabelecer diretamente uma relação entre o tempo anterior e o novo tempo de processamento. Assim sendo, os testes começaram pela página dos testes aos produtos. Inicialmente, a página era apresentada em 5.6495 segundos, aproximadamente 6 segundos (Fig.6.41.). Depois das mudanças no sistema a página demora menos de 1 segundo a ser carregada (0.7206 segundos), isto significa uma redução na ordem dos 80% como é possível confirmar na Figura 6.42.

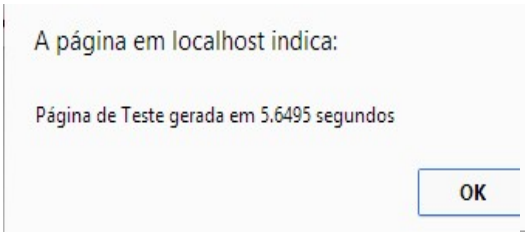

Figura 6.41: Tempo Anterior

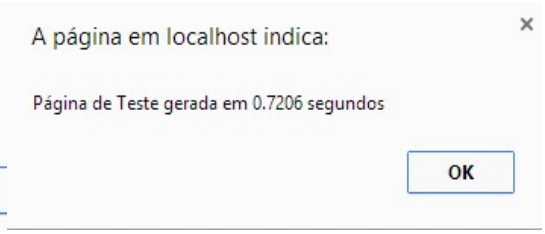

Figura 6.42: Tempo Atual

Também a página de Montagem era apresentada em 5.2879 segundos e na nova implementação é apresentada em 0.8488 segundos. Esta implementação permite uma redução de cerca de 85% do tempo gasto anteriormente.

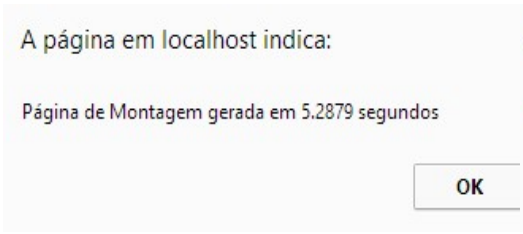

Figura 6.43: Tempo Anterior

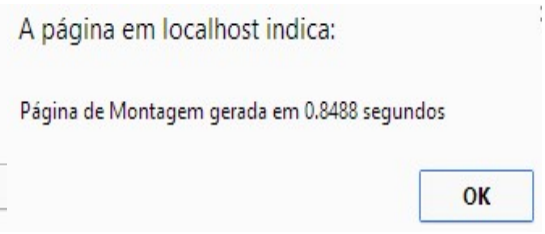

Figura 6.44: Tempo Atual

Resultado:

9. Introduzir Paginação no Histórico

Como foi descrito no requisito correspondente a este teste, no sistema anterior eram apresentados somente os últimos 10 serviços efetuados, como é possível comprovar em seguida.

Concluído este projeto, este histórico apresenta uma paginação para os últimos 100 serviços efetuados, em listas de 10 serviços por página. Em seguida, será apresentado o resultado dos índices 1 e 2 desta paginação e é possível comprovar que estão em lista de 10 e ainda que os 10 serviços do índice 2, são os 10 imediatamente anteriores aos que aparecem no índice 1 e assim sucessivamente.

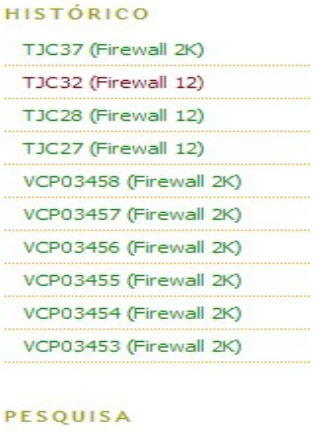

Figura 6.45: Anterior Histórico

Figura 6.46: Histórico Atual com Paginação

Resultado:

Capítulo 7

Conclusões e Trabalho Futuro

7.1 Conclusões

O projeto desenvolvido permitiu uma importante melhoria do SI da empresa Nibble, que beneficia assim de uma maior qualidade e rapidez de todo o processo de produção e reparação, efetuado na empresa.

É importante realçar a melhoria do tempo de resposta do SI, o que permite reduzir o tempo despendido na realização dos testes e na informatização das montagens dos produtos em cerca de 80%, aumentando assim a eficiência do processo de produção e reparação da empresa.

Também foi implementada a avaliação das falhas que ocorrem durante os testes, sendo possível uma mais rápida inserção das mesmas aquando o processo de testes e consequentemente uma mais eficaz atuação no processo de reparação. Estas podem ser posteriormente listadas, surgindo associadas ao produto e/ou teste em que ocorrem, e traduzidas sobre a forma de estatísticas, permitindo uma mais célere deteção de fontes de erros e respetiva correção.

A cada produto foi alargada a possibilidade de associação de componentes, o que aumenta a rastreabilidade dos produtos, e foi acrescentada uma lista de possíveis componentes a associar ao produto, aquando da inserção dos dados da montagem no SI.

Foi também implementada a diferenciação dos utilizadores do SI, para que apenas os administradores tenham acesso privilegiado às funcionalidades do sistema, como sejam a página de estatísticas e a possibilidade de adição de novos tipos de produtos e de novas versões de SW e HW (cuja execução foi simplificada também durante este projeto).

Foi ainda ampliado o histórico de serviços recentemente efetuados, que se traduzem numa lista com paginação de mais fácil acesso.

Houve também melhoria ao nível da pesquisa de informação, sendo introduzida a possibilidade de pesquisa de serviços através do número de serviço.

Fazendo então uma retrospetiva do que foi realizado, posso concluir que os objetivos inicialmente propostos foram cumpridos e que a Nibble, neste momento, tem um sistema mais qualificado e adequado ao seu método de trabalho.

Da minha parte, também houve uma evolução quer ao nível do conhecimento, quer de experiência, que adquiri durante estes meses de desenvolvimento do projeto e que serão muito importantes para o meu futuro profissional.

Esta foi uma experiência muito enriquecedora e só posso tirar dela impressões muito positivas, esperando, naturalmente, que a empresa também usufrua das implementações realizadas.

7.2 Trabalho Futuro

Durante o desenvolvimento deste projeto, foi realizada uma significativa melhoria do sistema de produção da empresa, no entanto, e durante o tempo dedicado a este, foram identificados ainda outros pontos, sobre os quais valeria a pena investir e, desta forma, aperfeiçoar, ainda mais, o sistema em questão.

Assim, e de forma a ficarem registadas sugestões de trabalho a desenvolver no futuro são, em seguida, descritas algumas tarefas nas quais seria importante atuar.

A página de testes apresenta já uma boa dinâmica, mas poderia ser acrescentada a capacidade de o administrador incluir, editar e remover dados na mesma, isto é, sem recorrer à BD, cujo manuseamento não é tão simples. Isto facilitaria a utilização da página de testes e permitiria a realização de um serviço mais rápido.

Também seria importante acrescentar a capacidade de, no sistema, alterar as versões de HW e SW previamente inseridas e gravadas, sem ter que recorrer à BD. Atualmente, só é possível inseri-las e, no caso da existência de alguma imprecisão nos dados acrescentados, não é permitido editar as informações presentes, pelo que tem de ser adicionada, de novo, a informação das versões.

Aquando da adição de extras dos produtos, seria igualmente importante haver uma sequência dos números de série dos extras a acrescentar, sendo relevante criá-la e pré-defini-la, de forma a acelerar todo o processo de montagem.

Referências

[1] Silva M. *JavaScript – Guia do Programador*. Novatec, 2010.

[2] W3C World Wide Web Consortium, Abril 2005. http://www.w3schools.com/ajax/ajax_example.asp,acedidoem23Maio2014.

[3] Nibble - Engenharia Lda, 2014. http://www.nibble.pt.

[4] Buschamn F.; Meunier R.; Rohnert H; Sommerland P e Stal M. *Pattern-Oriented Software Architecture, a System of Patterns*. Wiley, 1996.

[5] Lamin J. Mvc – o padrão de arquitectura de software. *Proceedings of the Third Conference on Object-Oriented Technologies and Systems*, Novembro 2010. Disponível em: http://www.oficinadanet.com.br,acedidoem23Maio2014.

[6] Torres J. Modelo relacional vs modelo orientado por objetos, 1997. Tese de Mestrado, Faculdade Engenharia da Universidade do Porto.

[7] Nibble Engenharia Lda. Manual de montagem do sistema de deteção de incêndios, Outubro 2006.

[8] Marques J. e Serrão C. *Programação com PHP 4*. FCA – Editora Informática, 2001.

[9] Niederauer J. *Web Interativa com Ajax e PHP*. Novatec, 2013.

[10] The PHP Group, 2014. http://www.php.net,acedidoem23Maio2014.

[11] jQuery UI Team, 2014. http://jqueryui.com/menu/,acedidoem23Maio2014.

[12] StackOverflow, 2014. http://www.stackoverflow.com,acedidoem23Maio2014.

[13] Moreira B. Central de alarmes com interface web, 2010. Tese de Mestrado, Faculdade Engenharia da Universidade do Porto.

[14] Luke Welling e Laura Thomson. *PHP e MySQL: Desenvolvimento Web*. CAMPUS, Terceira edição, 2005.