

Faculdade de Engenharia da Universidade do Porto



**Análise, Especificação e Desenvolvimento de
Aplicação Web para a Gestão dos Serviços
Auxiliares da FEUP**

Rui Manuel de Sousa Pena

Tese submetida no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Prof. Dr. António Miguel Pontes Pimenta Monteiro

Junho de 2008

A Dissertação intitulada

**“Análise, especificação e desenvolvimento de aplicação web para a gestão dos
serviços auxiliares da FEUP”**

foi aprovada em provas realizadas em 17/Julho/2008

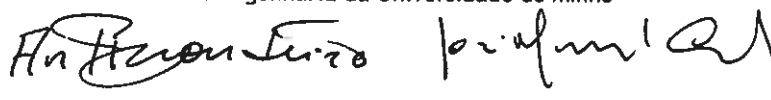
o júri

Presidente

Professor Doutor Rui Manuel Esteves Araújo
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto



Professor Doutor José Manuel Tavares Vieira Cabral
Professor Auxiliar da Escola de Engenharia da Universidade do Minho



Professor Doutor António Miguel Pontes Pimenta Monteiro
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto



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

Autor - Rui Manuel de Sousa Pena



Faculdade de Engenharia da Universidade do Porto

Aos meus Pais

"No fim tudo dá certo, se ainda não deu certo é porque ainda não chegou ao fim"

Fernando Sabino

Resumo

Este projecto teve por objectivo analisar e desenvolver uma aplicação Web para apoiar a gestão do trabalho na Unidade de Serviços Gerais da FEUP e resultou de uma iniciativa conjunta do Gabinete de Qualidade (QualiFEUP) e dos Serviços Técnicos e de Manutenção (STM). Até ao momento, os STM não dispunham de suporte informático para a gestão dos seus serviços auxiliares, a qual era efectuada manualmente. A existência de uma ferramenta que providencie um suporte para a organização e gestão de todo o trabalho revela-se, portanto, crucial.

Esta aplicação teve como base um protótipo previamente realizado, com o intuito de melhor compreender as necessidades do serviço. No entanto, o protótipo sofria de lacunas graves de implementação, não tendo sido utilizada uma arquitectura concisa na sua elaboração.

Inicialmente foi elaborada uma análise às necessidades do serviço em termos de grandes funcionalidades, e a partir destas foi planeada e desenvolvida uma aplicação que lhes desse suporte.

Especificou-se uma arquitectura sobre a qual é possível implementar a aplicação de um modo modular e escalável de fácil utilização e expansão futura. A arquitectura foi elaborada com o objectivo de se obter um padrão de desenho, tornando-se uma solução de implementação não só para esta aplicação mas para outras que se insiram no mesmo âmbito. Este padrão pretende ser mais conceptual do que prático, podendo ser modificado consoante as necessidades. Para tal, são apresentados os conceitos abstractos da sua lógica permitindo a sua reutilização e implementação.

Este trabalho concentra-se em dois módulos distintos da aplicação desenvolvida. Sobre cada um dos módulos é apresentada uma análise diferente, em concordância com as peculiaridades de cada um. O módulo Gestão de Tarefas é conceptualmente mais complicado, sendo utilizado para descrever a análise do desenho de interfaces, reutilização de código e a problemática de navegação. O módulo Gestão de Anomalias é tecnologicamente mais exigente, sendo, portanto utilizado como exemplo da implementação da arquitectura proposta e do uso das novas tecnologias introduzidas (Smarty e jQuery).

Abstract

This project, aimed to analyze and develop a Web application to support the work and data management in the Unit of General Services of FEUP, resulted from a joint initiative of the Office of Quality (QualiFEUP) and the Technical Services and *Maintenance* (STM). In the past, the STM lacked a computerized management system, as consequence, all management was done manually. The existence of a tool that supports the organization and management of all the work is, therefore, crucial.

This application was based on a yearly prototype conducted with the intent of better understanding the needs of the service. However, the prototype suffered from a faulty implementation resulting from a poorly developed architecture.

This report presents a structured analysis of the method used in the application planning (from the requirements to the implementation).

The specified architecture is used to implement a modular, scalable and user-friendly application prepared for future expansion. The architecture was established with the aim of achieving a standard design, becoming a solution to implement not only this application, but also others that fall within the same framework. This standard is intended to be more conceptual than practical and can be modified as necessary. So, the concepts of its logic are abstract, enabling them to be re used and implemented.

This work focuses two separate modules of the application. Accordingly with the peculiarities of each module, different analyses are presented. The tasks management module is theoretically more complicated. For that reason, it is used to describe the conceptual analysis of the interface design, code reuse and navigation issues. The anomalies management module is technologically more demanding, therefore, it is used as an example of the implementation for the proposed architecture and of the use Smarty and jQuery.

Agradecimentos

O meu profundo reconhecimento e agradecimento ao Professor António Miguel Pontes Pimenta Monteiro e ao Professor José António Rodrigues Pereira de Faria por toda a dedicação, disponibilidade e empenhamento com que acompanharam o meu trabalho.

A todos os meus amigos e colegas que sempre me apoiaram e “aturaram” o meu feitiço difícil, ao António João, Luís Rei, Rui Cachorreiro, Eduardo Sousa, Carlos Oliveira e em especial ao Eng. André Moreira pela ajuda na elaboração deste trabalho. Os amigos são para as ocasiões e vocês estiveram sempre lá.

Aos grandes amigos “Frickas” (Hugo Faria) e “Zapping” (Vitor Silva) pela amizade e bons momentos nas noites passadas em conjunto no desenvolvimento deste e outros trabalhos ao longo do nosso curso, a vossa boa disposição tornou trabalho em divertimento.

A minha namorada Alexandra Estanislau por todo amor, apoio, dedicação e compreensão e ajuda inestimável, sem ti tudo era mais difícil e sem ti todo o meu mundo era mais triste. Obrigado por me fazeres feliz.

A minha irmã Liliana Pena, pela força, garra e fé que tem em mim, sendo uma bênção constante na minha vida.

Em especial ao meu pai João Pena e a minha mãe Angelina Pena, a quem eu dedico este trabalho por serem os melhores pais do mundo e a quem devo tudo.

Índice

Capítulo 1	1
Introdução.....	1
1.1 - Motivação e Contexto	1
1.2 - Organização do documento	2
Capítulo 2	5
Análise Geral dos Requisitos da Aplicação	5
2.1 - Módulo Gestão de Avisos	6
2.2 - Módulo Gestão de Tarefas.....	6
2.3 - Módulo Gestão do Estacionamento	7
2.4 - Módulo Gestão de Anomalias	7
2.5 - Módulo Gestão de Utilizadores	8
2.6 - Casos de Uso	8
Capítulo 3	11
Arquitectura e Tecnologias	11
3.1 - Arquitectura Lógica	11
3.2 - Arquitectura Física	16
3.3 - Breve Referência às Tecnologias Utilizadas.....	17
Capítulo 4	27
Aprofundamento da análise do Módulo Gestão de Tarefas	27
4.1 - Estados de uma tarefa	27
4.2 - Esboços dos Diferentes Interfaces	32
4.3 - Análise detalhada dos formulários.....	38
Capítulo 5	41
Descrição da Análise e Implementação do Módulo Gestão de Anomalias	41
5.1 - Análise da Interface com utilizador	42
5.2 - Análise detalhada dos “ <i>Scripts</i> ” do módulo Gestão de Tarefas	46
5.3 - Detalhes de implementação	52
Capítulo 6	57
Conclusões e Perspectivas de Desenvolvimento	57
6.1 - Perspectivas de desenvolvimento.....	58
Referências	61
Anexo A.....	63

Anexo B.....	83
--------------	----

Lista de figuras

Figura 2.1 - Principais Módulos Identificados do Sistema.....	5
Figura 2.2 - Lista dos principais casos de uso associados aos módulos.	10
Figura 3.1 - Modelo da Arquitectura em camadas	12
Figura 3.2 - Arquitectura de camadas dividida em blocos funcionais.	13
Figura 3.3 - Como interagem os blocos na geração de uma página.....	15
Figura 3.4 - Arquitectura Física	16
Figura 3.5 - Organização dos ficheiros em servidor.....	17
Figura 3.6 - Exemplo simples jQuery	19
Figura 3.7 - Exemplo de manipulação do DOM com o jQuery	19
Figura 3.8 - Exemplo de código sem Smarty	20
Figura 3.9 - Exemplo de código com smarty	20
Figura 3.10 - Ferramenta de "debugging" do Smarty	21
Figura 3.11 - Interacção entre o PHP, o servidor Web e o navegador do cliente.	23
Figura 3.12 - Exemplo das definições dos estilos numa CSS.....	24
Figura 4.1 - Diagrama de Estados de uma tarefa.....	28
Figura 4.2 - Esboço dos formulários das tarefas e as suas ligações (navegação)	30
Figura 4.3 - Esboço do formulário "Lista Tarefas Hoje/Atrasadas"	32
Figura 4.4 - Esboço do formulário "Lista Tarefas Mês/Semana"	32
Figura 4.5 - Esboço do formulário "Lista Tarefas Periódicas"	33
Figura 4.6 - Esboço do formulário "Gestão Tarefas Periódicas"	33
Figura 4.7 - Esboço do formulário "Arquivo de Tarefas"	34
Figura 4.8 - Esboço do formulário "Imprimir Tarefas"	34

Figura 4.9 - Esboço do formulário Gestão Tarefas Únicas - Criar nova	35
Figura 4.10 - Esboço do formulário Gestão Tarefas Únicas - Consultar Detalhes - Estado Planeada.....	35
Figura 4.11 - Esboço do formulário Gestão Tarefas Únicas - Editar - Estado Planeada	36
Figura 4.12 - Esboço do formulário Gestão Tarefas Únicas - Registar (Executada).....	36
Figura 4.13 - Esboço do formulário Gestão Tarefas Únicas - Registar (Não executada/Parcialmente Executada)	37
Figura 4.14 - Esboço do formulário Gestão Tarefas Únicas - Consultar Detalhes - Estado Executada	37
Figura 4.15 - Esboço formulário Gestão Tarefas Únicas - Alterar - Estado Executada	38
Figura 4.16 - Divisão do formulário Gestão Tarefas Únicas em elementos.....	39
Figura 5.1 Esboço da estrutura do formulário tabela de anomalias.....	42
Figura 5.2 - Exemplo da primeira alternativa para inserção de anomalias	43
Figura 5.3 - Exemplo da segunda alternativa para inserção de anomalias.....	43
Figura 5.4 - Exemplo inserção de anomalia usando um "tooltip"	44
Figura 5.5 - Modelo da base de dados para suportar os formulários de inserção de anomalias.	46
Figura 5.6 - Matriz de atribuição das identidades às células do formulário.	51
Figura 5.7 - Formulário de Controlo de Anomalias - Sanitários	52
Figura 5.8 - Inserção de valores no formulário de Controlo de Anomalias - Sanitários	53
Figura 5.9 - Mecanismos de controlo de inserção de anomalias (divisões escondidas).	53
Figura 5.10 - Consultar a descrição completa de uma anomalia de inserção livre e/ou alterar.	54
Figura 5.11 - Inserção de um valor numérico no formulário.....	54
Figura 5.12 - Exemplo da facilidade de edição de um formulário.	55
Figura 5.13 - Exemplo da criação de um novo formulário com uma estrutura nova.	56
Figura 6.1 - Arquitectura com suporte para mobilidade	59

Lista de tabelas

Tabela 2.1 - Exemplo da especificação de um caso de uso.	10
Tabela 3.1 - Características do PostgreSQL.....	24
Tabela 4.1 - Lista de operações possíveis sobre uma tarefa no estado Planeada.....	28
Tabela 4.2 - Lista de operações possíveis sobre uma tarefa no estado Parcialmente Executada	28
Tabela 4.3 - Lista de operações possíveis sobre uma tarefa no estado Executada	29
Tabela 4.4 - Lista de operações possíveis sobre uma tarefa no estado Não Executada.....	29
Tabela 4.5 - Lista de operações possíveis sobre uma tarefa no estado Arquivada.....	29
Tabela 4.6 - Distribuição das operações pelos formulários identificados.....	31

Abreviaturas e Símbolos

Lista de abreviaturas

BD	Base de Dados
CSS	Cascading <i>Style</i> Sheet
DEEC	Departamento de Engenharia Electrotécnica e de Computadores
DOM	Document Object Model
DB Scripts	Data Base Scripts
FEUP	Faculdade de Engenharia da Universidade do Porto
HTML	Hyper Text Markup Language
LAN	Local Area Network
ODBC	Open Data Base Connectivity
PDA	Personal Digital Assistant
PHP	PHP: Hypertext Preprocessor
QualiFEUP	Gabinete de Qualidade da Faculdade de Engenharia da Universidade do Porto
STM	Serviços Técnicos e de Manutenção
SQL	Structured Query Language
WEB	World Wide Web
XML	Extensible Markup Language

Capítulo 1

Introdução

1.1 - Motivação e Contexto

O projecto aqui apresentado teve por objectivo a análise e o desenvolvimento de uma aplicação para apoiar a gestão do trabalho na Unidade de Serviços Gerais da FEUP e resultou de uma iniciativa conjunta do Gabinete de Qualidade (QualiFEUP) e dos Serviços Técnicos e de Manutenção (STM).

O QualiFEUP tem por missão apoiar os serviços centrais da FEUP, a melhorar a qualidade dos serviços que prestam á comunidade FEUP e melhorar a eficiência dos seus processos internos.

O STM é o serviço responsável pela gestão e manutenção dos edifícios, espaços, equipamentos e infra-estruturas da Escola, de modo a garantir as melhores condições para os processos de ensino e de investigação em engenharia.

Dentro dos STM, a Unidade de Serviços Gerais exerce a sua actividade no âmbito da gestão corrente das instalações, sendo responsável pela execução de múltiplas tarefas quotidianas indispensáveis ao bom funcionamento da Escola como, por exemplo, a abertura e fecho das salas de aula, substituição de consumíveis nas salas de aula e instalações sanitárias, registo e resolução de anomalias relativas ao mobiliário, equipamento didáctico, sistemas eléctricos, etc.

Esta unidade inclui cerca de 20 colaboradores cujo trabalho, até ao momento, era gerido sem qualquer suporte informático, o que constitui um factor de perda de eficiência devido às dificuldades de comunicação e de controlo do trabalho (planeamento e atribuição de tarefas e controlo da sua execução). Todos os registos eram feitos em papel onde havia alguém responsável pela organização e análise da informação. As tarefas eram atribuídas pessoalmente por vezes sem registos de execução ou de atribuição. Considerando um serviço tão abrangente por onde diariamente circula uma grande quantidade de informação e no qual se encontram envolvidos um grande número de colaboradores, a necessidade de uma ferramenta que ajude a organização e gestão do trabalho providenciando um suporte eficaz a toda organização é essencial.

Como é normal existe uma inércia natural à mudança e era necessário sensibilizar os futuros utilizadores da aplicação para as vantagens que esta podia trazer no seu trabalho diário. Isto sem criar uma atmosfera de desconfiança ao tentar forçar alterações aos métodos de trabalho existentes. É um problema delicado que o gabinete de qualidade da faculdade (QualiFEUP), teve de endereçar com cuidado e sensibilidade.

Tendo em conta a pouca experiência na área de informática de grande parte dos funcionários dos serviços auxiliares, numa fase anterior a este projecto, foi elaborado um protótipo da aplicação com o intuito de validar conceitos e mostrar as vantagens de utilização de um sistema deste tipo.

Esse primeiro protótipo, permitiu obter uma visão sobre as necessidades dos utilizadores e do próprio serviço assim como estabelecer um ponto de partida para uma aplicação definitiva. Nesse aspecto, o protótipo cumpriu os seus objectivos pois permitiu quebrar a primeira barreira na adaptação de um recurso informático na ajuda da gestão do serviço e clarificar algumas das funcionalidades básicas que deveriam, ser oferecidas. Como é natural, o protótipo sofria de falta de rigor e organização não tendo sido feito uma análise cuidada no seu desenvolvimento nem cuidados tomados na escrita do seu código. As múltiplas alterações a que este foi sendo sujeito, com o objectivo de se encontrar as soluções mais eficazes, sem uma arquitectura bem definida comprometeram de um modo irreversível a sua estabilidade e expansão.

Este projecto teve então por objectivo desenvolver uma nova versão da aplicação mais completa funcionalmente e sobretudo com uma melhor arquitectura, adicionar novas funcionalidades fornecendo assim aos serviços uma aplicação totalmente capaz. Foi portanto fundamental realizar uma análise detalhada dos requisitos e funcionalidades necessárias e desenvolver uma base sólida sobre a qual a aplicação possa crescer sem dificuldades.

1.2 - Organização do documento

Este documento encontra-se estruturado em 6 capítulos.

No primeiro capítulo, são apresentados os objectivos do projecto e o contexto em que foi desenvolvido, o que inclui uma breve referência ao Gabinete de qualidade da FEUP - QualiFEUP e dos Serviços Técnicos e de Manutenção da FEUP - STM. São também apresentados, os principais resultados que se pretendia alcançar.

O segundo capítulo contém a análise global dos requisitos do sistema e dos respectivos componentes, sobre os quais incide o projecto. Aqui são identificadas as grandes funcionalidades que a aplicação deve conter. Após identificadas as funcionalidades são especificados os principais casos de utilização.

O terceiro capítulo descreve, a arquitectura proposta para a aplicação e as principais tecnologias utilizadas assim como a razão da sua escolha.

No quarto capítulo é efectuada uma análise aprofundada sobre o módulo de Gestão de tarefas concentrando-se em aspectos mais conceptuais. Este confere uma melhor compreensão da especificação e desenvolvimento dos restantes módulos em termos de interfaces, reutilização de código e navegação. É exemplificado o método utilizado para analisar e detalhar os constituintes da aplicação.

No quinto capítulo é detalhado todo o processo levado a cabo desde o desenho à implementação do módulo de Gestão de Anomalias. Neste capítulo obtém-se uma melhor

visão sobre a arquitectura apresentada e sobre a sua aplicação prática. São também aqui apresentadas as vantagens em termos da flexibilidade conferidas por esta.

Por fim, no sexto capítulo, são enunciados os resultados do trabalho desenvolvido, identificando dificuldades encontradas, mais-valias obtidas e perspectivas de trabalho futuro.

Capítulo 2

Análise Geral dos Requisitos da Aplicação

Neste capítulo pretende-se dar uma perspectiva geral sobre toda a aplicação e dos seus principais constituintes. Foi para tal necessário fazer um levantamento global dos requisitos da aplicação que visam satisfazer as necessidades do serviço.

Os requisitos identificados estão agrupados em conjuntos lógicos representados na **Figura 2.1**, estes conjuntos formam os principais módulos que compõem o sistema e representam as grandes funcionalidades deste.

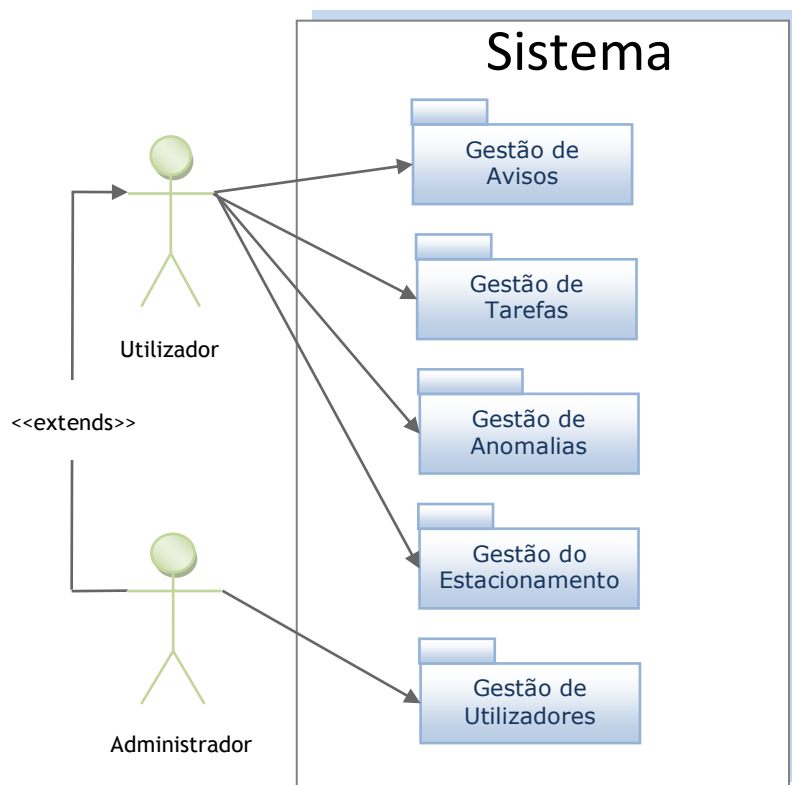


Figura 2.1 - Principais Módulos Identificados do Sistema

De seguida faz-se uma breve explicação dos módulos identificados e das suas características, especificando em termos gerais os principais pontos que estes devem satisfazer.

No final, para cada um dos módulos, serão apresentados os casos de uso identificados.

2.1 - Módulo Gestão de Avisos

Um serviço que envolve um grande número de pessoas, com as mais diversas funções, que se encontram dispersas por um espaço tão grande como é o campus da Faculdade de Engenharia necessitam de um meio integrado e único de comunicarem para evitar a repetição de informação, centralizar a sua consulta e facilitar a sua transmissão. O módulo de avisos tem como base de funcionamento uma espécie de quadro electrónico onde qualquer utilizador pode comunicar para todo o serviço informação relevante e ao mesmo tempo consultá-la.

Para satisfazer este objectivo os utilizadores têm de ser capazes de facilmente criar um aviso e de colocá-lo na aplicação, editar um aviso em caso de alguma lacuna ou erro, eliminar avisos que estejam obsoletos e serem capazes de consultar todos os avisos de um modo simples, intuitivo e ordenado.

Um aviso tem de ter uma breve descrição, que servirá como cabeçalho, um campo de observações, onde é possível detalhar com mais pormenor o intuito do aviso, o nome do autor e a data de criação. Para se obter uma melhor organização e consulta deve ser possível especificar se o aviso é ou não prioritário e para quem se destinam os avisos, caso não sejam de teor geral.

Um aviso poderá consistir, por exemplo, na comunicação de que foram abertas mais salas de estudo, devido à época de exames, e que não devem ser fechadas como é normal.

2.2 - Módulo Gestão de Tarefas

Do mesmo modo que é importante uma fácil comunicação de simples informação útil ao serviço, existe também uma grande importância em agendar e atribuir tarefas individualmente aos funcionários. Através de um módulo de gestão de tarefas será possível organizar o serviço a ser realizado e agendar o trabalho.

É necessário portanto um modo fácil para o responsável pelo serviço criar tarefas e atribuí-las aos funcionários e facilmente gerir o trabalho. Uma tarefa ao contrário de um aviso tem um propósito mais específico e a sua atribuição e gestão deve ser limitada aos responsáveis. No entanto a consulta deve ser possível a todos os utilizadores.

Por intermédio deste módulo um funcionário pode facilmente obter informação sobre o trabalho que lhe está atribuído bem como comunicar o estado de execução do mesmo. Cada funcionário pode ter uma ou mais tarefas atribuídas e um sistema de prioridades deve ajudá-lo a agendar o seu trabalho conforme o grau de importância da conclusão da tarefa. Logo uma tarefa deverá conter uma descrição do trabalho a realizar, um campo opcional de inserção de observações à execução da tarefa, data e hora para a realização da tarefa (podendo estes campos serem opcionais consoante o caso em questão), prioridade de execução e a quem se destina a tarefa.

Um exemplo de uma tarefa seria informar o funcionário “x” que devido ao evento que vai decorrer no auditório “y” este deve colocar uma mesa e duas cadeiras no primeiro piso do auditório e dois sinaleiros para o evento, um na entrada principal e outro nas escadas de acesso ao auditório.

O funcionário ao comunicar o estado de execução de uma tarefa deve poder inserir comentários sobre a boa (ou não) execução da tarefa. Seguindo o exemplo anterior, este comunicaria que não executou completamente a tarefa porque não existiam mesas disponíveis. Podia então colocar um aviso para alguém, mais tarde, rectificar esta situação.

Durante a execução de uma tarefa um funcionário pode encontrar uma anomalia. Esta anomalia pode ou não estar associada à correcta execução da tarefa em causa e pode ou não impedir a execução desta. Tendo isto em mente, deve existir a possibilidade de ao efectuar o registo de uma tarefa associar uma anomalia encontrada na execução da mesma.

As tarefas após concluídas devem poder ser arquivadas saindo do sistema geral, ficando apenas disponíveis para consulta pelos responsáveis.

Resumidamente, uma tarefa deve ser gerida (criar, alterar, apagar, arquivar) apenas por responsáveis, mas deve poder ser consultada por todos os utilizadores e principalmente por aquele a quem se destina. As tarefas devem ser dadas como concluídas ou seja existir um registo de execução. Ao registar deve ser possível inserir observações à execução da tarefa e associar avisos ou anomalias. Como fim do ciclo de vida de uma tarefa esta deve ser arquivada pelos responsáveis.

2.3 - Módulo Gestão do Estacionamento

Todos os dias um número imenso de utilizadores estaciona o seu carro nas instalações da Faculdade, existindo sérias dificuldades neste âmbito. Por vezes os condutores infringem as regras de estacionamento dificultando a utilização dos parques. Normalmente são avisados pelos vigilantes por intermédio de um autocolante amarelo que é colocado no pára-brisas. Por vezes esta medida mostra-se ineficaz em relativamente a condutores mais desrespeitadores, tornando-se então importante manter um registo das utilizações indevidas dos parques, com o intuito de tomar outras acções mais eficazes. Por exemplo um condutor encontrado em desrespeito “x” vezes veria o seu acesso ao parque negado por determinado tempo.

Sendo assim pretende-se uma fácil inserção de registos de estacionamento indevido, de modo a manter um histórico de fácil consulta de todos as infracções.

2.4 - Módulo Gestão de Anomalias

Os Serviços Gerais têm como função também a manutenção dos equipamentos e instalações da faculdade. Na eventual ocorrência de uma anomalia estes são os responsáveis pela sua resolução. Como tal, a capacidade de registar e gerir eventuais anomalias é indispensável a um sistema que vá apoiar o serviço.

As anomalias podem ser de vários tipos, eléctrica, mobiliário, sanitários etc. Uma anomalia é por exemplo “A sala “x” tem o estore avariado” ou “O sanitário “y” tem o autoclismo avariado” ou “No corredor “z” estão duas lâmpadas avariadas”. É importante manter estes registos de modo a uma fácil consulta, providenciando assim também uma

rápida resolução. Existe a necessidade do registo de anomalias ser efectuado em formulários tipo tabela oferecendo uma consulta mais intuitiva e concisa da informação, possibilitando ainda um cruzamento de dados entre as várias anomalias, facilitando a estimação do material danificado e do material necessário para a substituição. Existe um grande número de anomalias e tipos de formulários possíveis. Logo o módulo deve ser o mais dinâmico possível, permitindo constantes alterações, e suportando diversos tipos de formulários e de inserção de dados, consoante as necessidades inerentes a cada tipo de anomalia.

2.5 - Módulo Gestão de Utilizadores

A especificação anterior, em termos globais, dos níveis de utilizadores (**Figura 2.1**) foi uma aproximação simplista da gestão de permissões, na realidade vão existir diversos níveis intermédios entre um utilizador “básico” e um utilizador “administrador”. Uma das preocupações na aplicação é a implementação de um sistema de gestão dinâmico de permissões dos utilizadores. Pretende-se implementar uma solução flexível onde existem diversos papéis, permitindo especificar com detalhe as permissões associadas a cada papel.

A aplicação deverá conter uma lista de permissões associada a cada uma das acções que é possível efectuar sobre o sistema. Cada utilizador terá um tipo de acesso e cada acção sobre o sistema estará associada a um ou mais tipos de acesso. Esta lista será usada para filtrar a informação e as próprias interfaces que o utilizador tem acesso. Os tipos de acesso não são hierárquicos, isto é, as acções disponíveis para um tipo não devem ser totalmente disponíveis para outro. Pretende-se que estes sejam personalizáveis individualmente e que a qualquer altura se possa criar um novo tipo de acesso e livremente atribuir-lhe uma lista de acções à medida.

2.6 - Casos de Uso

Depois de, nos pontos supracitados, ter sido proporcionada uma perspectiva geral de cada módulo, são apresentados aqui os respectivos casos de uso identificados.

Foi elaborada uma listagem exaustiva dos principais casos de uso que o sistema deve oferecer aos utilizadores e uma descrição detalhada para cada caso de uso, sendo esta descrição uma importante referência para fases posteriores do projecto.

Módulos	Casos de Uso Associados		
Gestão de Avisos	Pesquisa Consultar Lista Avisos do dia Avisos anteriores	Detalhes Consultar Detalhes Alterar Aviso Apagar Aviso Adicionar Aviso Como Prioritário Destinado a Vigilantes	Imprimir
Gestão de Tarefas	Pesquisa Consultar Lista de Tarefas Dia Atrasadas Semana Mês Histórico	Detalhes Consultar Detalhes Tarefa Única (Instância) Periódica (Planeamento) Adicionar Tarefa Única Periódica Registar Execução da Tarefa Alterar Tarefa Planeamento Instância Planeamento Periódica Registo de Execução Apagar Tarefa Simples (Instância) Periódica Arquivar Tarefa	Imprimir Uma Tarefa Várias Tarefas
Gestão do Estacionamento	Pesquisa Consultar Lista Registos Recentes Histórico de Registos	Detalhes Consultar Detalhes Alterar Registo Apagar Registo Adicionar Registo	Imprimir Folha de Aviso

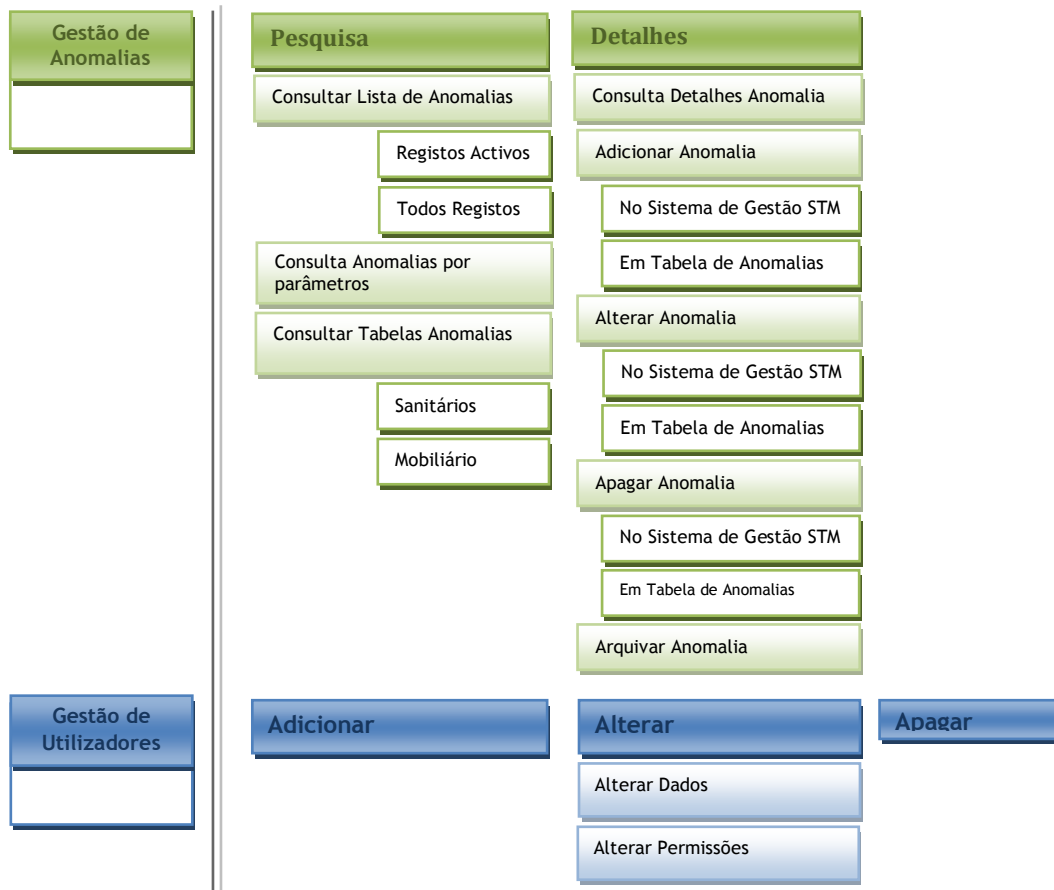


Figura 2.2 - Lista dos principais casos de uso associados aos módulos.

Para cada caso de uso identificado, foi elaborada a respectiva descrição. Devido ao grande número de casos de uso, aqui apenas será apresentado, a título de exemplo, a descrição de um caso de uso, podendo todos os outros ser consultados em anexo (Anexo A).

Tabela 2.1 - Exemplo da especificação de um caso de uso.

Caso de Uso
Consultar Lista de Tarefas do Dia e/ou Atrasadas
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema
Sequência de Acções
1. Selecção de “Salas” no formulário “Tarefas” 2. São apresentados dois formulários onde é possível consultar as tarefas. 2.1 O formulário “Hoje” lista as tarefas do próprio dia 2.1 O formulário “Atrasadas” lista as tarefas atrasadas
Nota
Por intermédio de botões nestes formulários (“Todas” e “Vigilantes”) é possível filtrar a informação entre todas as tarefas e as apenas destinadas a vigilantes.

Capítulo 3

Arquitectura e Tecnologias

Actualmente a arquitectura é reconhecidamente um dos pontos mais críticos no desenvolvimento de qualquer aplicação onde a especificação da estrutura e das suas interacções é essencial. A arquitectura desenvolvida teve como objectivo obter um padrão de desenho e uma solução de implementação não só para esta aplicação mas para outras que se encaixem no âmbito desta. Este padrão pretende ser mais conceptual do que prático podendo ser modificado consoante as necessidades, mantendo no entanto conceitos abstractos que podem ser reutilizados e implementados.

3.1 - Arquitectura Lógica

A arquitectura lógica de uma aplicação descreve como os componentes lógicos de uma solução se integram e como estão organizados.

Um dos grandes desafios no desenvolvimento foi elaborar uma arquitectura capaz de atingir os seguintes objectivos:

- **Simples:** Tendo em mente o tipo de utilizadores a que se destina aplicação é de extrema importância conceber uma aplicação e as suas interfaces o mais intuitivas possível.
- **Flexível:** Ser possível facilmente ajustar ou alterar a configuração da aplicação e ao mesmo tempo responder com rapidez e qualidade.
- **Eficaz:** A aplicação deverá ter uma grande fiabilidade operacional mantendo o nível de confiança dos utilizadores elevado, proporcionando-lhes um ambiente de trabalho estável e que responda de um modo uniforme às suas necessidades.
- **Escalável:** Um factor importante em termos de desenvolvimento da aplicação é esta ser capaz de evoluir de uma forma fácil e estruturada. Deve ser simples e transparente, com o mínimo de esforço por parte de quem implementa as alterações mantendo ao mesmo tempo uniformidade e coerência.
- **Gestão das permissões:** Como ponto-chave, a arquitectura desenvolvida deve suportar a gestão fina de permissões discutida no capítulo anterior. Para tal deve

permitir um controlo dinâmico e ao mesmo tempo simples, isto é, evitar que o código se torne demasiado denso e intrincado.

- **Reutilização do código:** A reutilização de código é sempre uma boa política em termos de implementação. Escrever menos código torna mais rápido e eficiente o desenvolvimento, transmitindo uniformidade ao mesmo, acabando mesmo por se reflectir na experiência do utilizador final.

Estes requisitos tornaram o desenvolvimento um desafio interessante, para tal é necessário uma arquitectura sólida e estruturada como base a seguir.

A arquitectura da solução proposta está dividida em camadas, esta organização é a chave para a independência entre os componentes e é por intermédio desta independência que é possível atingir os objectivos antes mencionados. O seccionamento da aplicação em camadas funcionais torna mais fácil a compreensão e modificação de cada unidade em particular.



Figura 3.1 - Modelo da Arquitectura em camadas

- **Camada de apresentação ou visualização:** Esta é a camada de interface com o utilizador, recebe os dados inseridos por este e apresenta o resultado às suas acções, apenas exhibe a informação não tendo preocupação em como a informação foi obtida nem de onde foi obtida.
- **Camada de controlo:** Esta camada controla o fluxo de informação. É o intermediário entre as camadas adjacentes e controla, segundo as regras de negócio, a informação passada entre elas.
- **Camada de acesso a base de dados:** Esta camada preocupa-se apenas com o armazenamento, manipulação e geração dos dados. É responsável pelos acessos a base de dados.

Dentro de cada camada foi feita uma subdivisão em blocos, aumentando, desta forma, o nível de detalhe da arquitectura implementada. A este nível está-se mais perto da especificação de implementação e menos dos conceitos conceptuais. Os blocos funcionais apresentados vão-se traduzir em ficheiros e *scripts* que compõem a estrutura da aplicação.

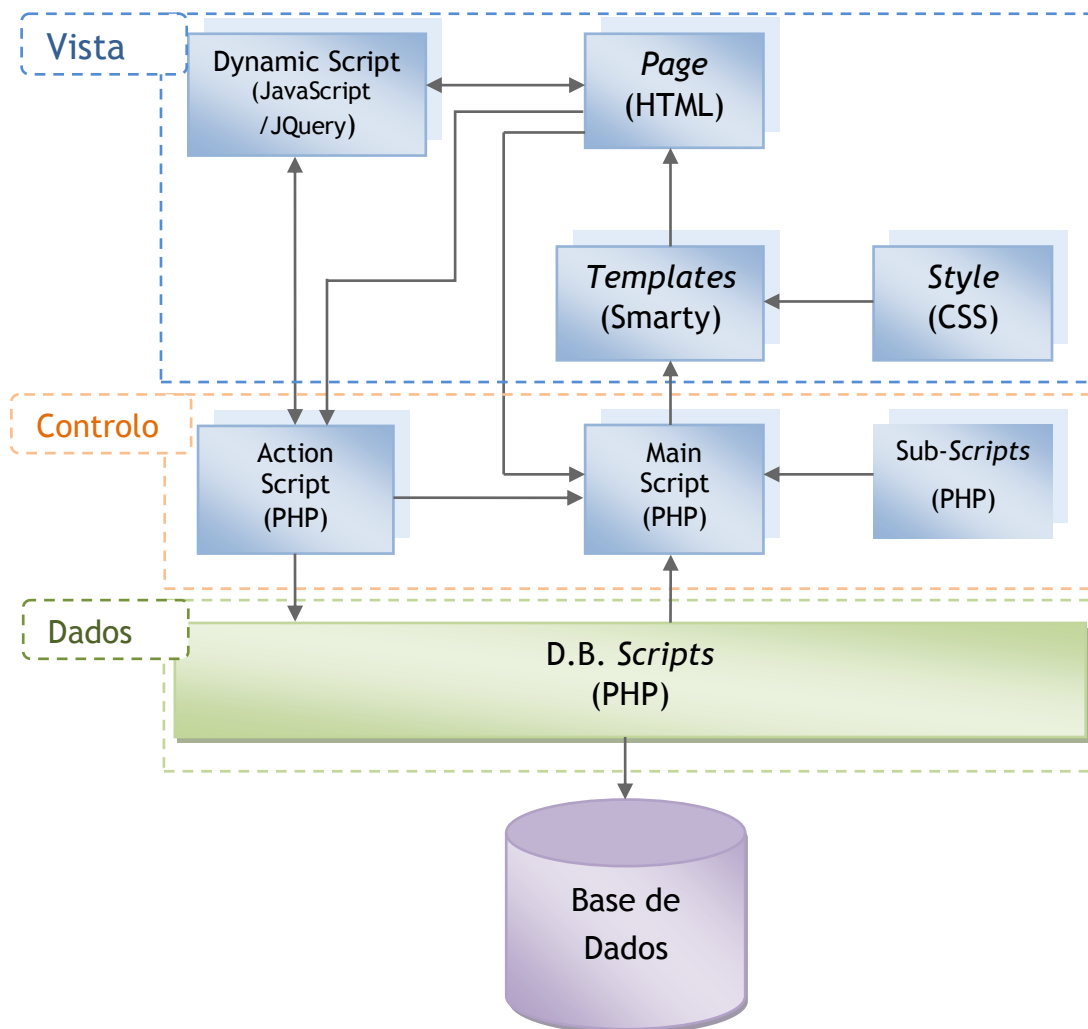


Figura 3.2 - Arquitectura de camadas dividida em blocos funcionais.

De seguida apresenta-se uma explicação de cada um dos blocos, desde o seu funcionamento á interligação com os restantes elementos que constituem em primeiro lugar a camada e depois numa visão de conjunto a arquitectura em si.

- **Main Script:**

Este bloco representa os *scripts* principais da aplicação responsáveis por toda a gestão e organização, obtendo, manipulando e estruturando a informação necessária para gerar a interface final a que o utilizador tem acesso. Neste script é implementado o controlo de permissões activado quando o utilizador vai “criar” uma página final diferente, conferindo a este mais ou menos opções de controlo consoante o caso.

- **Sub-Scripts:**

Este bloco representa *scripts* de inicialização e outros pequenos *scripts* repetitivos que são reutilizados por vários “*Main Scripts*”, por exemplo rotinas de manutenção, verificações de autentificação, etc.

- **Action Scripts:**

Este bloco representa os *scripts* associados a acções sobre o sistema, normalmente de actualização, remoção ou adição de informação da base de dados; estas acções podem ser requeridas por acções do utilizador sobre a interface ou mecanismos internos da própria aplicação.

- **Templates:**

Este bloco representa os *templates* usados pelos “*Main Scripts*” para construir a página a ser visualizada pelo utilizador. Uma página pode ser constituída por diversas partes, muitas dessas partes podem-se repetir noutras páginas. A utilização dos *templates* facilita o processo criativo tornando-o mais rápido e fiável. Reutilizar é sempre melhor do que estar a repetir todo processo de criação de raiz. Além disso, obtêm-se benefícios tanto a nível de organização como sobre o processo de edição. Por exemplo, para alterar o cabeçalho de toda aplicação basta editar um único ficheiro.

O processo de seccionamento das páginas pode requerer uma análise cuidada. É importante, antes de avançar no processo de criação, analisar os formulários existentes e identificar partes comuns que podem ser generalizadas em *templates*.

- **Style:**

Este bloco representa as folhas de estilo que podem ser atribuídas aos *templates*, definindo o aspecto em geral das interfaces.

- **Dynamic Scripts:**

Este bloco representa os ficheiros javascript/jquery associados à página, responsáveis pelos efeitos dinâmicos da página (animações e efeitos), permitindo uma interactividade mais alargada com o utilizador e tornando a experiência de utilização mais satisfatória. São também estes que permitem a interacção com a Base de dados de um modo assíncrono por intermédio de Ajax, evitando o refrescamento da página quando for conveniente.

- **Page:**

Este bloco representa a página propriamente dita, será este o elemento visível pelo utilizador final e é através deste que se efectuará a sua interacção com a aplicação. Ao contrário dos outros elementos este não é fisicamente representado num ficheiro mas sim o resultado da junção de outros blocos.

- **D.B. Scripts:**

Este bloco representa os diversos *scripts* desenvolvidos responsáveis pelos acessos à base de dados. Cada script é associado a um módulo específico e dentro deste estão as funções responsáveis pela ligação à base de dados para esse módulo.

A Figura 3.3 apresenta conceptualmente como interactivam os vários elementos para gerar uma página.

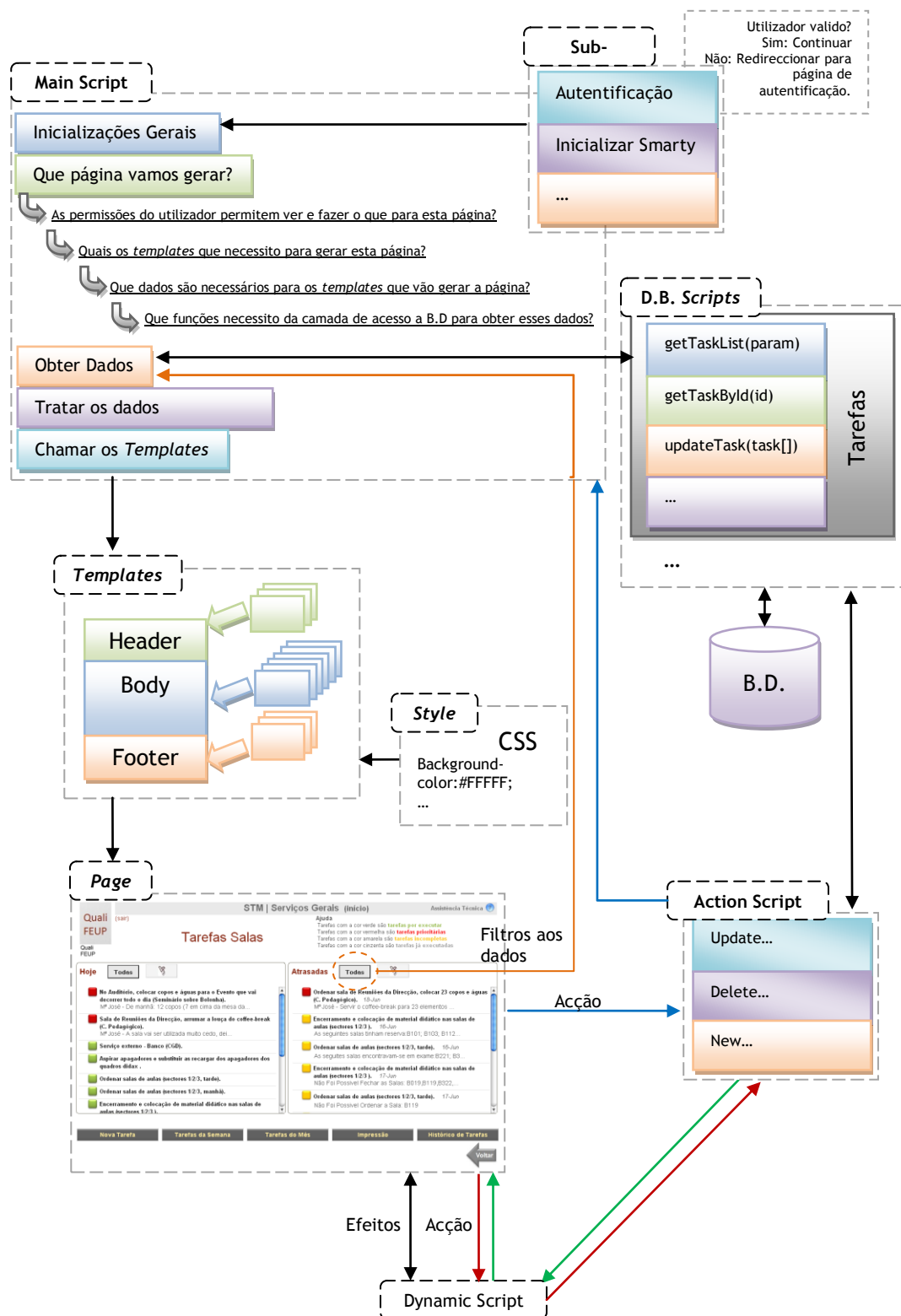


Figura 3.3 - Como interagem os blocos na geração de uma página.

O "Main Script" efectua em primeiro lugar as inicializações gerais necessárias para gerar uma página da aplicação. Estas inicializações sendo gerais e idênticas a todas as páginas estão guardadas em pequenos *scripts* designados por "Sub-Scripts". Em seguida é necessário

saber que tipo de página se vai gerar, que *templates* são necessários para gerar essa página, que dados necessitam esses *templates* e que “D.B. Scripts” necessita para obter esses dados. É então incluído o “D.B. Script” necessário que contém as funções de acesso à base de dados responsáveis pelo tratamento dos dados. Ainda no “Main script” os dados são manipulados conforme o que for necessário para os *templates*. Por fim, são chamados os “*templates*”. Sobre os *templates* são aplicadas as folhas de estilo “*Style*” atribuindo o aspecto e formatação. A geração da página é então concluída, sendo o resultado final aquele que o utilizador vai visualizar (“*Page*”). A página pode conter efeitos dinâmicos, isto é, alterações de aspecto sem o refrescamento da página, como por exemplo elementos que aparecerem e desaparecerem, alterações de cor, etc. Estes efeitos são atribuídos à página pelos “*Dynamic Scripts*”.

Sobre a página o utilizador pode efectuar acções. As acções podem ser das mais variadas, desde filtrar os dados a visualizar e até como mudar de página, ou acções que incidem directamente sobre a base de dados como criar, alterar, apagar, etc. Estas acções tanto podem ocorrer sincronamente como assincronamente. Acções síncronas envolvem o refrescamento da página. Por outro lado as acções assíncronas decorrem sem necessitar desse refrescamento, utilizando para isso os “*Dynamic Scripts*”. Como referido anteriormente existem vários tipos de acções, umas vão alterar as características de visualização e outras são acções de manipulação de informação. Por exemplo, o utilizador pode premir um botão de navegação (filtro) e sincronamente vai refrescar a página atribuindo novos parâmetros para a obtenção de dados do “Main Script”, como exemplificado na **Figura 3.3** em “Filtro aos dados”. Por outro lado quando se efectua, por exemplo, a submissão de um formulário esta acção vai efectuar uma operação sobre a base de dados. Para tal vai utilizar os “*Action Scripts*” que por sua vez utilizam “D.B. Scripts” para aceder à base de dados. Dependendo se foram chamados directamente pela página (sincronamente) ou pelos “*Dynamic Scripts*” (assincronamente) vão voltar ao “Main script” refrescando a página, ou vão responder aos “*Dynamic Scripts*” que actualizam página.

Em qualquer altura, se o utilizador sair da página navegando para outra, este processo será repetido, ou seja, vai ser invocado um outro “Main Script” responsável por essa página.

3.2 - Arquitectura Física

Em termos de arquitectura física temos um modelo genérico de uma aplicação Web cliente-servidor.

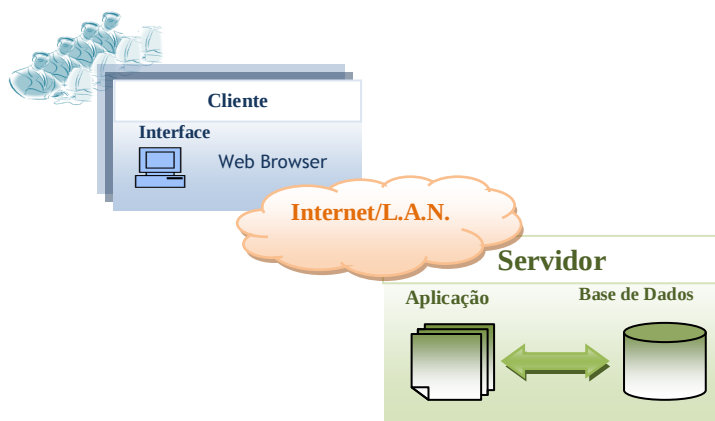


Figura 3.4 - Arquitectura Física

A aplicação encontra-se no mesmo servidor que o da base de dados, embora isso não seja estritamente necessário. Os utilizadores do sistema interagem com a aplicação por intermédio de uma interface Web apresentada num navegador numa ou mais máquinas onde o canal de comunicação é a internet ou rede local.

Um outro factor relevante é a organização dos ficheiros no servidor, é importante que a organização seja um paralelo físico da arquitectura lógica (**Figura 3.5**). Com um elevado número de ficheiros torna-se indispensável mantê-los dispostos de um modo fácil de navegar e aceder.

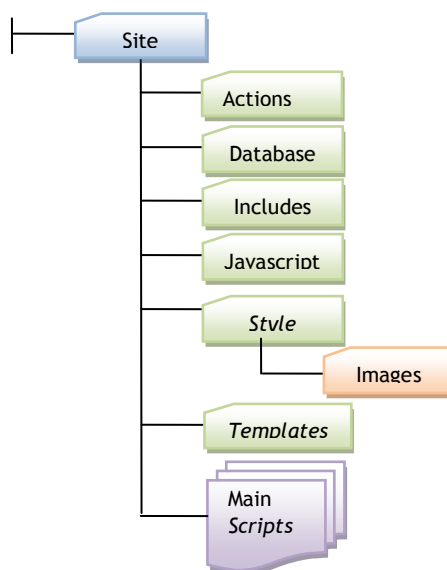


Figura 3.5 - Organização dos ficheiros em servidor.

Ao seccionar a aplicação em vários ficheiros, se não forem tomadas as devidas precauções rapidamente perde-se o controlo sobre a organização tornando-se extremamente complicado a sua manutenção futura. Tendo em conta a **Figura 3.5** pode fazer-se uma correspondência com os ficheiros pertencentes aos blocos da camada lógica. Os “*Main Scripts*” estão na raiz do site, os “*Action Scripts*” estão guardados na pasta “*Actions*”, os ficheiros da camada de acesso a base de dados “*D.B. Scripts*” vão estar guardados na pasta “*Database*”, os “*Sub-scripts*” na pasta “*Includes*”, os “*Dynamic Scripts*” na pasta Javascript, os ficheiros das folhas de estilo (CSS) na pasta “*Style*” e os “*templates*” na pasta “*Templates*”

3.3 - Breve Referência às Tecnologias Utilizadas

De seguida faz-se uma breve referência às tecnologias utilizadas no desenvolvimento deste trabalho, tentando-se evidenciar as razões que justificam tais escolhas enquadradas no âmbito deste projecto.

As escolhas tecnológicas tiveram como base não só as necessidades do projecto mas também a experiência passada, associada ainda às necessidades inerentes ao gabinete QualiFeup. Pretende-se manter uma certa uniformidade nas tecnologias utilizadas nos projectos do gabinete, por razões óbvias de continuidade e manutenção.

- **Jquery**

O Javascript durante muitos anos foi considerada uma linguagem de programação pouco séria e fiável devido a diferenças de desenvolvimento entre navegadores e ao seu uso em aplicações mal intencionadas que infelizmente abundam na internet. Nesta nova era do Ajax e da Web 2.0 tem-se sentido um renovado interesse dando origem a páginas cada vez mais interactivas e dinâmicas.

Para ajudar ao desenvolvimento nesta linguagem têm aparecido diversas “frameworks”/bibliotecas de javascript, todas com o mesmo objectivo: aumentar a simplicidade e a produtividade. Existem actualmente vários projectos de código aberto neste sentido. Seria, no entanto, fora do âmbito deste trabalho uma comparação exaustiva de todas.

Foi, portanto, necessário optar por uma e a escolha recaiu sobre o jQuery. É uma biblioteca com excelentes referências na internet com uma grande comunidade de programadores e utilizadores, cujas funcionalidades oferecidas se enquadram perfeitamente nas necessidades da aplicação.

A biblioteca jQuery é capaz de simplificar a manipulação dos vários elementos de uma página Web e facilmente percorrer e endereçar estes elementos do Description Object Model (DOM). Qualquer destes elementos pode ser seleccionado usando qualquer selector CSS 1 a 3 (mesmo aqueles que os navegadores não suportam), assim como qualquer designação de XPath - XML Path Language - que torna o DOM um livro aberto e melhor ainda, bem escrito.

A biblioteca encontra-se contida em apenas um único pequeno ficheiro (cerca de 14KB na versão comprimida) e contém uma vasta documentação disponível na internet. Permite a adição de funcionalidades por intermédio de módulos, que se encontram disponíveis, bem documentados e com grande suporte, tanto pela equipa que desenvolveu o jQuery como pelos seus utilizadores. De facto, esta extensão de funcionalidades é tão versátil que rapidamente um utilizador cria os seus próprios módulos de modo a poder reutilizá-los em projectos diferentes.

Com o jQuery é rápido e fácil enriquecer qualquer aplicação Web com efeitos, estilos, animações e a utilização de Ajax. Este tipo de programação em javascript requereria um código extenso, complexo e difícil de manter. Com jQuery são necessárias poucas linhas de código para obter as mesmas funcionalidades, deixando que o utilizador se concentre mais no processo de implementação.

“I started replacing older code in previous projects to see how jQuery would simplify the pages.” [1].

Uma das vantagens que vamos encontrar, imediatamente, na utilização do jQuery é no evento “*window.onload*”. Este é o tradicional “cavalo-de-batalha” da programação em javascript, utilizado pelos programadores para dar início às funcionalidades dinâmicas do lado do cliente logo que a página é completamente carregada. Mas, por vezes, esperar que a página seja completamente carregada torna-se incómodo, podendo existir diversas imagens que atrasam o processo. Podem também surgir problemas de incompatibilidades entre navegadores com este evento. Desta forma, o jQuery oferece uma solução que permite executar o código javascript assim que o DOM esteja pronto a ser manipulado, coisa que acontece normalmente antes de a página estar completamente carregada. A **Figura 3.6** ilustra a esta solução e o simples código necessário para a executar. Esta função pode ser utilizada as vezes que forem necessárias.

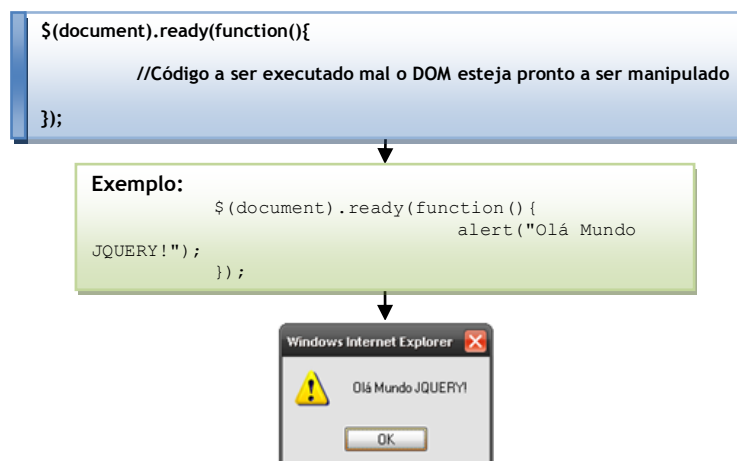


Figura 3.6 - Exemplo simples jQuery

O jQuery como foi referido anteriormente facilita a inserção de efeitos dinâmicos com pouco e simples código. A Figura 3.7 tenta ilustrar isso mesmo.

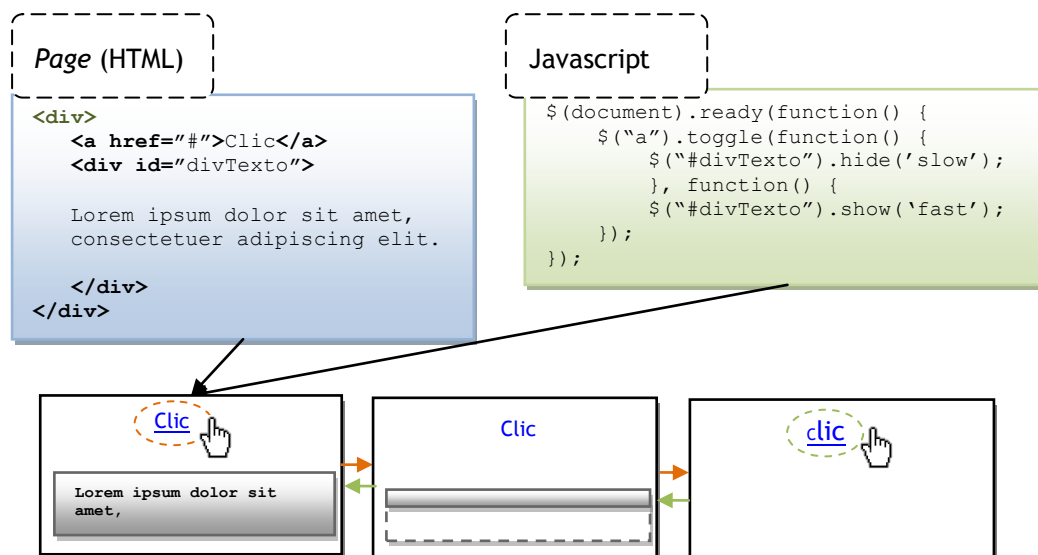


Figura 3.7 - Exemplo de manipulação do DOM com o jQuery

Esta função esconde ou mostra a divisão "divTexto" (`$("#divTexto")`) sempre que se clica numa hiperligação da página (`$("#a")`).

Para obtermos o objecto "divTexto" utilizamos `$("#divTexto")`, que é muito semelhante ao `document.getElementById("divTexto")`. No entanto, é muito mais simples e não gera problemas de compatibilidade entre navegadores.

Por intermédio do jQuery é possível adicionar funcionalidades à página sem introduzirmos javascript no código HTML. Esta é uma grande vantagem na utilização desta biblioteca, pois permite-nos interagir com os elementos do DOM de uma forma fácil e não intrusiva, tornando mais fáceis possíveis alterações no futuro.

Esta é umas das filosofias utilizadas no desenvolvimento desta aplicação: código simples, dinâmico, bem estruturado, de fácil manipulação, adaptação e adição.

- **Smarty**

Smarty é um motor de *templates* muito usado em aplicações Web, em especial para aplicações escritas em PHP. A principal função do Smarty é efectuar uma separação da lógica da aplicação e do seu aspecto. É uma ferramenta que permite poupar imenso tempo, quando usada adequadamente.

Quando se desenvolve uma aplicação é importante ter em mente a evolução, isto é, a aplicação deve ter capacidade de se adaptar consoante as necessidades. Normalmente, ao implementar uma aplicação Web, a primeira preocupação é responder aos requisitos impostos. Infelizmente, ao contrário da construção de uma ponte onde é pouco provável que se efectuem alterações, neste tipo de aplicações é muito frequente aparecerem novas funcionalidades e requisitos, ou simplesmente uma lista de problemas que é necessário resolver. Quando isto acontece, o código necessita de ser adaptado e em muitos casos completamente reestruturado e reescrito.

O conceito de modularidade é muitas vezes posto de parte na concepção de aplicações Web e a aplicação desenvolve-se sem a capacidade de acoplar alterações futuras. Até mesmo pequenas aplicações tornam-se praticamente impossíveis de modificar quando estas contêm informação gerada dinamicamente e se encontra código PHP misturado no meio de HTML.

```
<?PHP
echo "<table>";
echo "<th>Id</th><th>Nome</th><th>Morada</th><th>Código
Postal</th>";
foreach $utilizador as $elemento{
    echo "<td>".$elemento['id']. "</td>";
    echo "<td>".$elemento['nome']. "</td>";
    echo "<td>".$elemento['morada']. "</td>";
    echo "<td>".$elemento['codigo_postal']. "</td>";
}
echo "</table>";
?>
```

Figura 3.8 - Exemplo de código sem Smarty

Embora o código da **Figura 3.8** esteja bem estruturado e aparentemente bem legível, torna-se um processo difícil de adaptar e alterar quando é encontrado disperso no meio de uma página com código HTML com grandes quantidades de código dinâmico.

```
<?PHP
$smarty->assign('utilizador', $utilizador);
$smarty->display('utilizador.tpl');
?>
```

utilizador.tpl

```
{html_table loop=$utilizador cols="Id, Nome, Morada, Código Postal"}
```

Figura 3.9 - Exemplo de código com smarty

Ao separar o código responsável pela lógica (PHP) do código de design (HTML) é possível criar, alterar e apagar *templates*, mantendo toda a lógica intacta por trás.

O Smarty facilita a criação de *templates* bem estruturados, de fácil compreensão e reutilização. Um template criado em Smarty é simples HTML com alguma sintaxe própria, com a vantagem adicional de conter ferramentas poderosas para manipular os dados recebidos. É de simples aprendizagem e utilização, conferindo ainda uma redução no tempo de desenvolvimento da própria aplicação.

Um outro elemento muito útil da utilização de Smarty é a excelente ferramenta de “*debugging*” que este possui, permitindo facilmente encontrar erros e verificar o estado das variáveis em jogo.

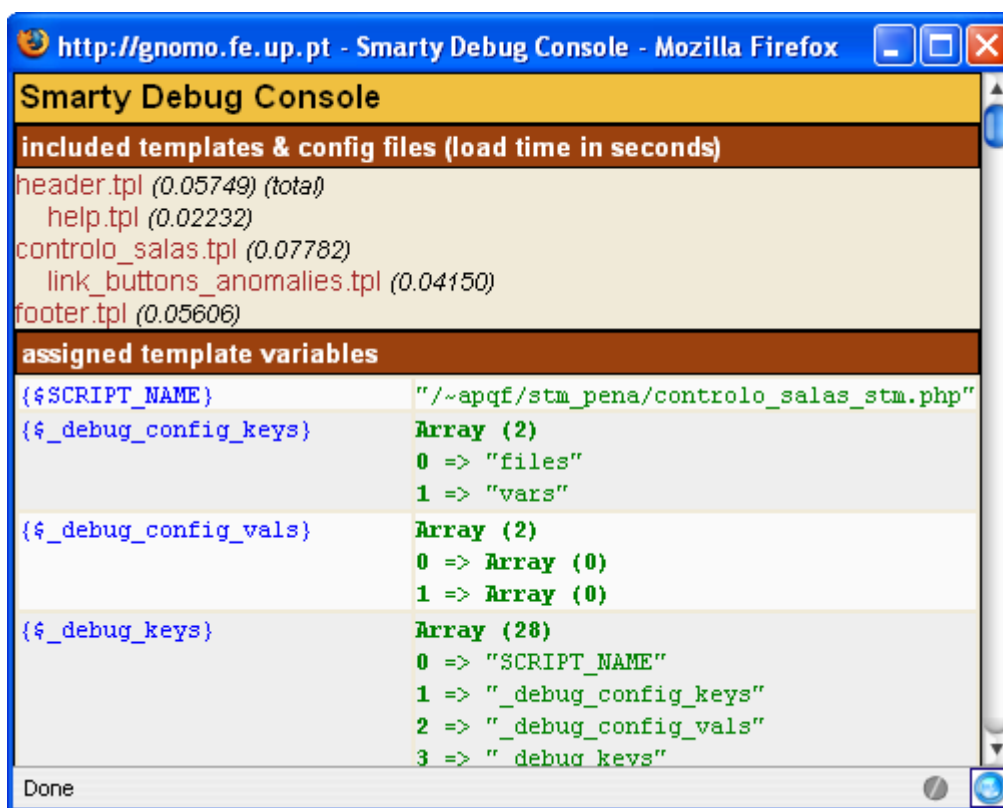


Figura 3.10 - Ferramenta de “debugging” do Smarty

- **Outras Tecnologias**

Pequena referência a outras tecnologias utilizadas no âmbito deste trabalho.

- **PHP**

Até algum tempo atrás, a Internet era composta basicamente de páginas estáticas, ou seja, páginas que dia após dia tinham o mesmo conteúdo, e que não interagiam com seus visitantes. Isto impedia que atitudes mais profissionais fossem tomadas nestes sites, pois sem a interactividade não era possível que empresas trocassem informações com seus clientes, com outras empresas e consigo mesma (troca de dados entre departamentos, entre a empresa e seus vendedores, etc.). Devido a estas limitações começaram a surgir algumas ferramentas que “deram vida” à Internet: “Perl”, “TCL” e outras linguagens padrão CGI (“Common Gateway Interface”).

Porém, essas linguagens eram limitadas quanto ao seu uso e complexas na sua utilização. No entanto, foram estas iniciativas que permitiram que outras linguagens fossem

desenvolvidas exclusivamente para Internet. Entre elas, podemos citar: PHP, ASP, Java, JavaScript, VBScript e outras, sendo que algumas destas linguagens são executadas no servidor e outras no cliente.

O PHP foi criado originalmente por *Rasmus Lerdorf* em meados de 1994 e escrito em “Perl”, sendo reescrito mais tarde em “C” para incluir acessos a bases de dados. Rapidamente, tornou-se popular e conhecido, propagando-se pela Internet. *Rasmus* disponibilizou alguma documentação do software e baptizou-o oficialmente de PHP v.1.0. Com a crescente utilização do PHP, mais e mais recursos foram incluídos (ciclos, “arrays”, suporte para objectos, etc.), tornando-se cada vez mais uma linguagem poderosa. Entretanto outros programadores juntaram-se a *Rasmus* contribuindo, de um modo valioso, para o aprimoramento da linguagem. A linguagem PHP, de um modo geral, é uma combinação de linguagem de programação e servidor de aplicações. O PHP destaca-se entre as linguagens, citadas anteriormente, por ser multi-plataforma, ou seja, suporta vários sistemas operativos, como Windows, Unix, Linux, etc. É uma linguagem de fácil aprendizagem e compreensão. Permite a conexão directa com uma grande quantidade de bases de dados relacionais, enquanto outras ferramentas precisam de drivers ODBC para realizar a mesma tarefa. Em relação às bases de dados em que é possível uma conexão directa podem-se referir: Oracle, Sybase, Informix, Postgresql, MySQL, mSQL, etc. Para outras bases de dados, o PHP disponibiliza acesso por intermédio de ODBC. Uma outra grande vantagem do PHP é o facto de este ser totalmente gratuito podendo ser facilmente adquirido no seu site oficial.

- **Principais características do PHP**

Código Aberto: Todo o código fonte do PHP está disponível para consulta e alteração;

Custo Zero: O PHP é gratuito.

Multi-plataforma: O PHP pode correr em Unix, Linux, Windows, etc.

Eficiência: O PHP consome poucos recursos do servidor, permitindo que programas complexos sejam desenvolvidos, sem que isto implique um tempo elevado na sua execução. Além disso, o PHP sendo um módulo nativo do servidor Web, evita chamadas externas, o que o torna ainda mais eficiente;

Acesso à Base de Dados: Podemos aceder directamente aos principais sistemas de gestão de base de dados utilizados actualmente e a qualquer base de dados do mercado por meio de ODBC;

Processamento de Imagens: O PHP pode criar imagens dinamicamente e enviá-las ao navegador Web do utilizador.

Além destas características, podemos referir ainda a capacidade de ler informação XML, processamento de ficheiros (leitura e escrita, tanto no formato de texto como binário), a manipulação de variáveis complexas, a utilização de funções e classes e geração de código JavaScript, ou outro qualquer para processamento no lado do cliente, a manipulação de e-mails, a gestão de documentos PDF (“Portable Document Format”) e muitas outras

características que tornam o PHP uma linguagem realmente potente e indicada para a construção de aplicações Web dinâmicas, tornando-se na escolha natural para este tipo de aplicação.

A Figura 3.11 ilustra a interação entre o PHP, o servidor Web e o navegador do utilizador.

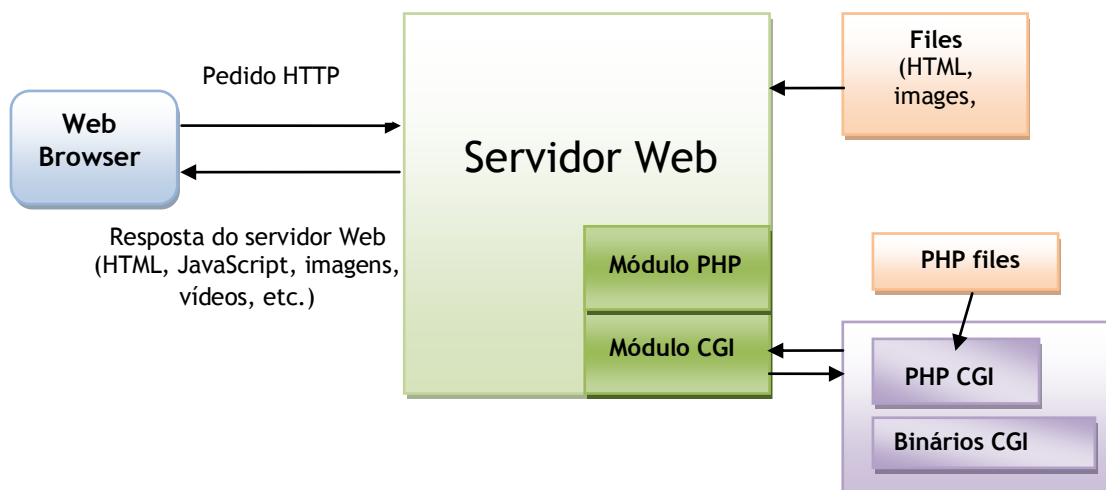


Figura 3.11 - Interação entre o PHP, o servidor Web e o navegador do cliente.

➤ CSS

Cascading Style Sheets, ou simplesmente **CSS**, é uma linguagem de estilo utilizada para definir a apresentação de documentos escritos numa linguagem de marcação, como HTML ou XML. O principal benefício é prover a separação entre o formato e o conteúdo de um documento. Estas foram desenvolvidas com o objectivo de criar um método consistente e unificado de atribuir a formatação e aspecto a páginas Web.

Em vez de se colocar a formatação dentro do documento, cria-se uma ligação para uma página que contém os estilos, procedendo de forma idêntica para todas as páginas de um “site”/aplicação Web. Para alterar a aparência basta, portanto, modificar apenas um ficheiro.

O aparecimento das folhas de estilo está intrinsecamente ligado com a evolução do HTML, este evoluiu de modo a comportar cada vez mais e diversas variedades estilísticas, permitindo o desenvolvimento de páginas mais ricas e complexas. Esta evolução atribui ao designer mais controlo sobre a aparência mas teve um custo elevado na complexidade do código HTML tornando difícil de escrever e manter. As variações nos navegadores Web tornaram complicado manter a consistência na aparência de um site e o modo como este era visualizado. Consequentemente, foram introduzidas as folhas de estilo.

As CSS permitem que o aspecto de uma página seja influenciado por várias folhas de estilo, uma folha de estilo pode herdar em cascata atributos estilísticos de outras folhas permitindo uma mistura de estilos, controladas pelo utilizador e pelo designer do site.

A Figura XXX ilustra como são definidas as folhas de estilo, e como estas atribuem o estilo que uma página vai conter.

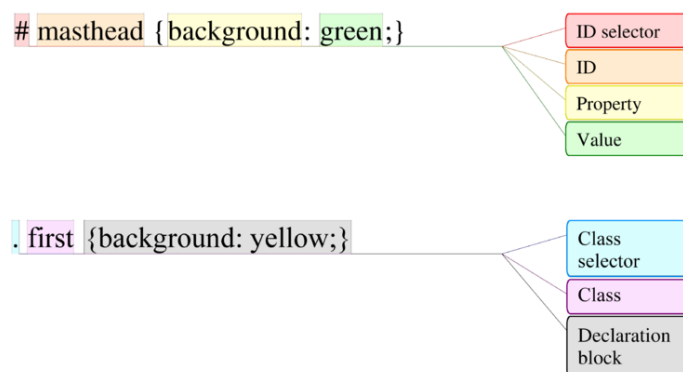


Figura 3.12 - Exemplo das definições dos estilos numa CSS

➤ PostgreSQL

PostgreSQL é um poderoso sistema de base de dados relacional de código aberto. Tem mais de 15 anos de contínuo desenvolvimento e uma arquitectura comprovada e com a qual adquiriu uma forte reputação de fiabilidade e integridade. Oferece suporte para os principais sistemas operativos, incluindo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e Windows. Este respeita totalmente as propriedades ACID (Atomicidade, Consistência, Isolamento, Durabilidade), propriedades essas que em termos da ciência de computação, garantem o correcto funcionamento das transacções da base de dados. Esta suporta por completo, chaves estrangeiras, junções, vistas, “*triggers*” e procedimentos (em vários idiomas). Suporta diversos tipos de dados SQL92 e SQL99, nomeadamente: “*INTEGER*”, “*BOOLEAN*”, “*CHAR*”, “*VARCHAR*”, “*DATA*”, “*INTERVAL*”, e “*TIMESTAMP*”. Suporta também o armazenamento de grandes objectos binários, incluindo imagens, sons, ou vídeo. Possui interfaces de programação nas linguagens C / C + +, Java, Net, Perl, Python, Ruby, Tcl, ODBC, entre outras, e uma vasta documentação.

É uma base de dados de nível empresarial possuindo sofisticados recursos, como o *Multi-Version Concurrency Control* (MVCC), evitando conflitos de acessos múltiplos, métodos de recuperação de dados por intermédio de pontos de recuperação, replicação assíncrona de dados, transacções aninhadas, um sofisticado sistema de planeamento e optimização das consultas. É altamente escalável, tanto na grande quantidade de dados que pode guardar e gerir como no número de utilizadores simultâneos. A Tabela 3.1 ilustra algumas das características do PostgreSQL.

Tabela 3.1 - Características do PostgreSQL

Valor	Limite
Tamanho máximo de dados	Ilimitado
Tamanho máximo de uma tabela	32TB
Tamanho máximo de uma linha	1,6TB
Tamanho máximo de um campo	1GB
Número de linhas por tabela	Ilimitado
Número máximo de colunas por tabela	250 - 1600 (dependendo dos tipos de coluna)
Número máximo de índices por tabela	Ilimitado

PostgreSQL tem ganho elogios e reconhecimento da indústria e dos seus utilizadores, incluindo o “*Linux New Media Award* “ de melhor sistema de base de dados e cinco vezes vencedor do “*Linux Editors' Choice Award*” para o melhor sistema de gestão de base de dados.

PostgreSQL executa procedimentos armazenados em mais de uma dúzia de linguagens de programação, incluindo Java, Perl, Python, Ruby, Tcl, C / C + +, e o seu próprio PL / pgSQL, que é semelhante ao PL / SQL da *Oracle*.

O melhor de tudo é o facto de o PostgreSQL ser gratuito e o seu código fonte está disponível sob a licença “*open source*” mais liberal: a licença BSD. Esta licença oferece a liberdade de usar, modificar e distribuir PostgreSQL, sob qualquer forma. Como tal, PostgreSQL não só é um poderoso sistema de base de dados capaz de gerir uma empresa, como também é uma plataforma sobre a qual é possível o desenvolvimento fácil e rápido de aplicações não comerciais e comerciais.

Tendo em conta os requisitos do projecto não é necessária uma solução comercial. Desta forma, conclui-se que o PostGreSQL tem todas as funcionalidades requeridas, é uma solução sem custo acrescido e acima de tudo é o sistema de gestão de base de dados já utilizado pelo QualiFeup.

Capítulo 4

Aprofundamento da análise do Módulo Gestão de Tarefas

Este capítulo vai concentrar-se em aspectos ligados à conceptualização do módulo de Gestão de Tarefas. Este módulo é o mais exigente e complexo em termos de requisitos. Logo, por intermédio de uma análise detalhada do seu funcionamento, é possível obter uma melhor compreensão da especificação e desenvolvimento dos restantes módulos em termos de interfaces e reutilização de código.

Em primeiro lugar é necessário referir que a entidade tarefa é uma entidade complexa que atravessa vários estados. Desta forma, foi necessário elaborar um diagrama de estados associado a esta identidade. A partir deste diagrama foram identificadas as operações que é possível efectuar em cada um dos estados. Em seguida, foram identificados os formulários necessários para cobrir essas operações. Dentro destes formulários são identificados os que podem ser reutilizados. Para cada um desses formulários é efectuado um esboço da possível interface que satisfaz as necessidades associadas a cada um deles. Por fim, para cada um dos formulários é efectuada uma análise detalhada dos elementos que os constituem.

Durante todo o processo de análise dos formulários e dos respectivos elementos que os constituem são consideradas as permissões dos utilizadores, bem como os estados da entidade tarefa.

4.1 - Estados de uma tarefa

Uma tarefa, tal como referido anteriormente, atravessa diversos estados durante o seu “ciclo de vida”. Estes estados representam todo o processo desde a criação até à execução. Dentro dos diferentes estados que a entidade pode assumir, os utilizadores vão visualizar diferentes formulários e vão ter acesso a diferentes operações. Os estados identificados para a entidade tarefa são:

- **Planeada:** A tarefa foi planeada para execução e encontra-se no sistema
- **Executada:** A tarefa foi executada com sucesso.
- **Parcialmente Executada:** A tarefa foi executada em parte.

- **Não executada:** A tarefa não foi executada.
- **Arquivada:** A tarefa foi colocada em arquivo.

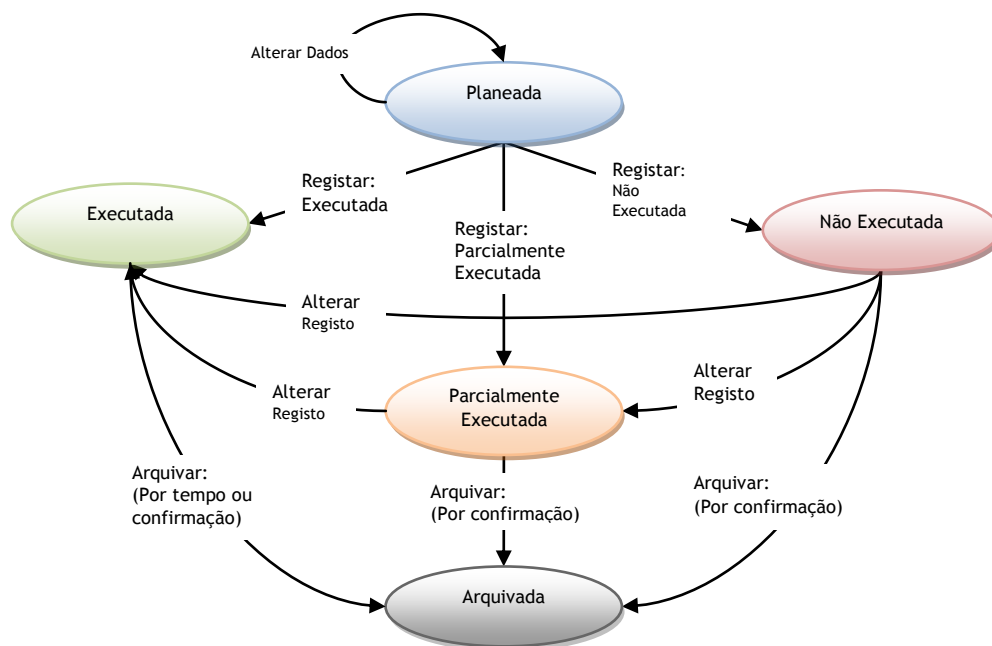


Figura 4.1 - Diagrama de Estados de uma tarefa

A partir da listagem de casos de uso faz-se uma ligação com os diferentes estados da entidade tarefa, o que torna possível definir as operações possíveis para cada estado. As tabelas que se seguem são o resultado da análise efectuada.

➤ Planeada

Tabela 4.1 - Lista de operações possíveis sobre uma tarefa no estado Planeada

Lista de Acções
Pesquisar:
- Consultar na lista de tarefas
Detalhes:
- Consultar detalhes da tarefa
- Alterar planeamento
- Apagar tarefa
- Registar
Imprimir:

➤ Parcialmente Executada

Tabela 4.2 - Lista de operações possíveis sobre uma tarefa no estado Parcialmente Executada

Lista de Acções
Pesquisar:
- Consultar na lista de tarefas
Detalhes:
- Consultar detalhes da tarefa
- Alterar tarefa

- Alterar registo
- Apagar tarefa
- Arquivar tarefa
Imprimir:

➤ **Executada**

Tabela 4.3 - Lista de operações possíveis sobre uma tarefa no estado Executada

Lista de Acções
Pesquisar:
- Consultar na lista de tarefas
Detalhes:
- Consultar detalhes da tarefa
- Alterar tarefa
- Apagar tarefa
- Arquivar tarefa
Imprimir:
* Responsável pelo registo.

➤ **Não executada**

Tabela 4.4 - Lista de operações possíveis sobre uma tarefa no estado Não Executada

Lista de Acções
Pesquisar:
- Consultar na lista de tarefas
Detalhes:
- Consultar detalhes da tarefa
- Alterar registo
- Apagar tarefa
- Arquivar tarefa
Imprimir:

➤ **Arquivada**

Tabela 4.5 - Lista de operações possíveis sobre uma tarefa no estado Arquivada

Lista de Acções
Pesquisar:
- Consultar na lista de tarefas
Detalhes:
- Consultar histórico
- Apagar tarefa

Através da análise das tabelas apresentadas, conclui-se que muitas das operações são semelhantes entre si. Em seguida, procedeu-se à análise de cada uma das operações, sendo identificados quais os formulários necessários para as executar. Através da análise efectuada, concluiu-se que a entidade tarefa se subdividia em dois grandes tipos a gerir: Tarefas Únicas e Tarefas Periódicas.

Para cada um dos tipos em que a entidade tarefa se subdivide, foram identificados dois géneros de formulários diferentes. Formulários que listam tarefas, e formulários que trabalham sobre uma instância específica. Ou seja, formulários que permitem listar as diferentes tarefas, e formulários que permitem efectuar operações sobre os diferentes tipos

Tabela 4.6 - Distribuição das operações pelos formulários identificados.

Operações	Formulários						
	Lista Tarefas hoje/atrasadas	Lista Tarefas Mês/Semana	Lista Tarefas Periódicas	Gestão Tarefas Periódicas	Gestão Tarefas Únicas	Arquivo de Tarefas	Imprimir Tarefas
Consultar Lista de Tarefas Hoje	X						
Consultar Lista de Tarefas Atrasadas	X						
Consultar Lista de Tarefas Semana		X					
Consultar Lista de Tarefas Mês		X					
Consultar lista de Tarefas Periódicas			X				
Consultar Detalhes Tarefa Única					X		
Consultar Detalhes Tarefa Periódica				X			
Criar Nova Tarefa Única					X		
Criar Nova Tarefa Periódica				X			
Alterar Tarefa Única					X		
Alterar Tarefa Periódica				X			
Apagar Tarefa Única					X		
Registrar Tarefa					X		
Arquivar Tarefa					X		
Consultar Arquivo						X	
Imprimir Tarefa							X

4.2 - Esboços dos Diferentes Interfaces

Para os diferentes formulários identificados é desenvolvido um esboço das respectivas interfaces, apresentadas de seguida.

➤ Lista Tarefas hoje/atrasadas

O formulário é dividido em duas colunas principais: 'Hoje' e 'Atrasadas'. Cada coluna tem uma sub-cabeçalha com 'Todas' e 'Vigilantes'. Abaixo, há uma área de lista com ícones (um ícone vermelho na coluna 'Hoje' e um ícone verde na coluna 'Atrasadas') e uma barra de rolagem. Na base do formulário, há uma barra de navegação com os seguintes botões: 'Nova', 'Tarefas Periódicas', 'Tarefas do Mês', 'Tarefas da Semana', 'Histórico de Tarefas', 'Imprimir' e 'Volta' (com uma seta para trás).

Figura 4.3 - Esboço do formulário “Lista Tarefas Hoje/Atrasadas”

A listagem de tarefas do dia e tarefas atrasadas é praticamente igual, o que implica que não existe a necessidade de criar formulários separados. A distinção que existe entre estes formulários ao nível da interface deve-se apenas ao facto de facilitar a consulta por parte dos utilizadores. Isto faz com que os utilizadores identifiquem de uma forma mais fácil quais as tarefas que foram atribuídas ao próprio dia, daquelas que ficaram em atraso de dias anteriores. A listagem destas tarefas torna-se, então, mais intuitiva se for feita na mesma interface, como está exemplificado na **Figura 4.3**.

➤ Lista Tarefas Mês/Semana

O formulário apresenta uma interface para a listagem de tarefas por período. No topo, há uma barra de cabeçalho com o texto 'Sexta-Feira dia 11 de Abril'. Abaixo, há uma área de lista com ícones (um ícone vermelho e um ícone cinza) e uma barra de rolagem. Na base do formulário, há uma barra de navegação com os seguintes botões: 'Nova', 'Tarefas Periódicas', 'Tarefas do Mês', 'Tarefas da Semana' e 'Histórico de Tarefas'. Abaixo desta barra, há uma barra de cabeçalho com o texto 'Sábado dia 12 de Abril'.

Figura 4.4 - Esboço do formulário “Lista Tarefas Mês/Semana”

A listagem de tarefas da semana e do mês vão necessitar de um outro tipo de interface. O esboço representado na **Figura 4.4** representa a interface para este tipo de tarefas. O interface é semelhante quer a lista de tarefas seja do mês ou da semana, apresentando apenas uma diferença no tamanho da lista (no caso da lista de tarefas de um mês o número de elementos é maior).

➤ **Lista Tarefas Periódicas**

Esboço do formulário "Lista Tarefas Periódicas". O formulário contém uma tabela com três colunas: "Prioridade", "Descrição" e "Data". A coluna "Prioridade" possui um botão vermelho. A coluna "Descrição" possui uma barra de rolagem vertical. A coluna "Data" possui uma barra de rolagem vertical. Abaixo da tabela, há um botão "Nova Periódica" e um botão "Volta" com uma seta apontando para a esquerda.

Figura 4.5 - Esboço do formulário "Lista Tarefas Periódicas"

➤ **Gestão Tarefas Periódicas**

Esboço do formulário "Gestão Tarefas Periódicas". O formulário contém dois grupos de botões "Sim" e "Não". Abaixo, há uma linha de botões para dias da semana: "2ª", "3ª", "4ª", "5ª", "6ª", "SAB", "DOM". À direita, há campos para "De:" e "Até:" e um campo para "Hora:". Abaixo, há campos para "Descrição da tarefa:" e "Observações da tarefa:". Abaixo dos campos, há um botão "Confirmar" e um botão "Voltar" com uma seta apontando para a esquerda.

Figura 4.6 - Esboço do formulário "Gestão Tarefas Periódicas"

Uma tarefa periódica é uma tarefa que se vai repetir. Por exemplo, para que os utilizadores do sistema não sejam obrigados a criar uma nova instância da tarefa todas as semanas, o sistema permite que seja planeada uma tarefa repetitiva e vai criar instâncias da tarefa para todos os períodos de um modo automático. A criação de formulários diferentes das tarefas únicas para gerir e listar tarefas periódicas é, portanto, necessária.

Temos então conceitos diferentes que devem ser geridos de um modo separado. Assim, obtém-se um melhor controlo sobre o funcionamento geral da aplicação. É essencial, portanto, a utilização de formulários separados para tarefas periódicas e tarefas únicas, isolando assim estas duas lógicas diferentes, e embora os formulários para ambos os tipos de tarefas sejam parecidos.

O formulário Gestão Tarefas Periódicas vai ser o mesmo para as operações: criar, editar e alterar. A diferença reside nos dados apresentados. No caso da operação criar, o formulário aparece por preencher, enquanto que para a operação consultar este exhibe os dados da tarefa, existindo a possibilidade de editar ou apagar. Para a operação alterar, os dados

apresentam-se num estado editável. As **Figuras 4.5 e 4.6** representam um esboço dos formulários necessários.

➤ **Arquivo de Tarefas:**

Registos entre: e:

Utilizador: Tipo de Operação:

Prioridade	Descrição	Data
.	.	.
.	.	.
.	.	.

Figura 4.7 - Esboço do formulário "Arquivo de Tarefas"

O esboço da interface para o formulário “Arquivo de Tarefas” é representado na **Figura 4.7**. Este formulário destina-se aos utilizadores responsáveis pela gestão de trabalho com funções de administrador, só estes terão acesso à utilização deste histórico dos registos. Uma tarefa quando é arquivada só poderá ser consultada por intermédio deste formulário. A utilização deste formulário vai portanto permitir ao utilizador responsável consultar tarefas arquivadas. Esta consulta de tarefas pode ser feita por utilizador, operação ou intervalo temporal.

➤ **Imprimir Tarefas**

☒ ☐ ☒ ☒ ☐

Figura 4.8 - Esboço do formulário "Imprimir Tarefas"

A **Figura 4.8** representa o esboço do formulário “Imprimir Tarefas” utilizado, como o nome indica, para imprimir uma descrição das tarefas. Este vai poder ser chamado por outros formulários e a listagem de tarefas que apresenta vai depender desses formulários que o invocam. Na listagem apresentada é possível escolher quais as tarefas a imprimir.

➤ Gestão Tarefas Únicas

Uma vez que este formulário é o mais complexo será feita uma análise mais detalhada nos pontos seguintes. O formulário, além de agrupar as várias operações referidas na **Tabela 4.6**, vai depender do estado em que a entidade tarefa se encontra e das permissões do utilizador. Como tal, toda a informação e opções disponíveis vai variar. Apresentam-se para já as diversas configurações possíveis que este pode ter.

Este formulário para a criação de uma nova tarefa única inclui os seguintes elementos:

- Campos de seleção para "Prioritária" (Sim/Não) e "Vigilante" (Sim/Não).
- Campos de entrada para "Data" e "Hora".
- Campos de texto para "Descrição da tarefa:" e "Observações da tarefa:".
- Botões "Confirmar" e "Voltar" (representado por uma seta).

Figura 4.9 - Esboço do formulário Gestão Tarefas Únicas - Criar nova

Este esboço (**Figura 4.9**) exemplifica uma possível solução da configuração do formulário para a operação “Criar Nova Tarefa Única”.

Este formulário para consultar os detalhes de uma tarefa única no estado "Planeada" inclui os seguintes elementos:

- Indicadores de status: "Prioritária - Sim" (com ícone de alerta) e "Vigilante - Sim".
- Campos de entrada para "Data" e "Hora".
- Campos de texto para "Descrição da tarefa:" e "Observações da tarefa:".
- Campos de entrada para "Criada em:" e "Autor:".
- Campos de entrada para "Tarefa por executar.".
- Botões "Alterar", "Apagar" e "Registrar".
- Botão "Voltar" (representado por uma seta).

Figura 4.10 - Esboço do formulário Gestão Tarefas Únicas - Consultar Detalhes - Estado Planeada

Este esboço (**Figura 4.10**) exemplifica uma possível solução da configuração do formulário para a operação “Consultar Detalhes Tarefa Única”, quando esta se encontra no estado planeada.

Prioritária	Sim	Não	Vigilante	Sim	Não	Data		Hora	
Descrição da tarefa: 									
Observações da tarefa: 									
Criada em:			Autor:						
Tarefa por executar.									
Confirmar						Voltar			

Figura 4.11 - Esboço do formulário Gestão Tarefas Únicas - Editar - Estado Planeada

Este esboço (Figura 4.11) exemplifica uma possível solução da configuração do formulário para a operação “Alterar Tarefa Única”, quando a tarefa se encontra no estado planeada.

☒	Prioritária - Sim	Vigilante - Sim	Data		Hora	
Descrição da tarefa: 						
Observações da tarefa: 						
Criada em:		Autor:				
A tarefa executada com sucesso?			Sim	Executei parte	Não	
Observações sobre a execução da tarefa: 						
Alguma anomalia?		Sim	Não			
Confirmar			Voltar			

Figura 4.12 - Esboço do formulário Gestão Tarefas Únicas - Registar (Executada)

Este esboço (Figura 4.12) exemplifica uma possível solução da configuração do formulário para a operação “Registar Tarefa”, em que a tarefa é dada como executada com sucesso.

☐ Prioritária - Sim	Vigilante - Sim	Data _____	Hora _____
Descrição da tarefa: _____			
Observações da tarefa: _____			
Criada em: _____		Autor: _____	
A tarefa executada com sucesso?		Sim	Executei parte
Porquê?		Não	

Pretende colocar um aviso?		Sim	Não

Alguma anomalia?		Sim	Não

Confirmar		Voltar	

Figura 4.13 - Esboço do formulário Gestão Tarefas Únicas - Registar (Não executada/Parcialmente Executada)

Este esboço (**Figura 4.13**) exemplifica uma possível solução da configuração do formulário para a operação “Registar Tarefa”, em que a tarefa é dada como não executada ou parcialmente executada.

☐ Prioritária - Sim	Vigilante - Sim	Data _____	Hora _____
Descrição da tarefa: _____			
Observações da tarefa: _____			
Criada em: _____		Autor: _____	
Tarefa executada com sucesso.			
Observações sobre a execução da tarefa: _____			

Anomalia associada _____			

Alterar tarefa		Apagar tarefa	
		Voltar	

Figura 4.14 - Esboço do formulário Gestão Tarefas Únicas - Consultar Detalhes - Estado Executada

Este esboço (**Figura 4.14**) exemplifica uma possível solução da configuração do formulário para a operação “Consultar Detalhes Tarefa Única”, quando esta já se encontra registada no sistema, ou seja, no estado: Executada, Parcialmente Executada ou Não Executada.

☐ Prioritária - Sim Vigilante - Sim Data _____ Hora _____

Descrição da tarefa: _____

Observações da tarefa: _____

Criada em: _____ Autor: _____

A tarefa executada com sucesso?

Observações sobre a tarefa: _____

Alguma anomalia?

Figura 4.15 - Esboço formulário Gestão Tarefas Únicas - Alterar - Estado Executada

Este esboço (**Figura 4.15**) exemplifica uma possível solução da configuração do formulário para a operação “Alterar Tarefa Única”, quando esta já se encontra registada no sistema, ou seja, no estado: Executada, Parcialmente Executada ou Não Executada

4.3 - Análise detalhada dos formulários

Para cada um dos formulários foi efectuada uma análise detalhada. O objectivo é identificar dentro dos formulários, quais são os elementos que os constituem, caso existam, e como estes elementos se vão alterar e apresentar consoante os estados das tarefas e das permissões dos utilizadores

Aqui é apresentado com detalhe essa análise ao formulário Gestão de Tarefas Únicas pois é o mais complexo. A análise dos restantes formulários seguiu uma lógica idêntica.

É possível verificar que este formulário é bastante complexo devido às diversas variantes que apresenta com os estados das tarefas e as operações a realizar. Com uma análise mais cuidada é possível identificar, logo à partida, três divisões lógicas neste formulário, facilitando bastante a sua compreensão.

O diagrama mostra a estrutura visual do formulário, dividido em três seções principais:

- Planeamento:** A seção superior, contendo campos de entrada para planeamento.
- Registo/Estado:** A seção do meio, destinada a registrar o estado da tarefa.
- Navegação/Opções:** A seção inferior, que contém botões para navegação e opções de ação.

Figura 4.16 - Divisão do formulário Gestão Tarefas Únicas em elementos.

Estes três elementos vão variar, dependendo do estado em que se encontra a tarefa, da operação que se está a executar e das permissões do utilizador. O acesso às diferentes “configurações” vai depender das permissões do utilizador. O formulário tem de ser capaz de consultar a lista de operações a que o utilizador autenticado no sistema tem acesso e filtrar a informação e o aspecto.

Descreve-se de seguida cada um destes elementos e como se relacionam e apresentam, consoante a operação que se está a executar.

➤ Criar Tarefa Única

Quando se está a efectuar esta operação sobre o formulário, este apresenta-se conforme a **Figura 4.9**.

O elemento “Planeamento” apresenta-se por preencher e o elemento “Navegação/Opções” permite confirmar a criação da nova tarefa ou cancelar voltando à página anterior.

➤ Consultar Tarefa Única

Nesta operação o elemento “Planeamento” apresenta os dados de planeamento da tarefa, seja qual for o estado da tarefa, e não se encontra editável.

O elemento “Registo/Estado” vai apresentar a informação associada ao registo da tarefa, no entanto esta depende do estado em que a tarefa se encontra. Dá-se o exemplo da **Figura 4.10** e **Figura 4.14**.

O elemento “Navegação/Opções” vai igualmente variar consoante o estado da tarefa. Em todos os estados vai permitir “voltar” e “Apagar”. No estado “Planeada” (**Figura 4.10**) vai oferecer ainda a opção “Registar” e “Alterar”. No estado “Executada”, “Parcialmente Executada” e “Não Executada” (**Figura 4.14**) vai oferecer a opção “Alterar” e “Arquivar”.

➤ **Registar Tarefa**

Nesta operação o elemento “Planeamento” apresenta os dados de planeamento da tarefa e não se encontra editável.

O elemento “Registo/Estado” toma o aspecto descrito na **Figura 4.12** ou **Figura.4.13** encontrando-se no estado editável.

O elemento “Navegação/Opções” permite confirmar ou cancelar voltando à página anterior.

➤ **Alterar Tarefa Única**

Nesta operação o elemento “Planeamento” apresenta-se no estado editável quando a tarefa se encontra no estado “Planeada” (**Figura 4.11**) e não editável quando está em qualquer um dos outros estados.

O elemento “Registo/Estado” toma o aspecto descrito na **Figura 4.11** no estado “Planeada” ou seja não editável. E toma o aspecto descrito na **Figura 4.15** ou seja editável quando se encontra em qualquer outro dos estados excepto “Arquivada”.

O elemento “Navegação/Opções” permite confirmar ou cancelar a operação em causa voltando à página anterior seja qual for o estado da tarefa.

Após desta análise somos agora capazes de definir o formulário “Gestão de Tarefas Únicas”. Este formulário, tendo em conta a estrutura definida na arquitectura lógica será um “Main Script” e os elementos os vários *templates*.

Capítulo 5

Descrição da Análise e Implementação do Módulo Gestão de Anomalias

Este capítulo descreve com pormenor as soluções utilizadas no desenvolvimento do módulo Gestão de Anomalias, visto que em termos técnicos e tecnológicos ser o mais exigente, sendo possível ter uma clara visão sobre a arquitectura e a estruturação de blocos referida no capítulo 3 na **Figura 3.2**.

Para o Módulo de Gestão de Anomalias seguiu-se uma abordagem algo diferente dos restantes. Embora a análise seguida anteriormente para o módulo de Gestão de Tarefas possa ser aplicada a todos os outros módulos, incluindo o módulo de anomalias, este, no entanto, é uma particularidade que não existe nos outros módulos, que é a necessidade de um formulário de inserção de dados do tipo tabela. Conforme referido no ponto 2.3, existem múltiplos tipos de anomalias que se pretende registar, por exemplo, mobiliário, eléctricas, limpeza, equipamento, etc.

Na análise e concepção do módulo foram consideradas duas alternativas:

- 1^a. Projectar um sub-módulo específico para cada formulário de registo de anomalia.
- 2^a. Projectar um módulo genérico que permita criar e configurar novos formulários sem necessidade de proceder a alterações ao nível das camadas de acesso a dados e lógica.

Como o formulário de gestão de anomalias necessita de ser facilmente utilizado, reconfigurado e permitir a inserção todo o tipo de dados, como tal a escolha recaiu sobre a 2^a opção, tendo sido desenvolvido um módulo que permita configurar novos formulários de registo de anomalias, especificando para cada formulário:

- As salas em que podem ser registadas as anomalias
- Os tipos de anomalias aplicáveis
- O domínio de valores possíveis

Segue-se a apresentação do processo de análise utilizado no desenho deste módulo.

5.1 - Análise da Interface com utilizador

Foi necessário identificar uma estrutura tipo para os formulários que seriam utilizados. Inicialmente foi definida a estrutura ilustrada na Figura 5.1.

Legenda para o tipo de navegação - (Sector, Edifício, Piso etc)

↓

Botões de navegação (Sector, Edifício, Piso)

Piso	Sala	Anomalia 1	Anomalia 2	

Figura 5.1 Esboço da estrutura do formulário tabela de anomalias

A análise revelou os seguintes tipos de dados a registar para cada anomalia:

- “Numérico”: Inserção de um valor numérico, por exemplo “Lâmpadas fundidas”
- “Sim/Não”: Do tipo verdadeiro ou falso, por exemplo “Falta secador de mãos”
- “Escolha Múltipla”: Lista de opções pré-definida, por exemplo “Saboneteira: - Avariada; - Sem Sabonete; - Não tem saboneteira; ...”
- “Inserção Livre”: Inserção de um texto livre.

Analisando o tipo de anomalias, associou-se de uma maneira lógica as seguintes soluções para a sua inserção no formulário:

- “Sim/Não” - “Checkbox”
- “Múltipla Escolha” - “Radio Button”
- “Numérica”- “Text input”
- “Inserção livre” - “Text input”

Estes seriam os mecanismos de inserção. A estes mecanismos vamos chamar “tipo de controlo da anomalia”. No entanto surgiu um problema de interface, ou seja, como teriam os utilizadores acesso a estes mecanismos?

Várias alternativas foram pensadas e testadas, todas elas com vantagens e desvantagens quer ao nível da interface com o utilizador, quer ao nível da implementação. Segue-se a apresentação de algumas das várias alternativas analisadas:

1ª - Um botão geral no formulário de Editar/Inserir, que ao ser accionado todas as células passariam ao estado editável.

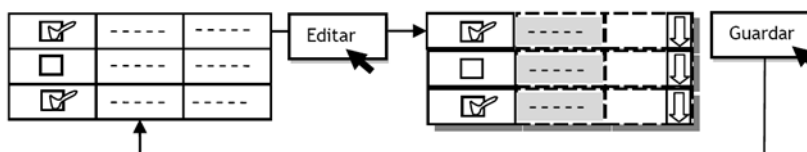


Figura 5.2 - Exemplo da primeira alternativa para inserção de anomalias

Vantagens: Fácil implementação, código construído totalmente pelo servidor.

Desvantagens: Dois formulários distintos, duplicação de código. Desconforto por parte do utilizador, devido ao carregar da nova página, e às diferenças de interface. Aspecto demasiado “carregado” e confuso.

Envio das alterações para a BD: Botão geral no formulário para guardar em que o envio seria feito sincronamente por “Post”.

2ª - Ao seleccionar uma célula, esta iria alterar-se para um estado editável.

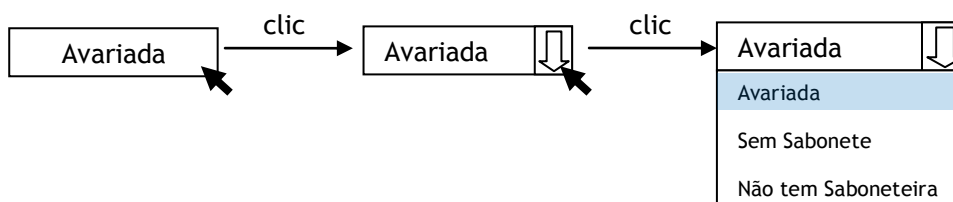


Figura 5.3 - Exemplo da segunda alternativa para inserção de anomalias

Vantagens: Melhor fluxo de trabalho, aspecto mais uniforme, melhor identificação do elemento seleccionado.

Desvantagens: Implementação mais difícil. Como cancelar a operação de edição? Espaço limitado das células faria com que a inserção do elemento de controlo alterasse o aspecto do formulário. A inserção de uma anomalia de texto livre seria difícil com o espaço limitado.

Envio das alterações para a BD: Ao premir a tecla “Enter” seria feito o envio por “Post” sincronamente refrescando a página ou seria feito um envio assincronamente utilizando “Ajax”.

3ª - O formulário de edição/inserção encontra-se permanentemente na página, e ao seleccionar uma célula esta é preenchida com os dados da anomalia e sala em causa.

Vantagens: Maior facilidade de implementação, código totalmente gerado pelo servidor.

Desvantagens: Grande diversidade de formulários, pouco intuitivo, não funcional.

Envio das alterações para a BD: Ao premir o botão “Confirmar” seria feito o envio por “Post”, sincronamente e refrescando a página.

4ª - Ao seleccionar uma célula toda a página é refrescada e num ponto fixo da página é inserido o formulário de alteração/adição.

Vantagens: Maior facilidade de implementação, código totalmente gerado pelo servidor

Desvantagens: Pouca fluidez de trabalho e pouco intuitivo, inexistência de um ponto ideal para colocar o formulário (se o formulário for maior que a resolução do ecrã e existir a necessidade de fazer “scroll”, trará desconforto associado à quebra do fluxo de trabalho)

Envio das alterações para a BD: Ao premir o botão “Guardar” seria feito o envio por “Post”, sincronamente e refrescando a página.

5ª - Ao seleccionar uma célula utiliza-se um “tooltip”, configurado para apresentar um formulário de inserção da anomalia.

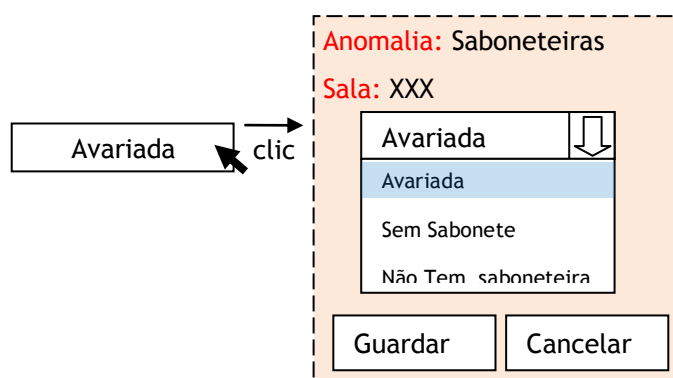


Figura 5.4 - Exemplo inserção de anomalia usando um "tooltip"

Vantagens: Boa identificação do elemento seleccionado que se traduz num melhor fluxo de trabalho, resolução dos problemas de espaço e modificação do aspecto do formulário.

Desvantagens: Implementação dependente de bibliotecas externas e código de terceiros, limitado ao funcionamento previsto por estas. Código gerado na máquina do cliente (javascript). Isto obrigava a passagem de toda a informação para a máquina do cliente para permitir gerar o “tooltip”.

Envio das alterações para a BD: Ao premir o botão “Guardar” seria feito o envio por “Post” sincronamente refrescando a página ou seria feito um envio assincronamente utilizando “Ajax”.

6ª- Ao seleccionar uma célula do formulário recarregar a página e criar uma divisão flutuante posicionada como se fosse um “*tooltip*”, carregar essa divisão com o tipo de formulário e dados adequados.

Vantagens: Facilidade de implementação, todo o código seria gerada pelo servidor.

Desvantagens: Desconforto de utilização associado a uma grande quebra do fluxo de trabalho.

Envio das alterações para a BD: Ao premir o botão “Guardar” seria feito o envio por “Post” sincronamente refrescando a página ou seria feito um envio assincronamente utilizando “Ajax”.

7ª - Ao seleccionar uma célula tornar visível uma divisão escondida que foi carregada na página aquando da criação desta, posicionando-a dinamicamente como se tratasse de um “*tooltip*”, preenchendo apenas os campos necessários. Seria inserida uma divisão para cada tipo de formulário de inserção.

Vantagens: Boa identificação do elemento seleccionado e melhor fluxo de trabalho, resolução dos problemas de espaço e modificação do aspecto do formulário, código do servidor com efeitos no cliente.

Desvantagens: Implementação mais difícil.

Envio das alterações para a BD: Ao premir o botão “Guardar” seria feito o envio por “Post” sincronamente refrescando a página ou seria feito um envio assincronamente utilizando “Ajax”.

No final, a solução escolhida recaiu sobre esta última opção. Embora mais complicada de implementar revela ser a mais adequada para resolver o problema em causa. Esta solução no fundo é uma conjugação das outras opções, tentando acoplar as vantagens minimizando as desvantagens.

Partindo desta solução, foi criado um template para cada uma das divisões utilizadas para cada um dos tipos de controlo. Estes *templates* serão exemplificados mais adiante.

Estando o tipo de dados definido, assim como o modo como estes seriam controlados, segundo uma lógica de interface, foi elaborado um modelo de dados capaz de suportar esta inserção de anomalias e ao mesmo tempo escalável o suficiente para acoplar alterações futuras. O seguinte modelo exemplifica a estrutura de dados criada com vista a suportar estas funcionalidades (**Figura 5.5**).

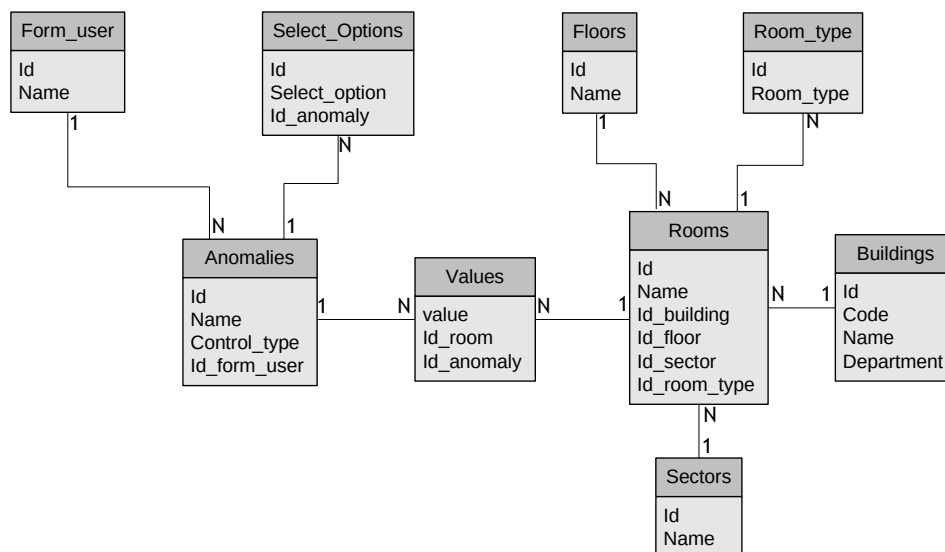


Figura 5.5 - Modelo da base de dados para suportar os formulários de inserção de anomalias.

Na tabela “Form User” guarda-se a referência aos formulários criados, enquanto que na tabela “Anomalies” se registam as várias anomalias que são utilizadas. A tabela “Select_options” contém as várias opções para as anomalias de escolha múltipla. As várias salas estão guardadas na tabela “Rooms”, sendo atribuído a cada sala um tipo, que fica guardado na tabela “Room_type”. Este valor distingue os vários tipos de salas (salas de aulas, estudo, gabinetes, sanitários, etc.). Os dados de localização, Edifício, Piso e Sector estão guardados nas tabelas “Building”, “Floor”, “Sector”, respectivamente. A tabela “Values” regista o valor que uma sala tem para uma determinada anomalia.

5.2 - Análise detalhada dos “Scripts” do módulo Gestão de Tarefas

- “D.B. Scripts”

Para inserir e obter o valor de uma anomalia para uma sala vamos necessitar de funções de acesso a base de dados. Um “D.B. Script” irá conter todas as funções de acesso a dados para este módulo, este foi chamado de dbAccessAnomalies.php. As funções definidas para este “D.B. Script” foram:

- `getAnomalyValue()`
Esta função recebe como parâmetros o “id” da anomalia e o “id” da sala e retorna o valor associado ao par.
- `newAnomalyValue()`
Esta função recebe como parâmetros o “id” da anomalia, o “id” da sala e o valor associado e insere na base de dados.

- Um array para os valores:

```
$Values = Array ( [id_room] => Array ([ id_anomaly])
```

- **“Action Scripts”**

Para se efectuar uma operação sobre o formulário é necessário um “action script” responsável por gerir os dados a inserir e por intermédio do “D.B. Script” actualizar os valores. Temos então o “action script”:

- `action_InsertUpdateAnomalyValues.php`

Este “script” recebe o “id” da anomalia, o “id” da sala e o valor associado. Verifica por intermédio da função `getAnomalyValue()` se já existe esse valor inserido na base de dado, se sim faz uma actualização (`updateAnomalyValue()`) se não insere o novo valor (`insertAnomalyValue()`). Para apagar uma anomalia basta enviar um valor nulo para o par “id” anomalia, “id” sala.

- **“Main Script”**

Tendo sido identificado uma estrutura para o formulário, os dados que este vai conter e as funções para aceder a esses dados, é possível agora descrever o “Main Script” que controla toda a lógica do processo. Para cada formulário vai existir um “Main Script” com nome diferente. Estes mantêm a estrutura e lógica de funcionamento mas vão ser responsáveis por gerar formulários distintos.

O “Main Script” é responsável por obter os dados, por os organizar e chamar os *templates* que vão gerar o formulário adequado. É no “Main Scrip” que também se define qual o formulário que se vai criar, e para que salas se destina. A estrutura e funcionamento deste “script” pode então descrever-se como se segue.

No “Main Script” invoca-se o “D.B. Script” (`dbAcessAnomalies.php`) que contém a função `getDataAnomalies()`, define-se o tipo de formulário (`$user_form_type`), e os filtros para as salas (`$rooms_select_options[‘...’]`). Enquanto o formulário é um valor fixo, o *array* dos filtros vai ser actualizado consoante a escolha de navegação do utilizador (obtido por `$_GET`). A função retorna, então, os três *arrays* (anomalias, salas e valores) que serão passados aos *templates*. É necessário definir alguns parâmetros de inicialização dos *templates*, qual o tipo de navegação do formulário (edifício, sector, piso), o nome que vai ter a página, entre outros, pois os *templates* para serem o mais gerais possíveis não contêm informações específicas.

Por fim, invocam-se os *templates*, um para o “Header” e outro para o “Footer”, sendo estes gerais a toda aplicação (cada “Main Script” carrega o “Header” e o “Footer” com informação específica para cada caso, mas os *templates* são gerais à aplicação), e também o “Body”, que neste caso específico será o responsável pela criação do formulário de anomalias.

- “*Templates*”

Foi criado um template modelo para gerar formulários, com a estrutura descrita na **Figura 5.1**. Este utiliza os arrays que o “*Main script*” lhe passou para construir o formulário, utilizando Smarty. O seguinte pseudocódigo exemplifica de um modo geral como o template cria a estrutura:

```

Tabela>
  <Cabeçalho>
    Piso
    Sala Nº
    {Foreach anomalia}
      Nome da anomalia
    {/Foreach}
  </Cabeçalho>

  <Corpo>
    {Foreach Sala}
      <Linha>
        {Foreach Anomalia}
          {If Anomalia tipo de controlo = “checkbox”}
            <Célula>{If Valor[id_sala][id_anomalia] = “checked”}
              Checkbox (checked)
            {Else}
              Checkbox (unchecked)
            </célula>
          {If Anomalia tipo de controlo = escolha múltipla}
            <celula nome= id_salaXid_anomalia classe = mostra_divisão >
              Valor [id_sala][id_anomalia]
            </célula>
          {if Anomalia tipo de controlo = texto}
            <celula nome= id_salaXid_anomalia classe = mostra_divisão >
              Valor [id_sala][id_anomalia]
            </célula>
          ...
        {/Foreach}
      </Linha>
    {/Foreach}
  </Corpo>
</Tabela>

```

Por fim, o template insere as divisões escondidas para controlar cada uma das anomalias, onde cada uma destas divisões é, por si só, um outro template, ou seja, para cada tipo de

controlo temos uma divisão e cada divisão tem um ficheiro de template. Caso seja necessário efectuar alguma alteração a uma divisão apenas se tem de editar um ficheiro.

```
{Foreach Anomalia}

  {If Anomalia tipo de controlo = escolha múltipla}
    {include file = divisão_anomalia_escolha_multipla.tpl}
  {if Anomalia tipo de controlo = texto}
    {include file=divisão_anomalia_texto.tpl}
  ...

{/Foreach}
```

- **“Style”**

Sobre os *templates* é aplicado a CSS, esta é uma folha de estilo que contém a formatação para definir o aspecto final dos *templates*. Caso seja necessário alterar a cor, tamanho, tipo de letra, etc., bastará fazer esta alteração uma vez na folha de estilo e a alteração irá reflectir-se sobre toda a aplicação.

- **“Dynamic Scripts”**

É de notar, que no pseudocódigo não se faz nenhuma referência de como são tratadas as acções sobre o próprio formulário, apenas se definiu como esta é construída, devido à utilização dos “*Dynamic Scripts*”. São estes que detectam as acções e conferem o tratamento dinâmico da página, por intermédio do uso do jQuery, e do javascript não intrusivo retira-se o tratamento de eventos do código.

Os “*Dynamic Scripts*” são os responsáveis pelo tratamento de todos os efeitos que conferem o funcionamento dinâmico dos formulários. Foram criados dois “*Dynamic Scripts*”, um para lidar com os efeitos (“hidden_div_effects.js”) e outro para tratar do envio assíncrono dos dados para a base de dados (“hidden_div_ajax_submit.js”). Estes *scripts* estão programados de um modo não intrusivo, isto é, o tratamento de todos os eventos são feitos sem se inserir código javascript nos *templates*. O uso de jQuery facilita imenso este processo, permitindo de um modo simples indexar o elemento, propriedades e eventos. Assim estes “*Dynamic Scripts*” podem ser utilizados modularmente para qualquer formulário.

➤ **“Dynamic Script” - “hidden_div_effects”**

Ao criar o formulário, foi atribuído a cada célula um nome específico, o “id” da anomalia e o “id” da sala, criando-se assim um identificador único para cada célula (**Figura 5.11**). Este identificador é utilizado para registar em base de dados o valor associado ao par específico anomalia/sala. Quando o formulário é gerado, verifica-se qual o valor que o par contém, sendo este colocado na célula. Do mesmo modo que nas células, a cada divisão criada é-lhe também atribuído, dinamicamente, um nome. Este vai ser igual ao “id” da anomalia que esta divisão suporta, permitindo, por intermédio deste, identificar qual a divisão a mostrar.

Anomalias

Salas	Id anomalia = a	Id anomalia = b	Id anomalia = c
Id sala = 1	Id = 1xa	Id = 1xb	Id = 1xc
Id sala = 2	Id = 2xa	Id = 2xb	Id = 2xc
Id sala = 3	Id = 3xa	Id = 3xb	Id = 2xc
...			

Figura 5.6 - Matriz de atribuição das identidades às células do formulário.

Às células que utilizam as divisões escondidas para registar os valores é atribuída uma classe “mostra_divisão” (no pseudocódigo) que o “*Dynamic Script*” utiliza quando detecta o evento de selecção sobre a célula, não sendo portanto necessário inserir o código nos *templates* para esse efeito. O “*Dynamic Script*” quando identifica a selecção vai efectuar várias operações. Em primeiro lugar, é necessário remover o efeito de memória que possa, eventualmente, existir no formulário. Por exemplo, um utilizador pode seleccionar uma célula e logo de seguida (sem efectuar nenhuma operação sobre a divisão) seleccionar uma outra, sendo por isso necessário retirar o realce de todas as células e esconder todas as divisões. Uma vez que não se sabe qual das células foi seleccionada anteriormente, não se pode esconder apenas a divisão que se encontra visível (se estiver alguma) e retirar o realce de apenas uma célula. De igual modo torna-se necessário “limpar” os formulários da divisão que se vai utilizar porque no caso de ter sido inserida uma anomalia numa outra célula em branco, a divisão vai conter os dados inseridos anteriormente. Tal não aconteceria numa operação de edição, pois a divisão era posteriormente preenchida com os valores que se encontram em base de dados. Depois destas pequenas inicializações atribui-se o realce à célula seleccionada executando um *script* de posicionamento. Este *script* vai detectar em que local foi efectuado o accionamento do botão do rato (posição na janela do navegador), qual o tamanho da janela e qual a posição das barras de posicionamento (“*scrollbars*”). Na posse destes dados, vai estimar a posição óptima para o posicionamento da divisão. Esta estimacção tem, dinamicamente, em conta o tamanho da divisão, uma vez que este varia consoante a informação que a divisão contém, assim como da formatação atribuída (caso não fossem tomados em conta todos estes factores e apenas a posição do rato a divisão poderia ser colocada fora da janela).

➤ *Dynamic Script*” - “hidden_div_ajax_submit”

Na presença da divisão o utilizador pode preencher os campos apresentados, inserir alterações ou cancelar a operação. Se cancelar a operação, a divisão é escondida. Se por outro lado for efectuado o envio dos dados, o “*Dynamic Script*” detecta o evento “*submit*” e chama o “*Action Script*” (“*action_InsertUpdateAnomalyValues.php*”). Este último é responsável pela inserção dos valores na base de dados, como descrito anteriormente. Toda esta dinâmica desenrola-se assincronamente. Durante este processo efectua-se também uma verificação dos dados a enviar. Por exemplo, no caso das anomalias com tipo de controlo numérico verifica se os dados são realmente do tipo numérico.

5.3 - Detalhes de implementação

Este módulo foi desenvolvido com o intuito de ser o mais dinâmico e flexível possível, tal como foi especificado nos pontos anteriores.

Para uma melhor compreensão e visualização dos conceitos aplicados descreve-se, com algum pormenor o funcionamento de um formulário, sendo depois demonstrado como estes podem facilmente ser criados e alterados.

A **Figura 5.7** exemplifica a interface que o utilizador vê para um formulário de inserção de anomalias de sanitários. O formulário inicialmente apresenta-se completamente em branco.

Quali FEUP (sair)

STM | Serviços Gerais (início)

Sanitários

Edifícios

Piso	Sala N°	Falta Secador das Mãos	Tambor Papel Higiénico	Saboneteiras	Suporte Papel Inter-folhas
Piso 0	093	☐			
Piso 0	094	☐			
Piso 0	095	☐			
Piso 0	096	☐			
Piso 0	099	☐			
Piso 1	194	☐			
Piso 1	195	☐			
Piso 2	294	☐			
Piso 2	295	☐			

Voltar

Figura 5.7 - Formulário de Controlo de Anomalias - Sanitários

Para inserir uma anomalia selecciona-se uma das células. Como especificado no capítulo anterior uma divisão escondida é então dinamicamente tornada visível e colocada numa posição adequada (**Figura 5.8**).

Quali FEUP (sair)

STM | Serviços Gerais (início)

Sanitários

Quali FEUP

Edifícios

A	B	C	D	E	F	G	H	I	J	L	M
Piso	Sala N°	Falta Secador das Mãos	Tambor Papel Higiénico	Saboneteiras	Suporte Papel Inter-folhas						
Piso 0	093	☐									
Piso 0	094	☐									
Piso 0	095	☐									
Piso 0	096	☐									
Piso 0	099	☐									
Piso 1	194	☐									
Piso 1	195	☐									
Piso 2	294	☐									
Piso 2	295	☐									

Tambor Papel Higiénico

Anomalias:

- ☐ Encaixe partido
- ☐ N/T porta n/ mola
- ☐ Não tem mola
- ☐ Avariado
- ☐ Mola avariada
- ☐ Dobradiça partida
- ☐ Não tem porta

Remover registo de anomalia:

- ☐ Apagar registo de anomalia

Cancelar Cancelar

Voltar

Figura 5.8 - Inserção de valores no formulário de Controlo de Anomalias - Sanitários

A célula seleccionada, assim como as células com a designação da sala e piso mudam de cor para uma melhor identificação da actual selecção. A divisão que é tornada visível, por sua vez, identifica a anomalia contendo o nome da anomalia em cabeçalho. Na criação destes identificadores visuais vários factores foram tomados em conta. Por exemplo, quando a lista se torna demasiado grande o cabeçalho do formulário deixa de estar visível logo é necessário um modo de identificar a anomalia em causa, pelo que foi adicionado o nome da anomalia às divisões. Por outro lado, embora as linhas do formulário alternem de cor, com um grande número de linhas podia-se tornar difícil identificar ao certo qual a sala em causa. Para solucionar esse possível problema fez-se então sobressair não apenas a célula seleccionada mas também o número e piso da sala.

A divisão tornada visível terá os controlos adequados para a inserção da anomalia dependendo do tipo de controlo especificado para esta. Foram desenhadas três divisões diferentes, uma para cada tipo de controlo identificado, podendo o controlo por “checkbox” ficar inserido directamente no próprio formulário (Figura 5.9).

Sim/Não - Checkbox

☒

☐

Inserção Livre - Text Input

Outros Danos

Descrição da anomalia:

Confirmar Cancelar

Numérico - Text Input

Tomadas de Rede

Número de ocorrências:

Confirmar Cancelar

Múltipla-Escolha - Radio

Tambor Papel Higiénico

Anomalias:

- ☐ Encaixe partido
- ☐ N/T porta n/ mola
- ☐ Não tem mola
- ☐ Avariado
- ☐ Mola avariada
- ☐ Dobradiça partida
- ☐ Não tem porta

Remover registo de anomalia:

- ☐ Apagar registo de anomalia

Confirmar Cancelar

Figura 5.9 - Mecanismos de controlo de inserção de anomalias (divisões escondidas).

Quando uma anomalia é inserida, automaticamente a divisão é escondida e o valor inserido no formulário. Ao mesmo tempo este é inserido na base de dados.

Quando se faz uma inserção livre de uma anomalia, como o espaço do formulário é limitado, apenas fica visível parte da descrição. Para consultar a descrição completa deve-se seleccionar a célula aparecendo então uma divisão com a descrição completa (**Figura 5.10**). Para apagar ou alterar uma inserção basta seleccionar uma célula já preenchida ou uma “checkbox”.

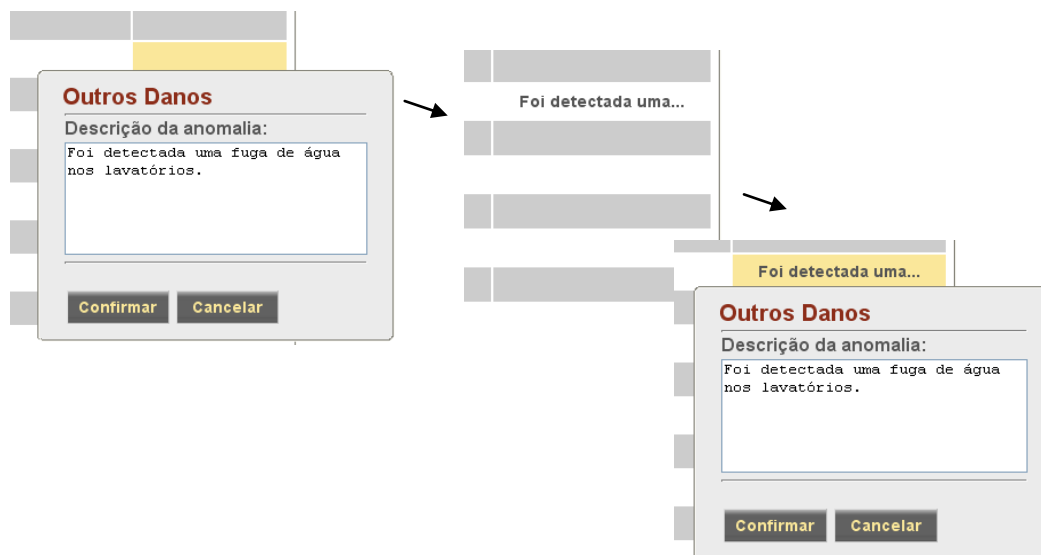


Figura 5.10 - Consultar a descrição completa de uma anomalia de inserção livre e/ou alterar.

Na inserção de uma anomalia do tipo numérico esta detecta se o valor inserido é realmente um número, caso contrário emite um aviso para o utilizador corrigir a sua inserção (**Figura 5.11**).

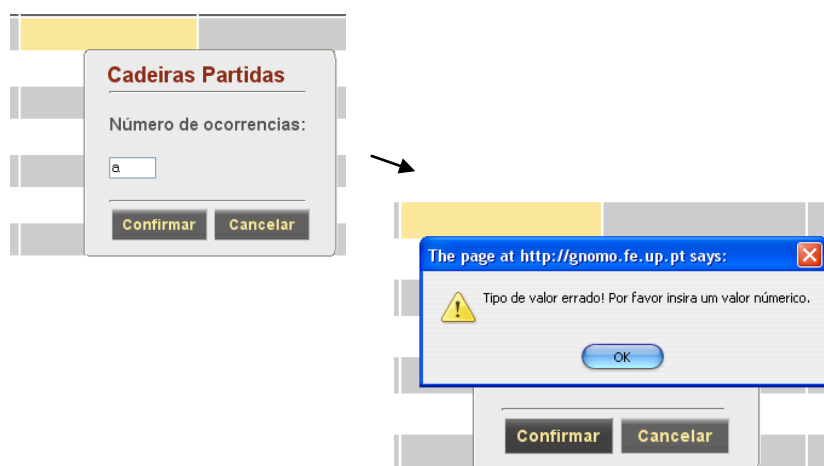


Figura 5.11 - Inserção de um valor numérico no formulário.

Os formulários foram desenhados para serem dinâmicas não só em termos de interface mas também no processo de alteração/adicação. Imaginemos agora que se necessita de inserir

uma nova anomalia no formulário. Se isso acontecer não haverá necessidade de nenhuma programação; basta na base de dados inserir essa nova anomalia, indicar a que tipo de formulário esta pertence (por exemplo “sanitários”) e qual o tipo de controlo, (se o tipo de controlo for de escolha múltipla deve-se indicar quais as opções) e refrescar a página. Automaticamente será criada a nova coluna da anomalia com o tipo de controlo escolhido e pronta a utilizar, sendo os valores inseridos nesta guardados em base de dados. Tudo isto de um modo simples e transparente.

A título de exemplo mostra-se a inserção de duas novas anomalias no formulário descrito na **Figura 5.1**, uma de múltipla escolha chamada “Anomalia Teste 1” com as opções “Opção 1” e “Opção 2” e uma outra do tipo “checkbox” chamada “Anomalia Teste 2”. Como resultado, após refrescar a página, aparece-nos o formulário da **Figura 5.1**.

The screenshot displays the 'Sanitários' (Toilets) management interface. At the top, there's a header with 'Quali FEUP' and 'STM | Serviços Gerais (início)'. Below this, the title 'Sanitários' is prominently displayed. The main area is titled 'Edifícios' and contains a table with columns for 'Piso', 'Sala N°', and various maintenance items: 'Falta Secador das Mãos', 'Tambor Papel Higiênico', 'Saboneteiras', 'Suporte Papel Inter-folhas', 'Anomalia Teste 1', and 'Anomalia Teste 2'. The table lists several floors (Piso 0, 1, 2) and room numbers (e.g., 093, 094, 095, 096, 099, 194, 195, 294, 295). A modal window titled 'Anomalia Teste 1' is open, showing options for 'Opção 1' and 'Opção 2'. The modal also includes a section for removing the anomaly record with a radio button for 'Apagar registo de anomalia' and 'Confirmar'/'Cancelar' buttons.

Figura 5.12 - Exemplo da facilidade de edição de um formulário.

No caso de se pretender apagar uma anomalia bastará removê-la da base de dados. Para alterar as opções de uma anomalia de escolha múltipla basta fazer essa alteração na base de dados e automaticamente os formulários adaptam-se, ou seja, qualquer alteração em termos dos dados de um formulário pode ser feita directamente na base de dados.

O processo de criação de um novo formulário deve ser tão transparente quanto o de edição. Para tal na implementação deste módulo teve-se o cuidado de estruturar o código do modo mais simples e dinâmico possível, utilizando uma arquitectura completamente modular, permitindo que a criação de novos formulários reutilize a maior parte do código dos diversos blocos.

Para criar um novo formulário é necessário então definir na base de dados qual o seu tipo de novo formulário na tabela “Form_user” e as anomalias que este vai conter. Ao criar um novo “Main Script” define-se esse novo formulário e as salas que se vão utilizar. Se a estrutura definida anteriormente para o template da tabela se mantiver, basta então adicionar esse template e a nova tabela estará pronta a ser utilizada.

Sector

Todos A B C

Piso 0	Aberta aula	Piso 1	Aberta aula	Piso 2	Aberta aula	Piso 3	Aberta aula
B004	☐	B101	☐	B201	☐	B301	☐
B005	☐	B102	☒	B202	☐	B302	☒
B006	☒	B103	☐	B203	☐	B303	☐
B007	☐	B104	☐	B204	☐	B304	☐
B008	☐	B105	☐	B205	☒	B305	☐
B009	☐	B106	☐	B206	☐	B306	☐
B010	☒	B107	☒	B207	☐	B307	☐
B011	☐	B108	☒	B208	☐	B308	☐
B012	☒			B209	☐	B309	☐
B013	☐			B210	☒	B310	☒
				B211	☐	B311	☐
				B212	☐	B312	☐
				B213	☐	B313	☐
						B314	☐

Figura 5.13 - Exemplo da criação de um novo formulário com uma estrutura nova.

Em qualquer caso, se for preciso criar uma nova estrutura, cria-se sempre um novo template e todo o resto dos blocos podem ser reutilizados. Como exemplo numa fase final da implementação surgiu a necessidade de uma nova estrutura (**Figura 5.8**) tendo sido esta rapidamente implementada, pois concentrando-se todos os esforços no desenho da estrutura, reutilizando-se os efeitos, formatação (CSS), envio e acesso a dados. Os dados para este novo formulário foram inseridos na base de dados, e foi criado um “Main Script” que utilizou esses dados, ficando a página pronta.

Capítulo 6

Conclusões e Perspectivas de Desenvolvimento

Este capítulo apresenta uma síntese dos principais resultados alcançados. São enunciadas as principais dificuldades encontradas e apontadas perspectivas de desenvolvimento futuro para esta aplicação.

Em termos de especificação e análise foram alcançadas conclusões que se revelaram extremamente úteis, além de se terem revelado como um importante ponto de partida para o processo de implementação. O tempo dispendido durante todo o processo de análise forneceu uma base sólida a seguir, incutindo uniformidade e estruturação no desenvolvimento da aplicação. A organização do código foi a grande beneficiada, visto que, foram definidos métodos de navegação, passagem de dados e atribuição de nomenclaturas às variáveis definidas, antes de se iniciar o processo prático. A arquitectura proposta foi testada com grande sucesso, revelando-se bastante sólida e garantindo as principais funcionalidades propostas na elaboração deste trabalho (uma estrutura flexível, modular e escalável). Atendendo a estes aspectos, considera-se que o trabalho desenvolvido foi bem sucedido.

Em relação ao protótipo prévio foram alcançadas grandes melhorias e desenvolvimentos, nomeadamente ao nível de reengenharia do código, análise detalhada dos módulos (particularmente do módulo de gestão de tarefas) e desenvolvimento do módulo flexível de anomalias. A inclusão de novas tecnologias como jQuery e Smarty (o jQuery em termos de efeitos de apresentação e inclusão do Ajax e o Smarty em termos de organização e estruturação) permitiram a reestruturação das funcionalidades e formas de trabalhar que não estavam presentes.

Em qualquer projecto é normal que surjam dificuldades e este não foi excepção. Sendo crítico em relação ao trabalho desenvolvido aponta-se como a principal dificuldade encontrada o factor tempo. O aprofundamento sobre os temas abordados nem sempre foi tão elaborado como o esperado, visto ser incompatível com a janela temporal atribuída para o desenvolvimento desta aplicação. Como tal, alguns aspectos inicialmente previstos para estudo foram mais ligeiramente estudados em detrimento de uma melhor análise e compreensão de outros.

Em termos práticos, as principais dificuldades encontradas foram ligadas ao processo de implementação. As soluções tecnológicas escolhidas revelaram-se muito poderosas e

promissoras. Durante a implementação destas soluções constatou-se, no entanto, um certo atrito inicial na sua utilização. Estas contêm peculiaridades próprias às quais é necessário adaptar-se, especialmente quando as necessidades entram em detalhes fora da utilização normal. Estas dificuldades foram no entanto superadas com sucesso servindo como experiência e referência para uma melhor preparação futura.

6.1 - Perspectivas de desenvolvimento

Existem funcionalidades adicionais que estão previstas para desenvolvimento a curto prazo, trazendo benefícios à utilização da aplicação.

Nomeadamente a criação de interfaces, assentes na arquitectura proposta, através das quais, utilizadores com as devidas permissões poderiam, sem intervenção de um técnico do QualiFeup, alterar os formulários dinâmicos para inserção de anomalias (adicionar, retirar ou alterar anomalias que estes achem necessárias). Estas interfaces devem permitir, não só a edição e alteração, mas também a criação de novos formulários assentes nas estruturas pré-definidas. É de igual modo vantajosa a introdução de mecanismos de pesquisa a estes formulários, possibilitando a filtragem de anomalias, consulta de número de ocorrências ou mesmo um cruzamento entre diferentes anomalias para uma mesma sala, contidas em diferentes formulários. Estes mecanismos terão de ser implementados mantendo as funcionalidades existentes, ou seja, dinâmicos e flexíveis.

A criação de um suporte para mobilidade, ou seja, uma contraparte desta aplicação desenvolvida para PDA's, é uma futura perspectiva de desenvolvimento. Esta nova aplicação terá de ser autónoma, permitindo aos utilizadores trabalhar fora do alcance da rede WiFi da faculdade. Embora exista a possibilidade de se adaptar a aplicação já existente para a interface destes dispositivos, deverá ser desenvolvida uma aplicação de raiz. Esta nova aplicação não terá de possuir todas as funcionalidades da “aplicação mãe”. Numa primeira fase, a nova aplicação será especialmente vantajosa para o módulo de Gestão de Estacionamento, permitindo aos utilizadores, por intermédio de um PDA, efectuar os registos no “campo”. Esta deve conter uma pequena base de dados própria, onde serão guardados os dados utilizados. O sincronismo com a base de dados “geral” será efectuado sempre que o utilizador estiver dentro do alcance da rede WiFi. Se este processo se desencadeia automaticamente ou a pedido do utilizador, deverá ser alvo de análise com mais detalhe numa fase posterior. Será necessário adicionar um novo bloco à arquitectura desenvolvida para permitir esta interacção, bem como novas funções da camada de acesso à base de dados. Esta comunicação deverá ser efectuada através do uso de “*Web Services*”, uma vez que estes funcionam como funções remotas que podem ser invocadas por outras máquinas utilizando protocolos “*standard*” independentes da plataforma ou software que estão a correr. A arquitectura para esta implementação e a sua conjugação com a já implementada foi já pensada e estruturada e apresenta-se na **Figura 6.1**.

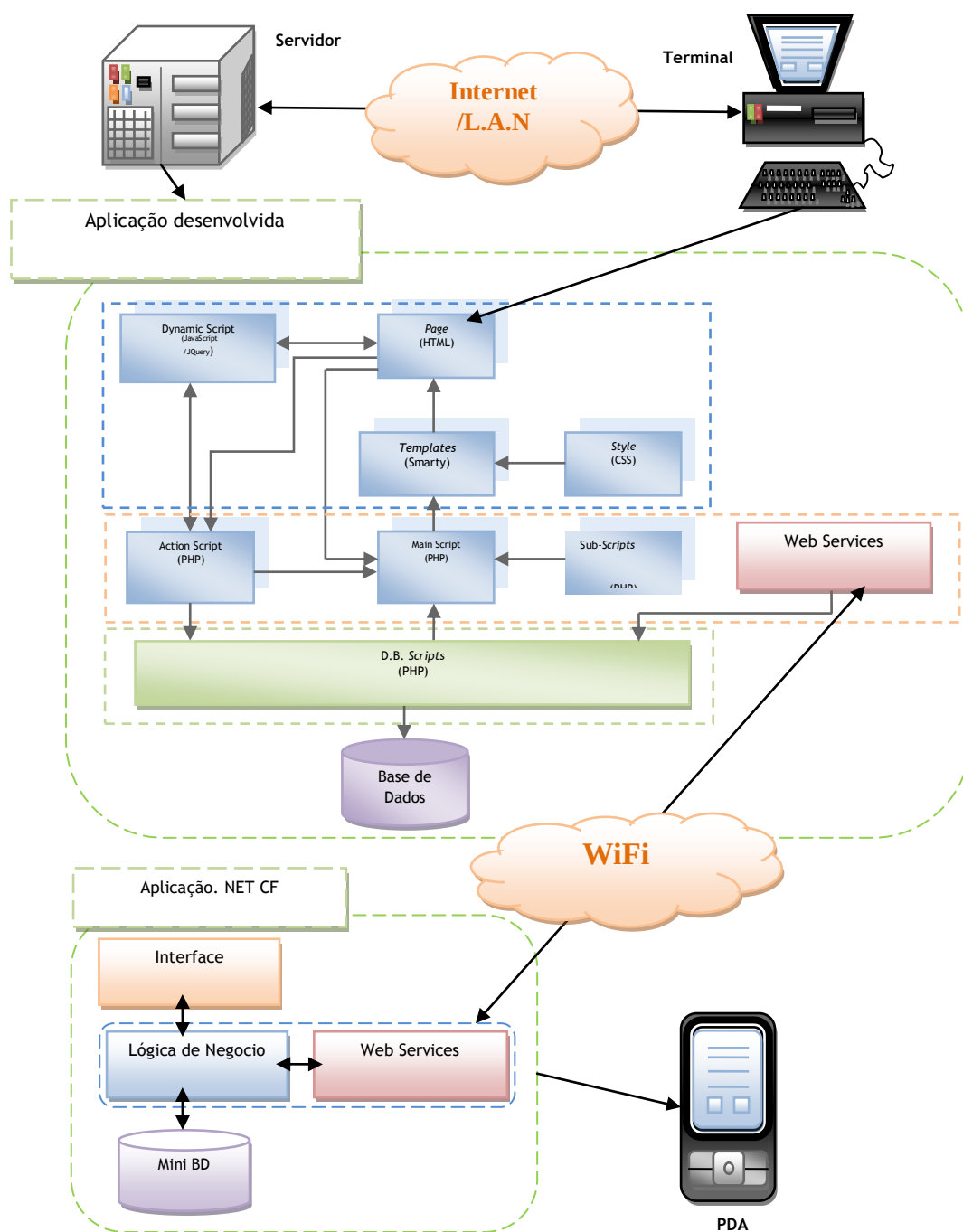


Figura 6.1 - Arquitectura com suporte para mobilidade

Referências

- [1] Bibeault, B. e Katz, Y. (2008). *jQuery in Action*. Manning Publications Co
- [2] Crane, D., Pascarello, E. e James, D. (2006). *Ajax in action*. Manning Publications Co
- [3] Gilmore, W.J. (2006). *Beginning PHP5 and MySQL 5 - From Novice to Professional*. 2th Edition
- [4] Goodman, D. e Morrison, M. (2004). *JavaScript Bible*. 5th Edition, Wiley Publishing Inc.
- [5] Kalbach, J. (2007). *Designing Web Navigation*. 1th Edition, O'Reilly Media Inc.
- [6] Niederst, J. (1999). *Web Design in a nutshell*. 1th Edition, O'Reilly Media Inc.
- [7] Nauman, R. (2008). Getting started with PHP and Smarty. *Crossroads - The ACM Student Journal*. 143: 9-15
- [8] Schmitt, C. (2006). *CSS Cookbook*. 2th Edition, O'Reilly Media Inc.
- [9] Sklar, D. (2004). *Learning PHP 5*. 1th Edition, O'Reilly Media Inc.
- [10] Welling, L. e Thomson, L. (2001). *PHP and MySQL Web Development*. 1th Edition, Sams Publishing
- [11] Enode (2002). *Model-View-Controller pattern*. Acedido em 14 Abril de 2008, em: <http://www.enode.com/x/markup/tutorial/mvc.html>
- [12] HTML Code Tutorial (2007). *HTML Quick list*. Acedido em 16 Abril de 2008, em: <http://www.htmlcodetutorial.com/quicklist.html>
- [13] Infowester (2003). Linguagem PHP. Acedido em 26 de Julho de 2008, em: <http://www.infowester.com/php.php>
- [14] JQuery (2008). Acedido em 11 Abril de 2008, em: <http://jquery.com/>
- [15] Macoratti, J. *Padrões de projecto: O modelo MVC - Model View Controller*. Acedido em 14 Abril de 2008, em: http://www.macoratti.net/vbn_mvc.htm
- [16] Marston, T. (2007). *The model-view-controller (MVC) design pattern for PHP*. Acedido em 14 Abril de 2008, em: <http://www.tonymarston.net/php-mysql/model-view-controller.html>
- [17] Microsoft Developer Network (2008). Acedido em 16 Abril de 2008, em: [http://msdn.microsoft.com/en-us/library/ms533050\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms533050(VS.85).aspx)

- [18] PostgreSQL (2008). About. Acedido em 26 de Julho de 2008, em: <http://www.postgresql.org/about/>
- [19] Quirkmode (2007). *Event properties*. Acedido em 24 Abril de 2008, em: http://www.quirkmode.org/js/events_properties.html
- [20] Rocha, A. (2002). *O essencial dos sistemas de informação*. Acedido em 27 Abril de 2008, em: <http://www2.ufp.pt/~amrocha/lecciona.htm>
- [21] Smarty Template Engine (2008). Acedido em 11 Abril de 2008, em: <http://www.smarty.net/>
- [22] Stump, Joe (2003). *Smarty: A closer look*. Acedido em 11 Abril de 2008, no Web site da: Zend Developer Zone: <http://devzone.zend.com/article/1257-Smarty-A-closer-look>
- [23] *Tutorial JQuery - Introduction*. Disponível em <http://www.istomesmo.com/en/2008/02/27/tutorial-jquery-introduo-com-2-exemplos/>
- [24] Visual jQuery 1.1. Acedido em 14 Abril de 2008, em: <http://visualjquery.com/1.1.2.html>
- [25] Wikipidia (2008). Cascading Style Sheets. Acedido em 26 de Julho de 2008, em: http://en.wikipedia.org/wiki/Cascading_Style_Sheets
- [26] W3 Schools (2008). Acedido em 22 de Abril de 2008, em: <http://www.w3schools.com/>

Anexo A

- Casos de Uso Gerais:

Caso de Uso
Fazer autenticação no sistema (“ <i>Login</i> ”)
Actores
Utilizador, Administrador
Pré-Condições
Sequência de Acções
1. No menu inicial seleccionar por inspecção visual o nosso perfil, accionando-o por intermédio de um menu de fotografias 2. Caso necessário fornecer senha de autenticação 3. Accionar o botão “Entrar”

Caso de Uso
Sair do sistema (“ <i>Logout</i> ”)
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. A qualquer altura accionar a sua fotografia ou o botão “Sair”
Nota
Ao manter-se inactivo no sistema durante um tempo superior a 5 minutos é feito o logout automático por razões de segurança.

- Casos de Uso Módulo Gestão de Avisos

Caso de Uso
Consultar Lista de Avisos do próprio dia ou anteriores
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Após validação no sistema é nos apresentado dois formulários onde é possível consultar os avisos 1.1 O formulário “Últimos avisos” apresenta os avisos do dia em qualquer estado (Para vigilantes, prioritários e normais) 1.2 O formulário “Avisos anteriores” apresenta os avisos anteriores (Para vigilantes, prioritários e normais)
Nota
Por intermédio de botões é possível filtrar a lista de entre todos os avisos, de avisos prioritários e ainda dentro deste se são destinados a vigilantes ou não.

Caso de Uso
Consultar Detalhes de um Aviso
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor encontra-se a visualizar a lista de avisos.
Sequência de Acções
1. Seleccionar o aviso que se pretende consultar os detalhes da lista por intermédio do accionamento do botão esquerdo do rato 2. Uma nova página é carregada com os detalhes

Caso de Uso
Criar um novo aviso
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. No formulário “Últimos avisos” accionar “novo aviso” 2. Inserir descrição do aviso no campo “Descrição do Aviso” 3. (Opcional) Inserir uma observação ao aviso no campo “Observações” 4. Definir prioridade do aviso, seleccionar a opção “Sim” ou “Não” no formulário “Marcar como prioritário?” 5. Definir a atribuição do aviso a vigilante, seleccionar a opção “Sim” ou “Não” no formulário “Destinado a vigilante” 6. Accionar o botão “Registar Aviso”

Caso de Uso
Alterar um aviso
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema
O actor seleccionou um aviso para visualizar os detalhes deste.
Sequência de Acções
1. Accionar o botão “Alterar Aviso” 2. (Opcional) Alterar a descrição do aviso no campo “Descrição do Aviso” 3. (Opcional) Alterar as observações ao aviso no campo “Observações” 4. (Opcional) Alterar prioridade do aviso, seleccionar a opção “Sim” ou “Não” no formulário “Marcar como prioritário?” 5. (Opcional) Alterar a atribuição do aviso a vigilante, seleccionar a opção “Sim” ou “Não” no formulário “Destinado a vigilante” 6. Accionar o botão “Guardar Alterações”

Caso de Uso
Apagar aviso
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor seleccionou um aviso para visualizar os detalhes deste.
Sequência de Acções
1. Accionar o botão “Apagar Aviso” 2. Reconfirmar a intenção de apagar no menu, “Tem a certeza que deseja apagar?” accionando o botão “Ok”

Caso de Uso
Imprimir aviso
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor seleccionou um aviso para visualizar os detalhes deste.
Sequência de Acções
1. Accionar o botão “Versão para Impressão” 2. Accionar o botão com formato de impressora

- Casos de Uso Módulo Gestão Estacionamento

Caso de Uso
Consultar Lista de Registos de Estacionamento Recentes
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Estacionamento” no formulário “Registos” 2. Um formulário “Últimos registos” lista os registos mais recentes

Caso de Uso
Consulta do Histórico de Registos de Estacionamento (Lista)
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor seleccionou “Estacionamento” no formulário “Registos”.
Sequência de Acções
1. Accionar o botão “Consultas” 2. Seleccionar critérios de consulta 2.1 Por matrícula, por número de ocorrências, por intervalo temporal (datas), ou por conjugação dos anteriores 3. Accionar o botão “Consultar”

Caso de Uso
Consultar Detalhes de um Registo de Estacionamento
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor encontra-se a visualizar a listagem de registos.
Sequência de Acções
1. Seleccionar o registo que se pretende consultar os detalhes da lista por intermédio do accionamento do botão esquerdo do rato 2. Uma nova página é carregada com os detalhes

Caso de Uso
Alterar Registo de Estacionamento
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou um registo específico, e encontra-se nos detalhes do registo.
Sequência de Acções
1. Accionar o botão “Alterar Registo” 2. Alterar os campos necessários 3. Accionar o botão “Guardar Alterações”

Caso de Uso
Apagar Registo de Estacionamento
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou um registo específico, e encontra-se nos detalhes do registo.
Sequência de Acções
1. Accionar o botão “Apagar Registo” 2. Reconfirmar a intenção de apagar no menu pop-up “Tem a certeza que deseja apagar?” accionando o botão “Ok”

Caso de Uso
Imprimir Folha de Aviso para Registos de Estacionamento
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou um registo específico, e encontra-se nos detalhes do registo.
Sequência de Acções
1. Accionar o botão “Folha de Aviso” 2. Accionar “Ficheiro” na barra de menu do browser 3. Seleccionar a opção “Imprimir”

Caso de Uso
Adicionar Registo de estacionamento
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou “Estacionamento” no formulário “Registos”.
Sequência de Acções
1. No formulário “Últimos registos” accionar “novo registo” 2. Inserir matrícula no campo “Matricula” 3. Inserir marca no campo “Marca” 4. Inserir a data no campo “Data” 5. Inserir hora de primeira inspecção no campo “Hora de controlo 1” 6. Inserir hora de segunda inspecção no campo “Hora de controlo 2” 7. (Opcional) Inserir cor do veículo no campo “Cor” 8. (Opcional) Inserir local no campo “Local” 9. (Opcional) Inserir observações ao registo no campo “Observação” 10. Accionar o botão “Registar”

- **Casos de Uso Módulo Gestão Anomalias**

Caso de Uso
Consultar Lista de Registos de Anomalias (Activos e Todos)
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Anomalias” no formulário “Registos” 2. Um formulário “Anomalias” lista os registos activos 2.1 Por intermédio de um botão pode-se seleccionar entre registos activos (por resolver) e todos os registos registados no sistema

Caso de Uso
Consulta de Registos de Anomalias por Parâmetros
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. Seleccção de “Anomalias” no formulário “Registos”.
Sequência de Acções
1. Accionar o botão “Consultas” 2. Seleccionar critérios de consulta 2.1 Inserindo o número do “<i>Trouble Ticket</i>” ou por intervalo temporal (datas) 3. Accionar o botão “Consultar”

Caso de Uso
Consultar Tabelas de Registo de Anomalias (Sanitários ou Mobiliário)
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. Seleccção de “Anomalias” no formulário “Registos”.
Sequência de Acções
1. Accionar o botão “Sanitários” ou “Mobiliário” 2. No formulário (“Sanitários” ou “Mobiliário”) utilizar os botões para filtrar a informação entre edifícios, pisos ou sectores

Caso de Uso
Consultar Detalhes de um Registo de uma Anomalia
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. O actor encontra-se a visualizar a listagem de registos.
Sequência de Acções
1. Seleccionar o registo que se pretende consultar os detalhes da lista por intermédio do accionamento do botão esquerdo do rato 2. Uma nova página é carregada com os detalhes

Caso de Uso
Adicionar Registo de Anomalia no sistema de gestão dos STM
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou “Anomalias” do formulário “Registos”.
Sequência de Acções
1. No formulário “Anomalias” accionar “nova anomalia” 2. No formulário “Registo de anomalias” accionar “Anomalia” 3. Inserir a descrição da anomalia na caixa de texto “Descreva a anomalia” 4. Accionar “Registar”

Caso de Uso
Adicionar Registo de Anomalia no sistema de gestão dos STM
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou “Anomalias” do formulário “Registos”.
Sequência de Acções
1. No formulário “Anomalias” accionar “nova anomalia” 2. No formulário “Registo de anomalias” accionar “<i>Trouble Ticket</i>” 2.1 Inserir a descrição da anomalia na caixa de texto “Descreva a anomalia” 2.2 Registar o “<i>Trouble Ticket</i>” no SiFEUP accionando “<i>Trouble Ticket</i>” 2.3 Inserir o número do “<i>Trouble Ticket</i>” criado na caixa de texto “Nº do <i>Trouble Ticket</i>” 3. Accionar “Registar”

Caso de Uso
Adicionar um Registo de uma Anomalia numa tabela de Anomalias
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor encontra-se a visualizar uma tabela de anomalias (“Sanitários” ou “Mobiliário”).
Sequência de Acções
1. Accionar a célula da tabela que se pretende adicionar uma anomalia 2. No formulário de inserção da anomalia dependo do tipo de anomalia: 2.1 Seleccionar dentro das opções disponíveis a anomalia 2.2 Inserir na caixa texto a descrição ou valor da anomalia 3. Accionar “Confirmar”
Nota
Em anomalias com o tipo de controlo “Sim/Não” efectuar apenas uma permutação entre os valores, não sendo necessário confirmação.
A qualquer altura é possível cancelar o processo accionando “Cancelar”.

Caso de Uso
Alterar um Registo de uma Anomalia no sistema de Gestão dos STM
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor seleccionou um registo específico, e encontra-se nos detalhes do registo.
Sequência de Acções
1. Accionar “Alterar Registo” 2. (Opcional) Modificar a descrição da anomalia na caixa de texto “Descrição da anomalia” 3. (Opcional) É possível efectuar o registo no SiFEUP caso ainda não tenha sido feito 4. (Opcional) Se foi feito o registo no SiFEUP deve-se inserir o número do “<i>Trouble Ticket</i>” na caixa de texto “<i>Trouble Ticket</i> Nº” 5. Accionar “Guardar Alterações”

Caso de Uso
Alterar Registo de uma Anomalia numa tabela de Anomalias
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor encontra-se a visualizar uma tabela de anomalias (“Sanitários” ou “Mobiliário”).
Sequência de Acções
1. Accionar a célula da tabela que se pretende alterar a anomalia 2. No formulário de inserção da anomalia dependendo do tipo de anomalia, seleccionar dentro das opções disponíveis a nova anomalia ou inserir na caixa texto nova descrição ou valor da anomalia 3. Accionar “Confirmar”
Nota
Em anomalias com o tipo de controlo “Sim/Não” efectuar apenas uma permutação entre os valores, não sendo necessário confirmação.
A qualquer altura é possível cancelar o processo accionando “Cancelar”.

Caso de Uso
Apagar Registo de uma Anomalia no sistema de Gestão dos STM
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor seleccionou um registo específico, e encontra-se nos detalhes do registo.
Sequência de Acções
1. Accionar “Apagar Registo” 2. Reconfirmar a intenção de apagar no menu, “Tem a certeza que deseja apagar?” accionando o botão “Ok”

Caso de Uso
Apagar Registo de uma Anomalia numa tabela de Anomalias
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema. O actor encontra-se a visualizar uma tabela de anomalias (“Sanitários” ou “Mobiliário”).
Sequência de Acções
1. Accionar a célula da tabela que se pretende apagar a anomalia 2. No formulário de inserção da anomalia dependo do tipo de anomalia, seleccionar nas opções disponíveis “Apagar registo de anomalia” ou apagar na caixa texto a descrição ou valor da anomalia 3. Accionar “Confirmar”
Nota
Em anomalias com o tipo de controlo “Sim/Não” efectuar apenas uma permutação entre os valores, não sendo necessário confirmação. A qualquer altura é possível cancelar o processo accionando “Cancelar”.

Caso de Uso
Arquivar Registo de uma Anomalia no sistema de Gestão dos STM
Actores
Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou um registo específico, e encontra-se nos detalhes do registo.
Sequência de Acções
2. Accionar “Arquivar Registo” 3. Reconfirmar a intenção de arquivar no menu, “Tem a certeza que deseja arquivar?” accionando o botão “Ok”

- Casos de Uso Gestão Tarefas

Caso de Uso
Consultar Lista de Tarefas do Dia e/ou Atrasadas
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Salas” no formulário “Tarefas” 2. É nos apresentado dois formulários onde é possível consultar as tarefas. 2.1 O formulário “Hoje” lista as tarefas do próprio dia 2.2 O formulário “Atrasadas” lista as tarefas atrasadas
Nota
Por intermédio de botões nestes formulários (“Todas” e “Vigilantes”) é possível filtrar a informação entre todas as tarefas e as apenas destinadas a vigilantes.

Caso de Uso
Consultar Lista de Tarefas da Semana ou Mês
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Salas” no formulário “Tarefas” 2. Accionar o botão “Tarefas da Semana” ou “Tarefas do Mês”

Caso de Uso
Consultar Histórico de Tarefas
Actores
Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Salas” no formulário “Tarefas” 2. Accionar o botão “Histórico de Tarefas” 3. No novo formulário “Consulta do histórico de tarefas” parametrizar a pesquisa: 3.1 Escolhendo a que funcionário se destinava a tarefa por intermédio de um menu de selecção (não escolher nenhum é igual a todos) 3.2 Escolhendo o tipo de operação que se pretende consultar (“Criação”, “Alteração”, “Alteração do Estado”, “Eliminação”, “Registo de Execução”), não escolher nenhum é igual a seleccionar todos 3.3 Escolhendo um intervalo temporal (datas)

Caso de Uso
Consultar detalhes de uma tarefa (Instância)
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor está a consultar uma listagem de tarefas.
Sequência de Acções
1. Seleccionar o aviso que se pretende consultar os detalhes da lista por intermédio do accionamento do botão esquerdo do rato 2. Uma nova página com é carregada com os detalhes

Caso de Uso
Consultar detalhes de uma Tarefa Periódica (Planeamento)
Actores
Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Salas” no formulário “Tarefas” 2. Accionar “Tarefas Periódicas” 3. Seleccionar a tarefa que se pretende consultar os detalhes da lista por intermédio do accionamento do botão esquerdo do rato 4. Uma nova página é carregada com os detalhes

Caso de Uso
Adicionar uma Tarefa Única (Instância)
Actores
Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Salas” no formulário “Tarefas” 2. Accionar “Nova Tarefa” 3. Definir a prioridade da tarefa no formulário “Prioritária” (Sim ou Não) 4. Definir atribuição a vigilante no formulário “Vigilante” (Sim ou Não) 5. Definir data para realização da tarefa no formulário “Data” (Hoje, Amanhã ou outra data) 6. (Opcional) Definir hora de execução 7. Inserir a descrição da tarefa no formulário “Descrição da tarefa” 8. (Opcional) Inserir observações à realização da tarefa no formulário “Observações” 9. Accionar “Confirmar”

Caso de Uso
Adicionar uma Tarefa Periódica

Actores
Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Salas” no formulário “Tarefas” 2. Accionar “Tarefas Periódicas” 3. Accionar “Nova Periódica” 4. Definir a prioridade da tarefa no formulário “Prioritária” (Sim ou Não) 5. Definir atribuição a vigilante no formulário “Vigilante” (Sim ou Não) 6. Definir data de execução, escolher dias a que a tarefa se deve realizar (Segunda-Feira a Domingo), definir intervalo de datas para tarefa periódica “De”, “Até” 7. (Opcional) Definir hora de execução 8. Inserir a descrição da tarefa no formulário “Descrição da tarefa” 9. (Opcional) Inserir observações à realização da tarefa no formulário “Observações” 10. Accionar “Confirmar”

Caso de Uso
Registar execução de uma tarefa
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
O actor seleccionou uma tarefa específica, e encontra-se nos detalhes dessa tarefa.
Sequência de Acções
1. Accionar “Alterar Tarefa” 2. No formulário “Tarefa executada com sucesso?” seleccionar a opção “Sim”, “Executei Parte” ou “Não” 3. Se seleccionada a opção “Sim” 3.1 (Opcional) Inserir observações à execução da tarefa em “Observações sobre a execução da tarefa” 4. Se seleccionada as opções “Executei Parte” ou “Não” 4.1 Inserir observações à não execução no formulário “Porquê?” 5. Accionar “Guardar Alterações”

Alterar dados de planeamento de uma tarefa (Instância)
Actores
Administrador
Pré-Condições
O actor está validado no sistema.
O actor seleccionou uma tarefa específica, e encontra-se nos detalhes dessa tarefa.
Sequência de Acções
1. Accionar “Alterar Tarefa” 2. (Opcional) Alterar prioridade da tarefa no formulário “Prioritária” (Sim ou Não) 3. (Opcional) Alterar atribuição a vigilante no formulário “Vigilante” (Sim ou Não) 4. (Opcional) Alterar a data de execução 5. (Opcional) Alterar hora de execução 6. (Opcional) Alterar descrição da tarefa no formulário “Descrição da tarefa” 7. (Opcional) Alterar observações no formulário “Observações” 8. Accionar “Guardar Alterações”

Caso de Uso
Alterar dados de planeamento de uma tarefa periódica
Actores
Administrador
Pré-Condições
O actor está validado no sistema.
O actor encontra-se a visualizar os detalhes de planeamento de uma tarefa periódica.
Sequência de Acções
1. Accionar “Alterar Tarefa” 2. (Opcional) Alterar prioridade da tarefa no formulário “Prioritária” (Sim ou Não) 3. (Opcional) Alterar atribuição a vigilante no formulário “Vigilante” (Sim ou Não) 4. (Opcional) Alterar a data de execução (dias de execução e intervalo de datas) 5. (Opcional) Alterar hora de execução 6. (Opcional) Alterar descrição da tarefa no formulário “Descrição da tarefa” 7. (Opcional) Alterar observações no formulário “Observações” 8. Accionar “Guardar Alterações”

Caso de Uso
Alterar registo de execução de uma tarefa
Actores
Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou uma tarefa específica, e encontra-se nos detalhes dessa tarefa.
Sequência de Acções
1. Accionar “Alterar Tarefa” 2. No formulário “Tarefa executada com sucesso?” alterar a escolha prévia 3. Se seleccionada a opção “Sim” 3.1 (Opcional) Inserir observações à execução da tarefa em “Observações sobre a execução da tarefa” 4. Se seleccionada as opções “Executei Parte” ou “Não” 4.1 Inserir observações à não execução no formulário “Porquê?” 5. Accionar “Guardar Alterações”

Caso de Uso
Apagar uma tarefa (Instância)
Actores
Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou uma tarefa específica, e encontra-se nos detalhes dessa tarefa.
Sequência de Acções
1. Accionar “Apagar Tarefa” 2. Reconfirmar a intenção de apagar no menu, “Tem a certeza que deseja apagar?” accionando o botão “Ok”

Caso de Uso
Apagar uma tarefa periódica (Planeamento)

Actores
Administrador
Pré-Condições
O actor está validado no sistema. O actor encontra-se a visualizar os detalhes de planeamento de uma tarefa periódica.
Sequência de Acções
1. Accionar “Apagar Tarefa” 2. Reconfirmar a intenção de apagar no menu, “Tem a certeza que deseja apagar?” accionando o botão “Ok”

Caso de Uso
Arquivar uma tarefa
Actores
Administrador
Pré-Condições
O actor está validado no sistema. O actor seleccionou uma tarefa específica, e encontra-se nos detalhes dessa tarefa.
Sequência de Acções
1. Accionar “Arquivar Tarefa” 2. Reconfirmar a intenção de arquivar no menu, “Tem a certeza que deseja arquivar?” accionando o botão “Ok”

Caso de Uso
Imprimir tarefas
Actores
Utilizador, Administrador
Pré-Condições
O actor está validado no sistema.
Sequência de Acções
1. Selecção de “Salas” no formulário “Tarefas” 2. Accionar o botão “Imprimir” 3. Seleccionar as tarefas a imprimir da listagem de tarefas “Tarefas para hoje” e “Tarefas atrasadas” 4. Accionar “Versão para Impressão” 5. Accionar o botão imprimir

Anexo B

A Figura 4.2 do capítulo 4 em tamanho A3 para uma melhor visualização.

