IWBDC – Interface WAP para Base de Dados Clínica

André Estevão Bexiga, Filipe Miguel Augusto

Resumo - Este artigo aborda o desenvolvimento de um modelo de interface multi-plataforma decorrente da implementação de infra-estruturas Cliente-Servidor baseadas no protocolo *WAP (Wireless Access Protocol)* que permitam a comunicação entre terminais (móveis ou fixos), e um sistema de informação clínica.

Elaborado no âmbito do projecto, *IWBDC – Interface WAP para Base de Dados Clínica*, sob o contexto actual de *Internet Móvel Como Mercado Emergente*, pretende ser uma interface de utilização de um sistema através de ambiente móvel, tal como dispositivos *WAP* e *PDA's*, visando também criar as mesmas estruturas para ambiente fixo sob o protocolo *HTTP (HiperText Transfer Protocol)*.

Os serviços proporcionados por tais infra-estruturas deverão orientar-se para aspectos de visualização de informação e tarefas de gestão do sistema.

A tecnologia *XML*, *XSL* foi utilizada para criação e formatação dinâmica de conteúdos de acordo com direitos de acesso para diferentes perfis de utilizadores. Regras rígidas de segurança e políticas de acesso foram implementadas assegurando uma transmissão segura de dados através da Internet. O modelo de aplicação implementado foi desenvolvido para interagir com uma base de dados clínica já existente, sendo no entanto facilmente ajustável a outros tipos diferentes de aplicações.

Na fase final da implementação, aspectos de usabilidade de interface foram então alvo de um cuidadoso estudo, tentando conjugar a natureza dos serviços de informação clínica e as capacidades de visualização e interacção dos diferentes tipos de terminais.

Abstract - **This article describes the development of a multi-platform interface model as an implementation of Client-Server infrastructures based in *WAP (Wireless Access Protocol)* protocol, to achieve communication between mobile and fixed terminals, and a system of clinical information.**

Elaborated in conjunction with the project, *IWBDC – Interface WAP Para Base de Dados Clínica (WAP Interface for Clinical Database)*, under the actual context market of *Mobile Internet As A Emergent Market*, intends to be an interface to a database system through an mobile environment, such as *WAP* enabled devices and *PDA's*, and also aiming to create the same structures in a fixed environment using *HTTP (Hypertext Transfer Protocol)* protocol.

The supplied services for such infrastructures will have to be orient for visualization aspects of the information and system management tasks.

The *XML*, *XSL* technology was employed for dynamic content creation and formatting, according to the access rights of different user profiles. Strict security and access control policy were implemented to ensure secure transmission of medical data through the Internet. The implemented model was designed to collaborate with existing clinical patient record systems and to be easily adjustable to different types of applications.

Aspects of interface usability had also been matter of study in the implementation fase, trying to conjugate the nature of the services of clinical information and the capacities of visualization and interaction of the different types of terminals.

I. INTRODUÇÃO

A *Internet móvel* é um fenómeno que só agora começa a ter a devida atenção. Os dispositivos mais frequentemente usados para aceder a esta rede são principalmente os telefones móveis e computadores de bolso referidos ultimamente como *PDA's (*equipados com interface de comunicação por rádio frequência). A nova geração de computadores pessoais ficará certamente entre os *PDAs* e os *notebooks* no respeita à portabilidade e capacidade de visualização.

Figura 1 – Dispositivos Móveis

Muito do conteúdo existente na *Internet* não é adequado aos pequenos ecrãs dos dispositivos móveis e também às baixas taxas de transferência existentes nas redes sem fios. Resulta então que aplicações e serviços para o ambiente móvel podem ser muito diferentes daqueles mais frequentemente usados na Internet hoje em dia.

Assim, faz cada vez mais sentido sistemas multi-plataforma capazes de fornecer serviços perfeitamente adequados às características e capacidades de cada, independentemente do sistema que tenha solicitado a aplicação.

II. ENQUADRAMENTO E PERSPECTIVA DE MERCADO

Em 2002, o número de dispositivos com capacidades *WAP (Wireless Application Protocol)* (na maioria são telefones móveis) usados para aceder à Internet excedeu o número de dispositivos com acesso convencional.

Figura 2 – Palm OS organizer

Também o mercado dos PDA após varias experiências sucedidas mal no início dos anos 90 começou a aumentar já no final dessa década com o sucesso nos E.U.A. dos *"Palm OS organizers"*, originalmente da *Palm Computing* e agora também dos seus licenciados (incluindo a *Handspring*, *IBM e Sony*).

Ao mesmo tempo, a mais recente geração de dispositivos baseada no sistema operativo *Microsoft PocketPC* (terceira geração do *Windows CE*) começou também a ganhar alguma fatia do mercado.

Figura 3 -Microsoft Pocket PC

Provavelmente, ainda mais importante que a presença de novos dispositivos é o aumento crescente da qualidade das comunicações de dados sem fios através das redes móveis usadas para a transmissão de voz. A indústria das telecomunicações como um todo foi mudando de uma rede desenhada para voz para uma rede de transporte de dados estando as próprias redes de comunicação sem fios mudando elas próprias para comunicação de dados.

Consequência destas mudanças é a possível banda larga entre 64Kbps e 1.5Mbps no acesso à Internet através de redes sem fios, quando estiverem disponíveis a geração 2,5 e a terceira geração de telemóveis.

Outra razão na base do entusiasmo em torno da Internet Móvel é o sucesso de algumas antigas aplicações. O serviço de mensagens através de redes sem fios teve um enorme sucesso: na Europa, o tráfego de *SMS (Short Message Service)* só no Reino Unido e Alemanha atingiu os 4.5 biliões de mensagens por mês em Dezembro de 2001, no Japão do serviço *NTT DoCoMo's i-mode*, que fornece acesso ao conteúdo *WEB* através de um telefone móvel ligado a uma rede de troca de pacotes de dados: em apenas dois anos atraiu mais de 20 milhões de subscritores.

A. Aplicações possíveis para WAP

- Leitura, escrita e envio de *E-mail* (a partir de qualquer ponto do planeta).
- Mercado de Acções (sempre informado sobre as recentes actividade do mercado de Acções).
- Pagamento de contas sobre *WAP*.
- *Mobile-Banking* (ver o status bancário, fazer transferências, etc.).
- Verificar o estado do tempo, tráfego, calendário, agenda.
- Customer Service (um técnico pode rapidamente obter informação que necessita para reparar um problema, actualizar software através do telemóvel)
- Verificar o Stock (um empregado pode imediatamente saber se tem de encomendar o produto em stock).
- Mobile Organizer (organizer com ferramentas recentes de comunicação, como o E-mail).
- Navegação na Internet.

B. Exemplos Actuais de WAP na Área de Saúde

Relativamente à área da saúde actualmente encontram-se implementados serviços *WAP*, tais como:

- Notícias sobre Saúde.
- Conselhos de Saúde.
- Hospitais.
- Médicos.
- Bancos de Sangue.
- Dadores de Sangue.
- Ambulâncias
- Calendário de saúde.
- Clínicas Médicas

C. Limitações

Figura 4 – Constante miniaturização dos dispositivos móveis dificulta a prestação de serviços.

Enquanto os dispositivos móveis tiverem que ser minúsculos, práticos e fáceis de carregar dentro do bolso, o ecrã *WAP* tem que ser pequeno. O utilizador não pode ler a mesma quantidade de informação que vê no seu computador e a qualidade do ecrã não é tão boa. O utilizador pode enviar imagens através de *WAP*, mas se forem demasiado complicadas aparecerão meramente como borrões. Assim, a Internet Móvel apenas terá

sucesso se conseguir encontrar a sua própria aplicação indispensável.

III. ARQUITECTURA

A. Escolha do Modelo

A primeira etapa no desenvolvimento do projecto foi dedicada ao estudo de qual a abordagem a implementar na geração de conteúdos para vários dispositivos/clientes com a mesma aplicação a partir de um modelo baseado nas potencialidades dos ficheiros *XML* e *XSL*.

Foram consideradas as seguintes abordagens:

A. Pipeline Simples
B. Múltiplos Pipelines
C. Combinação de pipelines

B. Pipeline Simples

Na abordagem denominada de Pipeline Simples as paginas ASP geram os ficheiros para cada tipo de cliente, aplicando as transformações *XSL* à informação *XML* recebida. Cada tipo de cliente requer uma stylesheet diferente. Assim, o custo no desenvolvimento está associado com a criação e manutenção destas.

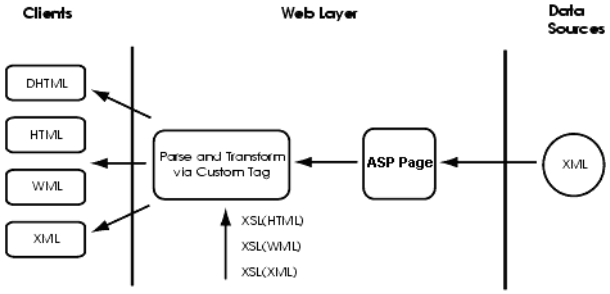

Figura 5 – Abordagem Pipeline Simples para geração de múltiplas linguagens de markup.

C. Múltiplos Pipelines

A abordagem denominada Múltiplos Pipelines usa um conjunto de páginas *ASP* específicas para cada cliente na geração dos resultados.

Quando comparado com a abordagem anterior baseada nas transformações *XSL*, este método mantém o trabalho da criação de conteúdo estático na fase desenvolvimento e a geração de conteúdo dinâmico é efectuada em tempo real.

Paralelamente à criação de páginas ASP específicas para os clientes, os custos no desenvolvimento também surgem aquando da criação e manutenção de objectos do lado do cliente, representando as abstracções de informação da aplicação, o que não é necessário na primeira abordagem. No entanto, o método de Múltiplos Pipelines pode ser mais eficiente em termos de custo que a Pipeline Simples

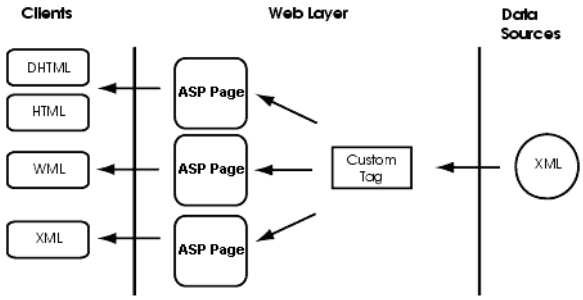

Figura 6 – Abordagem Múltiplos Pipelines para geração de múltiplas linguagens de markup.

pelas seguintes razões:

- As abstracções de dados podem ser reutilizadas pelas diferentes tipos de páginas ASP.
- As abstracções de dados são alteradas com uma frequência muito mais baixa que a formatação.
- Executar uma página ASP para a geração da linguagem é muito mais eficiente que efectuar uma transformação *XSL*T para gerar o mesmo conteúdo.

Método	Fase de desenvolvimento	Tempo Real
Pipeline simples	*Stylesheet* especifica para cada cliente	Validação de informação *XML* Validação das *stylesheet* Execução das transformações
Múltiplos *Pipelines*	Abstracção de dados Pagina ASP específica para cada cliente	Validação de informação *XML* Inicialização dos componentes de abstracção de dados Execução da página ASP

Tabela 1 – Os diferentes custos das várias abordagens

D. Combinação

É possível combinar as duas abordagens anteriores. Se os clientes utilizarem diferentes linguagens será mais coerente usar um pipeline para cada linguagem. Para gerar dialectos numa linguagem, poderá ser usada um transformação *XSL* no pipeline de cada linguagem.

As tecnologias *Active Server Pages* e *XML* são parceiros naturais para o desenvolvimento de aplicações *WEB* que usam fontes de informação heterogéneas e suportam clientes em múltiplas linguagens. Várias abordagens são

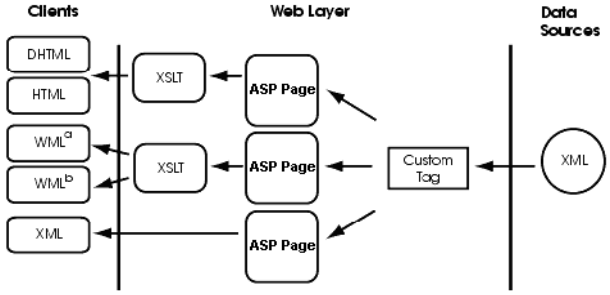

Figura 7 – Combinação de Pipelines para geração de múltiplas linguagens de markup

possíveis para cumprir estes requisitos, que devem ter em consideração o desenvolvimento e manutenção em função dos custos de processamento.

E. Modelo implementado

Desta forma, foi adoptado para o desenvolvimento deste projecto uma abordagem que reúne alguns dos aspectos anteriores e outros que melhor se adaptam ao objectivo pretendido. Assim, tal como é possível observar na Figura 8, independentemente do cliente, o pedido de informação à base de dados é sempre executado de igual forma para cada tipo de pedido. Entende-se como tipo de pedido, os diferentes serviços que podem ser disponibilizados. Posteriormente, mediante determinação do tipo de cliente, os dados são formatados com uma *XSL* diferente construindo assim um documento válido para esse cliente, ou seja, com uma linguagem de *markup* que ele consegue interpretar.

IV Os Conceitos Básicos sobre XML

O "ponto forte" da *XML* é que possibilita não só descrever o conteúdo, como também fixar a estrutura lógica desse mesmo conteúdo. A flexibilidade da *XML* provém da faculdade de transportar variadíssimos tipos de dados e de mantê-los estruturalmente coesos. A *XML* serve magnificamente para a estruturação de qualquer tipo de dados e para descrevê-los sem dúvidas ou ambiguidades em formato de texto. Isto porque a *XML* permite definir a linguagem de formatação mais adequada a um dado tipo de documentos, sendo portanto, a linguagem ideal para partilha de dados entre sistemas heterogéneos.

A Dados estruturados

Conteúdos estruturados são, sem dúvida, os conteúdos típicos dos ficheiros *XML*. São ficheiros de texto, mas os dados contidos assemelham-se muito mais às estruturas de uma base de dados do que a textos comuns, que normalmente não necessitam de um grau elevado de estrutura.

B. XML versus. HTML

Muitos documentos *XML* são criados para transportar dados exportados de bases de dados. A *eXtended Markup Language* foi, antes de mais, criada para descrever, armazenar e transportar informação estruturada. Este *Standard* internacional foi definido para estruturar dados em documentos de texto – um dos formatos mais simples e primitivos, mas compreensíveis para todos os tipos de computadores e aplicações.

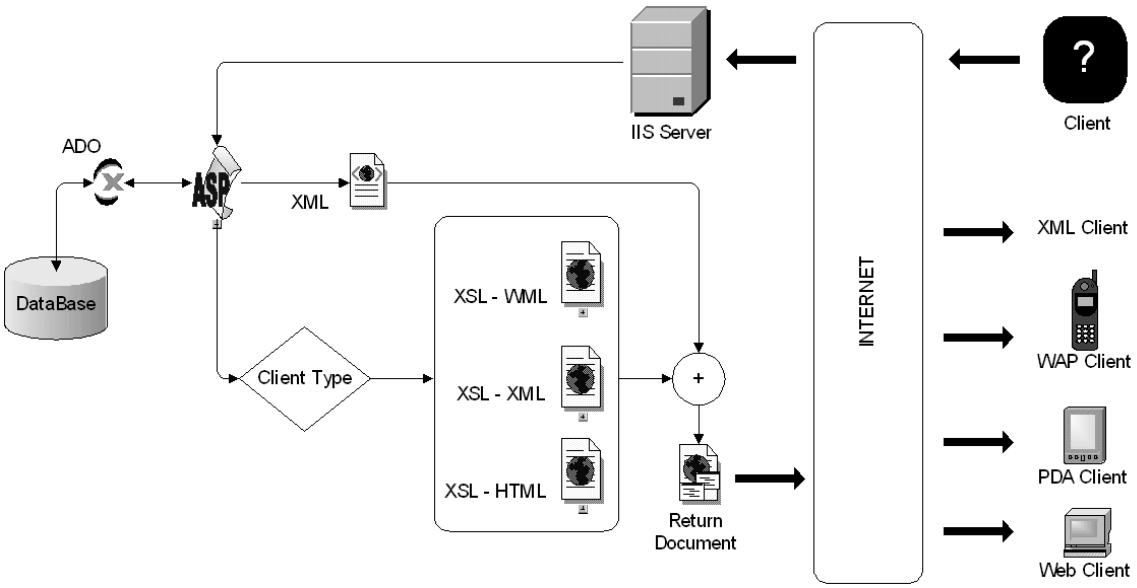

Figura 8 – Esquema geral do modelo implementado projecto

Contudo, a *XML* não serve para definir a maneira como devem ser apresentados esses conteúdos (como devem ser visualizados num browser, por exemplo); as funções de apresentação foram delegadas para a *XSL*, uma das "ferramentas" da *XML*

Entre as grandes diferenças que distinguem a *XML* da *HTML*, uma delas é capacidade da *XML* para separar o conteúdo da sua apresentação. Assim, ao formular um documento *XML*, a sua representação num browser não é preponderante, pelo que os esforços devem ser concentrados nos dados a incluir, e sobre os <marcadores> que devam estruturar esses dados, para que essas "etiquetas" evidenciem bem o significado dos mesmos.

Os dados *XML* são hierarquizados pela inserção de <marcadores> a vários níveis. Constituem uma árvore.

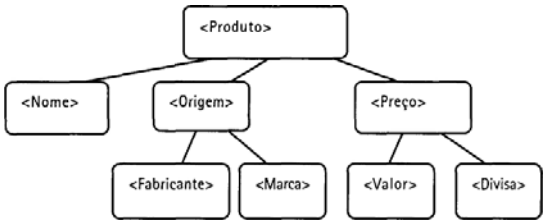

Figura 9 - Exemplo de hierarquia de um documento XML

A descrição de dados com a *XML* é muito mais legível por um computador do que as notações possíveis com a *HTML*. A informação em formato *XML* pode ser processada de forma automatizada por um parser como o *MSXML*, permitindo a troca de documentos comerciais entre computadores, por exemplo.

V. FORMATAÇÃO DE DOCUMENTOS XML

Para visualizar e formatar documentos *XML* temos a *XSL* que evita muitas vezes o recurso a uma linguagem de programação (como o Java) para efectuar transformações, selecções e formatações de documentos *XML*. Isso devido ao facto da *XSL* já integrar um extenso reportório de funções, que são bem mais típicas de uma linguagem de programação que de uma linguagem de apresentação.

A. Cross Media Publishing

Com a ajuda da *XSL*, é possível aplicar a um documento *XML* uma condizente folha de estilo e obter assim uma publicação adaptada a um meio específico.

É importante salientar, que sempre que se separa o conteúdo da sua apresentação, atingimos uma grande flexibilidade. Com diferentes formatações facilmente obtemos diferentes visualizações gráficas de um mesmo documento.

"Separar conteúdos da sua apresentação" é o princípio chave que governa a filosofia do *Cross Media Publishing*, da *WEB Content Management* (e da Gestão de Conteúdos em geral).

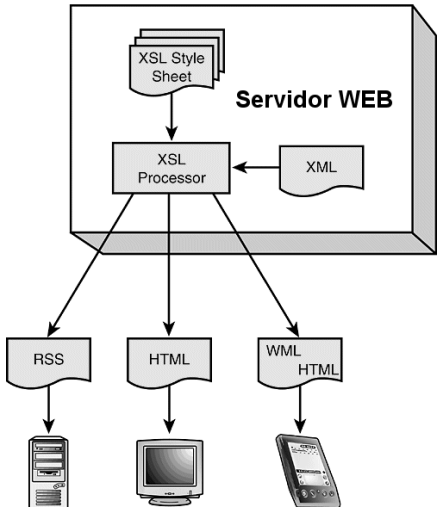

Figura 10 – Extensible Stylesheet Language (XSL), a linguagem de estilo que o W3C introduziu para visualização de documentos XML

Para visualizar um documento *XML* em diversos meios, com diversas resoluções, *WEB-browsers*, *PDAs* ou dispositivos *WAP*, serão distintos *Style Sheets* de tipo *XSL* que se encarregarão de fazer a transformação do documento *XML* para *HTML*, *WML*, *PDF* - ou para outros formatos incluído *XML* de novo.

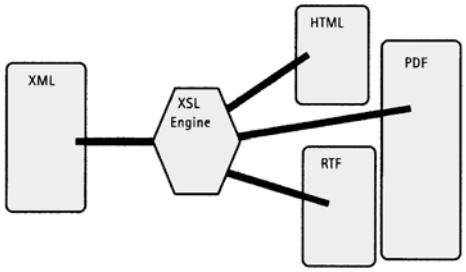

Figura 11 – Com a ajuda da XSL, é possível aplicar a um documento XML uma condizente folha de estilo e obter assim uma publicação adaptada a um meio específico.

B. XSLT Aplicada "Do lado do Servidor"

Ao efectuarmos a transformação num browser, recorremos necessariamente à aptidão desse browser para desempenhar JavaScript ou VBScript, um factor pouco previsível, visto alguns utilizadores desactivarem o desempenho desses módulos e à pouca capacidade de processamento de alguns dispositivos móveis como o caso de *WAP*. Além desse aspecto, nem todos os browsers dão suporte à *XML* e à *XSL*; e por essas três razões elaborou-se uma solução que não efectue a transformação *XSL* na componente-cliente (no browser) mas que a efectue directamente no servidor.

Temos assim como terceira alternativa uma "*Cross-Browser-Solution*"; pois será o servidor que se encarregará da transformação *XSL* e de mandar *HTML*, *WML* "limpo" ao browser, *PDA* ou dispositivo *WAP*.

O âmbito de trabalho deste projecto para realizar a alternativa "*XSL*T on the Server" pressupõe:

- A utilização do HTTP-Server Internet Information Server da Microsoft e
- A programação em código *VBScript* de uma aplicação muito simples do tipo *Active Server Pages (ASP).*

Nesta variante "*on the server*", o ficheiro *XML* não faz referência ao ficheiro *XSL*, e que o ficheiro *XSL* também não faz referência ao ficheiro *XML*.

C. O código ASP no servidor.

Este é o código usado para executar a transformação XSLT directamente no HTTP-Server, escrito em VBScript:

```
<°/o
'Carregar o XML
    set XML =
Server.CreateObject("Microsoft.XMLDOM")
    XML.async = false
    XML.load(Server.MapPath("catalogo.XML"))
'Carregar o XSL
    set XSL =
Server.CreateObject("Microsoft.XMLDOM")
    XSL.async = false
    XSL.load(Server.MapPath("catalogo.XSL"))
'Transformar o ficheiro
    Response. Write(XML.transformNode(XSL))
°/o>
```

O primeiro bloco do código *ASP* cria uma instância do *MS XML Parser (XML-DOM)*, e carrega o ficheiro catalogo.*XML* em memória.

O segundo bloco põe em activo outra instância do *Parser* e carrega o ficheiro catalogo.*XSL* em memória.

A última linha de código transforma o documento *XML* utilizando o ficheiro *XSL*, e envia o resultado ao browser do cliente em formato *HTML*, *WML*, ou *HTML* mais leve ,no caso do *PDA*.

VI MSXML E DOM

O *Document Object Model* (*DOM*) fornece um modelo de programação *standart* para trabalhar com ficheiros *XML*. O *DOM* foi desenhado para:

- Fornecer uma forma *standard* para através de programação construir, navegar ou actualizar e transformar o conteúdo de ficheiros *XML*.
- Estabelecer um conjunto básico de interfaces de aplicações programáveis de linguagem neutral que possam ser usadas para preencher as necessidades mais comuns dos programadores que trabalham e geram correctamente ficheiros de entrada e saída de dados nas linguagens *XML* e *HTML*.

O *Document Object Model* (*DOM*) apresenta um processo de interpretação fácil e *Standardizado* de um documento de *XML* para aplicações e *scripts*. A implementação do *DOM* nos serviços base de *XML* da Microsoft® (*MSXML*) permite que se carregue ou crie um documento, identificar erros, caso existam, aceder e manipular a informação e estruturas contidas no documento e gravar novamente a informação num ficheiro de *XML*, caso se torne necessário.

A. DOM e MSXML

A aproximação do Dom é criar um objecto tipo árvore que é gerido pelo interpretador do *MSXML*. Isto permite que ao criar os conteúdos *XML* se possa tirar vantagem da forma lógica como o *MSXML* executa essa criação, em vez de se criar esses conteúdos de uma forma não ordenada.

B. Construção do Documento em Nós

O *DOM* fornece uma interface para carregar, aceder, manipular e armazenar documentos *XML*. O *DOM* fornece uma representação completa de um documento *XML* guardado em memória, facultando acesso aleatório aos conteúdos de todo o documento. O *DOM* permite que as aplicações se baseiem na lógica fornecida pelo parser do *MSXML* para manusear informação assente em *XML*, usando as suas facilidades em vez de usarem código próprio para ler e processar *XML*.

C. Como funciona o DOM

O *MSXML* lê um documento *XML* e interpreta o seu conteúdo para um conjunto de recipientes de informação abstractos chamados nós. Estes nós representam a estrutura e o conteúdo desse documento, permitindo que as aplicações possam ler e manipular a informação desse documento sem necessariamente perceberem explicitamente a sintaxe *XML*. Depois de o documento ser interpretado, os seus nós podem ser explorados em qualquer direcção, não estando limitados ao convencional num ficheiro de texto, que é apenas um sentido único.

D. Persistência e o DOM

A informação guardada num *XML* Document Object Model (*DOM*) pode ser armazenada num ficheiro *XML*, e a informação armazenada num ficheiro *XML* pode ser aberta num *DOM*. Os ficheiros *XML* conferem uma forma prática de guardar a informação que pode ser processada, transportada em ambientes diversos.

VII FILESYSTEM OBJECT

Quando se usam tecnologias de *Scripting* em *ASP's* (*Active Server Pages*), ou em qualquer outra aplicação, é por vezes importante adicionar, mover, mudar, criar, ou apagar ficheiros e directórios no servidor *WEB*. Pode ser também necessário obter informação ou manipular *drives* ligadas ao esse mesmo servidor.

O *Scripting* permite o processamento de *drives*, directórios e ficheiros através do objecto FileSystemObject (*FSO*), que é explicado em seguida.

A. O modelo do objecto FileSystemObject

No modelo do objecto *FSO* é possível usar a sintaxe familiar objecto.metodo com um vasto conjunto de propriedades, métodos e eventos para processar ficheiros e directórios. É possível usar esta ferramenta orientada ao objecto conjuntamente com:

- *HTML* para criar páginas *WEB*.
- Windows Scripting Host para criar ficheiros batch para o Microsoft Windows.
- Controlo de Script para fornecer um capacidade de Scripting às aplicações desenvolvidas noutras linguagens.

Sendo que o uso do *FSO* do lado cliente levanta graves problemas de segurança, em termos do fornecimento de acesso ao sistema de ficheiros do cliente, assume-se que o objecto *FSO* é usado na criação de *scripts* que serão executados pelas páginas *WEB* do lado do servidor. Desta forma, os parâmetros de segurança por omissão dos diferentes browsers não permitem a utilização do objecto *FSO* do lado do cliente. Uma alteração desses parâmetros pode fazer com que um computador local fique vulnerável e seja possível alterar ou danificar informação residente nesse computador.

O objecto *FSO* fornece às aplicações que são executadas no servidor a capacidade de criar, alterar, mover e apagar directórios, ou detectar a existência de um directório em particular e em caso afirmativo, onde. É possível também com este objecto saber informações acerca dos directórios, tais como os seus nomes, a data em que foram criados ou modificados pela última vez.

B. Programação do FileSystemObject

Para programar com o objecto FileSystemObject:

- Usar o método CreateObject para criar um objecto FileSystemObject
- Usar o método apropriado no recém-criado objecto
- Aceder às propriedades do objecto

Trabalhar com ficheiros

Existem duas grandes categorias na manipulação de ficheiros:

- Criação e Leitura de ficheiros, Inserção e Remoção de dados nos ficheiros
- Mover, Copiar e Apagar ficheiros

VIII. INTRODUÇÃO A WEB CLIPPING E PQAS

A alguns anos atrás, navegar na WEB através de um dispositivo *handheld* era praticamente uma miragem. No entanto hoje em dia, isso deixou de ser um sonho para se transformar em realidade, devido aos esforços dos fabricantes de dispositivos.

Um dos mais populares dispositivos handheld no mercado é o *PDA Palm Organizer*, e o método que os seus programadores implementaram para resolver o problema da pouca largura de banda foi o chamado *WEB Clipping*.

O objectivo do *WEB Clipping* é minimizar tanto os requisitos de visualização quanto a largura de banda.

A. Modelo Usado Pelo WEB Clipping

WEB Clipping usa um modelo de "pergunta e resposta" em detrimento do sistema de *hyperlinks Standard* da *WEB*. Neste sistema de partilha de aplicação a parte do pedido da aplicação é armazenada loacalmente no utilizador, na chamada *Palm Query Application*, ou *PQA* (ou também Aplicação de *WEB Clipping* WCA).

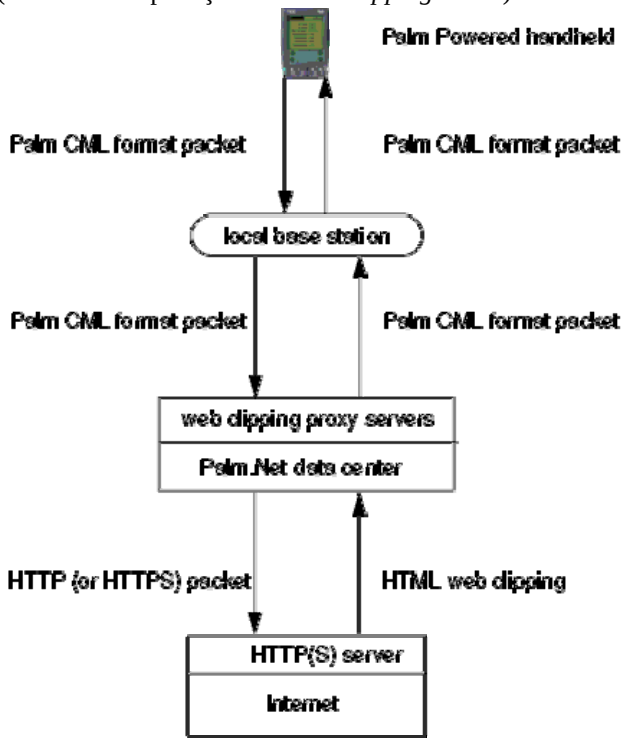

Figura 12 – Visão global de uma operação de WEB Clipping

O ficheiro PQA é basicamente um mini-*WEB* site, pré-instalado no *Palm* VII do utilizador.

É usualmente um pequeno formulário que transmite uma quantidade ainda mais pequena de dados para o servidor remoto.

Tipicamente, o que o *WEB Clipping* retorna do servidor são pequenas paginas *WEB* geradas dinamicamente através de um *script CGI* em resposta a um pedido por parte do utilizador. Também podem ser páginas estáticas num servidor *WEB*. O tamanho é essencial. A *Palm* recomenda que os pedidos efectuados ao servidor não ultrapassem alguns bytes, e que o retorno através da *WEB Clipping* seja mantido abaixo dos Kilobytes estando estes já comprimidos.

IX. IMPLEMENTAÇÃO

A. Utilização do objecto FileSystemObject na aplicação

O recurso à utilização deste objecto no modelo de projecto implementado prendeu-se essencialmente com a manutenção necessária a efectuar aos ficheiros criados pelo objecto *DOM*.

O objecto *DOM* é responsável por gerar um ficheiro *XML* por cada serviço e utilizador com a informação necessária proveniente da base de dados. Esta situação é necessária para separar a informação de cada utilizador, já que se dois ou mais utilizadores estivessem a consultar o mesmo serviço simultaneamente, não era possível garantir que cada utilizador iria receber o conteúdo sobre o qual tinha efectuado o pedido, já que iria sempre ser gerada a informação no mesmo ficheiro. Desta forma é possível garantir que cada utilizador gera um ficheiro com a informação que foi pedida e o resultado desse pedido nunca irá colidir com o pedido de outro utilizador.

Se o utilizador da aplicação, quando pretender terminar a sua aplicação, usar o serviço de *"logout"*, os ficheiros que ele próprio criou serão apagados e é actualizada a informação na base de dados respeitante à possível existência de ficheiros pertencentes a esse utilizador. Este processo resulta em que apenas durante a utilização da aplicação existem ficheiros de um determinado utilizador.

O reverso da situação anterior é o utilizador deixar a aplicação, quer por motivos a ele alheios, nomeadamente falhas de comunicação, ou deliberadamente sem usar o serviço de *"logout"*. Neste caso, os ficheiros que a ele pertencem permanecem no disco e só seriam apagados caso o utilizador em causa voltasse a usar a aplicação e procedesse de forma correcta (situação anterior).

Para prevenir estas possíveis falhas, foi implementado um sistema que permite saber, tal como foi referido anteriormente, se existem ficheiros pertencentes a um dado utilizador. À medida que cada novo utilizador acede à aplicação, verifica quais são os utilizadores que possuem ficheiros e que não têm qualquer actividade na aplicação por um tempo superior a 20 minutos. Os ficheiros que forem encontrados nesta dupla condição são apagados do disco do servidor *WEB*, resultando assim que os ficheiros existentes no disco são apenas os ficheiros em utilização.

O método do objecto *FSO* que essencialmente foi usado em todo este processo foi o DeleteFile.

B. Estudo da usabilidade.

Usabilidade é primordial em certos dispositivos móveis por causa do factor de miniaturização actual, da capacidade limitada de visualização de imagens e do ecrã pequeno, a usabilidade torna-se a regra mais importante quando chega a fase desenvolvimento de uma interface para a aplicação, assim deve-se ter em conta:

- Interface o mais simples possível.
- Cada toque numa tecla é uma ameaça à usabilidade.
- Não usar menus de introdução (Splash Screen)
- Excepto...para dar feedback relativamente à aplicação.
- A maioria dos dispositivos móveis, principalmente telemóveis, apresenta apenas 4 linhas de texto de cada vez.
- Minimizar ou evitar a entrada de texto.
- Evitar erros desnecessários do utilizador.
- Fazer uso da interacção oferecida pelo dispositivo.
- Limitar o tempo de download de um deck (cada página) em dispositivos móveis tais como o *WAP* e o *PDA*.
- Implementar ferramentas para elevar a usabilidade em qualquer tipo de plataforma.

Para aumentar a usabilidade em qualquer plataforma foram ainda implementadas as seguintes ferramentas:

- Paginação de conteúdos
- Ordenação de conteúdos
- Introdução assistida de dados
- Meios de atalho para serviços
- Feedback de acções
- Menus de Ajuda

C. Ficheiro de configuração e acesso à base de dados

Quando se pensa numa aplicação multi-plataforma deve ser pensado que provavelmente a aplicação poderá também ser portátil. Isto significa que deverá ser encontrada uma forma fácil de configurar os parâmetros necessários para que se possa pôr a aplicação a funcionar. Nesse sentido neste modelo optou-se por ter um ficheiro onde se inclui todos os parâmetros que permitam o seu correcto funcionamento, assim como fazer uma ligação ao objecto do projecto, a base de dados. Através da funcionalidade inclusão das *ASP* é

possível incluir este ficheiro em todos os outros ficheiros da aplicação fazendo com que todos disponham destes parâmetros.

No início deste ficheiro é invocada uma linha de código que permite que o servidor não devolva erros de linguagem ou outros ao cliente, isto é, mesmo que algo esteja errado do lado do servidor o cliente não deverá saber que isso aconteceu. Isto é uma das regras básicas da usabilidade que indica que o utilizador não deve ter contacto com a tecnologia.

Em seguida são definidos parâmetros de cache. Na aplicação deste projecto todos os dados são voláteis, ou seja, cada pesquisa, cada interacção que o utilizador tenha com a ferramenta são pedidos novos dados, ou são invocadas novas condições. Desta forma é de evitar que o servidor *IIS* faça cache das páginas *ASP* que são pedidas, já que o conteúdo das mesmas podem variar devido a constante actualização da base de dados. Outro factor a ter em conta é que o cliente deverá ter contacto com todos os dados ao mesmo tempo de maneira a que este tenha a sensação que todos os conteúdos que foram pedidos foram devolvidos, nesse sentido é dado ordem ao servidor que só deve devolver a informação ao cliente depois de toda a página *WEB* estar construída do lado do servidor. O código que implementa esta funcionalidade e que indica ao servidor *WEB* que só deverá enviar os dados ao cliente no final da construção e que não deve guardar a página em cache é o seguinte:

```
Response.Buffer = True
Response.Expires = -1
Response.CacheControl = "Private"
```

Sendo esta uma aplicação multi-plataforma é neste ficheiro que deve ser decido que tipo de conteúdo deve ser devolvido ao cliente. O servidor ao receber o pedido do cliente é capaz de identificar um conjunto de variáveis para além do próprio pedido. Uma das variáveis que é possível identificar é qual o agente de protocolo que o cliente está a usar. A variável em causa é denominada “HTTP_USER_AGENT” e mediante o seu conteúdo vai ser possível identificar qual a linguagem que melhor se adequa ao cliente. Como exemplo de conteúdo dessa variável pode-se enumerar:

- NS4.7: Mozilla/4.73 [en] (Windows NT 5.0; U)
- IE5: Mozilla/4.0 (compatible; MSIE 5.*)
- PocketPC: Mozilla/2.0 (compatible; MSIE 3.02; Windows CE; 240X320)
- WAP Phone: UP.Browser/3.0-UPG1 UP.Link/3.2
- PalmVII: Mozilla/2.0 (compatible; Elaine/1.0)

No sentido de determinar qual foi o cliente, é feita uma análise a esta variável, resultando num conjunto de variáveis que caso encontrem a string de comparação, retornam um valor superior a 0. Esta determinação é executada com o seguinte excerto de código:

```
client_4 = instr(1,
Request.ServerVariables("HTTP_USER_AGENT"),
"Mozilla/4", 1)
client_2 = instr(1,
Request.ServerVariables("HTTP_USER_AGENT"),
"Mozilla/2", 1)
client_L = instr(1,
Request.ServerVariables("HTTP_USER_AGENT"),
"Lynx", 1)
```

Estas variáveis em concreto permitem saber se é um cliente *WEB* (client_4>0), se é um cliente *PDA* (client_2>0) ou se é um cliente tipo texto (client_L>0). Caso estas três variáveis sejam todas iguais a zero então o cliente será do tipo *WAP*. Esta contextualização é feita noutro ficheiro que é incluído em todos os outros (init.asp) onde são feitas todas as inicializações. Essa contextualização é feita da seguinte forma:

```
cweb = FALSE
cpda = FALSE
cwap = FALSE
If client_4 > 0 Then cweb = TRUE End If
If client_2 > 0 Then cpda = TRUE End If
cwap = NOT (cweb OR cpda)
```

Estas novas variáveis são usadas em diversas situações nomeadamente depois da formação do *XML* para determinar qual a plataforma para a qual a aplicação deve ser redireccionada.

Outro dos parâmetros do ficheiro de configuração é a definição do nome da aplicação e qual o endereço do servidor onde esta está instalada. Estes parâmetros tornam-se necessários já que devido essencialmente aos clientes *WAP* se optou por fazer todos links existentes na aplicação como sendo *URL*, ou seja, todos os links devem ter o endereço completo. Os clientes em causa não conseguiam seguir o link se este fosse dado como sendo link local (relativo), pois os browsers implementados nestes dispositivos não reconheciam esse tipo de links. Estas configurações são traduzidas no excerto de código seguinte:

```
site_name = "IWBDC - Interface WAP para
Base de Dados Clínica"
base_addr =
"http://iwbdc.det.ua.pt/projecto/"
```

Um dos pontos-chave deste ficheiro de configuração é a ligação à base dados. A ligação à base de dados poderia ser feita estabelecendo e configurando uma ligação *ODBC* ao servidor *SQL* e à base de dados em causa. Isso iria requerer que para além de configurar este ficheiro fosse também necessário configurar essa ligação e isso é algo que se poderá tornar complicado. Além disso, para efectuar essa operação era necessário ter permissões de administrador de sistema o que poderá não acontecer. Em termos de funcionalidade e de desempenho uma ligação deste tipo não trás nem vantagens nem prejuízos.

No modelo idealizado a ligação à base de dados é feita por componentes *ADO* (*Access Data Object*) do *ActiveX*. Antes de se ler ou escrever informação na base

de dados é necessário estabelecer essa ligação, sendo que o *ADO* permite que a ligação seja feita através de linguagem de scripting. O *ADO* situa-se entre a aplicação e a camada *OLE-DB*. A ligação é um elo entre os objectos *ADO* e a base de dados e essa ligação permanece activa enquanto as partes necessitarem de comunicar.

A string de ligação (connection string) é um conjunto de dados que estabelece como deve ser feita a ligação. Uma *string* de ligação contêm tipicamente estes parâmetros:

- **Provider** – fabricante do tipo de OLE-DB usado na ligação
- **Driver** – Tipo especifico de driver ODBC para MS Access/SQL Server
- **Initial File Name or Data Source** – Nome do ficheiro
- **Initial Catalog** – Nome da base de dados
- **UserID** – o nome de utilizador necessário para ligar à base de dados
- **Password** – palavra chave do utilizador em causa

Escrever a string de ligação é apenas uma questão de juntar essas variáveis e parâmetros separados por ponto e vírgula. Exemplo da ligação efectuada na aplicação deste projecto é o excerto de código seguinte:

```
db_server = "IWBDC"
db_name = "site_iwbdc"
db_user = "xxxxxxx"
db_pass = "xxxx"
strConnect = "Driver={SQL Server};
Server="&db_server&"; Database="&db_name&";
UID="&db_user&"; PWD="&db_pass&";"
```

Em todas as ligações que são efectuadas à base de dados é invocada esta string de ligação, possibilitando assim que com apenas uma definição se possam efectuar todas as ligações.

D. Modelo de segurança

Numa aplicação multi-plataforma é fundamental garantir um grau de segurança. A segurança básica implementada refere-se ao pedido de Utilizador e *Password* ao iniciar cada sessão. Apenas utilizadores válidos podem aceder à aplicação. São considerados utilizadores válidos, todos os utilizadores que não tenham sido removidos e os utilizadores em que o binómio utilizador/*password* é considerado válido.

É necessário garantir que um utilizador válido após ter efectuado o processo de validação se mantém válido enquanto durar a sua utilização da aplicação. Não faria sentido que a cada pedido que fosse executado, fosse novamente verificada a *password* do utilizador, para validar o pedido. Nesse sentido as *ASP* permitem gerar variáveis de sessão que se mantém válidas enquanto o browser do utilizador estiver aberto e/ou o utilizador não fizer "*logout*". Acontece que estas variáveis têm um funcionamento semelhante aos *cookies*, que criam pequenos ficheiros no lado cliente que permitem ao servidor verificar determinadas condições. As variáveis de sessão são todas guardadas num ficheiro que tem um identificador único e não é detectável no sistema de ficheiros do sistema do utilizador já que é guardado em memória.

Este método tem alguns inconvenientes quando nos referimos à aplicação deste projecto. A utilização das variáveis de sessão acaba por ser falível se o browser do cliente estiver configurado para não aceitar *cookies*. Nessa situação as variáveis de sessão, não serão válidas e não garantem a estabilidade da aplicação. Por outro lado os dispositivos *WAP* não guardam as variáveis de sessão o que impede que o modelo de segurança assente nestas variáveis.

Foi então necessário criar um modelo alternativo. A primeira fase deste modelo é na validação de utilizadores. Após ser verificada a autenticidade do utilizador através do binómio utilizador/*password* são registados na tabela de utilizadores o endereço IP a partir do qual o utilizador fez a validação assim como a data e hora a que foi feita essa validação. Após este procedimento o redireccionamento é feito para a página principal da aplicação, independentemente da plataforma sob a qual o utilizador está a aceder, com o identificador único desse utilizador atribuído à variável de utilizador. Caso não seja verificado o binómio em causa então o redireccionamento é feito para a página de entrada e são novamente pedidos os dados do utilizador. O ficheiro responsável por esta verificação é o ficheiro "redirect.asp".

A outra fase do modelo é a verificação, pedido a pedido, das condições necessárias de segurança estarem garantidas. A primeira condição deste modelo é que cada vez que seja efectuado um pedido, este tem de ter a variável com o identificador único do utilizador atribuída. Caso esta variável não se encontre atribuída a aplicação cessa imediatamente e retorna ao ecrã de entrada com o valor da variável do utilizador atribuída negativamente, garantindo assim que nenhum utilizador possui esse identificador.

A segunda condição é que o utilizador esteja a fazer o novo pedido a partir do mesmo endereço IP no qual efectuou a validação inicial. Tal como no processo de validação é possível através da variável de servidor "*REMOTE_ADDR*" saber qual o endereço IP do utilizador. Caso não se verifique esta igualdade a aplicação cessa também nesse momento e é redireccionada para a página de entrada com o valor da variável de utilizador atribuída negativamente.

A terceira condição é o tempo de inactividade do utilizador. À semelhança do que se passa com as variáveis de sessão, que têm um tempo de validade de 20 minutos, este modelo também verifica à quanto tempo o utilizador efectuou último acesso. Caso o utilizador tenha efectuado o último pedido à menos de 20 minutos então é registado na tabela de utilizadores a data e hora actuais facultando assim que ele possa

utilizar a aplicação nos próximos 20 minutos. Caso o utilizador tenha um tempo de inactividade superior a 20 minutos a aplicação também cessa de imediato sendo redireccionada para a página de entrada com a variável de utilizador atribuída negativamente.

Estas três condições garantem que o utilizador é sempre o mesmo e que está a aceder do mesmo ponto em que acedeu da última vez num espaço de tempo válido.

A implementação destas condições é demonstrada no seguinte excerto de código, em que "zky" é a variável de utilizador:

```
If zky>0 Then
  ip =
Request.ServerVariables("REMOTE_ADDR")
  rsSet.Open ("SELECT nome,grupo,ip,sistema
FROM IDUtilizadores WHERE
cod_utilizador="&zky),strConnect,3,3
  If NOT rsSet.eof Then
    If ip=rsSet("ip") AND
DateDiff("n",rsSet("sistema"),Now())<20
Then
    rsSet2.Open("UPDATE IDUtilizadores SET
sistema=getdate() WHERE
cod_utilizador="&zky),strConnect,3,3
```

Outra das vertentes do modelo de segurança é acesso aos diferentes serviços por parte dos diferentes utilizadores. Neste modelo optou-se por agrupar os utilizadores em grupos e as permissões para acesso aos diferentes serviços é dada aos grupos. Cada grupo pode ter diferentes permissões mediante a plataforma pela qual o acesso é executado. As permissões encontram-se registadas na tabela de permissões e funciona como sendo uma matriz em que os serviços que tem o valor a 1 nessa plataforma são os serviços sobre os quais existem permissões. O menu da aplicação é sempre construído tendo em atenção quais são as permissões do grupo do utilizador e são apenas devolvidos os serviços para os quais o utilizador tem acesso.

No caso em que existe cruzamento entre serviços, é na geração dos ficheiros *XML* que são indicados quais os serviços aos quais o utilizador tem acesso, e é depois na formatação do documento *XSL* que se verificam essas permissões e são dados os links possíveis ao utilizador. A gestão das permissões de cada grupo é efectuada na gestão de utilizadores, pelo que apenas deve ter acesso a esse serviço, existente apenas na plataforma *WEB*, utilizadores responsáveis pelo sistema.

E. Esquema de ficheiros

O esquema de ficheiros utilizados representa fielmente o modelo implementado, e o conceito da linguagem *XML* por detrás da aplicação, separando de uma maneira clara o que é conteúdo e a formatação.

Primeiro os ficheiros encontram-se separados em 4 canais de derivação de conteúdo, ou seja o canal de informação comum às 3 plataformas, a chamada directório *common* (comum), mais os restantes 3 canais relativos as plataformas específicas da aplicação *WAP*, *PDA* e *WEB.*

Iwbdc

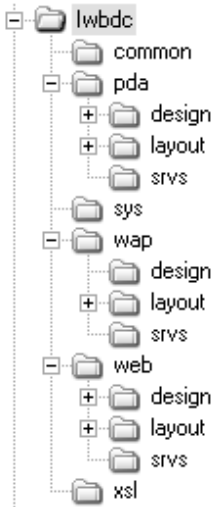

Figura 13 – Esquema geral de organização de ficheiros implementado.

No directório IWBDC existem os ficheiros *default.asp* e *main.asp* responsáveis pela, inicialização das váriaveis de sistema, pela captura do tipo de plataforma em uso, e pelo respectivo reencaminhamento para o canal de informação comum.

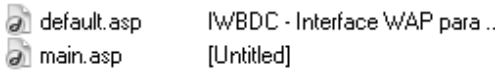

Figura 14 -Ficheiros na raiz da aplicação

***Iwbdc*\Common:**

Neste directório comum existem os ficheiros de serviço tal como estão referenciados na base de dados:

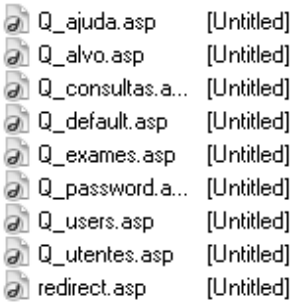

Figura 15 – Ficheiros na directoria \Iwbdc\Common

Estes ficheiros e serviços podem ser configurados, alterados a partir da base de dados.

Os ficheiros *Q_default.asp*, *Q_users.asp* neste directório gerem tudo o que é validação de utilizadores e gestão de pedidos de serviço, criação de resultados e

reencaminhamento destes para os 3 canais de informação específicos às 3 plataformas.

Quando um utilizador acede ao sistema é então redireccionado para o seu directório de acordo com o tipo de plataforma usada e serviço pedido.

\Iwbdc\wap,web ou pda:

O modelo de ficheiros para cada uma das plataformas é o mesmo. Cada directório raiz de uma plataforma contém:

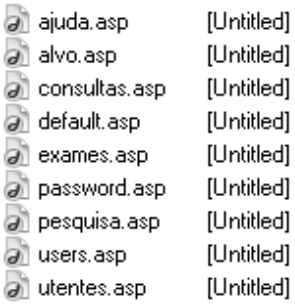

Figura 16 -Ficheiros na raiz de cada plataforma.

Estes ficheiros gerem todos os aspectos relativos aos menus dinâmicos de serviços, quer seja o menu principal gerado pelo *default.asp* para cada plataforma e de acordo com as permissões de serviços devolvidas para cada utilizador em *R_default.xml*

Figura 17 – Parte dos ficheiros no directório recipiente para ficheiros .xml

```
- <QueryResult>
  - <vars>
      <zky>134</zky>
      <dft>134</dft>
      <srvid>-1</srvid>
      <base_addr>http://192.168.0.1/projecto/</base_ac
    </vars>
  - <menus>
    - <menu>
        <id_servico>21</id_servico>
        <servico>Consultas</servico>
        <acronimo>cns</acronimo>
        <ficheiro>consultas.asp</ficheiro>
        <ficheiro_query>Q_consultas.asp</ficheiro_query>
        <id_servpai>-4</id_servpai>
      </menu>
    - <menu>
        <id_servico>22</id_servico>
        <servico>Exames</servico>
        <acronimo>exm</acronimo>
        <ficheiro>exames.asp</ficheiro>
        <ficheiro_query>Q_exames.asp</ficheiro_query>
        <id_servpai>-4</id_servpai>
      </menu>
    - <menu>
        <id_servico>23</id_servico>
```

Figura 18 – Excerto de ficheiro R_default.xml

Os ficheiros básicos de sistema são criados logo aquando o acesso a aplicação para cada utilizador identificados pelo número de utilizador como prefixo (neste caso o 134), e definem um meio de transporte :entre serviços das variáveis de sistema de cada utilizador. Definem por exemplo os serviços permitidos, para ser possível criar os menus

De acordo com este ficheiro o *default.asp* em cada directório raiz de plataforma coloca em memória todas estas variáveis (carregam os .xml em memória) e redirecciona o código para o directório de serviços respectivo, que neste caso é o wap\srv\default_principal o mesmo se passa com qualquer outro ficheiro no directório raiz de plataforma.

\Iwbdc\wap\srv

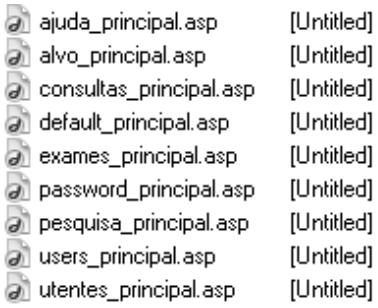

Figura 19 – Ficheiros de serviço de interacção com o utilizador

Estes ficheiros são responsáveis por toda a interacção com o utilizador quer criado, os menus, formulários de pesquisa, ou resultados de pesquisa. Tendo os ficheiros .xml em memória, estes carregam os ficheiros .xsl contidos no directório \Iwbdc\xsl e fundem a informação criando os ficheiros de saída para cada plataforma, ou seja criam os ficheiros *.WML* e *.HTML*

\Iwbdc\xsl:

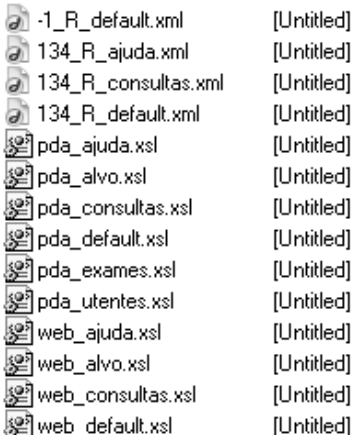

Figura 20 –Parte das XSL's contidas no directório \Iwbdc\xsl

Uma *XSL* por serviço para cada plataforma, respeitam o conceito implementado no modelo.

Os restantes directórios dizem respeitos a funções de formatação e visualização de conteúdos para cada plataforma.

F. Serviços Implementados

Tal como foi dito na introdução, este projecto visa implementar infra-estruturas cliente-servidor baseadas no protocolo *WAP (Wireless Access Protocol)* que permitam a comunicação entre terminais móveis e um sistema de informação clínica. Em complemento visa também criar as mesmas estruturas usando o protocolo *HTTP (HiperText Transfer Protocol e a plataforma PDA).* Os serviços proporcionados por tais infra-estruturas deverão orientar-se para os aspectos de visualização da informação e algumas tarefas de gestão do sistema.

Assim, neste projecto foram privilegiados os seguintes objectivos em detrimento da quantidade de serviços propostos:

- Ferramentas de Gestão de Sistema.
- Visualização clara de informação clínica nas 3 plataformas
- Diversidade de ferramentas de pesquisa.
- Elevada usabilidade da aplicação em qualquer tipo de plataforma.
- Mecanismos de segurança.
- Portabilidade de informação inter-plataforma através do *XML*.
- Estudo aprofundado quanto ao modelo baseado em *XML*.
- Ferramentas de Gestão de Sistema

Por motivos relacionados com a usabilidade da interface usada, o acesso à Gestão do Sistema só foi implementado na plataforma *WEB*, embora como todas as 3 plataforma assentem no mesmo modelo de gestão.

No modelo de Gestão adoptado existem vários utilizadores com acesso ao sistema podendo estes estarem activos ou inactivos, agrupados em grupos de utilizadores.

id_grupo	nome_grupo	blocked	alvo
100	Administradores	1	0
101	Médicos	1	1
102	Enfermeiros	1	1
103	Técnicos	1	1
104	Administrativos	1	0

Figura *21* – Tabela de Grupos de Utilizadores.

Cada grupo tem definidas permissões a nível de quais serviços tem acesso e em qual plataforma.

id_grupo	id_servico	web	wap	pda
100	21	1	1	1
101	21	1	1	1
101	22	1	1	1
101	23	1	1	1
100	23	1	1	1
100	1	1	0	0
100	22	1	1	1
100	100	1	1	1

Figura *22* – Tabela de Permissões.

Cada serviço tem definido os respectivos documentos de consulta, sendo estes comuns ás 3 plataformas.

id_servico	id_servpai	servico	acronimo	ficheiro	ficheiro_query
1	-3	Utilizadores	user	users.asp	Q_users.asp
20	-4	Home	df	<NULL>	<NULL>
21	-4	Consultas	cns	consultas.asp	Q_consultas.asp
22	-4	Exames	exm	exames.asp	Q_exames.asp
23	-4	Utentes	utn	utentes.asp	Q_utentes.asp
100	-5	Password	pwd	password.asp	Q_password.asp
24	-4	Alvo	alv	alvo.asp	Q_alvo.asp
101	-5	Ajuda	ajd	ajuda.asp	Q_ajuda.asp

Figura *23* – Tabela de serviços.

A ferramenta de Gestão só foi implementada na plataforma *WEB* devido ao facto da elevada extensão dos formulários utilizados.

As permissões são definidas quer em termos de serviços quer em termos de plataformas, tal como foi mostrado.

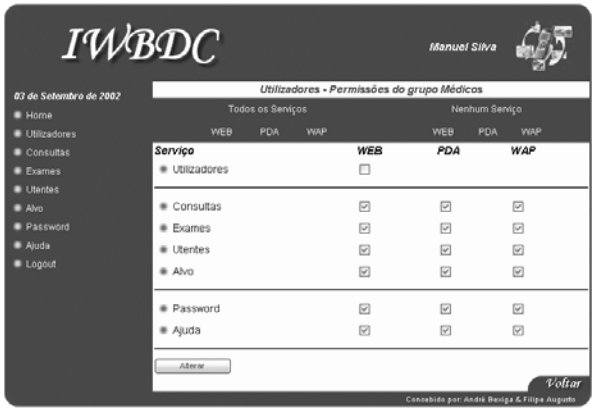

Figura *24* – O Menu de Gestão de Utilizadores

De referir ainda um aspecto bastante importante; neste modelo de gestão um utilizador não consegue apagar-se a si mesmo, e como é compreensível também não é possível apagar um grupo de utilizadores que não esteja vazio.

A partilha e transporte de dados entre serviços é feita a partir de *links* sobre dados pesquisáveis, resultando assim na triangulação de informação entre os 3 serviços

X. RESULTADOS

A. PDA

O nível de liberdade aquando da formatação de conteúdos no *PDA* é um nível intermédio. Se por um lado permite algumas facilidades gráficas, por outro é suposto ser um dispositivo móvel e como tal a transferência de dados é relativamente lenta, pelo que a quantidade de informação a enviar deve ser reduzida, adoptando-se assim uma solução de compromisso.

Figura 25 – Formatação nas diversas etapas de utilização em PDA

B. WAP:

A Usabilidade é um dos factores primordiais em *WAP*, onde se teve extremo cuidado relativamente à profundidade dos menus.

Figura 26 – Formatação nas diversas etapas de utilização em WAP

C. WEB

Neste tipo de interface o grau de liberdade é total e é aquele que permite um absoluto controlo da aplicação. Ao contrário dos dispositivos anteriores, este tipo de interface pressupõe a utilização de um computador pessoal tornando-se mais fácil a tarefa de introdução de informação, sendo como tal a interface de eleição para um uso mais prolongado da aplicação.

V. CONCLUSÕES

Tentando dar o enquadramento necessário ao projecto, começou-se por estudar a envolvente e perspectiva de mercado em torno do que são actualmente as condicionantes e objectivos desta área de actuação, tendo sempre em vista a aplicação que se pretendia desenvolver.

Verificou-se relativamente a toda a envolvente de mercado dos serviços *WAP* que a Internet Móvel pode ter sucesso apenas pelo desenvolvimento de aplicações adequadas às características únicas do ambiente móvel e suas limitações tecnológicas. Obviamente que, com o decorrer do tempo e com as melhorias esperadas a nível tecnológico tanto ao nível dos dispositivos como da velocidade das redes, as limitações do ambiente móvel irão diminuir e a separação entre Internet Móvel e convencional será cada vez menos nítida. Até lá, a Internet Móvel apenas terá sucesso se conseguir encontrar a sua própria aplicação indispensável.

Fez-se um estudo a nível do utilizador actual e futuro cliente *WAP*, tendo-se mostrado porque é que a Internet Móvel está a chamar tanta atenção no momento. Esta tem o potencial de expandir largamente o número de pessoas que acedem ao conteúdo actual da Internet; tem também o potencial de criar novas aplicações que serão únicas da Internet Móvel, sendo de acreditar que a segunda será aquela que mais consumidores irá atrair.

Foram também estudados serviços e aplicações que podem tornar a Internet Móvel em algo atractivo e concretamente os serviços que poderão despoletar a atenção dos futuros utilizadores da interface.

Tento sido verificadas as dificuldades actuais a nível tecnológico dos dispositivos *WAP*, assim como a dificuldade na prestação de serviços para esses mesmos dispositivos, avançou-se para outra tecnologia móvel, os *Personal Digital Assistant*, dando ênfase à capacidade de adaptação a multi-plataformas do binómio *XML*/*XSL*. A recente tecnologia de *WebClipping* trouxe ao projecto uma nova força permitindo a criação de uma plataforma diferente, a meio termo entre a reduzida capacidade do *WAP* e a vastíssima diversidade da *WEB*.

Como tecnologia base de todo o projecto, o *XML* foi alvo de um cuidado estudo quer em termos de linguagem quer em termos de modelo de construção. O estudo da sua envolvente e modelos de criação dinâmica, assim como ferramentas que permitem a manipulação dos ficheiros *XML* foram algo que contribuíram para uma mais valia do projecto, permitindo a correcta implementação e uso dos mesmos.

Um dos principais pontos de todo o projecto era o modelo a implementar. Este teria de ser um modelo válido para a expansão de novos serviços e de novas

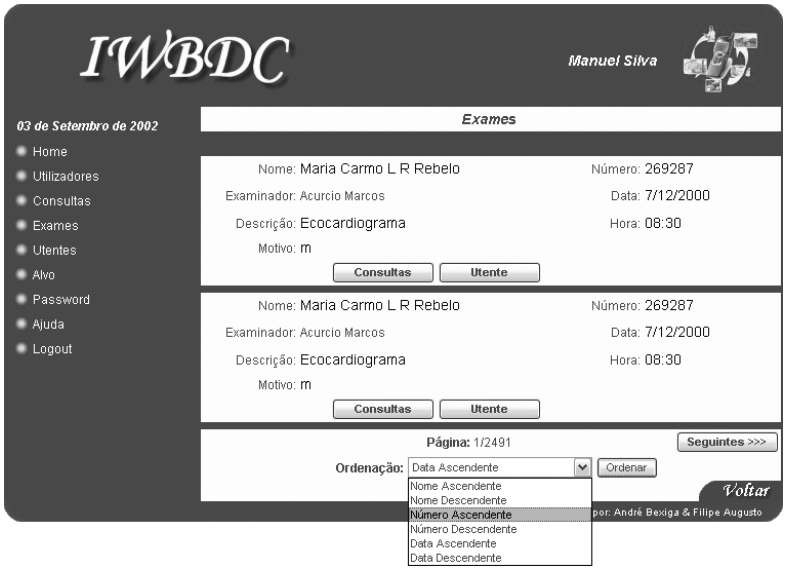

Figura 27 – Formatação nas diversas etapas de utilização em WEB

plataformas e teve em linha de conta os seguintes aspectos principais:

A integração das diferentes ferramentas

- A versatilidade da linguagem *XML* na aplicação à construção de conteúdos para multi-plataforma
- Custo no desenvolvimento associado à criação e manutenção das *XSL*
- A integridade do modelo de segurança a aplicar
- A consistência do modelo de utilizador
- Tipo de acesso à base de dados
- A organização de ficheiros

O trabalho desenvolvido foi testado em ambiente simulado e em ambiente real tendo obtido sucesso em ambos os ambientes, faltando apenas a integração com dados reais para testar as verdadeiras capacidades da aplicação.

REFERÊNCIAS

[1] "Technology Forecast 2001-2003 – Mobile Internet Unleashing the Power of Wireless", Price Water House Coopers.

[2] Heather Williamson, "XML :The complete reference", Osborne/McGraw-Hill.

[3] Forrest Houlette "Essential Skills For First-Time Programmers:SQL: A beginner's Guide ", Osborne/McGraw-Hill, 2001.

[4] Kris Jamsa, "Essential Skills For First-Time Programmers:WML & WML script", Osborne/McGraw-Hill, 2001.

[5] Richard Waymire, "Teach Yourself SQL Server 2000 in 21 Days", Sams

[6] Dave Mercer, "Essential Skills For First-Time Programmers: XML", Osborne/McGraw-Hill, 2001.

[7] Luca Passani, "Building Usable WAP Applications",, Cell Network A

[8] "Configure the Script mappings so we can use ASP with WAP", Microsoft Press.

[9] Wei Meng Lee, Ngee Ann, "Creating a Dynamic WAP Application", Polytechnic, Singapore.

[10] Deep Kapadia , "A report on WAP/WML ", CIS 690 Project Report

[11] Wei Meng Lee, Ngee Ann, "Tailoring Content Using XML and XSL", Polytechnic, Singapore

[12] Mikko Honkala, "Using XML to Develop Applications for WAP and WWW Environments", Helsinki University Of Technology, Department of Computer Science and Engineering, M.Sc. Thesis

[13] Karli Watson, "WAP for ASP Developers", In-House Author, Wrox Press Ltd.

[14] Ric Howell , "WAP Security", Concise Group Ltd.

[15] Andy Wigley , "The Future of WAP: v1.2 and Beyond", Secure Trading.

[16] Scott Hanselman, "XML – The Secret Sauce Anatomy Of An E-Commerce WEB Site", STEP Technology.