Faculdade de Engenharia da Universidade do Porto

Outliner
Browser de Documentação
Clínica Gráfica

Pedro Filipe Gomes Rodrigues

VERSÃO FINAL

Relatório de Projecto realizado no Âmbito do
Mestrado Integrado em Engenharia Informática e Computação

Orientador: Jorge Alves da Silva

Julho de 2008

Outliner

Browser de Documentação Clínica Gráfica

Pedro Filipe Gomes Rodrigues

Relatório de Projecto realizado no Âmbito do
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:
Presidente: Jaime dos Santos Cardoso
Arguente: Carlos Costa
Vogal: Jorge Alves da Silva

16 de Julho de 2008

Resumo

Este documento descreve todas as fases de desenvolvimento do projecto *Browser* de Documentação Clínica Gráfica, designado com o nome de código Outliner, desde a análise até à fase de implementação do projecto.

O Outliner é uma aplicação *web* que funciona como selector de diagramas médicos para a aplicação Siemens Diagrammer. Esta aplicação permite realizar anotações clínicas sobre diagramas da anatomia humana. O Outliner surge da necessidade da existência de uma forma fácil e intuitiva de escolher, entre os diversos diagramas possíveis, o diagrama a ser aberto no Diagrammer para a realização das anotações.

Na fase de análise do problema, foi realizado o levantamento do estado da arte na área das anotações clínicas sobre diagramas e analisado o estado actual do Diagrammer, o projecto em que o Outliner se insere.

Em seguida, na fase de especificação da solução, foram especificados os requisitos do projecto a desenvolver, realizados protótipos rápidos para previsão do funcionamento da futura aplicação, estudadas as tecnologias consideradas para o desenvolvimento do projecto e definida a sua arquitectura. A tecnologia escolhida, após realizado o estudo, foi o Microsoft Silverlight, pois é uma tecnologia *web* e que opera sobre imagens vectoriais, como pretendido neste projecto.

A fase de implementação guiou-se pelos protótipos realizados anteriormente, sendo o resultado final muito aproximado do planeado. Os diagramas disponíveis foram organizados em árvores de categorias – anatomia, área médica e patologias – e foi implementada a possibilidade de um utilizador seleccionar o diagrama que pretende de entre uma lista dos últimos que utilizou, o que lhe permite aceder mais rapidamente a diagramas que usou recentemente.

No entanto, o ponto mais relevante deste projecto é a possibilidade de escolha de diagramas através de imagens representativas da anatomia humana, desenhadas em XAML, um dialecto de XML que permite definir os elementos da interface gráfica em Silverlight. A manipulação destas imagens através de operações de ampliação, movimentação e selecção de zonas da imagem, melhoram a interactividade da aplicação. Estas operações foram implementadas através da gestão de eventos oferecida pelo Silverlight.

A implementação de todas estas funcionalidades é descrita mais detalhadamente neste documento, relatando os pormenores mais importantes e justificando as opções mais relevantes.

No final deste trabalho, é possível concluir que a aplicação foi implementada com sucesso, possuindo um elevado grau de interactividade, e variadas formas de selecção dos diagramas, como pretendido.

Abstract

This document describes all the development phases of the project “Browser de Documentação Clínica Gráfica”, that has the codename Outliner, from the analysis phase to the implementation phase of the project.

Outliner is a web application to select medical diagrams for Siemens Diagrammer. Diagrammer is an application that allows clinical annotations to be made over diagrams of the human anatomy. Outliner results from the need of existence of an easy and intuitive way to choose, from all the possible diagrams, the diagram, to be opened in Diagrammer, over which annotations will be done.

During the analysis of the problem, a revision of the state of the art, in the area of clinical annotations over diagrams, was done and the present status of Diagrammer, the project that Outliner is part of, was analyzed.

In the phase of solution specification, the requirements of the Project were defined, quick prototypes were developed, to predict the future functioning of the application, the technologies considered for the development were studied and the architecture of the solution was defined. After completing the study, the chosen technology was Microsoft Silverlight, because, as required by the project, it is a web technology that supports vector images.

The implementation phase was guided by the prototypes made before, and the final result is very close to the planned. The available diagrams were organized in trees – anatomy, medical area and pathologies – and the application was prepared to give the user the possibility of choosing the diagram that he/she wants to use, from a list of recently used diagrams; this can ease very much the diagram selection process, in some situations.

The most relevant point of this project is the choice of diagrams through images that represent the human anatomy, drawn in XAML, a XML dialect, which defines the elements of the Silverlight graphical interface. Those images can be manipulated (translated and zoomed) in order to select a region of interest, providing a high degree of interactivity to the application. These operations were implemented through the handling of events supported by Silverlight.

The implementation of all these functionalities is described more in detail in this document, reporting the most important aspects and justifying the most relevant options.

At the end of this work, it is possible to conclude that the application was successfully implemented, offering a high degree of interactivity, including various ways to select the diagrams, as desired.

Agradecimentos

Em primeiro lugar, quero agradecer a todos os meus colegas da Siemens SA onde efectuei o estágio, nomeadamente ao responsável pelo meu acompanhamento na empresa Eng. António Martins e ao Eng. Daniel Ramos pela colaboração ao longo do projecto.

Ao Prof. Doutor Jorge Alves da Silva um agradecimento especial por me ter orientado ao longo de todo o período do estágio e pela disponibilidade sempre demonstrada.

Agradeço também a todos os meus colegas da FEUP com quem me cruzei ao longo do meu percurso académico.

Finalmente, quero agradecer a todos os meus familiares, especialmente aos meus pais, irmãs e avós maternos, e à minha namorada por todo o apoio dado.

Conteúdo

Tabela de Figuras

1 Introdução

1.1 Enquadramento

O projecto descrito neste relatório surge englobado no projecto Siemens Diagrammer da empresa Siemens SA – Medical Solutions, que é uma aplicação vocacionada para a realização de anotações clínicas sobre diagramas anatómicos do corpo humano. O Diagrammer é uma aplicação médica que permite anotar graficamente, e directamente nos diagramas à disposição, informação, problemas e patologias de um determinado paciente. O projecto desenvolvido surge como um complemento desta aplicação, para colmatar a não existência de uma forma fácil e intuitiva de escolha do diagrama médico a abrir pelo Diagrammer.

Na Figura 1 apresenta-se um exemplo de um diagrama a ser anotado no Siemens Diagrammer. Neste exemplo, na parte da imagem representativa da coluna vertebral foi anotada, com um círculo vermelho, a patologia "Schober's signal". Esta anotação, após ter sido criada, passou a constar da lista de anotações realizadas até ao momento, presente na coluna do lado direito da aplicação, igualmente com um círculo da mesma cor.

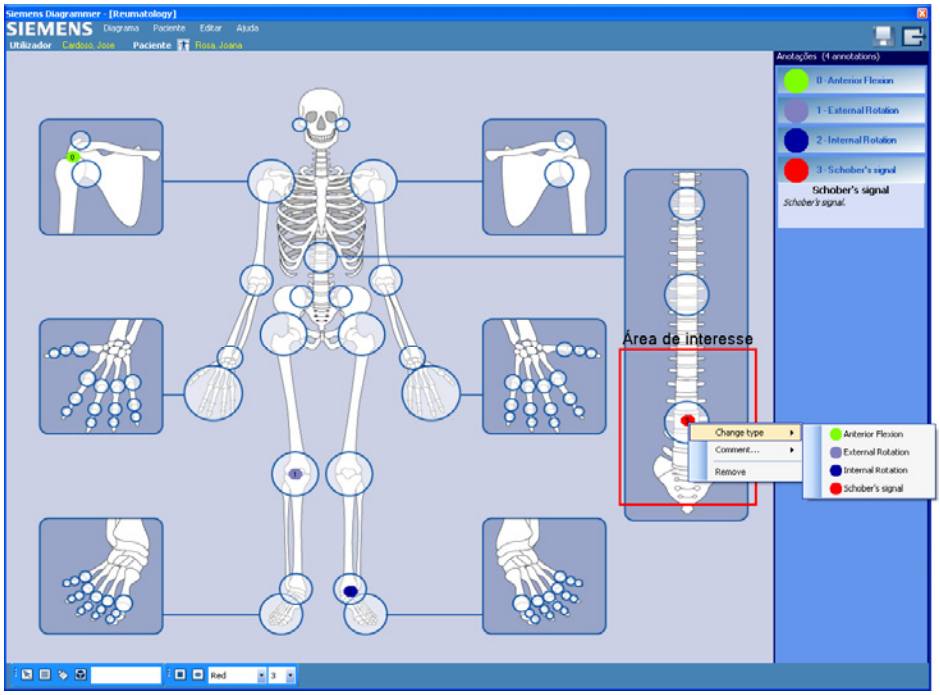

Figura 1 - Siemens Diagrammer

O exemplo da figura mostra um diagrama do esqueleto humano, no entanto existe um número variado de outros diagramas. É aqui que este projecto pretende complementar o Diagrammer, funcionando como uma ferramenta capaz de gerir a escolha entre todos os diagramas existentes.

1.2 Projecto

O projecto proposto pela Siemens SA com a designação de *Browser* de Documentação Clínica Gráfica, tomou posteriormente o nome de código Outliner, nome curto e simples para designar mais facilmente o projecto.

O Outliner, como mencionado anteriormente, tem a finalidade de possibilitar a um utilizador, pertencente a quadros hospitalares ou clínicos, uma forma fácil e intuitiva de seleccionar rapidamente o diagrama médico onde pretende realizar anotações no Diagrammer.

Esta selecção do diagrama pode ser realizada, através de uma imagem representativa do corpo humano, por categorias de diagramas, por pesquisa por nome do diagrama ou através de uma lista dos últimos diagramas utilizados. A selecção do diagrama através da imagem ocorre a partir dos seus mapas de selecção. Os mapas de selecção definem as zonas anatómicas da imagem que podem ser escolhidas e às quais os diagramas estão associados.

A realização do projecto passou por perceber o problema, analisar a melhor forma de o solucionar e definir bem a solução a desenvolver. Só posteriormente, com tudo bem definido, se iniciou a fase de implementação da aplicação, o que permitiu um desenvolvimento mais eficaz e célere. Para que tal sucedesse, foi realizado um plano de trabalhos no início do estágio que é apresentado em anexo.

1.3 Motivação e Objectivos

A motivação para o desenvolvimento deste projecto é dar resposta à necessidade de um selector de diagramas, entre um número indefinido de diagramas disponível, do Siemens Diagrammer.

Desta forma, o Outliner tem como objectivo apresentar um leque de soluções variadas para efectuar a escolha do diagrama pretendido pelo médico, tendo este a hipótese de usar o método da sua preferência e de acordo com o seu grau de conhecimento da aplicação.

1.4 Estrutura do Relatório

Este relatório encontra-se estruturado em cinco capítulos.

Neste primeiro capítulo, fez-se o enquadramento do projecto e apresentou-se o seu objectivo geral.

No segundo capítulo faz-se a análise do problema inerente a este projecto, descreve-se o estado da arte no campo das anotações clínicas sobre diagramas e analisa-se o estado actual do Diagrammer.

O terceiro capítulo é dedicado à especificação da solução que constituirá a aplicação Outliner. São, neste capítulo, citados os requisitos deste projecto, descrito o funcionamento previsto da aplicação, através da realização de um protótipo, é realizado o estudo das

tecnologias em que o projecto poderia ser implementado e é descrita a arquitectura da solução.

O quarto capítulo descreve detalhadamente a forma como foi realizada a implementação da aplicação. Neste capítulo, refere-se os pormenores da implementação mais relevantes e são justificadas as opções mais importantes, tomadas nesta fase.

Por fim, no quinto capítulo, apresenta-se as conclusões retiradas deste projecto, sendo realizada uma análise crítica e referido o trabalho futuro a desenvolver.

2 Análise do Problema

2.1 Estado da Arte

O projecto Outliner insere-se na área das anotações clínicas sobre diagramas, e é nesta área que é analisado o estado da arte.

Actualmente, não existem muitas aplicações que permitam efectuar anotações sobre diagramas médicos ou mesmo imagens do próprio paciente (radiografias e outros) e, das poucas que existem, a maior parte não apresenta uma solução completa que permita, a médicos de qualquer especialidade, tirar o partido de uma solução deste género.

O mercado apresenta soluções/aplicações que, apesar de permitirem realizar anotações sobre diagramas, são vocacionadas apenas para uma área médica específica, como é o caso dos serviços de urgência.

Por outro lado, é possível encontrar aplicações que utilizam imagens do próprio paciente, permitindo efectuar anotações sobre as mesmas. O problema deste tipo de aplicações prende-se com o facto de apenas ser possível reportar sobre imagens do próprio doente, não oferecendo a flexibilidade de que o médico necessita nos casos em que não é necessária a realização de exames que envolvam a aquisição de imagens, necessitando o médico de fazer apenas anotações sobre o paciente, por exemplo, relativas a sintomas que ele apresente em certas partes do corpo. De qualquer forma, não é no campo das anotações em imagens reais que o Outliner se insere, pelo que não se fará a revisão de aplicações existentes neste campo.

Além destas, existem algumas aplicações mais gerais, que trabalham sobre diagramas. Entre estas, encontram-se aplicações que trabalham com diagramas 2D e 3D, e com diferentes formas de anotação. Existe um conjunto de soluções possíveis neste âmbito, no entanto não existe uma solução óptima. Em seguida, faz-se uma breve descrição de algumas aplicações existentes no mercado actual.

2.1.1 Produtos para Anotações sobre Diagramas

2.1.1.1 Wellsoft

Esta é uma aplicação direccionada para os serviços de urgência [1], permitindo realizar operações como:

- Captar fotografias do paciente e das lesões que ele apresenta;
- Anotar graficamente sobre diagramas da representação exterior do paciente, os vários tipos de lesões por este sofridas;

- Adicionar texto às anotações gráficas;
- Alterar a orientação do diagrama da esquerda para a direita;
- Aproximar e afastar do diagrama (*zoom in* e *zoom out*);
- Alternar entre todas as imagens referentes a um registo de um paciente;
- Adicionar desenhos manuais (*sketches*) de lesões aos diagramas.

2.1.1.2 Alert EDIS

Tal como a aplicação *Wellsoft*, esta é também uma aplicação que visa os serviços de urgência [2]. Algumas das capacidades do Alert EDIS são:

- Anotar graficamente, sobre diagramas da representação exterior do paciente, os vários tipos de lesões por este sofridas;
- Mudar de diagrama através de um movimento de *drag and drop* a partir de uma pequena lista de diagramas disponíveis para a janela de visualização;
- Aproximar e afastar do diagrama (*zoom in* e *zoom out*).

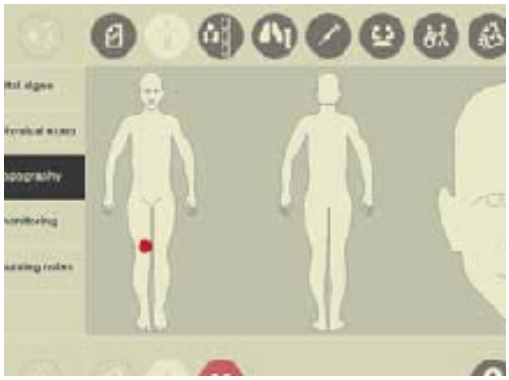

Figura 2 - Alert EDIS

2.1.1.3 VistA Imaging Diagram Annotation Tool

Esta ferramenta da VistA Imaging possibilita efectuar anotações em diagramas *template* acedidos *online* [6]. Os *templates* ficam armazenados num repositório partilhado do servidor, podendo ser acedidos *online* por todo o pessoal médico que tiver permissões. A partir destes, são feitas as anotações que são posteriormente guardadas nos registos do paciente. Ao ser feita a gravação, é criada uma nova imagem, nunca alterando assim o diagrama *template* base.

A Diagram Annotation Tool suporta imagens do tipo JPEG e TIFF, permitindo efectuar as seguintes operações:

- Criar anotações com polígonos, rectas e setas;
- Fazer o *zoom* da imagem (aproximar e afastar);
- Mover, redimensionar e apagar anotações.

Esta apresenta-se como uma ferramenta de anotação demasiado simples e pouco flexível, pois não permite editar diagramas anotados após terem sido gravados.

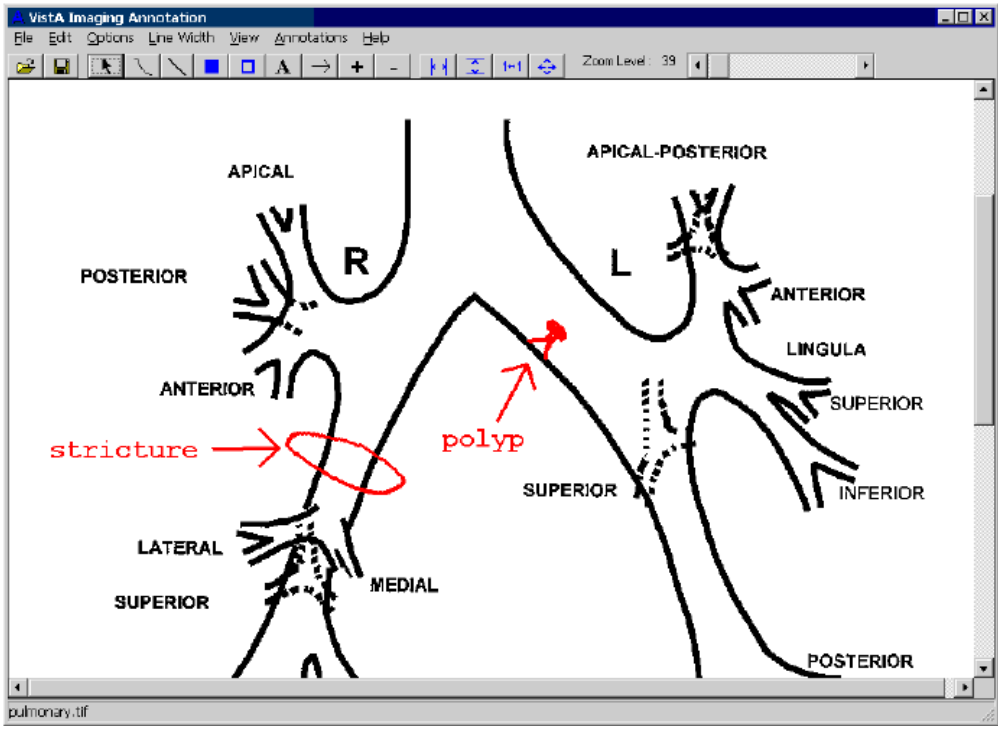

Figura 3 - VistA Imaging Annotation

2.1.1.4 e-Trace - Pen Based Graphical Annotation

Neste caso, as imagens com que esta aplicação trabalha, assim como as anotações possíveis, podem estar num espaço 2D ou 3D. Além disto, o e-Trace diferencia-se também por usar anotações do estilo *free-hand*, ou seja, efectuadas de forma livre no desenho através do uso de uma "caneta" virtual [8].

As três formas possíveis de anotação suportadas por esta aplicação são:

1. Anotações em imagens 2D

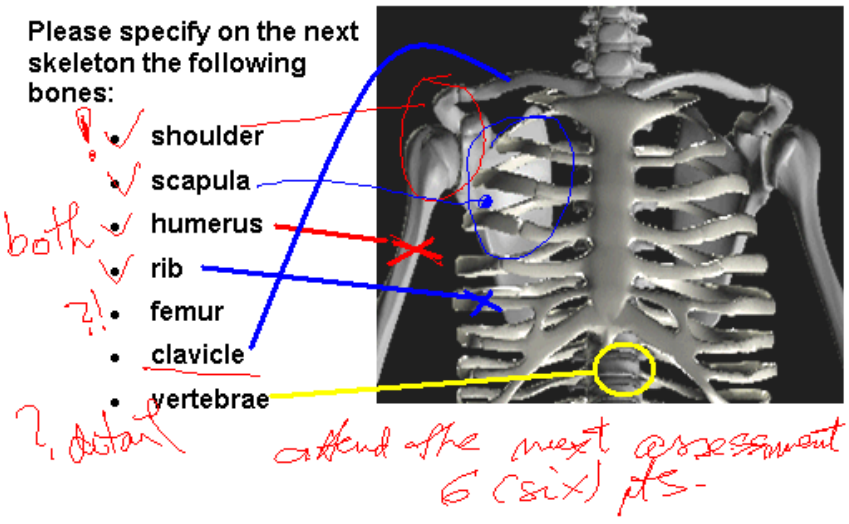

Figura 4 - e-Trace: anotações em documentos

2. Anotações 2D em objectos 3D

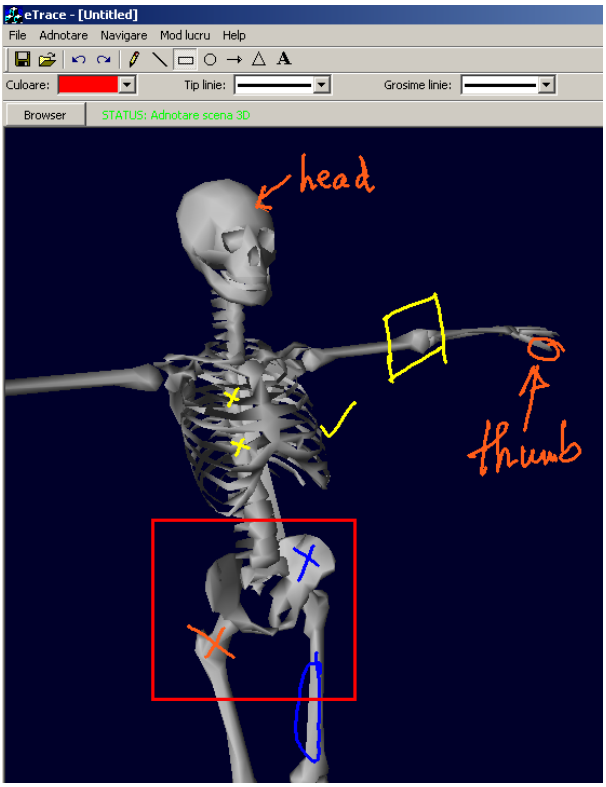

Figura 5 - e-Trace: anotações 2D sobre objectos 3D

3. Anotações 3D em objectos 3D

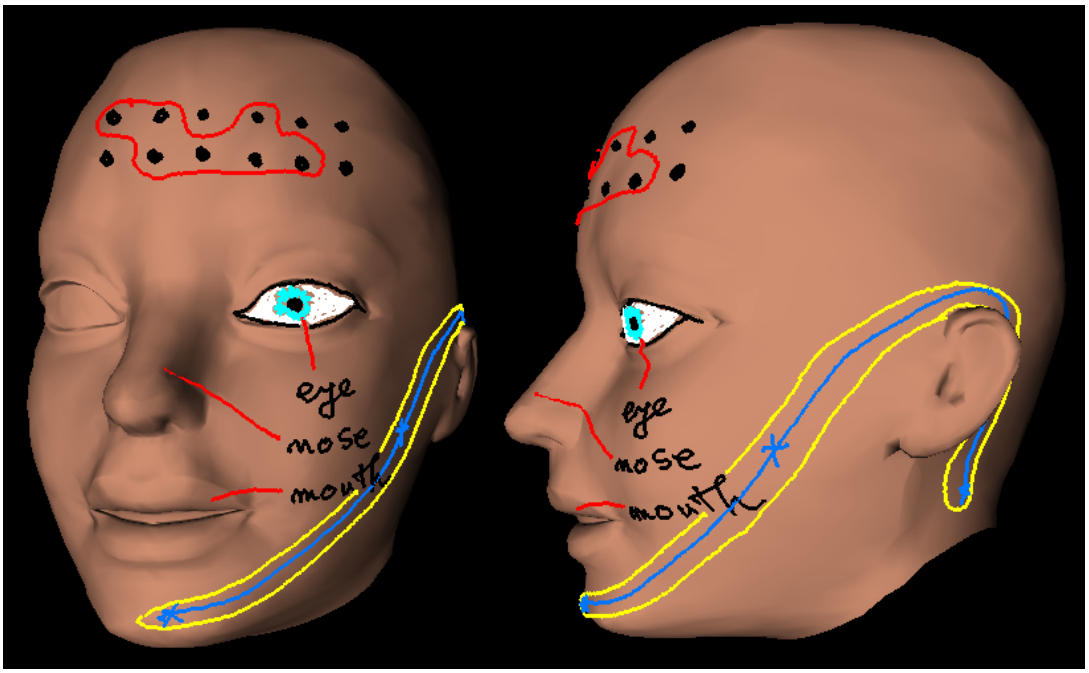

Figura 6 - e-Trace: anotações 3D aplicadas à superfície do objecto 3D

O e-Trace foi também desenvolvido em linha de conta com técnicas de interacção com o utilizador em anotações gráficas, tais como possuir uma apresentação simples, permitir trabalho cooperativo e, escolher e verificar os contornos.

2.1.1.5 ConceptDraw Medical

O ConceptDraw Medical é um programa de manipulação de diagramas médicos, que pode ser definido como um programa do mesmo tipo do Microsoft Visio mas operando sobre diagramas médicos. Esta aplicação abrange uma série de diagramas médicos que podem ser colocados e combinados numa página em branco, ou área de trabalho, da forma que se pretender. As anotações podem ser aqui realizadas também de diversas formas, através de linhas, polígonos, texto e diversos símbolos presentes na aplicação [9].

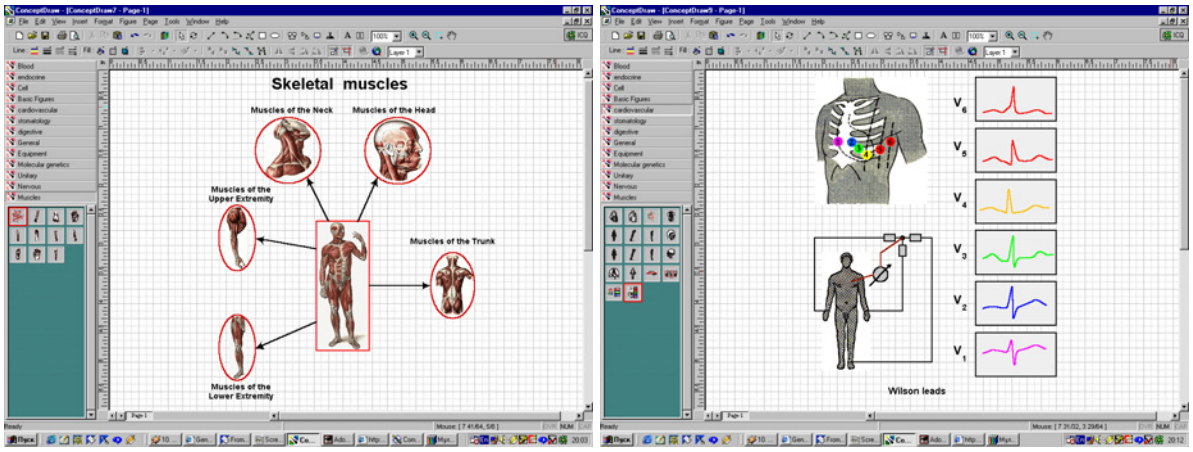

Figura 7 - ConceptDraw Medical

2.2 Estado Actual do Diagrammer

Como visto anteriormente, os produtos existentes na área das anotações médicas sobre diagramas não permitem a realização de anotações contextualizadas. Esta é uma das características que diferencia o Siemens Diagrammer de todos eles, além de que pretende ser uma solução geral para todas as áreas médicas e não específica para só uma delas, como é o caso de algumas das aplicações mencionadas anteriormente.

Actualmente o Diagrammer é uma aplicação ainda numa fase “de infância”, mas já em funcionamento no Hospital da Luz. Este conta de momento com a possibilidade de efectuar anotações contextualizadas e com um sistema de cores padrão, em que a mesma cor é atribuída à mesma patologia ou sintoma em qualquer diagrama, o que facilita a colaboração entre diferentes médicos.

Em termos do número de diagramas actualmente aceites e disponíveis no Diagrammer, este número é reduzido, estando limitado a três diagramas. No entanto, o número de diagramas possíveis no futuro é imenso e é com vista nessa evolução do Diagrammer que o Outliner aparece para dar uma resposta preparada para o presente e para o futuro.

2.3 Estudo das Tecnologias

Nesta secção são analisadas as tecnologias que foram consideradas para o desenvolvimento da aplicação. Estas devem estar de acordo com os requisitos e restrições de desenho definidos na secção seguinte.

2.3.1 Microsoft Silverlight

O Silverlight é um *plug-in* da Microsoft, *multi-browser* e multi-plataforma, para possibilitar o desenvolvimento de RIAs (*Rich Internet Applications*) e da nova geração de aplicações multimédia tendo por base as linguagens .NET. Inicialmente conhecido por WPF/E (*Windows Presentation Foundation Everywhere*), o Silverlight é um subgrupo da tecnologia WPF, introduzida pela Microsoft para proporcionar o desenvolvimento de aplicações de *desktop* mais ricas e poderosas graficamente.

O grande objectivo do Silverlight é permitir que finalmente, tanto *designers* como programadores, tenham uma ferramenta comum de desenvolvimento, facilitando assim a interacção entre eles. Ao mesmo tempo, pretende dotar a grande base de programadores .NET de uma ferramenta de fácil utilização, para eles próprios produzirem aplicações visualmente mais apelativas.

Apesar disto, só a partir da versão 1.1, agora designada por versão 2, é que o desenvolvimento de aplicações em Silverlight com as linguagens .NET se tornou possível, sendo assim muito mais fácil aos programadores habituados às linguagens .NET, como C# ou VB.NET, desenvolverem aplicações em Silverlight, usando inclusive a poderosa ferramenta de desenvolvimento da Microsoft para .NET, o Microsoft Visual Studio.

Mas o Silverlight não é uma tecnologia só para aplicações ASP.NET, ela pode ser integrada nas aplicações já existentes, sejam em PHP, JavaScript ou XHTML.

Os pré-requisitos para visualização de aplicações Silverlight são a instalação do *plug-in* com um tamanho de 4 a 6 MB para Windows e de cerca de 10 MB para Macintosh. Não requer a instalação de qualquer *framework* .NET no cliente.

As desvantagens do Silverlight passam por ser uma tecnologia recente, ainda pouco difundida, mas isto não é realmente uma desvantagem para a Microsoft. Tendo em conta a quantidade de utilizadores do Windows, o Silverlight poderá ser facilmente difundido para uma percentagem significativa de utilizadores via as actualizações do Windows.

Actualmente, o Silverlight é suportado nos *browsers* Internet Explorer, Firefox e Safari e nos Sistemas Operativos Windows e Mac OS. Apesar de não existir, de origem, suporte para Linux, está também em desenvolvimento o projecto Moonlight, com a aprovação da Microsoft, para portar o Silverlight para o Linux.

Sendo um subgrupo do WPF, o Silverlight não inclui as bibliotecas de suporte para renderização 3D, no entanto, projectos nesta área começam a ser desenvolvidos para permitir 3D no Silverlight [10].

O Silverlight possui o sistema gráfico do WPF que integra multimédia, gráficos, animações e interactividade num mesmo pacote. A linguagem XAML, *eXtensible Application Markup Language*, é a utilizada pela tecnologia para criar os gráficos vectoriais e animações nas suas aplicações.

2.3.2 Adobe Flash

O Flash da Adobe, é o adversário directo do novo Microsoft Silverlight. Sendo uma tecnologia já com 10 anos de existência, encontra-se actualmente em aproximadamente 90% dos computadores com ligação à internet (através da instalação do seu *plug-in*), contrastando

de momento com o Silverlight, que como mencionado anteriormente, é ainda uma tecnologia muito recente e por isso pouco difundida.

O Flash é suportado em todos os *browsers* e plataformas, incluindo o Linux, tendo neste aspecto uma pequena vantagem sobre o Silverlight que ainda não tem este suporte. No entanto, esta vantagem é pouco significativa para o projecto Outliner.

O Flash sempre foi mais direccionado para *designers*, para desenvolvimento de aplicações multimédia de elevada qualidade, sendo pouco *user-friendly* para os programadores. No entanto, esta lacuna foi de certa forma minimizada com a introdução do Flex que é uma ferramenta de desenvolvimento de Flash através da sua linguagem de programação, o ActionScript. Apesar disso, esta não pode ainda rivalizar com a qualidade da ferramenta mais madura que é o Visual Studio da Microsoft para .NET, sendo o ActionScript também uma linguagem que apresenta maior complexidade e dificuldade de aprendizagem comparativamente às linguagens .NET.

Tal como no Silverlight, as aplicações do Flash são ricas em conteúdos multimédia e em interactivadade, usando imagens vectoriais e animações sobre estas. Assim como o Silverlight, o Flash não tem suporte para renderização 3D; no entanto, existem também formas de representação 3D em Flash.

2.3.3 SVG

O SVG é um dialecto de XML e apresenta-se como o *standard* para representação de imagens vectoriais e respectivas animações na *Web*. No entanto, é muito pouco usado por não existirem ferramentas de desenvolvimento de gráficos e animações em SVG que atinjam níveis de qualidade que se aproximem sequer da qualidade das ferramentas acima citadas. Apresenta também alguns problemas de compatibilidade com alguns dos *browsers* actuais. O SVG está também limitado, pois não suporta conteúdo multimédia.

2.3.4 Java FX

O Java FX é um conjunto de produtos e tecnologias da Sun Microsystems, baseadas em Java. O objectivo é permitir o desenvolvimento de RIAs (Rich Internet Applications), possibilitando a criação de interfaces com elevado número de animações, seja para *desktops* ou dispositivos móveis.

Esta é uma tecnologia ainda mais recente que o Microsoft Silverlight, que carece ainda de bastante desenvolvimento, pois apresenta muitas limitações em comparação com os concorrentes mais directos: o Adobe Flash e o Microsoft Silverlight.

Essas limitações passam pela inexistência de ferramentas de desenho de imagens vectoriais, assim como de uma ferramenta adequada para criação de animações com base em *timelines*. O Java FX não consegue também fazer frente aos seus concorrentes em termos de leveza do ambiente de execução. Além disso, o Java FX carece, de momento, de suporte a conteúdos multimédia [12].

2.3.5 Tabela Comparativa Flash vs Silverlight

Como pôde ser constatado anteriormente, apenas o Silverlight e o Flash são verdadeiras opções para o desenvolvimento do Outliner, visto que são as únicas que

apresentam um conjunto de funcionalidades e de ferramentas de desenvolvimento capazes e com garantias de qualidade.

Por essa razão, compara-se na tabela seguinte, o Silverlight e o Flash relativamente às funcionalidades mais relevantes para o projecto, como base para a escolha final da tecnologia a usar.

2.3.5.1 Funcionalidades mais relevantes para o Outliner

	Silverlight	Flash
Animações/gráficos 2D ricos com áudio e vídeo	✓ Sim	✓ Sim
Suporta imagens vectoriais	✓ Sim	✓ Sim
Recursos de rede	✓ Capacidade de ler dados de *web services* e XML a partir de um URL. O Silverlight 2 suporta também comunicação por *sockets*.	✗ Consegue ler e usar dados de XML ou texto a partir de um URL. Isto, o Silverlight também consegue ler.
Biblioteca de controlos	✓ Os controlos padrão (botões, *checkboxes*, listas, grelhas, etc.) estão incluídos no Silverlight 2 beta1.	✓ O Flash possui uma biblioteca rica em controlos.
Formatos de imagem	✗ Suporta apenas os formatos JPG, PNG e GIF.	✓ Suporta todo o tipo de formatos de imagem.
Programação de sockets	✓ O Silverlight 2 suporta comunicação via *sockets*.	✓ O Flash permite criar um objecto *socket* em XML.
Desempenho	✓ O Silverlight é mais rápido a realizar cálculos computacionalmente intensivos [13].	✗ O Flash é melhor a desenhar itens em movimento quando os cálculos são triviais.
Definição de figuras	✓ O Silverlight usa XAML, que é baseado em texto e o que produz é um simples objecto XML.	✗ O Flash guarda as suas figuras em binário. Para escrever uma definição de uma figura é necessário adquirir a licença de um SDK de formatos de ficheiros Flash, ou criar um. Mesmo podendo não ser muito difícil, requer aprendizagem.
Debugging	✓ Mais simples	✗ Mais complexo

	Silverlight	Flash
Linguagens de desenvolvimento	✓ O Silverlight tem disponível um conjunto rico de linguagens de desenvolvimento, desde JavaScript até código VB.Net e C#. Incluindo também as linguagens dinâmicas (IronPython, IronRuby e Managed JScript).	✗ Apenas o Action Script pode ser usado como ferramenta de programação em Flash.
Linguagens de Interface com o Utilizador (UI)	✓ O Silverlight usa o XAML que é uma linguagem declarativa.	✗ O Action Script é mais uma vez a linguagem usada em Flash. O Action Script é uma linguagem imperativa, o que torna complexa a gestão da interface gráfica, obrigando os programadores a ter que gerir todos os aspectos da API, como era o caso nos primeiros tempos do DOS e do Windows.
Versão actual	✗ Silverlight 2 Beta 1	✓ Flash 9

2.3.5.2 Possibilidades 3D

	Silverlight	Flash
Suporta renderização 3D por hardware	✗ Não	✗ Não
3D simulado	✓ Sim [14] [15]	✓ Sim [16] [17]
Suporte futuro de 3D	✓ O Silverlight é um subconjunto do WPF e este tem suporte 3D completo. O Silverlight não inclui de momento o suporte 3D do WPF, mas esta possibilidade está em aberto para o futuro. Existem também rumores sobre a adição de suporte ao DirectX para o Silverlight [19].	✓ Existem rumores sobre o Flash vir a ter suporte real de 3D, talvez vindo a suportar Open GL ou um novo motor 3D [18] [19].

2.3.6 Justificação da Escolha Tecnológica

Com base em tudo o que foi mencionado anteriormente, tanto o Silverlight como o Flash apresentam-se como opções sólidas para o desenvolvimento do projecto Outliner. Ambos suportam imagens vectoriais e são tecnologias para desenvolvimento *Web*. No entanto, o Silverlight foi a tecnologia escolhida para o desenvolvimento do projecto.

Esta escolha deve-se principalmente ao facto de o Silverlight apresentar linguagens e ferramentas de desenvolvimento mais maduras, as linguagens e ferramentas .NET, que facilitam e agilizam em muito o desenvolvimento da aplicação. A experiência prévia do autor nas linguagens .NET, conjuntamente com o facto de a interface gráfica ser declarada em XAML (um dialecto XML), facilita muito a aprendizagem e compreensão da nova tecnologia, o Silverlight, em relação ao Flash e à respectiva linguagem, Action Script.

3 Especificação da Solução

3.1 Especificação de Requisitos

Nesta secção são listados os requisitos do projecto, estando estes qualificados a nível de prioridade. Apresenta-se a lista de requisitos funcionais e não-funcionais, as restrições de desenho e os casos de uso da aplicação.

3.1.1 Requisitos Funcionais

ID	Requisito	Prioridade
RF1	Implementação da janela principal do programa, com as áreas principais definidas	Muito Alta
RF2	Infra-estrutura de suporte de coordenadas normalizadas	Alta
RF3	Infra-estrutura de suporte de imagens vectoriais em 2D (criação de um ficheiro XML com a descrição vectorial da imagem, capacidade de ler a imagem e carregá-la). Implica a criação de duas imagens "mais reais" de teste num ficheiro XAML	Muito Alta
RF4	Escolha por diagramas – implementar funcionalidade de *zoom* (utilizando a roda do rato)	Muito Alta
RF5	Deve existir um ficheiro XML contendo todas as coordenadas de mapas de cada diagrama	Alta
RF6	Escolha por diagramas – clique lança uma nova imagem mais focalizada	Muito Alta
RF7	Escolha por diagramas – ter a possibilidade de seleccionar com um *slider*, qual a “camada” que se pretende visualizar (ex: exterior, músculos, órgãos ou esqueleto)	Alta
RF8	Criar ficheiros XML que descrevam as estruturas hierárquicas das categorias, e os últimos diagramas usados (*last used*).	Muito Alta
RF9	Carregar o ficheiro XML para os separadores	Muito Alta
RF10	Invocar o Diagrammer após ter sido seleccionado um diagrama para anotar (indicando o diagrama escolhido)	Muito Alta
RF11	Implementar o campo de filtragem da janela de diagramas. Caixa de texto onde se escreve letras respeitantes ao diagrama ou categoria que se procura	Alta
RF12	Escolha por diagramas – apresentar uma ajuda quando o rato está numa zona do diagrama	Média

ID	Requisito	Prioridade
RF13	Ao clicar num conteúdo de um separador, será aberta uma imagem no lado esquerdo com uma determinada ampliação e localização	Alta
RF14	Área onde surja uma breve descrição do diagrama e eventualmente uma utilização típica de anotação clínica	Alta
RF15	Possibilidade de estarem associados *thumbnails* dos diagramas, na própria caixa de escolha por categorias e nomes	Média
RF16	Histórico das últimas imagens, ampliações e localizações	Média
RF17	Escolha por diagramas – configurador gráfico capaz de definir de forma flexível, áreas de um diagrama, que lançam sub-diagramas	Média

3.1.2 Requisitos Não-Funcionais

ID	Requisito	Prioridade
NF1	Usabilidade	Muito Alta
NF2	Performance	Alta
NF3	Confiabilidade	Alta

3.1.3 Restrições de Desenho

ID	Restrição
RD1	Suporte de imagens vectoriais
RD2	A aplicação deve ser *Web*

3.1.4 Casos de Uso

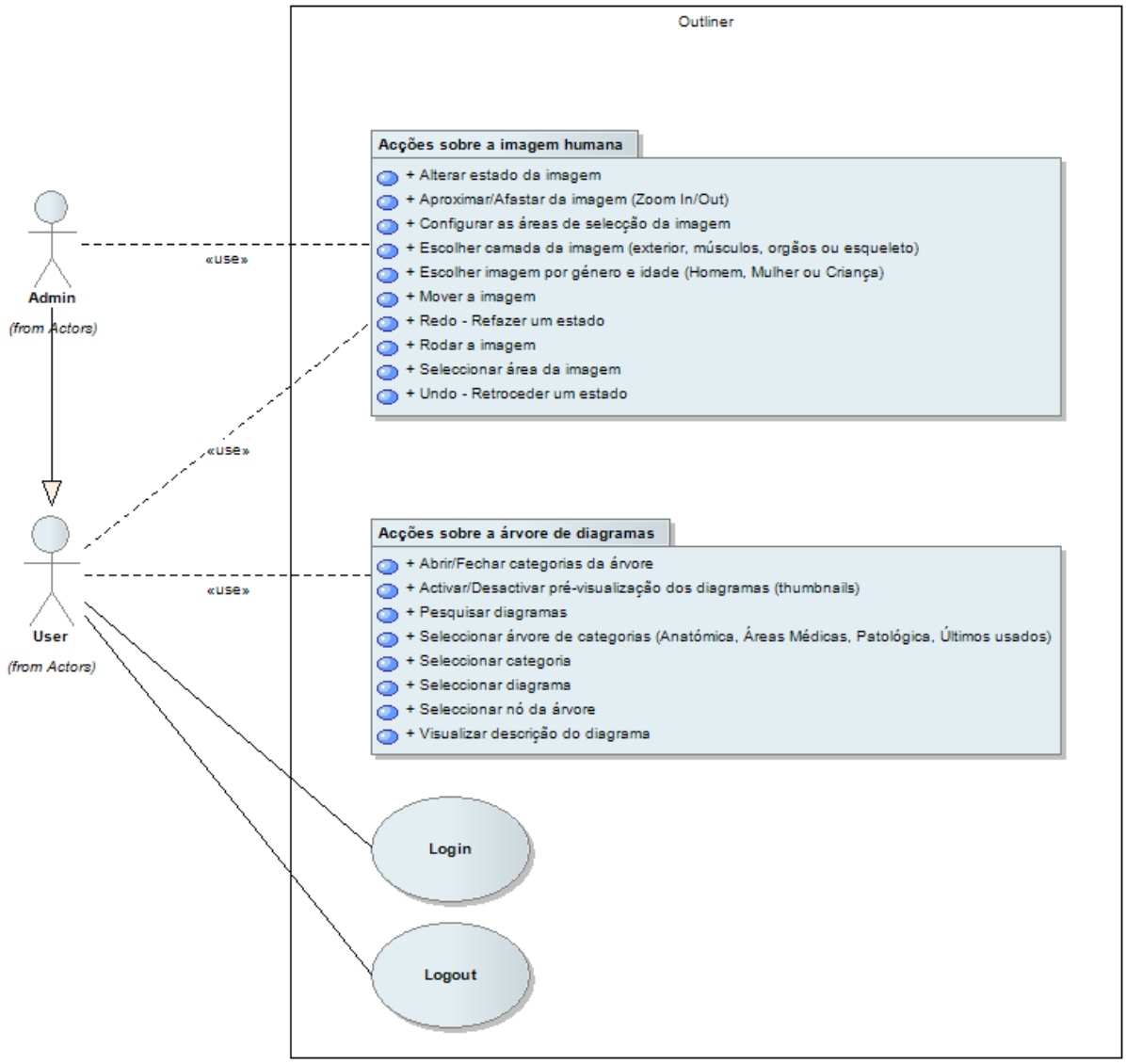

Figura 8 - Casos de uso

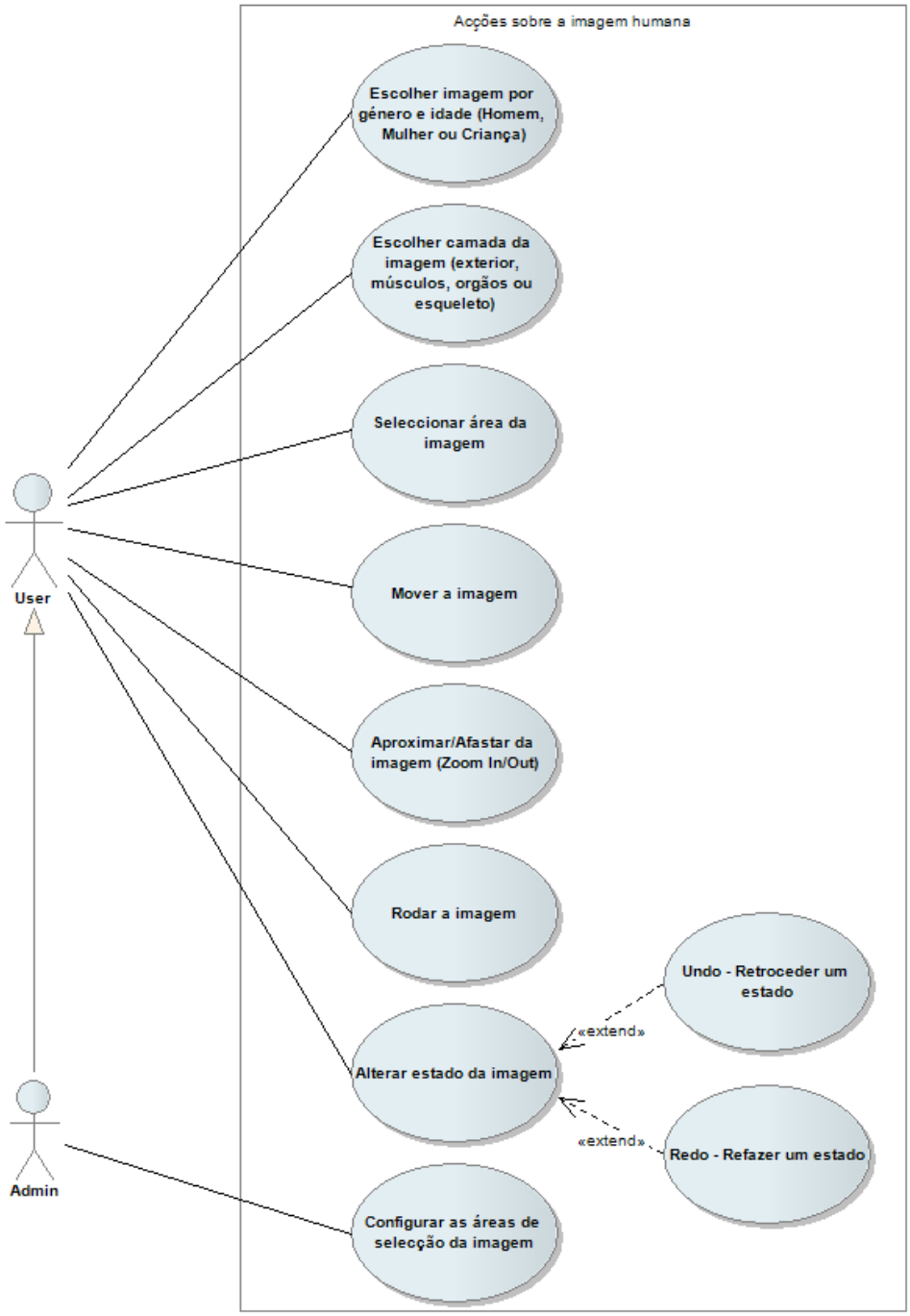

Figura 9 - Casos de uso: acções sobre a imagem humana

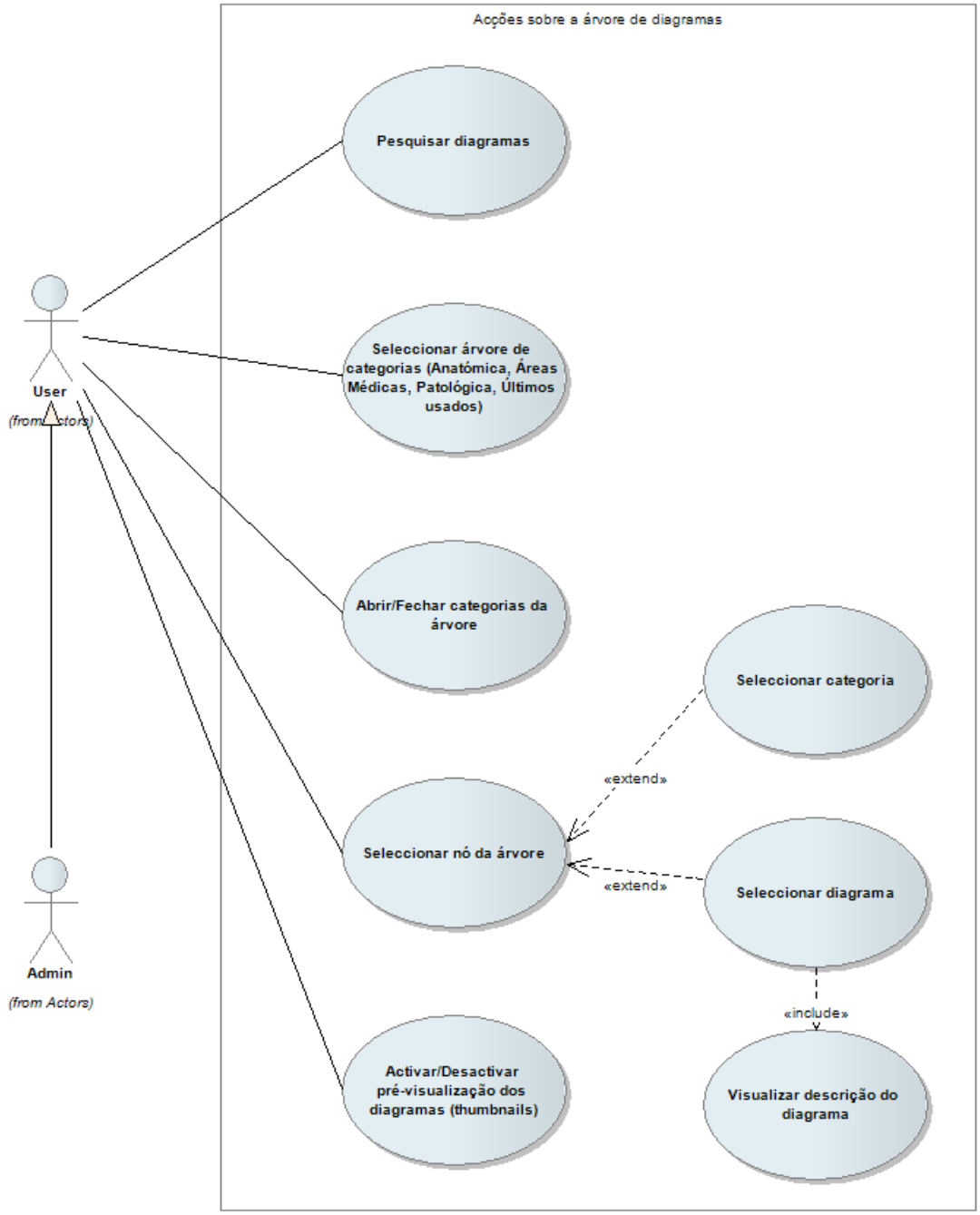

Figura 10 - Casos de uso: acções sobre a árvore de diagramas

3.1.4.1 Actores

- **User** – Utilizador do sistema, que neste caso, será um médico ou possivelmente um enfermeiro.
- **Admin** – Administrador do sistema responsável por configurar a aplicação e gerir os utilizadores.

3.2 Protótipos para Previsão de Funcionalidades

Tendo por base os requisitos iniciais e objectivos da aplicação, foram realizados protótipos rápidos, desenhados digitalmente, para aprofundar estes requisitos e recolher impressões sobre os requisitos e sobre as opções disponíveis para a solução proposta, detalhando assim o projecto a desenvolver.

São apresentadas de seguida as várias propostas de navegação para a correcta escolha do diagrama em que se pretende tomar notas, e de um esboço de uma interface gráfica para a aplicação Outliner. São apresentados diferentes tipos de filtragem da lista de diagramas e diferentes tipos de listagem dos mesmos.

A interface gráfica proposta para a aplicação é a apresentada na Figura 11.

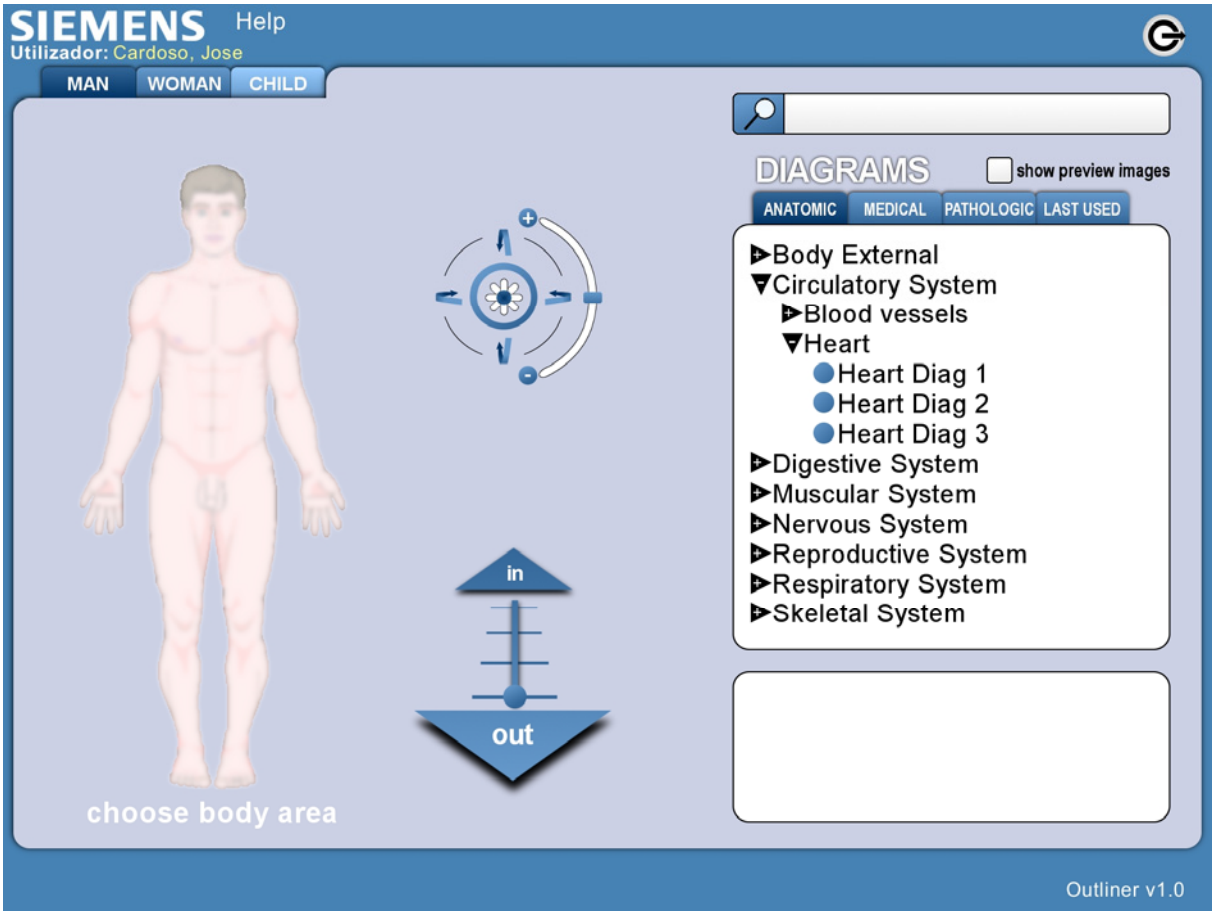

Figura 11 - Proposta de interface gráfica do Outliner

3.2.1 Tipos de Filtragem

Como pode ser visualizado na imagem, a aplicação apresenta 4 formas de filtragem dos diagramas:

1. Filtragem gráfica – através da imagem;
2. Filtragem por temas/categorias – através das *tabs* “Anatomic”, “Medical”, “Pathologic”;
3. Filtragem pelos últimos visitados – através da *tab* “Last Used”;

4. Filtragem por pesquisa do diagrama – através do campo de pesquisa.

De seguida será apresentada uma breve descrição de como estes diferentes tipos de filtragem podem funcionar.

3.2.1.1 Filtragem gráfica

Este tipo de filtragem é efectuado usando a representação do corpo humano, presente do lado esquerdo da Figura 11, e com o auxílio do *slider in/out*, que permitirá "entrar" e "sair" do corpo humano, desde a camada exterior até à camada óssea, e da roda de controlo da imagem (situada acima do *slider*), que possibilita operações como ampliação/redução (*zoom*), rotação e movimentação da imagem.

Navegando por cima da representação do corpo humano serão realçadas partes do mesmo, indicando a designação da parte realçada. Por exemplo, passando o rato por cima da zona da cabeça a representação será a ilustrada na Figura 12.

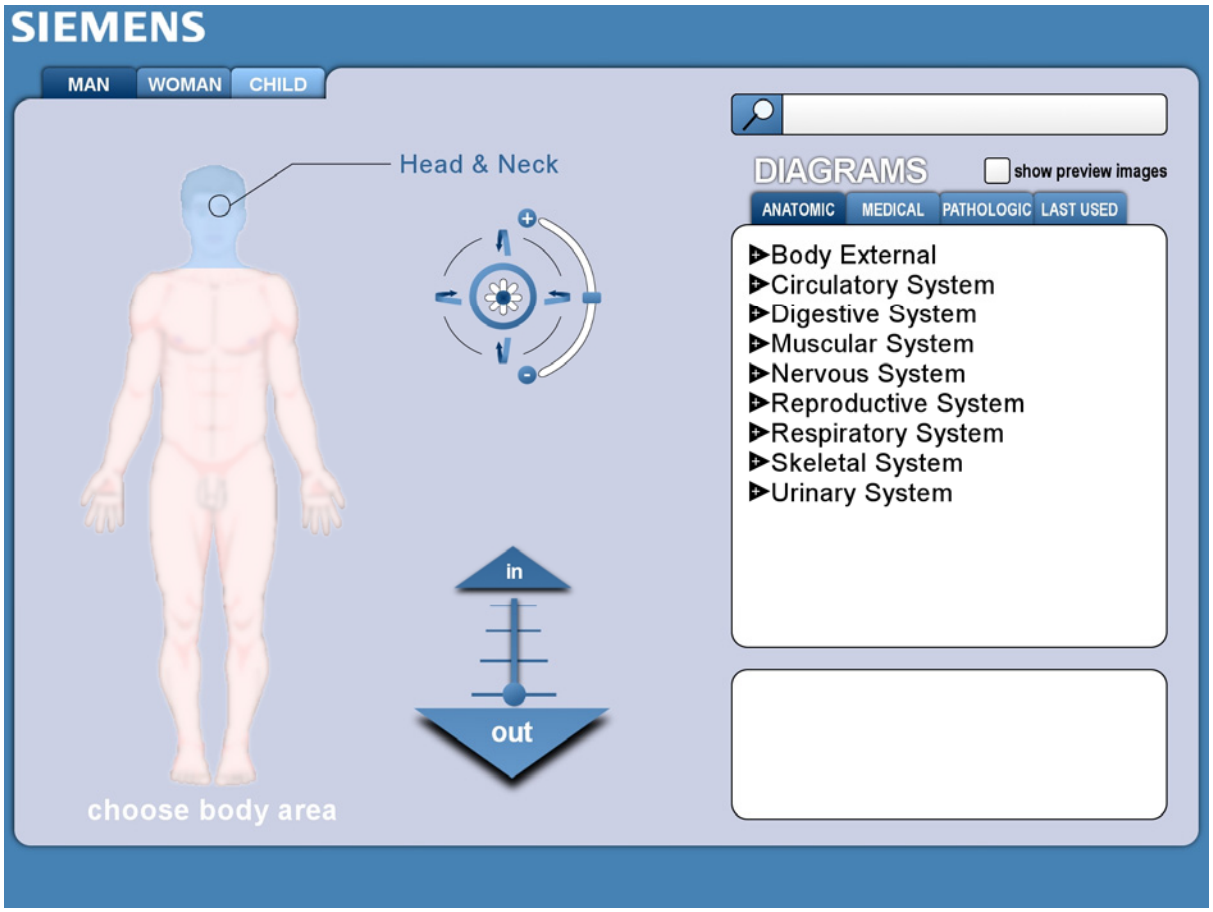

Figura 12 - *Mouse over* na cabeça, realçando essa zona

Com um clique do rato sobre esta zona, ela será seleccionada. Depois de escolhida a parte do corpo a analisar, será apresentada uma imagem ampliada e com maior detalhe da zona em questão. Além disso, serão filtrados os diagramas da caixa presente no lado direito do ecrã sendo apresentados apenas os diagramas referentes à zona seleccionada. O resultado desta selecção é o apresentado na Figura 13.

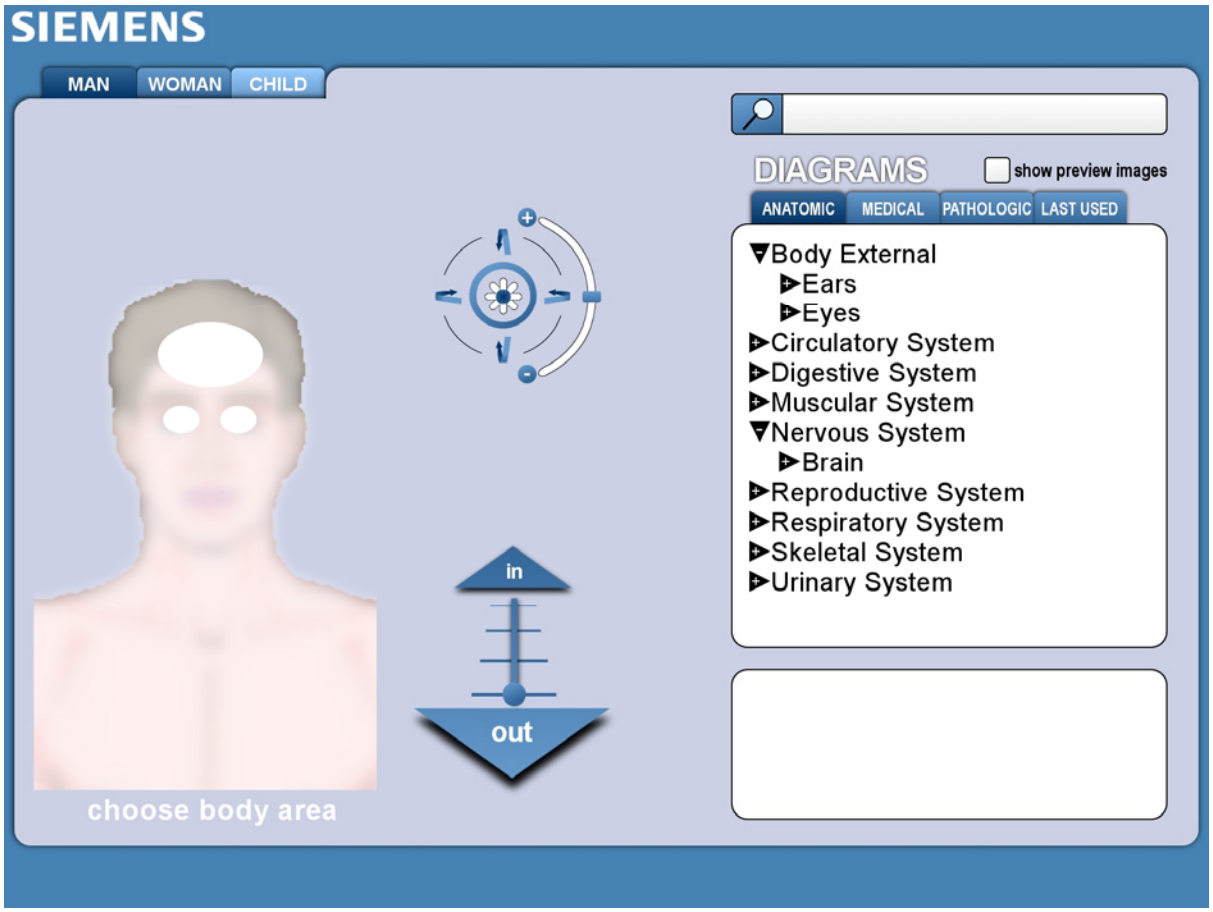

Figura 13 - Representação aumentada e detalhada da cabeça

Tal como referido, e ilustrado na imagem anterior, a título de exemplo, ao ser seleccionada a cabeça, os diagramas disponíveis na lista passam a ser os referentes ao cérebro (*Brain*), olhos (*Eyes*) e ouvidos (*Ears*), independentemente do sistema em que se encontrem. Por exemplo, orelhas e olhos são referentes ao sistema "Body External" e o cérebro referente ao sistema "Nervous System".

Por sua vez, no novo esquema gráfico apresentado (o detalhe da cabeça) é também possível escolher entre os diferentes órgãos presentes, passando o rato por cima desses órgãos. Um dado órgão pode pertencer a diferentes sistemas (ver Figura 14).

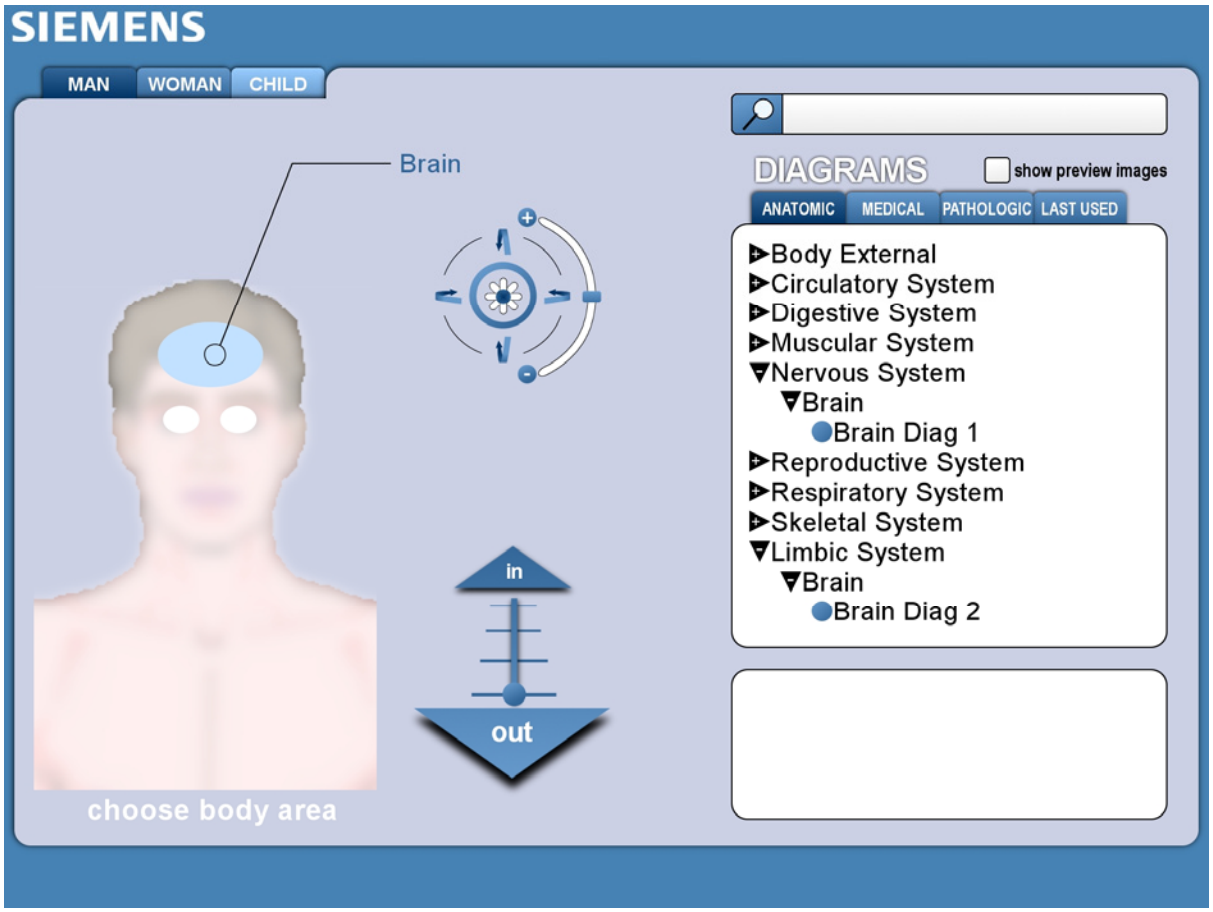

Figura 14 - Cérebro (*Brain*) escolhido

Mais uma vez, ao clicar numa dessas zonas do esquema gráfico, é filtrada a lista de diagramas, apresentando apenas os diagramas referentes ao órgão escolhido.

3.2.1.1.1 O Slider in/out

O *slider in/out* presente na interface (ver Figura 15) e que pode ser visualizado nas figuras anteriores, tem a função de permitir mudar a representação do corpo humano entre as várias camadas disponíveis (exterior, muscular, órgãos e esqueleto).

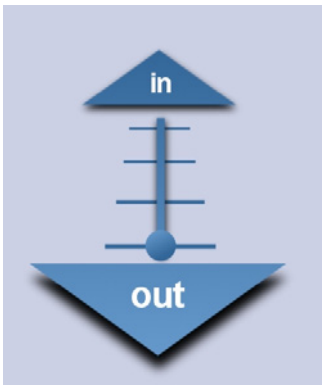

Figura 15 - *Slider in/out*

Nas imagens anteriores o *slider* está posicionado na posição mais exterior (*out*). Ao mover o *slider* uma unidade na direcção interior (*in*) será apresentada a representação da camada seguinte, neste caso, a camada muscular, como ilustrado na Figura 16.

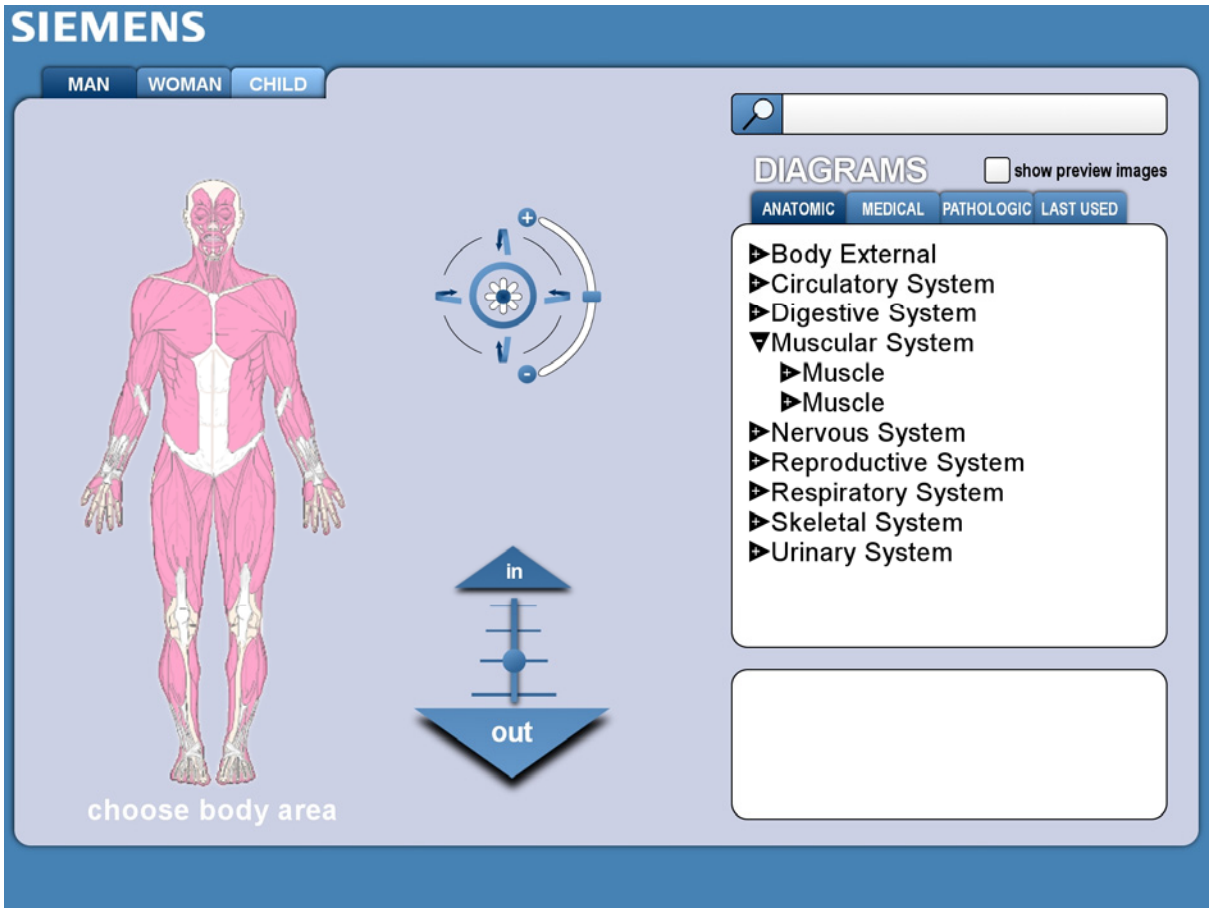

Figura 16 - Representação de um nível mais interior (músculos)

Ao mover o *slider* para a posição mais interior (*in*), será mostrado o esqueleto humano (ver Figura 17).

De notar que, ao mesmo tempo que se alterna entre as diferentes camadas de representação do corpo humano, a lista de diagramas é filtrada de acordo com essa mesma representação. Por exemplo, no caso da Figura 17, em que é apresentado o esqueleto humano, os diagramas abertos na lista são os referentes ao esqueleto.

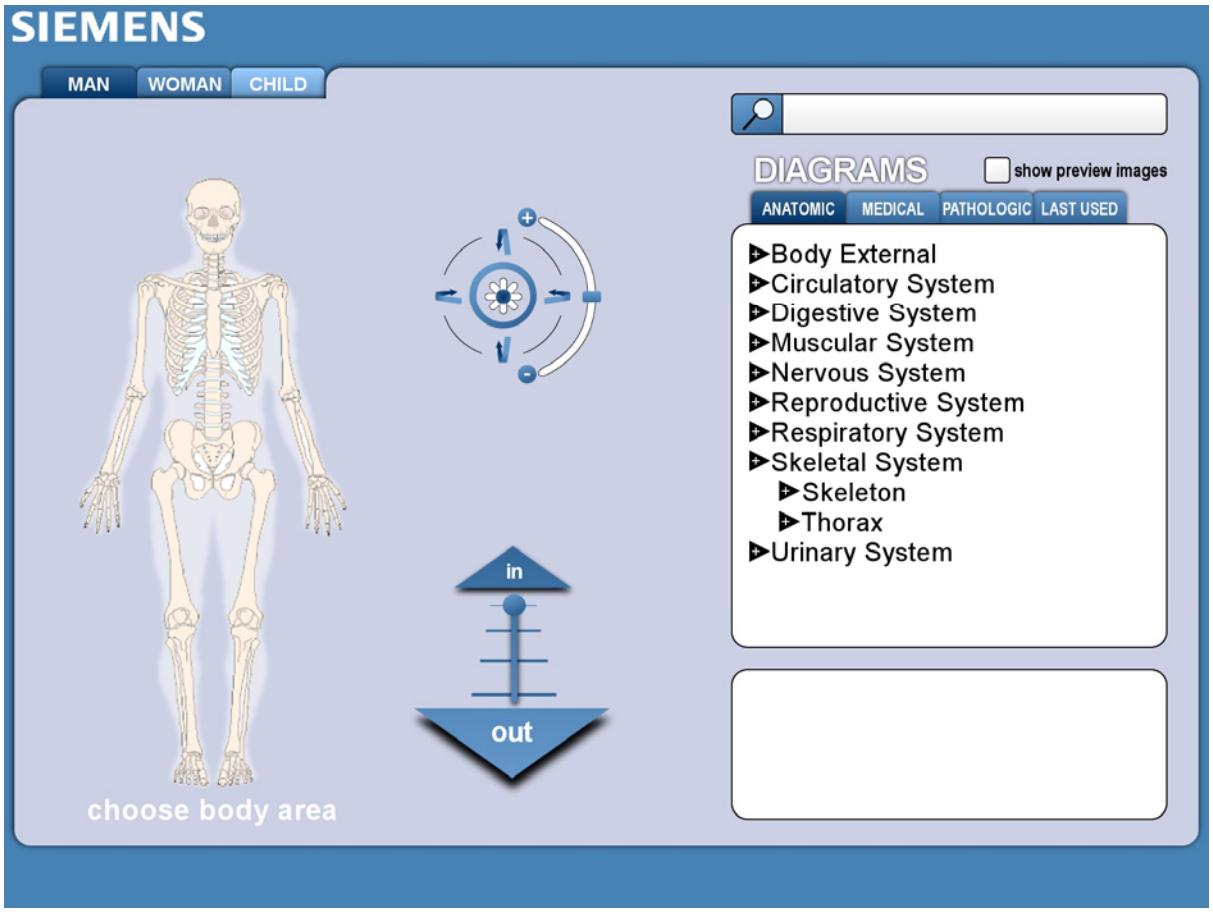

Figura 17 - Representação do nível interior máximo (esqueleto)

3.2.1.1.2 As Tabs MAN/WOMAN/CHILD

Figura 18 - *Tabs Man/Woman/Child*

As *tabs* presentes no canto superior esquerdo da interface (ver Figura 18), permitem alternar a representação gráfica do corpo humano entre Homem (*Man*), Mulher (*Woman*) e Criança (*Child*). Isto torna-se importante pois existem diferenças entre as estruturas físicas destes três tipos. Ao ser escolhido um deles, são listados apenas os diagramas que lhe dizem respeito.

Esta pode também ser considerada uma filtragem por categorias, neste caso a categoria “género e idade”.

3.2.1.1.3 A Roda de Controlo

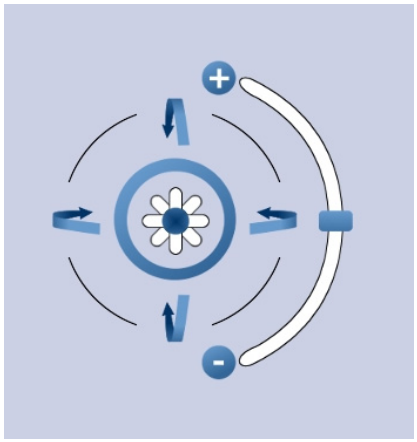

Figura 19 - Roda de controlo

A roda de controlo, que pode ser visualizada em figuras anteriores (por exemplo, na Figura 17), localizada ao centro por cima do *slider in/out*, ou então na Figura 19, permite controlar a imagem de representação do corpo humano realizando operações de *zoom in/out*, rotação horizontal e vertical e movimentação ao longo da imagem.

O *zoom* é realizado através do *slider* curvo presente no lado direito da roda. Ao movimentar o *slider* no sentido do botão "+", ou clicando no próprio botão, a imagem do corpo será aproximada/aumentada, ou seja, efectua-se o chamado *zoom in*. No sentido oposto, ao movimentar o *slider* no sentido do botão "-", ou clicando no botão, a imagem é afastada/diminuida (*zoom out*).

As rotações são feitas clicando nas quatro setas curvas existentes à volta da roda central. As duas setas horizontais permitem rodar o corpo horizontalmente, seja pela esquerda ou direita, apresentando depois como resultado, uma representação lateral ou traseira do corpo. As duas setas verticais permitem rodar o corpo verticalmente, resultando numa vista do topo (cimo da cabeça) ou do fundo (parte de baixo dos pés) do corpo.

O controlo central, permite movimentar o ponto de observação ao longo da imagem do corpo. Este é um controlo ao estilo de um "*joystick*", ou seja, "agarrando" no botão central e arrastando-o em qualquer uma das direcções (esquerda, direita, cima, baixo ou diagonais), o movimento será feito no mesmo sentido. É particularmente útil quando só se consegue visualizar parte da figura do corpo (por exemplo, a cabeça), devido ao *zoom in*, possibilitando assim seleccionar outra zona de visualização.

3.2.1.2 Filtragem por temas/categorias

No lado direito do ecrã, por cima da lista de diagramas, foram colocados *tabs* com diferentes categorias que o utilizador pode seleccionar: "Anatomic", "Medical" e "Pathologic" (ver Figura 20). Escolhendo uma delas, a lista de diagramas é categorizada em grupos diferentes.

Figura 20 - *Tabs* de categorias

A categoria “Anatomic” permite filtrar os diagramas da lista de acordo com o sistema do corpo humano escolhido: sistema circulatório, sistema respiratório, etc. A categoria “Medical” permite filtrar a lista de diagramas de acordo com a área médica escolhida: Cardiologia, Neurologia, Reumatologia, etc. Em ambas as categorias, é possível, num segundo nível, filtrar por órgãos.

A categoria “Pathologic” permite filtrar a lista de diagramas de acordo com patologias.

Nas figuras anteriores foram apresentados exemplos em que a categoria seleccionada era “Anatomic”. A Figura 21 ilustra o resultado da selecção da categoria “Medical”.

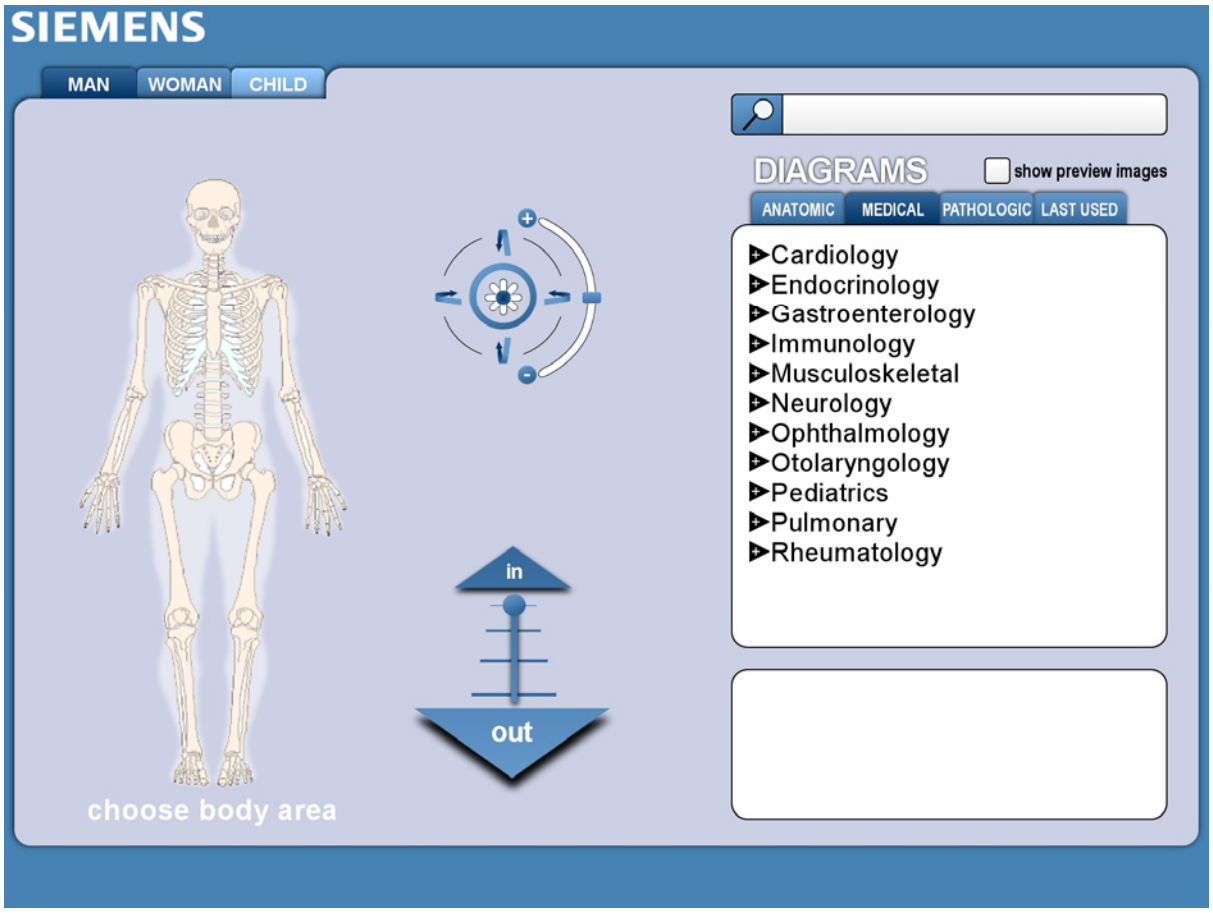

Figura 21 - Escolhida a categoria “Medical” são apresentadas as suas sub-categorias

3.2.1.3 Filtragem pelos últimos visitados

Esta filtragem tem a finalidade de permitir ao médico um acesso rápido aos últimos diagramas que ele usou. Assim, na *tab* “Last Used” (ver Figura 20) são listados os últimos 10 diagramas utilizados por um determinado médico.

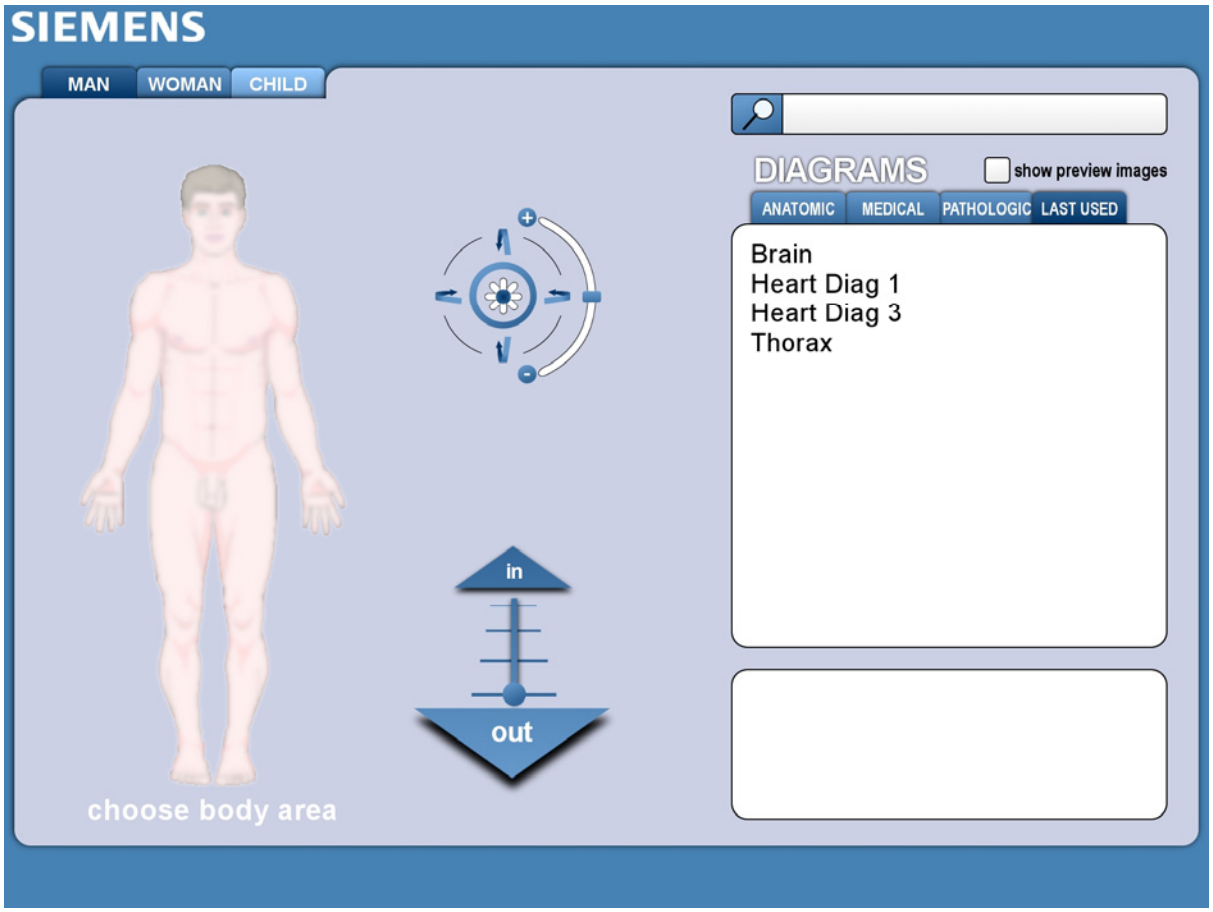

Figura 22 - Resultado (à direita) dos últimos 10 diagramas visitados

3.2.1.4 Filtragem por pesquisa por diagrama

A filtragem por pesquisa por diagrama é efectuada usando o campo de pesquisa presente no canto superior direito da interface, ilustrado na Figura 23.

Figura 23 - Campo de pesquisa

Escrevendo o nome completo ou parcial de um diagrama (ou categoria), a lista de diagramas será filtrada de forma a apresentar apenas os diagramas que têm essa designação.

Por exemplo, escrevendo “heart” no campo de pesquisa, o resultado será o que se apresenta na Figura 24.

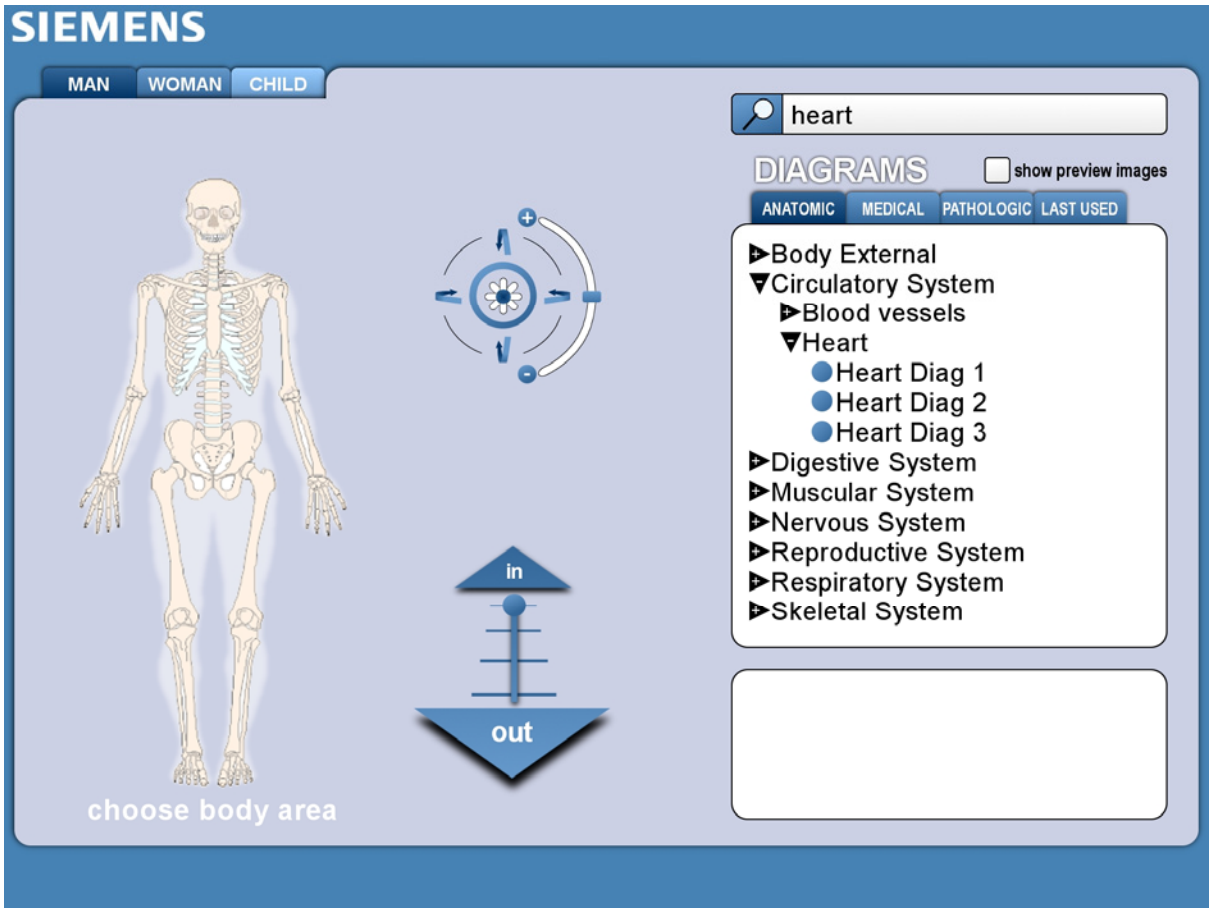

Figura 24 - Resultado, à direita, da pesquisa pela palavra “heart”

3.2.2 Tipos de Listagem

A apresentação da lista de diagramas a escolher (situada à direita), poderá ser feita de diferentes formas. Como pode ser visto nas imagens apresentadas até ao momento, o tipo de lista representado mais frequentemente é o do tipo árvore de categorias. No entanto, há outros tipos de listagem possíveis. Os diferentes tipos de listagem propostos são:

- Listagem textual
- Listagem em árvore
- Listagem com pré-visualização

3.2.2.1 Listagem textual

Neste tipo, a lista de diagramas é apresentada apenas como uma lista de nomes dos diagramas que estão de acordo com os filtros aplicados.

A Figura 25 apresenta como exemplo a lista dos “Last Used”.

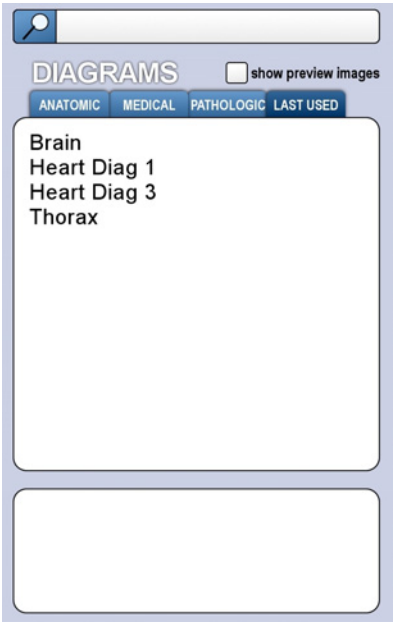

Figura 25 - Lista textual dos diagramas respeitantes ao *tab* "Last Used"

3.2.2.2 Listagem em árvore

A listagem em árvore é feita por agrupamento dos diagramas em categorias, sendo posteriormente possível "abrir" as categorias respectivas, descendo assim um nível na árvore.

A Figura 26 representa uma sequência que demonstra a acção de percorrer a árvore desde a categoria principal (neste caso, o "Circulatory System") até ao nível mais baixo da árvore, os diagramas (neste caso, os relativos ao "Heart").

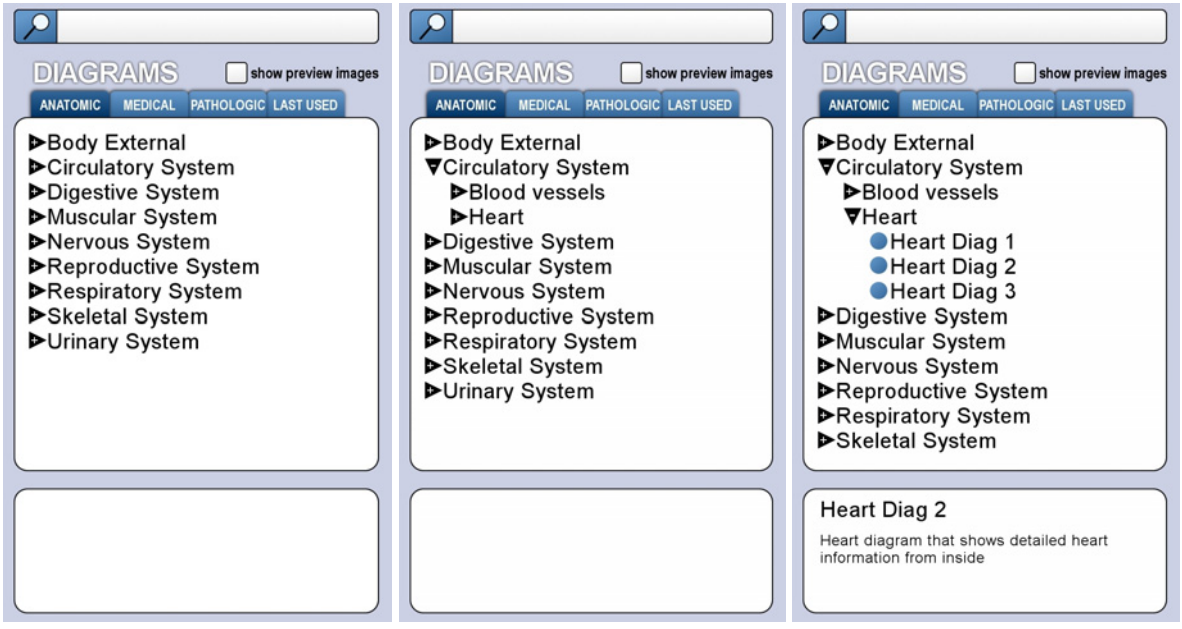

Figura 26 - Esquerda: categorias fechadas - Centro: "Circulatory System" aberto - Direita: "Heart" aberto, diagramas disponíveis para escolha

Notar que na imagem da direita aparece também representada em baixo, a descrição do diagrama "Heart Diag 2", que aparece quando este diagrama é seleccionado na lista.

3.2.2.3 Listagem com pré-visualização

Esta listagem efectua-se através da apresentação de imagens de pré-visualização do diagrama a ser aberto, após escolha, na aplicação Diagrammer. Assim, a lista apresentada é composta por pequenas imagens dos diagramas acompanhadas do seu nome na parte inferior.

Na Figura 27 mostra-se um exemplo deste tipo de listagem, que é activado a partir da *checkbox* "show preview images".

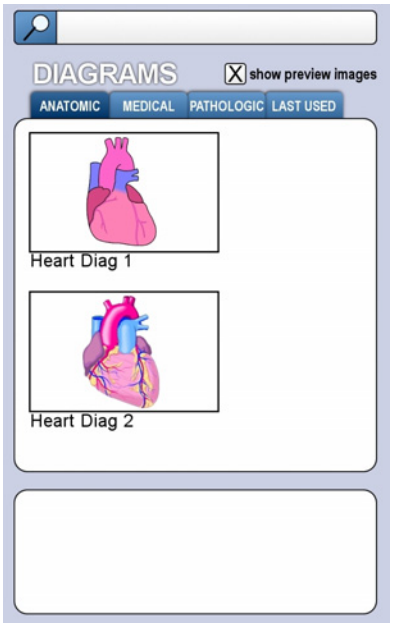

Figura 27 - Lista de diagramas com pré-visualização

3.3 Arquitectura

3.3.1 Descrição dos Componentes

Aqui são descritos os componentes da aplicação: a GUI (interface gráfica), a camada lógica e a base de dados. Na Figura 28 são ilustradas as ligações entre eles.

3.3.1.1 GUI – Graphic User Interface

Interface da aplicação, descrita no XAML do Silverlight. O XAML é um dialecto XML que especifica todos os elementos gráficos de uma aplicação Silverlight, desde os controlos mais comuns das aplicações *Web*, como, por exemplo, botões e caixas de texto, até todo o tipo de desenhos vectoriais, inclusive, os controlos mencionados são vectoriais.

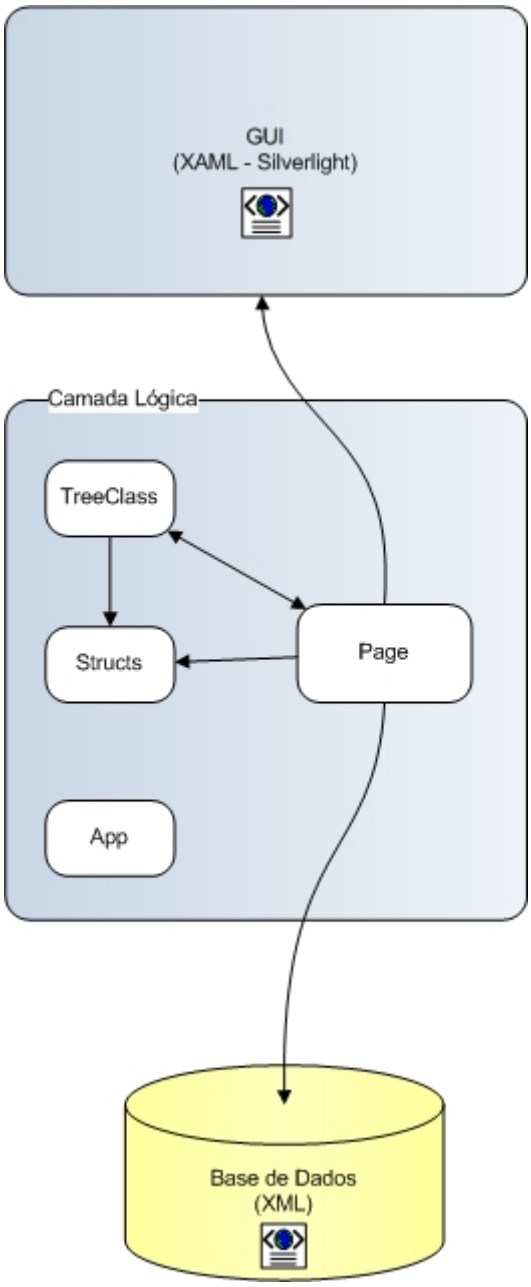

Figura 28 - Componentes

3.3.1.2 Camada Lógica

A camada lógica da aplicação faz o processamento de todos os eventos despoletados na interface e de todas as acções realizadas pelo utilizador, enviando o resultado para a GUI. Esta é a camada também responsável por ler, da base de dados, todos os dados necessários à execução da aplicação.

3.3.1.3 Base de Dados

A base de dados da aplicação contém toda a informação requerida, de acordo com o Diagrama de Classes definido abaixo.

3.3.2 Modelação da Base de Dados

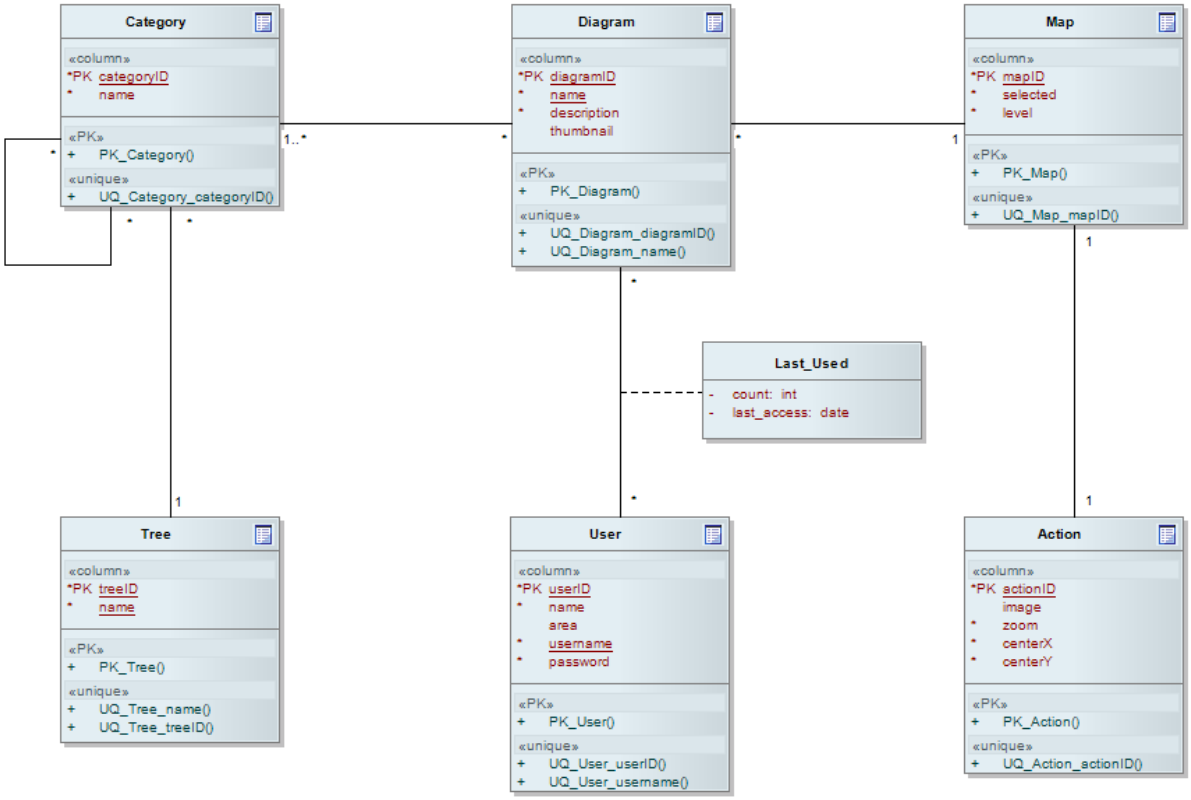

Figura 29 - Diagrama de classes da base de dados

3.3.2.1 Diagram

A tabela Diagram é a tabela que define a informação que é necessário guardar sobre cada diagrama médico presente no Siemens Diagrammer.

- diagramID – Identificador do diagrama na base de dados;
- name – Nome do diagrama;
- description – Descrição do diagrama, podendo indicar para que tipo de anotações o diagrama se adequa;
- thumbnail – Localização da imagem (URL) de pré-visualização do diagrama.

3.3.2.2 Category

Esta tabela define as categorias a que um diagrama pode pertencer. Cada categoria pode ter várias sub-categorias e pertence apenas a uma árvore de diagramas.

- categoryID – Identificador da categoria;
- name – Nome da categoria.

3.3.2.3 Tree

A tabela Tree define as árvores de diagramas. Cada árvore é dividida em várias categorias.

- treeID – Identificador da árvore de diagramas;

- name – Nome da árvore, correspondente com as *tabs* de categorias propostas para a aplicação (ver Figura 20).

3.3.2.4 Map

A tabela Map define os mapas de selecção da imagem representativa do corpo humano. Cada um dos mapas terá uma acção associada.

- mapID – Identificador do mapa de selecção;
- selected – Nome da figura (*path*) definida na GUI em XAML, que define uma zona da imagem do corpo humano, passível de ser seleccionada;
- level – Nome da camada do corpo (exterior, músculos, órgãos e esqueleto) que está activo aquando de uma selecção.

3.3.2.5 Action

Esta tabela define a acção associada a cada mapa de selecção. Essa acção é o resultado que um clique no mapa de selecção respectivo origina na imagem representativa do corpo humano, centrando a imagem num novo ponto, definindo um novo nível de *zoom* e a imagem a apresentar.

- actionID – Identificador da acção;
- image – Imagem a apresentar, que pode ser a mesma imagem ou uma nova imagem, como por exemplo, uma camada diferente (exterior, músculos, órgãos e esqueleto);
- zoom – Nível de ampliação (*zoom*) da imagem;
- centerX – Coordenada X do ponto em que a imagem será centrada;
- centerY – Coordenada Y do ponto em que a imagem será centrada.

As coordenadas são definidas, relativamente à imagem humana, na gama de valores de 0 a 1, em que, por exemplo, o ponto x,y = 0.5,0.5 refere-se ao centro da imagem e o ponto x,y = 0.5,0 refere-se ao topo da cabeça na imagem. Desta forma, a imagem é sempre centrada no mesmo local independentemente da resolução do ecrã e do tamanho da imagem.

3.3.2.6 User

A tabela User é a tabela que define a informação relativa a um utilizador da aplicação, desde os dados de autenticação até aos dados demográficos necessários à aplicação.

- userID – Identificador do utilizador na base de dados;
- name – Nome do utilizador;
- area – Área médica do utilizador;
- username – Nome de autenticação do utilizador na aplicação;
- password – Palavra-chave para autenticação do utilizador.

3.3.2.7 Tabela de ligação Last_Used

A tabela Last_Used é uma tabela de ligação entre a tabela User e a tabela Diagram. Esta tabela define a informação para registo dos diagramas usados por cada utilizador.

- count – contador do número de vezes que um diagrama foi acedido por um utilizador;
- last_access – data do último acesso realizado a um diagrama por um utilizador.

3.3.2.8 Ligação Tree-Category

A ligação entre a tabela Tree e a tabela Category é necessária para indicar que categorias pertencem a uma árvore. Uma categoria só pode pertencer a uma árvore.

3.3.2.9 Ligação Category-Category

A ligação entre a tabela Category e ela mesma permite definir quais as subcategorias de cada categoria, se existirem, criando assim a árvore com vários níveis de categorias.

3.3.2.10 Ligação Category-Diagram

Esta ligação define que diagramas pertencem a cada categoria. O mesmo diagrama pode estar em várias categorias diferentes da mesma árvore.

3.3.2.11 Ligação Diagram-Map

A ligação entre a tabela Diagram e a tabela Map é necessária para associar cada diagrama aos mapas de selecção da imagem representativa do corpo humano. Cada diagrama só pode estar associado a um mapa, para que, quando um diagrama é seleccionado na árvore de diagramas, seja mostrada, na imagem do corpo humano, a zona do corpo (definida pelo mapa) a que esse diagrama está associado.

Esta ligação Diagram-Map também funciona de forma inversa, possibilitando que um clique numa das zonas da imagem do corpo humano, respeitante a um mapa de selecção, origine a abertura da árvore de diagramas nas categorias que contêm os diagramas associados ao mapa seleccionado.

3.3.2.12 Ligação Map-Action

Esta é uma ligação de um para um entre a tabela Map e a tabela Action. Esta ligação é necessária para atribuir uma, e uma só, acção a cada mapa de selecção da imagem representativa do corpo humano.

3.3.3 Modelo de Classes

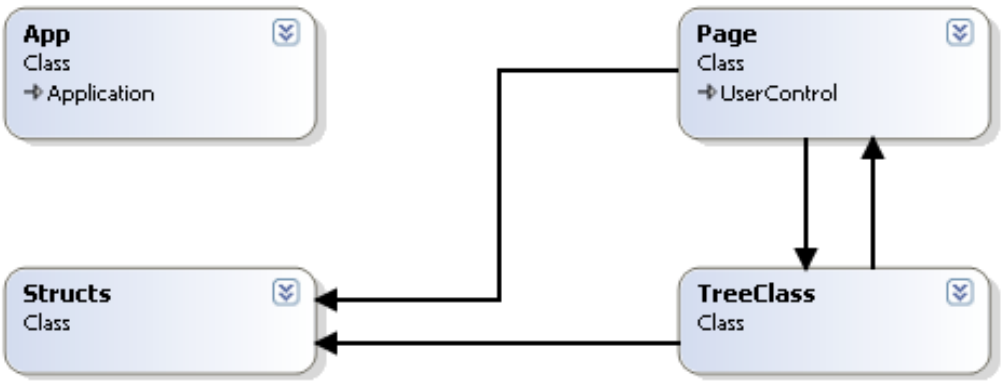

Figura 30 - Modelo de classes

3.3.3.1 Page

A classe Page é a classe central da aplicação e que está associada ao XAML que define o aspecto gráfico da aplicação. Esta classe lança a GUI e gere toda a interacção com um utilizador. É também esta classe que faz a leitura da base de dados.

3.3.3.2 TreeClass

Esta classe possui todos os métodos e funções para a criação e gestão da árvore de diagramas activa no Outliner.

3.3.3.3 Structs

A classe Structs é uma classe auxiliar das mencionadas anteriormente, pois tem definidas todas as estruturas necessárias para o manuseamento da informação lida da base de dados.

3.3.3.4 App

A classe App é a responsável por gerir os eventos de início e fim da execução da aplicação.

4 Implementação

4.1 Implementação da Base de Dados

A Base de Dados do Outliner foi implementada em ficheiros XML que guardam toda a informação, de acordo com a modelação descrita no capítulo anterior (ver Figura 29).

Para que estes XML cumprissem as restrições da modelação, foram definidos DTDs para cada um deles. Estes DTDs definem a estrutura dos dados guardados num ficheiro XML.

Descreve-se de seguida os ficheiros XML e respectivos DTDs que constituem a base de dados da aplicação.

4.1.1 XML_Diagrams.xml

Este ficheiro guarda os dados relativos aos diagramas existentes e às estruturas em árvore a que eles pertencem.

O XML facilita a implementação da estrutura em árvore, visto que um ficheiro XML descreve também os seus elementos de uma forma hierárquica (em árvore); um elemento-pai contém elementos-filho que também podem conter filhos e assim por diante. Foi desta forma que foram descritas as árvores de diagramas com as respectivas categorias e subcategorias.

Cada árvore no XML é definida pelo elemento *tree*, que corresponde à tabela Tree da modelação da base de dados e conterá um ou mais elementos *category* (as categorias), correspondentes à tabela Category. Além destes, foi necessário definir um elemento extra, o elemento *leaf*, que define uma folha de uma árvore. Cada folha representa um diagrama na árvore, permitindo assim saber a que categorias pertence cada diagrama. Visto que cada diagrama pode estar presente em várias categorias em cada árvore, este elemento-folha apenas contém, como atributo, uma referência para o identificador (ID) de um elemento diagrama, que por sua vez terá descrita toda a informação relativa a um diagrama.

Exemplo de uma estrutura em árvore definida no XML:

```
<tree name="Anatomic">
      <category name="Circulatory System">
            <category name="Heart">
                  <leaf idref="diag1"/>
                  <leaf idref="diag2"/>
            </category>
      </category>
      <category name="Respiratory System">
            <category name="Lungs">
```

```
                <leaf idref="diag3"/>
                <leaf idref="diag2"/>
            </category>
        </category>
</tree>
```

O código do exemplo mostrado acima corresponde à estrutura em árvore ilustrada na Figura 31.

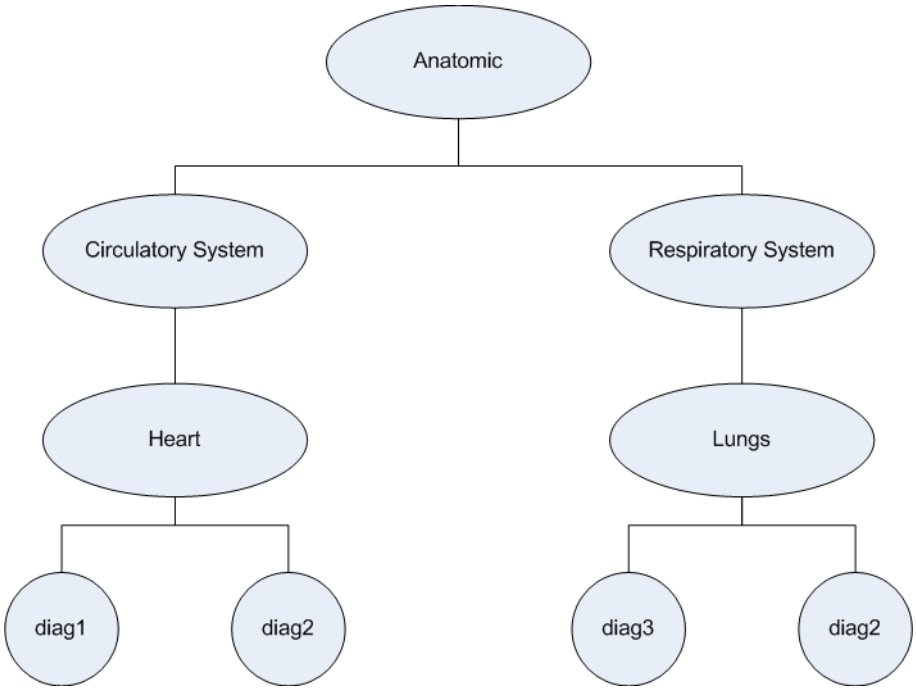

Figura 31 - Exemplo de uma árvore de diagramas

Os elementos diagrama, designados por *diagram* no XML, encontram-se descritos neste mesmo XML após os elementos árvore (*tree*). São os elementos diagrama que contêm a informação relativa a cada diagrama existente no Diagrammer. Essa informação engloba todos os campos definidos na tabela Diagram da base de dados e também um atributo *map* que identifica o mapa de selecção a que o diagrama está ligado. Assim, os elementos diagrama encontram-se descritos pelos atributos *id*, *name*, *map*, *thumbnail* e pelo sub-elemento *description*. A descrição (*description*) do diagrama está definida no XML como um sub-elemento e não como um atributo do elemento diagrama devido a que esta, ao contrário dos outros campos, não será apenas uma palavra ou um conjunto pequeno de palavras, mas antes, um texto relativamente extenso, sendo assim mais adequado defini-la em XML como um sub-elemento.

Exemplo de elementos diagrama definidos no XML:

```
<diagram id="diag1" name="Heart Diag1" map="trunk2"
thumbnail="Diagrams/heart_diag1.PNG">
      <description>Este é um diagrama do interior do coração</description>
</diagram>
<diagram id="diag2" name="Heart and Lungs" map="trunk2"
thumbnail="Diagrams/heart_&_lungs.PNG">
      <description>Diagrama do coração e pulmões</description>
</diagram>
```

```
<diagram id="diag3" name="Lungs" map="trunk1"
thumbnail="Diagrams/lungs.PNG">
      <description>Este diagrama serve para anotações de problemas nos
      pulmões</description>
</diagram>
```

Neste mesmo XML ficam também guardados os elementos referentes à tabela de ligação Last_Used (ver secção 3.3.2). Estes elementos foram aqui implementados de forma a que cada elemento *last_used* pertencesse a um utilizador e tivesse uma lista de sub-elementos referentes a cada diagrama usado por esse utilizador. Assim, os elementos *last_used* possuem um atributo *user* que referencia o identificador de um utilizador do sistema. Dentro de cada elemento *last_used* poderá aparecer, ou não, uma lista de elementos *diag* em que cada um deles referencia um diagrama pelo seu identificador, possuindo para isso o atributo *idref*. Cada elemento *diag* possui também os atributos *count* e *last_access*, que indicam o número de acessos efectuados àquele diagrama e qual a data e hora do último acesso, respectivamente.

Esta foi a maneira escolhida para implementar a tabela Last_Used, ou seja, guardar para cada utilizador uma lista de registos dos diagramas que ele utilizou recentemente, e não o contrário, isto é, para cada diagrama ter o registo dos utilizadores que lhe acederam. Isto deve-se ao facto de, a cada utilização, só estar autenticado na aplicação um utilizador, pelo que é bem mais rápido encontrar qual o elemento *last_used* daquele utilizador, e só aí ler todos os registos aí contidos como sub-elementos, do que, pelo contrário, procurar nos registos de todos os diagramas se aquele utilizador acedeu a cada um deles.

Exemplo do elemento *last_used* do utilizador cujo identificador é “u1”:

```
<last_used user="u1">
      <diag idref="diag5" count="1" last_access="20-05-2008 15:59:00"/>
      <diag idref="diag2" count="4" last_access="16-05-2008 23:42:00"/>
      <diag idref="diag7" count="25" last_access="16-05-2008 23:42:25"/>
</last_used>
```

De cada vez que o utilizador do exemplo abrir um diagrama no Diagrammer, ocorrerá uma das seguintes situações:

1. Caso o diagrama ainda não conste da lista de diagramas utilizados por este utilizador, um novo registo será adicionado à lista como um novo elemento *diag*, referente a esse diagrama, em que o contador de número de acessos, o *count*, será colocado a 1 e o *last_access* ficará com o registo da data e hora do momento em que o acesso foi realizado.
2. Caso o diagrama já possua o seu registo na lista de diagramas utilizados, o *count* será incrementado em uma unidade e o *last_access* será actualizado para a data e hora da realização deste novo acesso.

Todas estas regras e restrições, mencionadas acima e definidas na modelação da base de dados, são descritas no XML através do seu DTD apresentado a seguir:

```
<!DOCTYPE Diagrams [
  <!ELEMENT Diagrams (tree+, diagram+, last_used*)>
  <!ELEMENT tree (category+)>
  <!ELEMENT diagram (description)>
  <!ELEMENT last_used (diag*)>
```

```
  <!ELEMENT category (category*, leaf*)>
  <!ELEMENT leaf EMPTY>
  <!ELEMENT description (#PCDATA)>
  <!ELEMENT diag EMPTY>

  <!ATTLIST tree name (Anatomic|Medical|Pathologic) #REQUIRED>

  <!ATTLIST category name CDATA #REQUIRED>

  <!ATTLIST leaf idref IDREF #REQUIRED>

  <!ATTLIST diagram id ID #REQUIRED
                    name CDATA #REQUIRED
                    map IDREF #REQUIRED
                    thumbnail CDATA #IMPLIED
  >

  <!ATTLIST last_used user IDREF #REQUIRED>

  <!ATTLIST diag idref IDREF #REQUIRED
                 count CDATA #REQUIRED
                 last_access CDATA #REQUIRED
  >
]>
```

4.1.2 XML_MapsActions.xml

Este XML serve para guardar todos os mapas de selecção da imagem do corpo humano e as acções que cada um deles despoleta. Portanto, este é o XML que implementa as tabelas Map e Action, definidas na modelação da base de dados, e a respectiva ligação entre elas. Essa ligação, sendo de um para um, é aqui garantida pelo facto que cada elemento *map* contém como sub-elemento, um elemento *action*, tirando mais uma vez partido das características do XML.

Devido a esta forma de implementação, deixou de ser necessário usar o identificador de cada acção, o *actionID*, presente na tabela Action, visto que cada acção está contida num mapa e assim cada acção é referenciada pelo mapa a que pertence.

Exemplo de um elemento *map* e respectivo sub-elemento *action*:

```
<map id='trunk2'
      selected='ManTrunkSelect'
      level='ManSkeleton'>
      <action
            image='ManBodyExternal'
            zoom='1,5'
            centerX='0,5'
            centerY='0,4'
      >
      </action>
```

```
</map>
```

O DTD que define neste XML todas as regras e restrições é o seguinte:

```
<!DOCTYPE Maps [
  <!ELEMENT Maps (map+)>
  <!ELEMENT map (action)>
  <!ELEMENT action (#PCDATA)>

  <!ATTLIST map id ID #REQUIRED>
  <!ATTLIST map selected CDATA #REQUIRED>
  <!ATTLIST map level CDATA #REQUIRED>

  <!ATTLIST action image CDATA #IMPLIED>
  <!ATTLIST action zoom CDATA '1'>
  <!ATTLIST action centerX CDATA #REQUIRED>
  <!ATTLIST action centerY CDATA #REQUIRED>
]>
```

De notar que o nível de *zoom* original da imagem é 1, pelo que níveis de *zoom* diferentes funcionam como factores multiplicadores do tamanho original. Por exemplo, um nível de *zoom* igual a 2 implicará que o tamanho actual da imagem do corpo humano seja o dobro do seu tamanho original.

4.1.3 XML_Users.xml

Este XML guarda a informação relativa aos utilizadores com acesso à aplicação, isto é, uma lista de elementos *user* que correspondem à tabela User (ver secção 3.3.2).

Os campos da tabela User foram aqui implementados de forma a separar os dados demográficos do utilizador dos seus dados de autenticação no sistema. Desta forma, o elemento *user* apresenta os atributos *name* e *area*, respeitantes aos dados demográficos definidos acima como nome e área médica respectivamente, além do seu identificador, o *id*. Os dados para autenticação foram definidos como sub-elementos do elemento *user*, neste caso, os sub-elementos *username* e *password*.

Esta divisão também foi feita porque os dados demográficos não passam actualmente de elementos meramente informativos, a serem mostrados na aplicação, enquanto que o *username* e *password* são os dados realmente importantes para a aplicação, pois são estes que identificam um utilizador no sistema, permitindo assim o seu acesso e o carregamento da informação associada a este utilizador, como é o caso dos registos de diagramas por ele utilizados.

Exemplo de um utilizador guardado neste XML:

```
<user id="u1" name="Dr Jose Cardoso" area="Cardiology">
      <username>user1</username>
      <password>user1</password>
</user>
```

O DTD que define todos os elementos e respectivas regras e restrições deste XML é o apresentado seguidamente:

```
<!DOCTYPE Users [
  <!ELEMENT Users (user+)>
```

```
  <!ELEMENT user (username, password)>
  <!ELEMENT username (#PCDATA)>
  <!ELEMENT password (#PCDATA)>

  <!ATTLIST user id ID #REQUIRED
                 name CDATA #REQUIRED
                 area CDATA #REQUIRED
  >
]>
```

4.2 Implementação das Classes

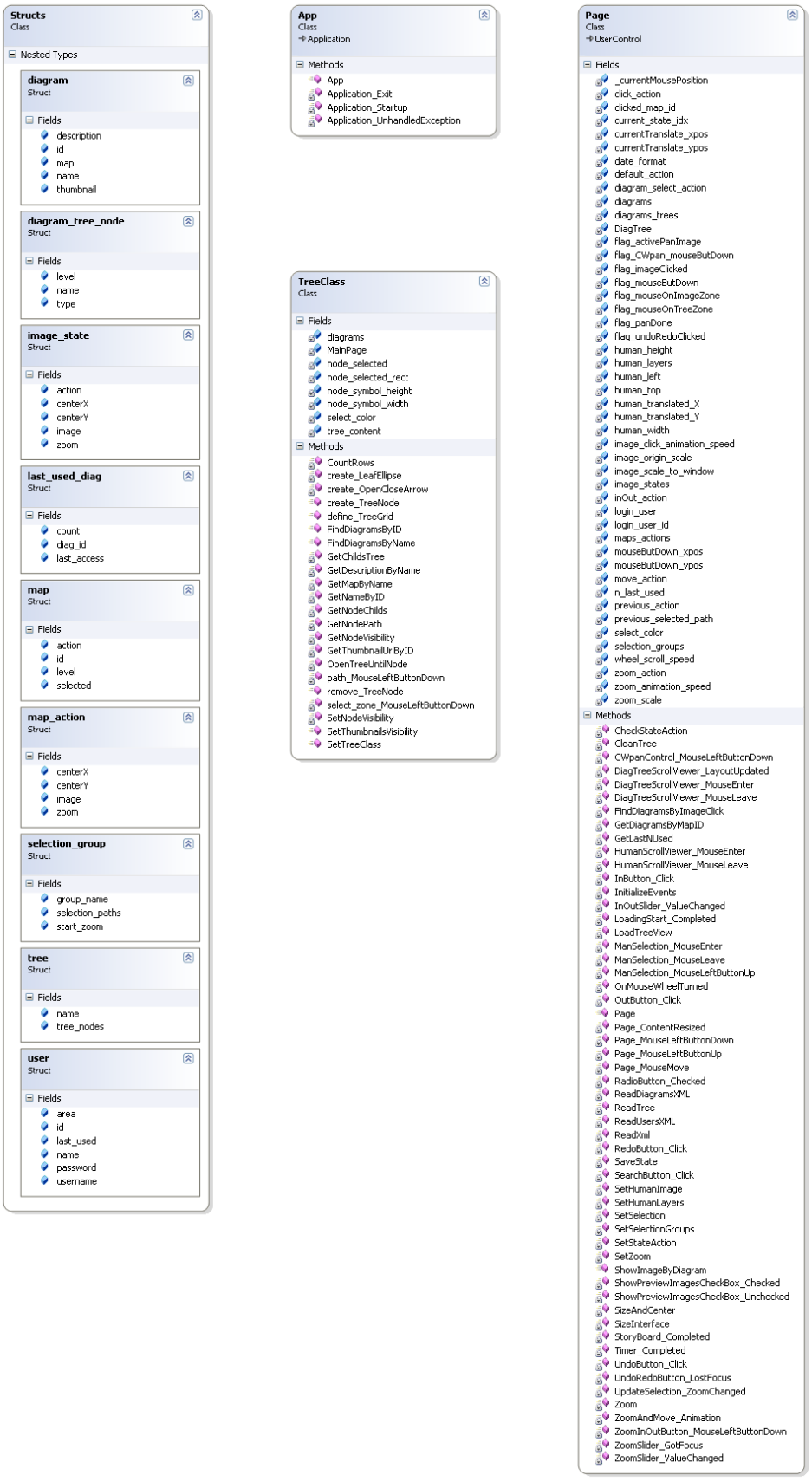

Figura 32 - Classes e métodos da aplicação

4.2.1 Page

A classe *Page* é a classe da página principal da aplicação em Silverlight e está associada com a interface definida no Page.xaml. Esta é a classe responsável por inicializar todos os componentes da aplicação e executá-la.

A interface é chamada nesta classe, inicializando todos os seus controlos com os valores necessários, incluindo os valores referentes aos seus tamanhos que são aqui ajustados ao tamanho actual da janela do utilizador.

Após a inicialização da interface, esta classe lê todos os dados da base de dados do Outliner, que estão guardados em ficheiros XML, e carrega-os para a memória utilizando as estruturas definidas na classe *Structs*. Esses dados contêm toda a informação sobre os utilizadores, os diagramas existentes e como estes estão organizados em categorias numa estrutura em árvore. Esta árvore é carregada e visualizada na aplicação a partir da *Tree View* localizada no lado direito da interface. Além disto, os dados incluem também a informação sobre os mapas de selecção da imagem representativa do corpo humano e a acção que cada um deles despoleta.

Toda esta informação fica assim disponível durante a execução da aplicação permitindo a selecção de um diagrama clínico, a ser aberto no Siemens Diagrammer, como o seu objectivo final.

Descreve-se em seguida as principais funcionalidades implementadas nesta classe.

4.2.1.1 Definir as dimensões da imagem humana e centrá-la

Esta funcionalidade permite redimensionar a imagem do corpo humano de forma a ficar ajustada ao tamanho e resolução da janela do *browser* do utilizador e centra-a na área da interface destinada a conter a imagem. Essa área foi definida por um controlo do Silverlight designado de *ScrollViewer*.

A implementação desta funcionalidade seguiu os seguintes passos:

1. Recebe o tamanho actual da janela do *browser* em pixeis;
2. Define o tamanho do *ScrollViewer*, ou seja, da área que contém a imagem, de acordo com o tamanho da janela;
3. Calcula o tamanho da imagem humana em que:
    - A largura é igual à distância entre as duas mãos, visto que estas são os extremos horizontais da imagem. Esta distância é igual à diferença entre a posição mais à direita da mão direita e a posição mais à esquerda da mão esquerda. Assim, o cálculo da largura implementado foi o seguinte:

```
human_width = Canvas.GetLeft(ManHandRight) - Canvas.GetLeft(ManHandLeft)
```

De notar que a função Canvas.GetLeft devolve a posição horizontal mais à esquerda de uma forma desenhada na interface da aplicação. No entanto, no caso da forma da mão direita, que foi desenhada como sendo uma inversão horizontal da mão esquerda, o Canvas.GetLeft devolve a posição horizontal mais à direita devido a essa inversão, visto que "a esquerda passou a ser a direita".

    - A altura é igual à distância entre a cabeça e os pés, pois são estes os extremos verticais. Este cálculo é feito através da diferença entre a posição

do ponto mais abaixo dos pés e o ponto mais acima da cabeça (no topo do cabelo):

```
human_height = Canvas.GetTop(ManFootLeft) + ManFootLeft.Height -
Canvas.GetTop(ManHair)
```

A função Canvas.GetTop devolve a posição vertical, em pixeis, do ponto mais acima de uma forma, sendo que o topo do *Canvas* (elemento do Silverlight que define uma área na aplicação) é a posição 0 e o seu fundo será a posição máxima. Por isso, para se conseguir a posição do ponto mais abaixo da forma que define um pé foi necessário adicionar a altura dessa forma (ManFootLeft.Height) à sua posição vertical (Canvas.GetTop(ManFootLeft)).

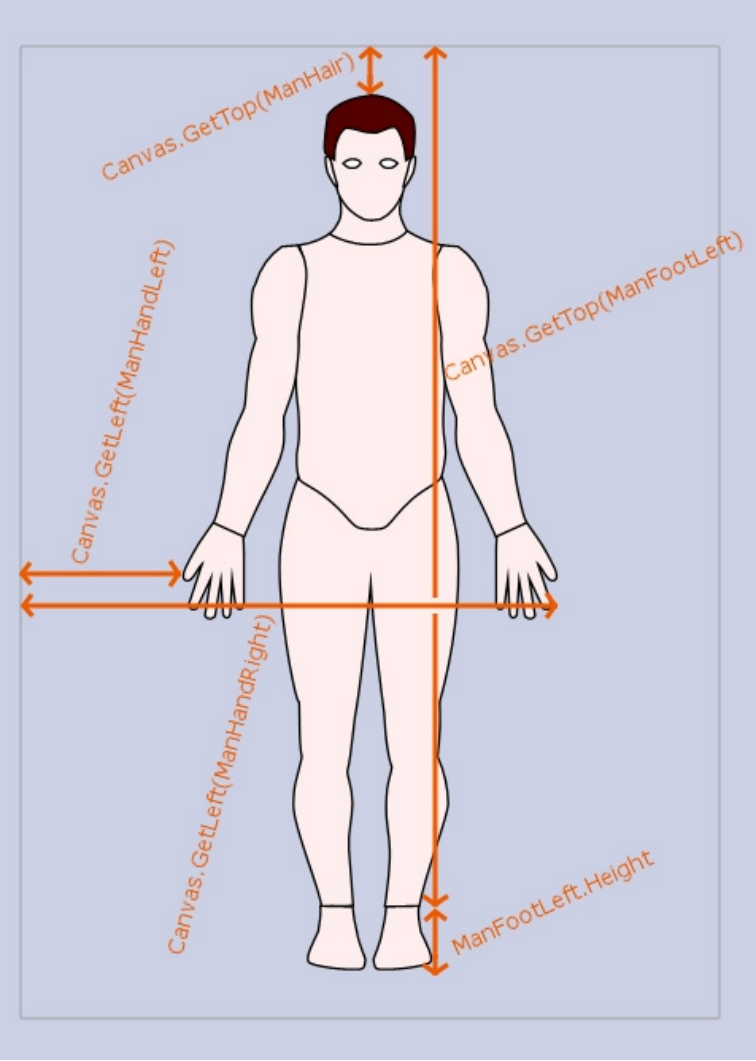

Figura 33 - Imagem humana com as distâncias que permitem calcular as suas dimensões

4. Calcula as margens que a imagem deve ter em relação aos limites do *ScrollViewer*:
   - Cada margem horizontal é igual a metade da diferença entre a largura do *ScrollViewer* e a largura da imagem;
   - Cada margem vertical é igual a metade da diferença entre a altura do *ScrollViewer* e a altura da imagem;

5. Calcula quanto é que tem que mover a imagem humana na horizontal e vertical fazendo a diferença entre a margem horizontal e a posição horizontal actual da imagem, e a diferença entre a margem vertical e a posição vertical actual da imagem, respectivamente;
6. Define a nova posição da imagem de acordo com os cálculos efectuados no ponto anterior;
7. Calcula o escalonamento a aplicar às dimensões da imagem para se adaptar ao tamanho da janela actual de acordo com a diferença relativa entre as dimensões do *ScrollViewer* e da imagem;
8. Define os valores de escalonamento da imagem (*ScaleX* e *ScaleY*) de acordo com o valor calculado anteriormente.

4.2.1.2 Ler os ficheiros XML que definem a base de dados

Para a leitura dos XML que constituem a base de dados da aplicação foi necessário criar para cada um dos XML uma instância da classe XmlReader, pertencente à biblioteca System.XML. Esta classe permite percorrer todos os elementos de um XML uma só vez, de forma sequencial, não possibilitando a releitura. Daí a necessidade de, ao longo da leitura de cada um dos ficheiros, gravar os dados lidos para a memória em *arrays* de estruturas definidas de forma a corresponderem aos elementos lidos no XML.

Todos estes *arrays* são definidos de forma dinâmica, ou seja, não são criados com um tamanho fixo, mas sim, aumentados a cada elemento lido, de forma a não haver desperdício de memória alocada a eles. De cada vez que um novo elemento é lido com sucesso, é guardado num *array* temporário todo o conteúdo actual do *array* principal. Todo esse conteúdo é depois novamente copiado para o *array* principal, após este ter sido redefinido com um tamanho maior em uma unidade que anteriormente. Assim, o *array* passa a ter um item livre em que é guardado em seguida os dados do novo elemento lido.

A necessidade de usar um XmlReader advém da inexistência de outra classe que ofereça maior facilidade e controlo sobre a leitura de um XML, pois apesar de estas existirem na *framework* completa do .NET, não estão disponíveis no Silverlight visto que, como já mencionado, este possui apenas um subgrupo dessa *framework*.

Os XML a serem lidos foram adicionados como *Resources* (recursos) da aplicação para poderem ser recebidos como um *stream* aceite pelo XmlReader.

O XmlReader também não possui métodos de validação do XML segundo o seu DTD, pelo que foi necessário verificar ao longo do código para leitura de cada um dos XML se o conteúdo estava de acordo com o definido, guardando os erros se estes existissem. Em caso de existência de erros, os dados não são guardados e a leitura do XML é interrompida.

As verificações gerais são:

- Verifica se existem atributos, caso devam existir;
- Verifica se falta algum dos atributos definidos;
- Verifica se os atributos existentes são os definidos;
- Verifica se os elementos lidos são os definidos;
- Verifica a existência dos sub-elementos de um elemento.

Dado que cada XML contém uma estrutura específica, foi definida uma função para ler cada um deles. Menciona-se em seguida os pormenores de leitura de cada um dos XML.

4.2.1.2.1 Ler o XML_MapsActions.xml

Neste XML apenas dois tipos de elementos podem ser encontrados: o elemento *map* e o elemento *action*. A leitura deste XML foi implementada de forma a que seja lido sempre um elemento *map* antes de um elemento *action*, pois este último é um sub-elemento do primeiro, e sempre nesta ordem, ou seja, após lido um *map* dever ser lido um *action*, garantindo assim o definido na secção 4.1.2 da implementação da base de dados.

4.2.1.2.2 Ler o XML_Diagrams.xml

Neste caso, podem ser lidos três tipos de elementos principais, que são o elemento *tree*, o elemento *diagram* e o elemento *last_used*. Estes elementos são lidos sem qualquer dependência entre eles.

Elemento *tree*

Os elementos *tree*, sendo elementos que definem uma árvore, são lidos de forma recursiva percorrendo cada nível da árvore. Para cada nível é criada uma nova instância do XmlReader mas com o subgrupo do XML correspondente a um elemento desse nível com todos os seus descendentes. Isto é conseguido através da função ReadSubtree() do XmlReader.

Cada árvore fica guardada num *array* de nós, em que cada nó guarda o nome do nó, o seu tipo e o seu nível na árvore (ver secção 4.2.3.4).

A função que lê uma árvore (ou sub-árvore), designada ReadTree(), recebe como parâmetros o XmlReader devolvido por ReadSubTree() e o nível da árvore que está a ser lido. A função ReadTree() segue os seguintes passos:

1. Ignora o elemento-pai (*tree* ou *category*) que agrupa todos os elementos da árvore, pois o que se pretende é ler todos os seus nós;
2. Lê o próximo elemento da sub-árvore;
3. Se o elemento for uma categoria (elemento *category*):
    a. Guarda esse nó, como sendo do tipo *node*;
    b. Chama a função ReadTree() para ler a sub-árvore contida nesta categoria recebendo o *array* de nós dos seus descendentes. Os parâmetros passados serão o XmlReader da sub-árvore e o nível incrementado de uma unidade;
4. Se o elemento for uma folha (elemento *leaf*), ou seja, um diagrama:
    a. Guarda esse nó mas com o tipo *leaf*;
5. Agrupa os *arrays* lidos num só *array* em que os elementos lidos mais recentemente são adicionados no fim;
6. Após todos os elementos terem sido lidos, devolve o *array* com todos os nós guardados.

Para o exemplo da árvore representada na Figura 31, o processo de leitura descrito acima é ilustrado na Figura 34. O *array* criado no final do processo ilustrado e que corresponderá ao *array* dos nós da árvore de exemplo "Anatomic", pode ser visualizado na Figura 35.

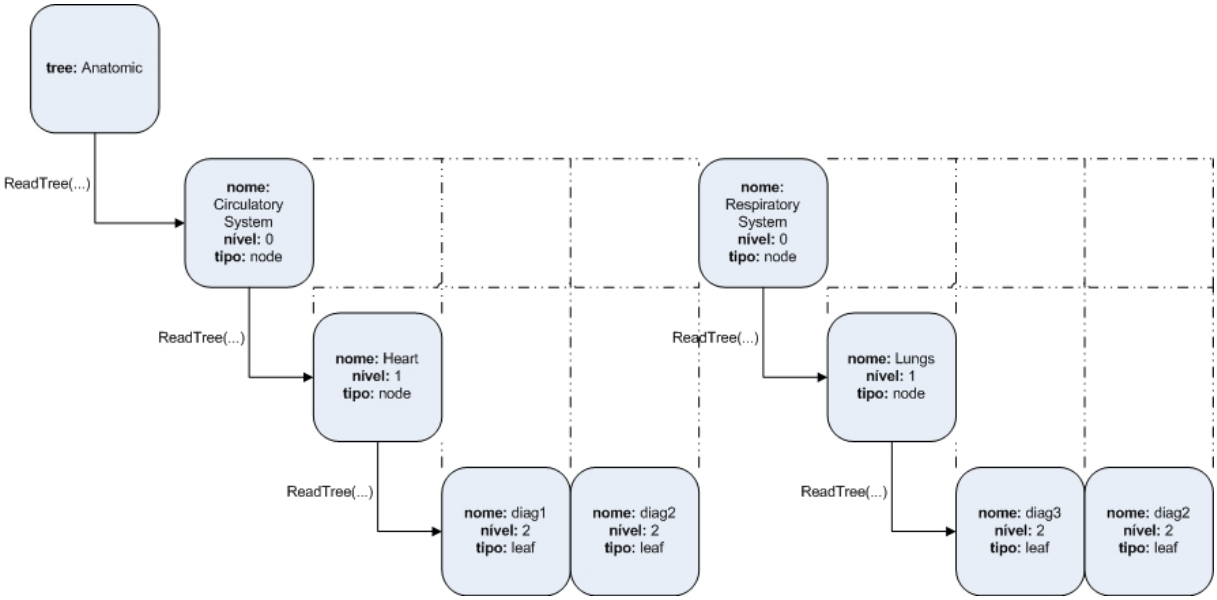

Figura 34 - Ilustração do processo de leitura da árvore exemplo Anatomic

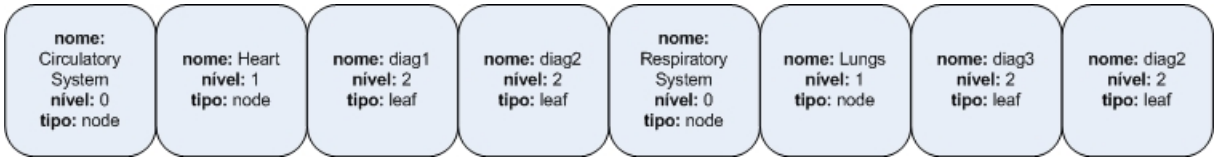

Figura 35 - *Array* de nós da árvore exemplo Anatomic

Elemento *diagram*

A leitura dos elementos *diagram* consiste em ler os seus atributos e o seu sub-elemento *description*. Assim, após a leitura dos atributos, lê-se o próximo elemento do XML que deve ser o elemento *description*, pois caso contrário estar-se-ia perante uma situação de erro. Este elemento tem a particularidade de ter os seus dados no seu conteúdo, ou seja, entre as suas *tags* (`<description>` e `</description>`), e não como atributos. No entanto, tanto como para a leitura dos atributos, o XmlReader possui funções para efectuar a leitura destes dados.

Elemento *last_used*

Em cada utilização da aplicação, só um elemento *last_used* é lido. Esse elemento é o respeitante ao utilizador autenticado no sistema. Por isso, ao ser efectuada a leitura dos elementos *last_used*, é verificado se o utilizador associado a cada um deles é o utilizador autenticado no sistema. Após ser encontrado o elemento *last_used* desse utilizador guarda-se todos os seus sub-elementos que constituem a lista de todos os diagramas usados pelo utilizador.

4.2.1.2.3 Ler o XML_Users.xml

Neste XML, ao longo da leitura dos elementos *user*, verifica-se se o elemento *user* lido corresponde ao utilizador autenticado no sistema a partir do seu ID e, se assim for, é guardada a informação desse utilizador. A implementação foi feita desta forma porque o desenvolvimento do módulo de autenticação no sistema não estava incluído no âmbito do projecto de estágio.

4.2.1.3 Determinar os últimos n diagramas utilizados

No tópico anterior, foi mencionado que após a leitura dos elementos *last_used* são guardados todos os registos dos diagramas utilizados por um utilizador. No entanto, o que a aplicação pretende apresentar é os últimos n diagramas utilizados, em que n é um número inteiro maior que zero.

Por esta razão, foi implementada esta funcionalidade que verifica quais são os últimos n diagramas utilizados a partir do *array* de registos, da seguinte forma:

1. Se o número de registos é menor ou igual a n:
    a. Guarda todos os registos;
2. Se o número de registos for maior que n:
    a. Cria um *array* temporário com todos os registos;
    b. Percorre todo o *array* comparando todos os seus elementos 2 a 2, guardando no fim o elemento com a data de último acesso mais recente. Se houver 2 elementos com a mesma data, o desempate é feito pelo número de vezes que cada diagrama foi utilizado, sendo escolhido o mais utilizado;
    c. Elimina o elemento encontrado do *array*, para em seguida ser procurado o próximo registo mais recente;
    d. Quando o número de elementos encontrados for igual a n, termina a procura.
3. Cria a árvore “Last Used” só com nós folha, que constituem a lista dos últimos n diagramas utilizados a ser visualizada na aplicação.

A opção de tornar a lista dos últimos n diagramas utilizados como uma árvore de um só nível, deve-se ao facto de tanto esta lista como as árvores de categorias serem visualizadas na mesma caixa da interface. O facto de se tornar a lista numa árvore possibilita que esta seja tratada pela aplicação da mesma forma das árvores de categorias, utilizando assim as mesmas funções.

4.2.1.4 Zoom da imagem

A funcionalidade de fazer *zoom in/out* à imagem humana foi implementada de forma a ser accionada pelo uso da roda do rato (*mouse wheel*) e pela manipulação do *slider* da roda de controlo (ver Figura 36).

A imagem humana está contida num *Canvas*. Esse *Canvas* proporciona a possibilidade de ser adicionado a ele um grupo de transformação que inclui o escalamento (*scale*) do

Canvas e, por conseguinte, da imagem que está nele contida. Daí que a realização de *zoom* à imagem passe apenas por alterar o valor de escalamento, originalmente igual a 1.

O Silverlight não possui de momento suporte de raiz para os eventos da roda do rato. No entanto, existe uma maneira de suportar estes eventos, através da adição de eventos da página Html que contém a aplicação em Silverlight. Esses eventos recebem a propriedade do *browser*, em que a aplicação está a correr, relativa à roda do rato. O valor dessa propriedade indica a rotação realizada na roda do rato e conforme esse valor é negativo ou positivo, sabe-se em que sentido ocorreu a rotação. Perante esse facto, aplicou-se um factor de multiplicação que aumenta ou diminui o valor de escalonamento.

O *slider*, para realizar o *zoom*, altera também o valor de escalonamento, mas simplesmente conforme o valor escolhido no *slider*.

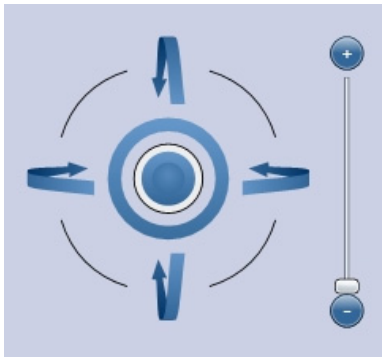

Figura 36 - Roda de controlo implementada

4.2.1.5 Mover a imagem

A funcionalidade de mover a imagem do corpo humano pode ser efectuada através de um movimento de *drag and drop* sobre a zona que contém a imagem ou com o controlo *joystick* da roda de controlo (ver Figura 36).

Tal como na funcionalidade de *zoom*, a movimentação da imagem é realizada pelo grupo de transformação do *Canvas* que contém a imagem. Neste caso, os valores que são afectados são os referentes à transformação de translação (*translate*). Estes valores estão inicialmente definidos a zero, tanto na coordenada X como na coordenada Y.

Para se mover a imagem por um movimento de *drag and drop* o utilizador tem que efectuar os seguintes passos:

1. Pressionar o botão esquerdo do rato sobre uma zona da imagem;
2. Movimentar o rato em qualquer direcção mantendo o botão pressionado;
3. Largar o botão esquerdo do rato terminando assim o movimento.

Na aplicação, a realização destes passos processa-se da seguinte forma:

1. Quando o rato entra na zona da imagem é activada a *flag* que o indica;
2. Ao ser pressionado o botão esquerdo do rato com esta *flag* activa, significando que o botão foi pressionado sobre a zona da imagem:
    a. são guardados os valores actuais de translação da imagem, ou seja, os valores iniciais da translação;

   b. são guardadas as coordenadas da posição do rato, ou seja, a sua posição inicial;

   c. é activada a *flag* que indica que uma translação pode ocorrer;

3. Ao ser movimentado o rato com o botão pressionado e com a *flag* que indica a ocorrência de uma translação activa:

   a. são calculados os valores de translação da imagem como a soma entre o valor da translação guardado inicialmente, quando o botão esquerdo do rato foi pressionado, e a distância percorrida pelo rato. Essa distância é igual à diferença entre a posição actual do rato e a posição inicial do rato. A fórmula de cálculo da translação é a seguinte para a coordenada X (sendo idêntica para a coordenada Y):

```
Translação.X = valor_inicial_da_translação.X + (posição_actual_do_rato.X – posição_inicial_do_rato.X)
```

4. Quando o botão esquerdo é libertado, o movimento termina, desactivando a *flag* que indica a ocorrência de uma translação.

A outra forma de mover a imagem é usando o controlo *joystick* da roda de controlo da seguinte forma:

1. Pressionar o botão esquerdo do rato sobre o controlo;
2. Movimentar o rato em qualquer direcção mantendo o botão pressionado. A imagem humana movimenta-se de acordo com a direcção em que o controlo foi posicionado em relação ao seu centro;
3. Largar o botão esquerdo do rato terminando assim o movimento.

O *joystick* pode ser movido dentro de um limite circular, sendo que quanto mais afastado do centro, mais rápido será o movimento da imagem. Tendo em conta que a imagem deve continuar a mover-se enquanto o *joystick* estiver fora do centro, foi necessário definir um temporizador que de 1 em 1 centésimo de segundo repetisse a translação de acordo com a translação actual do *joystick* relativamente ao seu centro. Um temporizador deste tipo pode ser implementado, em Silverlight, recorrendo a uma animação (*Storyboard*) vazia com a duração de 1 centésimo de segundo e usando a função que gere o evento de conclusão da animação para saber quando esse tempo passou.

A implementação deste *joystick* seguiu então os seguintes passos:

1. Ao ser pressionado o botão esquerdo do rato sobre o controlo:

   a. é activada a *flag* que indica que o controlo foi pressionado;

   b. são guardados os valores actuais de translação da imagem, ou seja, os valores iniciais da translação;

   c. é iniciada a animação respeitante ao temporizador;

2. Em cada conclusão da animação:

   a. são adicionados os valores actuais de translação do controlo aos valores de translação da imagem; inicialmente os valores de translação do controlo são ambos iguais a 0, visto o controlo estar posicionado no centro;

b. é iniciada nova animação de forma a ser criado um ciclo, mantendo a contagem temporal de 1 em 1 centésimo de segundo.

3. Ao ser movimentado o rato com o controlo pressionado:

   a. é calculada a distância de movimentação do rato, para verificar se está dentro do limite máximo de movimentação do controlo *joystick*. Essa distância é calculada através do teorema de Pitágoras, visto que a distância percorrida em X forma um triângulo rectângulo com a distância percorrida em Y:

```
Distância^2 = movimento_do_rato.X^2 + movimento_do_rato.Y^2
```

   b. Se a distância for menor ou igual ao limite, a translação do controlo é igual ao movimento do rato;

   c. Se a distância for superior ao limite, a translação do controlo será a translação máxima permitida pelo limite, em que:

      i. é calculada a percentagem da distância em excesso em relação ao limite;

      ii. os valores das coordenadas de translação do controlo são iguais à diferença entre os valores das coordenadas de distância do movimento do rato e os valores em excesso dessas coordenadas de acordo com a percentagem calculada;

4. Ao ser libertado o botão esquerdo do rato:

   a. é desactivada a *flag* que indica que o controlo está pressionado;

   b. termina-se a animação do temporizador, parando assim o movimento da imagem;

   c. termina-se o movimento do *joystick* recolocando-o no seu ponto central.

4.2.1.6 Seleccionar uma zona da imagem

Para seleccionar uma zona da imagem humana, o utilizador tem que passar o rato por cima da imagem para posteriormente clicar uma das zonas. Ao passar o rato por cima da imagem, a zona em que o rato está é colorida com uma cor diferente e é indicado o nome da zona, servindo de apoio à escolha do utilizador. Para isto foi definido na interface gráfica, em XAML, uma camada 100% transparente, dividida em várias zonas de selecção, com a forma da imagem humana e sobreposta a esta. Quando o rato passa por cima dessa camada, a zona de selecção em que o rato está é pintada com uma cor azulada semitransparente através da propriedade de cor das formas em Silverlight. Todas as zonas pintadas anteriormente são novamente pintadas com a cor totalmente transparente, ficando assim invisíveis e ficando só pintada, a zona em que o rato está actualmente (ver Figura 37).

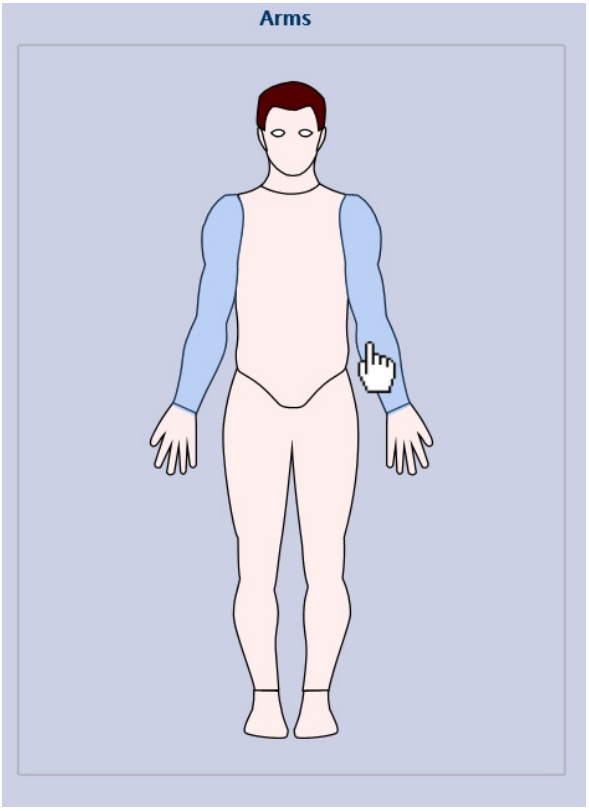

Figura 37 - Rato sobre os braços da imagem humana

Ao ser clicada uma das zonas da imagem, é despoletada a acção, definida na base de dados, para essa zona seleccionada (que define um mapa de selecção). A implementação desta ocorrência, segue os seguintes passos:

1. A partir da designação da zona seleccionada e da camada da imagem humana actualmente activa, é procurado o mapa respectivo no *array* de mapas guardados em memória após a leitura dos XML;
2. Após ser encontrado o mapa, realiza-se a acção associada a esse mapa da seguinte forma:
    a. é definida a camada da imagem a visualizar;
    b. são calculados os valores de translação da imagem a partir do seu ponto central para o novo ponto em que a imagem será centrada; os valores de translação são iguais à diferença de coordenadas entre o ponto central (x,y) = (largura/2, altura/2) e o novo ponto; as coordenadas do novo ponto são calculadas a partir das suas coordenadas relativas, ou seja, as coordenadas são (x,y) = (largura*centerX, altura*centerY) em que centerX e centerY são as coordenadas relativas definidas na tabela Action da base de dados;
    c. é realizada uma animação com o valor de *zoom* a aplicar e os valores de translação calculados; esta animação é criada dinamicamente, sendo aplicada sobre os elementos de escalamento e de translação do grupo de transformação do *Canvas* que contém a imagem humana; os valores recebidos pela animação são os valores de destino a ser atingidos no fim da animação; ao longo do tempo de duração da animação, a

imagem é deslocada e escalada de forma constante, desde os seus valores iniciais até aos valores de destino;

3. São abertos, na árvore de diagramas, todos os nós que contêm diagramas associados à zona clicada, através do identificador do mapa seleccionado.

Como exemplo, a selecção da zona da cabeça (Figura 38) despoleta a acção que centra e aproxima a visualização da imagem dessa mesma zona, mantendo a mesma camada da imagem, a camada exterior, como é ilustrado na Figura 39. Neste exemplo, a selecção da cabeça na imagem da camada exterior do corpo humano origina a abertura dos nós que contêm o diagrama dos olhos (Figura 40).

Na implementação do ponto 3, surgiram alguns problemas, no sentido em que a procura e abertura dos nós da árvore fazia interromper a animação de escalonamento e translação mencionada no ponto 2.c. A solução mais óbvia para a resolução deste problema, seria realizar a abertura da árvore após a animação ter terminado, a partir do evento que indica a conclusão da animação. No entanto, aqui surgiu outro problema, mas este derivado do próprio Silverlight que, de momento, apresenta uma falha ao accionar estes eventos. Essa falha origina que o evento é accionado cedo demais e não realmente quando a animação termina, pelo que o problema de interrupção da animação se mantinha.

A única forma de resolver o problema foi implementando uma animação de carregamento da árvore com uma duração breve (meio segundo), mas suficiente para garantir que a animação da imagem termina antes de iniciar o carregamento da árvore.

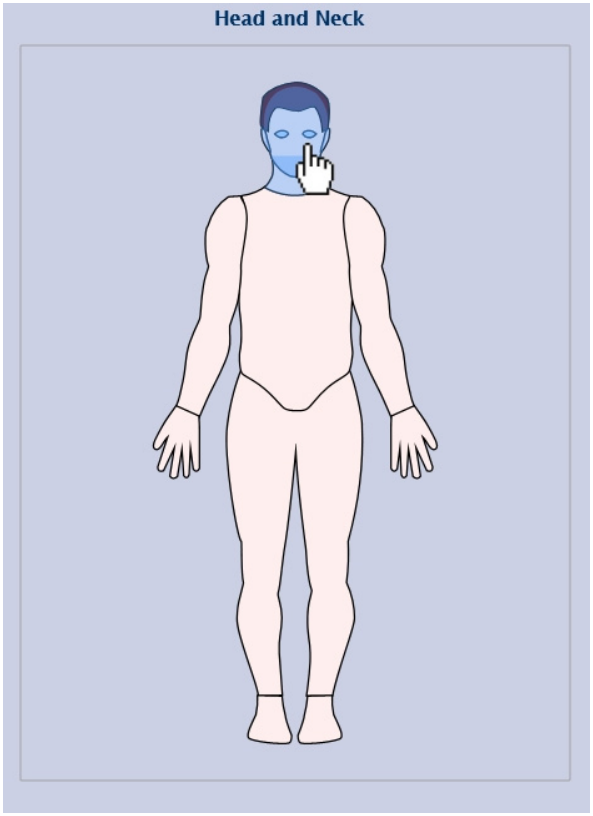

Figura 38 - Rato sobre a zona da cabeça da imagem humana

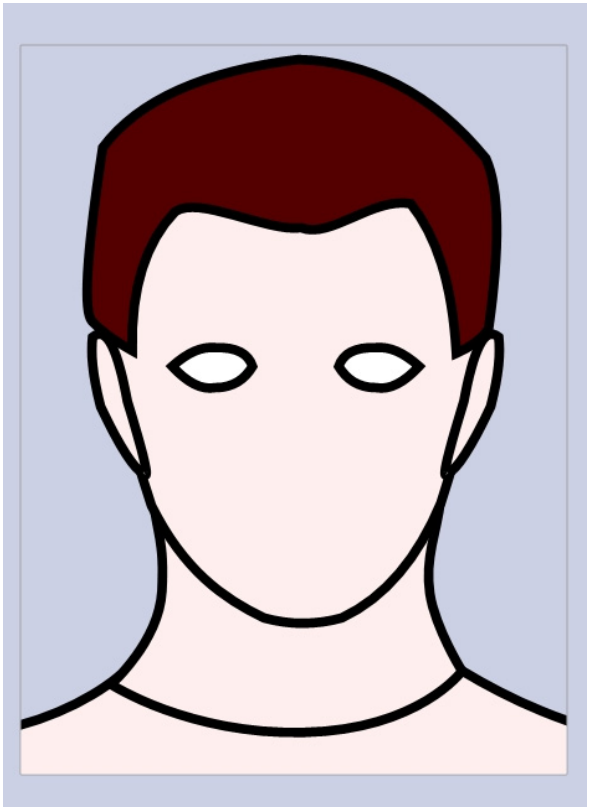

Figura 39 - Resultado da selecção da cabeça da imagem humana

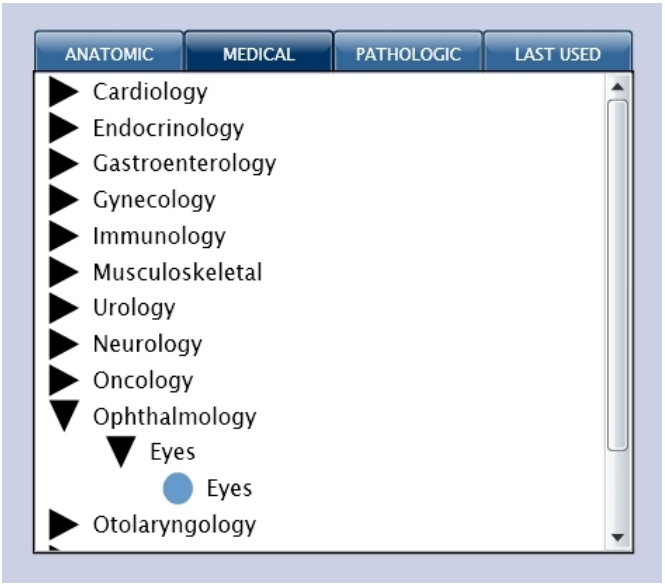

Figura 40 - Nós abertos na árvore Medical como resultado da selecção da cabeça na camada exterior

4.2.1.7 Alterar a imagem para outra camada do corpo humano

Esta funcionalidade foi implementada a partir do *slider in/out* ilustrado na Figura 41. Esse *slider* alterna apenas entre valores inteiros, em que cada um deles está associado a uma camada diferente. Esta associação foi feita a partir de um *array* de *Canvas* cujos elementos estão associados a cada uma das camadas. O valor inteiro escolhido no *slider* refere-se assim ao índice no *array* da camada activa.

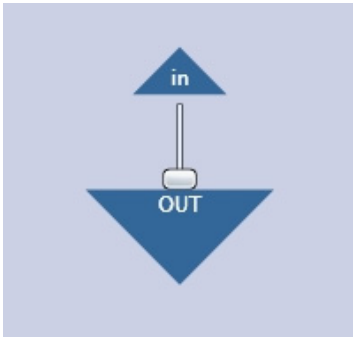

Figura 41 - O *slider in/out* implementado

Ao ser escolhida uma nova camada, essa camada é colocada visível, através da alteração da propriedade *Visibility* dos *Canvas*. Todas as outras camadas, excepto a camada exterior, são colocadas invisíveis. A camada exterior não é colocada invisível mesmo quando é seleccionada outra camada, dado que a camada exterior servirá de sombra à nova camada seleccionada, servindo de guia para as zonas de selecção. A camada exterior é, nestes casos, colocada semitransparente, atrás da imagem da nova camada (ver Figura 42). Ao ser seleccionada novamente, a camada exterior volta a tornar-se totalmente opaca (ver Figura 43).

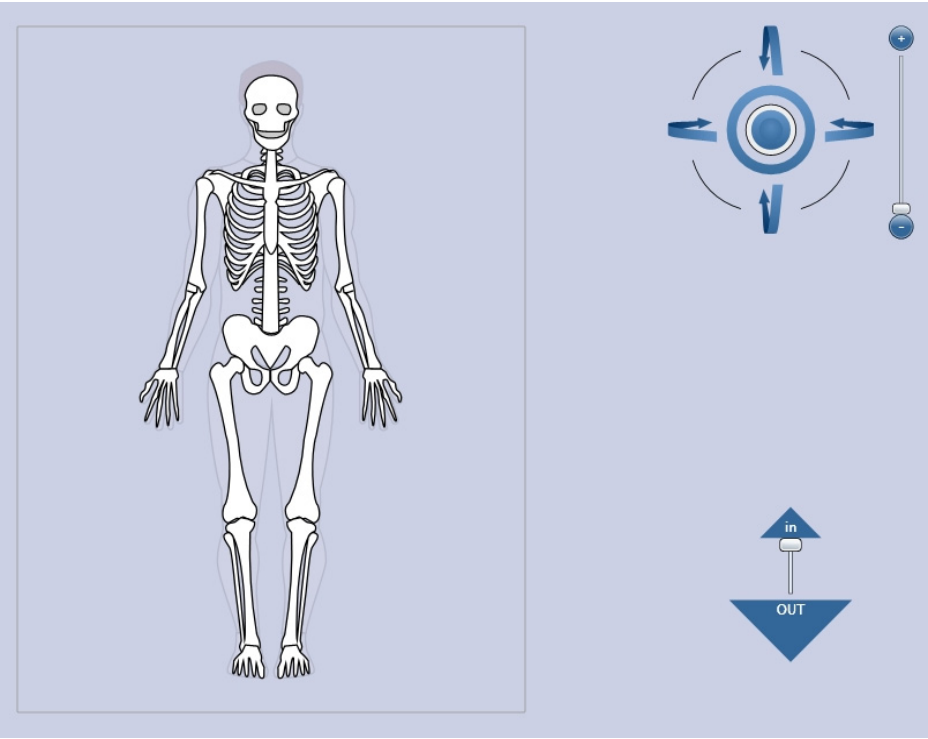

Figura 42 - Camada do esqueleto seleccionada

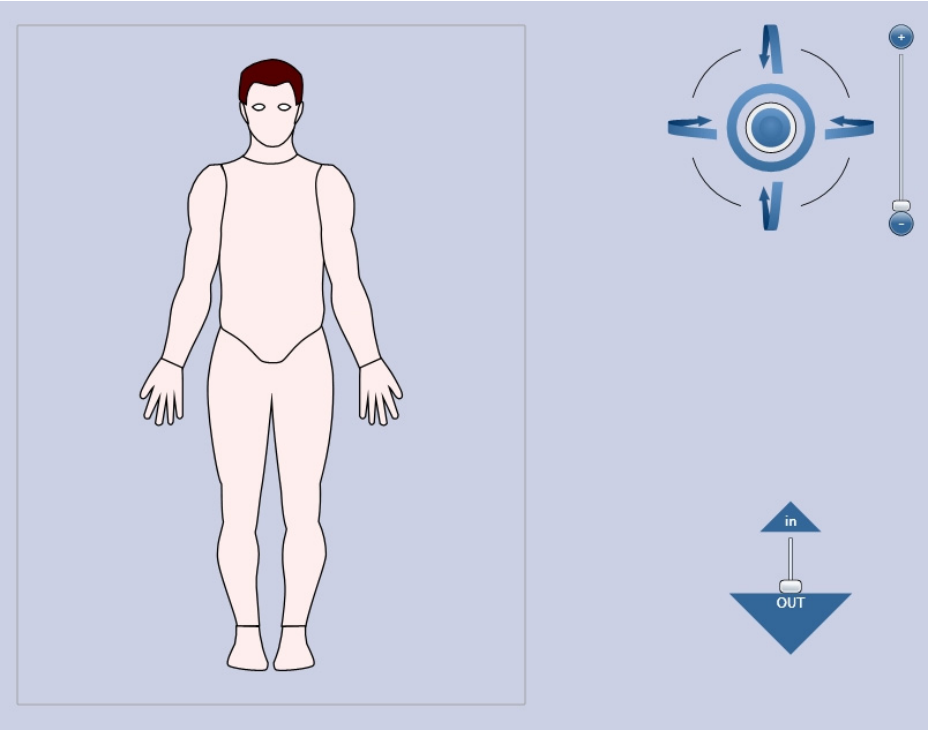

Figura 43 - Camada exterior do corpo humano seleccionada

4.2.1.8 Alterar o estado da imagem (Undo e Redo)

A aplicação mantém um histórico das acções realizadas sobre a imagem humana, para que o utilizador possa sempre voltar ao estado inicial ou a outro estado anterior após realizar uma acção indesejada.

Este histórico é mantido através de um *array*, de dimensão dinâmica, de estados da imagem. Cada estado guarda a seguinte informação:

- nível de *zoom* ;
- coordenadas relativas do ponto em que a imagem se encontra centrada;
- imagem da camada do corpo humano activa;
- nome da acção realizada sobre a imagem que originou a mudança de estado.

As coordenadas relativas são iguais à razão entre as coordenadas absolutas do ponto em que a imagem está centrada e as dimensões da imagem. As coordenadas absolutas são iguais à diferença entre as coordenadas do meio da imagem e os valores de translação actuais. As fórmulas de cálculo das coordenadas relativas são as seguintes:

```
centerX = (largura / 2 – translação.X) / largura;
centerY = (altura / 2 – translação.Y) / altura;
```

Um novo estado é gravado no histórico quando são realizadas as seguintes acções:

- *Zoom in/out* da imagem;
- Movimentação da imagem;
- Selecção de uma zona da imagem;
- Selecção de um diagrama da árvore, que origina a visualização na imagem, da localização do diagrama;

- Alteração da camada visível do corpo humano.

Acções sucessivas de *zoom* ou movimentação da imagem são agrupadas num estado único. Assim, o estado que ficará guardado será o da última operação da sequência de acções do mesmo tipo. Desta forma, poupa-se espaço em memória.

Os estados gravados podem ser percorridos através das operações de *undo* e *redo*. Ao ser realizada uma operação de *undo*, é carregado o estado imediatamente anterior ao actual, gravado no *array* de estados.

A operação de *redo* realiza o inverso, permitindo carregar um estado posterior ao estado actual. Isto só pode acontecer quando tiver sido realizada uma operação de *undo* anteriormente.

4.2.2 TreeClass

Esta classe define a *Tree View* do Outliner e todas as suas funções e métodos necessários para o seu manuseamento. É chamada pela classe *Page* que lhe transmite a informação da árvore seleccionada e dos diagramas disponíveis. Tendo essa informação, a *TreeClass* cria a *Tree View*, inicialmente apenas com os nós do primeiro nível visíveis mas fechados, dispostos numa tabela do Silverlight (elemento *Grid*).

A classe trata também dos eventos de abertura e fecho dos nós da árvore, de forma que, quando um evento de abertura de um nó é accionado, esta lê os dados dos nós-filho do nó seleccionado e cria uma nova linha, imediatamente após a linha a que pertence o nó clicado, na tabela criada anteriormente, definindo uma nova tabela (*Grid*), com os nós-filho, dentro dessa nova linha.

Por outro lado, fechar um nó apenas remove a linha criada previamente, na abertura do nó, com os nós-filho, quando o nó-pai é clicado para ser fechado.

A *TreeClass* contém também métodos para abrir os nós quando é feito um clique na imagem representativa do corpo humano, através das suas ligações com o mapa da imagem clicado, ou quando é realizada uma pesquisa, pelo campo de pesquisa da aplicação.

Descreve-se em seguida as principais funcionalidades implementadas nesta classe.

4.2.2.1 Definir um nível da árvore

Esta funcionalidade permite definir a lista de nós de um nível de uma árvore. No caso do primeiro nível da árvore, o nível 0 neste caso, é definida a lista de todos os nós desse nível. Nos níveis seguintes, é definida a lista de todos os nós desse nível que pertencem ao nó-pai que os contém.

Esta funcionalidade foi implementada da seguinte forma:

1. Procura todos os nós do nível que está a ser criado e que pertencem à tabela do nó-pai que os contém, guardando esses nós num *array*;
2. A partir do número de nós encontrados, define o número de linhas para a tabela que conterá a lista de nós. Estas tabelas têm todas duas colunas, em que a primeira coluna contém um símbolo que indica um nó passível de ser aberto ou fechado, ou indica uma folha;
3. O *array* de nós criado anteriormente é percorrido e para cada nó realiza-se o seguinte:

a. é criado um símbolo de acordo com o tipo de nó, ou seja, se é um nó ou uma folha. Esse símbolo é adicionado à primeira coluna da tabela na linha respectiva à sua posição no *array*;

b. se o nó for do tipo folha, é criado o seu *thumbnail* que é definido como um objecto imagem do Silverlight. O *thumbnail* é colocado inicialmente visível ou invisível de acordo com a *checkbox* que activa ou desactiva a visualização dos *thumbnails*;

c. é criado um bloco de texto com o nome do nó, em que:

   i. se o nó for do tipo nó, o nome será o nome do nó no *array* de nós, que corresponderá ao nome de uma categoria;

   ii. se o nó for do tipo folha, a aplicação irá buscar o nome ao *array* de diagramas, criado na leitura dos XML que definem a base de dados, a partir do identificador desse diagrama que está guardado no nó;

d. é criado um objecto rectângulo com cor transparente que define a zona de selecção do nó.

Os elementos criados de 3.b a 3.d, ao contrário do símbolo, são todos adicionados à segunda coluna da tabela.

Na Figura 44 apresenta-se um exemplo de uma tabela de nós, do nível 0 de uma árvore, com os limites das suas colunas e linhas visíveis e com os elementos de uma das suas linhas identificados.

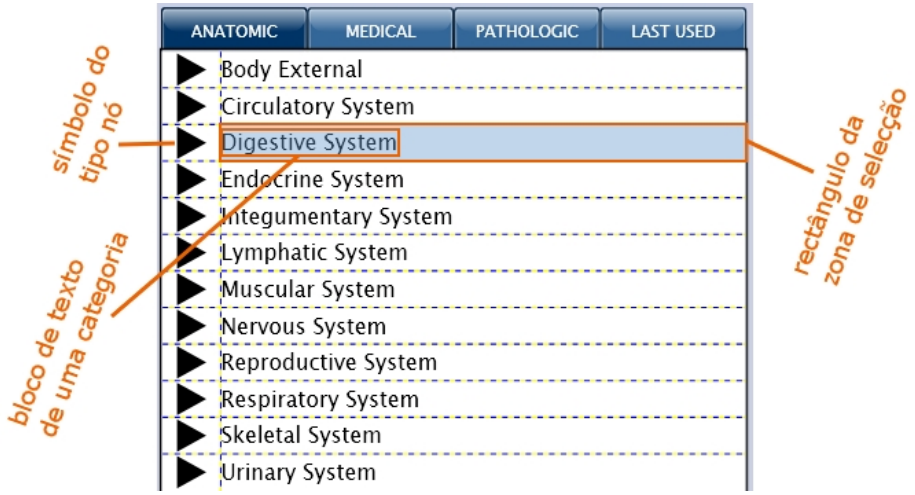

Figura 44 - Exemplo de uma tabela de nós do nível 0 e os seus elementos

4.2.2.2 Abrir um nó da árvore

A funcionalidade de abrir um nó da árvore de diagramas ocorre quando é clicado o símbolo de um nó fechado, como os representados na Figura 44. Quando isto ocorre a aplicação segue os seguintes passos:

1. é definida uma nova tabela com os nós-filho do nó que foi clicado;
2. na tabela-pai, a tabela à qual o nó clicado pertence, é definida uma nova linha que é sempre criada no fim da tabela;

3. todos os elementos da tabela-pai que pertencem às linhas abaixo da linha do nó clicado, são movidas uma linha para baixo; desta forma, a nova linha criada fica uma posição imediatamente abaixo da linha do nó clicado;
4. nessa nova linha, na segunda coluna, é adicionada a nova tabela definida no passo 1.

Este processo é ilustrado na Figura 45. Nela, o símbolo clicado está representado a laranja, a nova linha criada a amarelo e a nova tabela com os nós-filho a azul-escuro.

Na Figura 46 apresenta-se, como exemplo, o resultado da abertura do nó referente à categoria “Circulatory System”. Nela visualiza-se a nova tabela criada com os seus nós-filho.

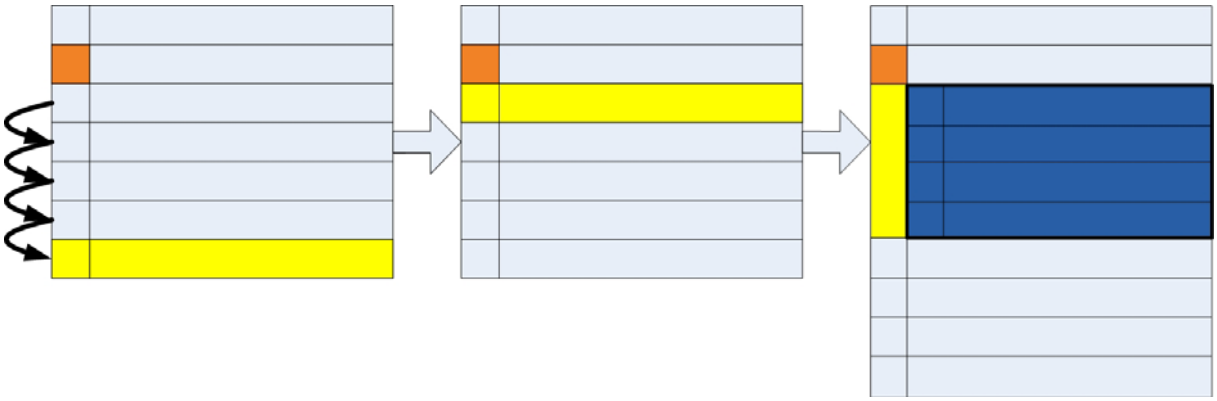

Figura 45 - Ilustração da criação de uma nova linha com a tabela dos nós-filho

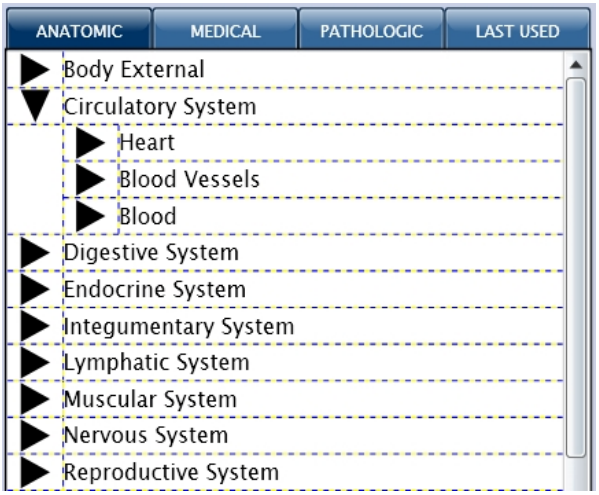

Figura 46 - Exemplo da árvore “Anatomic” com o nó “Circulatory System” aberto

4.2.2.3 Fechar um nó da árvore

Fechar um nó da árvore ocorre quando o símbolo de um nó aberto é clicado. Ao suceder isso, é eliminada a linha imediatamente a seguir à linha do nó clicado. Esta funcionalidade processa-se de forma inversa à funcionalidade anterior:

1. é removida a tabela dos nós-filho do nó clicado;
2. todos os elementos da tabela-pai, pertencentes a linhas abaixo da linha da tabela dos nós-filho, são movidos uma linha acima;
3. é removida a última linha, que se encontra vazia, da tabela-pai.

A implementação desta funcionalidade seguiu estes passos porque o Silverlight não permite eliminar uma linha de uma tabela antes dessa linha estar vazia.

O processo descrito é ilustrado na Figura 47 com o mesmo código de cores mencionado na funcionalidade anterior.

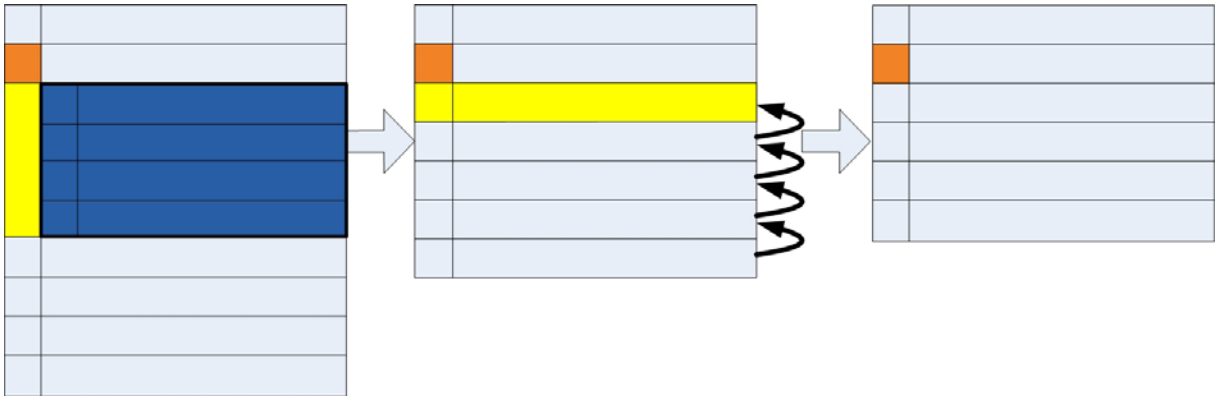

Figura 47 - Ilustração da remoção da linha dos nós-filho de uma tabela

4.2.2.4 Seleccionar um nó da árvore

Um nó da árvore de diagramas é seleccionado quando é clicado o rectângulo que define a zona de selecção desse nó. Nesse momento, a aplicação segue os passos descritos a seguir:

1. O rectângulo clicado é pintado com uma cor azul semitransparente e qualquer rectângulo seleccionado anteriormente é repintado com uma cor totalmente transparente;
2. Esse rectângulo clicado é guardado numa variável global, de forma a que, quando um outro nó (rectângulo) é clicado, o rectângulo volte a ser pintado de transparente como mencionado no ponto anterior. Assim, poupa-se bastante tempo de processamento, visto que não se necessita de procurar em todos os nós da árvore qual o que está actualmente seleccionado;
3. É procurado o bloco de texto que possui o conteúdo do nó, correspondente ao rectângulo clicado. Este bloco de texto foi definido anteriormente com uma propriedade que indica se o nó é do tipo nó ou do tipo folha. Assim, poupando mais uma vez tempo de processamento, visto que não é necessário procurar essa informação no *array* de nós da árvore actualmente activa. Esta informação é importante para o ponto seguinte;
4. Se o nó for do tipo folha:
    a. é procurada a descrição do diagrama seleccionado, no *array* de diagramas, definindo a descrição na caixa de texto definida na GUI para a visualização da descrição de um diagrama;
    b. através do identificador do mapa, que define uma zona da imagem humana, associado ao diagrama seleccionado, é despoletada na imagem, a acção respeitante a esse mapa, que mostrará a zona na imagem a que o diagrama corresponde.

Como exemplo, ao ser seleccionado o nó-folha, ou seja, o diagrama, referente aos olhos (*Eyes*) na árvore de diagramas (ver Figura 48), a imagem executará uma acção igual à executada aquando de um clique na cabeça da imagem (ver secção 4.2.1.6), mas usando o identificador do mapa associado a esse diagrama. O resultado na imagem é o mesmo que o ilustrado na Figura 39.

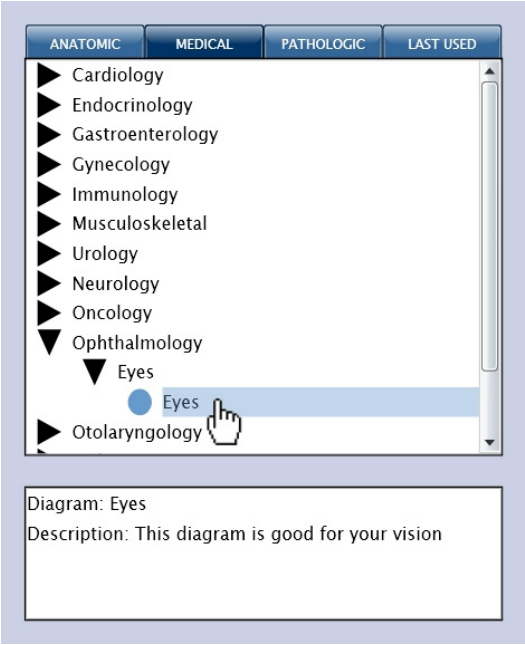

Figura 48 - Selecção do diagrama *Eyes* na árvore de diagramas “Medical”

4.2.2.5 Encontrar nós da árvore através do campo de pesquisa por nome

A aplicação possui um campo de pesquisa por nome do diagrama ou da categoria (ver Figura 49). Este possibilita ao utilizador procurar diagramas, ou categorias de diagramas, escrevendo o nome ou parte do nome deles.

Figura 49 - Campo de pesquisa da aplicação

Quando o utilizador escreve o texto que pretende procurar e clica no botão *Search*, a aplicação percorre o *array* de nós da árvore de diagramas e para cada nó:

1. recebe o seu nome, em que:
    a. se o nó for do tipo nó, ou seja, uma categoria, o nome será o nome guardado no nó;
    b. se o nó for do tipo folha, ou seja, um diagrama, a aplicação irá buscar o nome do nó ao *array* de diagramas a partir do identificador do diagrama guardado no nó;
2. se a árvore activa for a árvore “Last Used”, que é na verdade uma lista de diagramas:

a. verifica se o texto escrito no campo de pesquisa é igual ou está contido no nome do nó: esta verificação não toma em consideração letras maiúsculas ou minúsculas:

   i. se o nó corresponder à pesquisa, coloca todos os elementos visíveis da linha do nó;

   ii. caso contrário, coloca todos os elementos invisíveis da linha do nó;

3. se a árvore activa for qualquer uma das árvores de categorias de diagramas:

   a. se o nome do nó corresponder ao texto escrito no campo de pesquisa ou o contiver, são abertos todos os nós da árvore até ao nó encontrado da seguinte forma:

      i. é recebido o caminho de nós até ao nó encontrado, pesquisando desde a posição do nó para as posições anteriores no *array* de nós, guardando o primeiro nó de nível superior encontrado e após esse, guardando o primeiro nó do nível superior seguinte a esse, e assim sucessivamente até ao nó de nível 0, sendo guardados todos os nomes dos nós que são ascendentes do nó encontrado;

      ii. para cada nó do caminho de nós, é procurada a linha do nó na tabela a que pertence, sendo lançada a funcionalidade de abrir o nó da árvore.

Na Figura 50 pode ser visualizado um exemplo da realização de uma pesquisa sobre a lista de diagramas “Last Used”, enquanto que na Figura 51 pode ver visualizado um exemplo de uma pesquisa sobre a árvore “Medical”.

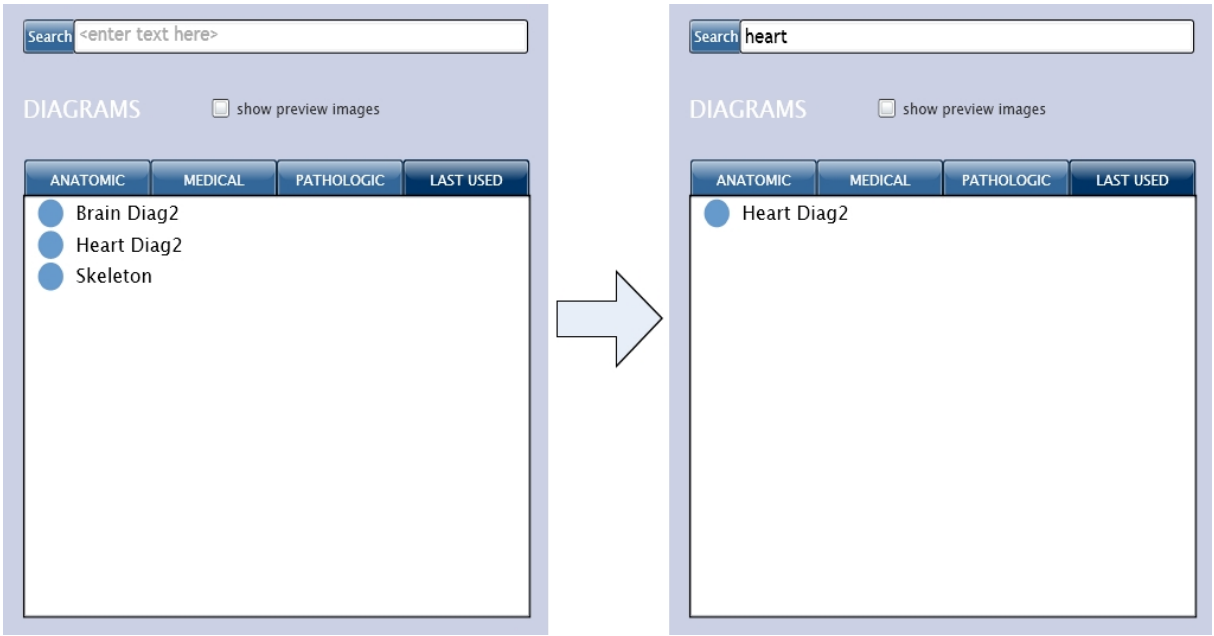

Figura 50 - Pesquisa pela palavra “heart”, sobre os diagramas da lista “Last Used”

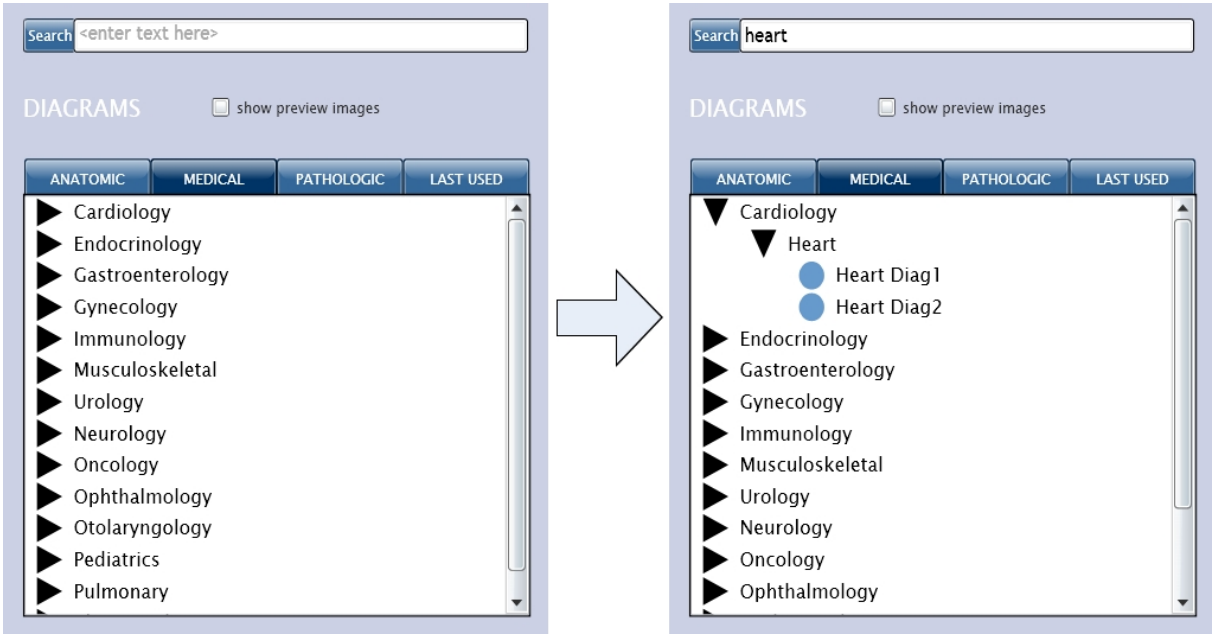

Figura 51 - Pesquisa pela palavra “heart” na árvore “Medical”

4.2.2.6 Definir a visibilidade dos thumbnails dos diagramas

Esta funcionalidade possibilita activar ou desactivar as imagens de pré-visualização dos diagramas (os *thumbnails*) na árvore de diagramas. A activação/desactivação dos *thumbnails* acontece de acordo com a activação/desactivação da *checkbox* “show preview images” (ver Figura 52).

Figura 52 - Esquerda: *checkbox* desactiva - Direita: *checkbox* activa

Os passos que a implementação desta funcionalidade segue são:

1. são percorridos todos os elementos de um nível da árvore e, se o elemento for uma imagem:
    a. a aplicação verifica se o nó, ao qual a imagem pertence, é visível ou não;
    b. se o nó for visível, o valor da visibilidade da imagem será definido conforme a activação ou desactivação da *checkbox*;
    c. os nós não visíveis ocorrem apenas na árvore que corresponde à lista “Last Used” e quando estes existem não se pode definir a visibilidade da imagem, visto que ela tem que se manter não visível como o resto do nó, caso contrário, apareceria a imagem sem o resto do conteúdo do nó;
2. se o elemento da árvore for uma *Grid*, ou seja, uma tabela, é chamada esta mesma função mas agora para essa tabela que corresponde a um nível seguinte da árvore, repetindo-se assim, o ponto 1 para este nível.

Na Figura 53 pode ser visto o resultado da activação/desactivação dos *thumbnails* dos diagramas na aplicação.

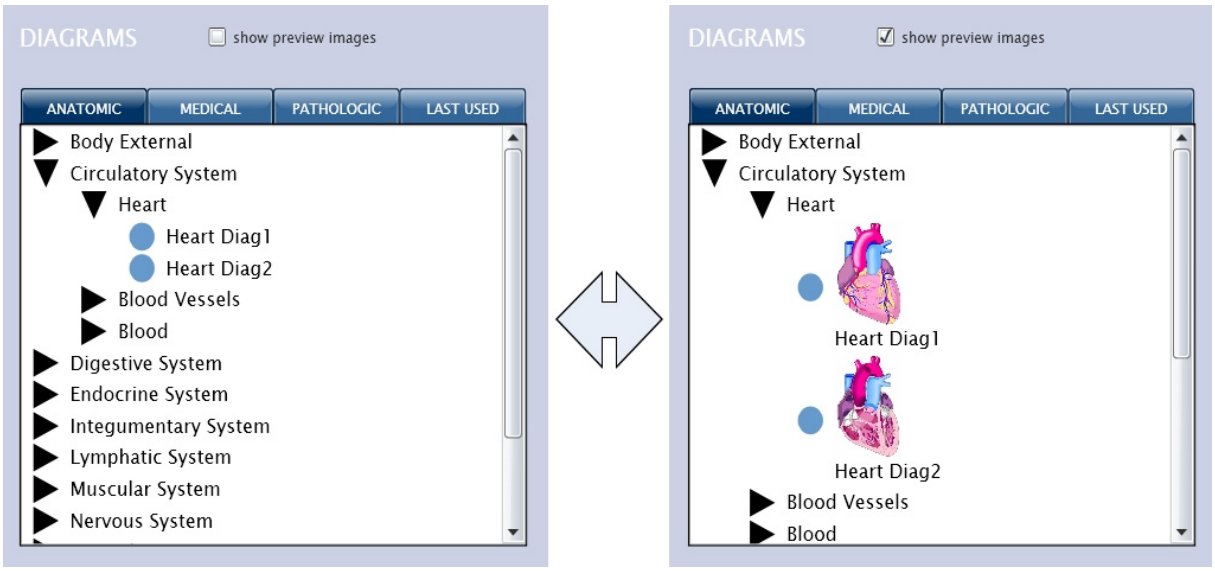

Figura 53 - Activação/desactivação dos *thumbnails* dos diagramas

4.2.3 Structs

A classe *Structs* é uma classe auxiliar às referidas anteriormente. Esta classe tem definidas todas as estruturas da aplicação necessárias para guardar a informação durante a execução. Estas estruturas fazem correspondência com os dados provenientes dos ficheiros XML, que são lidos pela aplicação através da classe *Page*. Desta forma, esta classe auxiliar é usada por ambas as classes *Page* e *TreeClass* para tratarem toda a informação associada às estruturas.

Em seguida descrevem-se todas as estruturas incluídas nesta classe.

4.2.3.1 Selection_group

Esta estrutura serve para definir grupos de selecção de zonas da imagem humana, dependentes do nível de *zoom* da imagem, o que permite existirem zonas de selecção mais alargadas quando a imagem é mais pequena e, pelo contrário, existir a possibilidade de escolher zonas mais específicas quando a imagem é mais alargada.

Os campos desta estrutura são os seguintes:

- Nome do grupo de selecção;
- Nível de *zoom* a partir do qual este grupo é activado;
- *Array* de figuras (*paths*) da imagem humana que fazem parte do grupo.

4.2.3.2 Map_action

Esta estrutura corresponde na aplicação, à tabela Action da base de dados, mencionada anteriormente. Os seus campos correspondem aos dessa tabela implementada em XML:

- Nome da imagem do corpo humano a apresentar;
- Nível de *zoom* a aplicar à imagem;

- Coordenada X do ponto em que a imagem será centrada;
- Coordenada Y do ponto em que a imagem será centrada.

4.2.3.3 Map

Esta estrutura corresponde na aplicação, à tabela Map da base de dados, mencionada anteriormente.

Os campos desta estrutura são assim os seguintes:

- Identificador do mapa;
- Nome da figura (*path*) da imagem humana que define a zona de selecção;
- Nome da imagem da camada do corpo humano activa quando ocorrer a selecção deste mapa;
- Acção que ocorre quando este mapa é seleccionado, definida pela estrutura *map_action* descrita anteriormente.

4.2.3.4 Diagram_tree_node

Esta estrutura serve para definir um nó da árvore de diagramas.

Os campos que definem um nó são os seguintes:

- Nível da árvore a que o nó pertence;
- Nome do nó, que no caso de ser uma categoria, é o nome da categoria, caso contrário, sendo um diagrama, é o identificador do diagrama;
- Tipo do nó, que poderá ser do tipo nó (categoria) ou do tipo folha (diagrama).

4.2.3.5 Tree

Esta estrutura serve para guardar a informação de uma árvore com os seguintes campos:

- Nome da árvore;
- *Array* de nós da árvore, em que cada nó é definido pela estrutura *diagram_tree_node.*

4.2.3.6 Diagram

Esta estrutura serve para definir um diagrama, correspondendo à tabela Diagram da base de dados, mencionada anteriormente.

Os campos desta estrutura são:

- Identificador do diagrama;
- Nome do diagrama;
- Identificador do mapa (*map*) da imagem humana a que o diagrama está associado;
- Localização (*url*) do *thumbnail* do diagrama;
- Descrição do diagrama.

4.2.3.7 Last_used_diag

Esta estrutura define um item da lista dos últimos diagramas usados (*last used*) por um utilizador.

Os campos de um desses itens são os seguintes:

- Identificador do diagrama utilizado;
- Contador do número de utilizações do diagrama;
- Data e hora do último acesso ao diagrama.

4.2.3.8 User

Esta estrutura serve para definir um utilizador. Os seus campos correspondem aos da tabela User da base de dados, sendo os seguintes:

- Identificador do utilizador;
- Nome do utilizador;
- Área médica do utilizador;
- *Username* para autenticação no sistema;
- Palavra-chave para autenticação no sistema;
- *Array* de registos dos diagramas utilizados pelo utilizador, em que cada um deles é definido pela estrutura *last_used_diag.*

4.2.3.9 Image_state

Esta estrutura serve para definir um estado da imagem do corpo humano. Estes estados são os que permitem as operações de *undo* e *redo* da imagem.

Os campos guardados por um estado da imagem são os seguintes:

- Nome da acção que originou a criação do novo estado;
- Nome da imagem humana activa;
- Nível de *zoom* da imagem;
- Coordenada X do ponto em que a imagem estava centrada;
- Coordenada Y do ponto em que a imagem estava centrada.

4.2.4 App

Esta é a classe da aplicação para tratar os eventos de inicialização e término da aplicação. O ficheiro XAML associado, o App.xaml, contém os recursos da aplicação, ou seja, todos os *templates* dos botões e outros controlos personalizados.

4.3 Implementação da Interface com o Utilizador

A implementação da interface com o utilizador foi realizada através dos diversos elementos disponíveis no XAML do Silverlight. Esta implementação teve o protótipo inicial como guia, sendo o resultado visualizado na Figura 54.

No âmbito deste projecto foram desenhadas duas imagens de exemplo de camadas da anatomia humana (camada exterior e esqueleto), para servir de prova de conceito das funcionalidades associadas à manipulação das imagens humanas. Estes desenhos foram realizados na ferramenta Microsoft Expression Blend 2.5 que permite desenhar formas vectoriais definidas em XAML.

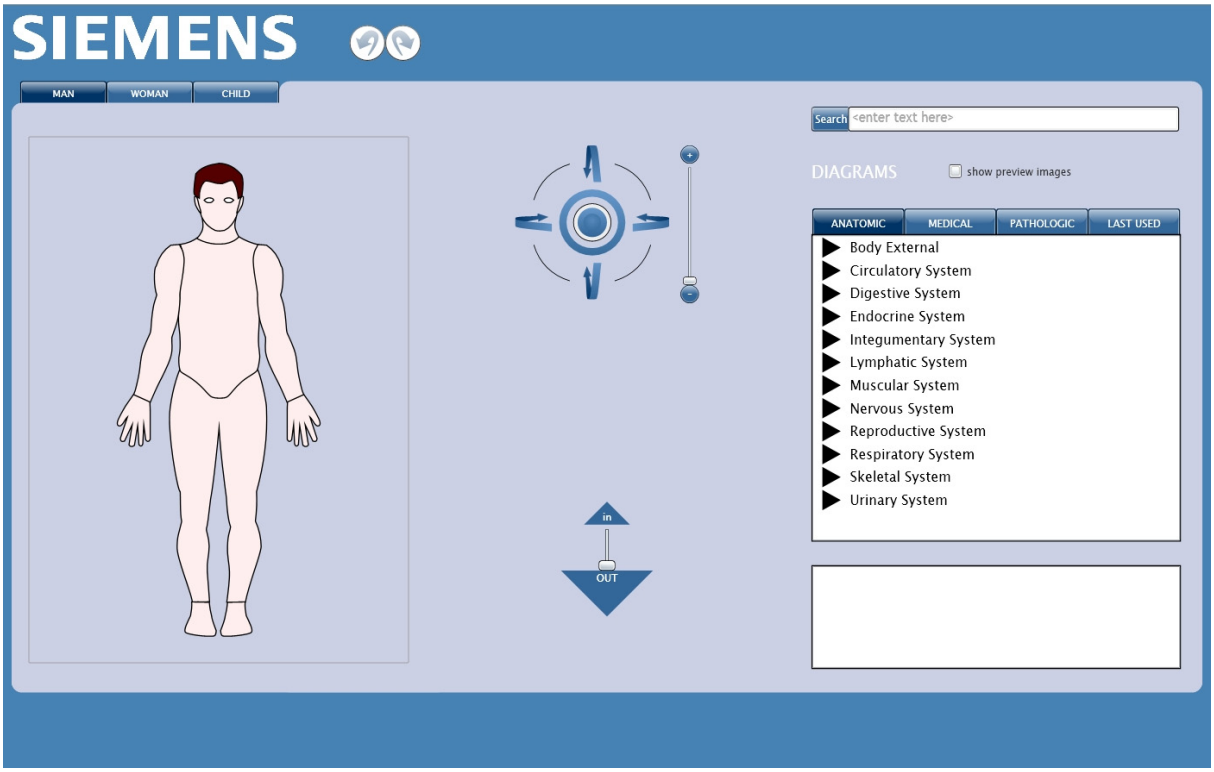

Figura 54 - Interface do Outliner

5 Conclusões e Trabalho Futuro

Neste capítulo, são apresentadas as conclusões resultantes do desenvolvimento do projecto Outliner, algumas dificuldades sentidas e as perspectivas de trabalho futuro.

A área das aplicações de *software* para a saúde revelou-se como uma área muito obscura, em que todas as empresas disponibilizam informação muito reduzida sobre os seus produtos. Isto verificou-se no levantamento do estado da arte do projecto em que, além de ser muito difícil encontrar produtos no âmbito deste projecto, foi também difícil encontrar informação sobre esses mesmos produtos quanto às funcionalidades que eles disponibilizam. No entanto, mesmo com estas dificuldades foi possível apresentar um número significativo de produtos existentes que permitem realizar anotações sobre diagramas médicos.

Após a análise do estado actual do Siemens Diagrammer, projecto em que se insere o Outliner, concluiu-se que este era um projecto ainda numa versão longe do pretendido no futuro, visto que algumas das funcionalidades que são pretendidas ainda não estão implementadas na totalidade. Além disso, o Diagrammer actual ainda aparece limitado em vários aspectos, em que o mais relevante para o Outliner é a falta de diagramas médicos que o Diagrammer possui. No entanto, o Outliner, foi desenvolvido com base nas perspectivas de futuro para o Diagrammer, em que virá a existir um número elevado de diagramas. Esta limitação actual apenas impediu que o projecto Outliner usasse os diagramas existentes no Diagrammer para testes, no entanto isto não foi relevante, pois foram usadas outras imagens como exemplos.

No levantamento dos requisitos, a realização de protótipos rápidos, para previsão do funcionamento da aplicação, revelou-se de extrema importância, visto que contribuiu para detalhar a pouca informação inicial sobre o que era pretendido e serviu como um bom guia para a implementação da solução. Como referido, a informação inicial sobre o projecto era muito limitada, visto que só era conhecido o objectivo do projecto e uns requisitos muito rudimentares. Daí que, a opção de realizar os protótipos, permitiu definir o que era pretendido com clareza e detalhar os requisitos iniciais. Desta forma, o projecto ficou bem definido, agilizando todo o trabalho seguinte.

Posteriormente, no estudo das tecnologias que permitiam implementar a solução pretendida, conclui-se que apenas duas apresentavam as condições necessárias: Microsoft Silverlight e Adobe Flash. Qualquer uma das duas permitiria implementar com sucesso a aplicação, no entanto, o Silverlight foi a tecnologia escolhida pois apresentava ferramentas mais eficazes a nível de desenvolvimento, permitindo uma implementação mais célere do projecto. Tendo em conta o tempo limitado para o desenvolvimento do projecto, este factor era de extrema importância para ser possível conclui-lo com sucesso.

A implementação da solução foi realizada com sucesso, visto que durou metade do tempo inicialmente previsto e todas as funcionalidades principais foram implementadas.

Nesta fase, foi decidido implementar a base de dados da aplicação em XML, visto não haver acesso à base de dados actual do Diagrammer. O uso de XML apresentou-se como uma boa opção, visto o Silverlight possuir bons meios de leitura e manipulação de XML.

Na implementação das classes da aplicação foram detectadas algumas limitações da tecnologia, mais propriamente no que respeita às animações. Verificou-se que as animações da imagem humana eram interrompidas quando a aplicação processava outros dados, nomeadamente, por ocasião da abertura de nós da árvore de diagramas. Neste caso, concluiu-se que a solução seria processar a abertura da árvore após a conclusão da animação. Mas, aqui surgiu outro problema, ao verificar-se que o evento de conclusão de uma animação era accionado cedo demais e não na altura real da conclusão da animação. O problema foi ultrapassado com a inclusão de uma animação a funcionar como um temporizador que acciona a abertura da árvore meio segundo após o fim da animação da imagem, garantindo que esta acção não se processa cedo de mais. Esta foi, no entanto, a única limitação relevante encontrada no Silverlight, pois a implementação de tudo o resto ocorreu sem grandes dificuldades.

No final, o resultado do projecto é bastante positivo. A aplicação desenvolvida oferece um grande nível de interactividade que contribuirá para uma agradável experiência de utilização. São disponibilizadas diversas formas de selecção do diagrama que se pretende abrir no Diagrammer, pelo que um utilizador poderá escolher aquela com que se sentir mais confortável para encontrar, de forma fácil e rápida, o diagrama que pretende, como era objectivo deste projecto. Esta aplicação apresenta-se assim como um elemento diferenciador do Diagrammer em relação a outras aplicações de anotações sobre diagramas médicos, sendo por isso uma mais valia para a Siemens SA.

5.1 Trabalho Futuro

Em relação à lista de requisitos definida para o projecto no âmbito do estágio realizado, apenas não foi concluído o configurador gráfico dos mapas que definem zonas de selecção da imagem humana, pelo que, o desenvolvimento deste configurador será o próximo passo na evolução deste projecto.

Já fora do definido para o período do estágio, será necessário ainda desenvolver o módulo de autenticação do sistema e colocar o Outliner a usar a mesma base de dados do Diagrammer. Esta última tarefa não foi definida para o estágio visto que não existiu possibilidade de acesso à aplicação Diagrammer, tendo existido apenas a possibilidade de experimentar a aplicação numa máquina acedida remotamente.

Além disto, pretende-se que este projecto evolua para uma realidade 3D, ou seja, que a imagem de representação do corpo humano e sua manipulação venha a ser realizada de forma tridimensional. Para isso, é necessário que a tecnologia Silverlight evolua nesse sentido, no qual já deu os primeiros passos, como foi referido neste relatório.

6 Bibliografia e Referências

[1] Anónimo, “Wellsoft Image Capture”, http://www.wellsoft.com/product_overview/features/image_capture.php (consultado em 2008-03-24)

[2] Anónimo, “Alert EDIS”, http://www.alert.pt (consultado em 2008-03-24)

[3] Anónimo, “Medical Imaging Application LLC”, http://www.mia-llc.com/products/cartilage.htm (consultado em 2008-03-24)

[4] Anónimo, “Leadtools Medical Annotations SDK”, http://www.leadtools.com/sdk/medical/Medical-Addon-Annotations.htm (consultado em 2008-03-24)

[5] Anónimo, “Volume4D Pro”, http://www.sciencegl.com/volume_4d/volume.html (consultado em 2008-03-24)

[6] Department of Veterans Affairs, “VistA Diagram Annotation Tool User & Setup Guide”, 2005, http://www.va.gov/vdl/documents/Clinical/Vista_Imaging_Sys/mag3_0_diag_annot_ug.pdf (consultado em 2008-03-24)

[7] Jennifer Simpson, Janna Balling, “Annot3D”, University of Utah, 2004, http://www.sci.utah.edu/~simpson/documentation/projects/annotation/webdocs/image_gal.html (consultado em 2008-03-24)

[8] Dorian Gorgan, Teodor Stefanut, Bogdan Gavrea, “Pen Based Graphical Annotation in Medical Education”, Technical University of Cluj-Napoca, Romania, http://users.utcluj.ro/~gorgan/res/cgis/itrace/Lesson_Medicine/index.htm (consultado em 2008-03-24)

[9] Anónimo, “ConceptDraw Medical: Medical and Healthcare software”, http://www.conceptdraw.com/en/solutions/healthcare/overview.php (consultado em 2008-03-24)

[10] Einar Ingebrigtsen, “3D in Silverlight 1.1 Alpha”, http://www.dolittle.com/blogs/einar/archive/2007/05/19/3d-in-silverlight-1-1-alpha.aspx (consultado em 2008-04-08)

[11] Jon Galloway, “Why Silverlight 2 Deep Zoom Really is Something New”, 2008, http://weblogs.asp.net/jgalloway/archive/2008/03/21/why-silverlight-2-deep-zoom-really-is-something-new.aspx (consultado em 2008-04-08)

[12] Bryan, “JavaFX in Perspective”, 2007, http://www.scruffles.net/blog/2007/05/javafx-in-perspective.html (consultado em 2008-04-08)

[13] Anónimo, “Introducing Silverlight 1.1”, http://files.cnblogs.com/renji/introducing%20silverlight%201.1.pdf (consultado em 2008-04-08)

[14] Ric Smith, “AJAX, Flash, Silverlight, or JavaFX...Must we choose?”, 2007, http://www.web2journal.com/read/430983.htm (consultado em 2008-04-08)

[15] Grant Skinner, “A Flash of Silverlight? My Analysis”, 2007, http://www.gskinner.com/blog/archives/2007/05/a_flash_of_silv.html (consultado em 2008-04-08)

[16] Jeff Woelker, “Adobe Flash versus Microsoft Silverlight Comparison - Flash Wins for the Short Term”, 2007, http://www.jeffwoelker.com/2007/05/18/adobe-flash-versus-microsoft-silverlight-comparison-flash-wins-for-the-short-term/ (consultado em 2008-04-08)

[17] George P. Alexander Jr., “Is Microsoft Silverlight THE Flash Killer?”, 2007, http://blogs.ittoolbox.com/c/programming/archives/is-microsoft-silverlight-the-flash-killer-15993 (consultado em 2008-04-08)

[18] http://www.digiwebencoding.com/blog/?p=71 (consultado em 2008-04-08)

[19] Alex Iskold, “RIA: What is it good for?”, 2007, http://www.readwriteweb.com/archives/ria_what_is_it_good_for.php (consultado em 2008-04-08)

[20] Michael Schwarz, “Rich Media Platform Comparison”, 2007, http://weblogs.asp.net/mschwarz/archive/2007/04/19/rich-media-platform-comparison.aspx (consultado em 2008-04-08)

[21] Anónimo, “Silverlight vs. Flash – An Analysis Report”, 2007, http://silverlight.net/forums/p/3015/8462.aspx (consultado em 2008-04-08)

[22] Jesse Ezell, “Silverlight vs. Flash: The Developer Story”, 2007, http://weblogs.asp.net/jezell/archive/2007/05/03/silverlight-vs-flash-the-developer-story.aspx (consultado em 2008-04-08)

[23] Chad Z. Hower, “SilverLight, Flash, and SVG”, http://www.kudzuworld.com/blogs/Tech/20070502.EN.aspx (consultado em 2008-04-08)

[24] Anónimo, “A perspective on Apollo vs Silverlight vs JavaFX vs Flash/Flex”, 2007, http://ttlnews.blogspot.com/2007/05/test_22.html (consultado em 2008-04-08)

[25] Anónimo, “Microsoft Silverlight rivals Flash, AJAX”, 2008, http://www.techq.com/news/33599.shtml?request_locale=en (consultado em 2008-04-08)

ANEXO A: Plano de Trabalhos

Com o início do projecto, foi definido o plano de trabalhos para a sua realização, de acordo com os objectivos do projecto e das restrições temporais definidas pelo MIEIC.

Esse plano é apresentado na tabela abaixo:

ID	Tarefa	*Deliverable*	Prazo
1	Documento de estudo das opções de implementação com protótipos	Página *Web* protegida com *Password* Documento de protótipos	7-Mar
2	Estudo do estado da arte onde são apresentados os melhores produtos nesta área, e comparados com as funcionalidades especificadas para este projecto	Elaborar documento do estado da arte	18-Mar
3	Estudo das técnicas de computação gráfica com coordenadas normalizadas e imagens vectoriais em 2D		25-Mar
4	Estudo e definição das ferramentas de desenvolvimento adequadas ao cumprimento dos requisitos nos prazos apresentados	Elaborar documento onde são apresentados benefícios/desvantagens do estudo realizado sobre as ferramentas de desenvolvimento. Deve responder "qual a melhor tecnologia para este projecto?"	3-Abril
5	Implementação segundo Scrum (conforme tabela abaixo)	Iteração do *Product backlog* e *Sprint backlogs*	2-Jun
6	Manual de programação, com descrição dos módulos funcionais com diagrama de classes do projecto e diagramas gerais	Manual de programação	6-Jun
7	Manual de instalação do programa, incluindo os requisitos de hardware e software	Manual de instalação	13-Jun
8	Redacção do Relatório de Projecto	Relatório de final de projecto	4-Jul
9	Poster	Poster	11-Jul