
Universidade de Aveiro
2005

Departamento de Electrónica e Telecomunicações

Óscar Narciso Mortágua Pereira

abcNet – alfabetização na Net

Universidade de Aveiro
2005

Departamento de Electrónica e Telecomunicações

Óscar Narciso Mortágua Pereira

abcNet – alfabetização na Net

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Joaquim Manuel Henriques de Sousa Pinto, Professor Auxiliar do Departamento de Engenharia Electrónica e Telecomunicações da Universidade de Aveiro e pelo Professor Doutor António José Batel Anjo, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro.

o júri

presidente

Prof. Dr. Joaquim Arnaldo Carvalho Martins
professor Catedrático da Universidade de Aveiro

Prof. Dr. Gabriel de Sousa Torcato David
professor Associado da Faculdade de Engenahria da Universidade do Porto

Prof. Dr. Joaquim Manuel Henriques de Sousa Pinto
professor Auxiliar da Universidade de Aveiro

Prof. Dr. António José Batel Anjo
professor Auxiliar da Universidade de Aveiro

agradecimentos

Quero agradecer a todos aqueles que contribuíram e que me apoiaram neste meu objectivo de obtenção de título de Mestre. Especial agradecimento à minha esposa que pela sua disponibilização, apoio e importantes contributos marcou indelevelmente não só este período da minha vida mas também o trabalho realizado. Não posso deixar também de agradecer aos meus orientadores em especial e a todos aqueles que no contacto dia a dia me foram ajudando na persecução dos meus objectivos, nomeadamente o Professor Dr. João Carlos David Vieira, o Mestre Pedro Manuel Correia Almeida e o Eng.º Cláudio Teixeira. Especial destaque deve ser dado às pessoas que de uma forma desinteressada permitiram o ensaio de campo de abcNet que se realizou na Escola Primária de Mira. De entre essas pessoas destaco o professor João Luís Dias, director da escola, o professor Luís Lourenço, professor do 1º ano, e finalmente os alunos como elementos essenciais e activos do ensaio.

resumo

A dissertação apresenta o desenvolvimento de uma plataforma informática baseada na *Web* vocacionada para a alfabetização. Apesar de a alfabetização ser o objectivo último de abcNet, a estratégia seguida para a arquitectura interna de abcNet merece especial relevo. abcNet apresenta uma arquitectura que deposita no professor a responsabilidade da criação dos conteúdos necessários à alfabetização. É o professor, por configuração de modelos, que em cada instante e em cada caso tem o poder de decidir qual a metodologia e quais os conteúdos que melhor se aplicam. Neste contexto, apesar de o princípio pedagógico não ser novidade, a implementação realizada através de abcNet é concerteza um avanço significativo nas TIC.

abstract

The dissertation presents the development of a computer science platform based on the Web which aims at teaching how to read and write. Although the teaching how to read and write is the aim goal of abcNet, the strategy followed for the internal architecture of abcNet deserves special relief. abcNet presents an architecture that deposits in the professor the responsibility of the creation of the necessary contents to the teaching process. It is the teacher, by the configuration process of models, that in each instant and in each case has the power to decide which the methodology and which the contents are better to be applied. In this context, although the pedagogical principle not to be newness, the implementation carried through abcNet is surely a significant advance in the CIT.

Índice

Índice de Figuras

Índice de Tabelas

I. Apresentação

1 Introdução

As tecnologias de informação e de comunicação (TIC) criaram excelentes oportunidades quer para o desenvolvimento quer para a realização humana, acentuando também o desafio de as tornar acessíveis a todos, principalmente àqueles que por razões diversas, maiores dificuldades revelam na sua utilização. Se às vulgares dificuldades inerentes à utilização das TIC, adicionarmos dificuldades derivadas do analfabetismo, iliteracia ou de necessidades educativas especiais, teremos um quadro cuja evolução, numa primeira abordagem, nos precipita rapidamente na insolubilidade. Este é um dos maiores desafios colocados às TIC, que tem merecido uma atenção especial, não só da comunidade científica, mas também da comunidade em geral e das organizações governamentais e institucionais em particular. abcNet [OSCAR-eSOC] é uma aplicação baseada na *Web*, tendo como objectivo principal colocar à disposição de cada um, conteúdos vocacionados para a alfabetização, com recurso a mecanismos comuns de interacção com os utilizadores, tais como botões, hiperligações, efeitos visuais, texto, imagens, áudio, etc. Através das TIC, abcNet procura eliminar uma das principais barreiras à utilização das mesmas TIC: o analfabetismo.

A metodologia utilizada para o processo de alfabetização é baseada no método criado e defendido por Paulo Freire [CRB]. Paulo Freire[IPF-BIOG] nasceu a 19 de Setembro de 1921 no Recife, capital do estado de Pernambuco no nordeste brasileiro, um dos estados mais pobres de todo o país. Apesar de ter crescido no seio de uma família da classe média e de se ter licenciado em direito, Paulo Freire desde cedo se interessou pelas questões relacionadas com a educação das classes sociais mais desfavorecidas.

Paulo Freire deixou uma vasta obra escrita, sendo considerado por muitos se não o mais pelo menos um dos mais influentes pensadores sobre a educação no final do século XX [INFED]. O livro *Pedagogia do Oprimido* [PF-PO] é considerado o livro mais significativo de toda a sua extensa obra.

1.1 Objectivos

abcNet tem por objectivo o desenvolvimento de uma aplicação baseada na Web vocacionada para a alfabetização, tendo como base a metodologia apresentada por Paulo Freire [CRB]. No livro "*Educação como Prática de Liberdade*"[PF-ECPL], Paulo Freire apresenta as cinco etapas a seguir na implementação do seu método:

1) selecção do vocabulário mais utilizado pelos alunos;

2) de entre todo o vocabulário, selecção das palavras a utilizar no método, denominadas de palavras geradoras;

3) recriação de cenários reais que condensem situações da vida real dos alunos;

4) planeamento da acção de alfabetização;

5) criação de modelos de alfabetização a partir das palavras.

abcNet apresenta uma estrutura que se adapta às etapas anteriormente enunciadas, se complementado com algumas intervenções dos professores, nomeadamente na selecção das palavras a utilizar para a alfabetização. Se o estudo do vocabulário mais utilizado e a selecção das palavras a utilizar não são tarefas a executar por abcNet, a disponibilização de uma plataforma que permita a utilização das palavras seleccionadas pelo professor já se apresenta como sendo um dos requisitos fundamentais de abcNet. Esta colaboração entre professores e abcNet é uma constante na definição e caracterização dos mecanismos de implementação dos processos de alfabetização.

Os utilizadores finais de abcNet são professores e alunos, alunos estes que em termos de formação se encontram numa fase ainda muito inicial, não tendo tido muitas das vezes, qualquer contacto com as TIC. Esta situação leva a que abcNet deva prestar especial atenção aos mecanismos de interacção implementados. Tendo também em consideração a previsível dispersão geográfica dos seus utilizadores, é requisito essencial que abcNet seja uma aplicação baseada na *Internet* e cujos recursos necessários à sua utilização sejam: um computador pessoal com acesso à *Internet* e um qualquer explorador de *Internet*.

1.2 Motivação

Desde a minha formatura em 1983 até 2000 estive sempre ligado ao mundo empresarial privado, onde o fruto do trabalho realizado era para ser aplicado em algum local, em algum instante. Esta postura tem-me acompanhado e é ela que me conduz, sempre que possível, para intervenções cujos resultados se encontrem em consonância. A possibilidade de desenvolvimento de uma aplicação vocacionada para alfabetização com características inovadoras, é concerteza um desafio aliciante. O desafio comporta diversas vertentes que vão desde as associadas a questões tecnológicas às associadas a questões pedagógicas. As questões tecnológicas conduzem-me para um mundo no qual já me encontro inserido e que necessita de formação complementar de modo a atingir os objectivos propostos. Já as questões pedagógicas são concerteza uma completa novidade, não só pelo tema em si mas também pela distância científica a que normalmente se encontra da engenharia.

Apesar de o ensino ter sido algo que sempre me fascinou, nunca tinha tido a oportunidade de reflectir em detalhe sobre questões supostamente tão básicas como aprender a ler e a escrever. Digo supostamente porque uma atenção mais cuidada leva-nos a afirmar que saber ler e escrever são tarefas de exigências elevadas em termos intelectuais e psico-motores, como se verá no capítulo 2.

1.3 Organização da dissertação

A dissertação encontra-se organizada em 6 grandes blocos:

I. Apresentação: corresponde ao bloco corrente.

II. Enquadramento: neste bloco são apresentados aspectos pedagógicos associados à alfabetização, os requisitos a satisfazer por abcNet e também a apresentação do enquadramento tecnológico no qual abcNet se integra.

III. Aplicação: abNet – alfabetização na Net: neste bloco é apresentada a aplicação abcNet do ponto de vista do desenvolvimento da aplicação e também da sua utilização.

IV. Anexos

V. Acrónimos: acrónimos utilizados

VI. Bibliografia e Referências

.

II. Enquadramento

2 Práticas e Métodos Pedagógicos

Muita da pesquisa em curso centra-se no entendimento de como se processa o desenvolvimento das competências que adquirimos para ler. O processo é muito elaborado e engloba [PE-DISLX, pág 4]:

(...)

- *reconhecimento e discriminação de símbolos gráficos(grafemas);*
- *a sua associação aos correspondentes símbolos auditivos(fonemas);*
- *a análise e síntese auditiva e visual dos vários elementos constitutivos da palavra e da palavra como um todo;*
- *a constante combinação de ambas (análise e síntese);*
- *a compreensão ou atribuição de significado às palavras (referida à experiência anterior).*

Nos parágrafos seguintes apresentam-se algumas abordagens associadas aos processos de alfabetização.

2.1 Abordagem estrutural

A abordagem estrutural preocupa-se com a metodologia utilizada para a aprendizagem da leitura de palavras na vertente da forma como se agrupam e se desagrupam os diversos elementos de uma palavra. Os elementos das palavras tanto podem ser letras individualizadas como um conjunto de letras ou um misto das duas situações. A abordagem estrutural ao processo de alfabetização pode seguir duas vias distintas: sintética e analítica [CRE-MSA]. Para além destas duas abordagens, uma terceira pode ser definida e classificada como mista.

Método Analítico

O método analítico parte sempre de uma palavra dando de seguida enfoque a padrões existentes na própria palavra. Numa primeira fase, pretende-se apresentar as letras, uma de cada vez. Inicialmente com recurso à primeira letra de palavras seleccionadas e de seguida a letras situadas quer no meio quer no final de palavras também seleccionadas. Posteriormente, sempre com base em palavras, é dada ênfase a grupos crescentes em número de letras situados quer no princípio quer no final de palavras.

Método Sintético

O método sintético consiste fundamentalmente no estabelecimento de uma correspondência entre o oral e o escrito, partindo sempre da parte para o todo. A base de partida materializa-se num conjunto de

letras cujas pronúncias são ensinadas aos alunos. Seguidamente as letras são agrupadas para construir sílabas e palavras, dando-se realce quer à leitura do agrupamento quer ao som característico de cada letra ou conjunto de letras.

Num estudo elaborado no Reino Unido[RHONA] , foi constatado uma supremacia significativa da abordagem sintética relativamente à analítica.

Mista

Métodos mistos baseiam-se na utilização dos dois métodos anteriormente apresentados existindo variadíssimas combinações possíveis.

2.2 Abordagem fonética

Do ponto de vista fonético, a alfabetização tem tido e pode ter várias abordagens. A capacidade de leitura depende da associação de 3 factores [HENRY]: S - o significado associado à palavra; L - capacidade de leitura da palavra escrita; P - forma oral ou pronúncia. O aluno mesmo antes de saber ler, consegue estabelecer a ponte entre os factores P e S, faltando-lhe o estabelecimento da relação entre o factor L e os factores P e S. Desta observação foram desenvolvidos uma série de métodos denominados de métodos fonéticos [HENRY]. A abordagem fonética trata de questões associadas à estratégia a seguir na divisão das palavras em blocos fonéticos de modo a facilitar a aprendizagem da sua leitura. A abordagem ao processo de alfabetização na vertente da fonética, evoluiu ao longo dos anos, dando origem a métodos muito diferentes. De entre estes métodos destacam-se: alfabético, fonético, alfabeto estendido, silábico e o fonogramático.

Alfabético

Os alunos aprendem a pronunciar as letras do alfabeto. A partir da pronúncia alfabética de cada letra o aluno deveria ser capaz de ler as palavras. Este método dificultava a aprendizagem, pois a pronúncia alfabética normalmente pouco tem de comum com a pronúncia no contexto. Veja-se o caso da palavra *bola* cujas divisão em pronúncias alfabéticas origina *bê-ó-éle-á*.

Fonético

Com o objectivo de colmatar as deficiências do método alfabético, foi introduzida uma variante que consiste na leitura não da pronúncia alfabética das letras mas sim na leitura da pronúncia do fonema associado à letra. Através de uma selecção criteriosa de palavras, este método tem tido aplicação com elevado sucesso. As dificuldades levantam-se quando surgem letras mudas, grupos de letras com

pronúncia específica, letras ou grupos de letras com mais de uma pronúncia, letras cuja leitura depende do contexto, etc. Como exemplos de palavras de difícil aplicação do método, indicam-se:

- letras mudas: *pel<u>e</u>* , *escadot<u>e</u>* , *clarinet<u>e</u>* , ... ;
- grupos de letras com pronúncia específica: *cami<u>nh</u>*ar , *gal<u>lh</u>o* , *cora<u>ção</u>* , ... ;
- letras com mais de uma pronúncia: *c<u>o</u>la* , *r<u>o</u>lha* , *gal<u>o</u>* , ... ;
- função do contexto: *<u>c</u>ola* , *<u>c</u>ela, ...* .

Alfabeto estendido

Este método definiu um conjunto adicional de símbolos que complementavam os existentes de modo a identificar univocamente a leitura de cada elemento de uma palavra. Os símbolos adicionais tiveram duas versões distintas: uma que aumentava o número de letras do alfabeto; outra que por agregação de símbolos específicos a cada letra ou grupo de letras lhe atribuía uma pronúncia única. De entre as várias fragilidades deste método destaca-se a não coincidência entre a representação das palavras no processo de aprendizagem e a sua representação no mundo real devido à utilização de simbologia específica.

Silábico

De modo a contornar as dificuldades apresentadas pelo método anterior, houve como que um regresso à utilização das sílabas como elemento basilar da aprendizagem. Neste método, as sílabas são apresentadas individualmente e posteriormente justapostas para formar palavras. As fragilidades deste método foram apontadas na obrigatoriedade da divisão das palavras em sílabas, divisão essa que foi considerada como sendo não natural, ou seja, é uma convenção. Apesar das fragilidades apontadas, este método continua a ter grande sucesso e utilização nas escolas portuguesas do 1º ciclo.

Fonogramático

Alguns defendem que a divisão das palavras em sílabas é uma convenção e não um mecanismo natural da linguagem. No método fonogramático, ao contrário do silábico, a divisão das palavras é baseada nos sons que a constituem, como por exemplo: p-*ão*, c-*ão*, n-*ão*, m-*ão*, etc. Algumas das vantagens atribuídas ao método fonogramático são as seguintes:

- a divisão das palavras é baseada nos seus sons;
- a divisão pode ser ajustada ao perfil do aluno;
- são realçados grupos de pronúncias que podem ser repetidas em outras palavras;

- as palavras de leitura irregular podem ser facilmente incorporadas no método.

As abordagens fonética, silábica e fonogramática, no que diz respeito à sua integração na abordagem estrutural, aproveita em cada instante aquilo que melhor se adapta à sua estratégia, estratégia que pode variar de situação para situação. No início pode seguir uma abordagem sintética evoluindo posteriormente para uma abordagem analítica. O aluno começa por aprender as letras passando-se de seguida para a apresentação e leitura de palavras; estas são divididas, permitindo, inicialmente, a leitura de cada divisão e posteriormente a sua integração para a leitura global da palavra. Por sua vez, cada divisão pode ainda ser analisada em detalhe maior, por exemplo ao nível de cada letra, podendo-se também agregar as letras para se obter a leitura de cada divisão.

2.3 Método Paulo Freire

"*Ler não é passear por cima das palavras. Ler é ter uma compreensão profunda do lido e também da estética do lido. Se este país (Brasil) levasse a sério o exercício da leitura da palavra associada à leitura do mundo, com todas as suas implicações de ordem estética e boniteza e também de liberdade de criação, eu acho que ensinar a ler e escrever, numa perspectiva como essa, faz parte da pedagogia da democracia*" [IPF-HOME]. Destas palavras conclui-se que para Paulo Freire, saber ler não se resume ao mero exercício da leitura, mas sim também da tomada de consciência e compreensão global daquilo que se lê, complementado de uma capacidade crítica do mundo circundante. Assim, o processo de alfabetização é visto mais como um meio do que como um fim em si mesmo, sendo o objectivo principal muito mais ambicioso, a que Paulo Freire denominava de concientização [PF-PA, pág 60] . Só através da concientização é dada a possibilidade plena aos educandos de assumirem a sua própria liberdade. Na continuidade do raciocínio apresentado, Paulo Freire afirma que aprender não se resume a um acto de assimilação de conhecimentos, tendo afirmado no seu livro Pedagogia da Autonomia [PF-PA, pág. 77], que "(...) *aprender é uma aventura criadora, algo, por isso mesmo, muito mais rico do que meramente repetir a lição dada. Aprender para nós é construir, reconstruir, constatar para mudar, o que não se faz sem abertura ao risco e à aventura do espírito*".

A acção de Paulo Freire centrou-se essencialmente na alfabetização de adultos, os quais, pelas suas condições sociais e económicas, não se libertaram do analfabetismo. A intervenção de Paulo Freire ultrapassa em muito o objectivo da aprendizagem da leitura e da escrita. Da sua pedagogia, vamos aproveitar apenas o método por ele utilizado no processo de alfabetização e que é conhecido pelo Método *Paulo Freire* – MPF. Para o presente trabalho, o entendimento do método Paulo Freire, complementado com as algumas anotações, é suficiente para o desenvolvimento de uma plataforma vocacionada para a alfabetização.

2.3.1 Descrição do Método

Nos processos tradicionais de alfabetização, o material utilizado, nomeadamente os manuais escolares, é fornecido pelas editoras que seguem normas e programas estabelecidos pelos respectivos Ministérios de Educação. O professor é responsável por transmitir os conhecimentos e os alunos têm a responsabilidade de assimilar os conhecimentos transmitidos. É o princípio do transmitir o "saber-de-quem-sabe" no suposto "vazio-de-quem-nada-sabe" [CRB, pág 21]. Paulo Freire pensou num método de educação baseado no diálogo professor-aluno, onde cada um traz o seu próprio contributo para o processo de educação. Assim, *"(...) ninguém educa ninguém -, ninguém se educa a si mesmo – os homens se educam entre si (...)"* [PF-PO, pág. 79]. Naquele contexto, o livro tradicional é um elemento pré-fabricado, desajustado e pouco flexível, não possibilitando uma interacção construtiva e participativa dos principais intervenientes: os alunos. O método Paulo Freire transporta em si mesmo esta preocupação que pode ser constatada na sua própria descrição. O método Paulo Freire engloba um conjunto de cinco etapas [PF-ECPL, pág. 112-115] que a seguir se resumem.

Definição do Universo das Palavras Geradoras

As palavras a utilizar no processo de alfabetização devem ser seleccionadas de entre aquelas que de alguma forma pertencem ao mundo real de vivência e de interesse dos alunos a alfabetizar. Nenhuma palavra pode ser vista como elemento neutro na educação, pois uma palavra dá origem a um tema e o tema pode condicionar o pensamento. Do mundo, da vivência e da experiência, resumidamente, a partir dos interesses dos alunos deve emanar aquele que é definido como sendo o *Universo das Palavras Geradoras*.

Selecção das Palavras Geradoras

Do universo das palavras geradoras deve emanar o conjunto das palavras geradoras a utilizar no método. É a partir de cada uma destas palavras que o método Paulo Freire se desenvolve, pois estas são a menor unidade de pesquisa e as sílabas das palavras são a menor unidade do método. Cada palavra serve dois objectivos distintos mas complementares no método Paulo Freire: a) instrumento de leitura da língua; b) instrumento de leitura da realidade social onde a língua existe. De modo a satisfazer cada um dos objectivos enunciados, cada palavra geradora seleccionada deve ter em consideração os seguintes tópicos:

- riqueza fonética;
- dificuldades fonéticas da palavra e da língua;
- densidade e importância contextual para os alunos.

Uma boa palavra geradora é aquela que congrega o máximo dos 3 tópicos apontados [CRB, pág 31]. A selecção do conjunto das palavras geradoras é uma etapa fundamental e muito exigente em termos de esforço a ser desenvolvido pelo professor.

Recriação de cenários reais

Os cenários reais devem traduzir situações que abrem perspectivas para a análise de problemáticas de vertentes tão diversas quanto as associadas às vivências e interesses dos próprios alunos, às associadas a questões locais, regionais, nacionais ou internacionais, etc.

Planeamento da acção

Esta etapa é dirigida ao planeamento da acção, não de uma forma rígida e imutável, mas de forma a que o próprio planeamento seja uma entidade dinâmica que se ajusta à evolução da acção de alfabetização.

Criação de modelos de alfabetização

No método Paulo Freire, os modelos de alfabetização partem sempre de uma palavra geradora. A palavra geradora é o elemento a partir do qual se desenvolve o processo de alfabetização. A palavra é a unidade fundamental [CRB, pág 30], dando origem a uma aprendizagem baseada nos seus elementos básicos: as sílabas. É através das sílabas que o processo de alfabetização se desenvolve, quer pela divisão das palavras em sílabas, abordagem analítica, quer pelo agrupamento de sílabas em palavras, abordagem sintética. Nesta fase, pretende-se que os alunos construam palavras a partir de um conjunto de sílabas que são obtidas a partir das sílabas de uma palavra geradora. Ao grupo de sílabas utilizadas para a construção de novas palavras denomina-se ficha da descoberta [PF-ECPL, pág 116]. Resumidamente, o método Paulo Freire propõe o seguinte faseamento [PF-ECPL, pág 115-119]:

- eleição de uma palavra geradora e de uma imagem a ela associada;
- apresentação das sílabas da palavra geradora;
- apresentação das sílabas família de cada sílaba da palavra geradora;
- construção de novas palavras a partir das sílabas família.

Sílabas família de uma sílaba, são as sílabas que se obtêm a partir da sílaba original, pela substituição da vogal por cada uma das quatro restantes vogais. Assim, as sílabas família da sílaba *me* são *ma*, *me*, *mi*, *mo* e *mu*. Genericamente pode-se afirmar que as sílabas família de $m\alpha$, $\alpha \in \{a,e,i,o,u\}$ é o conjunto $\mathcal{F}_{m\alpha}=\{ma, me, mi, mo, mu\}$.

De modo a ter uma visão global, vamos apresentar um exemplo de aplicação do método Paulo Freire a partir da palavra geradora *parede*, cujo desenvolvimento se encontra apresentado na Tabela 1. As palavras apresentadas como construídas, representam apenas algumas das possibilidades existentes.

Descrição	**palavra / sílabas**
palavra geradora	Parede
imagem geradora	apresentar imagem de uma parede
sílabas	pa – re – de
sílabas família	pa – pe – pi – po – pu, $\mathcal{F}_{p\alpha}$ ra – re – ri – ro – ru, $\mathcal{F}_{r\alpha}$ da – de – di – do – du, $\mathcal{F}_{d\alpha}$
palavras construídas	dado, dedo, raro, papa, pipa, papo, ripa, para, rede, roda, rapado, dora, parado, ...

Tabela 1 – Exemplo de aplicação do método de alfabetização de Paulo Freire

2.3.2 Análise do Método Paulo Freire

A pedagogia de Paulo Freire ultrapassa em muito a questão relacionada com a alfabetização. Para Paulo Freire, a alfabetização é um passo necessário mas insuficiente. A leitura da palavra abrange um objectivo mais profundo que é a sua compreensão e contextualização sempre com o objectivo de reinterpretação do mundo. Neste contexto, a selecção das palavras a utilizar ocupa uma posição determinante nos objectivos a atingir. A pedagogia de Paulo Freire afasta-se significativamente da utilizada no programa Mobral [MOBRAL] e dos normalmente utilizados nas nossas escolas, cujos objectivos se centram na aquisição de competências para a leitura e escrita. Tradicionalmente, a alfabetização é um processo vocacionado apenas para a aprendizagem da leitura e da escrita e está assente em regras definidas em etapas anteriores ao próprio processo de alfabetização.

Daqui resultam duas diferenças essenciais e que são pontos basilares no método Paulo Freire:

- a alfabetização deve ser planeada para cada grupo populacional;
- a leitura e a escrita são objectivos essenciais mas não suficientes.

Em termos de abordagem estrutural, o método Paulo Freire pode ser considerado um método misto pois recorre quer a uma abordagem analítica quer a uma abordagem sintética, aproveitando em cada instante aquela que melhor se ajusta às necessidades:

- leitura da palavra como um todo;
- divisão da palavra em sílabas – analítico;
- agrupamento de sílabas em palavras – sintético.

Na abordagem fonética, o método Paulo Freire é essencialmente silábico.

2.3.3 Campos de aplicação

Paulo Freire aplicou a sua pedagogia essencialmente na alfabetização de adultos. Hoje em dia é reconhecido que a sua pedagogia é aplicável não só na alfabetização de adultos mas também de crianças, podendo inclusivamente cobrir áreas não afectas à alfabetização e tão distintas quanto o jornalismo [EDU].

Ao nível dos meios a utilizar nas acções de alfabetização, nada justifica qualquer tipo de restrição ou constrangimento, inclusivamente Paulo Freire recorreu a elementos multimédia [SONIA] nas suas acções de alfabetização. Assim, nos dias que correm, as TIC podem e devem ser encaradas como uma mais valia a incorporar no método Paulo Freire.

Apesar de Paulo Freire ter utilizado a sua pedagogia em sentido lato, pois ia para além da alfabetização, englobando não só a leitura mas também a compreensão e contextualização, nada impede que o método seja utilizado em sentido restrito, isto é, apenas com o objectivo último: a alfabetização. A sua utilização numa vertente mais abrangente é possível e depende apenas da perspectiva e objectivos definidos pelo professor.

2.4 Aplicação dos Métodos de Alfabetização

Neste ponto vamos analisar alguns métodos utilizados com base em suportes distintos: manuais escolares e portais na *Internet*.

2.4.1 Manuais Escolares

Nas escolas portuguesas, são utilizados simultaneamente vários métodos para a alfabetização, ficando ao critério do professor a utilização da abordagem que mais se ajustar às necessidades dos alunos. Dos métodos utilizados, muitos recorrem, na vertente estrutural quer à abordagem sintética quer à analítica e na vertente fonética recorrem quer à abordagem fonética quer à silábica. Vamos analisar, de uma forma muito sintética, alguns manuais escolares do 1º ano do 1º ciclo, vocacionados para a alfabetização.

Trampolim 1

"Trampolin 1" [PE-TRAMP] é um manual editado e adoptado no ano lectivo 2004/2005. São propostas várias abordagens à alfabetização das quais se dão alguns exemplos. Exemplos das páginas 25, 28, 33, 85, etc. dão especial ênfase à utilização de uma dada letra no contexto de várias palavras, como é o caso da letra *L* na página 33. Nesta situação está-se a aplicar uma abordagem analítica e fonética onde se realça uma letra de uma palavra, pronunciando-se a sua fonética e depois utilizando-a em palavras.

Exemplos das páginas 44, 50, 70, etc., dão especial ênfase à leitura de sílabas e a posterior construção de palavras a partir dessas mesmas sílabas. Nestes casos temos um método silábico que é analítico no estudo de cada uma das sílabas e é sintético no agrupamento de sílabas para a construção de palavras.

Palavra a Palavra – Método das 28 Palavras

"Palavra a Palavra" [PE-M28P] é um livro editado em 2003 e adoptado também no ano lectivo 2004/2005. Baseia-se essencialmente num método de alfabetização que na sua essência parte de uma palavra na qual se dá ênfase à primeira letra, passando de seguida à sua decomposição em sílabas e finalmente para a construção de novas palavras a partir de sílabas. Especial ênfase é dada também às sílabas família tal como se pode verificar nas páginas 81, 84, 87, etc. Em termos de métodos podemos dizer que é analítico quando:

- foca uma palavra e uma das letras em especial, com abordagem fonética;
- divide a palavra em sílabas, com abordagem silábica;

e que é sintético quando, a partir de sílabas, se constróem novas palavras, recorrendo a uma abordagem silábica.

Língua Portuguesa

"Língua Portuguesa" [CONST-LP] é um manual que esteve em vigor no ano lectivo 1999/2000. Apesar de terem decorrido 5 anos lectivos, não se notam diferenças significativas nas metodologias utilizadas para a aprendizagem da leitura. Exemplos das páginas 9, 11, 13, 15, 17, etc. centram-se essencialmente no destaque de uma dada letra numa forma isolada e também integrada em diversas palavras. Exemplos das páginas 23, 24, 25, 26, etc. centram-se em grupos de letras e a sua integração em diversas palavras. Exemplos das páginas 32, 33, 34, 35, etc. centram-se nas sílabas família e a sua integração em palavras. Apesar das metodologias serem idênticas a abordagem neste manual parece ser menos abrangente em termos da exploração das metodologias. Caso notório é o referente à divisão das palavras em sílabas e posterior construção de novas palavras a partir dessas mesmas sílabas.

2.4.2 Portais na Internet

Para análise dos portais da *Internet* vamos dividi-los em 2 grupos: os dedicados à língua portuguesa e os dedicados à língua inglesa.

2.4.2.1 Portais para a Língua Portuguesa

Infelizmente, não foram encontrados muitos portais dedicados à alfabetização e os que foram encontrados são muito pouco ambiciosos. Os portais aqui referidos são amostras de produtos comerciais pelo que se supõe que sejam versões parciais de produtos comerciais.

Sítio dos Miúdos

O "Sítio dos Miúdos" [PE-SITIO] é um portal dedicado às crianças que disponibiliza múltiplas actividades. Na vertente da alfabetização, é muito rudimentar limitando-se a apresentar um esquema, com som, que através da selecção feita pelo aluno, percorre cada uma das letras do alfabeto, apresentando seguidamente uma palavra que inclua a referida letra.

Clube Júnior

O "Clube Júnior" [TE-JUNR] é mais ambicioso que o Sítio dos Miúdos mas muito menos interactivo, não possuindo sequer qualquer som. Recorre a várias abordagens desde a apresentação das letras de uma forma isolada, à apresentação de palavras como resultado da justaposição de letras, à apresentação de palavras divididas em sílabas. O Clube Júnior, tal como se apresenta na *Internet*, não consegue atingir qualquer objectivo em termos de alfabetização, pois os alunos, pelo facto de não saberem ler, não conseguem apreender qualquer informação que se encontra disponível, pelo facto de toda ela se apresentar apenas sob a forma escrita.

2.4.2.2 Portais para a Língua Inglesa

Existem muitos portais em inglês com conteúdos vocacionados para a alfabetização. De entre os que analisámos, vamo-nos limitar a apresentar os dois que mais se destacam: Starfall e NINE.

Starfall

"Starfall" [STARFALL] é um portal de serviço público de suporte à alfabetização. O processo de alfabetização encontra-se organizado em quatro níveis:

- nível 1: organizado por letras do alfabeto; por cada letra pretende-se atingir um conjunto alargado de objectivos que são a leitura alfabética, a sua leitura fonética e finalmente a leitura de palavras que incluam a referida letra;
- nível 2: neste nível são utilizados três métodos distintos, em função dos objectivos a atingir:
  - método fonogramático: é apresentado um conjunto de letras base, que são utilizadas em todas as palavras a construir, a partir das quais se constróem palavras pela justaposição de letras que se encontram apresentadas; exemplo: com as letras base *ato* construir palavras pela justaposição das seguintes letras *f*, *m*, *r*, *pr*, etc.;
  - método fonético: leitura de palavras pela justaposição das fonéticas de cada letra;
  - silábico: leitura de palavras pela justaposição da leitura de cada uma das suas sílabas;
- nível 3: leitura de frases através da justaposição da leitura de cada uma das suas palavras;
- nível 4: leitura de histórias através da justaposição da leitura de cada uma das suas palavras.

NINE – Nortwest Instructional 'N Educational Enterprises

"The Phonogram Page" [NINE] é um portal da NINE que disponibiliza conteúdos de alfabetização baseados no método fonogramático, sendo apresentados grupos de letras e as suas possíveis leituras.

2.5 Conclusões

Não é comum adoptar-se uma metodologia única e rígida para o processo de alfabetização. Em função quer do contexto quer dos objectivos a atingir em cada instante, a alfabetização socorre-se dos diversos métodos disponíveis. O método Paulo Freire é um método essencialmente silábico que parte de uma abordagem analítica para se centrar, na sua fase mais criativa, numa abordagem sintética. Vários métodos de aprendizagem podem ser adoptados para além do utilizado por Paulo Freire, mantendo sempre como pilar a possibilidade da alfabetização centrada no aluno.

Da análise efectuada aos manuais escolares do 1º ano do 1º ciclo, constata-se a utilização simultânea e sistemática de vários métodos. Mesmo no caso do manual do método das 28 palavras, cuja principal base é o método silábico, o recurso a outros métodos é também constante.

Pela pesquisa efectuada, a oferta de conteúdos para alfabetização na *Internet* ainda é muito escassa. A abundância da oferta de conteúdos em inglês é notória, mas muita dela ainda muito pouco ambiciosa. A oferta de conteúdos em português é praticamente inexistente.

Pode-se afirmar que os métodos são aplicados em função de objectivos específicos a atingir e também em função do público a que se destinam.

O recurso às TIC é uma mais valia no processo de alfabetização, permitindo não só a disponibilização de conteúdos idênticos aos proporcionados pelos manuais como também a disponibilização de mecanismos multimédia de interacção com os alunos que são concerteza bem mais apelativos.

Se a exigência de plataformas informáticas pressupõe que as TIC apresentam um custo adicional associado ao investimento inicial, também não pode ser nem ignorada nem menosprezada a mais valia subjacente à sua utilização. Na vertente associada a actualização dos conteúdos, o recurso às TIC pode revelar-se mais económico que a tradicional compra de novos manuais.

3 Definição dos Requisitos para a Criação de uma Aplicação de Alfabetização Baseada na Internet

Pretende-se desenvolver uma aplicação vocacionada para a alfabetização e que seja baseada nos fundamentos do método Paulo Freire, nomeadamente no que coloca especial enfoque no aluno como elemento central do processo de alfabetização. A aplicação é denominada de abcNet. Instrumentalmente, abcNet deve ser baseada na *Internet* permitindo deste modo um acesso geográfico mais abrangente. A estratégia de utilização da *Internet* como suporte a abcNet permite uma gestão tecnológica e pedagógica centralizada o que evita a disseminação quer de recursos humanos quer de recursos tecnológicos, logo uma grande economia em termos financeiros para os seus utilizadores.

Vamos definir alguns conceitos e características adicionais tais como o que são elementos de alfabetização, o que são utilizadores de abcNet, o que são entidades / unidades e modelos de alfabetização e ainda tecer algumas considerações sobre a escalabilidade de abcNet. Adicionalmente definem-se algumas novas propriedades a incorporar no método Paulo Freire sem que tal introduza qualquer desvirtuação do mesmo.

3.1 Elementos de Alfabetização

Os elementos de alfabetização (EleAlf) englobam sílabas e palavras. abcNet deve ter a ambição de poder vir a ser utilizado em países onde se não fala o português mas sim outro qualquer idioma, pelo que os elementos de alfabetização devem estar armazenados num repositório e apresentarem uma organização que facilite a sua manipulação por parte dos professores durante o processo de configuração.

Sílabas

Sílaba é um grupo de letras, uma ou mais, que satisfaz os seguintes requisitos:

- uma sílaba tem uma só ortografia (SílOrt);
- uma sílaba tem uma ou mais pronúncias (SílPro).

A cada concretização de SílOrt e SílPro denomina-se se entidade de sílaba (SílEnt).

Tendo em consideração as possibilidades de agrupamento das sílabas, desde já se definem quatro grupos distintos de sílabas: sílabas família, sílabas homófonas, sílabas homógrafas e sílabas adjacentes. Estes grupos encontram-se caracterizados em 3.3.1.

Palavras

Uma palavra é um grupo de letras, uma ou mais, que satisfaz os seguintes requisitos:

- uma palavra só tem uma ortografia (PalOrt);
- uma palavra tem uma ou mais pronúncias (PalPro);
- uma palavra subdivide-se em SílEnt.

A cada concretização de PalOrt e PalPro denomina-se de entidade de palavra (PalEnt).

A implementação destas entidades baseada numa base dados permite uma selecção diversificada de informação, como por exemplo, a partir de uma palavra poder saber as sílabas família de cada uma das suas sílabas.

3.2 Acções de Formação

Uma acção de formação é caracterizada pelos utilizadores a ela associados (professores e alunos) e por um planeamento. A acção de formação é uma entidade lógica e não física, pois numa mesma sala podem estar a decorrer várias acções de formação em simultâneo, cada uma com os seus professores, com os seus alunos e com o seu planeamento.

3.2.1 Utilizadores

O acesso a abcNet deve ser personalizado. Para tal os seus utilizadores possuem um *login* constituído por um nome de utilizador e uma palavra chave. A cada utilizador deve estar associado um de três perfis: administrador, professor ou aluno.

O perfil associado a um administrador permite o acesso a um conjunto de informação associada à administração de abcNet e que se encontra vedado aos utilizadores dos outros perfis. Por motivos de segurança, o acesso às funções de administração devem apenas estar disponíveis através de uma aplicação que não seja baseada na *Web*.

O perfil associado a alunos permite o acesso à acção de formação em que o aluno se encontra inscrito. Em cada instante, um aluno apenas se pode encontrar inscrito numa única acção de formação. O aluno tem acesso à acção de formação por intermédio da utilização de um explorador da *Internet*.

O perfil associado a professor permite o acesso não só às acções de formação pelas quais é responsável, mas também à configuração das mesmas. Um professor pode, em cada instante, ser responsável por uma ou mais acções de formação. O professor tem acesso às acções de formação por intermédio da utilização de um explorador de *Internet*.

3.2.2 Planeamento

Um planeamento está sempre associado a uma acção de formação e consiste num conjunto de unidades de alfabetização (ver 3.4) que se encontram com determinada ordenação. O professor é quem define quais as unidades de alfabetização a utilizar e a ordem pela qual elas devem ser apresentadas aos alunos para execução.

3.3 Extensão do Método Paulo Freire

O método Paulo Freire apresenta um modelo de alfabetização e um esquema de desenvolvimento desse mesmo modelo. Vamos apresentar uma extensão do método sem que isso colida de alguma forma com a sua essência. A extensão do método assenta na apresentação de uma ficha de descoberta mais ampla e também na apresentação de novos modelos de alfabetização.

3.3.1 A ficha de descoberta

No método Paulo Freire, as sílabas representam, além das letras, a divisão elementar de cada palavra, sendo o elemento mínimo do método. O método parte sempre de uma palavra geradora que é subdividida nas suas sílabas. Destas, obtêm-se as sílabas família que são a matéria prima, ficha de descoberta, para construção de novas palavras. A introdução do conceito de sílabas família permite o alargamento da base de sílabas a partir da qual se constróem novas palavras, para além das sílabas da palavra geradora. Uma análise mais cuidada das sílabas revela a necessidade da definição de novos grupos de sílabas, que podem introduzir não só ainda maior flexibilidade na construção de novas palavras, como também abrir novas perspectivas em termos pedagógicos, logo também no processo de alfabetização. Definimos 3 grupos adicionais de sílabas, a saber: homófonas, homógrafas e adjacentes [OSCAR-eSOC].

Sílabas homófonas

Duas sílabas dizem-se homófonas quando têm a mesma pronúncia mas grafia distintas, como por exemplo *ça* e *ssa* em ca*ça* e ma*ssa*. A introdução deste grupo de sílabas vem permitir que os alunos possam constatar a existência de sílabas que apesar de terem a mesma pronúncia, possuem ortografias distintas.

Sílabas homógrafas

Duas sílabas dizem-se homógrafas quando têm a mesma ortografia mas pronúncias distintas, como por exemplo *ca* em *ca*sa e *ca*ma. A introdução deste grupo de sílabas vem permitir que os alunos possam constatar a existências de sílabas que partilham a mesma ortografia apesar de possuírem pronúncias distintas.

Sílabas adjacentes

Sílabas adjacentes são as que se obtêm pela adição ou remoção de acentos ou sinais de nasalação, como por exemplo *ma*, *má* e *mã*. A introdução deste grupo de sílabas vem permitir que os alunos possam constatar a existência de variâncias de ortografia de sílabas pelas simples utilização de símbolos que têm implicações ao nível da pronúncia.

A introdução destes três novos grupos, abre novas possibilidades em termos de construção de novas palavras a partir de uma palavra geradora. Para além da ficha de descoberta ser mais ampla, introduzem-se variantes que ampliam significativamente aspectos relacionados quer com a ortografia quer com a leitura, pois passam a estar explicitamente disponíveis sílabas diferentes com a mesma pronúncia, sílabas iguais com pronúncias diferentes e sílabas acentuadas e com sinais de nasalação. A combinação de todos estes elementos proporciona um ambiente de alfabetização mais ambicioso que o definido apenas pelas sílabas família.

Concretizando para o caso da sílaba *do* da palavra *dado*:

sílabas família:	da (*da*ta), de (*de*la), di (*di*tongo), do (*do*ca), du (*du*ende)
sílabas adjacentes:	dó (dócil)
sílabas homófonas:	do (da*do*), du (*du*ro)
sílabas homógrafas:	do (da*do*), (*do*ca *)*

A SílOrt *do* (da*do*) pertence a três dos quatro grupos definidos.

3.3.2 Novos Modelos de Alfabetização

Novos modelos podem ser derivados do modelo apresentado por Paulo Freire sem que tal desvirtue o fundamental do seu método. Um dos fundamentos do seu método exige que o ensino seja centrado no aluno. Assim, os modelos a criar devem ter sempre em consideração a possibilidade de ser o professor a definir os conteúdos a serem utilizados.

3.4 Entidades / Unidades e Modelos de Alfabetização

Entidade de alfabetização (EntAlf) é uma entidade lógica que define um objectivo pedagógico a atingir e os mecanismos disponibilizados para o efeito. Uma das EntAlf a definir é a que implementa o método Paulo Freire. A cada tipo de EntAlf está sempre associado um modelo de alfabetização (ModAlf) e uma unidade de alfabetização (UniAlf).

Um ModAlf é o mecanismo disponibilizado aos professores para configuração da EntAlf a que está associada. No processo de configuração, o professor pode, por exemplo, indicar que palavras, sílabas, etc. vão ser utilizadas pelos alunos para a construção de outras palavras. Os ModAlf são objecto de configuração, e por cada configuração é gerada uma UniAlf que por sua vez deve estar associada a uma acção de formação. Por acção de formação, um ModAlf pode gerar tantas UniAlf quantas as necessárias.

Uma UniAlf deriva da configuração de um ModAlf. Cada UniAlf deve ser objecto de uma ordenação na acção de formação em que se encontra integrada.

3.4.1 Entidades de Alfabetização

Por cada acção de formação, o professor deve seleccionar as EntAlf que vai utilizar. Por cada EntAlf, o professor deve gerar o número de UniAlf que julgar necessárias. Finalmente, deve efectuar a ordenação de todas as UniAlf.

De momento, o protótipo de abcNet disponibiliza duas EntAlf, denominadas de entidade 1 de alfabetização (Ent1Alf) e entidade 2 de alfabetização (Ent2Alf), cujos detalhes funcionais podem ser analisados em [OSCAR-CELDA].

3.4.2 Unidades de Alfabetização

As unidades de alfabetização (UniAlf) derivam do processo de configuração de ModAlf. As UniAlf são materializadas por intermédio de páginas HTML que são apresentadas aos alunos. Tendo em consideração o perfil dos alunos, ainda com poucas aptidões para a leitura e escrita, as UniAlf devem implementar mecanismos activos de ajuda (MAA) aos alunos. De entre os MAA a disponibilizar, destacam-se:

- explicação do objectivo a atingir com a UniAlf;
- por EleAlf:
  - visualização da ortografia;
  - audição da pronúncia;
- audição de ajudas relativas quer à função quer aos mecanismos de interacção disponíveis.

A disponibilização de MAA é o princípio geral a seguir, podendo existir situações em que alguns dos MAA não se possam ou não se devam aplicar.

Unidade 1 de Alfabetização

A unidade 1 de alfabetização (Uni1Alf) é a UniAlf que disponibiliza os interfaces necessários à implementação do método Paulo Freire. Em termos de estrutura, a Uni1Alf deve apresentar:

- palavra geradora (PalEnt) a utilizar e suas sílabas (SílEnt);
- imagem geradora a utilizar;
- grupos de SílEnt a utilizar além das sílabas família para a construção de novas palavras;
- local de construção de cada palavra proposta; a construção é efectuada sílaba a sílaba a partir da ficha de descoberta; a pronúncia da palavra a construir deve estar disponível para ser ouvida pelo aluno;
- montra com todas as palavras (PalEnt) já construídas.

Por Uni1Alf, o professor pode configurar até quinze palavras a construir pelo aluno. As palavras devem ser apresentadas ao aluno uma de cada vez e na sequência definida pelo professor. Cada uma das quinze palavras são construídas a partir da mesma ficha de descoberta que deriva directamente ou indirectamente das sílabas da palavra geradora. Apenas após a correcta construção de uma palavra se pode dar início à construção da palavra seguinte. À medida que o aluno for avançando na resolução de uma Uni1Alf, as palavras já construídas devem ser apresentadas numa *montra* para consulta por parte do aluno, quer em termos de PalOrt quer em termos de PalPro.

Unidade 2 de Alfabetização

A unidade 2 de alfabetização (Uni2Alf) possui uma filosofia distinta da Uni1Alf. Na Uni1Alf todas as palavras a construir são condicionadas pelas sílabas da palavra geradora. Na Uni2Alf, cada palavra a construir tem associada uma ficha de descoberta que é definida pelo professor aquando da configuração do Mod2Alf. Isto significa que o professor pode escolher primeiro a palavra a construir e de seguida quais as sílabas que o aluno deve utilizar para a construção da mesma. O número máximo de palavras possíveis para construção por Uni2Alf é de quinze, tal como se verifica na Uni1Alf. Em termos de estrutura, a Uni2Alf deve apresentar, por palavra a construir:

- imagem associada à palavra;
- palavra a construir (opcionalmente PalOrt e obrigatoriamente PalPro);
- ficha de descoberta que é composta por uma ou mais SílEnt;
- local de construção da palavra;
- montra com todas as palavras (PalEnt) já construídas.

A Uni2Alf deve também disponibilizar mecanismos que facilitem a iniciação à utilização de abcNet, nomeadamente dando oportunidade ao aluno, por opção do professor durante a configuração do Mod2Alf, de construir palavras cuja ortografia se encontra já totalmente ou parcialmente disponível para consulta.

Deste modo, a selecção das sílabas a utilizar para a construção da palavra pode ser facilitada pela exposição da palavra a construir. À medida que o aluno for avançando na resolução de uma Uni2Alf, as palavras já construídas devem ser apresentadas numa *montra* para consulta por parte do aluno, quer em termos de PalOrt quer em termos de PalPro.

3.4.3 Modelos de Alfabetização

Os modelos de alfabetização (ModAlf) permitem a configuração das UniAlf a utilizar nas acções de formação. Cada ModAlf centra-se num objectivo pedagógico e através do processo de configuração executado pelo professor, geram-se UniAlf com as especificidades requeridas pelo professor. Ao professor deve ser, sempre que possível, dada a opção de definir os conteúdos a utilizar em cada UniAlf, satisfazendo assim os requisitos do método Paulo Freire.

Modelo 1 de Alfabetização

O modelo 1 de alfabetização (Mod1Alf) é o modelo integrado na Ent1Alf e a partir do qual se configuram as Uni1Alf. Por análise da Uni1Alf verifica-se que a configuração de uma Uni1Alf requer a definição de:

- palavra geradora a utilizar;
- imagem geradora a utilizar;
- outros grupos de sílabas a utilizar além do grupo das sílabas família;
- até quinze palavras a construir.

De realçar que nem as sílabas da palavra geradora, nem as sílabas família das sílabas da palavra geradora necessitam de ser configuradas, pois essa informação é obrigatória no Mod1Alf e pode ser automaticamente obtida a partir do repositório de abcNet. Os grupos de sílabas família são de utilização obrigatória, pelo que apenas os restantes grupos são sujeitos a confirmação. A identificação exacta das sílabas a utilizar, ficha da descoberta, também é obtida por leitura da informação existente no repositório central. No método Paulo Freire, a selecção das palavras a escrever é da responsabilidade do aluno. Aqui, é o professor quem define as palavras a serem construídas pelo aluno, estando as suas pronúncias disponíveis para audição nas Uni1Alf.

Modelo 2 de Alfabetização

O modelo 2 de alfabetização (Mod2Alf) é o modelo integrado na Ent2Alf e a partir do qual se configuram as Uni2Alf.

Por análise das Uni2Alf verifica-se que a configuração de uma Uni2Alf requer a definição de:

- até quinze palavras a construir pelo aluno;
- por cada palavra:
    - que sílabas utilizar na ficha de descoberta (SílOrt e SílPron) e sua localização na ficha de descoberta;
    - que grupos de sílabas utilizar das sílabas anteriores;
    - que elementos ortográficos da palavra a construir se encontram expostos, de modo a ajudar o aluno na tarefa de construção da palavra; os elementos ortográficos referidos são a PalOrt e cada uma das SílOrt.

3.5 Escalabilidade

De momento não existe uma estimativa para o número possível de utilizadores de abcNet. Entretanto, quer a arquitectura definida quer as plataformas seleccionadas devem ter em consideração a probabilidade de virem a existir várias centenas de alunos se não mesmo milhares de alunos a utilizarem em simultâneo abcNet. A expansão futura de abcNet não deve ser comprometida por opções restritivas tomadas logo no início do projecto.

4 Abordagem Tecnológica

Neste capítulo vamos abordar as questões relacionadas com as opções tomadas nas vertentes tecnológicas.

As tecnologias a utilizar no desenvolvimento de um produto é uma das decisões fundamentais a tomar na fase inicial. Opções erradas podem não só condenar o produto na sua fase de desenvolvimento, como também na sua fase de manutenção, quer correctiva, quer preventiva, quer evolutiva. Muitos factores devem ser ponderados de modo a minorar não só a probabilidade de ocorrência de decisões erradas como também do impacto que essas mesmas decisões possam ter.

Para uma aplicação do género de abcNet com uma arquitectura baseada na *Internet*, por si só, introduz algumas exigências em termos funcionais, tais como:

- sistema operativo;
- base de dados acessível através de uma rede;
- servidor de páginas *HTML* dinâmicas com acesso à base de dados;
- explorador de *Internet* consumidor das páginas *HTML* geradas pelo servidor *Web.*

Em termos técnicos, alguns dos critérios de seriação das ferramentas de desenvolvimento a utilizar são:

- quais as características do ambiente de desenvolvimento (IDE-Interface Development Environment) ?
- qual o nível de integração das diversas vertentes (bases de dados, páginas dinâmicas, programas)?
- qual o nível de unificação das diversas vertentes (uma só tecnologia)?
- qual o estado da arte?

Em termos de custos das ferramentas, alguns dos critérios a ter em consideração, são:

- quanto custa a ferramenta?
- quanto custa a formação?
- qual a curva de aprendizagem?

Em termos de custos de desenvolvimento, alguns dos critérios a ter em consideração, são:

- quanto custa o desenvolvimento?
- quanto custa a manutenção?

Muitos outros critérios poderiam ainda ser definidos, nomeadamente:

- quais os custos a suportar pelos clientes para montagem e manutenção da plataforma necessária à instalação e utilização de abcNet?
- qual o desempenho, segurança, escalabilidade, tolerância a falhas, etc. exigidas?

De todos os critérios enumerados, apenas alguns tiveram peso na opção tomada:

- custos de aquisição das ferramentas;
- curva de aprendizagem;
- facilidade de utilização;
- nível de integração das diversas vertentes (bases de dados, páginas dinâmicas, programas);
- nível de unificação das diversas vertentes (uma só tecnologia);
- estado da arte.

Baseado nestes critérios, a opção recaiu numa plataforma WISA[APEN], toda baseada em tecnologias da Microsoft. No que diz respeito às ferramentas, pelos mesmos motivos, as opções recaíram no Visual Studio.Net e Sql-Server. Das tecnologias, algumas merecem uma atenção mais detalhada, nomeadamente, a plataforma .NET, C#, ASP.NET e Web Services.

4.1 Plataforma .NET

A plataforma Microsoft.NET [MS-dotNET] [THUN] é o mais recente ambiente disponibilizado pela Microsoft quer para o desenvolvimento quer para a execução de aplicações informáticas. A plataforma .NET introduziu modificações substanciais na forma de desenvolver aplicações, nomeadamente nas aplicações dirigidas para a *Web*.

A arquitectura geral de .NET engloba quatro grupos de produtos a saber:

Framework .NET – novo ambiente para o desenvolvimento de aplicações.

Produtos .NET - aplicações da Microsoft baseadas na plataforma .NET que incluem a base de dados SQL Server [MS-SQL], o servidor de *e-mail* Exchange [MS-EXC], o servidor de conteúdos BizTalk [MS-BIZ], etc.

Serviços – disponibilização de serviços aos utilizadores em que o projecto Hailstorm [MS-HS] é o caso paradigmático.

Dispositivos – suporte para dispositivos não-PC, tais como as *game boxes* ou os telemóveis.

A plataforma .NET é constituída por cinco componentes principais, tal como apresentado na Figura 1. No nível inferior situa-se o sistema operativo que pode ser qualquer versão do Windows XP, Windows

2000 ou posterior ou Windows CE. Como veremos, existem outras possibilidades para além do sistema operativo *Windows*.

Acima do Sistema Operativo encontram-se três blocos:

- Servidores .NET – produtos da Microsoft que se destinam a ser integrados em aplicações e que vão desde BizTalk, Exchange, SQL Server, etc. ;
- Framework .NET – framework .NET a ser descrita em detalhe;
- Serviços .NET – serviços disponibilizados pela Microsoft e outros fornecedores, tais como o Microsoft Passport [MS-PASSP], projecto HailStorm [MS-HS], etc.

No topo da plataforma encontra-se o Visual Studio.NET (VS.NET) [MS-VS], que é o sucessor do Visual Studio 6, e que está vocacionado para o desenvolvimento de aplicações para a plataforma .NET.

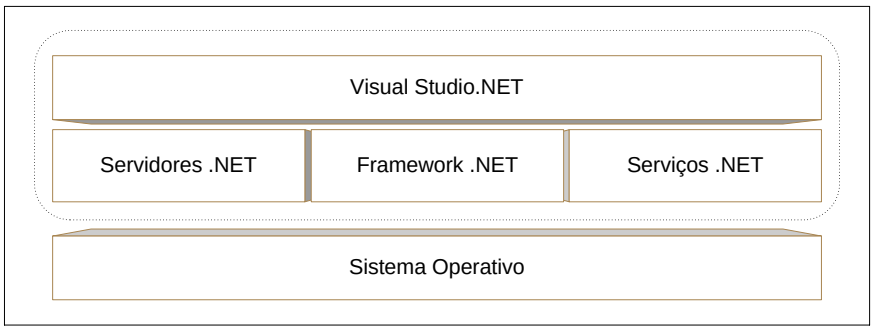

Figura 1 – Plataforma .NET

4.1.1 Objectivos da *Framework* .NET

Os objectivos a atingir pela *framework* são, entre outros, melhor suporte a componentes, integração de diversas linguagens de programação, inter-operação entre aplicações, facilidade de desenvolvimento e de distribuição, etc.

Componentes

A tecnologia COM (Component Object Model) já permitia a utilização de componentes em aplicações, mas requeria a satisfação de um conjunto de requisitos que tornava a sua utilização engenhosa.

A *framework* .NET apresenta uma nova abordagem, eliminando questões mais complexas tais como o registo, suporte para IUnknown [MS-IUK], etc.

Integração de Múltiplas Linguagens

A tecnologia COM já suportava o desenvolvimento de componentes em qualquer linguagem desde que satisfizessem os requisitos necessários. A reutilização era possível ao nível dos componentes mas não ao nível das classes. A *framework* .NET suporta não só independência ao nível dos componentes mas também a integração ao nível da linguagem. Significa que a herança, tratamento de excepções, polimorfismo são aplicáveis entre linguagens diferentes. A integração é possível desde que o compilador gere um código intermédio, *Common Intermediate Language*(CIL) [MS-CIL], que satisfaça um conjunto de requisitos especificados na *Common Language Specification* (CLS) [MS-CLS]. Além das linguagens Microsoft suportadas, tais como o C#, Visual Basic, etc., outros fornecedores também já disponibilizaram as suas linguagens para esta *framework* tal como o Delphi [DELPHI], o Cobol [COBOL], Fortran [SALFORD], Perl [PERL], etc.

Independência do Sistema Operativo e de Processador

A Microsoft enveredou por uma estratégia distinta da que sempre seguiu no passado, tendo desenvolvido um componente "Common Language Runtime" (CLR) [MS-CLR] que conceptualmente é idêntico a uma *Java Virtual Machine* (JVM) da Sun e cujo objectivo principal é a abstracção da plataforma na qual assenta a própria *framework*. A CLR é responsável por um conjunto vasto de tarefas das quais se destaca desde já a compilação e execução da CIL. De modo a abrir a possibilidade a que vários fabricantes possam implementar soluções da CLR, a Microsoft submeteu à ECMA [ECMA] a aprovação da *Commmon Language Interface* (CLI) [ECMA-335] para normalização. A CLR é a implementação Microsoft da CLI. Além da CLR, destaca-se o projecto Mono [MONO], cujo resultado é a disponibilização de uma plataforma .NET para o sistema operativo Linux [GNU].

Suporte a Normas não Proprietárias

Ao contrário de COM, .NET pode recorrer a SOAP [W3C-SOAP] para a comunicação entre aplicações distribuídas na *Web*. O protocolo SOAP é baseado em normas tais como o XML [W3C-XML] e HTTP [W3C-HTTP]. Além do SOAP, .NET também suporta UDDI [UDDI] e WSDL[W3C-WSDL]. Da plataforma, além da CLI, a Microsoft também submeteu à ECMA normalização a sua nova linguagem de programação C#, que é neste momento a norma ECMA-334 [ECMA-334].

Desenvolvimento

Tradicionalmente, cada linguagem de programação possui o seu próprio ambiente de desenvolvimento que vai desde o IDE, às APIs, às bibliotecas de classes, etc. .NET fornece apenas um conjunto de classes que são partilhadas por todas as linguagens. Com o recurso ao Visual Studio.NET é possível utilizar sempre o mesmo IDE para as linguagens suportadas.

Distribuição

Quem já trabalhou com DLLs conhece o significado de DLL Hell [CP-HELL]. O processo de instalação de uma aplicação COM envolve sempre um conjunto vasto de ficheiros a serem instalados num outro conjunto vasto de directorias, vários registos, instalação de componentes, etc. Remover completamente uma aplicação é uma tarefa praticamente impossível pois resta sempre alguma informação não removida. Para além do processo de instalação e de remoção ainda podem surgir questões relacionadas com as versões de componentes que podem implicar que dada aplicação deixe de funcionar correctamente devido à instalação de uma terceira aplicação as quais partilham determinados componentes mas com versões distintas. .NET proporciona um ambiente de distribuição significativamente distinto do apresentado. Os componentes .NET não necessitam, podendo ser muitas vezes aconselhável ou até obrigatório, de ser registados pois contêm em si mesmos quer toda a informação necessária à sua correcta utilização quer os mecanismos necessários à sua disponibilização, através do recurso à utilização de metadados e da acção de *reflection* [MS-REFL]. Numa grande maioria de situações é suficiente copiar os componentes para as directorias destino pois cada aplicação usa sempre as DLLs com as quais foi desenvolvida e testada.

Fiabilidade

Na vertente associada à fiabilidade realçam-se os seguintes aspectos:

- ao contrário do que se verificava com a linguagem C++, todas as classes de .NET são derivadas de uma classe base que suporta identificação e caracterização em tempo de execução. Esta característica possibilita que a CLR possa efectuar validações antes de tomar qualquer iniciativa que possa provocar situações anómalas;
- .NET apresenta uma estrutura de tratamento de excepções ao nível da CLR o que permite um mecanismo consistente de tratamento de excepções comum a todas as linguagens;
- ao contrário do que acontecia com C++, mas à semelhança do JVM, .NET possui um sistema de gestão de memória permitindo a libertação automática dos recursos já não utilizados.

Segurança

.NET permite a implementação de mecanismos de segurança não só ao nível dos componentes que são utilizados mas também ao nível dos métodos de cada classe. A segurança em .NET é iniciada quando uma classe é carregada pela CLR, analisando não só direitos de acesso mas também os interfaces disponibilizados e tipos de argumentos trocados. À medida que forem sendo necessários recursos adicionais, as credenciais da classe são tidas em consideração para a tomada de decisão. Também são

implementados mecanismos de segurança ao nível das fronteiras entre processos e máquinas de modo a evitar o acesso a informação sensível num ambiente distribuído [KAMRAN].

Desempenho

Com .NET o aumento do desempenho no desenvolvimento de aplicações é um objectivo, podendo eventualmente em algumas vezes ter associado um desempenho menos bom na execução das aplicações. Nos dias que correm, o aumento de produtividade associada à mão de obra é essencial no custo final de um produto. O preço a pagar é a necessidade de dispor de uma máquina com recursos acrescidos, nomeadamente em termos de CPU e de memória. Se tal solução se verificar insuficiente, resta o recurso a um processamento distribuído. Em qualquer dos casos, normalmente, o custo adicional a pagar é inferior ao custo da mão de obra especializada adicional que seria necessária.

4.1.2 Arquitectura da Framework .NET

A arquitectura da *framework* .NET encontra-se apresentada na Figura 2.

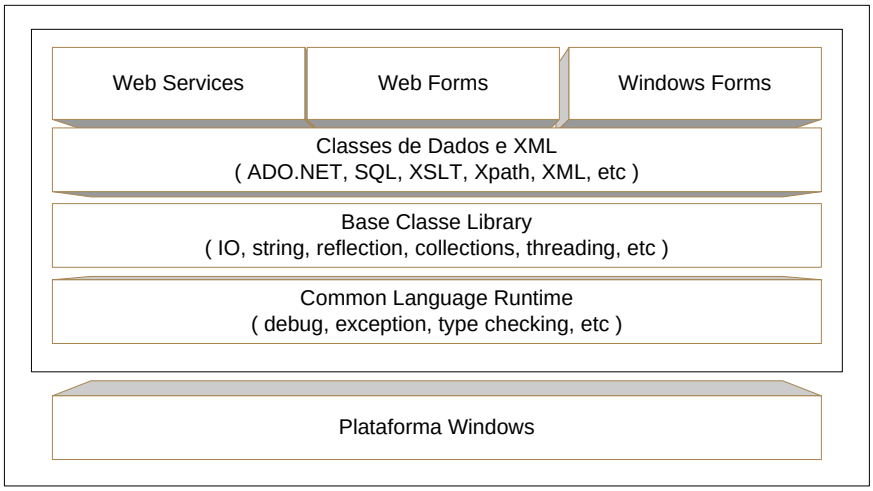

Figura 2 – Arquitectura da framework .NET

Web Services

Conjunto de classes que suportam o desenvolvimento de componentes distribuídos.

Web Forms

Conjunto de classes que permitem o desenvolvimento de aplicações com interfaces gráficos para a *Internet*.

Windows Forms

Conjunto de classes que permitem o desenvolvimento de aplicações gráficas para o sistema operativo *Windows*.

Classes de Dados

As classes de dados são um conjunto de classes vocacionadas para a manipulação de dados residentes quer em bases de dados, ficheiros, ficheiros em formato XML, etc. ADO.NET é desenvolvido posteriormente em maior detalhe.

Base Class Library

A *Base Class Library* (BCL) [MS-BCL] é uma biblioteca de classes, interfaces e tipos de valores. É a partir desta biblioteca que o sistema de desenvolvimento (System Development Kit - SDK) da *framework* obtém acesso às funcionalidades da plataforma.

Common Language Runtime

A CLR é o componente mais importante da *framework* .NET pois é ele que gere e executa o código. De entre as diversas características do CLR, destaca-se: gestão de memória, segurança, execução e compilação do código, etc.

Apesar das semelhanças com a JVM, algumas diferenças são de realçar:

- JVM está disponível para várias plataformas de sistema operativo; CLR, além do sistema operativo Windows, também se encontra disponível para a plataforma Linux [GNU];
- JVM suporta apenas a linguagem Java enquanto que CLR suporta todas as linguagens que tiverem tradução para a CLS;
- o código é sempre executado a partir da JVM enquanto que em .NET o código é compilado para a respectiva arquitectura.

4.1.3 Web Forms

Os *Web Forms* (WF) permitem a criação de páginas *Web* a partir de um interface gráfico. Os WF geram páginas que podem ser apresentadas em um qualquer explorador de *Internet*, sendo o servidor o responsável por gerar a lógica necessária à sua apresentação. Os WF suportam as linguagens tradicionais que são utilizadas nas páginas tais como o HTML, XML e ECMAScript [ECMA-Index].

Características dos WF:

- são baseados na tecnologia ASP.NET cujo código é executado no servidor;
- as páginas são geradas no servidor em conformidade com o explorador de *Internet* do cliente;

- pode ser utilizada qualquer linguagem de programação suportada pela plataforma .NET;
- são suportados no Visual Studio.NET que disponibiliza ferramentas de desenvolvimento rápido (Rapid Application Development – RAD);
- permitem a utilização quer dos controlos disponíveis quer de controlos a desenvolver pelo utilizador.

WF permitem, entre outras::

- a construção de páginas com conteúdos dinâmicos construídos à imagem das necessidades do programador, com recurso à utilização dos *Web User Controls* [MS-WUC] e dos *Web Custom Controls* [MS-WCC]. Vantagens e desvantagens na utilização de um ou de outro tipo de controlos encontram-se apresentadas na Tabela 2;
- a criação de um modelo que possibilita ver cada WF como uma única entidade e não como duas unidades separadas, uma do cliente e outra do servidor; o modelo permite uma programação de forma mais intuitiva, tal como a atribuição de valores a propriedades de elementos HTML, resposta a eventos, etc.; genericamente pode-se afirmar que os controlos do servidor podem ser utilizados como se tratassem de uma aplicação *Windows*;
- o tratamento de eventos no servidor, quer ocorram do lado do cliente quer do servidor; no que diz respeito aos eventos com origem do lado do cliente a serem processados no servidor, estes são tratados no servidor de uma forma transparente e a sua programação é idêntica aos eventos dos controlos numa aplicação *Windows*;
- uma gestão automática do estado dos formulários, não havendo necessidade da reposição explícita de valores sempre que a página é requisitada ao servidor, com excepção das palavras chave.

Web User Controls	**Web Custom Controls**
Mais fáceis de criar	Mais difíceis de criar
Suporte limitado para a utilização em ambientes com visualização em tempo de desenvolvimento	Suporte completo para ambientes com visualização em tempo de desenvolvimento
Uma cópia do controlo por projecto	Uma só cópia do controlo para toda a aplicação
Não podem ser adicionados à caixa de ferramentas	Podem ser adicionados à caixa de ferramentas do Visual Studio
Bons para disposições estáticas	Bons para disposições dinâmicas

Tabela 2 – Quadro comparativo dos WUC e dos WCC

4.1.4 Web Services

Um *Web Service* (WS) [W3C-WS] é um interface que disponibiliza mecanismos normalizados de inter-operação entre distintas aplicações informáticas, correndo em distintas plataformas. Os WS recorrem apenas a protocolos normalizados para a sua implementação, aglutinando desta forma as maiores vantagens quer da *Internet* quer das aplicações baseadas em componentes. Funcionalmente, tal como os componentes, os WS correspondem a caixas negras que podem ser reutilizadas sem preocupação quanto à forma da sua implementação. Ao contrário das tecnologias baseadas em componentes, os WS não são acedidos via protocolos baseados no modelo *object-oriented*, tal como DCOM, RMI ou CORBA, sendo acedidos por intermédio dos protocolos normalizados utilizados na *Internet* tal como o HTTP para a comunicação e o XML para os dados. No essencial, um WS consiste na disponibilização de funcionalidades numa rede através de um interface assente em tecnologias normalizadas tais como HTTP, XML, SMTP [IETF-2821], SOAP, etc.

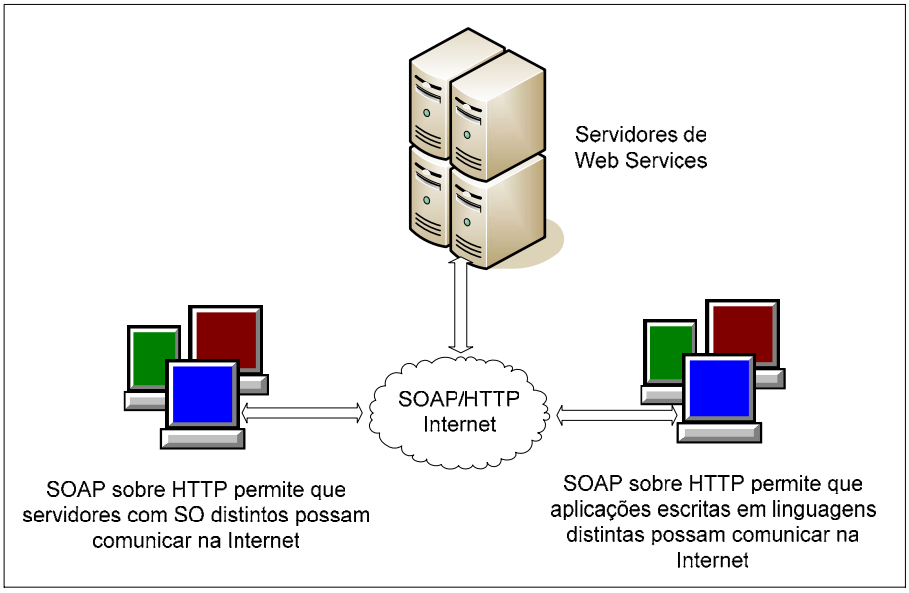

Figura 3 - Mecanismos de acesso aos *Web Services*

A Figura 3 apresenta uma visão geral quer da topologia quer dos mecanismos de acesso aos WS. Os WS funcionam como uma camada de abstracção, separando quer a plataforma do fornecedor (IBM, HP, UNIX, Windows, etc), quer os detalhes do código (OOL, C++, Java, SQLServer, Oracle, assembler, etc), das funcionalidades disponibilizadas e simultaneamente expondo os serviços da aplicação numa rede com recurso às tecnologias normalizadas já citadas. O acesso aos serviços disponibilizados é materializado através do recurso a mecanismos de troca de mensagens. Estes mecanismos de troca de mensagens há já muito que não são novidade, sendo o DCOM e CORBA alguns dos mais conhecidos, sendo protocolos proprietários. A novidade nos WS é a utilização de protocolos normalizados para a transferência de mensagens sobre HTTP, que é o SOAP. SOAP permite que aplicações escritas em

linguagens distintas, correndo em plataformas também distintas, efectuem chamadas a métodos remotos (RPC – *remote procedure call*), mesmo através de *firewalls*, visto utilizar o porto 80.

Em termos de estrutura interna, um WS apresenta dois componentes distintos mas intimamente interligados:

1) Service Listener (SL) que é responsável:

   a) pela implementação dos protocolos normalizados,

   b) pela recepção das mensagens de pedido,

   c) pelo envio das mensagens de resposta,

2) Service Proxy que é responsável:

   a) pela descodificação dos pedidos recebidos do SL,

   b) pela activação das funcionalidades solicitadas nas mensagens recebidas do SL, tal como a activação de métodos,

   c) se necessário, pela formatação de uma mensagem resposta a enviar ao solicitador do serviço.

Do que já foi dito pode-se deduzir que os WS vieram introduzir alguns novos desafios associados quer à pesquisa dos WS existentes quer aos interfaces específicos de cada um dos serviços disponibilizados. No passado, os desafios situavam-se em questões tais como: a) qual a linguagem de programação utilizada, b) qual o sistema operativo utilizado, c) qual o motor de gestão de base de dados utilizado, etc. Até agora, aplicações Java funcionavam melhor com outras aplicações Java, aplicações DCOM funcionavam melhor com outras aplicações DCOM, etc. Geralmente seria necessário algum esforço adicional para que aplicações heterogéneas pudessem funcionar em colaboração. Os WS vêm permitir a interligação de todas estas tecnologias e simultaneamente permitir que cada uma delas possa continuar a evoluir per si. Os WS são fruto de regulamentação e normalização com base no *World Wide Web Consortium – W3C* [W3C-WS].

Localização e Descoberta

Os WS só fazem sentido se se encontrarem disponíveis para utilização. O esquema geral utilizado para atingir este objectivo centra-se num conjunto de passos que a seguir se descrevem. O *service provider* (SP) – é uma entidade que publica uma descrição de todos os serviços disponibilizados via um *service registry* (SR). Normalmente, estes registos encontram-se em sítios na *Internet*, os quais podem ser de acesso público ou condicionado. Pode pensar-se nestes servidores como sendo as páginas amarelas dos WS. O *service consumer* (SC) pesquisa o SR com o objectivo de encontrar os WS que satisfaçam as suas

necessidades. No caso de ser encontrado um WS que satisfaça os requisitos tem também acesso à descrição desses mesmos serviços e ainda aos mecanismos exigidos para acesso a esses mesmos serviços.

Camadas

Os WS são implementados em cinco camadas que se empilham. As cinco camadas encontram-se apresentadas na Figura 4, cuja descrição individual se passa a efectuar:

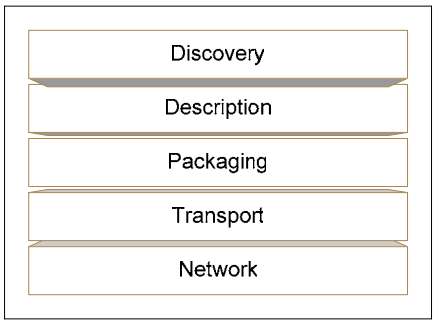

Figura 4 – Camadas dos *Web Services*

Discovery

Por *Discovery* deve-se entender o acto de localização de uma descrição de um WS que satisfaz determinados requisitos funcionais. Apesar de a semântica não ajudar, o serviço *Discovery* é também utilizado na publicação dos serviços e dos seus requisitos funcionais. Quer o registo quer a pesquisa devem ser efectuados com recurso a tecnologias normalizadas. De entre estas, a mais utilizada para o efeito é o UDDI – *Universal Description, Discovery and Integration* [UDDI].

Description

Por *Description* deve-se entender os mecanismos disponibilizados para a descrição dos WS. A descrição dos serviços engloba não só a localização do WS mas também a descrição das operações por ele disponibilizadas. A tecnologia utilizada na descrição dos WS é a WSDL – *Web Services Description Language* [W3C-WSDL].

Packaging

Por *Packaging* deve-se entender os mecanismos utilizados para empacotamento da informação a ser transmitida e que é referente aos WS. As tecnologias mais utilizadas para o efeito são XML e SOAP.

Transport

Transport é a camada responsável pelo transporte da informação entre as duas aplicações, no qual podem ser utilizadas tecnologias tais como o TCP, HTTP, SMTP, etc.

Network

Network é a camada que implementa as funcionalidades subjacentes à comunicação em rede recorrendo nomeadamente a IP.

Segurança

Os WS foram pensados para serem um meio que disponibilize mecanismos de acesso fácil através da *Internet*. SOAP é a norma mais utilizada no transporte da informação entre os WS e os que a eles acedem. O sucesso dos WS não se encontra garantido pelo facto de utilizarem apenas protocolos normalizados, nem pelo facto de as ferramentas de desenvolvimento existentes facilitarem significativamente não só o seu desenvolvimento mas também os mecanismos de acesso aos mesmos. Por defeito, os WS recorrem a mensagens que são escritas em texto não encriptado ficando desta forma totalmente expostos a utilizadores alheios à comunicação. Os aspectos relativos à segurança, se não forem salvaguardados condenam concerteza os WS ao insucesso. Cientes desta situação, empresas tais como a Microsoft e a IBM patrocinaram a constituição de um grupo de trabalho, OASIS [OASIS], que se dedica a questões relacionadas com a segurança nos WS. A designação geral pela qual são conhecidas as normas de segurança é *WS-Security* ou simplesmente WSS [OASIS-WSS].

WSS define extensões ao SOAP para implementação de mecanismos de autenticação, integridade e confidencialidade. Autenticação resolve questões tais como: quem é o utilizador, como é que o utilizador prova que ele é quem diz que é. A integridade assegura que a informação recebida não foi alterada durante a transmissão. Com a confidencialidade pretende-se que a mensagem não possa ser interpretada por elementos externos à comunicação.

WSS é um elemento do cabeçalho de SOAP, denominado de <Security>. Contém a informação relativa à implementação de mecanismos de segurança tais como *tokens*, assinaturas e cifragem.

4.1.5 C#

C# é a mais recente linguagem de programação com origem na Microsoft. Pertence à família das linguagens derivadas do C, tais como o C++ e o Java. C# foi submetida à ECMA [ECMA-334] de modo a ser uma norma aberta e de acesso universal. C#, ao contrário de C++, é uma linguagem totalmente orientada ao objecto. C# apresenta muitas características que são distintas [MS-C#:C++] das de C++ e que fazem de C# uma linguagem mais fácil de utilizar. A aproximação à filosofia utilizada pela Sun no Java é óbvia. Sendo C# uma linguagem mais recente que Java, a Microsoft aproveitou a experiência tida

já com C++ e J++ e a existente com Java para introduzir diversas melhorias que se traduzem numa linguagem mais moderna, mais elegante e mais poderosa [DOBBS].

4.1.6 ASP.NET

ASP.NET é mais do que a versão que se seguiu às ASP. ASP.NET fornece um modelo de ambiente de desenvolvimento unificado para o desenvolvimento de aplicações para a *Internet*. ASP.NET suporta a sintaxe anterior de ASP; mas as alterações e melhorias introduzidas são de tal ordem que os novos desenvolvimentos devem evitar o recurso à sintaxe de ASP. ASP.NET, fazendo parte integrante da plataforma .NET, suporta uma qualquer das linguagens disponíveis na plataforma, incorporando assim todos os benefícios que daí possam advir, tais como: CLR, programação orientada a objectos, etc. Na criação de aplicações ASP.NET podem ser utilizados dois modelos, a saber:

- *Web* Forms: os WF permitem a construção de páginas *Web*, tal como já descrito em 4.1.3.;
- *XML Web Service* [MS-XMLWS] é um mecanismo cliente-servidor para troca de informação com recurso a tecnologias normalizadas, tal como já referido em 4.1.4.

Cada um dos modelos pode recorrer às características quer de ASP.NET, quer da *framework* .NET quer da CLR:

- os conhecimentos de ASP podem ser aplicados em ASP.NET, contudo o modelo ASP.NET é bastante diferente do ASP, sendo mais estruturado e orientado ao objecto; isto significa que as páginas ASP necessitam de algumas adaptações para poderem ser compatíveis com o modelo ASP.NET;
- o acesso a base de dados é implementado com recurso ao ADO.NET que é descrito posteriormente em 4.1.7;
- ASP.NET disponibiliza um modelo que permite o desenvolvimento de programas associados a WF num ficheiro à parte do que contém o HTML; o ficheiro contém toda a lógica necessária à construção dinâmica da página do lado do servidor que vai desde o acesso a base de dados, tratamento de eventos, etc.; este ficheiro é compilado e executado, e reside do lado do servidor; tem a designação geral de *code-behind*;
- ASP.NET disponibiliza *application* [MS-APPST] e *session state* [MS-SESST];
- o código ASP.NET é compilado para CIL e não interpretado, o que se traduz numa melhoria em termos de desempenho; quando uma aplicação é activada pela primeira vez, o código CIL é compilado para a linguagem nativa não necessitando de voltar a ser compilado, a menos que o servidor seja reiniciado;

- ASP.NET disponibiliza um mecanismo de *trace* [MS-ASPTR] que pode ser globalmente activado ou desactivado em função do estado de desenvolvimento da aplicação;
- ASP.NET contém esquemas por defeito para autorização e autenticação, podendo ser alterados de modo a satisfazer as necessidades específicas [MS-ASPSC];
- a configuração [MS-ASPCNF] da aplicação pode residir num ficheiro XML que é o *Web.config*.

4.1.7 ADO.NET

ADO.NET [MS-ADO] é o componente da plataforma .NET para acesso a dados. A Figura 5 apresenta uma vista simplificada dos principais objectos de ADO.NET. As classes de ADO.NET encontram-se divididas em duas categorias, 1) *Data Providers* [MS-ADODP] que são responsáveis pela comunicação com a entidade física que armazena a informação e 2) *DatSet* [MS-ADODS] que é uma representação local, em memória, da informação. Qualquer um dos consumidores, *Data Consumer*, pode comunicar com qualquer um dos componentes das duas categorias.

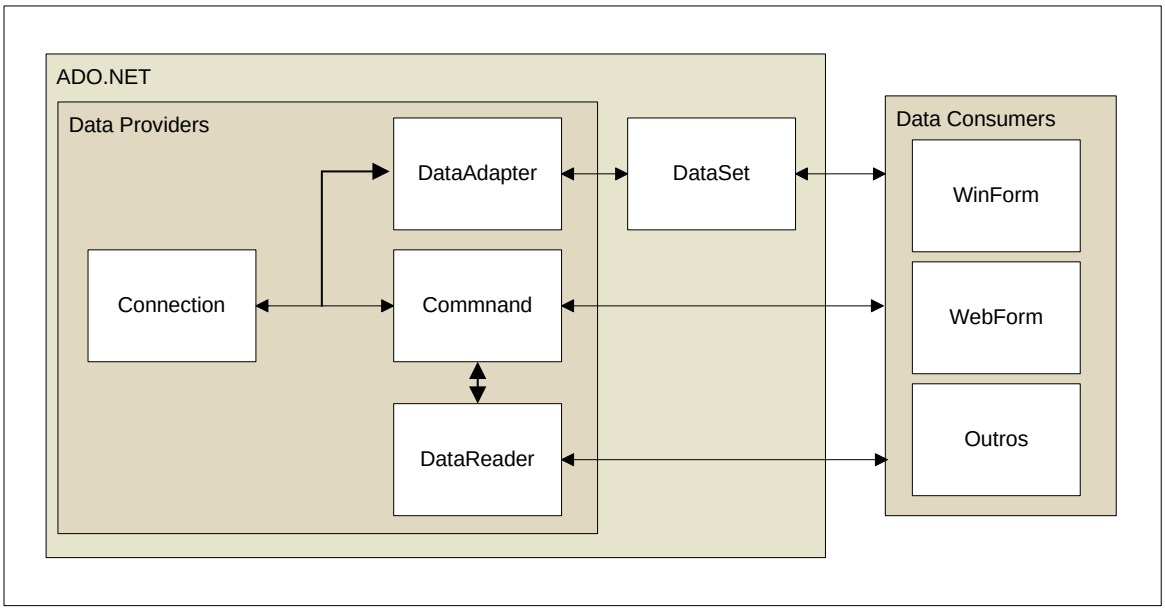

Figura 5 – Principais objectos de ADO.NET

4.2 Sistema de Gestão de Base de Dados

Segundo Thomas Connolly [CONN-DB, pág 14], uma base de dados é uma colecção de dados logicamente relacionados, com descrição desses mesmos dados, desenhados de modo a satisfazer as

necessidades de informação de uma organização. Isto significa que uma base de dados deixou de ser um conjunto de ficheiros desgarrados cada qual com a sua informação, e muita dela redundante, para passar a ser um conjunto de dados integrados com o mínimo de redundância. A base de dados deixou de ser pertença de A ou B para passar a ser uma entidade partilhada por todos os intervenientes. Além dos dados propriamente ditos, a base de dados contém informação descritiva dos mesmos dados que ela contém, dizendo-se por isso auto-descritiva. A descrição dos dados é conhecida como o catálogo do sistema ou dicionário de dados ou metadados.

Antes do aparecimentos dos SGBD (Sistemas de Gestão de Base de Dados), os dados eram guardados em ficheiros. A utilização de SGBD, em comparação com a utilização de informação residente em ficheiros, tem vantagens e desvantagens que passamos a descrever:

Vantagens:

- redundância de dados: a redundância de dados pode ser minimizada concentrando-se apenas nas ligações necessárias entre entidades;
- consistência de dados: controlando a redundância, reduz-se o risco de inconsistência, pois se determinada informação apenas existir numa localização, não existe qualquer risco de a mesma informação assumir, em cada instante, mais do que um valor;
- mais informação com menos dados: o facto de os dados possuírem relacionamentos entre si, permite a obtenção de informação que de outro modo exigiria que os dados estivessem duplicados;
- integridade de dados: a integridade dos dados é garantida por mecanismos que evitam a violação de determinadas regras que são definidas aquando do desenho da base de dados;
- segurança: os SGBD implementam mecanismos de segurança de modo a definir para cada utilizador quais os dados a que tem acesso e que tipo de operações pode efectuar sobre os mesmos;
- cópias de segurança e serviços de recuperação: mecanismos para cópias de segurança e serviços de recuperação fazem parte dos SGBD.

Desvantagens:

- complexidade: aproveitar as potencialidades de um SGBD exige conhecimentos profundos por parte dos diversos intervenientes, desde o administrador aos utilizadores finais;

- dimensão: um SGBD, pela sua complexidade, exige grandes recursos em termos de hardware;
- custos: um SGBD pode ter um custo significativo, dependendo das opções tomadas;
- desempenho: questões relacionadas com o desempenho nem sempre são acauteladas desde o início, o que pode levar a situações intoleráveis;
- risco de falha: o facto de os dados se encontrarem centralizados, uma única falha pode ter consequências graves.

Os SGBD são a actual plataforma de gestão de informação que veio introduzir alterações profundas no modo de trabalhar e de organizar a informação. Muitos problemas ainda subsistem por resolver. O SGBD, tal como outros sistemas informáticos, têm tido uma evolução ao longos dos anos, tendo surgido vários tipos de SGBD.

Tradicionalmente, a engenharia de *Software* e os SGBD relacionais têm coexistido como entidades distintas. A tecnologia das bases de dados têm-se centrado na representação estática da informação enquanto que a engenharia de *Software* se tem centrado em aspectos dinâmicos do *Software*. O aparecimento da nova geração de SGBD, nomeadamente SGBD Orientada a Objectos (SGBD-OO) e SGBD Relacional de Objectos (SGBD-RO), tem potenciado a convergência das duas tecnologias. O conflito entre o sucesso adquirido e inequívoco das SGBD-R e a necessidade de uma nova e radical abordagem aos SGBD levaram à criação de duas correntes distintas: os defensores da SGBD-RO defendem que é suficiente a introdução de funcionalidades orientadas ao objecto nos actuais SGBD-R; os defensores do SGBD-OO defendem que o actual modelo relacional é incapaz de satisfazer as necessidades de aplicações complexas tais como o desenho assistido por computador, sistemas de informação geográfica, etc.

Fragilidades do SGBD-R

Nas últimas décadas houve uma alteração significativa nos sistemas de informação. Os SGBD-R têm tido uma aceitação geral em aplicações relativas a facturação, gestão de produção, instituições bancárias, etc. Contudo, as SGBD-R têm-se mostrado incapazes de satisfazer novos requisitos que insistentemente têm vindo a aumentar a sua relevância e em áreas tão diversas como o desenho assistido por computador, produção assistida por computador, sistemas multimédia, publicação digital, sistemas de informação geográfica, etc. Algumas destas limitações têm origem em factores tais como:

- a linguagem SQL apenas suporta a utilização de números e de texto;

- as relações de um SGBD-R não suportam tipos de dados com mais de uma dimensão, tais como *arrays*, estruturas, etc.;
- num SGBD-R não é possível definir subtipos de relações, pois estas são sempre entidades que existem por si só;
- os SGBD-R não têm em conta as vantagens inerentes ao paradigma da programação orientada ao objecto.

Os defensores dos SGBD-OO e SGBD-RO invocam um conjunto de limitações dos SGBD-R que são contempladas pelos SGBD-OO e SGBD-RO. Algumas destas dificuldades são:

- entidades: o processo de normalização leva à criação de novas relações que não têm correspondência à realidade;
- ausência de semântica: tanto os dados como os relacionamentos são representados pelas relações, não havendo possibilidade de distinção entre si, como é o caso do desdobramento de um relacionamento M:N entre duas relações que dá origem a 3 relações uma das quais de relacionamento; a acrescentar, os relacionamentos não contêm semântica que permitam a identificação do tipo de relacionamento existente: tem, pertence, etc.;
- tipos de dados: o modelo relacional assume homogeneidade vertical e horizontal; assume ainda que a intersecção entre uma linha e uma coluna deve conter sempre um valor atómico;
- operações limitadas: as operações estão limitadas às indicadas na especificação SQL-92;
- *queries* recursivos: grande dificuldade na elaboração de *queries* recursivos, isto é de *queries* sobre relacionamentos de uma relação com ela mesma;
- afinidade: SQL é uma linguagem declarativa que manuseia linhas de dados e os relacionamentos entre si, enquanto que linguagens imperativas, tal como o C#, apenas podem lidar com uma linha de cada vez.

III. Aplicação abcNet – alfabetização na Net

5 Arquitectura e Ferramentas de Desenvolvimento

Neste capítulo apresentam-se a arquitectura de abcNet e as ferramentas utilizadas no seu desenvolvimento.

5.1 Arquitectura

abcNet é uma aplicação baseada na *Internet*, vocacionada para a alfabetização. Tendo em consideração a flexibilidade quer dos servidores *Web* quer dos servidores de base de dados, um mesmo servidor pode prestar serviços a várias aplicações abcNet. No caso dos serviços *Web*, disponibilizando um *URL* de acesso por aplicação, e no caso do serviço de base de dados, disponibilizando uma base de dados por aplicação.

A arquitectura de abcNet foi definida tendo em consideração factores associados a duas vertentes fundamentais: arquitectura estrutural e arquitectura funcional. Por razões de facilidade de explicação, a descrição das duas arquitecturas é apresentada com base numa única aplicação abcNet.

5.1.1 Arquitectura Estrutural

O funcionamento de abcNet depende da existência de vários serviços externos, alguns utilizados de uma forma transparente, outros de um modo interactivo. De entre os serviços utilizados de forma transparente podem-se destacar a *Internet*, sistemas operativos, exploradores de *Internet*, etc. Estes serviços são essenciais ao funcionamento de abcNet, mas atendendo quer à sua flexibilidade quer às exigências de abcNet, estes serviços não condicionam a arquitectura pretendida por abcNet. De entre os serviços utilizados de modo interactivo, destaca-se o servidor de páginas *Internet* e o servidor de base de dados. As Figura 6 e Figura 7 apresentam duas arquitecturas estruturais distintas, ambas suportadas por abcNet. A Figura 6 apresenta uma arquitectura descentralizada baseada em vários servidores *Web*, cada um a servir uma ou mais acções de formação, e apenas um servidor de base de dados que disponibiliza uma única base de dados que é partilhada pelos servidores *Web*. Nesta arquitectura os servidores *Web* não acumulam os serviços de base de dados pelo que recorrem a outro servidor cuja localização é irrelevante desde que seja acessível a partir dos servidores *Web*.

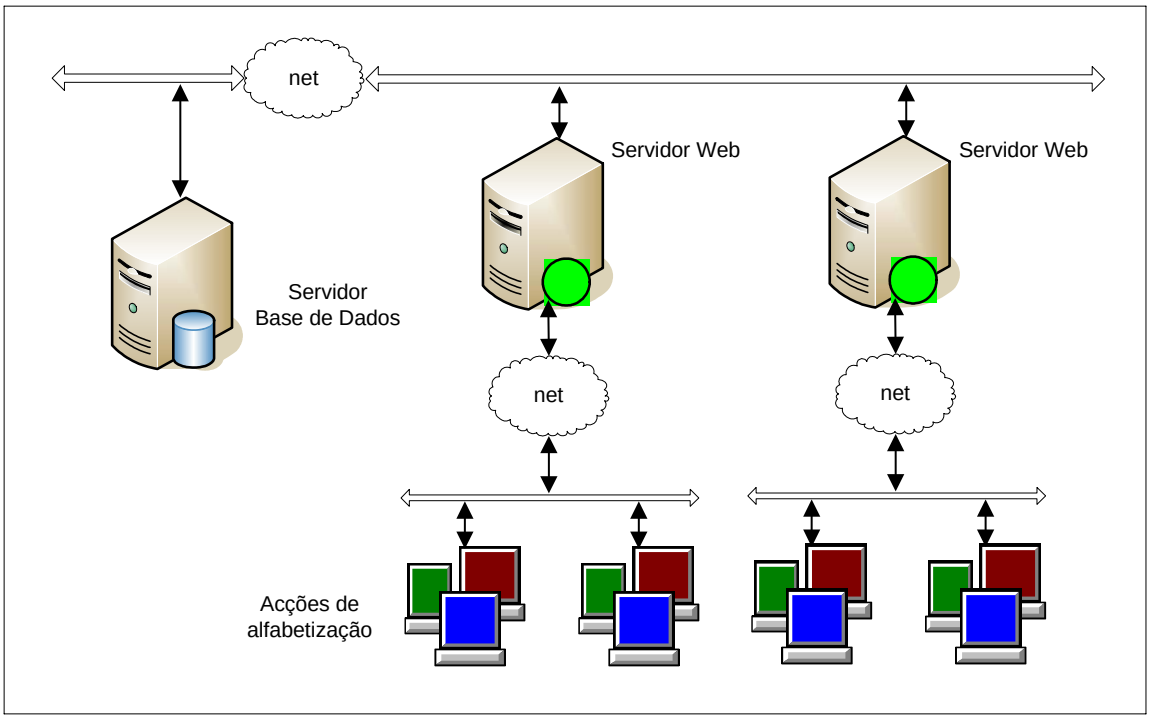

Figura 6 – Arquitectura estrutural descentralizada

Na Figura 7 apresenta-se uma arquitectura centralizada em que todas as acções de alfabetização se encontram associadas a um único servidor que presta não só os serviços de *Web* mas também o de base de dados.

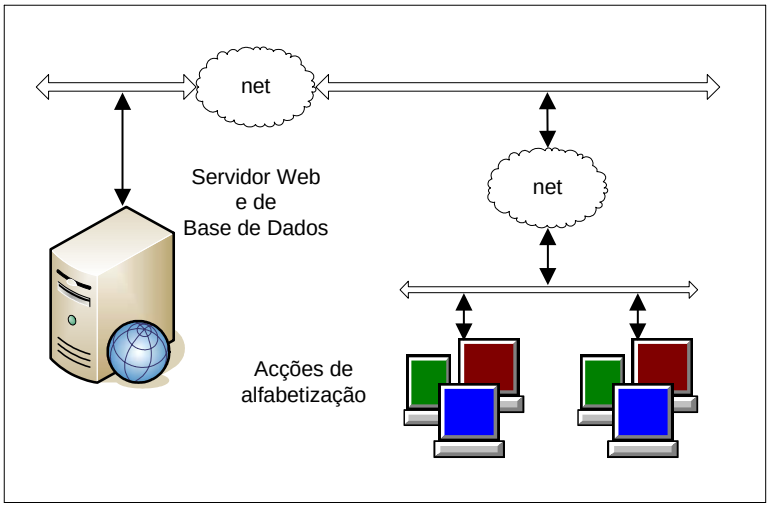

Figura 7 – Arquitectura estrutural centralizada

Situações mistas também são possíveis, tais como a de uma aplicação que apresenta dois servidores de serviços *Web* em que um deles também acumula o de serviços de base de dados.

5.1.2 Arquitectura Funcional

Ao contrário da arquitectura estrutural, a arquitectura funcional está associada aos perfis dos utilizadores. abcNet suporta três perfis distintos de utilizadores, correspondendo a cada um, um ponto de acesso a abcNet. abcNet suporta perfis de administradores, professores e de alunos.

5.1.2.1 Administrador

Ao perfil de administrador estão associadas funções de configuração global de abcNet, tais como administração de utilizadores da aplicação, administração de cursos, etc. Não confundir este administrador com o administrador da base de dados que tem acesso directo quer à base de dados quer ao SGBD, não necessitando de qualquer aplicação para o efeito. Os interfaces disponibilizados para o perfil de administrador, por razões de segurança, foram desenvolvidos como programa que corre sob um sistema operativo. Como programa, exige alguns requisitos operacionais, nomeadamente:

- plataforma com sistema operativo compatível;
- instalação do programa específico para o efeito.

O programa não exige qualquer servidor *Web* pois a aplicação corre em modo local e o acesso ao servidor da base de dados é gerido pelo próprio programa.

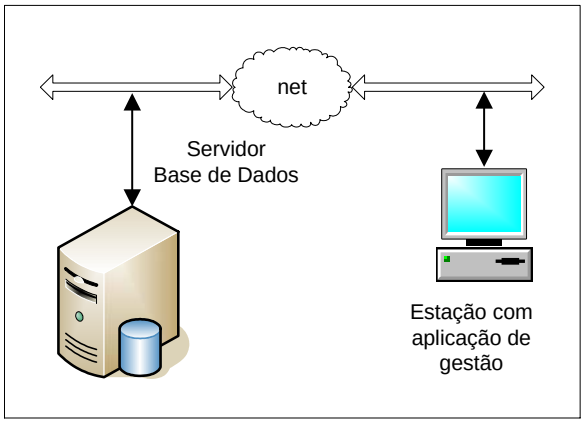

Figura 8 – Arquitectura funcional do perfil administrador

A Figura 8 apresenta a arquitectura funcional para o perfil de administrador. A autenticação no acesso à aplicação abcNet é efectuada através de um *login* de acesso, cuja validação é efectuada ao nível do utilizador da aplicação. Isto significa que a base de dados contém informação de *login* sobre os utilizadores do perfil administrador e que a validação é realizada pela própria aplicação através da comparação dos valores introduzidos com os valores existentes na base de dados. De realçar que apenas os utilizadores com perfil de administrador devem ter acesso ao programa de administração, evitando

deste modo qualquer tentativa de acesso não autorizado. A autorização de acesso à base de dados encontra-se embebida no programa, não requerendo qualquer iniciativa da parte do utilizador.

5.1.2.2 Professores e Alunos

Os perfis associados a professores e alunos, partilham a mesma arquitectura funcional. Ao contrário do que acontece com o perfil de administrador, os perfis de professores e de alunos podem ser considerados como menos exigentes em termos de segurança e também menos exigentes em termos de plataformas onde correm. Toda a aplicação abcNet associada a estes perfis está desenvolvida de forma a gerar páginas *HTML* que podem ser interpretadas por um qualquer explorador de *Internet*.

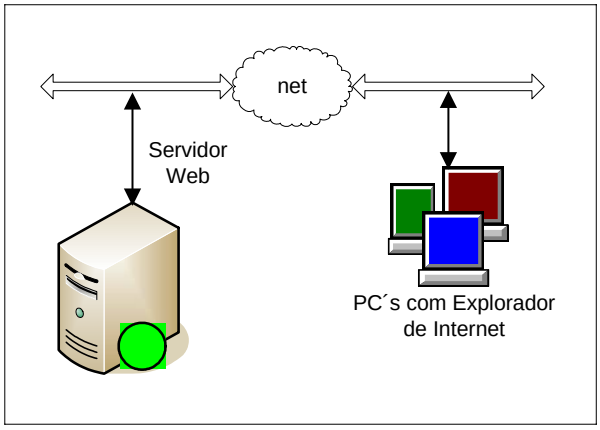

Figura 9 – Arquitectura funcional do perfil professor e de aluno

A Figura 9 apresenta a arquitectura funcional para os perfis de professor e de aluno. A autenticação no acesso à aplicação abcNet é realizada através de um *login* de acesso que é em tudo análogo ao apresentado para o perfil de administrador. Notar que na Figura 9 o servidor de base de dados não se encontra representado. O servidor da base de dados pode estar integrado numa das arquitecturas estruturais anteriormente apresentadas.

5.2 Ferramentas de Desenvolvimento

As ferramentas de desenvolvimento seleccionadas foram: SQL-Server 2000, Visual Studio.NET e Macromedia Flash. Nos parágrafos seguintes apresentam-se, de uma forma resumida, os argumentos utilizados para a sua selecção.

5.2.1 SQL Server 2000

Apesar de todas as desvantagens enunciadas para os SGBD-R, a opção recaiu na selecção do SQL-Server 2000 da Microsoft [MS-SQL]. A opção por SQL-Server teve por base os seguintes critérios:

- satisfação dos requisitos de abcNet;
- melhor integração com o IDE Visual Studio.Net;
- futura versão de SQL-Server (SQL Server 2005) [MS-SQL05] totalmente integrada na plataforma .NET;
- custos de aquisição de licenças;
- curva de aprendizagem.

SQL-Server 2000, além dos tipos de dados normais utilizados nos SGBD-R, suporta também alguns tipos de dados adicionais [MS-SQLDT] tais como *image*, *binary* e *table*.

5.2.2 Visual Studio.NET

Visual Studio.NET é o IDE da Microsoft para o desenvolvimento de aplicações para a plataforma .NET. Visual Studio.Net, além de estar vocacionado para a plataforma .NET, contém um conjunto muito poderoso de ferramentas de desenvolvimento e de *debug* de aplicações. Visual Studio.Net disponibiliza inclusivamente um interface integrado para o SQL-Server, o que no presente caso vem de encontro aos nossos requisitos.

5.2.3 Macromedia Flash

O Flash [MM-FLASH] é a ferramenta da Macromedia que permite o desenvolvimento de interfaces dinâmicos e interactivos para a *Web*. Flash é a tecnologia líder no mercado de desenvolvimento de aplicações gráficas vectoriais para a *Web*. O Flash vai ser utilizado em abcNet de uma forma muito limitada, estando de momento restringido à elaboração dos menus de abcNet. É previsível que em futuras versões de abcNet se atribua um papel de realce ao Flash devido às suas potencialidades gráficas.

6 Módulo de Base de Dados

6.1 Introdução

O desenvolvimento da aplicação abcNet, foi dividido em 3 módulos, a saber: módulo de base de dados (MódBD), módulo de administração (MódAdm) e módulo de formação (MódFor). Cada um dos módulos pode englobar um ou mais projectos. A arquitectura geral de abcNet encontra-se apresentada na Figura 10. Podemos verificar que os MódFor e MódAdm não comunicam entre si, mas partilham o MódBD como recurso comum.

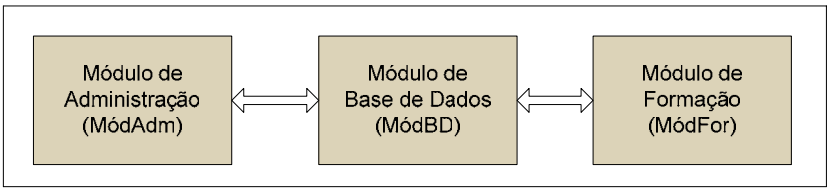

Figura 10 – Arquitectura geral de abcNet

O MódBD engloba a concepção e modulação de uma base de dados que satisfaça os requisitos de abcNet. Para além da base de dados em si mesma, o MódBD também acumula a responsabilidade da disponibilização dos mecanismos de acesso à base de dados. Os mecanismos definidos para o acesso à base de dados são:

- procedimentos armazenados;
- projecto de implementação de um interface orientado ao objecto para acesso aos procedimentos armazenados (PIOOBD).

Foi definido um conjunto de normas [OSCAR-AC1] que permite um acesso orientado ao objecto à informação residente na base de dados.

Além dos mecanismos de acesso definidos para acesso à base de dados, o MódBD engloba também um conjunto de ferramentas de teste aos mesmos mecanismos que se encontram concentrados no projecto de teste da interface da base de dados (PTIBD).

A Figura 11 apresenta a arquitectura geral do MódBD onde se podem distinguir os blocos enunciados: BD, PIOOBD e PTIBD.

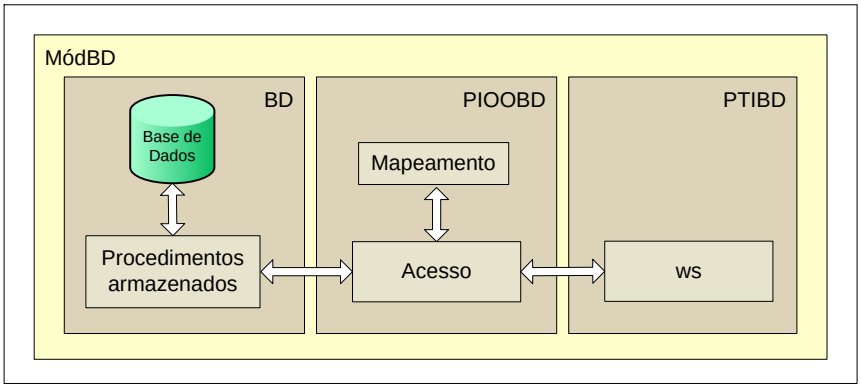

Figura 11 – Arquitectura geral do MódBD

6.2 Projecto da Base de Dados

O projecto da base dados engloba 2 etapas: 1) a concepção e modulação da BD e 2) os procedimentos armazenados.

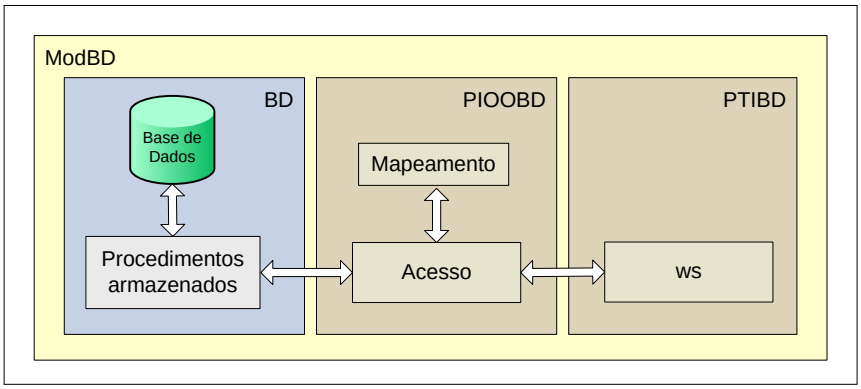

Figura 12 – Projecto da Base de Dados

6.2.1 Concepção e Modulação da Base de Dados

A Concepção e Modulação da Base de Dados engloba todo o trabalho realizado para a criação da base de dados de suporte à aplicação abcNet. O trabalho realizado engloba as seguintes etapas:

- implementação do modelo conceptual da base de dados;
- implementação do modelo lógico da base de dados;
- implementação do modelo físico da base de dados;
- definição dos mecanismos de segurança.

6.2.1.1 Modelo Conceptual

No modelo conceptual pretende-se desenvolver um modelo da base de dados que seja independente de qualquer dependência associada ao SGBD que vier a ser utilizado. Por dependência deve-se entender não

só o SGBD específico a utilizar, Oracle, SQL-Sever, MySQL, etc., como também o tipo de SGBD a utilizar isto é, se é relacional, hierárquico, orientado a objectos, etc.

O desenvolvimento do modelo conceptual engloba as seguintes etapas [CONN-DB, pág. 231]:

- definição das vistas por utilizador;
- por vista:
    - identificação dos tipos de entidades;
    - identificação dos relacionamentos entre entidades;
    - identificação e associação de atributos quer aos tipos de entidades quer aos relacionamentos;
    - determinação do domínio de cada atributo;
    - identificação dos atributos para as chaves candidatas e para as chaves primárias;
    - especialização e generalização de tipos de entidades;
    - elaboração do diagrama de Chen (opção seleccionada);
    - validação do modelo com os utilizadores.

Para além das etapas enunciadas também se deve elaborar por vista uma lista de transacções a satisfazer pelo modelo.

6.2.1.2 Modelo Lógico

No modelo lógico pretende-se desenvolver um modelo da base de dados que já tem em consideração o tipo de base de dados a ser utilizada no modelo físico. O tipo de base de dados a utilizar é baseada no modelo relacional pelo que todo o modelo lógico se irá desenvolver à volta desta topologia.

O desenvolvimento do modelo lógico engloba as seguintes etapas [CONN-DB, pág 244]:

- mapeamento dos dados do modelo conceptual para os dados do modelo lógico:
- remoção de relacionamentos N:M, complexos e recursivos;
- remoção de relacionamentos com atributos;
- remoção de atributos multi-valor;
- reexaminação dos relacionamentos 1:1;
- definição, quando necessário, das chaves candidatas, primárias e estrangeiras;
- normalização com recurso pelo menos às 3 primeiras formas normais;
- definição das restrições de integridade;

- apresentação do diagrama E-R.

Para além das etapas enunciadas também se deve validar os modelos de transacções definido no modelo conceptual.

6.2.1.3 Modelo Físico

No modelo físico pretende-se desenvolver um modelo da base de dados que efectue a implementação real da mesma. A opção tomada recaiu sobre o SQL-Server 2000, pelo que esta fase pode ser realizada recorrendo a várias metodologias, nomeadamente:

- a partir da ferramenta utilizada para o modelo lógico, caso seja possível;
- pela utilização do *Enterprise Manager* para a criação interactiva;
- pela utilização de scripts;
- pela codificação a partir de uma aplicação especialmente desenvolvida para o efeito.

No caso presente, o modelo físico foi desenvolvido com recurso quer ao *"Enterprise Manager*" quer ao ambiente que se encontra integrado no IDE do Visual Studio.Net. Criaram-se não só todas as relações mas também todos os restantes mecanismos essenciais à base de dados, nomeadamente: chaves primárias, chaves estrangeiras, índices únicos, constrangimentos de atributos, domínios de atributos, etc.

6.2.1.4 Normalização de Designações

A atribuição de designações quer às relações quer aos atributos seguiram um conjunto de critérios que se aplicam a todas eles. Os critérios aqui utilizados tiveram em consideração dois objectivos a atingir:

- facilidade de interpretação das designações pela simples leitura das designações;
- satisfação dos requisitos a anunciar para implementação de um mecanismo orientado ao objecto para acesso à base de dados.

Relações

A designação a atribuir a uma relação é sempre constituída por um prefixo 'tb' seguido de um ou mais elementos descritivos da mesma. A primeira letra de cada elemento deve ser maiúscula. Como exemplo consideremos uma relação que descreva códigos postais cuja designação poderia ser 'tbZipCode'.

Atributos

A designação a atribuir a cada atributo é única (não se repete em nenhuma outra relação) e é constituída por um prefixo correspondente à relação a que pertence, seguida de um ou mais elementos descritivos da mesma. O prefixo correspondente ao nome da relação é separado dos restantes elementos

por '_'. Tal como nas relações, a primeira letra de cada elemento deve ser maiúscula. Como exemplo consideremos o caso da relação 'tbZipCode' que contém os seguintes atributos: 'tbZipCode_code', 'tbZipCode_zone', 'tbPlace_placeName'. Pela designação dos atributos depreende-se imediatamente que os dois primeiros atributos são propriedade da própria relação enquanto que o terceiro é proprietário da relação 'tbPlace'.

6.2.2 Procedimentos Armazenados

Os procedimentos armazenados constituem o mecanismo privilegiado de acesso à base de dados de abcNet. Os procedimentos armazenados correspondem às necessidades de acesso à base de dados quer em termos de pesquisa quer em termos de inserção, alteração e de remoção de dados. A estrutura dos procedimentos armazenados segue um conjunto de critérios de modo a normalizar os mecanismos de acesso e de utilização dos mesmos. De entre as várias vantagens, a normalização vai permitir ao PIOOBD a implementação de interfaces também normalizadas de acesso aos procedimentos normalizados. Os critérios de normalização aplicam-se a questões tais como as designações dos procedimentos armazenados, as designações e tipo dos argumentos, os valores de retorno e relações de retorno.

6.2.2.1 Designação dos Procedimentos Armazenados

A designação dos procedimentos armazenados apresenta uma estrutura baseada em elementos, sendo o primeiro elemento sempre formado por 'sp' de *stored procedure*. A primeira letra de cada elemento, excepto a do primeiro, é sempre uma letra maiúscula. Os procedimentos armazenados podem ser de dois tipos: procedimentos armazenados de tabela ou procedimentos armazenados de objectivo. Os procedimentos armazenados de tabela são aqueles que efectuam operações de *Select*, *Insert*, *Update* ou de *Delete* sobre todos os atributos de uma única tabela. Neste caso, a designação dos procedimentos armazenados tem a seguinte estrutura: spXXXXX_YYYYY em que XXXXX representa o nome da tabela e YYYYY representa a operação SQL a ser executada, isto é, *Select*, *Update*, *Delete* ou *Insert*. No caso dos procedimentos armazenados de objectivo, a estrutura da designação é a seguinte: spOOOOO_QQQQQ em que OOOOO é um elemento cujo significado deve ser sugestivo quanto ao objectivo a atingir pelo procedimento armazenado e _QQQQQ, caso necessário, pode representar a operação SQL a ser executada.

6.2.2.2 Designações e Tipos de Argumentos

Os argumentos definidos nos procedimentos armazenados seguem um conjunto de critérios de normalização no que diz respeito quer à sua designação quer ao seu tipo de variável. Os argumentos utilizados nos procedimentos armazenados podem ser de dois tipos: argumentos derivados de atributos ou argumentos livres. Os argumentos derivados de atributos são argumentos com associação directa a atributos de tabelas. Estes argumentos têm a mesma designação que o atributo da tabela a que dizem

respeito e têm, também, quer o mesmo tipo de variável quer a mesma dimensão. Isto significa que se uma tabela 'tbExemplo' possuir um atributo 'tbExemplo_categoria' do tipo *varchar(50)*, então um procedimento armazenado que utilize um argumento derivado desse atributo, o argumento deve ter a seguinte designação e respectiva declaração:

declare tbExemplo_categoria varchar(50)

No caso dos argumentos livres, quer a designação, quer o tipo, quer o comprimento são sujeitos a critérios individuais de definição, condicionados apenas por cada caso particular.

6.2.2.3 Relações de Retorno

Por relações de retorno deve-se entender a relação devolvida por um procedimento armazenado na sequência de uma operação de 'Select'. Um procedimento armazenado não está condicionado nas operações que pode efectuar, podendo efectuar um ou mais de cada um dos quatro tipos de operações *SQL*: *Select*, *Delete*, *Insert*, *Update*. No entanto, no caso de um procedimento armazenado poder devolver o resultado de uma operação de *Select*, isto obriga a que esse procedimento armazenado devolva sempre e obrigatoriamente um resultado de *Select,* resultado esse que obrigatoriamente deve possuir a mesma relação sempre com o mesmo esquema. Esta condição permite que do lado da aplicação se possam desenvolver mecanismos normalizados de tratamento da informação devolvida pelos procedimentos armazenados.

6.2.2.4 Valores de Retorno

Os valores de retorno estão associados a valores devolvidos pelos procedimentos armazenados como resultado da execução de uma dada operação. Assim, se a operação for *Select*, *Update* ou *Delete* o valor de retorno indica quantas linhas foram afectadas pela operação. Se a operação for *Insert*, o valor de retorno poderá ou não ter significado, dependendo se a tabela possui ou não uma chave primária do tipo *identity*. Nesta situação o valor de retorno tem significado e corresponde ao valor da chave primária da linha introduzida na tabela.

6.2.3 Segurança

O acesso à base de dados é condicionado pela identificação do utilizador por intermédio de um nome de utilizador e de palavra chave. Esta informação encontra-se incorporada quer no programa associado à formação quer no programa associado ao administrador, o que permite aos utilizadores um acesso transparente à base de dados. Para além do acesso à base de dados, o acesso à aplicação abcNet está condicionada pela identificação mediante a introdução de um nome de utilizador e de uma palavra chave. Este mecanismo é válido para cada um dos três tipos de utilizadores suportados por abcNet. O perfil de administrador, através do programa de administração, efectua o *login* enquanto que os dois restantes

perfis acedem a abcNet via um login efectuado através do servidor *Web*. O login dos utilizadores tem como objectivo a validação da existência do seu registo como utilizadores de abcNet.

6.3 Projecto de Interface Orientado ao Objecto para a Base de Dados

O Projecto de Interface Orientado ao Objecto para a Base de Dados (PIOOBD), foi desenvolvido tendo como objectivo a disponibilização de um interface orientado ao objecto para acesso à base de dados. O acesso à base de dados é efectuado sempre por intermédio dos procedimentos armazenados. A arquitectura de PIOOBD e sua integração no MódBD encontram-se apresentados na Figura 13.

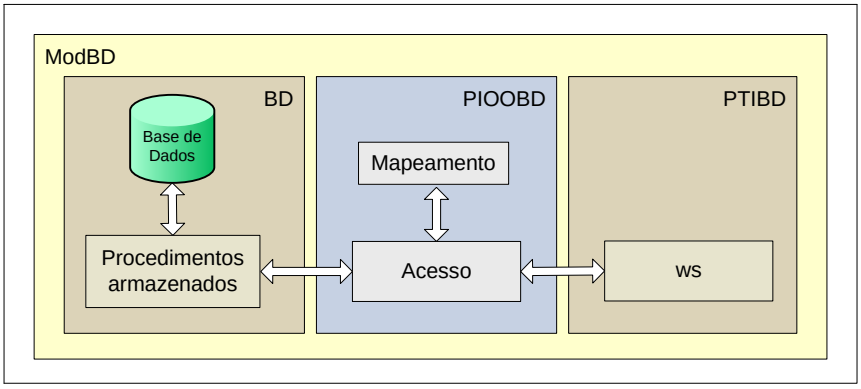

Figura 13 – Projecto de Interface Orientado ao Objecto para a Base de Dados

6.3.1 Problema

A utilização de uma base de dados como componente de armazenamento de informação é de utilidade inquestionável, facilitando não só o armazenamento da informação, como disponibilizando também, entre outros, mecanismos de segurança, de integridade, de manutenção, de acesso e actualização, etc. As questões relativas à implementação da base de dados foram já tratadas em pontos anteriores, focalizando-se agora a atenção na organização dos acesso à base de dados.

Nas descrições que se seguem, apresentam-se casos práticos de modo a facilitar a compreensão dos mecanismos implementados. Para todos os casos apresentados, deve-se ter como base a tabela a seguir descrita:

Nome	Atributos
tbUtilizador	tbUtilizador_nome varchar(12) tbUtilizador_chave varchar(12) tbPerfil_id int(4)

Tabela 3 – Caracterização da tabela a utilizar nos exemplos

Os dois primeiros atributos são propriedades da tabela ‘tbUtilizador’ enquanto que o último atributo é uma chave estrangeira de uma outra tabela denominada de ‘tbPerfil’, em coerência com a normalização definida em 6.2.1.4.

Considere-se também a existência de um procedimento armazenado cujo conteúdo se apresenta na Figura 14. Este procedimento aceita 3 argumentos que são derivados de atributos de tabelas. Este procedimento armazenado, em função dos valores dos argumentos, executa um de dois *queries*, tal como se pode verificar na Figura 14. O critério de decisão sobre qual dos *queries* deve executar está embebido nos valores dos próprios argumentos. Cada argumento tem um valor por defeito que, caso não seja alterado, significa que esse argumento não deve ser tido em consideração aquando da selecção da informação na base de dados. Assim, o primeiro *Select* é apenas executado se os argumentos associados ao nome do utilizador e à sua chave, tiverem valores distintos do valor por defeito e o perfil do utilizador tiver o valor por defeito. No caso do segundo *Select*, este só é executado se os 2 primeiros argumentos tiverem o valor por defeito e o argumento associado ao perfil tiver um valor distinto do valor por defeito.

```
ALTER PROCEDURE dbo.spTbUtilizador_Select
    (
        @tbUtilizador_nome varchar (12) = '',
        @tbUtilizador_chave varchar (12) = '',
        @tbPerfil_id int = -1
    )
AS
    if ( ( @tbUtilizador_nome != '' ) and
         ( @tbUtilizador_chave != '' ) and
         ( @tbPerfil_id = -1 ) )
    Select u.*
    From tbUtilizador u
    Where ( u.tbUtilizador_nome = @tbUtilizador_nome ) and
          ( u.tbUtilizador_chave = @tbUtilizador_chave )

     else if ( ( @tbUtilizador_nome = '' ) and
               ( @tbUtilizador_chave = '' ) and
               ( @tbPerfil_id != -1 ) )
    Select u.*
    From tbUtilizador u
    Where ( u.tbPerfil_id = @tbPerfil_id )

    RETURN
```

Figura 14 – Procedimento armazenado de selecção de dados da tabela ‘tbUtilizador’

As questões que agora se colocam estão relacionadas com a implementação dos mecanismos de acesso quer ao procedimento armazenado aqui apresentado, quer à relação que por ele é devolvida. Este

procedimento armazenado encontra-se normalizado quer no que diz respeito à sua designação, quer aos seus argumentos, quer relativamente à relação devolvida e respectivo esquema.

Invocação dos procedimentos armazenados

A primeira questão tem a ver com a implementação de mecanismos normalizados que permitam à aplicação um acesso fácil ao procedimento armazenado apresentado. Numa primeira aproximação, poder-se-ia implementar um método que tivesse como argumentos os três argumentos do procedimento armazenado e que efectuasse a implementação necessária para a invocação do procedimento armazenado, tal como se apresenta na Figura 15.

```
private DataSet spUtilizador( string nome, string chave, int perfilId )
{
    // inicialização
    SqlCommand cmd = new SqlCommand();
    cmd.CommandText = "spTbUtilizador";
    cmd.CommandType = System.Data.CommandType.StoredProcedure;
    cmd.Connection = conn;
    // argumentos para spTbUtilizador
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbUtilizador_nome", System.Data.SqlDbType.VarChar, 12 ) );
    cmd.Parameters[ "@tbUtilizador_nome" ].Value = nome;
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbUtilizador_chave", System.Data.SqlDbType.VarChar, 12 ) );
    cmd.Parameters[ "@tbUtilizador_chave" ].Value = chave;
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbPerfil_id" , System.Data.SqlDbType.Int, 4,
        System.Data.ParameterDirection.Input, false,
        ((System.Byte)(10)), ((System.Byte)(0)), "",
        System.Data.DataRowVersion.Current, null ) );
    cmd.Parameters[ "@Perfil_id" ].Value = perfilId;
    // execução
    SqlDataAdapter da = new SqlDataAdapter();
    da.SelectCommand = cmd;
    DataSet ds = new DataSet();
    da.Fill( ds );
    return ds;
}
```

Figura 15 – Primeira aproximação para implementação de um mecanismo de acesso

Este método está dividido em três blocos: 1) inicialização, onde são efectuadas algumas inicializações, nomeadamente a identificação do procedimento armazenado e a ligação à base de dados, 2) argumentos, onde é preparada a estrutura associada aos argumentos a transferir para o procedimento armazenado, 3) execução, acção de invocação do procedimento armazenado. A opção pela solução da passagem de todos os argumentos para o método implica que, ao nível dos valores por defeito, que são utilizados para os argumentos nos procedimentos armazenados, o controlo seja da responsabilidade da aplicação. Esta opção poderia, no futuro, acarretar algumas dificuldades em termos de manutenção, pelo que a

responsabilidade da gestão dos valores por defeito dos argumentos deve estar fora da alçada da aplicação. A utilização dos valores por defeito, ao nível dos procedimentos armazenados, significa apenas que esses argumentos não devem ser tidos em conta nos critérios de selecção. Assim sendo, não faz sentido que se transfira para a aplicação a responsabilidade e gestão de valores que ela própria não necessita de tratar.

Uma solução possível consiste na definição de vários métodos, cada um aceitando apenas os argumentos que são efectivamente utilizados pelo procedimento armazenado, repassando a responsabilidade dos valores por defeito para esse mesmo método. Assim, para o procedimento armazenado apresentado na Figura 14, passaríamos a ter dois métodos de acesso, tal como se apresenta nas Figura 16 e Figura 17, e não apenas um como apresentado na Figura 15.

```
private DataSet spUtilizador( string nome, string chave )
{
    // inicialização
       //...
    // argumentos para spTbUtilizador
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbUtilizador_nome", System.Data.SqlDbType.VarChar, 12 ) );
    cmd.Parameters[ "@tbUtilizador_nome" ].Value = nome;
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbUtilizador_chave", System.Data.SqlDbType.VarChar, 12 ) );
    cmd.Parameters[ "@tbUtilizador_chave" ].Value = chave;
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbPerfil_id" , System.Data.SqlDbType.Int, 4,
        System.Data.ParameterDirection.Input, false,
        ((System.Byte)(10)), ((System.Byte)(0)), "",
        System.Data.DataRowVersion.Current, null ) );
    cmd.Parameters[ "@Perfil_id" ].Value = -1; // por defeito
    // execução
       //...
}
```

Figura 16 – Interface de acesso para selecção por nome e chave de utilizador

A organização dos métodos por argumentos vem acentuar um outro problema que ainda não foi colocado. Regressando à Figura 15, relativamente ao bloco ‘argumentos’, podemos verificar a utilização de um argumento (‘tbPerfil_id’) associado a um atributo de uma tabela que não a ‘tbUtilizador’. Esta é uma situação corrente, e significa que um mesmo argumento pode estar disperso por um ou mais procedimentos armazenados. Com a pulverização do número de métodos, esta situação vem-se agravar acentuadamente, tal como já se pode verificar nos blocos dos argumentos dos métodos apresentados nas Figura 16 e Figura 17, em que todos os argumentos se encontram duplicados em termos de passagem de valores para o procedimento armazenado. Esta situação levanta várias questões das quais se destacam:

- repetição de blocos de código em diversos locais;
- dificuldades acrescidas na manutenção sempre que ocorrer uma alteração num argumento.

O ideal seria colocar a definição do argumento num dicionário público e disponível para utilização sempre que necessário, evitando deste modo as questões levantadas. Esta abordagem permite evitar os constrangimentos anteriormente referidos.

```
private DataSet spUtilizador( int perfilId )
{
    // inicialização
       //...
    // argumentos para spTbUtilizador
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbUtilizador_nome", System.Data.SqlDbType.VarChar, 12 ) );
    cmd.Parameters[ "@tbUtilizador_nome" ].Value = ""; // por defeito
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbUtilizador_chave", System.Data.SqlDbType.VarChar, 12 ) );
    cmd.Parameters[ "@tbUtilizador_chave" ].Value = ""; // por efeito
    cmd.Parameters.Add( new System.Data.SqlClient.SqlParameter
        ("@tbPerfil_id" , System.Data.SqlDbType.Int, 4,
        System.Data.ParameterDirection.Input, false,
        ((System.Byte)(10)), ((System.Byte)(0)), "",
        System.Data.DataRowVersion.Current, null ) );
    cmd.Parameters[ "@Perfil_id" ].Value = perfilId;
    // execução
       //...
}
```

Figura 17 - Interface de acesso para selecção por perfil de utilizador

Acesso à relação devolvida pelo procedimento armazenado

A segunda questão está relacionada com a forma como se efectua o acesso a cada atributo de cada linha da relação devolvida no *DataSet* [MS-ADODS]. ADO.NET disponibiliza algumas formas nomeadamente com recurso aos esquemas dos objectos. Apesar de ser uma boa solução, iremos também apresentar posteriormente uma solução própria para esta questão. A solução aqui proposta disponibiliza também um interface orientado ao objecto para cada atributo da relação.

Resumo

Resumindo, pode-se afirmar que as questões globais aqui levantadas e por solucionar são as seguintes:

- interfaces de invocação: interfaces disponibilizados à aplicação para invocação dos procedimentos armazenados;
- interfaces de atributos: interfaces disponibilizados à aplicação para acesso aos valores dos atributos de cada linha da relação obtida por intermédio dos interfaces de invocação.

Estes aspectos vão ser desenvolvidos nos próximos parágrafos.

6.3.2 Arquitectura de PIOOBD

A arquitectura de PIOOBD assenta em dois projectos, tal como se pode verificar na Figura 18: 1) 'Mapeamento': dirigido ao mapeamento das tabelas da base de dados , 2) 'Acesso' dirigido ao interface orientado ao objecto que é disponibilizado à aplicação para acesso à base de dados.

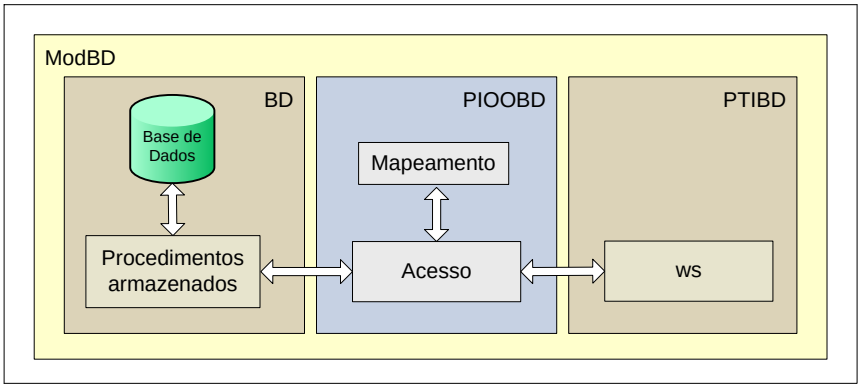

Figura 18 – Projecto de Interface Orientado ao Objecto para acesso à Base de Dados

6.3.3 Mapeamento

O projecto 'Mapeamento' implementa um mapeamento das tabelas da base de dados, disponibilizando a informação necessária para a sua caracterização numa estrutura orientada ao objecto. Para cada tabela da base de dados existe uma classe cuja estrutura se encontra normalizada não só quanto à forma mas também quanto ao conteúdo. A designação atribuída a cada classe coincide com o nome da tabela a que se encontrada relacionada. Assim, se uma tabela tiver a designação 'tbUtilizador' a classe tem também a mesma designação. No PIOOBD, o recurso à utilização destas classes é inevitável e permanente, pelo que se torna necessário ter cuidados redobrados no aspecto relativo aos recursos exigidos pelas mesmas. Tendo em consideração os serviços a prestar por estas classes, foi possível tomar algumas opções de convergência no sentido da minimização de recursos, nomeadamente a utilização de métodos estáticos e sempre que possível de entidades *struct* [MS-STRCT] em vez de classes. Nas descrições que se seguem apresentam-se casos práticos de modo a facilitar a compreensão dos mecanismos implementados. Tal como já referido anteriormente, para todos os casos apresentados, deve-se ter como base a Tabela 3.

Os dois primeiros atributos são propriedade da tabela 'tbUtilizador' enquanto que o último é uma chave estrangeira de uma outra tabela denominada de 'tbPerfil'.

A classe que efectua o mapeamento da tabela 'tbUtilizador', disponibiliza a informação que se passa a descrever:

Nome da tabela

O nome da tabela é disponibilizado por intermédio de uma propriedade pública, estática, tipo *string* com a estrutura apresentada na Figura 19. Esta propriedade tem sempre a designação *TableName*.

```
public static string TableName
{
    get { return "tbUtilizador"; }
}
```

Figura 19 – Método associado ao nome da tabela

Nome dos atributos

Tal como o nome da tabela, a classe disponibiliza também o nome de todos os atributos da tabela, inclusivamente os relativos a outras tabelas, como é o caso das chaves estrangeiras. Cada atributo é disponibilizado por intermédio de uma propriedade pública, estática e tipo *string*, com a estrutura apresentada na Figura 20.

```
public static string UtilizadorNome
{
    get { return TableName + "_nome"; }
}
public static string UtilizadorChave
{
    get { return TableName + "_chave"; }
}
public static string PerfilId
{
    get { return tbPerfil.tbPerfilId; }
}
```

Figura 20 - Propriedades associadas a nome de atributos

A designação das propriedades é derivada directamente do nome do atributo da relação, com as seguintes alterações:

- o prefixo ‘tb’ é suprimido;
- o separador ‘_’ entre o nome da relação e o sufixo de identificação do atributo é removido, passando a primeira letra do sufixo a ser maiúscula.

A Figura 20 apresenta as propriedades da classe que estão associadas à designação dos atributos da tabela ‘tbUtilizador’. Podemos constatar que os atributos próprios da tabela ‘tbUtilizador’ e o atributo

referente à tabela ‘tbPerfil’ são tratados de forma diferente. O nome dos atributos próprios da tabela ‘tbUtilizador’ são formados pelo prefixo que identifica o nome da tabela e pelo sufixo específico de identificação do atributo na tabela, separados por ‘_’. Esta é a estrutura que foi apresentada como estrutura base de definição das designações dos atributos de tabelas. Relativamente ao atributo associado à tabela ‘tbPerfil’, a propriedade limita-se a devolver a estrutura que está definida para o atributo e que se encontra na classe que caracteriza a referida tabela, neste caso ‘tbPerfil’. Desta forma, caso se verifique a alteração de uma designação de um atributo de uma qualquer tabela, apenas se torna necessária uma alteração numa única classe que é a que caracteriza a tabela. Visto as propriedades serem do tipo *static*, não é necessário efectuar a sua instanciação, o que facilita imenso a sua utilização e simultaneamente exige menos recursos quer em termos de memória quer em termos de processamento.

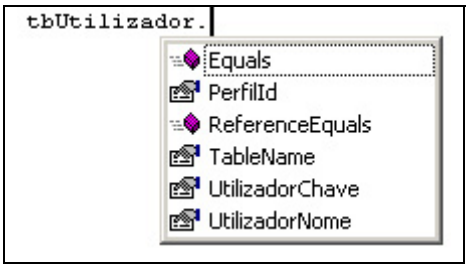

Figura 21 – Janela com propriedades da classe ‘tbUtilizador’

A inclusão de todos os atributos de uma tabela na classe de mapeamento associada, nomeadamente os referentes a outras tabelas, tal como o caso do atributo ‘tbPerfil_id’, não é obrigatória. A opção foi tomada tendo em consideração quer factores de coerência entre a estrutura da tabela e da classe a ela associada, quer a factores de ajuda ao utilizador destas classes. Como se pode verificar na Figura 21, a opção tomada proporciona ao utilizador da classe uma visão global de todos os atributos da tabela, o que já não aconteceria se se excluíssem os referentes a outras tabelas, nomeadamente as chaves estrangeiras.

Caracterização do Atributo

A caracterização dos argumentos é um dos elementos essenciais no acesso aos procedimentos armazenados tal como se pode verificar nos blocos associados aos argumentos nas Figura 15, Figura 16 e Figura 17. A caracterização dos argumentos derivados, que correspondem aos atributos de tabelas, encontra-se disponibilizada em propriedades especificamente criadas para o efeito, que são públicas, estáticas, tipo *SqlParameter* [MS-ADOPM] e sem argumento de entrada. *SqlParameter* é o objecto .NET utilizado para caracterização dos parâmetros a passar na invocação de acções sobre base de dados tal como pode ser constatado em várias figuras, nomeadamente nas Figura 15, Figura 16 e Figura 17.

```
public static SqlParameter SqlParam_utilizadorNome
{
    get { return new System.Data.SqlClient.SqlParameter
              ("@" + tbUtilizador.UtilizadorNome,
                System.Data.SqlDbType.VarChar, 12 ); }
}
public static SqlParameter SqlParam_utilizadorChave
{
    get { return new System.Data.SqlClient.SqlParameter
              ("@" + tbUtilizador.UtilizadorChave,
                System.Data.SqlDbType.VarChar, 12 ); }
}
```

Figura 22 – Caracterização dos atributos como parâmetros

A Figura 22 apresenta a caracterização dos atributos 'Nome' e 'Chave' da tabela 'tbUtilizador'. A designação a atribuir a cada propriedade tem a seguinte estrutura normalizada: SqlParam_XXXXXX em que XXXXXX é a designação da propriedade identificadora do atributo, mas cuja primeira letra é minúscula. Podemos também verificar que estas propriedades recorrem à propriedade da sua própria classe, que devolve a designação do atributo da tabela, como por exemplo: 'tbUtilizador.UtilizadorNome', no caso de SqlParam_utilizadorNome. A propriedade associada ao atributo 'tbPerfil_id' encontra-se definido e apenas disponível na classe associada à tabela 'tbPerfil' como já explicado anteriormente.

Preparação

Além das propriedades já indicadas, é disponibilizado um método que de alguma forma aglutina toda a informação associada à preparação de um parâmetro a passar para um procedimento armazenado.

```
public static void Prepare_utilizadorNome( SqlCommand cmd, string nome )
{
    cmd.Parameters.Add( tbUtilizador.SqlParam_utilizadorNome );
    cmd.Parameters[ "@" + tbUtilizador.UtilizadorNome ].Value = nome;
}
public static void Prepare_utilizadorChave( SqlCommand cmd, string chave )
{
    cmd.Parameters.Add( tbUtilizador.UtilizadorChave );
    cmd.Parameters[ "@" + tbUtilizador.UtilizadorChave ].Value = chave;
}
```

Figura 23 – Preparação de um parâmetro

A Figura 23 apresenta os métodos para o caso da tabela 'tbUtilizador'. Podemos verificar que pela simples invocação de um método, pode-se executar o conjunto das acções necessárias à preparação de um parâmetro a ser passado para um procedimento armazenado da base de dados. A designação destes métodos apresentam uma estrutura normalizada: Prepare_XXXXXX em que XXXXXX é a designação

da propriedade identificadora do atributo mas cuja primeira letra é minúscula. A estrutura da designação destes métodos é idêntica à estrutura dos métodos 'SqlParam'. Estes métodos executam todas as operações necessárias à passagem de um argumento para os procedimentos armazenados.

Valores Por Defeito

No que diz respeito à caracterização das tabelas, ainda nos falta definir como caracterizar os valores por defeito dos atributos. Ao detectar que um argumento associado a um dado atributo tem o valor por defeito, leva a que o procedimento armazenado tome determinadas acções sobre o mesmo e que dependem de caso para caso. A situação mais vulgar no nosso caso é a de considerar esse argumento como não pertencendo ao critério de filtragem na cláusula *where*. A Figura 24 apresenta as propriedades associadas aos valores por defeito para o caso da tabela 'tbUtilizador'. A designação das propriedades tem a seguinte estrutura normalizada: DefaultValue_XXXXXX em que XXXXXX é a designação do atributo, removida do prefixo da designação da tabela. O valor por defeito para o atributo 'tbPerfil_id' encontra-se definido na classe 'tbPerfil'.

```
public static string DefaultValue_nome
{
    get { return ""; }
}
public static string DefaultValue_chave
{
    get { return ""; }
}
```

Figura 24 – Estrutura dos métodos para os valores por defeito.

A estrutura definida no projecto 'Mapeamento' permite que o método apresentado nas Figura 16 e Figura 17 passe a ter novas versões que se encontram apresentadas nas Figura 25 e Figura 26, respectivamente, cujas melhorias se situam ao nível do bloco dos argumentos e em que:

- o método é muito mais compacto que o anterior;
- a informação respeitante aos argumentos encontra-se definida num dicionário.

Apesar de se encontrar um pouco fora do contexto, vamos apresentar uma versão ainda mais compacta, baseada no projecto 'Mapeamento', para os métodos 'spUtilizador' aqui descritos. Analisando os métodos apresentados nas Figura 25 e Figura 26 verificamos existirem muitas semelhanças nos blocos 'inicialização' e 'execução' e inclusivamente também no bloco 'argumentos' e que facilmente se pode evoluir para uma solução muito mais compacta.

```
private DataSet spUtilizador( string nome, string chave )
{
    // inicialização
    //...
    // argumentos para spTbUtilizador
    tbUtilizador.Prepare_utilizadorNome( cmd, nome );
    tbUtilizador.Prepare_utilizadorChave( cmd, chave );
    tbPerfil.Prepare_perfilId( cmd, tbPerfil.DefaultValue_perfilId );
    // execução
    //...
    return ds;
}
```

Figura 25 – spUtilizador(nome,chave) após integração com o projecto 'Mapeamento'

```
private DataSet spUtilizador( int perfilId )
{
    // inicialização
       //...
    // argumentos para spTbUtilizador
    tbUtilizador.Prepare_utilizadorNome( cmd, tbUtilizador.DefaultValue_nome );
    tbUtilizador.Prepare_utilizadorChave( cmd, tbUtilizador.DefaultValue_chave );
    tbPerfil.Prepare_perfilId( cmd, perfilId );
    // execução
       //...
    return ds;
}
```

Figura 26 – spUtilizador(perfilId) após integração com o projecto 'Mapeamento'

As formas compactas encontram-se apresentadas na Figura 27 e são constituídas com recurso a três métodos, dois dos quais herdados de uma classe base e um outro especificamente desenvolvido para o efeito. Os métodos da classe base implementam exactamente as tarefas que se encontravam definidas nos blocos 'inicialização' e 'execução' e que são comuns a todas as classes que implementam o acesso aos procedimentos armazenados. O método 'param' é o método responsável pela organização de todos os argumentos a passar ao procedimento armazenado e encontra-se apresentado na Figura 28.

```
private DataSet spUtilizador( string nome, string chave )
{
    base.inicializar( "spTbUtilizador" );
    param( nome, chave, tbPerfil.DefaultValue_perfilId );
    base.executar();
}
private DataSet spUtilizador( int PerfilId )
{
    base.inicializar( "spTbUtilizador" );
    param( tbUtilizador.DefaultValue_nome, tbUtilizador.DefaultValue_chave, PerfilId );
    base.executar();
}
```

Figura 27 – Formas compactas de spUtilizador.

A estrutura implementada para o método ‘param’ permite que apenas um único método satisfaça as necessidades de todos os métodos associados a um procedimento armazenado, independentemente do número de argumentos de cada um. O método ‘param’ cria sempre todos os argumentos do procedimento armazenado e atribui-lhes os valores recebidos como parâmetros. Como já foi descrito anteriormente, os métodos de invocação dos procedimentos armazenados podem ou não apresentar como argumentos, todos os argumentos definidos no procedimento armazenado associado. Aos que estiverem em falta deve ser atribuído o valor configurado como sendo o valor por defeito. Assim sendo, cada um dos métodos passa a ter conhecimento de todos os valores a utilizar para todos os argumentos do procedimento armazenado, uns por passagem de valor, outros por defeito. É esta a solução utilizada e que pode ser verificada na Figura 27. O primeiro método recebe os valores de ‘nome’ e de ‘chave’ e utiliza o valor por defeito para o ‘perfilId’ para invocar o método ‘param’. O segundo método recebe o valor de ‘perfilId’ e utiliza os valores por defeito para ‘nome’ e ‘chave’ para invocar o método ‘param’.

```
private void param( string nome, string chave, int perfilId )
{
    tbUtilizador.Prepare_utilizadorNome( cmd, nome );
    tbUtilizador.Prepare_utilizadorChave( cmd, chave );
    tbPerfil.Prepare_perfilId( cmd, perfilId );
}
```

Figura 28 – Método ‘param’

A apresentação aqui efectuada mostra que a implementação dos métodos que permitem o acesso aos procedimentos armazenados pode ser bastante simplificada com o recurso:

- a classes de mapeamento;
- a uma superclasse;
- à criação de um método ‘param’.

6.3.4 Acesso

O projecto ‘Acesso’ implementa o interface a utilizar pela aplicação quer para o acesso à base de dados por intermédio dos procedimentos armazenados quer para o manuseamento dos dados transferidos da base de dados. O interface é disponibilizado por classes que expõem métodos vocacionados para o efeito. O projecto ‘Acesso’ implementa dois tipos de classes: 1) classes de invocação que disponibilizam os métodos para invocação dos procedimentos armazenados e 2) classes de atributos que disponibilizam mecanismos para indexação dos dados recebidos ou enviados para a base de dados.

6.3.4.1 Classes de Atributos

As classes de atributos disponibilizam mecanismos de indexação dos atributos da relação devolvida por um procedimento armazenado. As classes de atributos como que representam os esquemas das relações devolvidas pelos procedimentos armazenados, permitindo um acesso a cada atributo como se se tratasse de uma propriedade de uma classe. Isto significa que para o caso do procedimento armazenado 'spUtilizador' se poderia aceder a uma das linhas da relação por intermédio de mecanismos tais como 'utilizador.nome', 'utilizador.chave' e 'utilizador.perfilId'. As classes de atributos encontram-se normalizadas quer no que diz respeito às suas designações, quer no que diz respeito aos interfaces disponibilizados. A designação das classes de atributos tem a seguinte estrutura 'dbSpXXXXXX_' em que 'dbSp' é um prefixo que indica que se trata de uma classe relacionada com um procedimento armazenado da base de dados; 'XXXXXX' é a designação do procedimento armazenado a que se refere a classe e '_' é um sufixo que permite a distinção entre as designações das classes de atributos e as designações das classes de invocação.

```
public dbSpUtilizador_( DataRow dr )
{
    _nome = (string) dr[ tbUtilizador.UtilizadorNome ];
    _chave = (string) dr[ tbUtilizador.UtilizadorChave ];
    _perfilId = (int) dr[ tbUtilizador.PerfilId ];
}
```

Figura 29 – Interfaces de inicialização de uma classe de atributos

Relativamente aos interfaces (métodos e propriedades públicos) da classe de atributos podemos classificá-los em duas categorias: 1) os de inicialização e os de indexação. Os interfaces de inicialização estão vocacionados para a recepção de valores associados a uma linha de informação da relação devolvida pelo procedimento armazenado. A Figura 29 apresenta uma implementação de um interface de inicialização da classe de atributos associada ao procedimento armazenado 'spUtilizador', que é identificado pelo construtor da própria classe. O interface de inicialização tem como missão principal a transferência de valores de uma linha da relação devolvida, para o interior da classe de atributos. Analisando o corpo do construtor com algum detalhe, verificamos que a indexação utilizada para identificação dos elementos de 'dr' (DataRow [MS-ADODR]) é efectuada com recurso a uma classe do projecto 'Mapeamento' e que já foi apresentada. Relembrando, estas classes disponibilizam, entre outra informação, a designação dos atributos das tabelas da base de dados e logicamente também a designação dos atributos da relação devolvida pelo procedimento armazenado.

Os interfaces de indexação têm como principal missão a disponibilização de um acesso orientado ao objecto para cada uma dos atributos da relação devolvida pelo procedimento armazenado.

```
public string UtilizadorNome
{
    get { return _nome; }
}
public string UtilizadorChave
{
    get { return _chave; }
}
public int UtilizadorPerfilId
{
    get { return _perfilId; }
}
```

Figura 30 – Propriedades de indexação de uma classe de atributos

A Figura 30 apresenta a implementação para o caso do procedimento armazenado 'spUtilizador'. Podemos verificar que os valores devolvidos pelos interfaces de atributos são os valores (variáveis) previamente guardados no interface de inicialização. A designação destes interfaces segue a estrutura normalizada XXXXXXYYYYYY em que XXXXXX é a designação da tabela com exclusão do prefixo 'tb' e YYYYYY é a designação do atributo, com exclusão da designação da tabela e do separador '_'.

```
public class dbSpUtilizador_
{
    private string _nome;
    private string _chave;
    private int _perfilId;

    public dbSpUtilizador_( DataRow dr )
    {
        _nome = (string) dr[ tbUtilizador.UtilizadorNome ];
        _chave = (string) dr[ tbUtilizador.UtilizadorChave ];
        _perfilId = (int) dr[ tbUtilizador.PerfilId ];
    }
    public string UtilizadorNome
    {
        get { return _nome; }
    }
    public string UtilizadorChave
    {
        get { return _chave; }
    }
    public int UtilizadorPerfilId
    {
        get { return _perfilId; }
    }
}
```

Figura 31 – Estrutura geral da classe de atributos para o spUtilizador

A Figura 31 apresenta a estrutura da classe de atributos para o procedimento armazenado 'spUtilizador'. Após a instanciação e inicialização da classe 'dbSpUtilizador_', o acesso aos atributos da relação devolvida pelo procedimento armazenado é conseguido por intermédio de um interface orientado ao objecto como se encontra apresentado na Figura 32.

```
private void classeAtrib( DataRow dr )
{
    dbSpUtilizador_ oUtiliz_ = new dbSpUtilizador_( dr );
    string nome = oUtiliz_.UtilizadorNome;
    string chave = oUtiliz_.UtilizadorChave;
    int perfilId = oUtiliz_.UtilizadorPerfilId;
    // ...
}
```

Figura 32 – Exemplo de aplicação da classe de atributos 'dbSpUtilizador_'

A Figura 33 apresenta o caso correspondente ao da Figura 32 mas sem o recurso à classe de atributos. Podemos verificar que do ponto de vista do programador, mesmo com recurso ao mapeamento, a utilização de uma classe de atributos facilita significativamente quer o seu trabalho de edição quer a percepção dos atributos da relação.

```
private void classeAtrib( DataRow dr )
{
    string nome = (string) dr[ tbUtilizador.UtilizadorNome ];
    string chave = (string) dr[ tbUtilizador.UtilizadorChave ];
    int perfilId = (int) dr[ tbPerfil.PerfilId ];
}
```

Figura 33 – Alternativa sem recurso à classe de atributos

O recurso a uma classe de atributos facilita o trabalho de edição e de percepção dos atributos por via das facilidades integradas no IDE, tal como se apresenta na Figura 34. O recurso a uma classe de atributos tem a desvantagem de ser um consumidor adicional de recursos, nomeadamente de memória e possivelmente também de CPU. A facilidade e rentabilidade na utilização das classes de atributos prevaleceram sobre as desvantagens decorrentes da sua utilização.

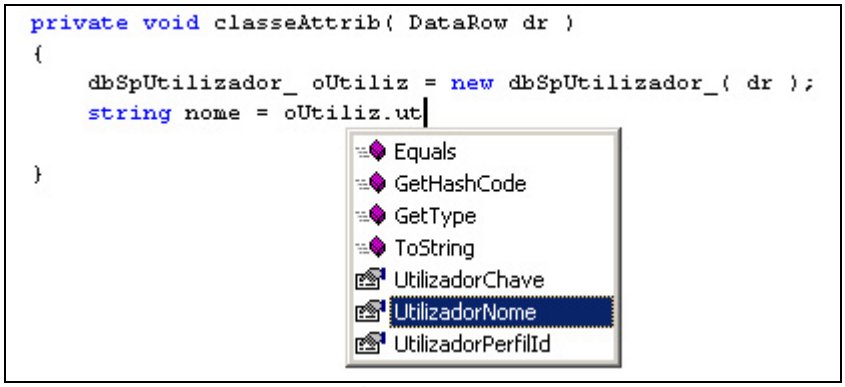

Figura 34 – Ajuda disponível na utilização da classe de atributos

6.3.4.2 Classes de Invocação

As classes de invocação são responsáveis pela implementação e disponibilização dos interfaces necessários à aplicação para o acesso aos procedimentos armazenados da base de dados. As classes de invocação encontram-se normalizadas quer no que diz respeito às suas designações quer aos interfaces disponibilizados quer à implementação da estrutura interna.

A designação das classes de invocação encontra-se normalizada e tem a seguinte estrutura ‘dbSpXXXXXX’ em que ‘dbSp’ é um prefixo que indica que se trata de uma classe relacionada com um procedimento armazenado da base de dados, ‘XXXXXX’ é a designação do procedimento armazenado a que se refere a classe. A estrutura da designação das classes de invocação apenas difere da das classes de atributos pela não existência do sufixo ‘_’.

As classes de invocação disponibilizam essencialmente um interface caracterizado por 2 tipos de métodos: 1) os métodos de invocação, que são vocacionados para a invocação de procedimentos armazenados; 2) os métodos de atributos, que são vocacionados para o acesso à relação devolvida pelo procedimento armazenado, caso tal se verifique.

Métodos de Invocação

Os métodos de invocação disponibilizam interfaces para a invocação dos procedimentos armazenados. Estes métodos encontram-se normalizados no que diz respeito quer à sua designação quer à forma como executam a invocação do procedimento armazenado.

A designação dos métodos de invocação encontra-se normalizada e tem a seguinte estrutura XXXXXX_YYYYYY(aaa, bbb, ...) em que XXXXXX pode assumir uma das quatro designações indicativas da categoria de acção a desencadear sobre a base de dados: Select, Insert, Update ou Delete; YYYYYY é a componente descritiva dos objectivos a atingir; aaa, bbb, ... são os argumentos a utilizar. A Figura 35 apresenta designações possíveis para os dois métodos de invocação do procedimento armazenado ‘spUtilizador’, já apresentado na Figura 27, em que a componente YYYYYY identifica os argumentos utilizados por cada um dos métodos. De referir que a distinção entre métodos sobrecarregados se pode efectuar pelo número e/ou tipos de argumentos, mas não pelo nome dos argumentos. Daí que na designação dos métodos de invocação se tenha optado, sempre que necessário, pela descrição dos argumentos. Aos argumentos não explicitamente indicados no método de invocação são atribuídos os valores tidos por defeito, como se pode verificar na Figura 35, quer no método ‘Select_NomeChave’ para ‘perfilId’, quer no método ‘Select_perfilId’ para ‘nome’ e ‘chave’. Esta metodologia de atribuição dos valores por defeito já tinha sido apresentada anteriormente.

```
public DataSet Select_NomeChave( string nome, string chave )
{
    init( "spUtilizador" );
    param( _cmd, nome, chave, tbPerfil.DefaultValue_perfilId );
    select();
    return _ds;
}
public DataSet Select_perfilId( int perfilId )
{
    init( "spUtilizador" );
    param( _cmd, tbUtilizador.DefaultValue_nome,
           tbUtilizador.DefaultValue_chave, perfilId );
    select();
    return _ds;
}
```

Figura 35 – Métodos de invocação para 'spUtilizador'

Na vertente respeitante à invocação do procedimento armazenado, os métodos de invocação seguem a estrutura compacta apresentada na Figura 27 e que corresponde aos 3 blocos bem distintos, inicialmente apresentados na Figura 16 e Figura 17 e que são 'inicialização', 'argumentos' e 'execução'. A designação aqui utilizada para os métodos correspondentes aos 3 blocos são distintos dos anteriormente utilizados, porém, a estrutura e organização não sofreu qualquer alteração.

```
protected void init( string storedProcedure  )
{
    _cmd = new SqlCommand();
    _cmd.CommandText = storedProcedure;
    _cmd.CommandType = System.Data.CommandType.StoredProcedure;
    _cmd.Connection = _conn;
}
protected void select()
{
    _da = new SqlDataAdapter();
    _da.SelectCommand = _cmd;
    _ds = new DataSet();
    _da.Fill( _ds, TableName );
    _dt = _ds.Tables[ TableName ];
    _count = _dt.Rows.Count;
}
private void param( SqlCommand cmd, string nome, string chave, int perfilId )
{
    tbUtilizador.Prepare_utilizadorNome( cmd, nome );
    tbUtilizador.Prepare_utilizadorChave( cmd, chave );
    tbPerfil.Prepare_perfilId( cmd, perfilId );
}
```

Figura 36 – Métodos que implementam os 3 blocos dos métodos de invocação

A Figura 36 apresenta a implementação dos métodos associados a cada um dos 3 blocos, dos métodos de invocação 'Select_nomeChave' e 'Select_perfilId'. Uma análise mais cuidada ressalta o seguinte:

- os métodos ‘init’ e ‘select’ são idênticos e podem ser partilhados por qualquer método de invocação de qualquer classe de invocação;
- o método ‘param’ é específico para cada classe de invocação.

Estas e outras questões levam-nos a uma implementação das classes de invocação a partir de uma superclasse, da qual herdam uma determinada estrutura que será apresentada posteriormente.

Podemos também verificar que o método ‘param’ recorre às classes do projecto ‘Mapeamento’ para implementar a passagem dos argumentos para os procedimentos armazenados. Os valores recebidos por ‘param’ como argumentos, já traduzem os critérios a serem utilizados pelo procedimento armazenado para a filtragem da informação. Cada um dos métodos de invocação ao activar o método ‘param’ não só replica os valores por si recebidos como argumentos, como atribui aos que estiverem em falta os respectivos valores por defeito, tal como se pode ver na Figura 35.

Métodos de Atributos

Os métodos de atributos são os métodos das classes de invocação que disponibilizam um acesso à informação contida na relação devolvida pelos métodos de invocação. Cada procedimento armazenado no caso de devolver uma relação esta tem sempre com o mesmo esquema. Esta característica permite que cada classe de invocação possa implementar um mecanismo normalizado de acesso à informação contida na relação devolvida.

```
public dbSpUtilizador_ this[ int index ]
{
    get
    {
        dbSpUtilizador_ utiliz = new dbSpUtilizador_( _dt.Rows[ index ] );
        return utiliz;
    }
}
```

Figura 37 – Método de atributos para ‘dbSpUtilizador’

O mecanismo de acesso é implementado com recurso a duas estruturas, uma pertencente ao C# que são os *indexers* [MS-C#IDX] e outra já descrita e implementada, que são as classes de atributos. Os *indexers* permitem um acesso a instâncias de classes como se de *arrays* se tratassem. Isto significa que se determinada classe ‘xpto’ implementar o *indexer*, e se ‘oXpto’ for uma instância de ‘xpto’ então as seguintes expressões são válidas: ‘oXpto[0]’, ‘oXpto[i]’, etc. A Figura 37 apresenta o método de atributos para a classe de invocação ‘dbSpUtilizador’. Pode-se observar que se trata de um método cuja designação é *this* e cuja estrutura engloba características de um *array* devido aos [] e também características de um método porque contém não só um argumento do tipo inteiro mas também porque

retorna um valor. A estrutura de todos os métodos de atributos é a apresentada na Figura 37 e consiste na utilização de um *indexer* que retorna uma instância da classe de atributos associada à classe de invocação, cujas designações apenas diferem na existência ou não do sufixo '_', respectivamente, tal como já foi indicado anteriormente.

Estrutura Global

As classes de invocação foram implementadas recorrendo à utilização de uma superclasse a partir da

```
public abstract class dbSp
{
    protected SqlConnection _conn = null;
    protected SqlCommand _cmd = null;
    protected SqlDataAdapter _da = null;
    protected DataSet _ds = null;
    protected DataTable _dt = null;
    protected DataRow _dr = null;
    protected int _count;

    public dbSp( SqlConnection conn)
    {
        _conn = conn;
    }
    public int count
    {
        get { return _count; }
    }
    protected void init( string storedProcedure  )
    {
        _cmd = null;
        _cmd = new SqlCommand();
        _cmd.CommandText = storedProcedure;
        _cmd.CommandType = System.Data.CommandType.StoredProcedure;
        _cmd.Connection = _conn;
    }
    protected void select()
    {
        _da = null;
        _da = new SqlDataAdapter();
        _da.SelectCommand = _cmd;
        _ds = null;
        _ds = new DataSet();
        _da.Fill( _ds, tableName );
        _dt = _ds.Tables[ tableName ];
        _count = _dt.Rows.Count;
    }
    public abstract string tableName
    {
        get;
    }
}
```

Figura 38 – Estrutura global da classe base das classes de invocação

qual todas as classes de invocação são derivadas. A superclasse, designada por 'dbSp', disponibiliza não só um conjunto de métodos, mas também de dados de modo a que a implementação de cada classe de invocação seja o mais elementar possível.

```
public class dbSpUtilizador: dbSp
{
    public dbSpUtilizador( SqlConnection conn ): base( conn )
    {
    }
    public dbSpUtilizador_ this[ int index ]
    {
        get
        {
            dbSpUtilizador_ utiliz = new dbSpUtilizador_( _dt.Rows[ index ] );
            return utiliz;
        }
    }
    public DataSet Select_NomeChave( string nome, string chave )
    {
        init( "spUtilizador" );
        param( _cmd, nome, chave, tbPerfil.DefaultValue_perfilId );
        select();
        return _ds;
    }
    public DataSet Select_perfilId( int perfilId )
    {
        init( "spUtilizador" );
        param( _cmd, tbUtilizador.DefaultValue_nome,
               tbUtilizador.DefaultValue_chave, perfilId );
        select();
        return _ds;
    }
    private void param( SqlCommand cmd, string nome, string chave, int perfilId )
    {
        tbUtilizador.Prepare_utilizadorNome( cmd, nome );
        tbUtilizador.Prepare_utilizadorChave( cmd, chave );
        tbPerfil.Prepare_perfilId( cmd, perfilId );
    }
    public override string tableName
    {
        get { return "tbUtilizador"; }
    }
}
```

Figura 39 – Estrutura global da classe de invocação 'dbSpUtilizador'

A Figura 38 apresenta a estrutura geral da superclasse 'dbSp' na qual podemos observar:

- variáveis necessárias;
- construtor que recebe informação sobre a ligação a utilizar à base de dados;
- método 'count' que disponibiliza informação sobre o número de linhas na relação devolvida;
- método 'init' que implementa as acções correspondentes ao bloco de inicialização;
- método 'select' que implementa as acções correspondentes ao bloco de execução;

- método abstracto ‘tableName’ para a designação da relação a devolver no *DataSet*.

Após a apresentação da superclasse, falta-nos ainda apresentar a estrutura geral de uma classe de invocação que será materializada recorrendo à nossa já conhecida classe ‘dbSpUtilizador’. A Figura 39 apresenta a estrutura geral da classe de invocação de ‘dbSpUtilizador’. A estrutura aqui apresentada é generalizável para qualquer classe de invocação. Nas situações em que a categoria do procedimento armazenado for distinto do de *Select*, a metodologia aqui apresentada pode ser aplicada de uma forma directa, sendo suficiente a declaração de um método para o efeito e posteriormente o recurso às diversas classes já implementadas para acesso aos procedimentos armazenados.

A Figura 40 apresenta a implementação do método de invocação para a acção de *Insert* na tabela ‘tbUtilizador’ por intermédio do procedimento armazenado ‘spUtilizador_Insert’. Podemos verificar que na essência a estrutura geral se mantém, havendo apenas uma alteração significativa no bloco de execução que tem uma implementação mais simples que a anterior.

```
public void Insert( string nome, string chave, int perfilId )
{
    init( "spUtilizador_Insert" );
    param( _cmd, nome, chave, perfilId );
    _cmd.ExecuteNonQuery();
}
```

Figura 40 – Método de invovação para acção de *Insert*

Da Figura 39 pode-se verificar que a criação de uma nova classe de invocação, sempre que necessário, exige uma estrutura normalizada de fácil compreensão e também de fácil implementação. Reparar que numa fase inicial, quando se criam as primeiras classes de invocação, normalmente é necessário criar todas as restantes classes de suporte à mesma. Numa fase posterior, a criação de uma nova classe de invocação poderá exigir apenas a escrita de uma classe do tipo da apresentada na Figura 39, socorrendo-se de classes já implementadas.

6.3.4.3 Utilização

Após a apresentação e descrição quer das classes de atributos quer das classes de invocação resta-nos descrever, do lado da aplicação, como utilizar estes recursos e se efectivamente esses mesmos recursos vieram introduzir alguma simplificação no acesso à base de dados por parte da aplicação. A Figura 41 apresenta um caso concreto de aplicação de uma classe de invocação e a respectiva classe de atributos. Do lado da aplicação pretende-se obter da base de dados todos os utilizadores de determinado perfil, para o que foi desenvolvido um método privado para o efeito, denominado de ‘lerUtilizadores’. Este método deve obter a informação desejada, recorrendo a uma classe invocação existente para o efeito e que é a nossa já conhecida ‘dbSpUtilizador’. O primeiro passo consiste na instanciação da classe de invocação

'dbSpUtilizador', seguido da chamada do método de invocação correspondente à acção pretendida, que na presente situação é o da obtenção de todos os utilizadores com determinado perfil. Após a transferência da informação da base de dados para a classe de invocação através de uma relação, basta iterar pelo *indexer* para ter acesso a cada um dos utilizadores que satisfizeram o critério de selecção. Visto o *indexer* devolver uma classe de atributos, tal vem permitir que a aplicação possa não só efectuar uma iteração por cada utilizador mas também aceder aos atributos associados a cada utilizador. Desta forma,

```
private void lerUtilizadores( int perfilId, SqlConnection conn )
{
    string nome, chave;
    // instanciar classe de invocação
    dbSpUtilizador utiliz = new dbSpUtilizador( conn );
    // invocar método de activação
    utiliz.Select_perfilId( perfilId );
    // iterar por cada linha da relação devolvida
    for ( int nUtiliz = 0; nUtiliz < utiliz.count; nUtiliz++ )
    {
        // acesso aos atributos da relação devolvida
        nome = utiliz[ nUtiliz ].UtilizadorNome;
        chave = utiliz[ nUtiliz ].UtilizadorChave;
        // código para tratar informação
    }
}
```

Figura 41 – Exemplo de aplicação das classes de invocação e de atributos

pode-se verificar que do ponto de vista da aplicação, a obtenção dos dados dos utilizadores com determinado perfil é conseguida de uma forma lógica e de muito fácil implementação.

6.3.5 Vantagens e Exemplos

Esta metodologia implica que sejam seguidas algumas regras desde a designação das tabelas e seus atributos, à organização dos procedimentos armazenados, ao mapeamento das tabelas, às classes de atributos e classes de invocação. Apesar de parecer um esforço significativo em termos de organização, a experiência vem mostrar que tal esforço, apesar de não ser assim significativo, é altamente compensador em termos de manutenção correctiva e evolutiva. Concretizemos duas situações:

- inclusão de um novo atributo na tabela 'tbUtilizador';
- criação de um novo método de invocação em 'dbSpTbUtilizador'.

Em todos estes exemplos vamos reutilizar o procedimento armazenado 'spUtilizador' e as classes 'tbUtilizador', ''dbSpTbUtilizador' e 'dbSpTbUtilizador_'.

6.3.5.1 Inclusão de um novo atributo

A inclusão de um novo atributo numa tabela implica alterações nas classes 'tbUtilizador' e 'dbSpUtilizador_'. As alterações a afectuar em 'tbUtilizador' e 'dbSpUtilizador_' estão directamente relacionadas com a estrutura da tabela 'tbUtilizador' e apenas reflectem a introdução de um novo atributo. Vamos apresentar o caso da introdução de um atributo, que reflecte a data do último login válido efectuado por cada utilizador, cuja designação na tabela é 'tbUtilizador_dataLogin' e é do tipo *smalldatetime*.

```
else if ( @tbUtilizador_dataLogin != '' )
Select u.*
From tbUtilizador u
Where( u.tbUtilizador_dataLogin = @tbUtilizador_dataLogin )
```

Figura 42 – Alteração a 'spTbUtilizador' para suportar nova selecção

O procedimento armazenado continua a ser o apresentado na Figura 14 mas com o acrescento do código necessário à selecção dos utilizadores que efectuaram *login* em determinada data tal como se apresenta na Figura 42.

A classe de mapeamento 'tbUtilizador' caracteriza essencialmente os atributos associados a uma dada tabela: designação e objectos de passagem de parâmetros para os procedimentos armazenados. A caracterização do novo atributo encontra-se apresentada na Figura 43.

```
public static string UtilizadorDataLogin
{
    get { return TableName + " _dataLogin"; }
}
public static SqlParameter SqlParam_utilizadorDataLogin
{
    get { return new System.Data.SqlClient.SqlParameter
              ( "@" + tbUtilizador.UtilizadorDataLogin,
                System.Data.SqlDbType.DateTime, 4 ); }
}
public static void Prepare_utilizadorDataLogin( SqlCommand cmd, DateTime dt  )
{
    cmd.Parameters.Add( tbUtilizador.SqlParam_utilizadorDataLogin );
    cmd.Parameters[ "@" + tbUtilizador.UtilizadorDataLogin ].Value = dt.Date.ToString();
}
public static string DefaultValue_dataLogin
{
    get { return ""; }
}
```

Figura 43 – Alteração de 'tbUtilizador' para Inclusão de um novo atributo de tabela

De realçar que após a caracterização de um atributo de uma tabela, tal como apresentado na Figura 43, a reutilização desse atributo numa outra tabela qualquer não necessita de nova redefinição da estrutura, pois esta já se encontra definida.

De modo a se ter acesso à informação do novo atributo nas relações devolvidas pelo procedimento armazenado, é necessário que a classe de atributos associada ao procedimento armazenado inclua a sua definição. A Figura 44 apresenta as alterações a efectuar relativamente ao caso inicial e que se encontra apresentado na Figura 29 e na Figura 30. Assim, a inclusão de um novo atributo implica a implementação das alterações aqui apresentadas. Caso a alteração seja a inclusão de um novo atributo numa tabela, que seja por exemplo uma chave estrangeira associada a uma tabela já mapeada, as alterações a introduzir são muito mais reduzidas, não sendo por exemplo necessária a caracterização do atributo em termos de parâmetro, pois este já se encontra caracterizado na tabela de mapeamento associado à sua tabela de origem.

```
public dbSpUtilizador_( DataRow dr )
{
    _nome = (string) dr[ tbUtilizador.UtilizadorNome ];
    _chave = (string) dr[ tbUtilizador.UtilizadorChave ];
    _perfilId = (int) dr[ tbUtilizador.PerfilId ];
    _dataLogin = (DateTime) dr[ tbUtilizador.DefaultValue_dataLogin ];
}
public DateTime UtilizadorDataLogin
{
    get { return _dataLogin; }
}
```

Figura 44 – Alterações à classe de atributos

6.3.5.2 Criação de um novo método de invocação

Na sequência das alterações anteriores, vamos agora definir um novo método de invocação que permita a execução da nova acção de selecção apresentada na Figura 42.

A criação de um novo método de invocação é efectuada de uma forma muito simples e resume-se à definição de um novo método, que neste caso é designado por ‘Select_dataLogin’, que se limita a chamar 3 outros métodos já criados. A definição deste novo método de invocação exige que se efectuem também alterações nos métodos de invocação já definidos, devido à alteração efectuada ao método ‘param’. Assim, cada um dos outros métodos de invocação deverá continuar a chamar ‘param’ mas agora também com o ‘dataLogin’ passado com o valor definido por defeito, que se encontra definido na classe ‘tbUtilizador’. A Figura 45 apresenta quer o novo método que foi criado quer a nova versão do método ‘param’ que já suporta o argumento referente à data de *login*.

A metodologia aqui apresentada, no futuro, pode ser melhorada com a criação de ferramentas de geração semi-automática das classe de mapeamento, de invocação e de atributos.

```
public DataSet Select_dataLogin( DateTime dataLogin )
{
    init( "spTbUtilizador_Select" );
    param( _cmd, tbUtilizador.DefaultValue_nome,
        tbUtilizador.DefaultValue_chave,
        tbPerfil.DefaultValue_perfilId,
        dataLogin );
    select();
    return _ds;
}
private void param( SqlCommand cmd, string nome, string chave, int perfilId,
                    DateTime dataLogin )
{
    tbUtilizador.Prepare_utilizadorNome( cmd, nome );
    tbUtilizador.Prepare_utilizadorChave( cmd, chave );
    tbPerfil.Prepare_perfilId( cmd, perfilId );
    tbUtilizador.Prepare_utilizadorDataLogin( cmd, dataLogin );
}
```

Figura 45 – Alterações à classe de invocação

6.4 Projecto de Teste da Interface Base de Dados

O PIOOBD está vocacionado para a disponibilização de um interface de acesso à base de dados orientado a objectos. Do ponto de vista da aplicação, apenas os métodos de invocação e de atributos se encontram visíveis, permanecendo numa camada inferior toda a restante estrutura que engloba não só o próprio PIOOBD mas também o projecto 'Mapeamento', a ligação à base de dados, os procedimentos armazenados e a própria base de dados. Todas estas estruturas devem estar a funcionar correctamente para que a aplicação possa também funcionar correctamente. Daí a justificação para a existência de um projecto específico para o teste desta fronteira.

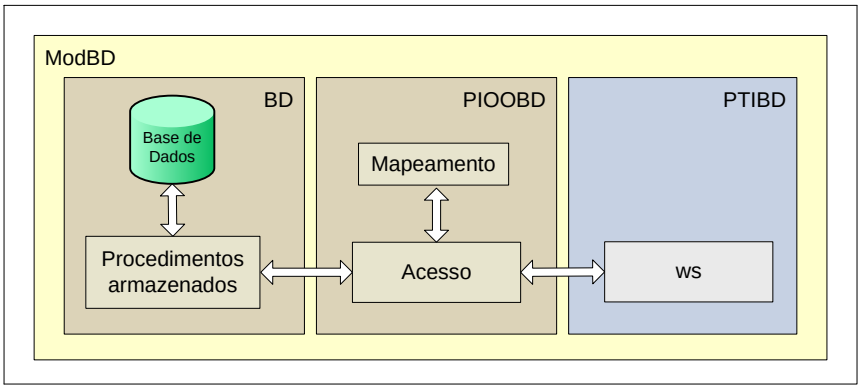

Figura 46 – Projecto de Teste do Interface para a Base de Dados

Como já foi afirmado, a aplicação tem acesso directo às classes de invocação e é através delas que se activam as acções sobre a base de dados e é também através delas que se obtêm os resultados de selecção de informação da base de dados. O PTIBD está vocacionado para o teste deste interface e é composto por apenas um projecto denominado de 'ws', acrónimo de *web service*. Como veremos mais tarde, todo o projecto está assente em WS, daí o nome atribuído. A apresentação do PTIBD e sua integração no MódBD encontra-se apresentados na Figura 46.

6.4.1 Requisitos Funcionais

As classes de invocação do projecto 'Acesso' permitem o acesso ao PIOOBD, disponibilizando mecanismos de interacção com os procedimentos armazenados, através de um interface orientado ao objecto. As classes de invocação recorrem a todas as estruturas implementadas desde o projecto de mapeamento, ligação à base de dados, procedimentos armazenados e a própria base de dados, de modo a poder desempenhar as tarefas que lhes estão incumbidas. As classes de invocação disponibilizam dois mecanismos essenciais à sua utilização: os métodos de invocação e os métodos de atributos.

Os métodos de invocação consistem em métodos que executam acções sobre a base de dados via os procedimentos armazenados. Caso a acção seja uma selecção de informação, o método retorna um *DataSet* contendo a relação devolvida pelo procedimento armazenado.

Os métodos de atributos permitem, através do *DataSet* e das classes de atributos, ter acesso a cada atributo de cada linha da relação devolvida pelo procedimento armazenado.

O desenvolvimento de um projecto que permita, de uma forma eficaz e expedita, o teste deste interface e de tudo o que se encontra abaixo dele, levanta diversas dificuldades. A opção recaiu no teste sobre as classes de invocação e dentro destas, nos métodos de invocação e nos métodos de atributos.

6.4.2 O problema

As estratégias de teste de programas/aplicações ou partes dos mesmos podem ser classificadas em duas distintas categorias: *black box* e *white box* [ROGER]. O teste baseado na estratégia *black box*, também conhecido como teste funcional, é baseado numa técnica de testes por blocos sem qualquer preocupação quanto à implementação interna de cada bloco. A preocupação encontra-se centrada nas entradas e nas saídas do bloco a ser testado. A estratégia *white box*, também conhecida por *glass box*, *structural*, *clear box* ou *open box*, é baseada no conhecimento explícito e detalhado da implementação dos blocos a testar. Globalmente pode-se afirmar que a estratégia *white box* se focaliza no teste do *software*, sem a garantia de que todos os requisitos tenham sido implementados; a estratégia *black box* focaliza-se no teste contra uma especificação.

A abordagem à nossa situação não pode ser implementada com o recurso apenas a uma das estratégias. O teste do funcionamento correcto passa não só pela verificação do funcionamento dos interfaces que são os métodos das classes de invocação, mas também pela verificação e teste dos detalhes da implementação de cada um dos interfaces, incluindo os recursos por eles utilizados, nomeadamente o projecto de mapeamento, procedimentos armazenados, etc. O projecto a desenvolver para o teste assenta numa estratégia *black box* para efectuar o teste dos interfaces das classes de invocação e se necessário, sem recurso a uma estratégia específica *white box*, efectuar o teste detalhado das classes e dos recursos por elas utilizados, recorrendo ao *debugger* disponibilizado pelo Visual Studio.Net.

Assim, o esforço de desenvolvimento recai sobre um projecto a desenvolver e que disponibilize mecanismos de teste dos interfaces das classes de invocação. As classes de invocação disponibilizam essencialmente três tipos de métodos que devem ser utilizados no teste: construtor, métodos de invocação e métodos de atributos. A Figura 47 apresenta os métodos a serem testados para o caso já apresentado e relativo a 'dbSpUtilizador'. Analisemos cada um dos três tipos de métodos.

```
public dbSpUtilizador( SqlConnection conn ): base( conn )
{
}
public dbSpUtilizador_ this[ int index ]
{
    get
    {
        dbSpUtilizador_ utiliz = new dbSpUtilizador_( _dt.Rows[ index ] );
        return utiliz;
    }
}
public DataSet Select_NomeChave( string nome, string chave )
{
    init( "spTbUtilizador_Select" );
    param( _cmd, nome, chave, tbPerfil.DefaultValue_perfilId );
    select();
    return _ds;
}
public DataSet Select_perfilId( int perfilId )
{
    init( "spTbUtilizador_Select" );
    param( _cmd, tbUtilizador.DefaultValue_nome,
           tbUtilizador.DefaultValue_chave, perfilId );
    select();
    return _ds;
}
```

Figura 47 - Métodos da classe de invocação a serem testados

Construtor

O construtor apenas exige que seja passado um objecto do tipo *SqlConnection* [MS-ADOCN], não havendo processamento de qualquer tipo de informação. O método limita-se a guardar uma referência para o referido objecto, a qual é posteriormente utilizada no acesso à base de dados.

Métodos de Invocação

Os métodos de invocação encontram-se já descritos e tipificados. Os métodos de invocação exigem a implementação de um interface a dois níveis distintos: ao nível da invocação do método e ao nível da análise dos valores devolvidos, caso se verifique necessário. Ao nível da invocação do método existe a necessidade não só de o invocar pela sua designação mas também de lhe passar os argumentos. Ao nível da análise dos valores devolvidos, os métodos que exigem maior esforço de teste são os relacionados com a acção *Select* devido à dificuldade de análise da informação armazenada numa relação contida num *DataSet.* A análise da informação armazenada numa relação requer um interface que não é fácil de desenvolver, quer pelo número de atributos que essa mesma relação pode conter, quer pela cardinalidade dessa mesma relação, que tanto pode ser fixa como variável, como pode ser de um como de centenas ou milhares. Para além do número de atributos e da cardinalidade, a análise dos valores de cada atributo é também um dado relevante nos testes a efectuar, o que vem aumentar significativamente a dificuldade da implementação de mecanismos de teste.

Métodos de Atributos

Os métodos de atributos implementam um mecanismo de acesso a um subconjunto da informação disponibilizada pelos métodos de invocação. A informação disponibilizada restringe-se aos atributos de uma só linha da relação contida no *DataSet*. As questões levantadas com os métodos de atributos são também um subconjunto das questões colocadas com os métodos de invocação e têm a ver com o número de atributos e a sua validação em termos de valor.

6.4.3 Estratégia

A estratégia a seguir pode ser escolhida de entre várias possibilidades. Vamos apresentar duas possibilidades para a implementação da estratégia, e de seguida justificar a opção tomada a favor de uma delas.

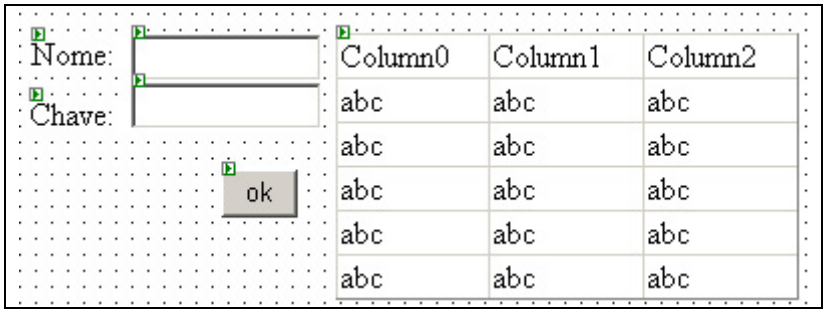

Figura 48 – Formulário de teste dos métodos de invocação

A estratégia a seguir poderia contemplar a criação de formulários que implementem mecanismos quer para introdução de valores a serem passados como argumentos para os métodos de invocação, quer mecanismos para a visualização da informação devolvida pelo procedimento armazenado. A Figura 48 apresenta uma das possibilidades de constituição do formulário de teste do método de invocação 'Select_NomeChave' de 'dbSpUtilizador'.

O formulário inclui:

- duas caixas de texto, uma para inserir o nome do utilizador e outra para inserção da palavra chave do utilizador;
- uma grelha para apresentação da informação devolvida pelo procedimento armazenado;
- um botão para iniciação do processo de acesso ao procedimento armazenado.

O código associado à implementação do teste encontra-se apresentado na Figura 49, onde podemos distinguir três partes distintas:

- instanciação da classe de invocação;
- chamada ao método de invocação;
- preenchimento da grelha com a informação devolvida pelo procedimento armazenado.

```
private void idBtnOk_Click(object sender, System.EventArgs e)
{
    dbSpUtilizador utiliz = new dbSpUtilizador( conn );
    DataSet ds = utiliz.Select_NomeChave( idTxtNome.Text, idTxtChave.Text );
    idDgrUtilizadores.DataSource = ds;
    idDgrUtilizadores.DataMember = utiliz.tableName;
    idDgrUtilizadores.DataBind();
}
```

Figura 49 – Código para implementação do teste.

Esta estratégia requer a implementação de mecanismos que possibilitem a identificação do método de invocação a chamar, pois como é sabido, para cada combinação de valores a utilizar como argumentos, existe um método de invocação específico.

A segunda estratégia proposta recorre a WS para implementação da plataforma de teste. O VS.NET disponibiliza um mecanismo de geração automática de interfaces para acesso e teste de WS, permitindo não só a introdução de valores (argumentos), a activação de métodos, como também uma visualização dos dados devolvidos pelos mesmos.

A Figura 50 apresenta o resultado da chamada a um *Web Method* que por sua vez chamou o método de invocação 'Select_NomeChave' de 'dbSpUtilizador', que devolveu uma relação 'tbUtilizador' com apenas um utilizador cujos atributos são: nome – administrad, chave – 123456, perfil – 1.

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://tempuri.org/">
  + <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  - <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
      xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
    - <NewDataSet xmlns="">
      - <tbUtilizador diffgr:id="tbUtilizador1" msdata:rowOrder="0">
          <tbUtilizador_nome>administrad</tbUtilizador_nome>
          <tbUtilizador_chave>123456</tbUtilizador_chave>
          <tbPerfil_id>1</tbPerfil_id>
        </tbUtilizador>
      </NewDataSet>
    </diffgr:diffgram>
  </DataSet>
```

Figura 50 – Apresentação dos dados devolvidos pelo serviço web

A plataforma disponibilizada prescinde do desenvolvimento de qualquer interface sendo apenas necessário criar uma classe do tipo W*eb Service* e incluir o código que se apresenta na Figura 51.

```
[WebMethod( Description="Obtém utilizador identificado pelo nome e palavra chave." ) ]
public DataSet Select_NomeChave( string nome, string chave )
{
    dbSpUtilizador utiliz = new dbSpUtilizador( conn );
    return utiliz.Select_NomeChave( nome, chave );
}
```

Figura 51 – Implementação do serviço para teste do método de acesso

Quer uma quer outra metodologia apresentam vantagens e desvantagens. A partir da metodologia baseada nos WS, as vantagens e desvantagens são:

Vantagens:

- não é necessário desenvolver qualquer interface ou formulário;
- uma só classe *Web Service* pode conter os testes de todos os métodos de uma classe de invocação;
- é de mais fácil organização e manutenção;
- potencia futuros desenvolvimentos nesta área.

Desvantagens:

- a apresentação da informação devolvida pelo procedimento armazenado de mais difícil leitura.

A opção recaiu sobre a estratégia associada aos *Web Services* pelas vantagens que apresenta, apesar de a desvantagem da apresentação da informação, em algumas situações, poder ser muito significativa, como são as situações em que a informação é a contida num *DataSet* com várias tabelas cada uma com muitas linhas.

6.4.4 Projecto *ws*

O projecto *ws* – PWS – destina-se ao teste do interface disponibilizado pelo projecto 'Acesso' e que engloba todas as estruturas por ele utilizadas que vai desde o projecto 'Mapeamento' aos procedimentos armazenados. A sua estrutura é baseada em WS e assenta na disponibilização de interfaces gráficos para a execução dos referidos testes. A estrutura dos WS implementados, segue um conjunto de critérios de modo a normalizar os mecanismos de acesso e de utilização dos mesmos. Os critérios de normalização aplicam-se a questões tais como as designações das classes, designação dos métodos e implementação dos testes.

Designação das Classes

Cada WS está implementado numa classe, podendo conter um ou mais *We*b *Methods*. Cada *Web Service* implementa todos os mecanismos de teste para uma única classe de invocação. A designação da classe que presta o serviço tem a seguinte estrutura: wsXXXXXX em que 'ws' é um prefixo identificador de *web service* e XXXXXX é a designação da classe de invocação a testar. Assim, para o caso que temos vindo a tratar, a designação do serviço *Web* associado é 'wsDbSpTbUtilizador'.

Designação dos Métodos

A designação atribuída a cada um dos *Web Methods* coincide com a designação dos métodos de invocação a testar. Assim, o *Web Method* associado ao teste do método de invocação 'Select_NomeChave' é também 'Select_NomeChave'. Os argumentos dos *Web Methods* coincidem com os argumentos dos métodos de invocação, com uma excepção. No caso de o teste se estender também ao método de atributos, isto é, pretender-se analisar o conteúdo de uma linha específica da relação devolvida no DataSet, então o *Web Method* requer mais um argumento que corresponde ao índice da linha na relação devolvida pelo procedimento armazenado. De modo a diferenciá-los, foi decidido que a designação destes se diferenciava pela existência de um 'i' após o caracter separador '_'. Na presente situação, o método tem a designação de 'Select_iNomeChave'.

Implementação

Os *Web Methods* implementados permitem testes a dois níveis: ao nível do método de invocação e ao nível do método de atributos. Como já foi referido, quer numa quer noutra situação, a designação base dos *Web Methods* é a mesma. Os *Web Methods* distinguem-se em dois aspectos particulares que são o tipo de valor de retorno e a existência de mais um argumento no caso do teste se efectuar ao nível do método de atributos. Assim, para o caso dos métodos de invocação, o método *Web* retorna um *DataSet* e os seus argumentos coincidem com os argumentos dos métodos de invocação. No caso de o teste se efectuar ao nível dos métodos de atributos, o *Web Method* retorna uma instância da classe de atributos associada, cujo conteúdo é o correspondente à linha indexada na relação existente no *DataSet*.

Analisemos o caso da implementação de um WS para o teste da classe de invocação 'dbSpUtilizador'. O WS tem a designação 'wsDbSpUtilizador' e contém um total quatro *Web Methods*, dois para cada método de invocação da classe de invocação. Dois dos métodos são vocacionados para o teste ao nível dos métodos de invocação sendo os restantes vocacionados para o teste ao nível dos métodos de atributos. O teste ao nível dos métodos de atributos também engloba o teste ao nível do método de invocação, pois este tem obrigatoriamente que ser invocado antes do método de atributos.

```
[WebMethod( Description="Obtém utilizador identificado pelo nome e palavra chave." ) ]
public DataSet Select_NomeChave( string nome, string chave )
{
    dbSpUtilizador utiliz = new dbSpUtilizador( conn );
    return utiliz.Select_NomeChave( nome, chave );
}
[WebMethod( Description="Obtém utilizadores identificados por determinado perfil." ) ]
public DataSet Select_PerfilId( int perfilId )
{
    dbSpUtilizador utiliz = new dbSpUtilizador( conn );
    return utiliz.Select_perfilId( perfilId );
}
```

Figura 52 – Métodos web associados aos métodos de invocação.

A Figura 52 apresenta os dois *Web Methods*, cada um associado a um método de invocação. As suas designações coincidem com as dos métodos de invocação e os argumentos também, retornando cada um deles um *DataSet*. A Figura 53 apresenta os dois *Web Methods*, cada um associado a um método de atributos e que retorna uma instância da classe de atributos. Como se pode verificar, as suas designações coincidem com as dos métodos de invocação, a menos da alteração já apresentada, e os argumentos, que além dos argumentos exigidos pelos métodos de invocação, inclui também um índice de selecção de linha da relação devolvida pelo procedimento armazenado. Ao contrário dos *Web Methods* associados a métodos de invocação, estes métodos retornam, através do método do *indexer*, uma instância da classe de atributos.

```
[WebMethod( Description="Obtém utilizador identificado pelo" +
                        " nome e palavra chave e índice." ) ]
public dbSpUtilizador_ Select_iNomeChave( string nome, string chave, int índice )
{
    dbSpUtilizador utiliz = new dbSpUtilizador( conn );
    utiliz.Select_NomeChave( nome, chave );
    return utiliz[ índice ];
}
[WebMethod( Description="Obtém utilizador identificado por perfil e índice." ) ]
public dbSpUtilizador_ Select_iPerfilId( int perfilId, int índice )
{
    dbSpUtilizador utiliz = new dbSpUtilizador( conn );
    utiliz.Select_perfilId( perfilId );
    return utiliz[ índice ];
}
```

Figura 53 – Métodos web associados a métodos de atributos

Vejamos agora qual o interface disponibilizado ao utilizador para efectuar determinado teste à classe de invocação 'dbSpUtilizador' e que utiliza o WS 'wsDbSpUtilizador'. Todo o controlo é efectuado dentro do Visual Studio.Net não havendo necessidade de qualquer intervenção do exterior. Os procedimentos a executar são os seguintes:

- definir o projecto 'ws' como projecto de arranque : *Set as StartUp Project*;
- definir 'wsDbSpUtilizador' como página de arranque: *Set as Start Page*;
- iniciar *debug* premindo a tecla F5, cujo resultado se encontra na apresentado na Figura 54; aqui podemos verificar a existência de 5 *links*, em que quatro correspondem aos quatro *Web Methods* do WS e o quinto (*Service Description*) corresponde a um *link* para uma página que apresenta a descrição do serviço (WSDL) recorrendo ao formato XML;

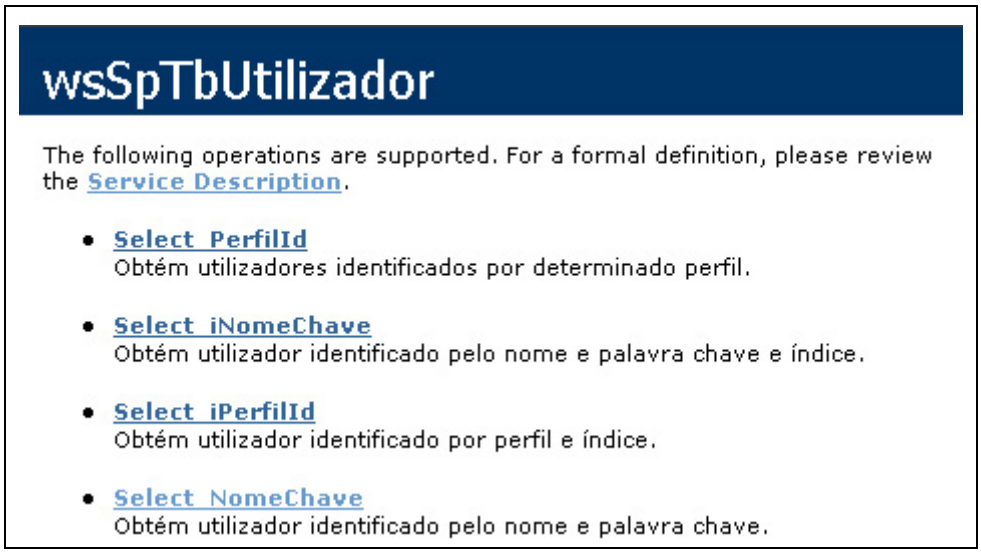

Figura 54 – Interface disponibilizado após F5

- seleccionar o *Web Method* que se pretende accionar, 'Select_PerfilId', para efectuar o teste do respectivo método de invocação; ao clicar no *link* associado ao *Web Method*, é aberta uma janela do explorador de *Internet* cujo conteúdo é o apresentado na Figura 55; nesta figura podemos verificar a existência quer de uma caixa de texto destinada à introdução de um valor para o argumento 'perfilId' quer de um botão '*Invoke'* para activação do *Web Method*; assim, neste interface, são criados todos os elementos necessários à introdução dos valores para os argumentos a enviar para os métodos de invocação, sem qualquer programação. Em termos comparativos, utilizando a metodologia alternativa aos WS (formulário), seria necessário definir tantas caixas de texto de entrada de valores quanto o número de argumentos a passar ao método de invocação;
- accionar *Web Method* premindo o botão '*Invoke'* e analisar o resultado devolvido que se encontra apresentado na Figura 56; podemos verificar que existem dois utilizadores com perfil igual a 2, cujos nomes são 'dalila' e 'ricardo' e cujas chaves são 'costabrava' e 'qqwweerr' respectivamente.

Figura 55 – Interface disponibilizado e inovação do *Web Method*

Este resultado pode ser confirmado por consulta directa à base de dados verificando assim o correcto funcionamento não só da classe de invocação mas também de todas as restantes estruturas envolvidas, para o caso analisado.

No caso de o teste se centrar no método 'Select_iPerfilId', é gerado o interface apresentado na Figura 57, onde, além da caixa de texto para introdução do valor de perfil, também aparece uma caixa de texto para indicação do valor de indexação a utilizar para selecção do utilizador na relação.

Figura 56 – Resultado da invocação do método web 'Select_PerfilId'

Como resultado da invocação obtém-se agora apenas um utilizador que corresponde ao segundo da lista obtida por activação do método 'Select_PerfilId', tal como se pode verificar na Figura 58.

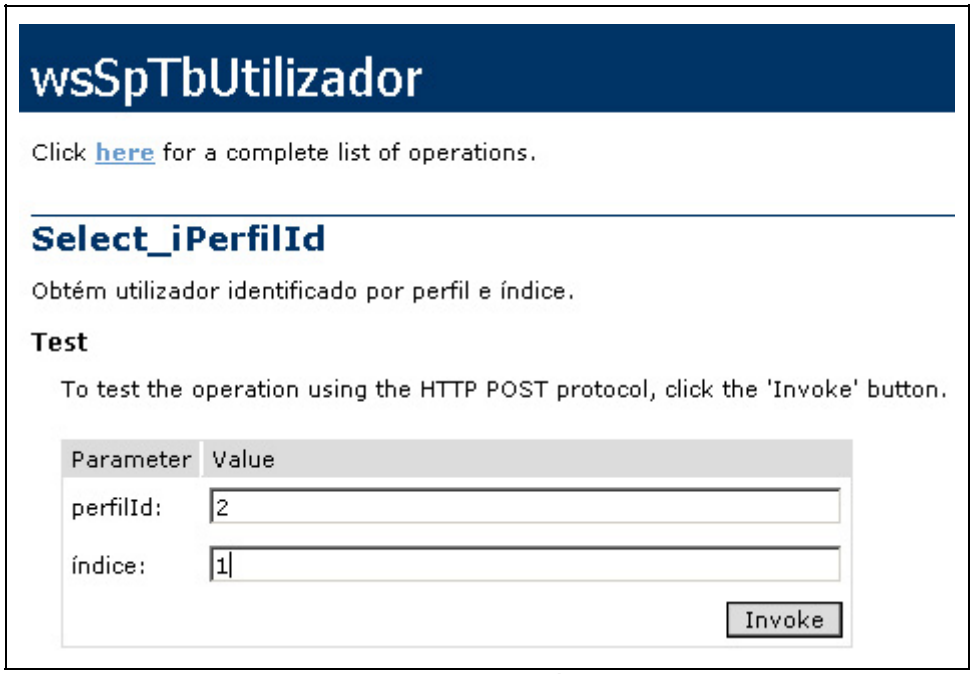

Figura 57 – Interface para o método 'Select_iPerfilId'.

Figura 58 – Resultado obtido por activação do método 'Select_iPerfilId'

6.4.5 Trabalhos futuros

A metodologia aqui utilizada para testes poderia ser melhorada em diversas vertentes, sendo necessário para tal desenvolver uma aplicação específica que, para além das funcionalidades já existentes na versão da Microsoft, disponibilizasse, por exemplo, outras alternativas, para além de XML, para apresentação da informação devolvida pelo WS. Pelo facto de os WS serem serviços normalizados e universalmente aceites, pelo facto de também serem acedidos remotamente, podem também abrir possibilidades de implementação de testes em modo remoto, em rede, etc. Julgo que poderá ser um campo a explorar.

6.5 Conclusões

Vamos apresentar algumas conclusões sobre o trabalho realizado e descrito neste capítulo.

Procedimentos Armazenados

Os procedimentos armazenados que foram desenvolvidos, num total de 41, apresentam-se enquadrados numa estratégia base que tem por objectivo o estabelecimento de um interface orientado ao objecto para acesso a esses mesmos procedimentos armazenados. Os requisitos definidos abrangem aspectos tais como: designação a atribuir aos argumentos, esquemas das relações devolvidas e valores devolvidos. As regras são fáceis de entender e de implementar, não representando restrições ao nível da operacionalidade dos procedimentos armazenados nem ao nível da capacidade criativa de cada um de nós.

Alguns aspectos ficam por solucionar, dos quais se destacam:

- atribuição de designação a argumentos não derivados;
- atribuição de designação a argumentos repetidos.

PIOOBD

A metodologia definida e implementada é o resultado de uma série de tentativas efectuadas no passado, no sentido de normalizar os procedimentos de acesso a base de dados. A abordagem apresentada mostrou-se adequada às necessidades exigidas, revelando:

- uma boa organização da informação;
- muito fácil utilização do ponto de vista da aplicação;
- ganhos de produtividade a partir do momento que existe uma base já construída para alguns procedimentos armazenados.

Algumas melhorais podem ser introduzidas no PIOOBD, das quais se destacam a geração automática do código referente às classe de mapeamento, às classes de atributos e também às classes de invocação.

PTIBD

O PTIBD é de grande importância permitindo efectuar de uma forma simples testes às classes de invocação. A opção pela utilização de WS veio-se revelar muito importante, tendo em consideração os seguintes aspectos:

- grande facilidade de implementação e de manutenção;
- todos os interfaces são gerados automaticamente, quer os de entrada de argumentos, quer os de activação dos métodos quer os de visualização da informação devolvida;

- permite uma boa organização, pela associação a cada classe de invocação a um único WS, agrupando assim todos os *Web Methods* de teste de uma classe de invocação numa única classe.

A experiência mostra que quer a criação de um novo WS, quer a criação de um novo *Web Method* requerem um esforço muito reduzido. Os valores aproximados indicam-nos que para criar um novo WS com quatro métodos são necessários não mais do que dois a três minutos. Outra vantagem significativa neste esquema de teste, reflecte-se nas situações em que se introduzem alterações, nomeadamente pela inserção, remoção ou alteração de colunas de tabelas da base de dados, que impliquem alteração em qualquer uma das classes do PIOOBD. O tempo de execução de novos testes é muito reduzido, bastando seguir os passos já aqui enumerados, e qualquer anomalia existente é imediatamente ressaltada pelos mecanismos aqui implementados. Com recurso ao *debugger* do *Visual Studio.Net*, a detecção e correcção dos erros têm sido efectuadas em tempos muito reduzidos, normalmente não ultrapassando os quatro minutos.

Pelo facto de os WS serem serviços normalizados e universalmente aceites, e ainda pelo facto de também serem acedidos remotamente, podem também abrir possibilidades de implementação de testes em modo remoto, em rede, etc. Julgo que poderá ser um campo a explorar.

7 Aplicações Desenvolvidas

Neste capítulo apresentam-se as aplicações que possuem interface com os utilizadores. As aplicações que possuem interface com os utilizadores são as referentes ao módulo de administração e ao módulo de formação.

7.1 Módulo de Administração

O Módulo de Administração, MódAdm, congrega um conjunto de funções vocacionadas para a gestão e manutenção de abcNet, principalmente em termos da informação residente na base de dados. É neste projecto que se centram as funções associadas a:

- gestão de utilizadores;
- gestão de cursos;
- gestão de elementos padagógicos tais como as sílabas, as palavras, as pronúncias e as imagens.

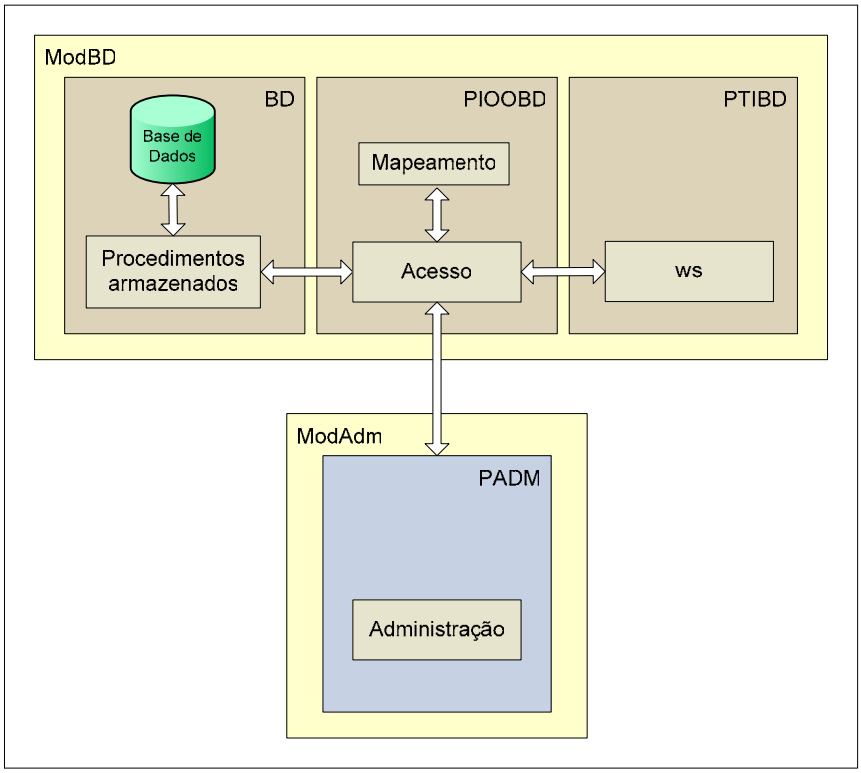

Figura 59 – Módulo de administração e sua integração com MódBD

Este módulo, apesar de importante, é visto como uma necessidade para que todo o resto possa funcionar e não como uma das tarefas fundamentais em termos de investigação. O desenvolvimento de uma aplicação *Windows* para manutenção de informação de uma base de dados não é concerteza um desafio aliciante nem motivante. Tendo também em consideração que esta fase de desenvolvimento de abcNet pretende ser uma primeira abordagem a uma aplicação baseada na *Web* vocacionada para a alfabetização, pode-se concluir que este módulo é o que menos questões levanta quer em termos de tecnologia quer em termos de inovação. Assim, deste projecto, apenas foram desenvolvidos os aspectos que dificilmente poderiam ser substituídos por operações manuais realizadas directamente sobre a base de dados. Apenas algumas questões relacionadas com elementos pedagógicos não prescindiam de um tratamento informático. Todos os outros aspectos, no actual momento, podem ser facilmente resolvidos por acesso directo à base de dados. A Figura 59 apresenta a estrutura do projecto e sua integração em abcNet.

7.1.1 Elementos Pedagógicos

De todos os elementos pedagógicos, apenas alguns mereceram o desenvolvimento de programas específicos para gestão dos mesmos. Estes elementos são as sílabas, os agrupamentos de sílabas e as referências para pronúncias de sílabas que são armazenados em tabelas.

As sílabas são constituídas por uma sequência de letras e de entre todas as possibilidades de combinação, algumas delas não são válidas na língua portuguesa. Foram implementados diversos algoritmos para geração automática quer das sílabas, quer dos agrupamentos, quer das referências aos ficheiros de pronúncia. As designações dos ficheiros que contêm quer as pronúncias das sílabas quer as pronúncias de palavras são objecto de utilização por parte dos professores para configuração dos ModAlf. Assim sendo, especial cuidado foi dado às designações atribuídas aos ficheiros de modo a permitir que pela simples leitura da designação, o professor tenha a percepção exacta da pronúncia que lhe está associada. Foi analisado o alfabeto fonético para o dialecto padrão do português europeu [LCO], mas rapidamente se verificou a impossibilidade da sua utilização para atribuição de designações a ficheiros, devido à utilização de caracteres que não podem ser utilizados na atribuição das mesmas, nomeadamente o @ entre outros. Além desta situação, o objectivo a atingir pelo alfabeto fonético vai muito para além das necessidades actuais de abcNet. O que se pretende neste caso, é que alguém que conheça a língua portuguesa tenha a possibilidade de pronunciar o som associado a um ficheiro de áudio, pela simples leitura da designação atribuída ao ficheiro. Para o caso de cada uma das cinco vogais pronunciadas no ‘aeiou’ foi dada a designação a+, e+, i+, o+ e u+. Algumas destas vogais podem ter pronúncias distintas das utilizados no ‘aeiou’, pelo que foi construído um esquema baseado em caracteres e que identificam a acentuação associada a cada vogal em cada contexto. Os caracteres utilizados foram ‘+’, ‘^’, ‘~’ e ‘-‘. Na Tabela 4 apresentam-se diversas situações de aplicação.

símbolo	Grafemas	exemplos
a+	a, à, á	g*a*lo, *à*, l*á*
a^	a	c*a*m*a*
a~	ã	l*ã*
e-	e	m*e*, entr*e*
e^	e, ê	t*e*r, v*ê*
e+	e, é	r*e*sto, r*é*
i+	i, í	*i*vo, ca*í*
o^	o, ô	*o*vo, av*ô*
o+	o, ó	l*o*go, av*ó*
o~	õ	p*õ*e
u+	u, ú, o	*u*m, h*ú*mid*o*
u-	o, u	macac*o*, cúm*u*lo

Tabela 4 – Designações de vogais

Podemos verificar que a todas as vogais está associado um ou mais dos seguintes sufixos: '+', '-', '~' ou '^'. Cada um destes sufixos indica a forma de leitura da respectiva letra: aberto(+), átono(-), nasal(~) ou fechado(^), respectivamente, segundo o nosso critério. Esta norma não pretende ser, nem é uma proposta de alfabeto fonético, é apenas uma norma que permite que qualquer vogal de uma sílaba escrita possa ter apenas uma e só uma forma de leitura. Também para os grafemas 's', 'x', 'z', 'ç', 'ss' e 'c' foram definidos alguns critérios tal como se apresenta na Tabela 5. Apesar de incompleto, as designações definidas na Tabela 4 e na Tabela 5 cobrem uma grande parte das situações correntes, sendo de momento suficiente para os objectivos que se pretendem atingir.

símbolo	grafemas	exemplos
z	s, x, z	a*s*a, ê*x*ito, *z*elo
s	s, ç, ss, c	sala, laço, russo, cela

Tabela 5 – Designação para 's' e 'z'.

Alguns exemplos de pronúncias de sílabas integradas em palavras.

palavra	pronúncia	palavra	pronúncia	palavra	pronúncia
por	pu+r	abelha	a^be^lha^	porco	po^rcu-
porca	po+rca^	estar	i+sta+r	macaco	ma^ca+cu-
ponte	po^nte-	chave	cha+ve-	casa	ca+za^

Tabela 6 – Exemplos de palavras

Alguns exemplos de sílabas homógrafas

Sílaba	palavras / pronúncias
le	lego (le+gu-), letra(le^tra^), trote(tro+te-)
go	gola(go+la^), golo(go^lu-), logo(lo+gu-)
sa	sala(sa+la^), casa(ca+za^)

Tabela 7 – Exemplos de designação de sílabas homógrafas

Alguns exemplos de sílabas homófonas

Sílaba	palavras / pronúncias
se,ze	coser(cu-ze^r), cozer(cu-ze^r)
sa,ça,ssa	saco(sa+cu-), caçada(ca^ça+da^)

Tabela 8 – Exemplos de designação de sílabas homófonas

A partir dos critérios aqui definidos, a leitura de sílabas e de palavras torna-se num processo determinístico quanto à pronúncia a utilizar. Analisemos alguns casos:

- à palavra *rato* está associado o ficheiro *ra+tu-*
- à sílaba *mo* estão associados os ficheiros *mo+*, *mo^*, *mu+* e *mu-*, em que a cada ficheiro corresponde uma das possíveis pronúncias de *mo*

7.1.2 Interfaces com o utilizador

O MódAdm implementa dois interfaces para tratamento dos elementos pedagógicos:

- um para criação automática dos elementos, como se pode ver na Figura 60;
- um para a visualização dos elementos pedagógicos existentes, como se pode ver nas Figura 61 e Figura 62.

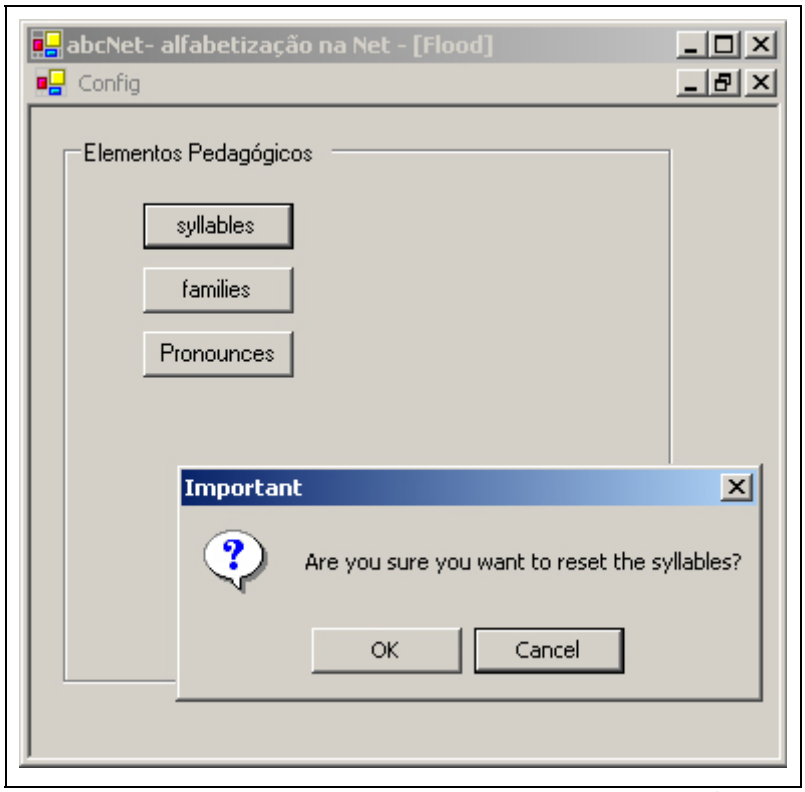

Figura 60 – Interface para criação do elementos pedagógicos

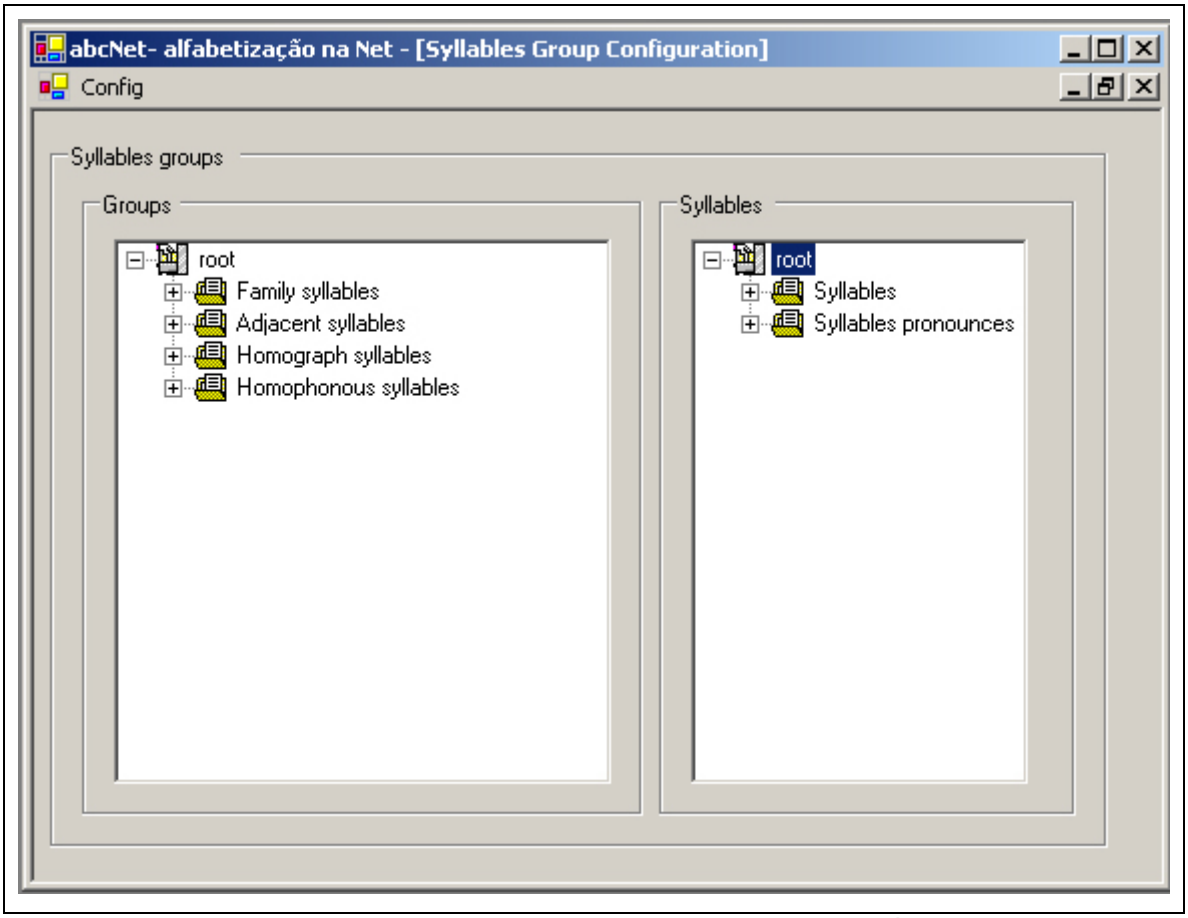

Figura 61 – Interface para visualização dos elementos pedagógicos criados

A Figura 61 apresenta os grupos dos elementos padagógicos existentes. Do lado esquerdo os elementos pedagógicos associados às sílabas encontram-se apresentados segundo o tipo de grupo em que se encontram inseridos. Do lado direito podemos verificar que a apresentação se organiza em SílOrt e SílPro.

A Figura 62 apresenta detalhes de cada um dos esquemas de visualização. Do lado esquerdo podemos ver as sílabas adjacentes da sílaba *ca* e respectivas pronúncias. Do lado direito podemos ver as pronúncias ordenadas de todas as sílabas existentes.

Os algoritmos implementados permitiram a criação de 4799 SílOrt. Apesar de não terem coberto todas as possibilidades existentes, as SílOrt disponibilizadas são concerteza suficientes para a actual utilização de abcNet. De fora ficaram SílOrt tais como as *qu**, *ç**, *gn**, *gu**,etc. A sua não inclusão nada teve a ver com dificuldades na implementação dos algoritmos, mas somente com a oportunidade da sua utilização.

Os algoritmos implementados criam não só as SílOrt mas também:

- os diversos grupos de sílabas;
- designação dos ficheiros que representam as diversas SílPro de cada SílOrt.

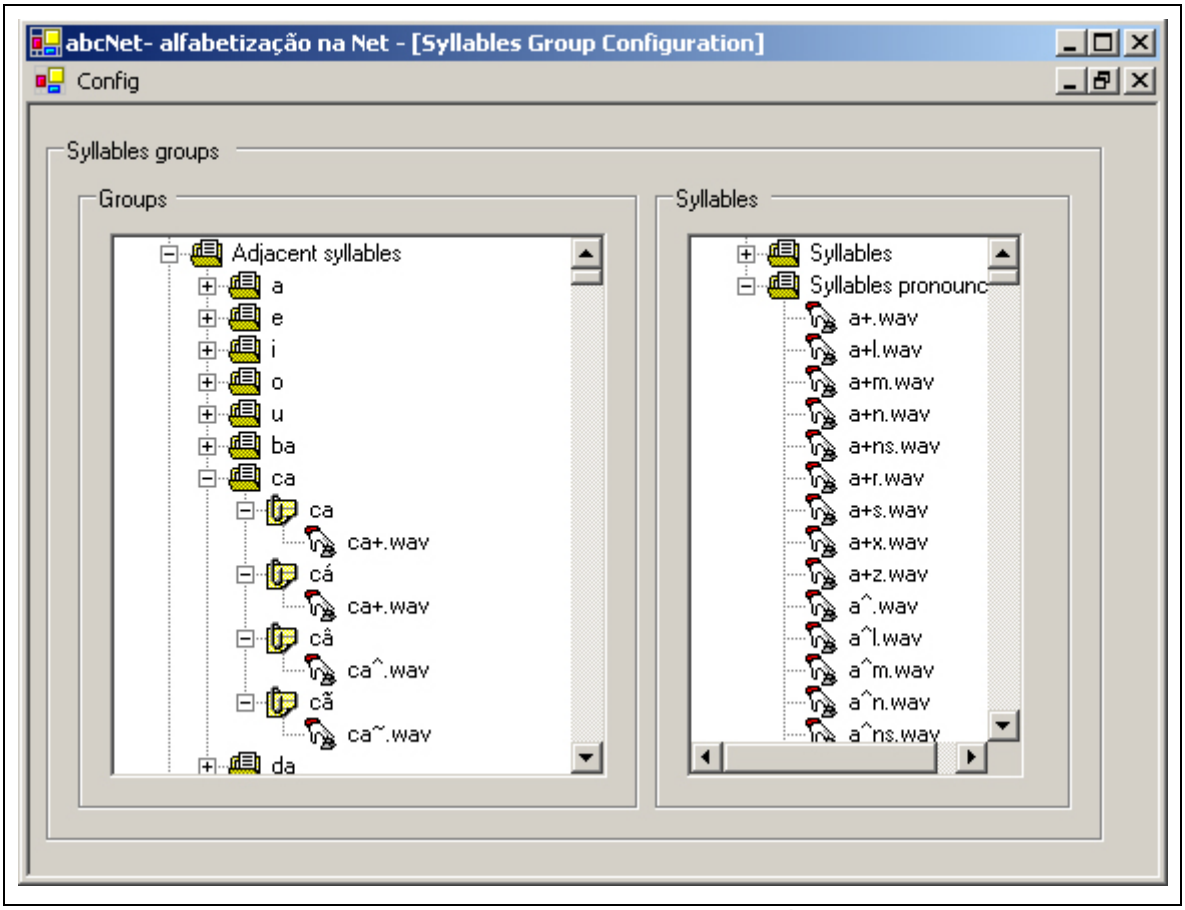

Figura 62 – Detalhes dos elementos pedagógicos criados

Os ficheiros áudio correspondentes às SílPro foram criados à medida que as necessidades assim o exigiam. Cada SílPro foi gravada num ficheiro áudio cuja designação foi definida pelos algoritmos já referidos, ver Figura 62.

7.2 Módulo de Formação

Todos os módulos anteriormente descritos têm como objectivo último, a criação das condições para que o Módulo de Formação (MódFor) possa desempenhar correctamente as suas funções. O MódFor engloba as componentes associadas quer à configuração das acções de formação quer às correspondentes ao desenrolar das aulas. A estrutura do MódFor está apresentada na Figura 63. O bloco ‘configurador’ é

de acesso exclusivo do professor enquanto que o bloco ‘aulas’ é partilhado pelos professores e pelos alunos. O acesso à base de dados, como seria de esperar, é também efectuado via o PIOOBD.

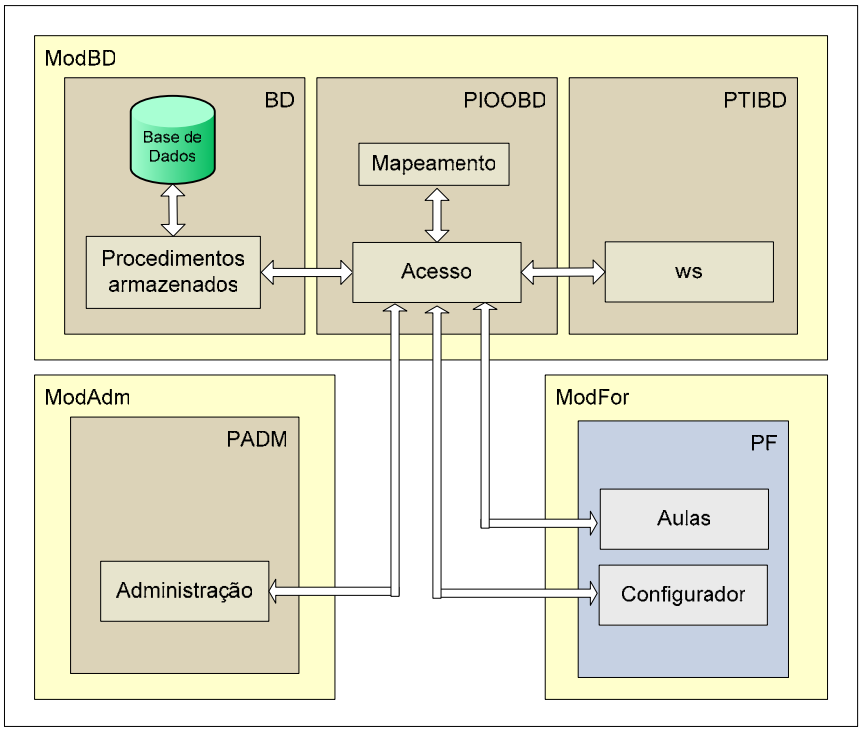

Figura 63 - MódFor e sua integração em abcNet

7.2.1 Esquemas de Navegação

Os blocos constituintes do MódFor localizam-se no servidor de páginas *Web*. Do ponto de vista do utilizador, os blocos são ficheiros com a extensão ‘aspx’ que podem ser acedidos via um qualquer explorador de *Internet* e quer o bloco ‘configurador’ quer o bloco ‘aulas’ são constituídos por páginas HTML cujos esquemas de navegação se apresentam nas Figura 64 e Figura 65. Como se pode verificar, os utilizadores de perfil ‘aluno’ partilham todas as suas páginas com os utilizadores de perfil ‘professor’.

A Tabela 9 apresenta o mapeamento de cada página para o ficheiro que lhe está associado e simultaneamente uma breve descrição das funções implementas pelo mesmo.

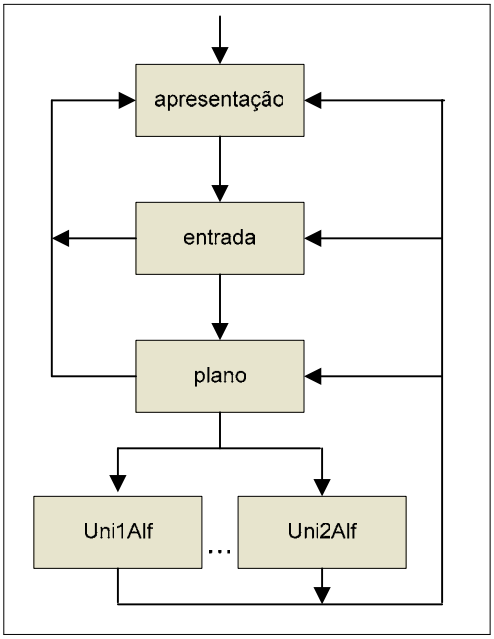

Figura 64 – Esquema de navegação para o perfil ‘aluno’

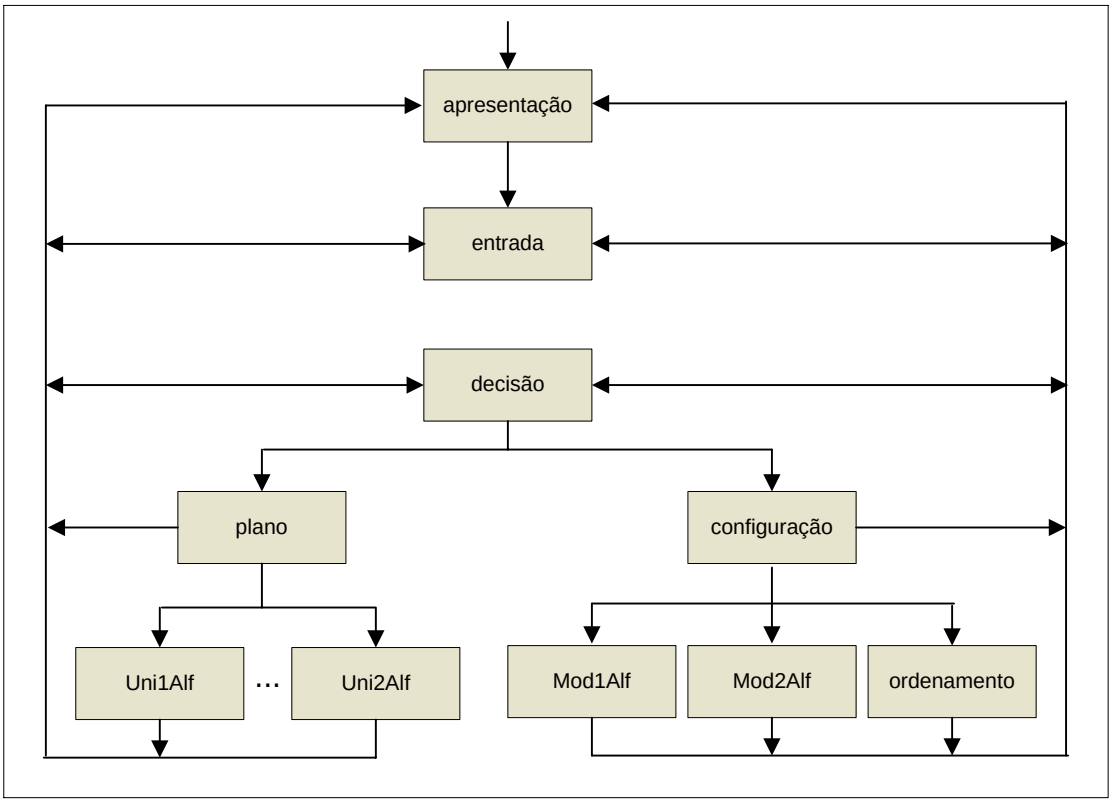

Figura 65 – Esquema de navegação para o perfil ‘professor’

página	ficheiro	descrição
apresentação	wfEntry.aspx	apresentação de abcNet.
entrada	wfLogin.aspx	identificação dos utilizadores
plano	wfCSchedule.aspx	plano de formação para uma dada acção de formação
Uni1Alf	wfCTemplateWr1_1.aspx	execução de Uni1Alf de acção de formação
Uni2Alf	wfCTemplateWr2_1.aspx	execução de Unid2Alf de acção de formação
decisão	wfTeacherHome.aspx	selecção da acção de formação e decisão sobre o que fazer
configuração	wfPlanning.aspx	configuração de acção de formação
Mod1Alf	wfPlanning.aspx	configuração de unidades de Uni1Alf (Mod1Alf)
Mod2Alf	wfPlanning.aspx	configuração de unidades de Uni2Alf (Mod2Alf)
ordenamento	wfPlanning.aspx	ordenação dos modelos configurados da acção de formação

Tabela 9 – Páginas do projecto 'formação'

7.2.2 Arquitectura Geral do MódFor

Para o desenvolvimento do MódFor, foi definido como premissa que a construção dinâmica das páginas deveria ser maximizada. Esta premissa tem como objectivo uma aposta forte na formação na vertente da construção das páginas em tempo de execução, apesar do esforço ser acrescido quer em termos de tempo de desenvolvimento quer em termos de aprendizagem. Assim, as soluções encontradas para as diversas situações poderão nem sempre ser as mais ajustadas em termos técnicos mas são concerteza as que melhor se ajustam às prioridades definidas.

A apresentação do MódFor proceder-se-à segundo uma lógica assente em tipos de blocos que foram desenvolvidos, a saber:

ambiente: gestão e controlo de utilizadores ao nível da sessão e da aplicação;

scripts: *scripts* implementados para serem executados do lado do cliente;

controlos: controlos desenvolvidos para reutilização no projecto;

páginas: páginas desenvolvidas para implementação dos interfaces.

7.2.3 Ambiente

Para se descrever o que se entende por Ambiente é necessário introduzir os conceitos de Ambiente de Aplicação e de Ambiente de Sessão. Por 'Ambiente de Aplicação' (AA) entende-se o conjunto de informação a ser partilhada por todas as sessões, ao qual se encontra associado um objecto .NET denominado de *Application* [MS-APPOB].

Por ‘Ambiente de Sessão (AS) entende-se o conjunto de informação associado a um utilizador, ao qual se encontra associado um objecto .NET denominado de *Session* [MS-SESOB].

7.2.3.1 Ambiente de Aplicação

.NET disponibiliza um objecto denominado de *Application* onde pode ser guardada a informação associada à aplicação e que pode ser partilhada por todas as sessões. De modo a permitir um acesso orientado ao objecto à informação aí guardada, foi criada uma classe ‘clsApplication’ que aglutina toda a informação relacionada com a aplicação. De entre a informação aí armazenada, destaca-se:

- estrutura de directorias para acesso a ficheiros de pronúncias de sílabas e de palavras, localização das páginas, etc.;
- informação sobre ligação à base de dados.

O AA é constituído pelo objecto *Application* e pela instância da classe ‘clsApplication’ que se encontra armazenada no objecto *Application*. Em cada instante, apenas deve existir uma instância da classe ‘clsApplication’ que contém toda a informação partilhada por todas as sessões. De modo a facilitar o acesso à instância de ‘clsApplication’ em Application, ‘clsApplication’ possui uma propriedade pública e estática, cuja designação é ‘key’ e que serve de identificação da instância da própria classe no objecto *Application*. Na Figura 66 apresenta-se a estratégia utilizada para a implementação do AA com recurso à classe ‘clsApplication’. Pode-se verificar que a identificação atribuída à instância criada de ‘clsApplication’ está contida na própria classe ‘clsApplication’.

```
protected void Application_Start(Object sender, EventArgs e)
{
    clsApplication _oApplication = new clsApplication();
    Application.Add( clsApplication.Key, _oApplication );
}
```

Figura 66 – ‘clsApplication’ em *Application*

Na Figura 67 apresenta-se um caso de aplicação do AA, que consiste essencialmente na obtenção da referência à instância de ‘clsApplication’ residente no objecto *Application*. O recurso a uma propriedade pública e estática da classe ‘clsApplication’, para identificação da instância da própria classe ‘clsApplication’ em *Application*, facilita significativamente a obtenção da sua referência. Este mecanismo de gestão da informação ao nível da aplicação é eficiente, disponibilizando também um interface amigável do ponto de vista do programador, pois o acesso à informação é orientado ao objecto.

Para a informação estática, a plataforma .NET disponibiliza uma alternativa, melhor dizendo, um complemento à solução apresentada, que é baseada num ficheiro de configuração designado por *Web.Config* [MS-ASPCNF], no qual a informação é organizada em formato XML e que pode

posteriormente ser lida a partir da aplicação. A Figura 68 apresenta um caso de definição de valores para 4 variáveis ao nível da aplicação, cujos valores podem ser lidos em tempo de execução tal como se encontra apresentado na Figura 69.

```
clsApplication oApp = (clsApplication) Application[ clsApplication.Key ];
SqlConnection oConn = oApp.Conn.connection;
```

Figura 67 – Utilização da classe 'clsApplication'

O mecanismo de leitura dos valores, quando comparado com o disponibilizado pela classe 'clsApplication', é muito mais elaborado e acima de tudo exige um conhecimento prévio das designações atribuídas aos elementos XML. Mas em contrapartida, os valores definidos desta forma, podem ser facilmente alterados, o que em muitas situações é uma grande vantagem, nomeadamente para situações das do tipo apresentadas na Figura 68, pois o ficheiro *Web.Config*, ao contrário da classe 'clsApplication', não é compilado.

```
<appSettings>
     <add key="logFilePath" value="d:\oscar\projectos\abcNet\log\log.txt."/>
     <add key="appHost" value="abcNet - Aveiro" />
     <add key="appDate" value="2004 Jun 26" />
     <add key="appMail" value="webmaster@who.pt" />
</appSettings>
```

Figura 68 – Utilização de *WebConfig* para configuração

Após a análise das vantagens e das desvantagens de cada implementação, foi decido aproveitar aquilo que de melhor cada uma das situações oferecia.

```
System.Configuration.AppSettingsReader
                oAppSetRd = new System.Configuration.AppSettingsReader();
_logFilePath = ((string)(oAppSetRd.GetValue("logFilePath", typeof(string))));
_appHost = ((string)(oAppSetRd.GetValue("appHost", typeof(string))));
_appDate = ((string)(oAppSetRd.GetValue("appDate", typeof(string))));
_appMail = ((string)(oAppSetRd.GetValue("appMail", typeof(string))));
```

Figura 69 – Leitura de valores de *WenConfig*

Assim, a inicialização da informação associada ao AA ficou organizada segundo os seguintes critérios:

Web.Config

No ficheiro *Web.Config* é definida a informação considerada como volátil, e que normalmente se encontra associada a uma dada instalação, tal como:

- responsável por uma dada instalação;
- data da instalação;
- localização de ficheiro de registo de acontecimentos (log);
- informação de conexão à base de dados, etc.

Os mecanismos utilizados para a sua definição encontram-se apresentados na Figura 68.

'clsApplication'

A classe 'clsApplication' implementa um interface orientado ao objecto não só para a informação considerada não volátil, como também para a definida em *Web.Config*. Este objectivo é conseguido, efectuando uma implementação de 'clsApplication' tal como se encontra apresentada na Figura 70. Os valores de *Web.Config* são lidos no construtor e disponibilizados através de propriedades públicas criadas especificamente para o efeito.

```
public clsApplication()
{
    System.Configuration.AppSettingsReader
                  oAppSetRd = new System.Configuration.AppSettingsReader();
    _logFilePath = ((string)(oAppSetRd.GetValue("logFilePath", typeof(string))));
    _appHost = ((string)(oAppSetRd.GetValue("appHost", typeof(string))));
    _appDate = ((string)(oAppSetRd.GetValue("appDate", typeof(string))));
    _appMail = ((string)(oAppSetRd.GetValue("appMail", typeof(string))));
}
public string LogFilePath
{
    get { return _logFilePath; }
}
```

Figura 70 – Informação de *WebConfig* através de 'clsApplication'

Para os valores não definidos na *Web.Config*, 'clsApplication' disponibiliza também interfaces específicos para cada um deles tal como se encontra apresentado na Figura 71. Neste caso, os interfaces específicos são respeitantes ao caminho de acesso para ficheiros que são utilizados por abcNet.

Com esta metodologia disponibiliza-se um mecanismo para definição de valores voláteis, no *Web.Config*, e para valores não voláteis, na *clsApplication*. *clsApplication* disponibiliza um interface orientado ao objecto quer para os valores voláteis quer para os não valáteis.

```
public static string imageFilesPath
{
    get { return "img/"; }
}
public static string SyllPronFilesPath
{
    get { return "syllPron/"; }
}
public static string WordPronFilesPath
{
    get { return "wordPron/"; }
}
public static string HelpFilesPath
{
    get { return "helps/"; }
}
```

Figura 71 – Propriedades afectas a valores intrínsecos a 'clsApplication'

7.2.3.2 Ambiente de Sessão

A estratégia seguida para a implementação da informação associada ao AS, é idêntica à utilizada para o AA. Foi definida uma classe 'clsSession' que contém a estrutura para toda a informação associada a cada sessão, que disponibiliza um interface orientado ao objecto para acesso a essa mesma informação. A 'clsSession' possui também uma propriedade estática, cuja designação é 'key', que serve de identificação da instância da própria classe no object *Session*.

Na Figura 72 apresenta-se a estratégia utilizada para a implementação do ambiente de sessão com recurso à classe 'clsSession'. Da figura podemos verificar que a instanciação da classe é efectuada no evento *Session_Start*, e que o seu construtor tem um argumento de entrada do tipo 'clsApplication' cujo valor corresponde à instância de 'clsApplication' no AA. Desta forma centraliza-se em 'clsSession', não só a informação referente ao AS como também a referente ao AA.

```
protected void Session_Start(Object sender, EventArgs e)
{
    Session.Add( clsSession.Key,
        new clsSession( (clsApplication) Application[ clsApplication.Key ] ) );
}
```

Figura 72 – 'clsSession' em *Session*

A Figura 73 apresenta não só a metodologia de utilização de 'clsSession' mas também a vantagem de a própria 'clsSession' conter uma referência à instância de 'clsApplication' existente no AA. Verificamos que desta forma se disponibiliza, por utilizador, um único ponto de partida para toda a informação necessária quer ao AS quer ao AA.

```
clsSession oSession = Session[ clsSession.Key ];
if ( oSession.Login == true )
    oSession.Application.Conn.OpenConn();
```

Figura 73 – Utilização combinada de 'clsSession e de 'clsApplication'

7.2.4 Scripts

A componente *Scripts* engloba um conjunto de programas desenvolvidos em *JavaScript* que tem por missão a execução de tarefas nas páginas HTML, mas do lado do cliente. A necessidade de execução de tarefas do lado do cliente tem a principal origem em exigências associadas a tempo de resposta a eventos gerados pelo utilizador, aliviando adicionalmente o servidor dessas tarefas. Vejamos o caso em que o utilizador prime um botão associado a um sílaba. Nesta situação, o utilizador espera que a pronúncia associada à sílaba possa ser ouvida mal o botão seja premido. Tal não se torna possível caso a decisão sobre as acções a desencadear após o accionamento do botão esteja centralizada no servidor, pois o pedido teria de ser enviado ao servidor e só depois da chegada da resposta haveria essa reacção. Este e outros casos foram analisados, tendo sido feito um levantamento de todas as situações que exigiam um tempo de resposta que se não compadecesse com uma implementação cliente-servidor. Do levantamento efectuado, resultou a identificação de 4 grupos distintos de necessidades, tendo os respectivos *scripts* sido centralizados também em 4 ficheiros distintos. Cada ficheiro contém um conjunto de *scripts* que são independentes e autónomos relativamente aos existentes noutros ficheiros. A designação dos ficheiros segue uma normalização, cuja estrutura é a seguinte scpXXXXXX em que: 'scp' é um prefixo que identifica o tipo do ficheiro, que neste caso são *scripts*, e XXXXXX é um descritivo dos objectivos a atingir pelos *scripts* aí existentes.

```
var playerControl = null;

function catalogPlayerControl( tag )
{
    playerControl = tag;
}
function stop()
{
    playerControl.controls.stop();
}
function play( file )
{
    playerControl.url = file;
    playerControl.controls.play();
}
```

Figura 74 – Catalogação de objecto e sua utilização.

Em termos de estrutura interna, os ficheiros também seguem uma normalização no que diz respeito a aspectos relacionados com a catalogação de objectos das páginas HTML. Por catalogação deve-se

entender o mecanismo de registo objectos, considerando como objecto um qualquer elemento HTML que tenha definido um valor para a propriedade *id*. Apesar de não ser essencial, em algumas situações, a catalogação vem evitar que se tenha de passar constantemente a referência de um objecto sempre que se pretende ter acesso ao mesmo por intermédio dos *scripts*. A Figura 74 apresenta um exemplo de catalogação de um objecto *Windows Media Player* e a disponibilização de dois métodos para controlo do mesmo.

Todas as funções de catalogação podem ser identificadas pela existência do prefixo 'catalog'. São permitidos dois tipos de catalogação, uns que correspondem aos objectos propriamente ditos, outros que correspondem ao *id*s dos objectos. No segundo caso, as funções apresentam um sufixo 'Id'. Assim, para o caso do exemplo apresentado na Figura 74, a função seria 'catalogPlayerControlId'. Esta situação pode ser importante para os casos em que os objectos ainda não existam no momento da sua catalogação. A Figura 75 apresenta uma possível implementação para o caso correspondente ao da Figura 74 mas com recurso ao 'Id'.

```
function catalogPlayerControlId( tagId )
{
    playerControlId = tagId;
}
function stopPlay()
{
    var playerControl = document.all( playerControlId );
    playerControl.controls.stop();
}
```

Figura 75 – Catalogação de objecto pelo seu *Id* e sua utilização

A utilização destes *scripts* requer que o ficheiro seja importado para a página onde vão ser utilizados e, posteriormente, se necessário, efectuar algumas inicializações através das funções 'catalog'.

Tal como já afirmado anteriormente, existem quatro ficheiros *.js que vão ser descritos de forma individualizada:

scpPlayer

O ficheiro 'scpPlayer' contém os *scripts* necessários ao controlo de som num objecto *Windows Media Player*. Os *scripts* permitem o controlo da execução dos ficheiros de som utilizados na alfabetização, nomeadamente ouvir as pronúncias das sílabas, das palavras, das ajudas, etc.

scpBtnMouseEvt

O ficheiro 'scpBtnMouseEvt' contém os *scripts* necessários ao controlo do *feedback* gerado pelos eventos de rato nos botões associados a elementos pedagógicos, nomeadamente os de *mouseover* e os de *mouseout*. Os botões associados aos elementos pedagógicos, tais como as sílabas e as palavras, possuem

um *feedback* visual que é activado por intermédio de funções existentes neste ficheiro e que consiste nomeadamente numa mudança de cor do botão e alteração do formato do cursor.

scpGuessWord

O ficheiro 'scpGuessWord' contém os *scripts* necessários para o controlo de todos os botões associados à composição de palavras nas UniAlf. São eles que permitem:

- a activação e a selecção de sílabas;
- a validação da construção da palavra proposta;
- o controlo da apresentação das sílabas associadas à construção da palavra proposta;
- o controlo do botão de passagem para a palavra seguinte após a correcta construção da palavra proposta.

scpOtherSyll

O ficheiro 'scpOtherSyll' contém os *scripts* necessários ao controlo da apresentação dos grupos de sílabas adjacentes, homófonas e homógrafas que estão associados à sílaba activa, da ficha de descoberta. Os grupos a serem apresentados são configurados nos respectivos modelos, como já explicado anteriormente.

7.2.5 Controlos

A plataforma .NET suporta o desenvolvimento de dois tipos de controlos para as páginas da *Internet*: *web user controls (*WUC*)* e os *web custom controls(*WCC*)*. Ambos apresentam vantagens e desvantagens comparativas entre si, que já foram apresentadas na Tabela 2.

Relembrando que uma das prioridades na implementação do MódFor se centra na construção de páginas em tempo de execução, foi tomada a opção de efectuar o desenvolvimento de controlos WUC em detrimento dos WCC. A construção dinâmica das páginas, não exige, entre outras condições, que os controlos possuam um interface visual em tempo de desenvolvimento, daí os WUC poderem satisfazer os requisitos pretendidos.

Os controlos já disponibilizados pelo Visual Studio.NET fornecem um conjunto vasto de funcionalidades, mas não cobrem todas as necessidades. Os WUC permitem a criação de controlos, recorrendo às técnicas já utilizadas na implementação de páginas HTML. Pode-se inclusivamente converter uma página HTML num WUC pela simples introdução de algumas pequenas alterações. Um WUC tem uma estrutura análoga aos Web Forms, possuindo um ficheiro de extensão 'ascx' que contém a componente associada ao HTML e um outro ficheiro de extensão ascx.cs (cs é de C Sharp – C#) que

correspondente ao *code-behind*. O ficheiro 'ascx' difere dos ficheiros 'aspx' associados a Web Forms, pelo facto de não conter os elementos <HTML>, <BODY> e <FORM>. A Figura 76 apresenta o conteúdo de um ficheiro 'ascx' onde podemos verificar a existência de apenas um elemento de atributos que se encontra a amarelo, e também um elemento <INPUT> do tipo botão. Efectivamente os elementos <HTML>, <BODY> e >FORM> não fazem parte da estrutura dos WUC.

```
<%@ Control Language="c#" AutoEventWireup="false"
    Codebehind="wucImageButton.ascx.cs" Inherits="abcNet.wuc.wucImageButton"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5"
    enableViewState="False"%>
<asp:ImageButton id="idWucImgBtn" runat="server" BorderWidth="1px"
     EnableViewState="False" CausesValidation="False"></asp:ImageButton>
```

Figura 76 – Estrutura de um ficheiro 'ascx'

Foram desenvolvidos alguns WUC, tendo-se definido algumas normalizações para os mesmos, que vão desde a designação do ficheiro, à designação de alguns dos seus métodos. Assim, a designação de todos os ficheiros associados a WUC possuem a seguinte estrutura 'wucXXXXXX' em que 'wuc' é um prefixo de identificação do tipo de ficheiro, e 'XXXXXX' é um descritivo dos objectivos a atingir pelo WUC.

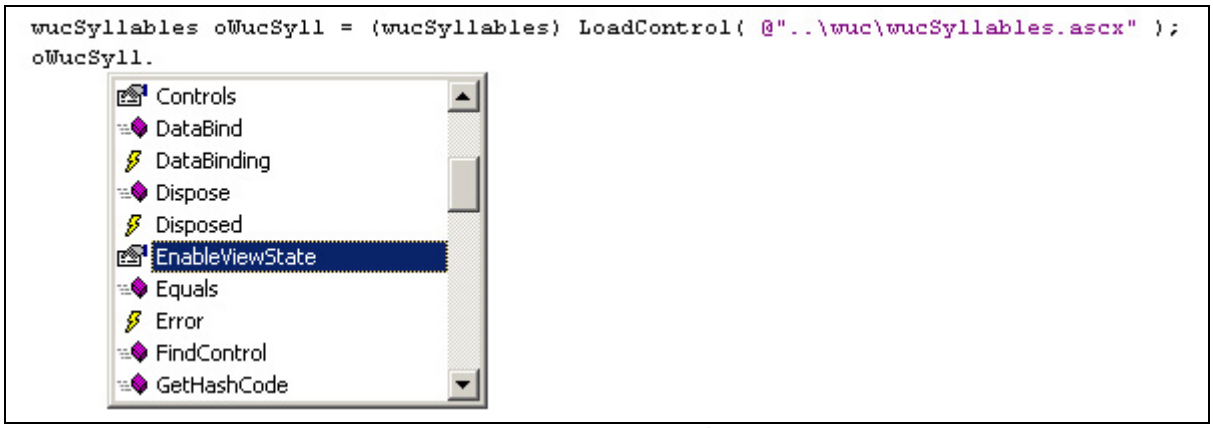

Figura 77 – Dificuldade na identificação de métodos e propriedades nos WUC

Em termos de designação dos métodos e propriedades, a atenção focalizou-se na dificuldade de identificação dos mesmos no ambiente de desenvolvimento, pois os WUC, por defeito, possuem um conjunto vasto de métodos e de propriedades públicas, tal como se encontra apresentado na Figura 77. Do ponto de vista do programador que utiliza os WUC, a identificação dos métodos e propriedades públicos por nós criados nos WUC, pode-se tornar um processo da maior simplicidade caso se siga estratégia da utilização de um prefixo único da sua identificação. A nossa opção recaiu na utilização do caracter '_' como prefixo de todos os métodos e propriedades públicas dos WUC .

Assim, os novos métodos e propriedades para 'wucSyllables' devem aparecer todos contíguos e no topo da janela de ajuda, tal como apresenta a Figura 78. Esta metodologia de desenvolvimento facilita significativamente quer a sua identificação quer a sua localização.

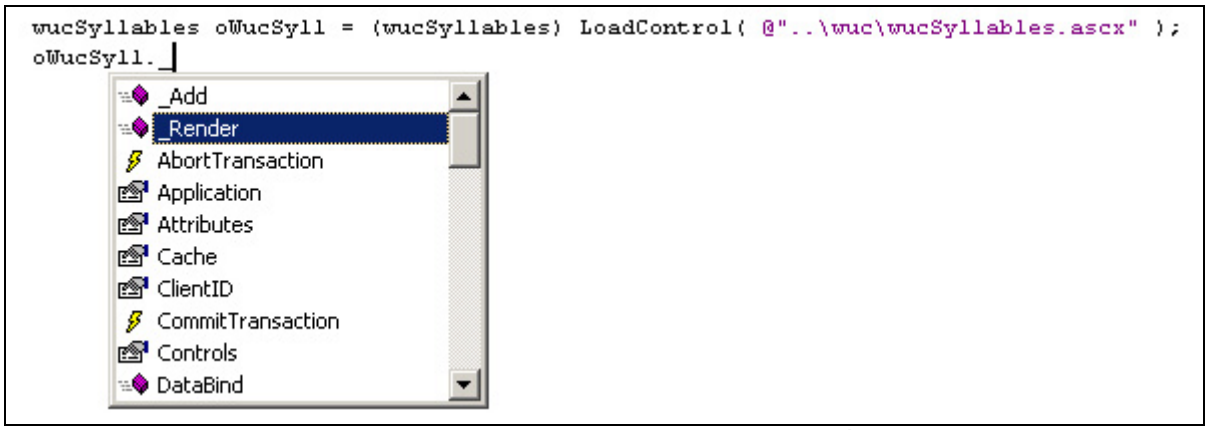

Figura 78 – Utilização do prefixo '_' para identificação de métodos e propriedades

Em Anexo I – Controlos apresentamos uma descrição de cada um dos controlos criados.

7.2.6 Páginas

As páginas HTML têm origem num tipo de ficheiros denominados de Web Forms – WF. Cada WF tem associado dois ficheiros distintos, tal como os WUC. Os ficheiros de extensão 'aspx' são os equivalentes aos ficheiros HTML já muito conhecidos. Os ficheiros de extensão 'aspx.cs' são responsáveis pela execução de tarefas no servidor que podem ir desde a construção da componente dinâmica das páginas HTML, quer ao acesso à base de dados, quer à execução de outras tarefas tidas como necessárias para a satisfação dos requisitos associados à página HTML.

A designação dos ficheiros tem a seguinte estrutura normalizada wfXXXXX em que 'wf' é um prefixo que identifica o tipo de ficheiro (Web Form) e XXXXXX é um descritivo dos objectivos a atingir.

7.2.6.1 Estrutura, Apresentação e Conteúdos

HTML, ou HyperText Markup Language é uma aplicação SGML que permite a marcação de documentos para inclusão na *World Wide Web*. HTML permite:

- a publicação de documentos na *Internet* num formato independente de qualquer plataforma;
- criação de links em documentos, para outros documentos relacionados;
- inclusão de imagens, sons e vídeo nos documentos.

O uso tradicional do HTML centra-se principalmente na vertente associada à representação de informação, potenciando a conivência entre a estrutura, a apresentação e conteúdo do documento. Com a

evolução da *Internet*, rapidamente se constatou a necessidade de se proceder à separação entre a estrutura, a apresentação e o conteúdo dos documentos. A sua separação oferece um conjunto vasto de vantagens, incluindo melhorias ao nível da acessibilidade, manuseamento e portabilidade. O desenvolvimento do HTML, associado ao aparecimento de novas tecnologias tais como, *Cascading Style Sheets – CSS* [W3C-CSS], XML [W3C-XML], *Active Server Pages* - ASP, etc, tem proporcionado soluções diversas para o problema enunciado. Com base em algumas tecnologias, vamos apresentar uma metodologia que permite a separação entre a estrutura, a apresentação o conteúdo de um documento HTML [OSCAR-AC2] e que foi utilizada no desenvolvimento de abcNet [OSCAR-ICALT].

7.2.6.1.1 Introdução

Estrutura, apresentação e conteúdo são os três blocos básicos na construção de páginas HTML. A implementação de uma metodologia que torne os 3 blocos independentes, potencia a construção de páginas HTML flexíveis, permitindo, por exemplo, que uma página HTML com uma dada estrutura, possa apresentar conteúdos dependentes do perfil do utilizador. Na estrutura define-se a localização e distribuição dos conteúdos pela página. Na apresentação define-se a forma como os conteúdos são apresentados. Os conteúdos a apresentar podem ser função de vários parâmetros que determinam que informação apresentar em cada circunstância. Os conteúdos são disponibilizados onde a estrutura o indicar e são mostrados segundo os critérios definidos na apresentação. Neste momento, podemos afirmar que alterações efectuadas na estrutura e/ou apresentação geram várias versões da mesma página HTML, mas sempre com o mesmo conteúdo. Se a estrutura permite a reordenação dos conteúdos, a apresentação permite que a forma como os conteúdos são apresentados possa ser ajustada, por exemplo, às necessidades particulares de determinados públicos. Para completar o círculo, apenas nos falta poder ajustar também o conteúdo em função dos públicos a que se destinam, que podem ser ou não os mesmos que foram referidos no bloco respeitante à apresentação. A metodologia que vamos apresentar recorre a 3 tecnologias distintas, 2 delas são o HTML e as CSS que permitem a implementação dos blocos respeitantes à estrutura e à apresentação. A terceira tecnologia é a que permite a implementação de conteúdos dinâmicos e que no nosso caso é o ASP.NET. Podemos ainda acrescentar, como culminar desta metodologia, que a construção dinâmica dos 3 blocos apresentados é também possível.

A metodologia aqui apresentada está em conformidade com a especificação da W3C – Web Content Accessibility Guidelines 1.0. [W3C-WAI].

7.2.6.1.2 Estrutura

A estrutura de um documento HTML pode ser toda definida com recurso a elementos HTML . Adicionalmente, os elementos podem ser organizados de forma a simular estruturas XML, tal como se apresenta na Figura 79.

```
<div id="idAluno" class="clsAluno">
    <div id="idNome" class="clsNome"> Carlos Silva Moura </div>
    <div id="idNumero" class="clsNumero"> 23490 </div>
    <div id="idCurso" class="clsCurso"> Filosofia </div>
    <div id="idMorada" class="clsMorada">
        <div id="idRua" class="clsRua"> R. Dr. Alberto Souto, 32-3º esq </div>
        <div id="idLocalidade" class="clsLocalidade"> Aveiro </div>
        <div id="idCP" class="clsCP"> 3800-002 Aveiro </div>
    </div>
</div>
```

Figura 79 – Estrutura de documento baseada em elementos HTML

A definição de propriedades adicionais, relativas à estrutura pode ser implementada com recurso a CSS aplicadas aos elementos tal como se apresenta na Figura 80.

```
.clsAluno { position: absolute; left: 300px; top: 200px }
.clsNome { position: relative; left: 0px; top: 0px }
.clsNumero { position: relative; left: 20px; top: 0px }
.clsCurso { position:relative; left: 20px; top: 0px }
.clsRua { position: relative; left: 20px; top: 0px }
.clsLocalidade { position: relative; left: 40px; top: 0px }
.clsCp { position: relative; left: 40px; top: 0px }
```

Figura 80 – Aplicação de CSS a selectores ID de elementos

A descrição até agora efectuada, permite-nos uma implementação da estrutura de um documento HTML com recurso a elementos , complementada com a utilização de CSS, aplicadas a selectores de classe dos elementos . A manipulação da propriedade *position* em elementos , permite efectuar uma disposição dos conteúdos sem necessidade de recurso a elementos do género <*table*>, <p>,
, etc. A utilização de elementos não é nem obrigatória nem são os únicos que podem ser adaptados à metodologia aqui apresentada. A utilização dos elementos é seleccionada apenas pelo facto de estes elementos serem de utilização genérica, normalmente utilizados para a definição de blocos.

Carlos Silva Moura
23490
Filosofia
R. Dr. Alberto Souto, 32-3º esq
Aveiro
3800-002 Aveiro

Figura 81 – Visualização do documento HTML

A Figura 81 apresenta a visualização do documento HTML construído a partir da informação apresentada na Figura 80 e Figura 81.

7.2.6.1.3 Apresentação

A metodologia utilizada para a definição da estrutura de um documento HTML, vai também ser utilizada para a definição da apresentação do mesmo documento. As CSS podem ser definidas com recurso quer aos selectores baseados no ID, quer em selectores baseados em classes, quer inclusivamente em selectores de contextos. Uma das características das CSS é a possibilidade da sua definição em diversos níveis, isto é, após termos definido as CSS apresentadas na Figura 80, podemos acrescentar novas CSS para os mesmos elementos HTML, baseadas num qualquer tipo de selector. Isto significa que podemos organizar as CSS por diversos níveis de modo a que cada nível represente um dos blocos definidos para a construção de uma página HTML.

```
.clsAluno { padding: 10px 10px 10px 10px; border: outset 2px #ff3388;
            background-color: #ee99cc; width: 300px }

.clsAluno { font-family:Verdana; font-style:normal; font-size:small; color:Navy; }
.clsNome { font-size:medium; font-weight:bolder; text-decoration: underline }
```

Figura 82 – Aplicação de CSS para definição da *Apresentação*

A Figura 82 apresenta as CSS aplicadas para definição da apresentação da página HTML. Optámos por dividir as CSS em 2 grupos. O primeiro grupo define a apresentação ao nível do enquadramento da informação enquanto que o segundo nível define a apresentação ao nível do texto.

Figura 83 – Visualização do documento HTML após definição da *Apresentação*

A Figura 83 apresenta o resultado obtido por aplicação das CSS da Figura 82 ao documento HTML apresentado na Figura 79.

7.2.6.1.4 Conteúdo

A última etapa centra-se na possibilidade da definição dinâmica dos conteúdos a apresentar. Tal como referido inicialmente, a implementação deste bloco vai recorrer à utilização da tecnologia ASP.NET.

ASP.NET implementa um mecanismo de criação dinâmica de páginas HTML baseado em 2 ficheiros.

```
<div id="idAluno" class="clsAluno" runat="server">
    <div id="idNome" class="clsNome" runat="server"></div>
    <div id="idNumero" class="clsNumero" runat="server"></div>
    <div id="idCurso" class="clsCurso" runat="server"></div>
    <div id="idMorada" class="clsMorada" runat="server">
        <div id="idRua" class="clsRua" runat="server"></div>
        <div id="idLocalidade" class="clsLocalidade" runat="server"></div>
        <div id="idCP" class="clsCP" runat="server"></div>
    </div>
</div>
```

Figura 84 – Página HTML sem conteúdo e com propriedade *runat="server"*

Um dos ficheiros, cuja extensão é aspx, consiste essencialmente numa página HTML. O outro ficheiro, que tem o mesmo nome da página HTML, mas cuja extensão é 'aspx.cs' (no caso de se utilizar o C#), contém o programa (code-behind) que é executado no servidor, cujo objectivo pode ser a construção dinâmica da página HTML a ele associada. Vamos reescrever a nossa página HTML tal como apresentada na Figura 84. Podemos verificar que relativamente à versão original existem as seguintes diferenças:

- os conteúdos foram removidos;
- foi acrescentada a propriedade *runat="server"* a cada um dos elementos .

```
protected System.Web.UI.HtmlControls.HtmlGenericControl idAluno;
protected System.Web.UI.HtmlControls.HtmlGenericControl idNome;
protected System.Web.UI.HtmlControls.HtmlGenericControl idNumero;
protected System.Web.UI.HtmlControls.HtmlGenericControl idCurso;
protected System.Web.UI.HtmlControls.HtmlGenericControl idMorada;
protected System.Web.UI.HtmlControls.HtmlGenericControl idRua;
protected System.Web.UI.HtmlControls.HtmlGenericControl idLocalidade;
protected System.Web.UI.HtmlControls.HtmlGenericControl idCP;

private void Page_Load(object sender, System.EventArgs e)
{
    idNome.InnerText = "Carlos Silva Moura";
    idNumero.InnerText = "23490";
    idCurso.InnerText = "Filosofia";
    idRua.InnerText = "R. Dr. Alberto Souto, 32-3º esq";
    idLocalidade.InnerText = "Aveiro";
    idCP.InnerText = "3800-002 Aveiro";
}
```

Figura 85 – Construção dinâmica do conteúdo a partir de *code-behind*

A propriedade *runat="server"* é a propriedade de ASP.NET que indica que o elemento deve ser disponibilizado para acesso a partir do ficheiro *code-behind*. O acesso ao elemento é efectuado a partir da sua propriedade ID tal como apresentado na Figura 85.

O conteúdo resultante é idêntico ao inicialmente apresentado, mas agora temos um mecanismo que permite definir em tempo de execução qual o conteúdo a apresentar. A possibilidade de acesso aos elementos HTML a partir do ficheiro *code-behind* permite um acesso orientado ao objecto, a um conjunto

vasto de propriedades e de métodos, dos quais se destaca, para o caso presente, a propriedade *innerHTML* que é a equivalente à *innerHTML* acessível no *DOM* via *JavaScript*.

7.2.6.1.5 Blocos Dinâmicos

O culminar desta metodologia e tecnologia centra-se na implementação de uma solução que propõe um mecanismo de construção dinâmica dos 3 blocos: estrutura, apresentação e conteúdo. A metodologia requer a reescrita do HTML inicialmente apresentado na Figura 79 e que passa pela não definição de valores para a propriedade *classe*. A definição da propriedade *class* passa a ser efectuada também em tempo de execução. A Figura 86 apresenta a nova versão para o HTML apresentado na Figura 79.

```
<div id="idAluno" runat="server">
    <div id="idNome" runat="server"></div>
    <div id="idNumero" runat="server"></div>
    <div id="idCurso" runat="server"></div>
    <div id="idMorada" runat="server">
        <div id="idRua" runat="server"></div>
        <div id="idLocalidade" runat="server"></div>
        <div id="idCP" runat="server"></div>
    </div>
</div>
```

Figura 86 – HTML para construção dinâmica da *estrutura*, *apresentação* e *conteúdo*

A Figura 87 apresenta a versão final da construção dinâmica dos blocos inicialmente definidos: estrutura, apresentação e conteúdo. Podemos verificar que quer o conteúdo quer as CSS que definem a estrutura e apresentação são atribuídos em tempo de execução a cada um dos elementos HTML.

```
idAluno.Attributes.Add( "class", "clsAluno" );
idNome.InnerText = "Carlos Silva Moura";
idNome.Attributes.Add( "class", "clsNome" );
idNumero.InnerText = "23490";
idNumero.Attributes.Add( "class", "clsNumero" );
idCurso.InnerText = "Filosofia";
idCurso.Attributes.Add( "class", "clsCurso" );
idRua.InnerText = "R. Dr. Alberto Souto, 32-3º esq";
idRua.Attributes.Add( "class", "clsRua" );
idLocalidade.InnerText = "Aveiro";
idLocalidade.Attributes.Add( "class", "clsLocalidade" );
idCP.InnerText = "3800-002 Aveiro";
idCP.Attributes.Add( "class", "clsCP" );
```

Figura 87 – Construção dinâmica da *estrutura*, *apresentação* e *conteúdo*

7.2.6.1.6 Resumindo

A metodologia aqui apresentada pode ser utilizada na sua globalidade, para quem utilizar ASP.NET ou outra tecnologia que disponibilize funcionalidades idênticas, ou parcialmente, recorrendo simplesmente aos elementos e CSS para separação dos dois primeiros blocos, estrutura e apresentação. A

metodologia pode ser utilizada em qualquer tipo de páginas HTML, das quais de destacam algumas situações particulares:

- páginas dirigidas a utilizadores com diversos tipos de deficiência no acesso aos conteúdos, sendo relevante o blocos respeitante à *Apresentação*;
- *e-learning* em que os conteúdos são ajustados ao perfil de cada utilizador.

O caso apresentado é muito simples e apenas reflecte uma situação com uma única estrutura, uma única apresentação e um único conteúdo. Numa situação geral, a construção da página teria em conta várias hipóteses de atribuição de CSS, cada uma vocacionada para uma situação particular, e o conteúdo seleccionado seria também função de um ou mais factores.

A metodologia descrita pode também ser facilmente generalizada para uma utilização mais abrangente, tal como o foi na construção de algumas páginas HMTL de abcNet e que são apresentadas em parágrafos subsequentes.

7.2.6.2 Menus

O esquema de navegação foi implementado com recurso a menus desenvolvidos em Flash. A utilização de objectos Flash em HTML, exige a utilização de elementos <Object> cuja estrutura possui alguma complexidade. De modo a evitar problemas com a utilização dos objectos, foi definido um mecanismo centralizado de gestão dos menus.

```
public static void BackgroundFlashImage( HtmlGenericControl tag,  string file )
{
    string str = "<OBJECT id='idBg' codeBase='http://download.macromedia.com/pul
                str += "<PARAM NAME='_cx' VALUE='29104'>";
                str += "<PARAM NAME='_cy' VALUE='22013'>";
                str += "<PARAM NAME='FlashVars' VALUE=''>";
                str += "<PARAM NAME='Movie' VALUE='" + file + "'>";
                str += "<PARAM NAME='Src' VALUE='" + file + "'>";
                str += "<PARAM NAME='WMode' VALUE='Transparent'>";
                // str += .......
                str += "<EMBED src=" + file + "quality='high' bgcolor='#FFFFFF'
                str += "</EMBED>";
                str += "</OBJECT>";
    tag.InnerHtml = str;
}
```

Figura 88 – Método para criação do elemento <object> associado aos menus

Este mecanismo é constituído por um método estático de uma classe que constrói os elementos <object> para cada WF, recebendo como argumento o elemento HTML onde deve ser criado o elemento <object> e ainda o ficheiro a ser utilizado para o efeito. Este método, cuja implementação parcial se apresenta na Figura 88, é invocado a partir de cada um dos WF, do ficheiro 'aspx.cs', tal como se apresenta na Figura 89.

Uma análise mais cuidada da Figura 89 mostra-nos que no caso presente, o menu a utilizar é função do perfil do utilizador, que pode ser de professor ou de aluno.

```
_oSession = (clsSession) Session[ clsSession.Key ];
switch ( _oSession.UserProfileId )
{
    case eIdTbProfile_id.student:
        clsWfs.BackgroundFlashImage( idImgBackground, "obj/frmSchedule.swf" );
        break;
    case eIdTbProfile_id.teacher:
        clsWfs.BackgroundFlashImage( idImgBackground, "obj/frmScheduleTeacher.swf" );
        break;
}
```

Figura 89 – Afectação do menu ao WF correspondente.

```
<div id="idImgBackground" runat="server">
</div>
```

Figura 90 – Elemento HTML para inclusão do menu

```
<div id="idImgBackground">
   <OBJECT id='idBg' codeBase='http://download.macromedia.com/p
       <PARAM NAME='_cy' VALUE='22013'>
       <PARAM NAME='FlashVars' VALUE=''>
       <PARAM NAME='Movie' VALUE='obj/frmScheduleTeacher.swf'>
       <PARAM NAME='Src' VALUE='obj/frmScheduleTeacher.swf'>
       <PARAM NAME='WMode' VALUE='Transparent'>
       <PARAM NAME='Play' VALUE='-1'>
       <PARAM NAME='Loop' VALUE='-1'>
       <PARAM NAME='Quality' VALUE='High'>
       <PARAM NAME='SAlign' VALUE=''>
       <PARAM NAME='Menu' VALUE='-1'>
       <PARAM NAME='Base' VALUE=''>
       <PARAM NAME='AllowScriptAccess' VALUE='always'>
       <PARAM NAME='Scale' VALUE='ShowAll'>
       <PARAM NAME='DeviceFont' VALUE='0'>
       <PARAM NAME='EmbedMovie' VALUE='0'>
       <PARAM NAME='BGColor' VALUE='FFFFFF'>
       <PARAM NAME='SWRemote' VALUE=''>
       <PARAM NAME='MovieData' VALUE=''>
       <PARAM NAME='SeamlessTabbing' VALUE='1'>
       <EMBED src='obj/frmScheduleTeacher.swf' quality='high' b
    </OBJECT>
</div>
```

Figura 91 – Resultado da composição HTML para geração do menu

Isto significa que um mesmo WF pode ter associado um ou mais menus distintos. Mostra-nos também que o elemento onde deve ser inserido o menu é identificado por ‘idImgBackground’ que é um elemento

HTML do WF, tal como é apresentado na Figura 90. O atributo *runat="server"* é essencial, pois só assim se tem acesso ao elemento a partir do ficheiro 'aspx.cs' tal como apresentado na Figura 89.

A Figura 91 apresenta parcialmente o resultado final da composição HTML obtida para incorporação do menu associado ao WF, onde se pode verificar a atribuição do ficheiro *flash* anteriormente definido na Figura 89.

7.2.6.3 Páginas utilizadas

Neste parágrafo apresenta-se resumidamente alguma informação sobre as páginas HTML desenvolvidas. As páginas aqui apresentadas correspondem às que são criadas explicitamente para serem apresentadas aos utilizadores.

Nestas páginas são utilizados os WUC apresentados no Anexo I – Controlos.

wfEntry

wfEntry é a página de apresentação de abcNet. Esta página é apresentada ao utilizador em duas situações distintas: uma das situações é obviamente quando o utilizador explicitamente acede à URL correspondente à página wfEntry; a segunda situação corresponde ao redireccionamento automático para esta página sempre que o utilizador tentar aceder a uma das páginas de abcNet, cujo acesso se encontra condicionado pela prévia identificação do utilizador, através de um login, evitando desta forma a possibilidade de utilização abusiva por parte de pessoas não autorizadas para o efeito. Esta protecção encontra-se implementada em todas as páginas excepto wfEntry e wfLogin e segue a estrutura apresentada na Figura 92. Uma vez mais pode-se verificar a facilidade de utilização de 'clsSession' para registo de informação sobre o estado de identificação do utilizador.

```
_oSession = (clsSession) Session[ clsSession.Key ];
if ( _oSession.Login == false )
    Response.Redirect( clsApplication.URL_wfEntry );
```

Figura 92 – Redireccionamento de utilizadores não identificados

wfLogin

wfLogin é a página onde se procede à identificação dos utilizadores. A identificação é constituída por um nome de utilizador e uma palavra chave cujas dimensões individuais são iguais ou superiores a 6 caracteres. A passagem para a página seguinte está condicionada à satisfação dos dois seguintes requisitos:

- nome de utilizador e palavra chave reconhecidas;

- o utilizador deve estar inscrição num curso.

Caso as condições não sejam satisfeitas, o utilizador mantém-se nesta página. A identificação de utilizadores por intermédio de nome e palavra chave, que são introduzidas em caixas de texto, permitem a utilização dos controlos de validação disponibilizados por ASP.NET. Foram utilizados dois controlos de validação para cada uma das caixas de texto:

- *RequiredFieldValidator*: para a obrigatoriedade de introdução de dados;
- *RegularExpressionValidor*: introdução de pelo menos 6 caracteres alfanuméricos.

Desta forma evita-se que algumas das validações efectuadas na identificação de utilizadores, provoque não só uma comunicação com o servidor, mas também a ocupação deste para tratamento de informação incorrecta e inconsequente. Uma alternativa poderia ser a utilização de scripts do lado do cliente para efectuar a validação. Visto ASP.NET disponibilizar controlos específicos para o efeito, a opção recaiu na sua utilização.

Após uma correcta identificação, é guardada no AS a informação sobre o utilizador, nomeadamente, nome, perfil, cursos em que se encontra inscrito, estado de identificação, etc.

Sempre que esta página é acedida, origina a transição do utilizador para o estado de não identificado, o que o obriga a um novo processo de identificação, de modo a poder transitar para as páginas de acesso condicionado. Em termos globais, o estado de identificação do utilizador é gerido da seguinte forma:

- sempre que se inicia uma sessão, é colocado a não identificado;
- sempre que requisita a página wfLogin, é colocado a não identificado;
- sempre que se identifica correctamente em wfLogin, é colocado a identificado.

wfTeacherHome

wfTeacherHome é a página que é apresentada aos utilizadores de perfil professor após a sua correcta identificação em wfLogin.

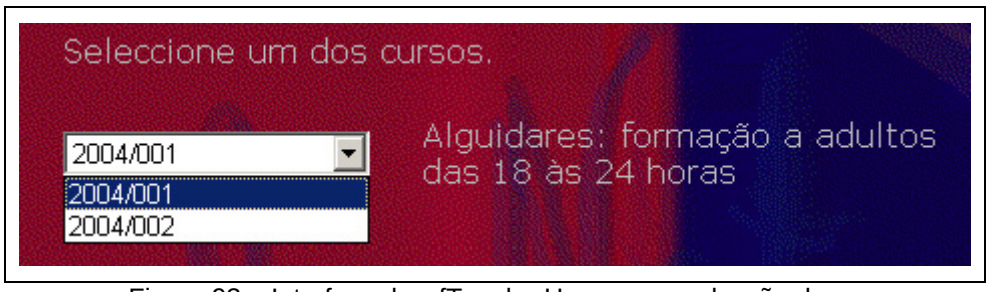

Figura 93 – Interface de wfTeacherHome para selecção do curso

Nesta página são apresentados ao professor, todos os cursos onde se encontra alocado para prestação de serviço docente. O professor deve optar por um dos cursos e de seguida seleccionar no menu a tarefa que pretende desempenhar, quer seja a participação quer seja a configuração de uma acção de formação.

wfPlanning

wfPlanning é a página onde o professor efectua a configuração e planeamento das unidades de alfabetização para uma dada acção de formação. wfPlanning disponibiliza não só os modelos Mod1Alf e Mod2Alf para a configuração das UniAlf, como também um interface específico para o ordenamento das mesmas UniAlf. A estrutura geral HTML encontra-se apresentada na Figura 94.

Da Figura 94 ressalta a não existência de elementos de organização da informação. A organização encontra-se definida por intermédio de CSS que definem a localização dos diversos elementos. Os elementos utilizados para o efeito são os elementos que possuem CSS associados e que definem, entre outros atributos, a localização das estruturas por eles englobadas.

```
<form id="wfPlanning" method="post" runat="server">
    <div id="idMenu">Tipos:
        <asp:dropdownlist id="idDdlSelection" runat="server"
                          Width="99px" AutoPostBack="True">
            <asp:ListItem Value="Wr1">Wr1</asp:ListItem>
            <asp:ListItem Value="Wr2">Wr2</asp:ListItem>
            <asp:ListItem Value="Order">Order</asp:ListItem>
        </asp:dropdownlist>
    </div>
    <div id="idWucConfig" class="clsText" runat="server">
        <ucl:wucWr1 id="idWucWr1" runat="server"></ucl:wucWr1>
        <ucl:wucWr2 id="idWucWr2" runat="server"></ucl:wucWr2>
        <ucl:wucOrdering id="idWucOrdering" runat="server"></ucl:wucOrdering>
    </div>
    <div id="idFooter" runat="server">
         <ucl:wucFooter id="idWucFooter" runat="server">
    </ucl:wucFooter></div>
    <div id="idImgBackground" runat="server"></div>
</form>
```

Figura 94 – Estrutura HTML de wfPlanning

Podemos também verificar a reutilização dos WUC wucWr1, wucWr2, wucOrdering e wucFooter. Os 3 primeiros WUC são responsáveis pela implementação dos interfaces associados à configuração das acções de formação.

A *dropddownlist* é o único elemento novo na estrutura apresentada, e cuja função é permitir que o professor possa seleccionar, em cada instante, qual das 3 operações de configuração disponíveis pretende executar: Mod1Alf, Mod2Alf ou ordenação. Assim, em cada instante, no wucPlanning, apenas um dos

WUC associados à configuração, se encontra visível. A Figura 95 apresenta o interface disponibilizado para o efeito, tendo como fundo a configuração efectuada para uma Uni1Alf.

Figura 95 – Interface disponibilizado para selecção de operação de configuração

wfCSchedule

wfCSchedule é a página onde é apresentado o planeamento efectuado para uma dada acção de formação. O planeamento é apresentado como uma sequência de botões, cada um com a responsabilidade da ligação a uma UniAlf. wfCSchedule suporta um máximo de 39 UniAlf, dispostas em 3 colunas, que são apresentadas segundo o planeamento efectuado pelo professor.

```
<body MS_POSITIONING="GridLayout">
    <form id="wfCSchedule" method="post" runat="server">
        <div id="idCol1" runat="server"></div>
        <div id="idCol2" runat="server"></div>
        <div id="idCol3" runat="server"></div>
        <div id="idFooter" runat="server">
            <ucl:wucFooter id="idWucFooter" runat="server"></ucl:wucFooter>
        </div>
        <div id="idImgBackground" runat="server"></div>
    </form>
</body>
```

Figura 96 – Estrutura geral HTML de wfCSchedule

A Figura 96 apresenta a estrutura HTML geral de wfCSchedule em que se pode verificar a existência de três elementos que definem a localização das 3 colunas. Além destes elementos, apenas são utilizados elementos já nossos conhecidos.

O número de UniAlf pode variar de acção de formação para acção de formação, suportando wfPlanning um máximo de 39 unidades. A selecção de uma dada UniAlf é realizada pelo premir com o rato num dos botões disponibilizados para o efeito. A identificação das UniAlf é efectuada pelo texto associado a cada botão, e que corresponde à designação atribuída pelo professor no processo de

configuração. Os botões são criados em tempo de execução e em número correspondente ao número de UniAlf da acção de formação, num máximo de 39 tal como já foi dito.

wfCTemplateWr1

wfCTemplateWr1 é a página onde se processam as componentes de formação relativas às Uni1Alf , que efectuam uma implementação directa do método de Paulo Freire para as acções de alfabetização. Tendo em vista que abcNet não se vocaciona para um público alvo específico, mas sim para vários públicos alvo, desde crianças a adultos, houve algum cuidado no sentido de se efectuar uma separação entre a formatação da página e os conteúdos associados à formação. Esse cuidado pode ser constatado na estrutura HTML utilizada em wfCTemplate_wr1 que se encontra apresentada na Figura 97. Pode-se verificar que wfCTemplate.aspx, além do elemento <form> apenas contém elementos os quais por si só não representam qualquer conteúdo nem qualquer apresentação a incluir na página.

```
<form id="wfCTemplateWrl_l" method="post" runat="server">
    <div id="idMenu" runat="server"></div>
    <div class="FrameBorder" id="idFrameIntrod" runat="server"></div>
    <div class="FrameBorder" id="idFrameWork" runat="server"></div>
    <div class="inFrameBorderWork" id="idFrameSyllables" runat="server"></div>
    <div class="inFrameBorderWork" id="idFrameSyllOther" runat="server"></div>
    <div class="inFrameBorderWork" id="idFrameComposition"
         align="center" runat="server"></div>
    <div class="FrameBorder" id="idFrameSolution" runat="server"></div>
    <div class="inFrameBorder" id="idIntrodText" runat="server"></div>
    <div class="inFrameBorder" id="idTitle" align="center" runat="server"></div>
    <div class="inFrameBorder" id="idFrameImage" align="center" runat="server"></div>
    <div class="inFrameBorder" id="idFrameWord" align="center" runat="server"><br>
        <br><br><div id="idWordGenerator" runat="server"> </div>
        <br><br><div id="idWordGeneratorSyllables" runat="server"></div>
    </div>
    <div class="inFrameBorder" id="idDivNext" runat="server"></div>
    <div class="inFrameBorder" id="idDivHelp" runat="server"></div>
    <div class="inFrameBorder" id="idDivPlayer" runat="server">player</div>
    <div id="idFooter" runat="server"></div>
    <div id="idImgBackground" runat="Server"></div>
</form>
```

Figura 97 – Estrutura HTML de wfCTemplate_Wr1

Os elementos , além do seu ID, também possuem uma *class* associada, que em comunhão com CSS permitem a definição da estrutura e parte da apresentação da página. A propriedade *class* é utilizada para a definição de propriedades que podem ser partilhadas por vários elementos, enquanto que o ID permite a definição de propriedades exclusivas de cada elemento. Desta forma podem-se efectuar alterações na formatação da página, ajustando-a porventura a cada um dos públicos alvo, mantendo os elementos associados aos conteúdos num nível distinto de definição e inclusão. A estratégia aqui utilizada é análoga à anteriormente apresentada para separação da construção das páginas HTML em 3 blocos.

Nesta página, tudo o que é conteúdo é criado em tempo de execução no servidor, e incluído na página através da referência ao elemento que o deve suportar.

wfCTemplate_Wr2

wfCTemplate_Wr2 é a página onde se processam as componentes de formação relativas às Uni2Alf. wfCTemplate_Wr2 é em tudo semelhante a wfCTemplate_Wr1 pelo que se torna desnecessário efectuar uma descrição da sua implementação.

7.3 Conclusões

Vamos apresentar algumas conclusões sobre o trabalho realizado e descrito neste capítulo.

Módulo de Administração

De todos os módulos, o MódAdm é o único que teve um desenvolvimento minimalista ambicionando apenas satisfazer os requisitos que de outra forma dificilmente poderiam ser implementados manualmente. De entre estes destacam-se os elementos pedagógicos associados às sílabas.

Módulo de Formação

O MódFor pode ser considerado o módulo central da aplicação abcNet sendo a componente com maior visibilidade. É através do MódFor que se implementam todos os interfaces associados às acções de alfabetização. Do ponto de vista de utilização, é de realçar a arquitectura baseada em EntAlf tendo sido desenvolvidas, até ao momento, 2 EntAlf. É esta arquitectura que sustenta não só o ensino centrado no aluno mas também a capacidade de expansão de abcNet pela definição de novas EntAlf.

Os interfaces desenvolvidos necessitam de uma melhoria, principalmente os associados às UniAlf. Os interfaces foram essencialmente desenvolvidos com recurso a HTML que não é concerteza o ambiente que maiores vantagens oferece para uma aplicação do tipo de abcNet. Uma alternativa é o recurso à utilização do *Macromedia Flash* que é uma ferramenta vocacionada para o desenvolvimento de interfaces dinâmicos.

Do ponto de vista do desenvolvimento é de realçar a continuidade da preocupação subjacente à definição de normas que de alguma forma potenciam o aumento de produtividade não só durante o desenvolvimento mas também durante a fase de manutenção. De entre essas normas pode-se destacar as

designações de ficheiros, designações de métodos e propriedades, estrutura para os ambientes AA e AS, metodologia para a estrutura, apresentação e conteúdos de páginas HTML, etc.

8 Utilização de abcNet

Neste capítulo apresentam-se quer os interfaces quer a lógica a seguir na utilização de abcNet. A informação aqui apresentada pode ser complementada com a disponibilizada em 10.1 Anexo I – Controlos e também em [OSCAR-CELDA].

8.1 Perfil Administrador

Os interfaces associados ao perfil de administrador, na essência, são os já apresentados em 7.1.2. O administrador tem duas funções essenciais a desenvolver:

- criação dos elementos pedagógicos cujos interfaces se encontram apresentados em 7.1.2;
- criação da estrutura de necessária à existência das acções de formação; esta estrutura é criada directamente na base de dados e, por acção de formação, consiste em:
    - criação dos utilizadores de perfil aluno e professor;
    - criação da acção de formação;
    - alocação dos utilizadores de perfil aluno e professor à acção.

O restante trabalho que é necessário para o desenrolar da acção de formação é da responsabilidade do professor.

8.2 Perfil Professor

Em termos de sequência de operações, o professor deve executar:

- identificar-se no abcNet através de um *login*;
- seleccionar uma das acções de formação em que se encontra alocado;
- optar pelo modo de operação que pretende continuar: modo aula ou modo configuração:
    - no modo aula efectua um acompanhamento da acção de formação como se de um aluno se tratasse;
    - no modo configuração tem acesso aos interfaces de configuração.

Vamos apresentar com maior detalhe cada uma das situações. A situação referente ao modo aula é apresentado no parágrafo associado perfil aluno.

Interface de login

Ao aceder ao portal de abcNet é apresentado o interface indicado na Figura 98. O professor deve premir o botão de *login* de modo a transitar para o interface que lhe permite efectuar a sua identificação.

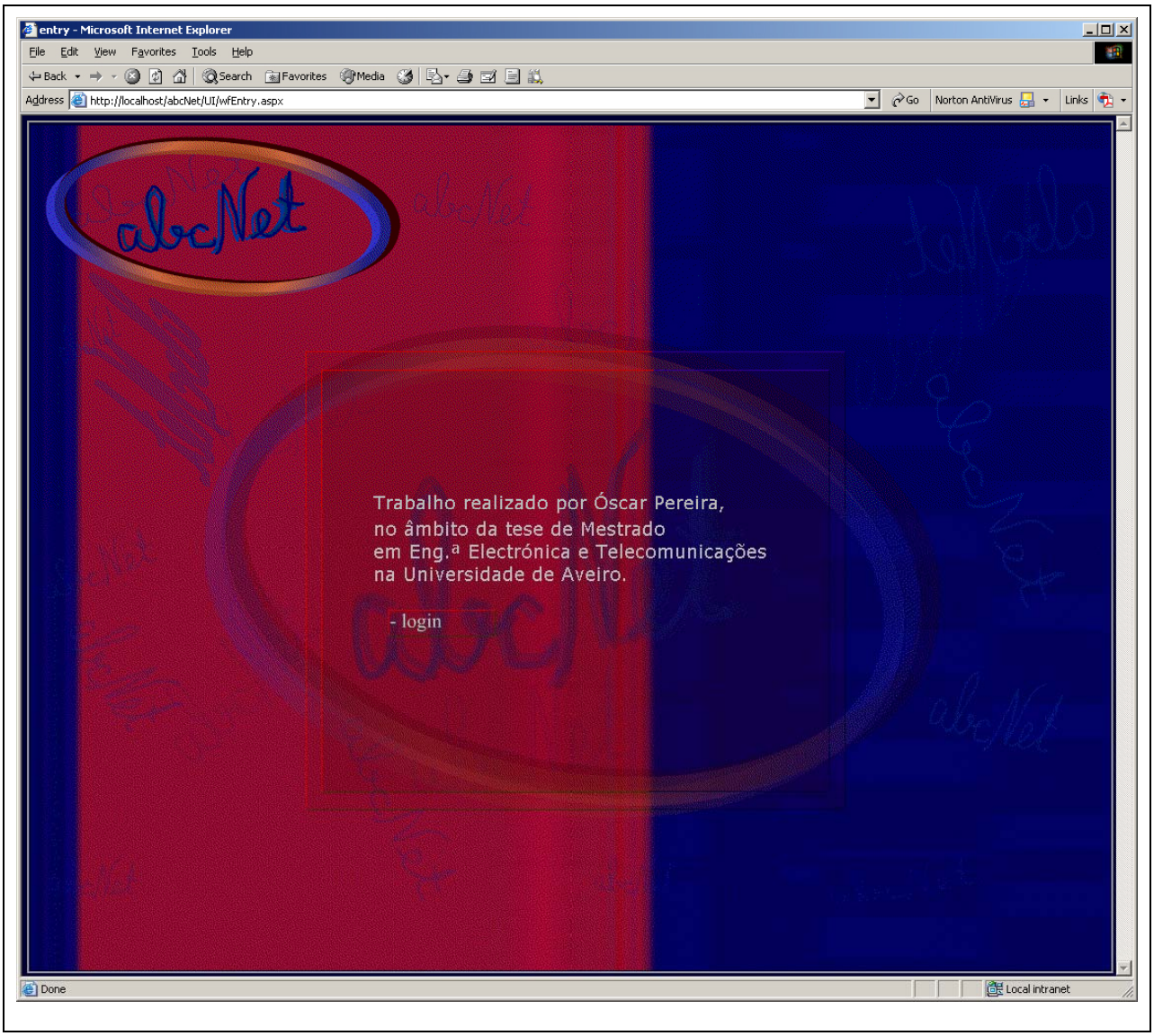

Figura 98 – Interface de apresentação de abcNet

O interface que permite a sua identificação encontra-se apresentado na Figura 99. Estes dois interfaces são também comuns para o perfil de aluno.

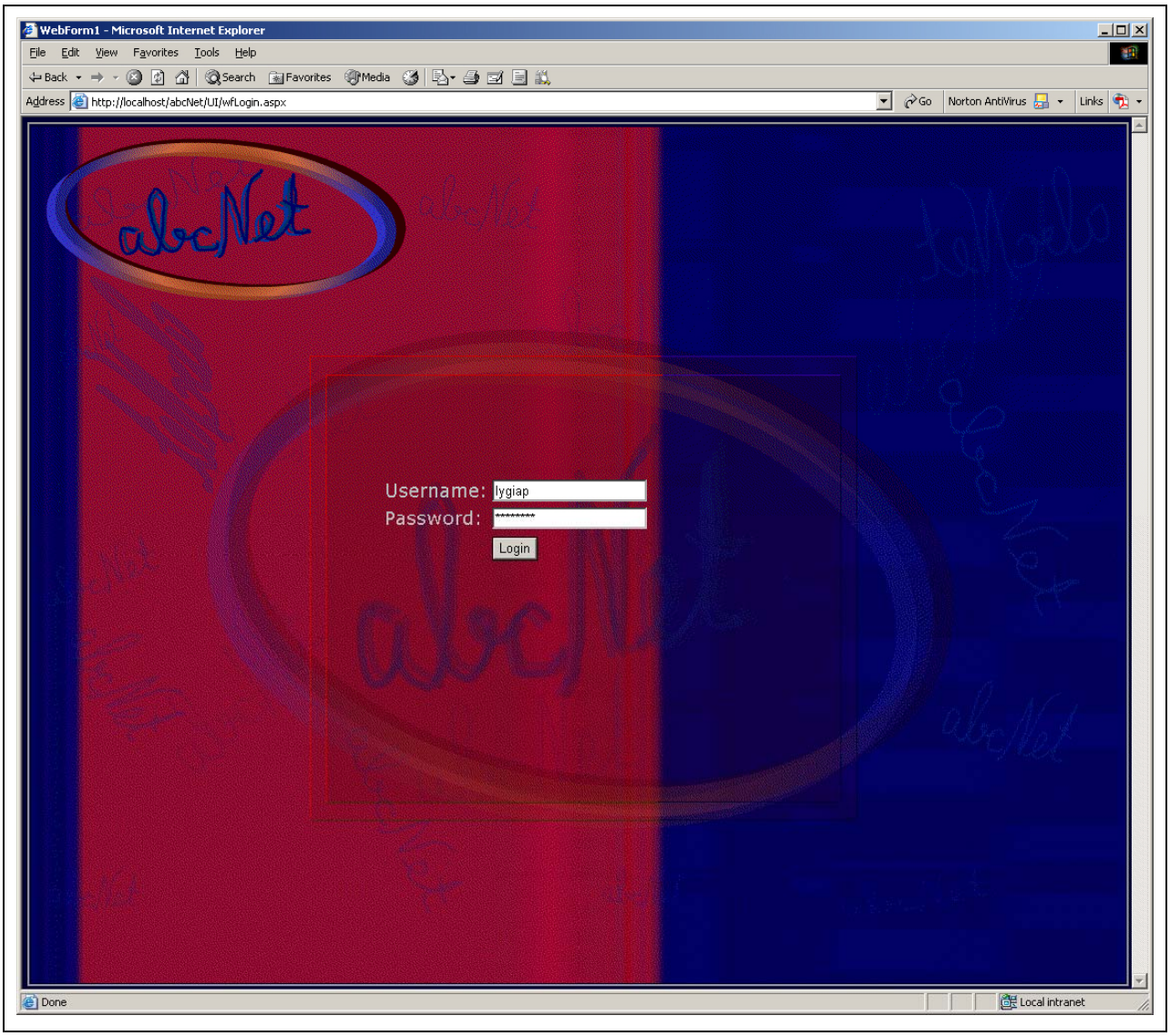

Figura 99 – Interface de identificação do utilizador

Interface de Decisão

Após a correcta identificação é solicitado ao professor que seleccione para qual das acções de formação a que se encontra alocado pretende transitar. O interface apresentado ao professor encontra-se indicado na Figura 100.

Após a selecção da acção de formação o professor pode seleccionar um de dois modos de operação:

- no modo aula o professor tem acesso ao conteúdo da acção tal como se de um aluno se tratasse; a transição para este modo é obtida através do botão “aula”;
- no modo configuração o professor pode configurar a acção de formação; a transição para este modo é obtida através da tecla “planear”.

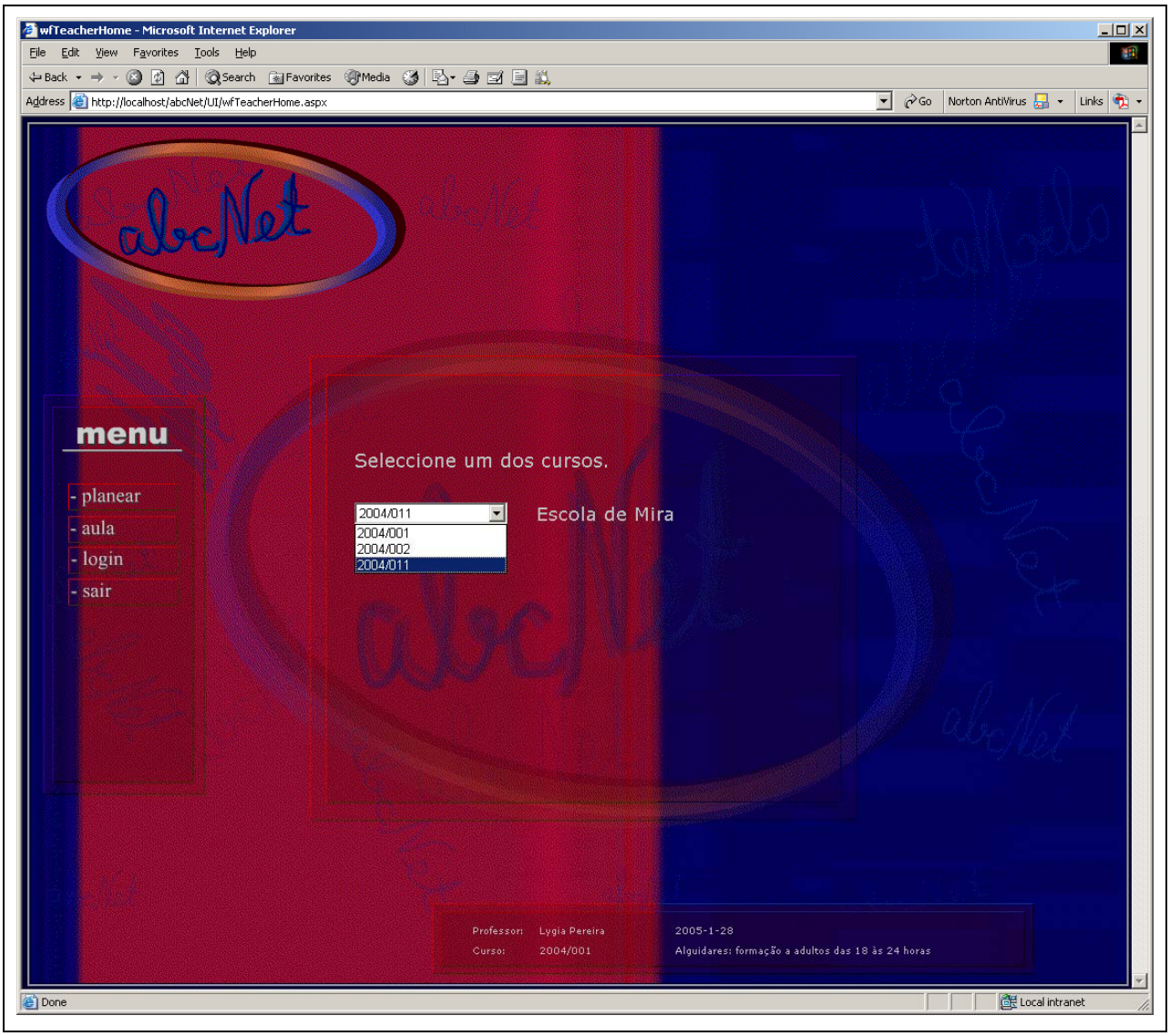

Figura 100 – Interface de escola

Interface Configuração

O interface de configuração é de acesso exclusivo dos professores. Este interface dá acesso a três outros interfaces cada qual com o seu objectivo específico:

- interface Wr1: associado ao Mod1Alf;
- interface Wr2: associado ao Mod2Alf;
- interface de ordenação: ordenação das UniAlf.

A selecção de um dos interfaces é obtida pela selecção de uma das opções apresentadas na caixa de selecção tal como se pode verificar na Figura 101.

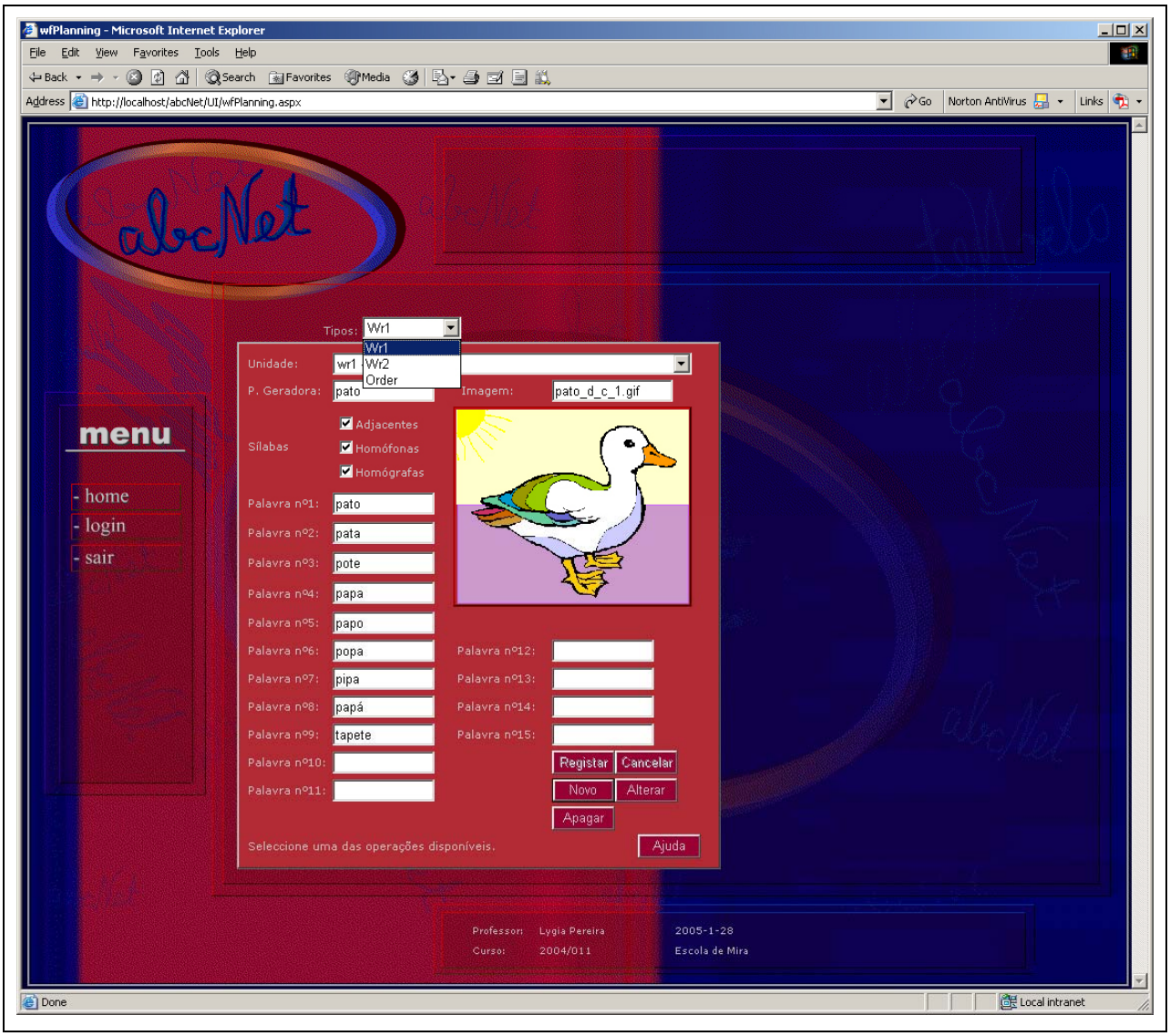

Figura 101 – Interface planear

Interface Wr1

O interface Wr1 implementa o Mod1Alf que permite a configuração de Uni1Alf. A Figura 102 apresenta o interface base que inclui a configuração de uma Uni1Alf já efectuada.

O interface Wr1 permite não só criar Uni1Alf mas também alterar as já existentes. As funções de edição e de criação estão disponíveis por intermédio do painel de botões que se pode ver no canto inferior direito. Cada um dos botões dá acesso a um estado de operação ou a uma acção sobre o estado de operação vigente. Os estados possíveis são:

- nova Uni1Alf: corresponde a criar uma Uni1Alf nova e está disponível através do botão "Novo";
- alterar Uni1Alf: corresponde à edição de uma Uni1Alf já existente e está disponível através do botão "Alterar";

- apagar Uni1Alf: corresponde à eliminação de uma Uni1Alf existente e está disponível através do botão “Apagar”;
- base: corresponde ao estado de repouso a partir do qual se pode transitar para um dos três estados apresentados.

A saída de um dos estados, excepto do estado base, está disponível através dos botões “Registar” e “Cancelar”. O botão “Registar” actualiza a informação introduzida ou alterada e definida no estado em curso. O botão “Cancelar” repõe a informação existente e anterior à entrada no estado em curso.

O botão “Ajuda” apresenta uma ajuda ao utilizador com descrição detalhada de todas as operações possíveis de realização e suas implicações.

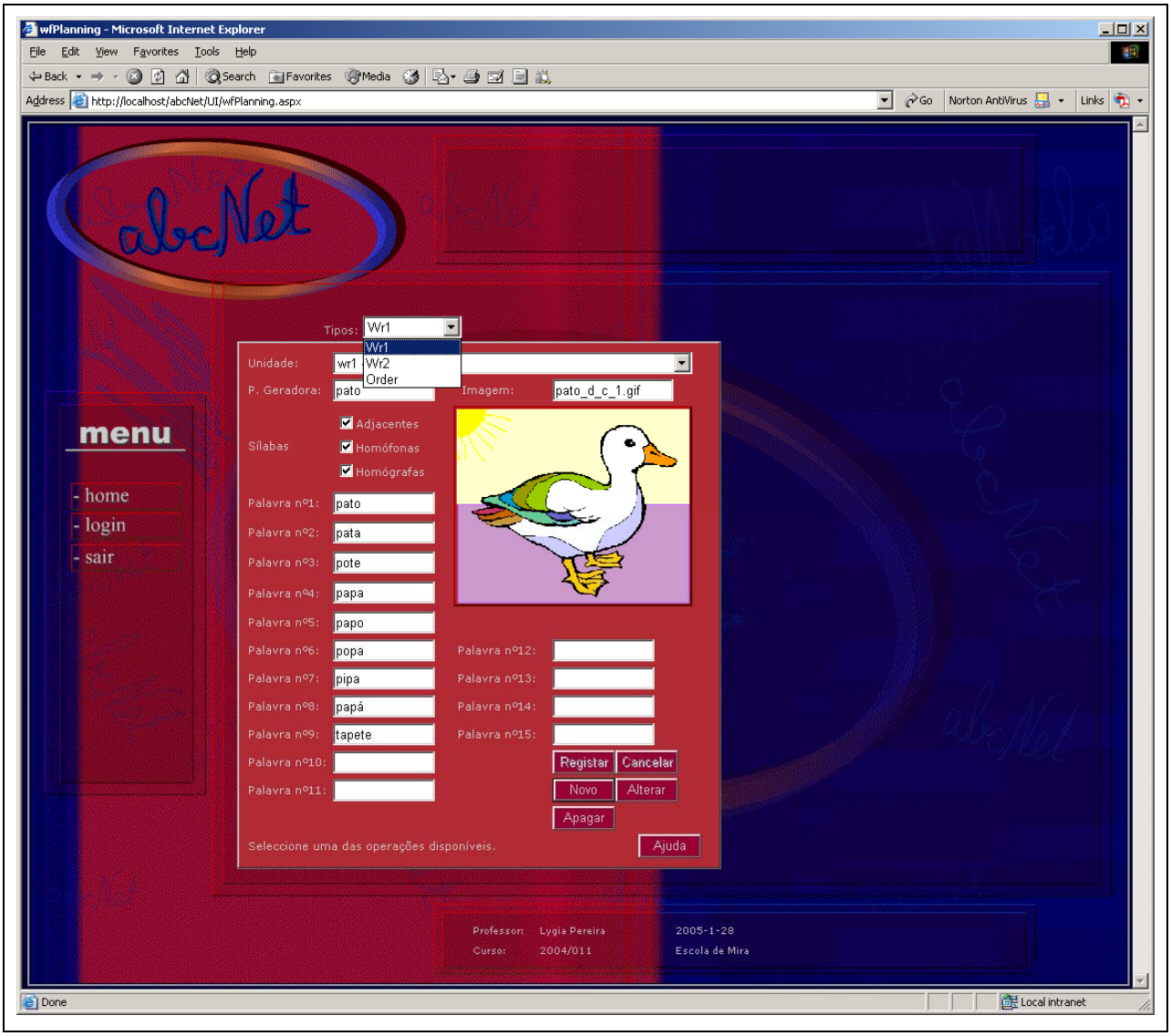

Figura 102 – Interface Wr1 base

A Figura 103 apresenta o interface de edição de uma Uni1Alf. Podemos verificar que relativamente ao interface apresentado na Figura 102 as caixas de texto se transformaram em caixas de selecção que permitem ao professor efectuar uma opção entre as possibilidades apresentadas. As opções disponibilizadas abrangem a palavra geradora, a imagem geradora e as palavras a construir pelo aluno.

A ordenação das palavras a construir pelo aluno é solidária com a ordenação das caixas de selecção das palavras. A Figura 103 apresenta a caixa de selecção, de modo expandido, associada à segunda palavra a ser construída pelo aluno. Ao professor basta optar por uma das palavras aí existentes.

Além das palavras, o professor pode também indicar que outros grupos de sílabas devem pertencer à ficha de descoberta para além dos grupos de sílabas família das sílabas da palavra geradora.

A designação aqui dada à Uni1Alf é a que aparece posteriormente para a identificação da mesma em todas as localizações onde a Uni1Alf se encontrar presente.

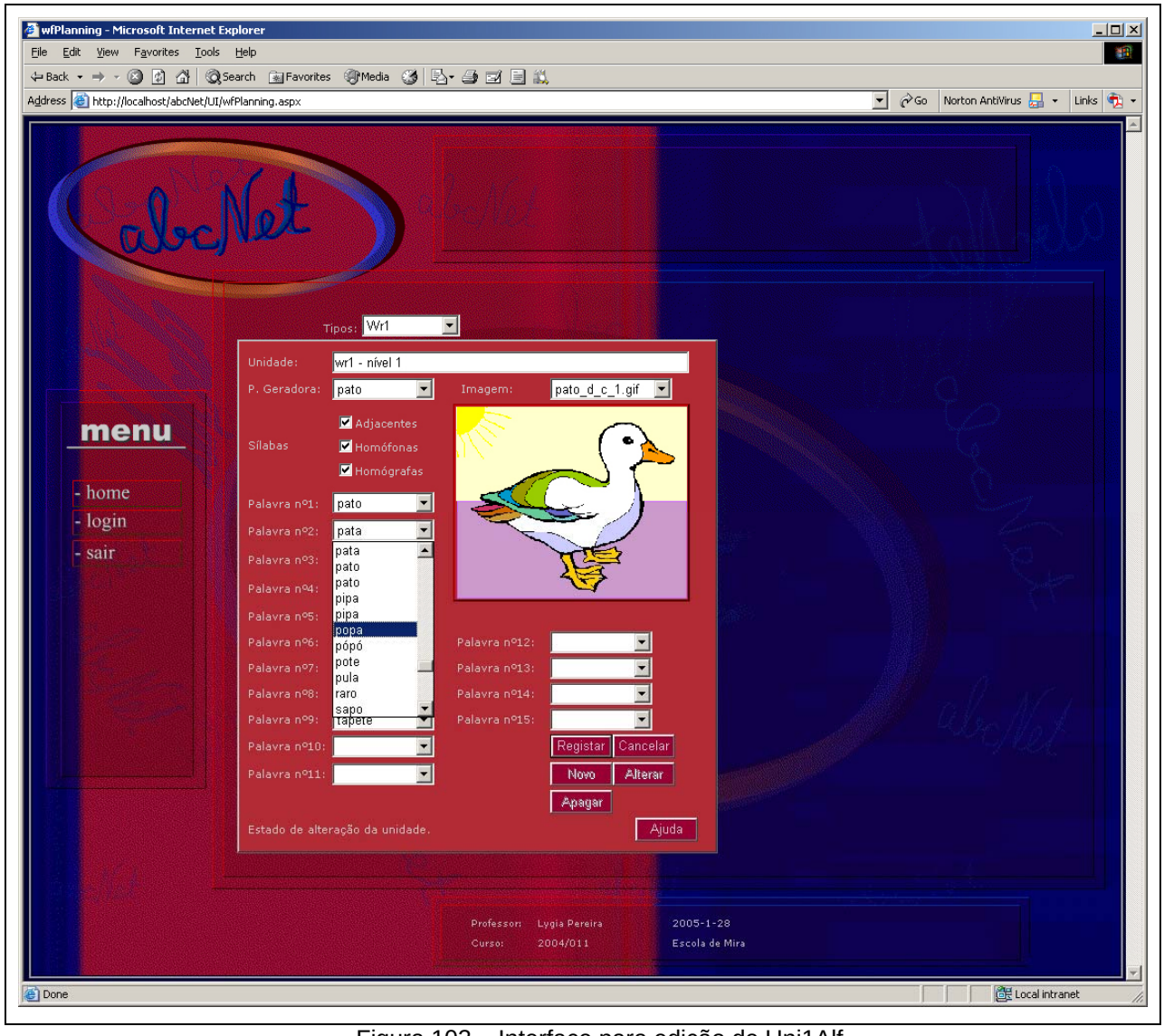

Figura 103 – Interface para edição de Uni1Alf

Interface Wr2

O interface Wr2 implementa o Mod2Alf que permite a configuração de Uni2Alf. A Figura 104 apresenta o interface Wr2 base com uma Uni2Alf já configurada.

A Figura 105 representa a mesma Uni2Alf mas com apresentação de algumas das palavras seleccionadas para serem construídas. As palavras que se encontram repetidas estão associadas às fases iniciais de aprendizagem de utilização de abcNet. A cada palavra repetida está associado um grau de dificuldade que cresce à medida que aumenta a ordenação da palavra. O grau de dificuldade de determinada palavra está associado, entre outros factores, aos elementos de ajuda disponíveis para a construção da mesma.

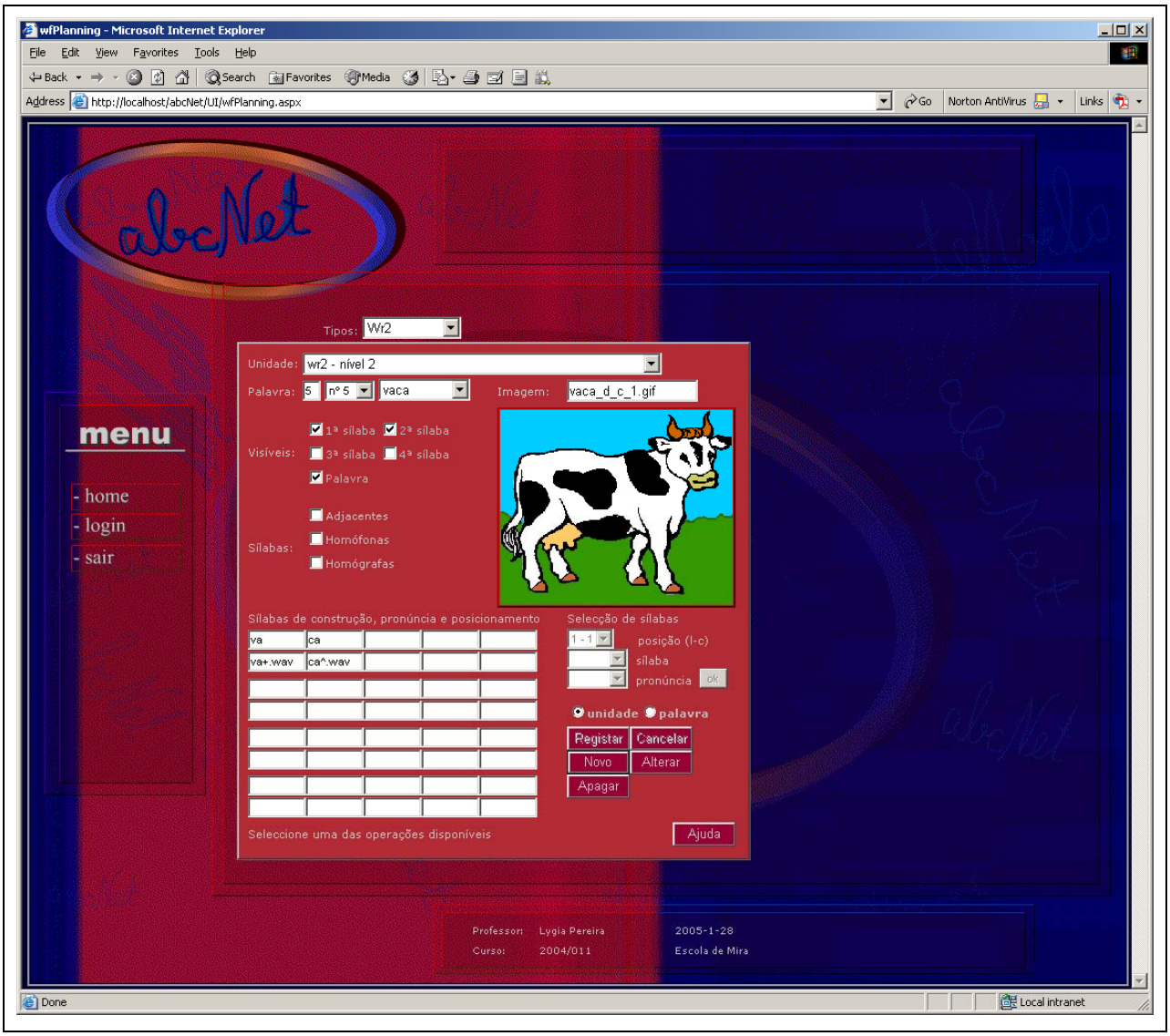

Figura 104 - Interface Wr2 base

A edição das Uni2Alf são um pouco mais elaboradas que a das Uni1Alf. Esta situação deve-se não ao facto de cada Uni2Alf poder conter várias palavras mas sim ao facto de a cada palavra estar associada uma ficha de descoberta dedicada. Assim, em cada instante tem de se saber se a edição da Uni2Alf é respeitante à sua designação ou se é respeitante à palavra a construir e da respectiva ficha de descoberta. A opção por uma destas situações encontra-se disponível nos botões de opção que se situam por cima do painel de botões no canto inferior direito. A opção pelo botão "unidade" indica que se pretende editar a Uni2Alf; a opção por "palavra" indica que se pretende editar a palavra e/ou a ficha de descoberta associada à palavra que se encontrar visível.

Cada um dos botões do painel dá acesso a um estado de operação ou a uma acção sobre o estado de operação vigente. Os estados possíveis são:

- botão de opção "unidade":
  - nova Uni2Alf: corresponde a criar uma Uni2Alf nova e está disponível através do botão "Novo";
  - alterar Uni2Alf: corresponde à edição de uma Uni2Alf já existente e está disponível através do botão "Alterar";
  - apagar Uni2Alf: corresponde à eliminação de uma Uni2Alf existente e está disponível através do botão "Apagar";
  - base que corresponde ao estado de repouso a partir do qual se pode transitar para um dos três estados apresentados;
- botão de opção "palavra":
  - nova palavra: corresponde a criar uma nova palavra na Uni2Alf e está disponível através do botão "Novo";
  - alterar palavra: corresponde à edição de uma palavra da Uni2Alf já existente e está disponível através do botão "Alterar";
  - apagar palavra: corresponde à eliminação de uma palavra da Uni2Alf existente e está disponível através do botão "Apagar";
  - base que corresponde ao estado de repouso a partir do qual se pode transitar para um dos três estados apresentados.

A saída de um dos estados, excepto do estado base, está disponível através dos botões "Registar" e "Cancelar". O botão "Registar" actualiza a informação introduzida ou alterada no estado em curso. O botão "Cancelar" repõe a informação existente e anterior à entrada no estado em curso.

O botão “Ajuda” apresenta uma ajuda ao utilizador com descrição detalhada de todas as operações possíveis de realização e suas implicações.

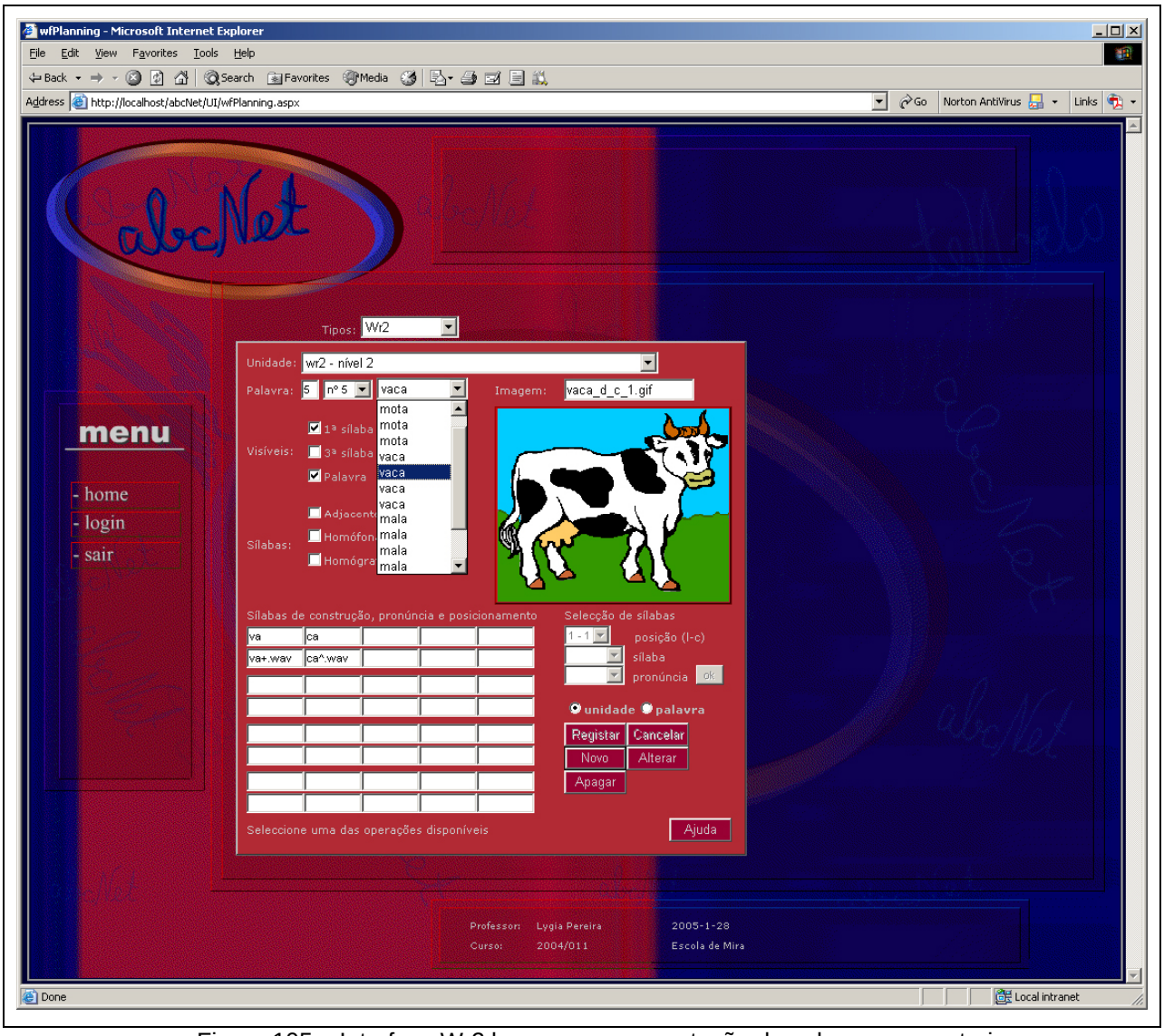

Figura 105 – Interface Wr2 base com apresentação de palavras a construir

A edição da ficha de descoberta carece de informação adicional. A edição da ficha de descoberta está englobada nos estados associadas à opção “palavra”. A inclusão de SílEnt na ficha de descoberta é um processo que se desenvolve SílEnt a SílEnt. Sobre o painel dos botões existem três caixas de selecção, que de cima para baixo são:

- posição: selecção da localização da SílEnt na ficha de descoberta a apresentar na Uni2Alf;
- sílaba: SílOrt a apresentar na posição seleccionada;
- pronúncia: SílPron a associar à SílOrt.

Assim, pode-se atribuir a cada SílOrt a pronúncia que melhor se ajustar aos objectivos definidos para a Uni2Alf. Simultaneamente, pode-se definir qual a posição a ocupar pela SílEnt na ficha de descoberta.

A Figura 106 apresenta a configuração da SílEnt a posicionar em 1 – 2 (linha 1, coluna 2).

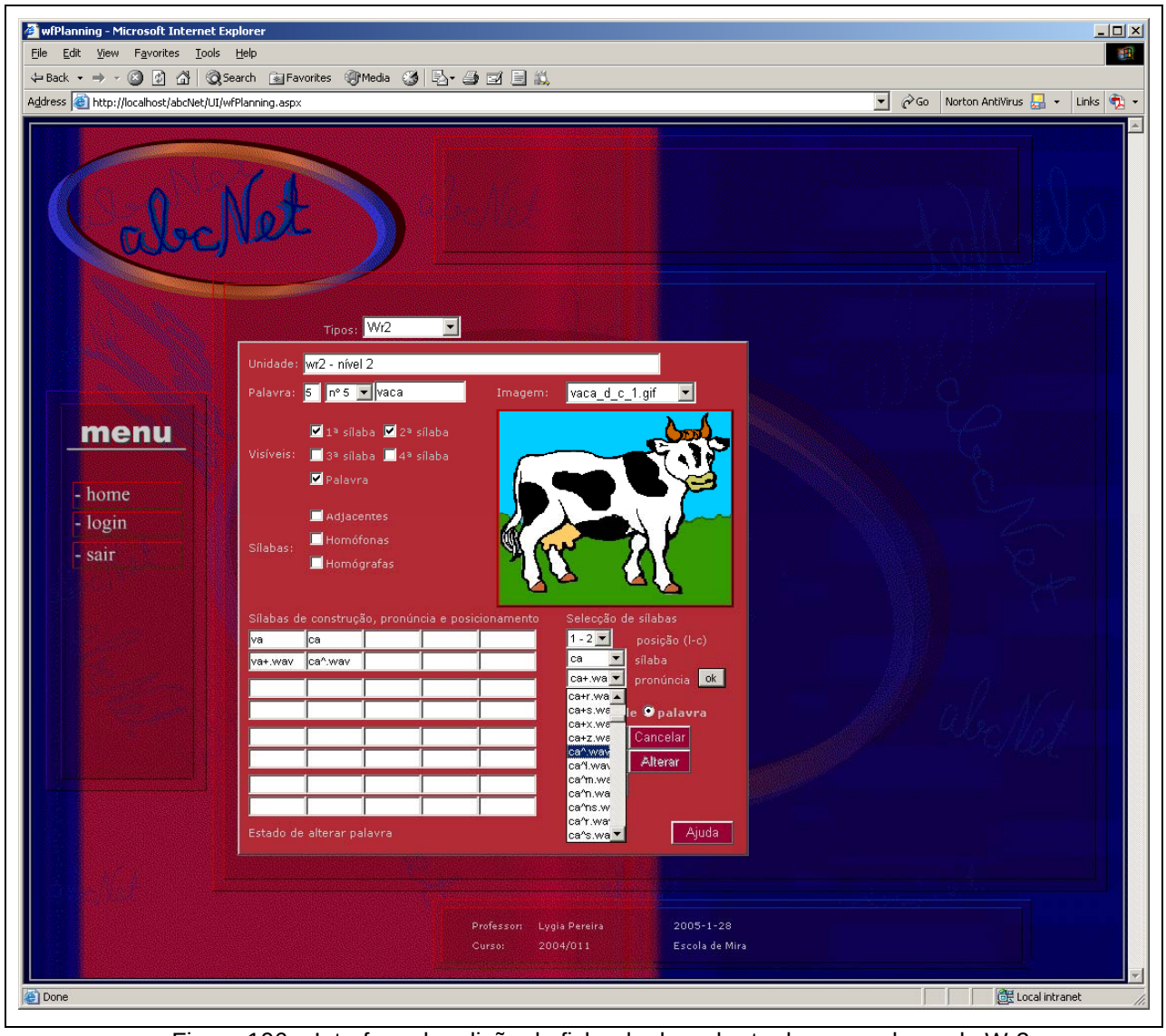

Figura 106 – Interface de edição da ficha de descoberta de uma palavra de Wr2

Interface Ordenar

O interface ordenar é o interface a utilizar para efectuar a ordenação das UniAlf configuradas quer em Wr1 quer em Wr2. A Figura 107 apresenta o interface base de ordenação onde se pode verificar a existência de uma série de UniAlf já ordenadas. No máximo são permitidas 39 UniAlf cuja ordenação é reflectida no plano apresentado aos alunos.

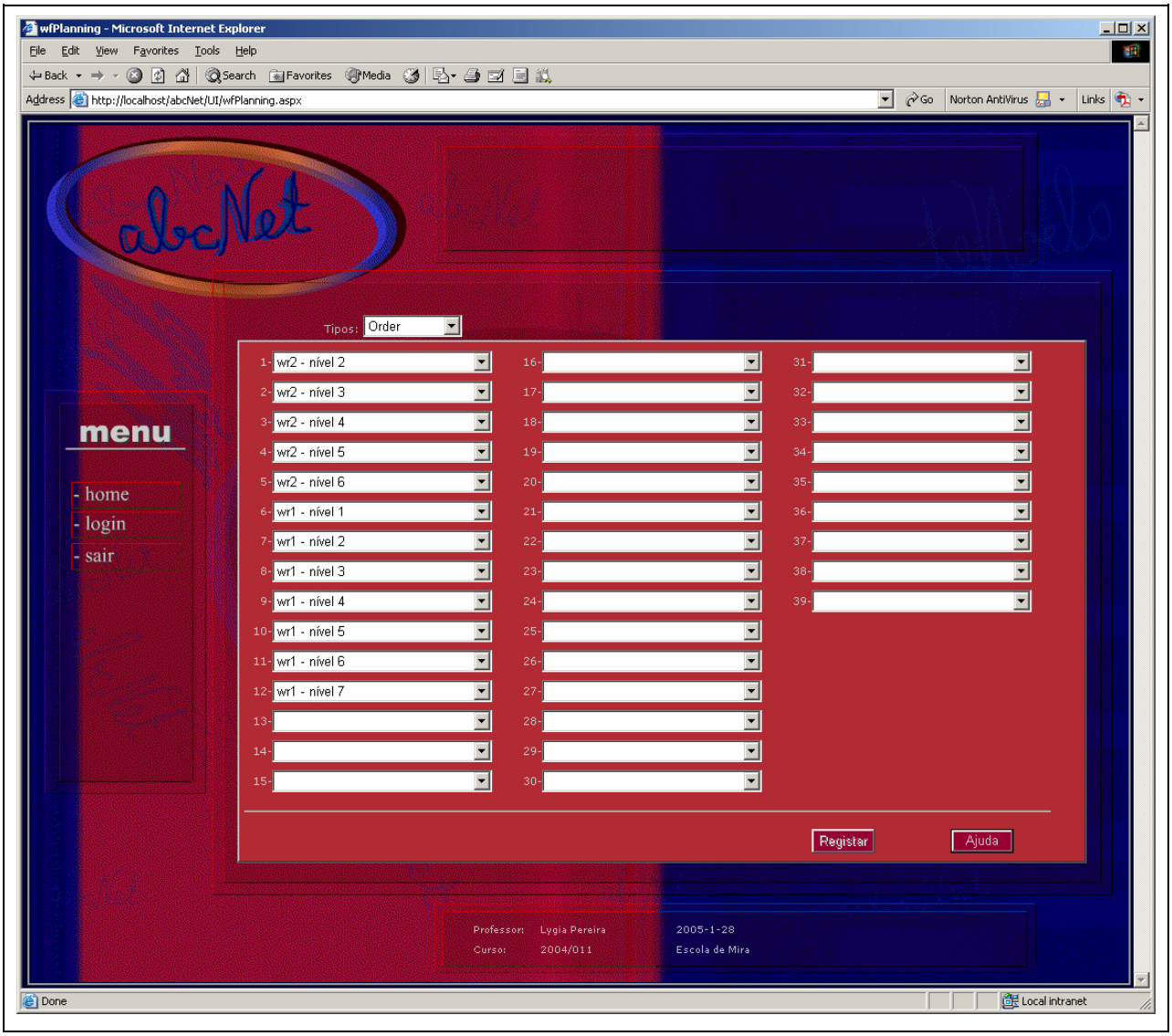

Figura 107 – Interface base de ordenação

Cada uma das primeiras caixa de selecção, em número igual ao das UniAlf configuradas, contém todas as UniAlf configuradas para a acção de formação, limitando-se o professor a seleccionar para cada caixa de selecção a UniAlf correspondente. Esta operação de ordenação pode ser visualizada na Figura 108. O botão “Registar” é utilizado para actualizar as alterações efectuadas. A operação de alteração é sujeita a uma prévia validação de modo a evitar erros associados a duplicação de UniAlf.

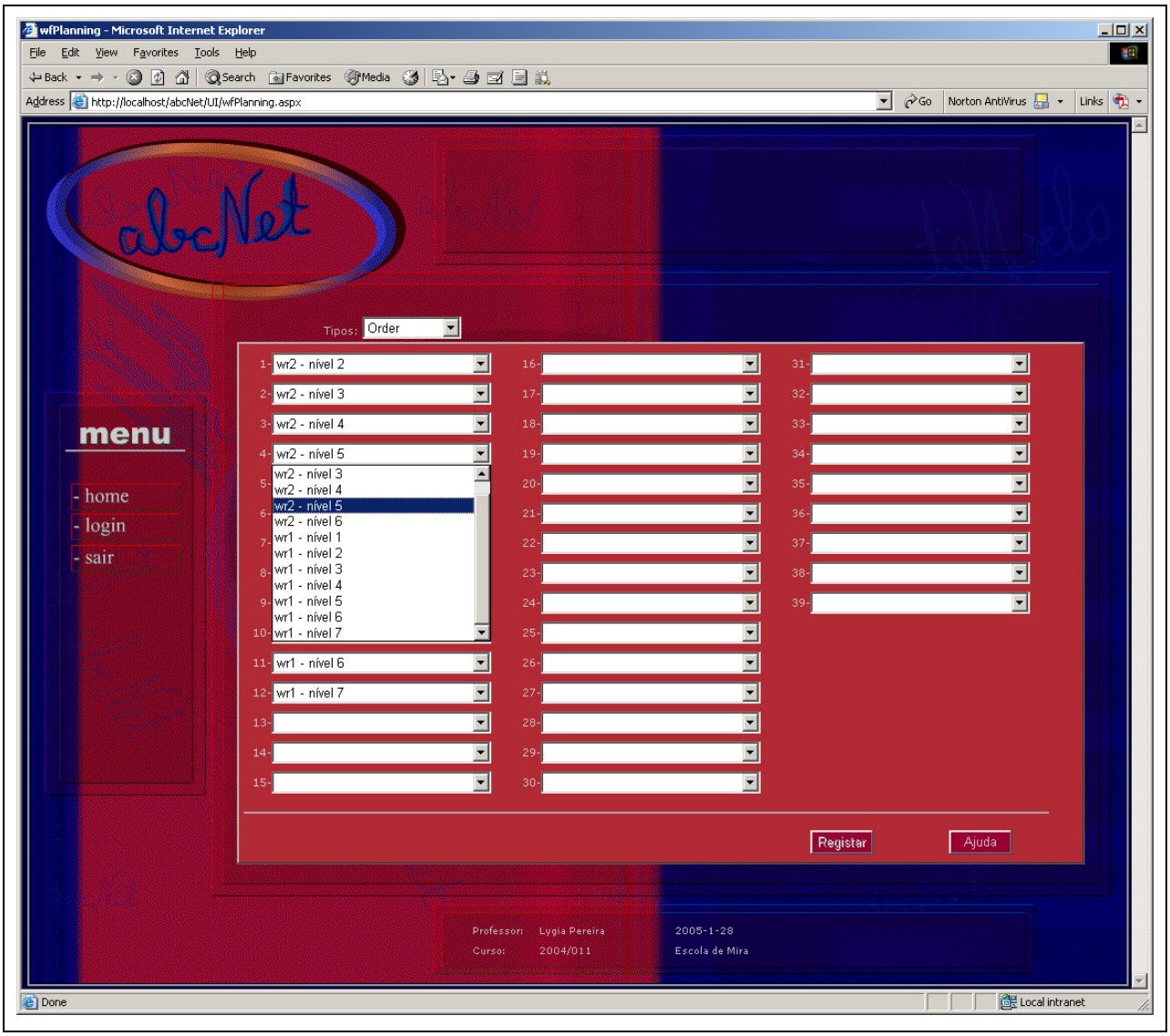

Figura 108 – Metodologia de ordenação da UniAlf.

8.3 Perfil Aluno

Os interfaces aqui apresentados são comuns aos perfis de professor e de aluno. Apenas os menus principais de navegação apresentam algumas ligeiras diferenças e que apenas têm implicação no esquema de navegação que já foi apresentado em 7.2.1. Os interfaces aqui apresentados são os correspondentes ao perfil aluno. Em termos de operação, o aluno, à semelhança do professor, deve aceder ao portal de abcNet e identificar-se. Ao contrário do professor, o aluno é imediatamente encaminhado para a acção de formação em que está inscrito.

Interface plano

O interface plano apresenta ao aluno todas as UniAlf configuradas para a sua acção de formação. As UniAlf encontram-se também ordenadas segundo os critérios definidos pelo professor no interface

ordenação tal como se pode verificar na Figura 109. Cada UniAlf é apresentada na sua correcta ordenação e é identificada pela designação atribuída nos respectivos ModAlf.

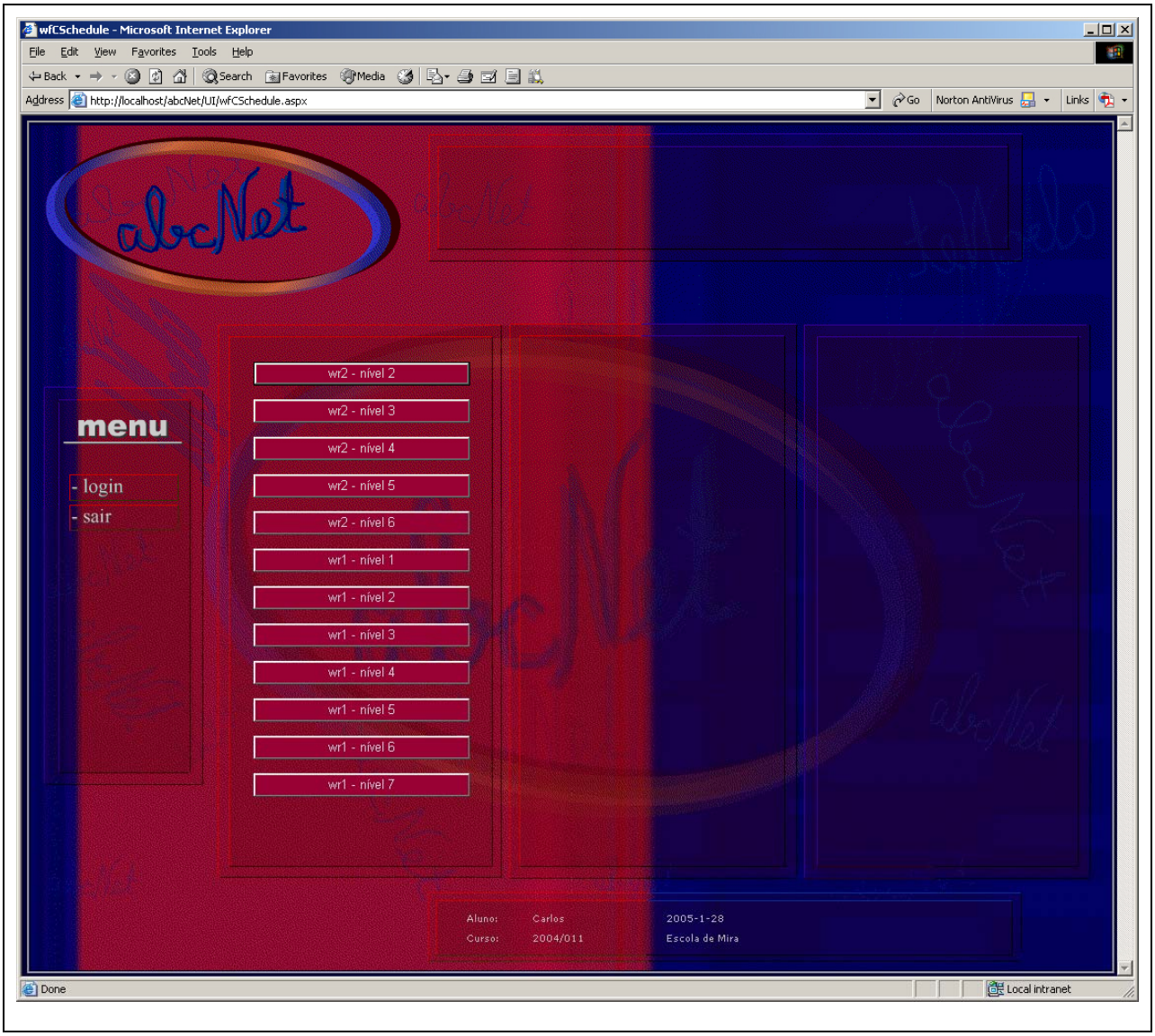

Figura 109 – Interface associado ao plano da acção de formação

O acesso a uma UniAlf é obtido pelo premir do botão a ela associado.

Vamos analisar os interfaces associados a uma Uni1Alf e a uma Uni2Alf.

Interface Uni1Alf

O interface de uma Uni1Alf é o apresentado na Figura 110.

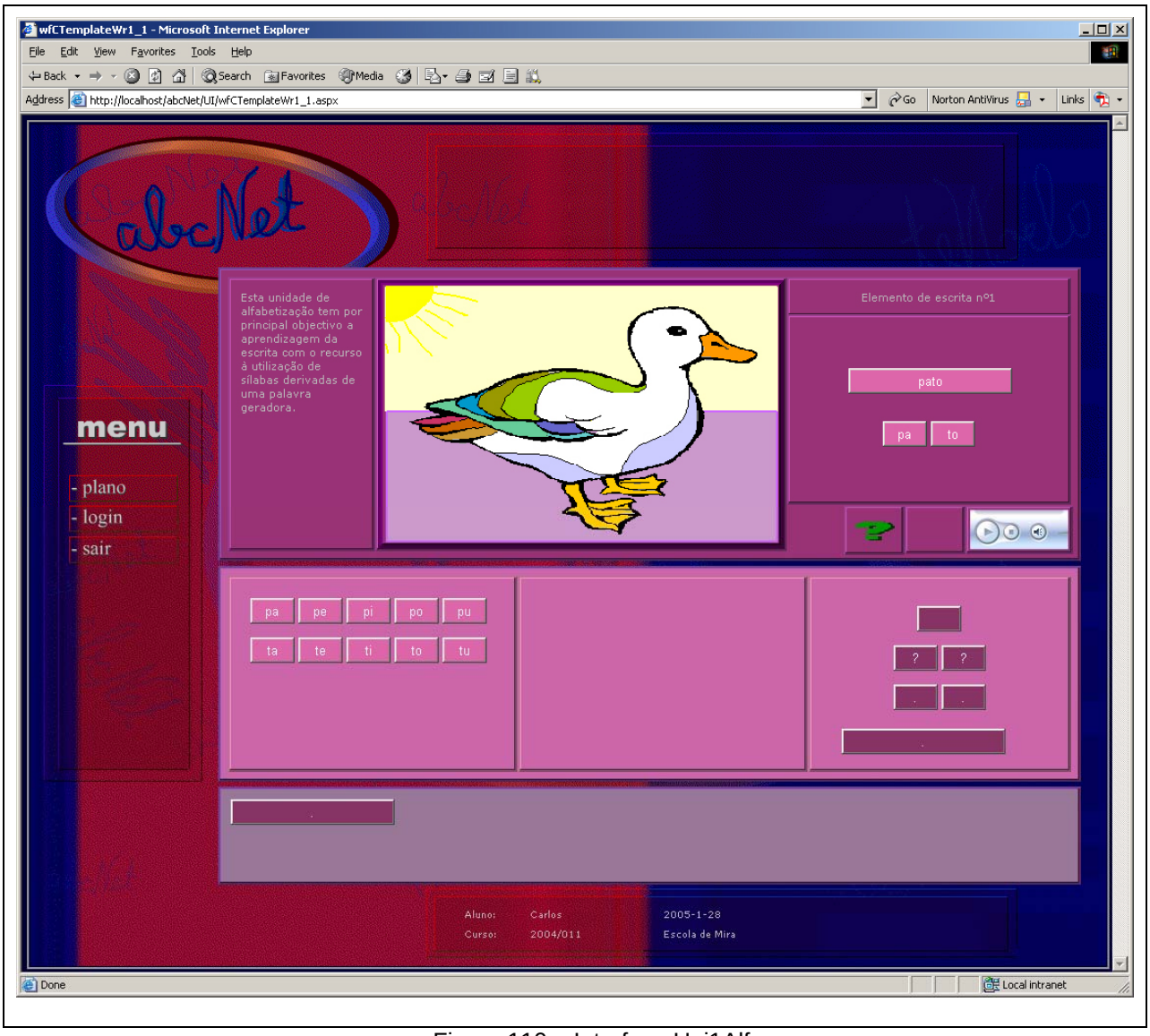

Figura 110 – Interface Uni1Alf

A partir deste interface o aluno deve construir cada uma das palavras propostas e que se apresentam uma de cada vez. As palavras construídas vão sendo apresentadas na montra e após a construção da última palavra é apresentado o interface indicado na Figura 111. O aluno deve premir o botão oval verde de modo a evoluir novamente para o interface plano onde deve seleccionar a UniAlf seguinte.

Em [OSCAR-CELDA] apresenta-se uma descrição detalhada do interface Uni1Alf.

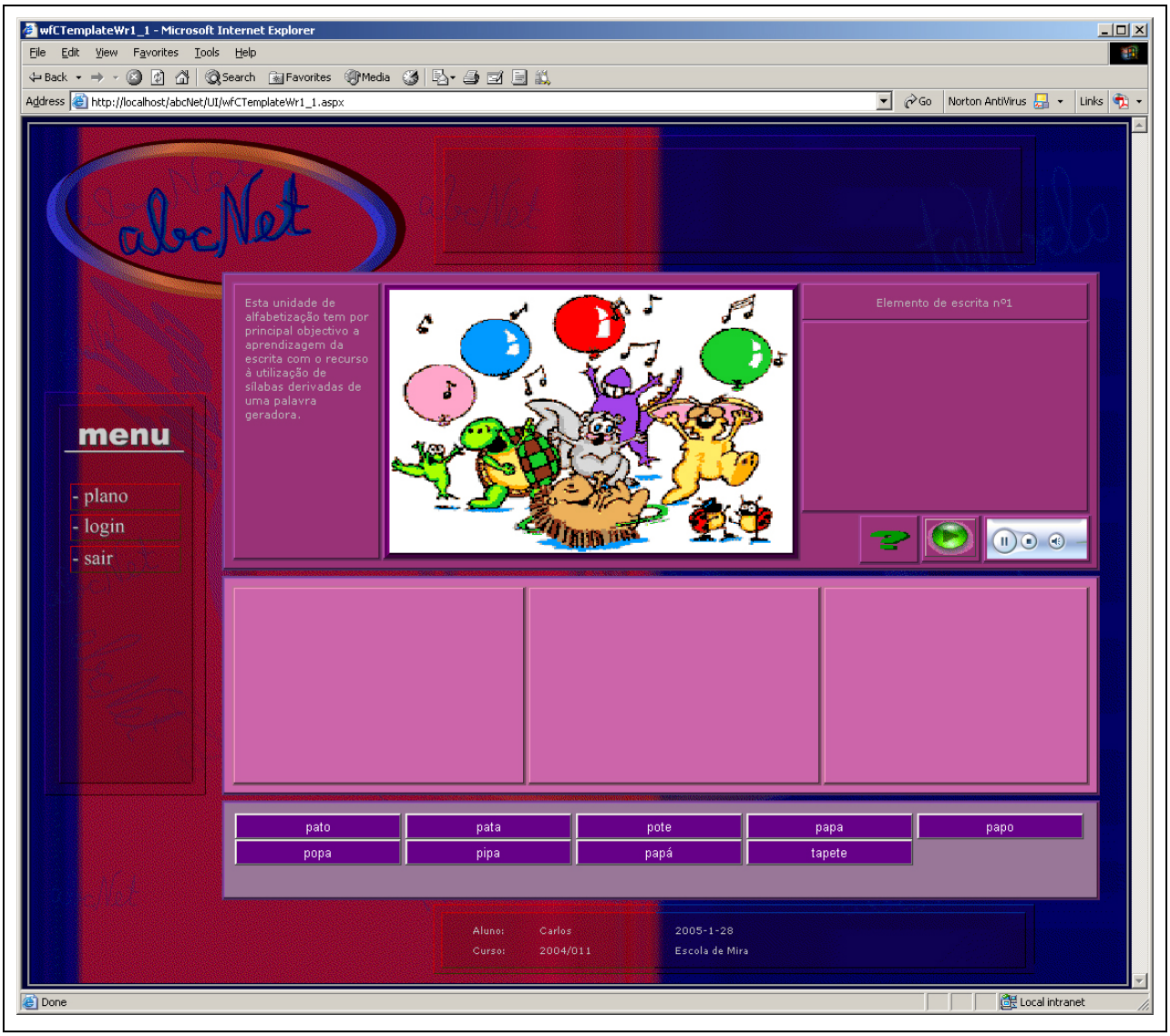

Figura 111 – Interface Uni1Alf que indica conclusão da UniAlf

Interface Uni2Alf

O interface Uni2Alf implementa as Uni2Alf, cujo interface é o apresentado na Figura 112.

A partir deste interface o aluno deve construir cada uma das palavras propostas e que se apresentam uma de cada vez. As palavras construídas vão sendo apresentadas na montra e após a construção da última palavra é apresentado um interface idêntico ao indicado na Figura 111. O aluno deve premir o botão oval verde de modo a evoluir novamente para o interface plano onde deve seleccionar a UniAlf seguinte.

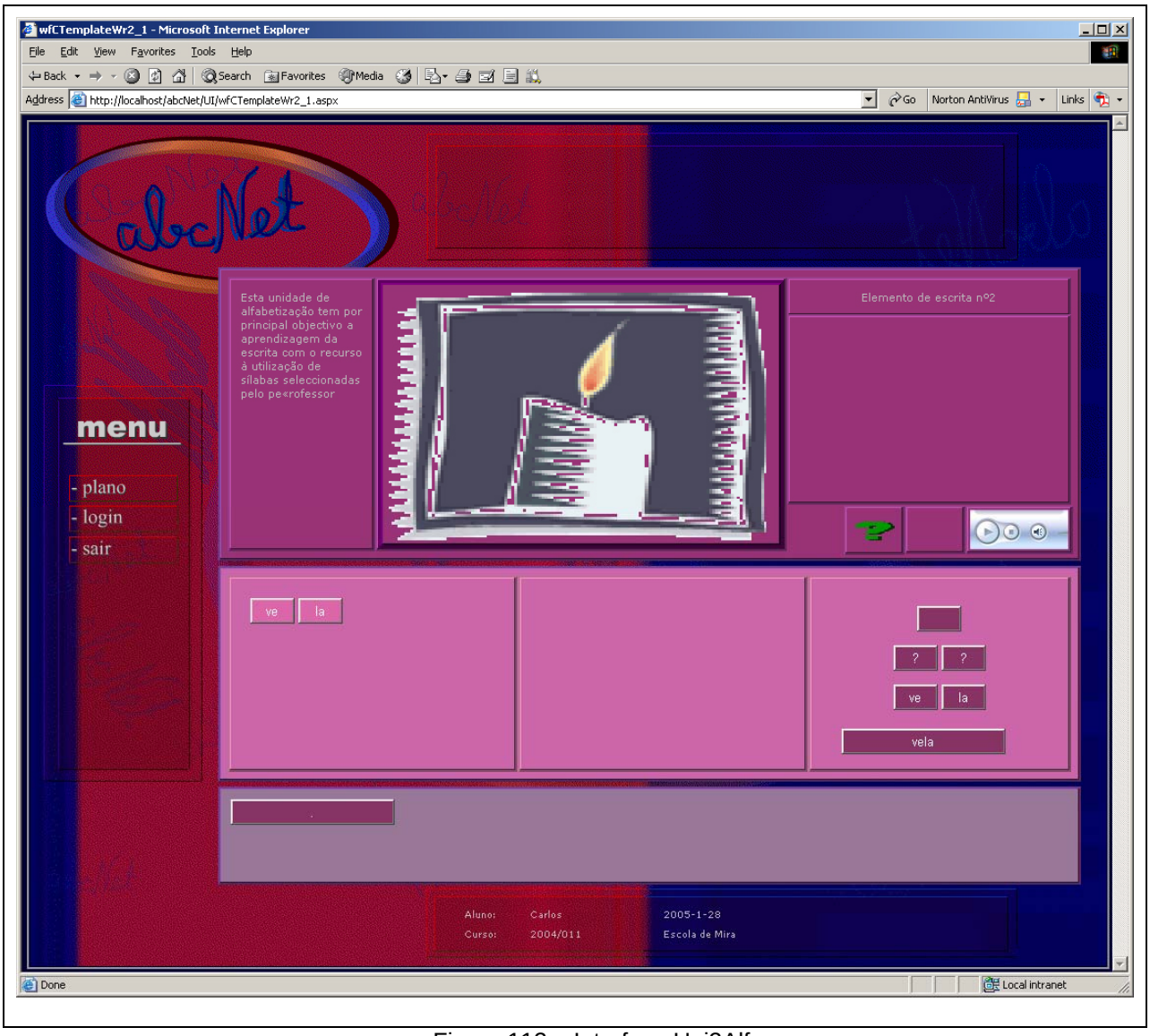

Figura 112 – Interface Uni2Alf

Em [OSCAR-CELDA] apresenta-se uma descrição detalhada do interface Uni2Alf.

9 Conclusões e Ensaio de Campo

Do trabalho realizado, devem ser retiradas conclusões, nomeadamente as afectas a vertentes relacionadas com a utilização real de abcNet, com as tecnologias utilizadas, e com o trabalho desenvolvido.

9.1 Ensaio de Campo

O desenvolvimento de abcNet sustentou-se essencialmente na minha experiência passada havida quer na utilização quer no desenvolvimento de aplicações, mas nenhuma delas análoga a abcNet. A obtenção de um qualquer retorno relativo à utilização de abcNet em ambiente real de utilização revelou-se um momento essencial na reflexão a efectuar sobre o trabalho realizado. Restrições de calendário e de programação do ensaio de campo condicionaram significativamente quer o âmbito quer a ambição dos resultados a esperar. Um ensaio alargado envolveria a selecção de vários públicos alvo que poderiam incluir adultos, crianças, crianças com necessidades educativas especiais, etc. Para cada público, de modo a evitar ou apenas minorar os previsíveis erros de análise dos dados, haveria a necessidade de criar vários grupos paralelos de alunos o que exigiria um esforço de dimensão ciclópico. A acrescentar ter-se-ia ainda que definir planeamentos ajustados às necessidades específicas dos alunos e que deveriam também ter em consideração os objectivos a atingir. Estas e concerteza muitas outras questões teriam de ser equacionadas e posteriormente materializadas. O esforço requerido e o tempo consumido impediria inevitavelmente a conclusão do presente trabalho em tempo oportuno. Apesar de não ter ainda sido referido, não deve ser negligenciável o esforço relativo às questões burocráticas associadas a autorizações para se proceder a um ensaio deste tipo.

9.1.1 Objectivos

O ensaio de campo teve por objectivo analisar a atitude assumida pelos alunos quando confrontados com a utilização de abcNet. Não se pretendeu analisar competências apreendidas nem tão pouco se alguma evolução se verificou ao nível da alfabetização.

9.1.2 Público Alvo

A selecção do público alvo condiciona inevitavelmente quer os ensaios a realizar quer a interpretação dos resultados obtidos. De modo a se poder enfatizar os aspectos relacionados quer com a utilização de uma aplicação multimédia quer com os mecanismos implementados nas UniAlf, foi seleccionado um grupo de crianças com necessidades especiais no domínio da alfabetização. O grupo é constituído por 4 alunos que por dificuldades de aprendizagem na leitura e na escrita, não transitaram para o 2º ano do 1º

ciclo. Dos 4 anos alunos, um encontra-se pelo 3º ano consecutivo no 1º ano. De modo a completar os aspectos relevantes do grupo, deve-se referir que nenhum deles tinha tido qualquer experiência com equipamentos informáticos. Os alunos são aqui identificados por A, B, C e D.

9.1.3 Planeamento

Tendo em consideração o público alvo, foi decidido ter particular cuidado nas primeiras UniAlf a disponibilizar. Estas deveriam incluir apenas palavras já utilizadas nas aulas e fornecidas pelo professor. Adicionalmente, cada palavra foi utilizada de modo a encontrar-se num contexto múltiplo de Uni2Alf. Isto significa que cada palavra deveria estar repetida tantas vezes quantas as necessárias e ordenadas de modo a disponibilizar um número decrescente de ajudas. Simultaneamente, as sílabas disponibilizadas deveriam ser apenas as necessárias para a construção da palavra. Particularizando para o caso da palavra *vela*:

- apenas as sílabas *ve* e *la* pertencem à ficha de descoberta;
- a palavra será sujeita a quatro iterações por parte do aluno. Assim:
    - na 1ª vez está visível quer a palavra quer cada uma das suas sílabas; ao aluno basta prestar atenção às sílabas visíveis e de seguida seleccionar na ficha de descoberta as que são iguais
    - na 2ª vez está visível a palavra e apenas uma das suas sílabas; nesta iteração, apesar de uma das sílabas não se encontrar explicitamente visível o aluno ainda tem a palavra visível
    - na 3ª vez, apenas a palavra está visível; nesta iteração o aluno apenas se poderá socorrer da palavra que se encontra visível
    - na 4ª vez nem a palavra nem nenhuma das sílabas está visível; nesta última iteração o aluno terá de conseguir por si só escrever a palavra correctamente.

Deste modo, os alunos eram conduzidos não só na aprendizagem da escrita da palavra *vela* mas também na lógica existente para utilização de abcNet: ouvir / ler palavra a construir, ouvir / ler sílabas da palavra a construir, selecção de sílabas para a construção da palavra, etc.

Nas duas primeiras Uni2Alf estavam disponíveis as seguintes palavras: *mota*, *vaca*, *mala*, *pato*, *vela*, *mapa* e *lupa*. Nas Uni2Alf seguintes aumentou-se o grau de dificuldade pelo aumento do número de sílabas disponíveis na ficha de descoberta. Passaram a estar disponíveis outras sílabas para além das estritamente necessárias para a construção da palavra proposta. Mesmo assim, cada palavra, à semelhança das Uni2Alf anteriores, apresenta-se também repetida e em crescendo de dificuldade.

Apenas na Uni2Alf nº 4 se introduziram palavras com 3 sílabas tais como: *tulipa*, *cavalo*, *macaco*, *tomate*, *apito* e *viola*. Também aqui foram seguidos esquemas de apresentação múltipla da mesma palavra com evolução crescente do grau de dificuldade. Ao todo foram configurados cinco Uni2Alf.

Relativamente às Uni1Alf, foram configuradas um total de sete. Nestas unidades, o grau de dificuldade encontra-se associado às sílabas da palavra geradora e às palavras propostas para construção. Assim, seleccionaram-se palavras geradoras cujas sílabas potenciem a construção de palavras sucessivamente mais elaboradas. As palavras geradoras seleccionadas foram: *pato*, *mota*, *mala*, *cavalo*, *tulipa,* lago e *bolota*. Dentro de cada palavra geradora, as palavras a construir apresentam graus de dificuldade crescente. Vejamos o caso da palavra geradora *bolota*. As palavras propostas para construção foram: *bolota*, *bolo*, *bola*, *bala*, *bota*, *bata*, *lata*, *lobo*, *batata*, *batota*, *tabela*, *totobola*, *tubo* e *bébé*.

Em termos de calendário foram definidas 2 sessões cada uma com duração não superior a 2 horas. É notório que este tempo é manifestamente insuficiente. Tendo em consideração quer os objectivos enunciados quer a disponibilidade exigida à escola e aos seus alunos (as sessões ocorreram dentro do horário escolar) esta foi a solução possível. No tempo disponível, cada um dos alunos, um de cada vez, teve ao seu dispor um computador pessoal complementado com a minha própria ajuda.

9.1.4 Resultados

Os resultados são analisados em duas vertentes distintas: 1) adaptação a abcNet e 2) sucesso no trabalho executado.

Adaptação a abcNet

A adaptação quer a abcNet quer à utilização de um dispositivo apontador (rato) processou-se de uma forma praticamente imediata, salvaguardando a tarefa associada ao duplo clique para selecção de sílabas. Esta tarefa foi sempre executada mas sempre à custa de alguma persistência e normalmente com o recurso a cliques sucessivos. É minha convicção que com mais algum tempo de utilização, esta tarefa passaria a ser também executada com relativa facilidade. Os alunos não tiveram dificuldade em entender o processo lógico de ouvir a palavra a construir, activar e seleccionar as sílabas para a construção das palavras propostas. Na 2ª sessão, os alunos A e B passaram inclusivamente a ouvir não só a palavra proposta para construção como também cada uma das sílabas que a compõem de modo a mais facilmente poderem seleccionar as sílabas correctas.

Trabalho realizado

O ambiente dentro da sala condicionava significativamente a capacidade de concentração do aluno envolvido na sessão. O aluno D, ao fim de 15 minutos, ficava sempre sem capacidade de concentração

que lhe permitisse continuar a sessão. Outros porém, sendo o aluno A o caso paradigmático, tinha de ser insistentemente avisado para se retirar da sessão de modo a dar oportunidade para que outros colegas seus também pudessem participar.

Relativamente às UniAlf realizadas, estas foram propostas de modo a acompanhar a capacidade de execução de cada um dos alunos. A Tabela 10 apresenta o trabalho realizado por cada aluno em cada uma das sessões.

	Sessão 1	**Sessão 2**
A	Uni2Alf(1, 2, 5, 6 e 7) Uni1Alf(1)	Uni1Alf(2, 3, 5 e 6)
B	Uni2Alf(1, 2, 3 e 4)	Unid1Alf(1, 2, 5 e 6)
C	Unid2Alf(1, 2, 3 e 5)	Unid1Alf(1, 2 e 3)
D	Uni2Alf(1, 2, 3 e 4)	Uni1Alf(1)

Tabela 10 – Trabalho realizado pelos alunos

Na primeira sessão, na execução das Uni2Alf, foi notório que à medida que a repetição de palavras ocorria para o processo de adaptação, os alunos rapidamente manifestavam desagrado. Esta situação poderá ser melhorada pela utilização de agentes que ajustam os conteúdos automaticamente ao desempenho de cada aluno.

Apesar de várias palavras já terem sido tratadas nas aulas, algumas dificuldades foram notadas em todos os alunos, nomeadamente na selecção da última sílaba das palavras que terminavam em *o*. Muito frequentemente era seleccionada a sílaba correspondente mas com terminação em *u*. Após nova explicação para cada caso, os alunos eram capazes de corrigir a situação. No que diz respeito às palavras introduzidas como novidade, não se verificou a ocorrência de dificuldades acrescidas na sua escrita mesmo quando eram constituídas por três ou mais sílabas. O recurso à audição das sílabas existentes na ficha de descoberta revelou-se um mecanismo eficiente para a selecção correcta das sílabas a utilizar para a construção das palavras. Por vezes também se socorriam da audição das sílabas individuais da palavra a construir, mostrando que estes recursos são importantes em situações de hesitação.

O aluno A foi de todos o que melhor se adaptou à utilização de abcNet. Tanto na primeira como na segunda sessão, com o seu ar um pouco sobranceiro, sempre mostrou não só entusiasmo como também bom desempenho na execução das UniAlf.

O aluno B mostrou um avanço significativo da primeira para a segunda sessão. Este aluno foi caracterizado pelo seu professor como sendo o que maiores dificuldades de concentração revelava durante as aulas. Assim sendo, é de realçar que na primeira sessão a sua dispersão era frequente o que obrigou a

várias chamadas de atenção. Por razões que ultrapassam o âmbito do presente trabalho mas que podem ser interpretadas à luz da utilização de TIC, o aluno B na segunda sessão revelou não só uma atenção muito significativa como também um desempenho que apenas foi ultrapassado pelo aluno A. De realçar que o aluno B é o que se encontra pela 3ª vez no 1º ano do 1º ciclo.

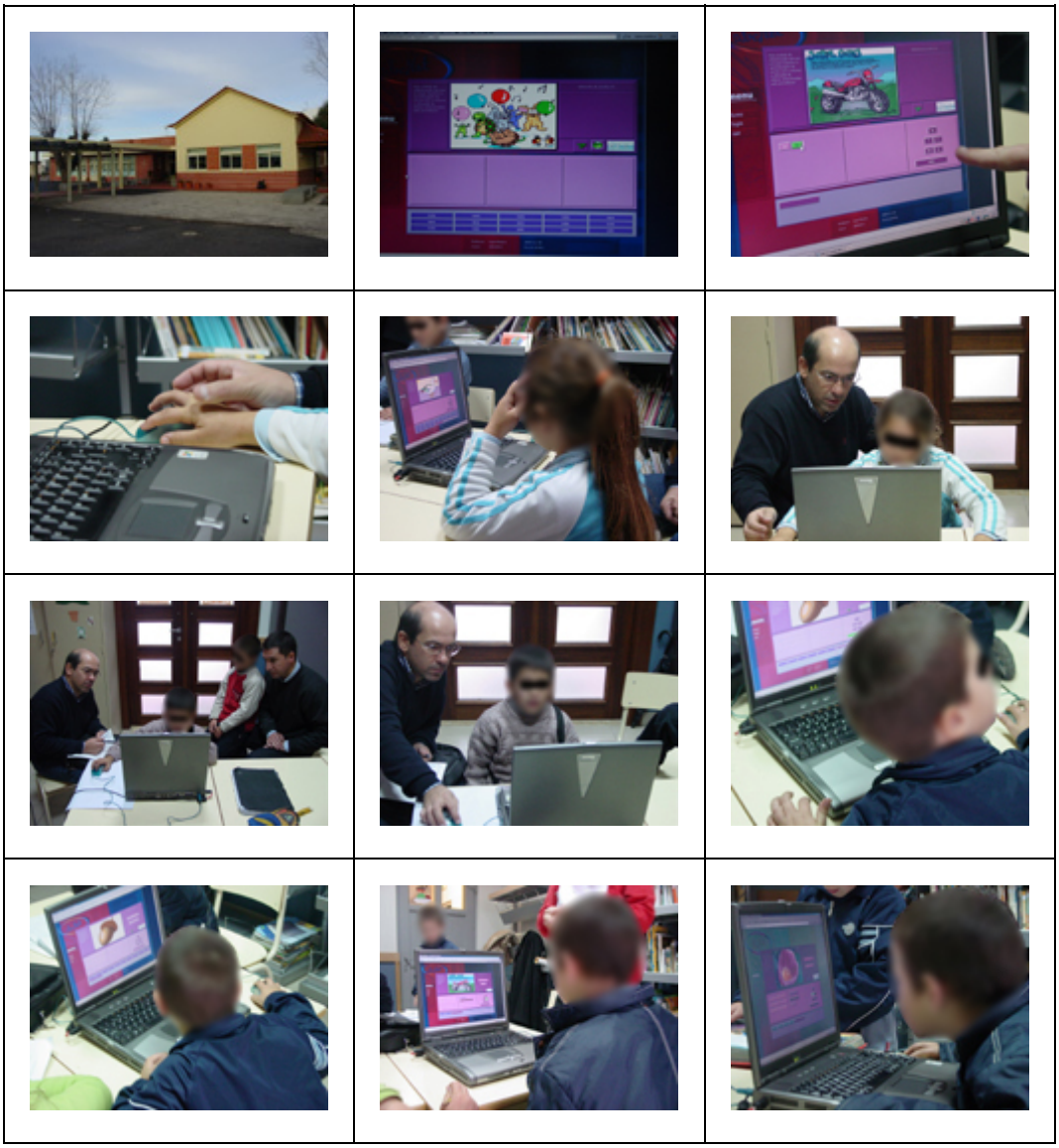

Figura 113 – Momentos do ensaio de campo

O aluno C teve um comportamento linear ao longo das duas sessões. De realçar que no decurso das sessões, este aluno transitava muito facilmente de uma situação de concentração para uma de desconcentração e vice-versa.

Relativamente ao aluno D, o seu desempenho foi razoável na primeira sessão tendo-se dispersado completamente na segunda sessão. Este aluno era o que mais se sentia incomodado com a repetição das palavras o que talvez também possa ter contribuído para a sua atitude.

A Figura 113 apresenta vários dos momentos ocorridos durante os ensaios de campo.

9.1.5 Conclusões do Ensaio

A adaptação às TIC e a abcNet em particular processou-se sem sobressaltos e de uma forma bastante rápida. Foi consensual entre os presentes que a forma positiva como ocorreu a adaptação superou todas as expectativas iniciais. Entretanto alguns aspectos vieram confirmar a necessidade de melhorias em abcNet, nomeadamente na vertente associada a agentes para geração de UniAlf em função do perfil e dos conhecimentos dos alunos. Ao nível do interface, algumas melhorias também podem vir a ser introduzidas. Estas melhorias focalizam-se essencialmente nos mecanismos de interacção com os alunos, nomeadamente o processo implementado para a construção das palavras e em particular no de selecção das sílabas.

Em termos globais, pode-se afirmar que a experiência obtida revelou que o trabalho iniciado se encontra no bom caminho mas que muito ainda há para fazer.

9.2 Conclusões Finais

O trabalho desenvolvido na implementação de abcNet foi muito importante e pode ser analisado em duas vertentes distintas: a tecnológica e a pedagógica.

Na vertente tecnológica, a experiência adquirida quer com as ferramentas utilizadas, quer com a integração das mesmas para persecução dos objectivos, foi muito importante, principalmente para a reflexão que vai ser necessário efectuar, para dar continuidade ao trabalho aqui executado. No caso específico do *Flash*, este foi utilizado como mera ferramenta acessória, pois a estratégia inicialmente adoptada obrigava à angariação de experiência na utilização da plataforma .NET. No futuro, o *Flash* poderá vir a acentuar a sua importância, devido às suas grandes potencialidades nos aspectos relacionados com as componentes gráficas e também à sua previsível fácil integração na plataforma .NET.

Em termos do trabalho desenvolvido merece especial atenção:

- aquilo a que se pode chamar de “boas práticas de programação” tais como a normalização de designações, quer de ficheiros, quer de métodos, quer de classes, etc;
- o projecto PIOOBD que talvez por lapso meu, não consegui encontrar referências para trabalhos idênticos;
- projecto desenvolvido para os testes do PIOOBD e que é baseado em WS

Na vertente pedagógica, a experiência adquirida é mais relevante que a associada à vertente tecnológica. As razões que justificam esta conclusão assentam essencialmente na experiência e conhecimentos existentes quando do início deste trabalho. A experiência e conhecimentos existentes na área da alfabetização podiam-se considerar como nulos. No momento actual, apesar de se não ter investido muito nas vertentes científicas associadas à pedagogia e à linguística, é inegável que o contacto tido com a problemática e as reflexões necessárias à implementação de abcNet criaram uma sensibilidade que então não existia.

Em termos de trabalho desenvolvido merece especial atenção:

- arquitectura utilizada para implementação de uma solução centrada no aluno e baseada na método Paulo Freire;
- extensão da ficha de descoberta a outros grupos de sílabas de modo a potenciar uma maior ambição nos objectivos a atingir pelas UniAlf;
- extensão do método Paulo Freire (Uni2Alf) de modo a permitir uma construção mais simples de nova palavras.

9.2.1 Trabalho Futuro

abcNet, na actual versão, foi uma primeira experiência com a problemática associada à alfabetização. É minha intenção, em próxima oportunidade, dar continuidade ao trabalho aqui iniciado no mestrado. O trabalho a desenvolver terá de ter em atenção vários aspectos, dos quais se podem desde já anunciar:

- definição de um público alvo, previsivelmente um associado a crianças com necessidades educativas especiais;
- criação de modelos que abranjam as diversas filosofias existentes relativas à alfabetização;
- utilização de agentes que propõem exercícios em função do perfil e desempenho dos alunos;
- utilização de agentes para acompanhamento e apoio aos alunos na execução dos exercícios.

IV. Anexos

10 Anexos

10.1 Anexo I – Controlos

Neste Anexo apresentam-se os WUC desenvolvidos para abcNet.

wucSyllable

'wucSyllable' é um WUC que tem por objectivo a implementação de uma estrutura necessária ao desempenho das funções associadas a uma sílaba, de entre as quais se destacam: apresentação do texto, audição da pronúncia, interacções com o rato, ajudas *on-line*, etc.

'wucSyllable' contém apenas um elemento HTML INPUT do tipo *button*, que é o elemento visível e que implementa todo o interface visual nas páginas HTML associado a uma dada sílaba.

```
<%@ Control Language="c#" AutoEventWireup="false"
    Codebehind="wucSyllable.ascx.cs" Inherits="abcNet.wuc.wucSyllable"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
<INPUT id="idBtn" style="WIDTH: 37px; HEIGHT: 26px"
       type="button" value="btn" name="idBtnHTML" runat="server">
```

Figura 114 – Conteúdo HTML associado a wucSyllbale

A Figura 114 apresenta o ficheiro 'ascx' associado a 'wucSyllable', confirmando o que atrás foi dito, quer quanto à estrutura de 'wucSyllable', quer quanto à estrutura dos WUC. A cada sílaba das UniAlf estão atribuídas uma série de tarefas que devem ser executadas do lado do cliente, as quais não se encontram reflectidas na Figura 114. Essas tarefas são definidas em tempo de execução, durante a construção dinâmica da página onde se encontra inserido o 'wucSyllable'.

A Figura 115 apresenta um resultado da página HTML gerada quando vista do lado do cliente. Podemos verificar, que comparativamente à Figura 114, o elemento HTML passou a ter associados vários eventos de rato, cada um com uma ou mais acções a serem executadas. As acções incluídas são função do contexto em que se encontra definida a sílaba. Assim, se a sílaba for uma sílaba de uma palavra geradora, as acções são umas; se a sílaba for uma sílaba família, as acções são outras; se a sílaba for uma sílaba da palavra a construir, as acções são outras, etc.

A mesma situação se aplica para outros controlos, isto é, as acções neles definidas são função do contexto em que forem inseridos. Estas acções correspondem aos *scripts* existentes nos ficheiros já anteriormente apresentados. Além dos eventos, podemos também verificar a existência de um *script* que

se encontra após o elemento HTML, cuja função é a invocação de uma função para catalogação do botão, e cuja execução é incondicional, não dependendo de qualquer tipo de evento.

```
<input name="_ctl3:0000?" id="_ctl3_0000?" type="button"
                          style="WIDTH:45 px;HEIGHT:26px;" value="?"
       onmouseout="btnEventMouseOut('_ctl3_0000?','composeSyllable');stopPlay()"
       onclick="playComposeSyllable('_ctl3_0000?');"
       onmouseover="btnEventMouseOver('_ctl3_0000?','composeSyllable');" +
          "pronounceWithDelay('helps/silaba_seleccionada_para_a_escrita.wav', 3000 );"
       ondblclick="compose('_ctl3_0000?');" />
<script>catalogComposeSyllableControl('0','_ctl3_0000?');</script>
```

Figura 115 – HTML gerado para “wucSyllable”

wucSyllables

‘wucSyllables’ é um WUC que tem por objectivo a criação de um interface construído com o recurso a um ou mais ‘wucSyllbale’. ‘wucSyllables’, ao contrário de ‘wucSyllable’, não contém qualquer elemento HTML, como se pode verificar na Figura 116, sendo todo o seu conteúdo construído em tempo de execução.

```
<%@ Register TagPrefix="ucl" TagName="wucSyllable" Src="wucSyllable.ascx" %>
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="wucSyllables.ascx.cs"
    Inherits="abcNet.wuc.wucSyllables"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
```

Figura 116 – Conteúdo de wucSyllables.ascx

O ‘wucSyllable’ é disposto de uma forma horizontal tal como pode ser verificado na Figura 117. O “wucSyllables” é utilizado na apresentação quer das sílabas família, nas sílabas da palavra geradora, etc.

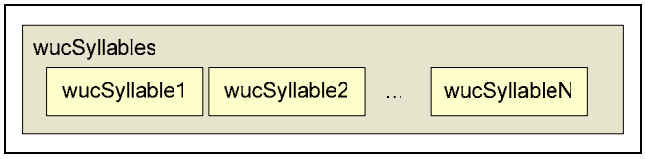

Figura 117 – Apresentação de wucSyllables

wucWord

‘wucWord’ é um *WUC* que tem por objectivo a criação de um interface específico para os elementos pedagógicos associados a palavras. A sua estrutura é em tudo semelhante à do ‘wucSyllable’ pelo que se torna desnecessária uma explicação mais detalhada.

wucSyllOther

O ‘wucSyllOther’ é um WUC que tem por objectivo a criação do interface dos grupos de sílabas (homógrafas, homófonas e adjacentes) associados a uma dada sílaba.

```
<%@ Register TagPrefix="ucl" TagName="wucSyllable" Src="wucSyllable.ascx" %>
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="wucSyllables.ascx.cs"
    Inherits="abcNet.wuc.wucSyllables"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
```

Figura 118 – Conteúdo de wucSyllOther.ascx

Tal como nos casos anteriores, a construção do interface é toda elaborada em tempo de execução, tal como se pode deduzir da análise da Figura 118. Em termos de apresentação, as sílabas de cada grupo são dispostas de forma horizontal e, entre si, os grupos são dispostos de uma forma vertical tal como se apresenta na Figura 119.

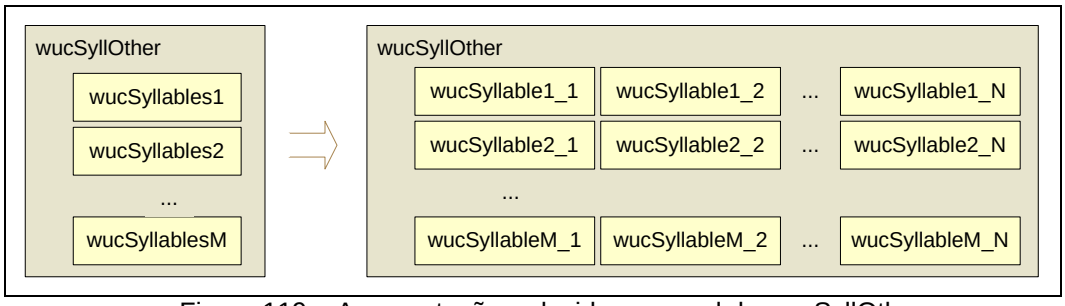

Figura 119 – Apresentação reduzida e normal de wucSyllOther

Da Figura 119 podemos também verificar que a construção do interface é efectuada com recurso ao WUC ‘wucSyllables’, que por sua vez recorre a ‘wucSyllable’

No servidor são criados tantos ‘wucSyllOther’ quantos os necessários para apresentação dos grupos de sílabas associadas a cada sílaba base de construção de palavras. Assim, no caso da Uni1Alf, se exitirem 15 sílabas família, são necessários 15 ‘wucSyllOther’. O controlo do ‘wucSyllOther’ a apresentar em cada instante é efectuado ao nível de *scripts* que correm do lado do cliente e que se encontram definidos no ficheiro *scpOtherSyll*.

A Figura 120 apresenta um interface implementado por ‘wucSyllOther’ para a sílaba ‘bo’ em que se pode verificar a apresentação de três grupos distintos de sílabas, cada um com várias sílabas. Neste caso, na primeira linha encontra-se o grupo de sílabas homógrafas, na segunda linha o grupo de sílabas adjacentes e na terceira linha o grupo de sílabas homófonas.

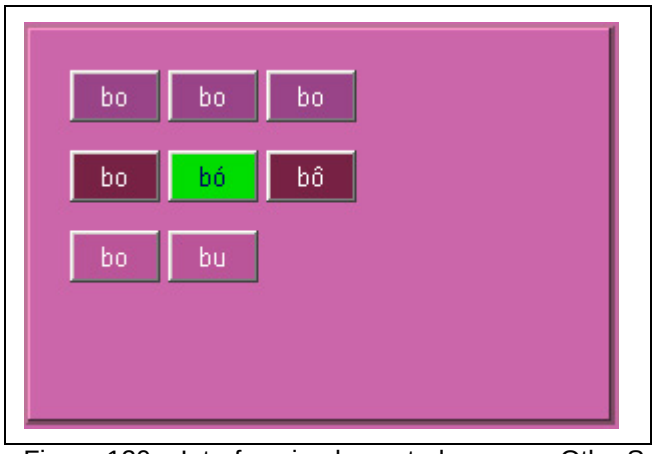

Figura 120 – Interface implementado por wucOtherSyll

Por cada sílaba activada são apresentados, com recurso a scripts do lado do cliente, os correspondentes grupos de sílabas homógrafas, homófonas e adjacentes.

wucGuessWord

O WUC 'wucGuessWord' implementa quer o interface quer a estrutura necessária à construção de palavras nas UniAlf. A Figura 121 apresenta o interface implementado por 'wucGuessWord', após a correcta construção da palavra lago.

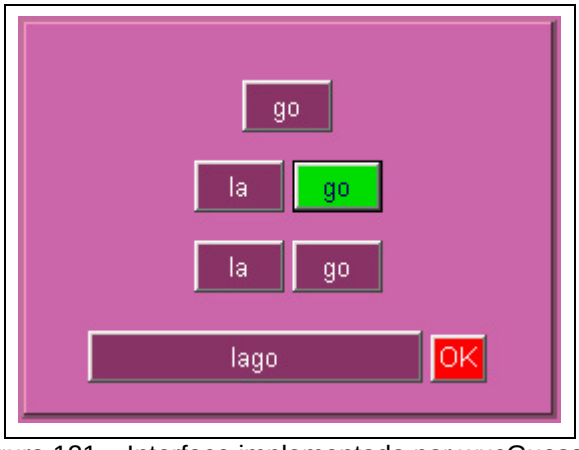

Figura 121 – Interface implementado por wucGuessWord

Tal como nos controlos anteriores, 'wucGuessWord' é todo construído em tempo de execução como se pode deduzir da Figura 122.

```
<%@ Control Language="c#" AutoEventWireup="false"
    Codebehind="wucGuessWord.ascx.cs" Inherits="abcNet.wuc.wucGuessWord"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
```

Figura 122 – Conteúdo de wucGuessWord.ascx

A estrutura de controlos a construir em tempo de execução por ‘wucGuessWord’ encontra-se apresentada na Figura 123.

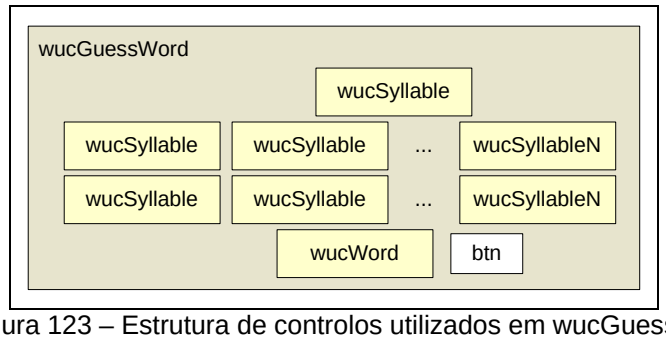

Figura 123 – Estrutura de controlos utilizados em wucGuessWord

wucFooter

O WUC ‘wucFooter’ implementa o interface correspondente ao rodapé a apresentar nas páginas HTML. Ao contrário dos casos anteriores, ‘wucFooter’ contém algum conteúdo HTML correspondente à organização da informação a apresentar.

A Figura 124 apresenta uma concretização do interface implementado por ‘wucFooter’, onde se pode verificar o perfil e nome do utilizador, o código e descrição do curso, e também a data. A menos da estrutura, todo o conteúdo de “wucFooter” também é definido em tempo de execução.

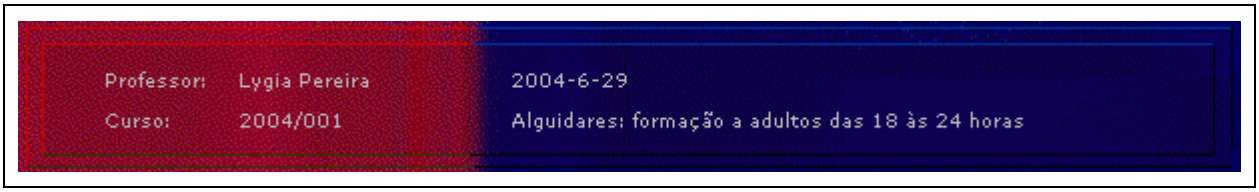

Figura 124 – Interface implementado por wucFooter

wucDialogYesCancel

O *WUC* ‘wucCancelYesCancel’ implementa um interface do tipo caixa de diálogo com o objectivo de proporcionar as condições necessárias à confirmação ou cancelamento de operações. A acção de confirmação ou de cancelamento é implementada com o recurso a dois botões, um para cada opção, cuja representação gráfica é apresentada na Figura 125.

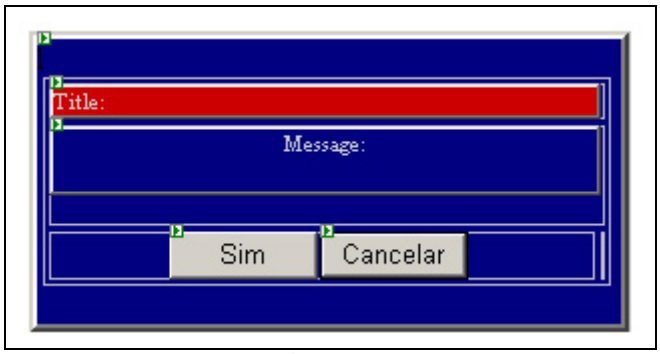

Figura 125 – Visualização gráfica do HTML de wucDialogYesCancel

A interacção deste WUC com a sua envolvente processa-se em dois sentidos: para o interior e para o exterior do controlo. A interacção para o interior envolve os mecanismos de tornar o controlo visível e é conseguido por activação do método dedicado para o efeito. Esse método não só torna o controlo visível como também aceita alguns argumentos que determinam parte do interface apresentado. Assim, quer o título a apresentar no controlo, quer a mensagem, quer os botões, são fruto de opções tomadas fora do controlo, que são passadas para o mesmo, via os argumentos do método.

A interacção para o exterior tem a ver com a informação a passar para a envolvente e que é referente à opção tomada pelo utilizador quando da selecção de um dos botões. De modo a atingir os objectivos enunciados, foi definida a seguinte estratégia. O pai do controlo deve implementar um interface através do qual 'wucDialogYesCancel' o informa da opção tomada por intermédio de um argumento.

A Figura 126 apresenta uma implementação do interface apresentado por 'wucDialogYesCancel' em que podemos descriminar três secções distintas, que de cima para baixo são:

- tipificação da mensagem;
- mensagem;
- botões.

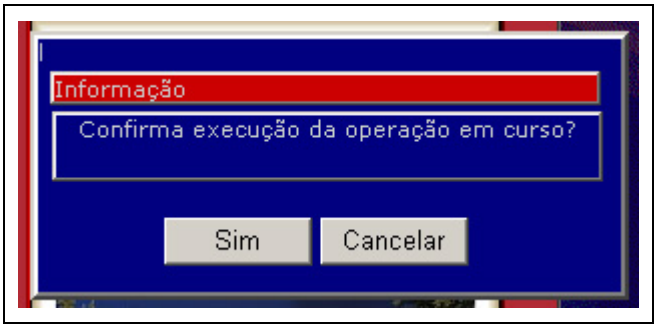

Figura 126 – Interface implementado por 'wudDialogYesCancel'

wucWr1

O WUC ‘wucWr1’ implementa o interface correspondente ao ‘Mod1Alf’. É a partir deste interface que se obtém a configuração das Uni1Alf. O processo de configuração está sustentado num mecanismo de edição que é gerido através de um painel de botões que permitem efectuar as seguintes operações:

- criar uma nova unidade de alfabetização: botão ‘Novo’;
- editar uma unidade de alfabetização já existente: botão ‘Alterar’;
- apagar uma unidade de alfabetização já existente: botão ‘Apagar’.

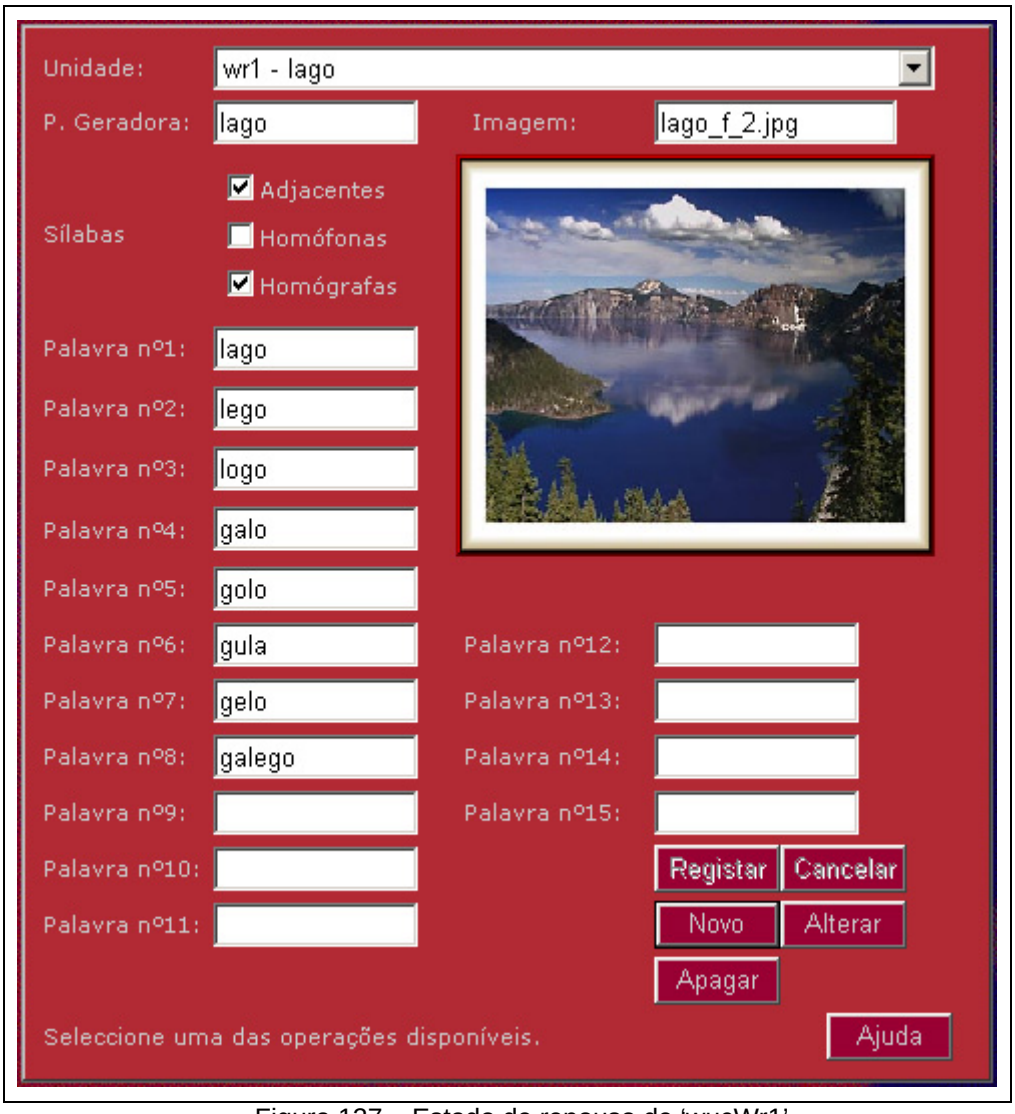

Figura 127 – Estado de repouso de ‘wucWr1’.

A confirmação ou cancelamento de cada uma destas operações é executada através do botão ‘Registar’ ou ‘Cancelar’, respectivamente. Mesmo após a confirmação ou cancelamento, o utilizador é convidado a reconfirmar ou cancelar a operação em curso, por intermédio de uma caixa de diálogo que lhe é apresentada e que é implementada pelo ‘wucCancelYesCancel’.

A Figura 127 apresenta o ‘wucWr1’ no estado de repouso com visualização da configuração já atribuída a uma unidade de alfabetização.

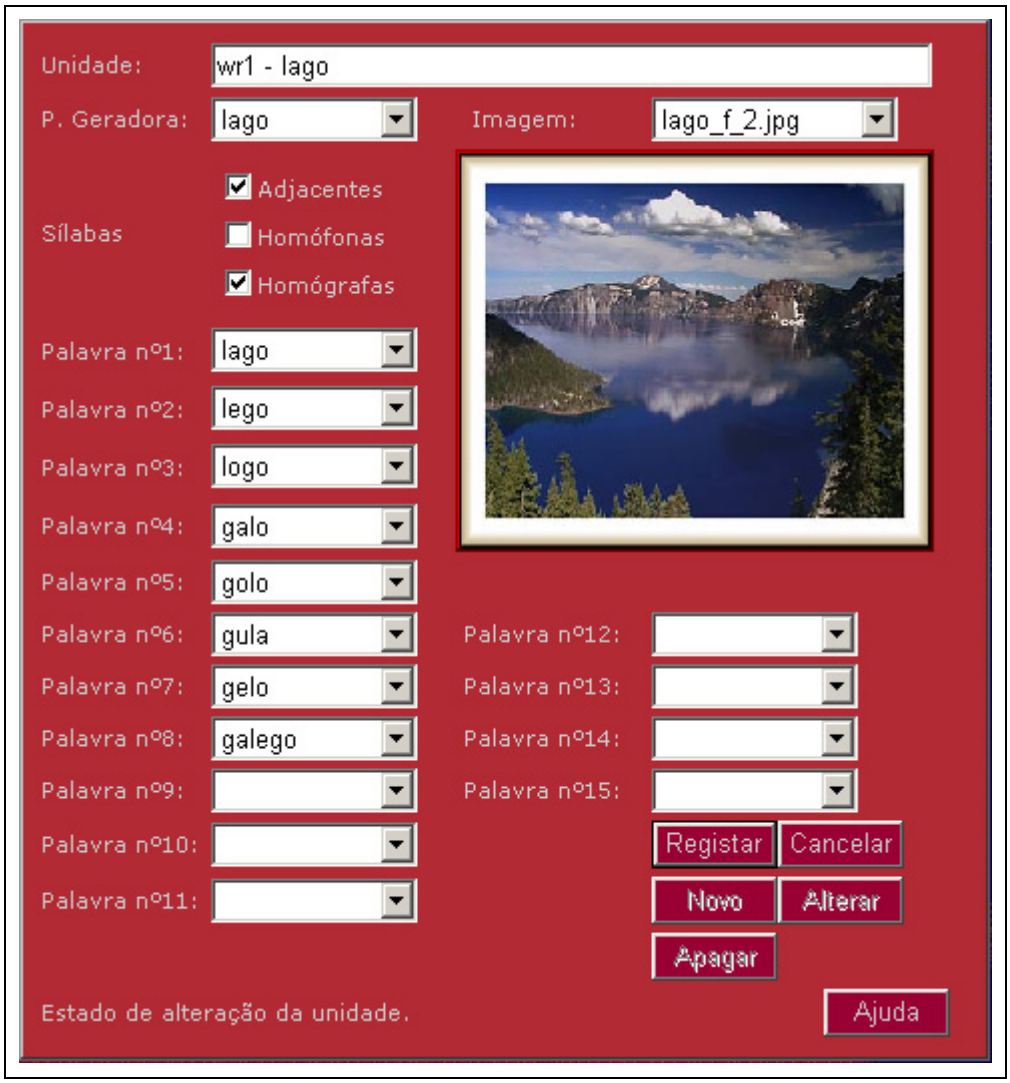

Figura 128 – Estado de edição de ‘wucWr1’

A Figura 128 apresenta o ‘wucWr1’ no estado de edição que pode ter sido estabelecido quer pela criação de uma nova UniAlf quer pela edição de uma já existente. Desta figura pode-se verificar o mecanismo implementado para disponibilização ao utilizador, quer das palavras geradoras, quer das imagens quer das palavras que devem ser propostas aos alunos para construção.

Caso a opção seja apagar uma UniAlf já existente, ‘wucWr1’ apresenta de imediato a caixa de diálogo para confirmação ou cancelamento da referida operação.

O botão ‘Ajuda’ apresenta uma página HTML onde se descrevem textualmente os mecanismos disponibilizados para a criação, alteração e remoção de UniAlf.

wucWr2

O *WUC* 'wucWr2' implementa o interface correspondente ao modelo 'Mod2Alf'. É a partir deste interface que se obtém a configuração das Uni2Alf. O processo de configuração está sustentado num mecanismo de edição que é gerido através de um painel de botões que permitem efectuar as seguintes operações:

- criar uma nova unidade de alfabetização: botão 'Novo';
- editar uma unidade de alfabetização já existente: botão 'Alterar';
- apagar uma unidade de alfabetização já existente: botão 'Apagar'.

A confirmação ou cancelamento de cada uma destas operações é executada através do botão 'Registar' ou 'Cancelar', respectivamente. Mesmo após a confirmação ou cancelamento, o utilizador é convidado a reconfirmar ou cancelar a operação em curso, por intermédio de uma caixa de diálogo que lhe é apresentada e que é implementada pelo 'wucCancelYesCancel'.

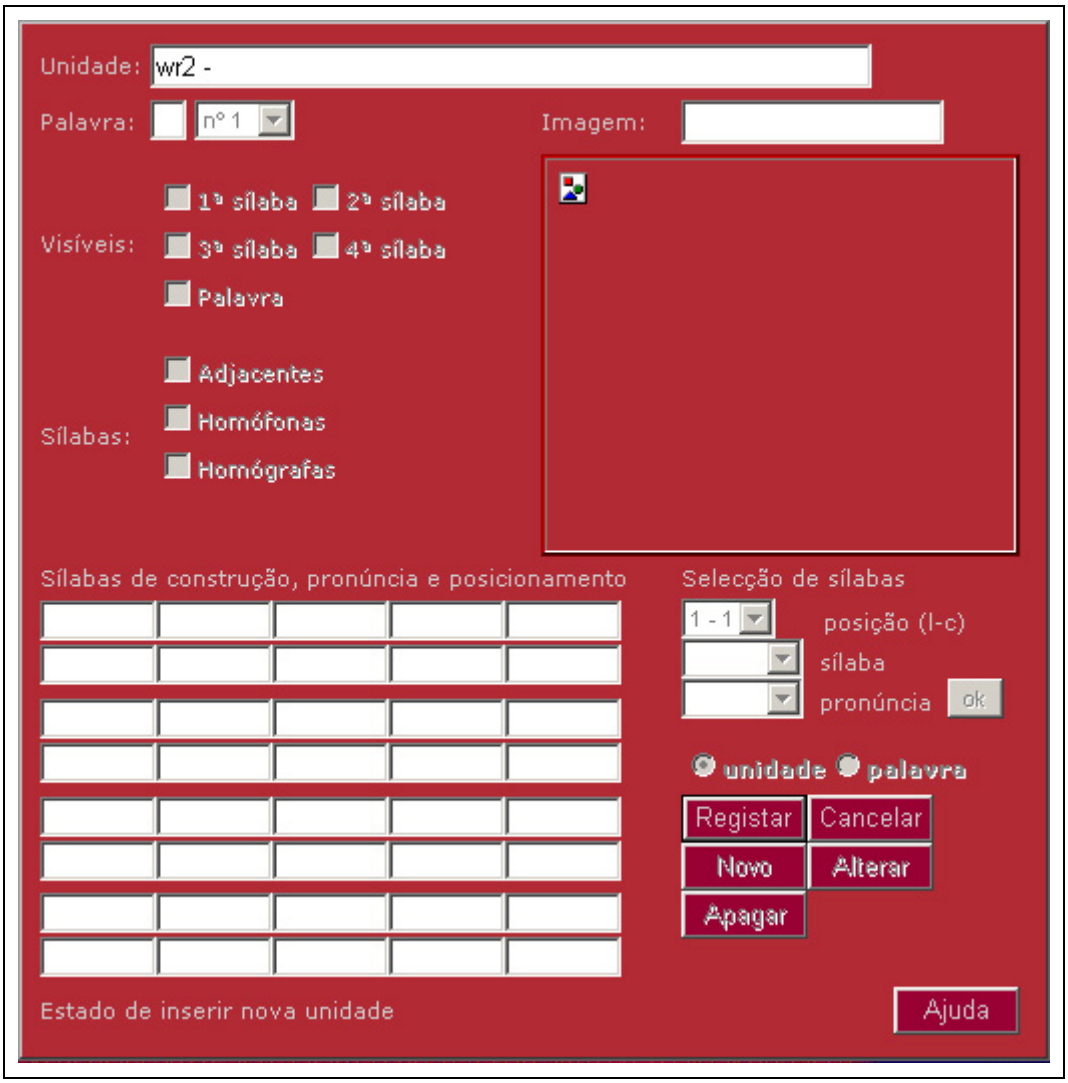

Figura 129 – Criação de uma nova unidade de alfabetização.

Ao contrário de ‘wucWr1’, em ‘wucWr2’ é necessário definir elementos de configuração por cada uma das palavras propostas para construção. Assim sendo, as opções de criação, edição e de remoção tanto podem incidir sobre a UniAlf como sobre uma dada palavra da UniAlf. O mecanismo utilizado para selecção da opção está materializado pelo recurso a botões de opção e que se encontram localizados por cima do painel de botões. Assim, a função a desempenhar por cada um dos botões do painel encontra-se dependente da opção seleccionada nos botões de opção.

A Figura 129 apresenta a situação correspondente à criação de uma nova Uni2Alf, em que apenas se define a designação a atribuir à unidade. Nos botões de opção, a opção seleccionada é a correspondente à unidade.

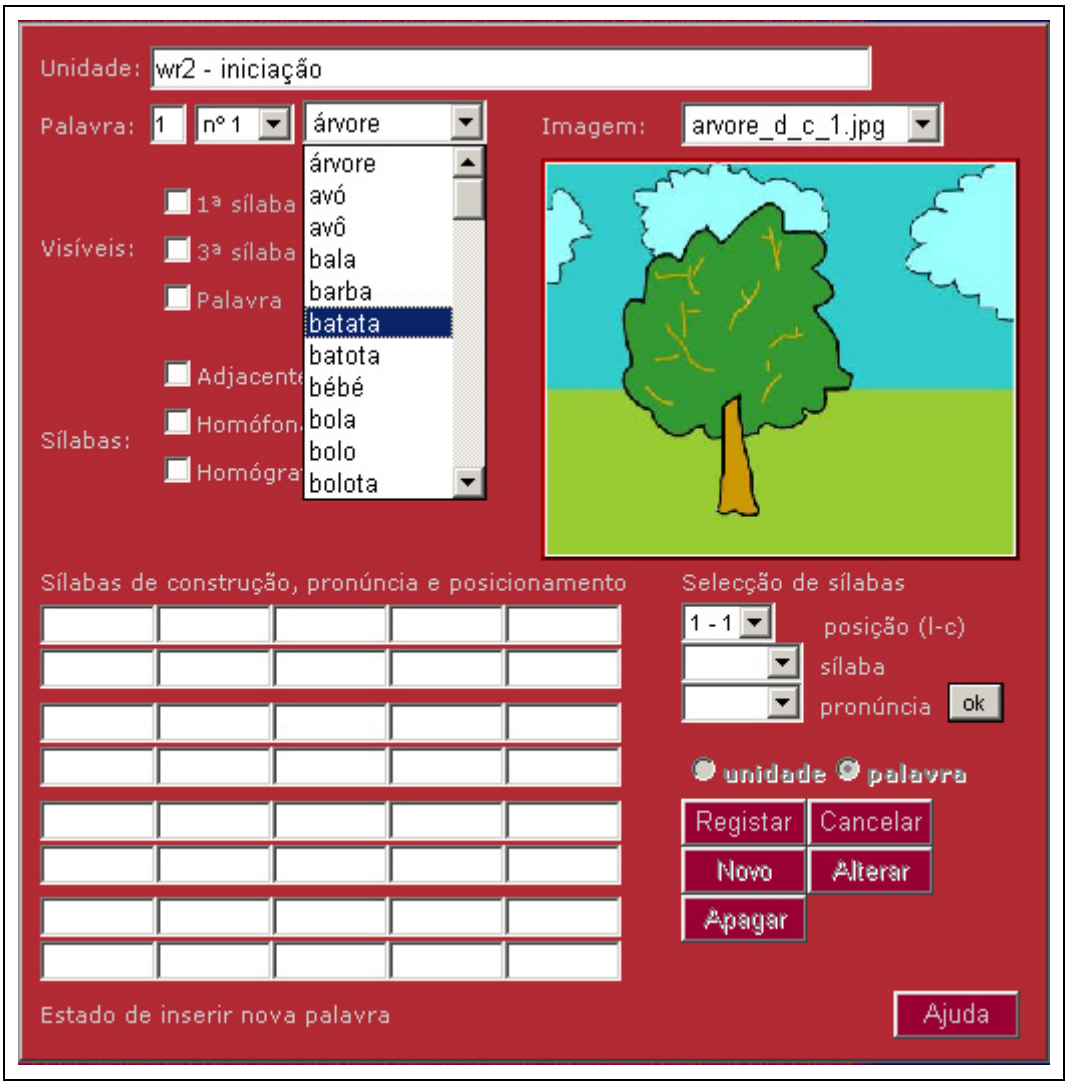

Figura 130 – Criação de uma nova palavra para a unidade de alfabetização

A Figura 130 apresenta a situação correspondente à criação de uma nova palavra e definição da restante informação a ela associada, nomeadamente a imagem, sílabas visíveis, sílabas e respectivas pronúncias a utilizar, entre outra informação.

Figura 131 – Edição de uma palavra da unidade de alfabetização seleccionada

A Figura 131 apresenta a situação correspondente à edição de uma palavra já configurada para a Uni2Alf seleccionada. Toda a informação a ela associada pode ser alterada excepto a própria palavra. Inclusivamente a ordem pela qual a palavra se apresenta para construção na Uni2Alf, também pode ser alterada, como se pode depreender da caixa de selecção que se encontra situada do lado esquerdo da palavra.

A Figura 132 apresenta o 'wucWr2' no estado de repouso com visualização da configuração já atribuída a uma Uni2Alf.

Não vamos apresentar todas as situações possíveis. Das já apresentadas, pode-se deduzir os interfaces implementados para as restantes situações.

O botão 'Ajuda' apresenta uma página HTML onde se descreve textualmente os mecanismos disponibilizados para a criação, alteração e remoção de UniAlf.

Figura 132 – Estado de repouso de 'wucWr2'

wucOrdering

O WUC 'wucOrdering' implementa o interface que possibilita a ordenação das diversas UniAlf configuradas, para uma dada acção de formação. O processo de ordenação está sustentado num mecanismo de edição que é gerido através de um botão e de caixas de selecção que permitem reordenar as diversas unidades de alfabetização.

A confirmação da operação é executada através do botão 'Registar'. Mesmo após a confirmação, o utilizador é convidado a reconfirmar ou cancelar a operação de reordenamento por intermédio de uma caixa de diálogo que lhe é apresentada, e que é implementada pelo 'wucCancelYesCancel'. Se por lapso o utilizador tiver alguma UniAlf repetida, a caixa de diálogo apenas o informa da ocorrência.

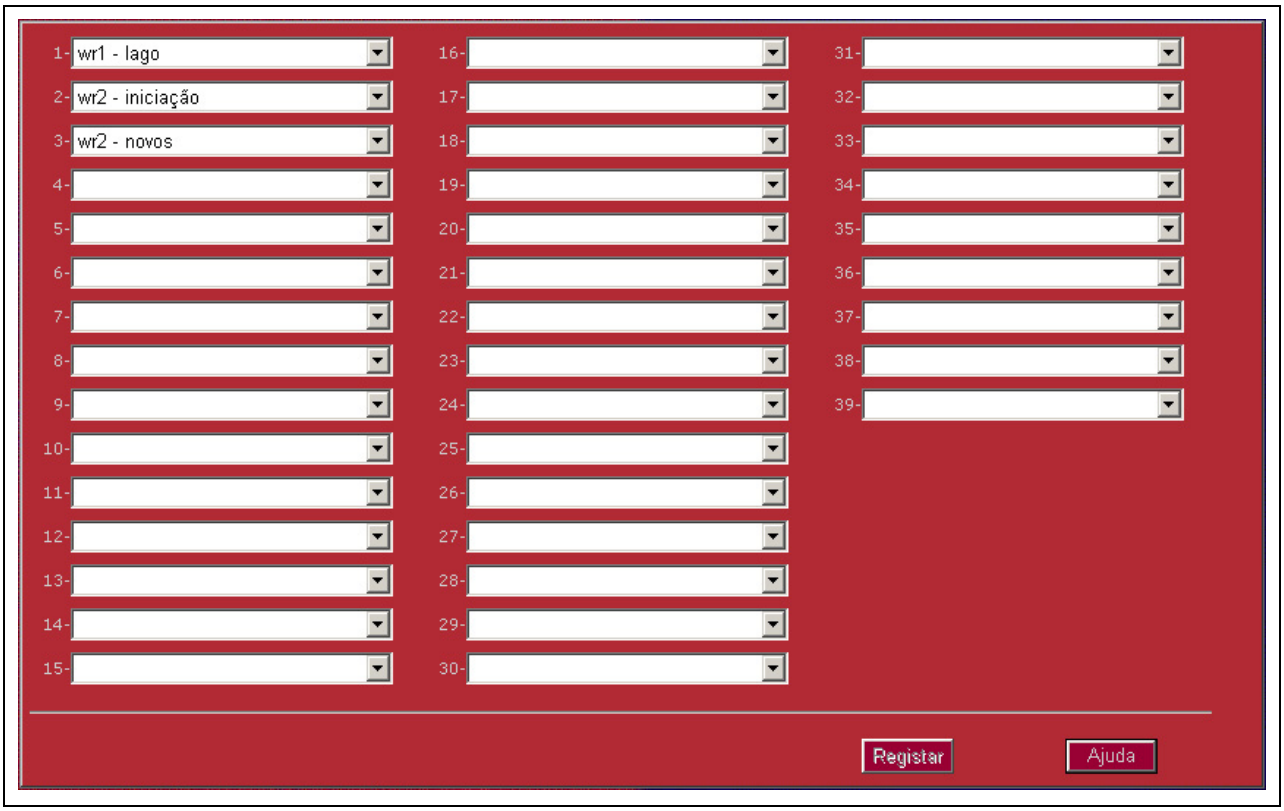

Figura 133 – Estado de repouso de ‘wucOrdering’

A Figura 133 apresenta o estado de repouso de ‘wucOrdering’ em que podemos verificar a ordenação de 3 UniAlf. Relembrar que as UniAlf pertencem todas a uma só acção de formação. O botão ‘Registar’ apenas fica activo quando existir pelo menos uma alteração na ordenação que ainda não foi registada.

O botão ‘Ajuda’ apresenta uma página HTML onde se descreve textualmente os mecanismos disponibilizados para a ordenação das unidades de alfabetização.

V. Acrónimos

BCL	Base Class Library
BD	Base de Dados
CIL	Common Intermediate Language
CLI	Common Language Infraestructure
CLR	Common Language Runtime
CLS	Common Language Specification
ECMA	European Computer Manufacturers Association
EleAlf	Elemento de alfabetização
EntAlf	Entidade de alfabetização
Ent1Alf	Entidade 1 de alfabetização
Ent2Alf	Entidade 2 de alfabetização
IDE	Interface Development Environment
GNU	GNU´s Not Unix
HTTP	Hypertex Transfer Protocol
JVM	Java Virtual Machine
MAA	Mecanismo activo de ajuda
MódAdm	Módulo de Administração
Mod1Alf	Modelo 1 de alfabetização
Mod2Alf	Modelo 2 de alfabetização
ModAlf	Modelo de alfabetização
MódBD	Módulo da Base de Dados
MódFor	Módulo da Formação
MS	Microsoft
OOL	Object Oriented Language
PalEnt	Entidade de palavra
PalOrt	Componente correspondente à ortografia de uma palavra
PalPro	Componente correspondente à pronúncia de uma palavra
PIOOBD	Projecto do Interface Orientado a Objectos para a Base de Dados
PTIBD	Projecto de Teste à Interface da Base de Dados
RAD	Rapid Application Development
SDK	System Development Kit
SGBD	Sistema de Gestão de Base de dados

SGBD-R	Sistema de Gestão de Base de Dados Relacional
SGBD-RO	Sistema de Gestão de Base de Dados Relacional de Objectos
SGBD-OO	Sistema de Gestão de Base de Dados Orientada a Objectos
SílEnt	Entidade de sílaba
SílOrt	Componente correspondente à ortografia de uma sílaba.
SílPro	Componente corresponte à pronúncia de uma sílaba
SOAP	Simple Object Application Protocol
SP	Stored Procedure
TIC	Tecnologias de Informação e de Comunicação
Uni1Alf	Unidade 1 de alfabetização
Uni2Alf	Unidade 2 de alfabetização
UniAlf	Unidade de alfabetização
W3C	World Wide Web Consortium
WCC	Web Custom Control
WF	Web Forms
WUC	Web User Control
WS	Web Service
WSDL	Web Service Definition Language
XML	Extensible Markup Language

VI. Bibliografia e Referências

Adrian Turtschi et al, “C# .NET Web Developer´s Guide”. Syngress publishing 2002. ISBN 1-928994-50-4.

Celso Cunha e Lindley Cintra, “Nova Gramática do Português Contemporâneo”. Edições João Sá da Costa. Lda, Lisboa, 1991, 8ª Edição, ISBN: 972-9230-28-5.

David Jorgensen, “Developiong .NET Web Services with XML”. Syngress publishing 2002. ISBN 1-928994-81-4.

Mesbah Ahmed et al, “ASP.NET Web Developer´s Guide”. Syngress publishing 2002. ISBN 1-928994-51-2.

[APEN] Andrew Pen. A Comparison of two major dynamic web platforms. World Wide Web, http://www.shawnolson.net/a/302. Visitado em Novembro de 2004.

[COBOL] Fujitsu – NetCobol. World Wide Web, http://www.adtools.com/products/windows/netcobol.html. Visitado em Abril de 2004.

[CONN-DB] Thomas Connoly. Database Systems – A Pratical Approach to Design, Implementaion and Management. Second Edition, 1999. Addison Wesley Logman Ltd, Inglaterra. ISBN: 0-201-34287-1

[CONST-LP] Isabel Trindade e Cristina Marcelino. Língua Portuguesa – 1º ano. Editora Constância, Carnaxide, Portugal, 1999. ISBN: 972-761-036-6.

[CP-HELL] SSW – DLL Hell. World Wide Web, http://www.ssw.com.au/SSW/Database/DLLHell.aspx. Visitado em Maio de 2004.

[CRB] Carlos Rodrigues Brandão. O que é o Método Paulo Freire. Editora Brasiliense, São Paulo, Basil, 1981. 2ª Edição.

[CRE-MSA] Centro de Referência Educacional – O Método Sintético e o Analítico. World Wide Web, http://www.centrorefeducacional.pro.br/metodo.html. Visitado em Abril de 2004.

[DELPHI] Borland – Microsoft .NET Framework. World Wide Web, http://www.borland.com/dotnet/. Visitado em Novembro de 2003.

[DOBBS] Marc Eaddy. C# Versus Java.Dr. Dobb´s Journal. Fevereiro de 2001. World Wide Web, http://www1.cs.columbia.edu/~eaddy/publications/csharpvsjava-eaddy-ddj-feb01.pdf. Visitado em Junho de 2004.

[ECMA] ECMA – Welcome. World Wide Web, http://www.ecma-international.org/. Visitado em Novembro de 2003.

[ECMA-334] Standard ECMA-334 C# Language Specification. World Wide Web, http://www.ecma-international.org/publications/standards/Ecma-334.htm. Visitado em Novembro de 2003.

[ECMA-335] ECMA International – Standard ECMA-335. World Wide Web, http://www.ecma-international.org/publications/standards/Ecma-335.htm. Visitado em Novembro de 2003.

[ECMA-INDEX] ECMA – Index to ECMA Standards. World Wide Web, http://www.ecma-international.org/publications/standards/Stnindex.htm. Visitado em Setembro de 2004.

[EDU] A filosofia de Paulo Freire e as práticas cognitivas no jornalismo. World Wide Web, http://www.infoamerica.org/teoria/articulos/freire1.htm. Visitado em Junho cde 2004.

[GNU] GNU Operating System – Free Software Foundation. World Wide Web, http://www.gnu.org. Visitado em Novembro de 2003.

[HENRY] Henry Suzzallo. Reading, Teaching Beginners. World Wide Web, http://donpotter.net/PDF/Suzzallo%20Beginning%20Reading.pdf. Visitado em Janeiro de 2004.

[IETF-2821] IETF – Simple Mail Transfer Protocol. World Wide Web, http://www.ietf.org/rfc/rfc2821.txt?number=2821. Visitado em Fevereiro de 2004.

[INFED] The encyclopeadia of informal education- Paulo Freire. http://www.infed.org/thinkers/et-freir.htm. Visitado em 20 de Março de 2004.

[IPF-BIBL] Instituto Paulo Freire – Paulo Freire. World Wide Web http://www.paulofreire.org/pf_livros.htm. Visitado em Janeiro de 2004.

[IPF-BIOG] Instituto Paulo Freire – Pequena biografia. World Wide Web, http://www.paulofreire.org/pfreire.htm. Visitado em Janeior de 2004.

[IPF-HOME] Instituto Paulo Freire. World Wide Web, http://www.paulofreire.org/. Visitado em Janeiro de 2004.

[KAMRAN] Kamran Shakil. Microsoft .NET Security for Newcomers. World Wide Web, http://searchvb.techtarget.com/vsnetTip/1,293823,sid8_gci881935_tax292991,00.html. Visitado em Julho de 2004.

[LCO] Luís Caldas de Oliveira. Alfabeto Fonético para o Dialecto Padrão do Português. World Wide Web, http://www.l2f.inesc-id.pt/~lco/ptsam/ptsam.pdf. Visitado em Dezembro de 2003.

[MM-FLASH] Macromedia Flash MX 2004. World Wide Web, http://www.macromedia.com/software/flash/. Visitado em Dezembro de 2003.

[MOBRAL] Movimento Brasileiro de Alfabetização – MOBRAL. World Wide Web, http://www.pedagogiaemfoco.pro.br/heb10a.htm. Visitado em Junho de 2004.

[MONO] Mono – What is Mono? World Wide Web, http://www.mono-project.com/about/index.html. Visitado em Novembro de 2003.

[MS-ADO] Microsoft – Overview of ADO.NET. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconoverviewofadonet.asp. Visitado em Fevereiro de 2004.

[MS-ADOCN] Microsoft – SqlConnection Class. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfSystemDataSqlClientSqlConnectionClassTopic.asp. Visitado em Janeiro de 2004.

[MS-ADODP] Microsoft - .NET Framework Data Providers. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconADONETProviders.asp. Visitado em Fevereiro de 2004.

[MS-ADODR] Microsoft – DataRow Class. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemdatadatarowclasstopic.asp. Visitado em Novembro de 2003.

[MS-ADODS] Microsoft – ADO.NET DataSet. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpcontheadonetdataset.asp. Visitado em Fevereiro de 2004.

[MS- Microsoft – SqlParameter Class. World Wide Web,

ADOPM] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemdatasqlclientsqlparameterclasstopic.asp. Visitado em Novembro de 2003.

[MS-APPOB] Microsoft – Application Object. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/iissdk/iis/ref_vbom_apo.asp. Visitado em Dezembro de 2003.

[MS-APPST] Microsoft – Application State. World Wide Web. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconapplicationstate.asp. Visitado em Maio de 2004.

[MS-ASPCNF] Microsft – ASP.NET Configuration. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconASPNETConfiguration.asp. Visitado em Janeiro de 2004.

[MS-ASPSC] Microsoft – ASP.NET Web Application Security. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconaspnetwebapplicationsecurity.asp. Visitado em Junho de 2004.

[MS-ASPTR] Microsoft – ASP.NET Trace. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpcontracefunctionality.asp. Visitado em Maio de 2004.

[MS-BCL] .NET Framework Classe Library. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/cpref_start.asp. Visitado em Março de 2004.

[MS-BIZ] Microsoft BizTalk Server. World Wide Web, http://www.microsoft.com/biztalk/. Visitado em Setembro de 2004.

[MS-CIL] The Microsoft Intermediate Language. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnhcvs03/html/vs03k1.asp. Visitado em Junho de 2004.

[MS-CLR] Microsoft – The Common Language Runtime. World Wide Web, http://msdn.microsoft.com/netframework/programming/clr/default.aspx. Visitado em Fevereiro de 2004.

[MS-CLS] What is the Common Language Specification?. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconwhatiscommonlanguagespecification.asp. Visitado em JUlho de 2004.

[MS-C#:C++] Microsoft – Comparison Between C++ and C#. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cscon/html/vclrfcomparisonbetweenccsharp.asp. Visitado em Julho de 2004.

[MS-C#IDX] Microsoft – Indexers. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vclrfindexedpropertiespg.asp. Visitado em Novembro de 2003.

[MS-dotNET] Microsoft .NET Framework Developer Center – Technology Overview. World Wide Web, http://msdn.microsoft.com/netframework/. Visitado em Novembro de 2003

[MS-EXC] Microsoft Exchange Server. Wordl Wide Web, http://www.microsoft.com/exchange/default.mspx. Visitado em Setembro de 2004.

[MS-HS] Microsoft – Introducing HailStorm. World Wide Web,

http://msdn.microsoft.com/theshow/Episode014/. Visitado em Setembro de 2004.

[MS-IUK] Microsoft – IUnknown. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/com/htm/cmi_q2z_9dwu.asp. Visitado em Setembro de 2004.

[MS-PASSP] Microsoft .NET passport. World Wide Web, http://www.passport.net/Consumer/default.asp?lc=1033. Visitado em Julho de 2004.

[MS-REFL] Microsoft – Discovering Type Information at Run Time. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpcondiscoveringtypeinformationatruntime.asp. Visitado em Novembro de 2004.

[MS-SESOB] Microsoft – Session Object. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/iissdk/iis/ref_vbom_seso.asp. Visitado em Dezmbro de 2003.

[MS-SESST] Microsoft – Session State. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconsessionstate.asp. Visitado em Maio de 2004.

[MS-SQL] Microsoft SQL Server. Wordl Wide Web, http://www.microsoft.com/sql/default.mspx. Visitado em Setembro de 2004.

[MS-SQLDT] Microsoft – Using sql_variant Data. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_da-db_7msw.asp. Visitado em Janeiro de 2004.

[MS-SQL05] Microsoft – Top 30 Features of SQL Server 2005. World Wide Web, http://www.microsoft.com/sql/2005/productinfo/top30features.asp#B. Visitado em Outubro de 2004.

[MS-STRCT] Microsoft – struct. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vcrefStructTypes.asp. Visitado em Novembro de 2003.

[MS-VS] Microsoft Visual Studio Developer Center. World Wide Web, http://msdn.microsoft.com/vstudio/. Visitado em Março de 2004.

[MS-WCC] Microsoft – Introduction to Web Custom Controls. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbconintroductiontocustomwebcontrols.asp. Visitado em Dezembro de 2003.

[MS-WUC] Microsoft – Introduction to Web User Controls. World Wide Web, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbconintroductiontowebusercontrols.asp. Visitado em Dezembro de 2003.

[MS-XMLWS] Microsoft – XML Web Services Overview. World Wide Web. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconwebservicesoverview.asp. Visitado em Outubro de 2004.

[NINE] NINE – The Phonogram Page. World Wide Web, http://www.phonogrampage.com/index.php. Visitado em Setembro de 2004.

[OASIS] OASIS. World Wide Web, http://www.oasis-open.org/home/index.php. Visitado em Dezembro de 2003.

[OASIS]-WSS] OASIS – Web Services Security TC. World Wide Web, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss. Visitado em Dezembro de 2003.

[OSCAR-AC1] Óscar Narciso Mortágua Pereira, Joaquim Sousa Pinto, António José Batel Anjo, "Object Oriented Platform to RDBMS Stored Procedures", In Proceedings of IADIS International Conference Applied Computing 2005 - Vol II, 22-25 Fevereiro de 2005, Carvoeiro, Algarve, Portugal, pág 99-106. ISBN: 972-99353-6-X

[OSCAR-AC2] Óscar Narciso Mortágua Pereira, Joaquim Sousa Pinto, António José Batel Anjo, "Methodology for Dynamic Web Pages Assembly", In Proceedings of IADIS International Conference Applied Computing 2005 – Vol II, 22-25 Fevereiro de 2005, Carvoeiro, Algarve, Portugal, pág 17-20. ISBN: 972-99353-6-X

[OSCAR-CELDA] Óscar Narciso Mortágua Pereira, Joaquim Sousa Pinto, "abcNet: Literacy Tool Based on Entities", In Proceedings of IADIS International Conference: CELDA 2004, Cognition and Exploratory Learning in Digital Age, 15-17 Dezembro 2004, Lisboa, Portugal, pág 59-66. Endorsed By IEEE Technical Committee on Learning Technology. ISBN: 972-98947-7-9

[OSCAR-eSOC] Óscar Narciso Mortágua Pereira, Joaquim Sousa Pinto, "abc-Net: Literacy Tool for Paulo Freire´s Method", In Proceedings of IADIS International Conference e-Society 2004, 16-19 Julho de 2004, Ávila, Espanha. Vol II, pág. 917-920. ISBN: 972-98947-5-2.

[OSCAR-ICALT] Óscar Narciso Mortágua Pereira, Joaquim Sousa Pinto, António Batel Anjo, Lygia Maria Marques da Conceição Pereira, "abcNet: a Literacy Tool in Developing Countries", In Proceedings of ICALT 2004, The 4th IEEE International Conference on Advanced Learning Technologies. IEEE Computer Society, Joensuu, Finland, 30 Agosto - 1 Setembro 2004, pág 948-952. ISBN: 0-7695-2181-9

[PERL] Perl.Designerz.com – Perl NET. World Wide Web, http://perl.designerz.com/perl-net.php. Visitado em Setembro de 2004.

[PE-DISLX] Cristina de Sá Coutinho. Cadernos de Reeducação Pedagógica - Dislexia. Porto Editora, Porto, 2001. ISBN: 972-0-34551-9.

[PE-M28P] Arminda Craveiro, Adriana Figueiredo, Maria Teresa Dias. Palavra a Palavra, Livro de Apoio ao Método das 28 Palavras. Porto Editora, 2003. ISBN: 972-0-11105-4.

[PE-SITIO] Sítio dos Miúdos. Porto Editora. World Wide Web, http://www.sitiodosmiudos.pt/sitio.asp. Visitado em Junho de 2004.

[PE-TRAMP] Felisbina Antunes e Fátima Lemos. Trampolin 1 – Língua Portuguesa. Porto Editora, Porto, 2004. ISBN: 972-0-11131-3.

[PF-ECPL] Paulo Freire. Educação como Prática de Liberdade. Editora Paz e Terra, Rio de Janeiro, Brasil, 1975. 5ª Edição

[PF-PA] Paulo Freire. Pedagogia da Autonomia – saberes necessários à prática educativa. Editora Paz e Terra, S. Paulo, Brasil, 2001. 17ª Edição. ISBN: 85-219-0243-3.

[PF-PO] Paulo Freire. Pedagogia do Oprimido. Editora Afrontamento, Porto, 1972.

[RHONA] Rhona Johnston, Joyce Watson. Literacy Today – Synthetically successful?. World Wide Web, http://www.literacytrust.org.uk/Pubs/synthetic.html. Visitado em Janeiro de 2004.

[ROGER] Roger T. Alexander et al. 2000. Criteria for Testing Polymorphic Relationships. 11th International Symposium on Software Reliability Engineerig (ISSRE'00), San Jose,

CA, pág. 15-23.

[SALFORD] Salford Software. World Wide Web, http://www.polyhedron.com/salford/. Visitado em Fevereiro de 2004.

[SONIA] O método Paulo Freire. World Wide Web, http://www.paulofreire.org/metodo.htm. Visitado em Junho de 2004.

[STARFALL] Starfall. World Wide Web, http://www.starfall.com. Visitado em Maio de 2004.

[TE-JUNR] Júnior. Texto Editora. World Wide Web, http://junior.te.pt/escolinha/anosLista.jsp?p=1&d=lp&t=apr. Visitado em Junho de 2004.

[THUN] Thun Thai e Hoang Q. Lam, ".NET Framework Essentials". First Edition, 2001. O´Reilly. ISBN 0-596-00165-7.

[UDDI] OASIS – UDDI. World Wide Web, http://www.uddi.org/. Visitado em Janeiro de 2004.

[W3C-CSS] W3C – Cascading Style Sheets Home Page. World Wide Web, http://www.w3.org/Style/CSS/. Visitado em Janeiro de 2004.

[W3C-HTML] W3C – HyperTextt Markup Language Home Page. World Wide Web, http://www.w3c.org/MarkUp/. Visitado em Janeiro de 2004

[W3C-HTTP] W3C – Hypertext Transfer Protocol. World Wide Web, http://www.w3c.org/Protocols/. Visitado em Dezembro de 2003.

[W3C-SOAP] XML Protocol Working Group. World Wide Web, http://www.w3c.org/2000/xp/Group/. Visitado em Julho de 2004.

[W3C-WAI] W3C – Web Accessibility initiative. World Wide Web, http://www.w3c.org/WAI/. Visitado em Janeiro de 2004.

[W3C-WS] W3C – Web Services Activity. World Wide Web, http://www.w3c.org/2002/ws/. Visitado em Dezembro de 2003.

[W3C-WSDL] W3C - Web Services Description Language. World Wide Web, http://www.w3.org/TR/wsdl. Visitado em Fevereiro de 2004.

[W3C-XML] W3C – Extensible Markup Language. World Wide Web, http://www.w3c.org/XML/. Visitado em Dezembro de 2003.