Universidade Federal do Ceará

Centro de Tecnologia

Departamento de Engenharia de Teleinformática

Curso de Graduação de Engenharia de Teleinformática

ARQUITETURA ORIENTADA A SERVIÇOS EXPANSÍVEL PARA ADAPTAÇÃO DE INTERFACES ACESSÍVEIS NA WEB – IMPLEMENTAÇÃO PARA DEFICIÊNCIA VISUAL

Francisco Daniel Coutinho Medeiros

Fortaleza – Ceará

Dezembro/2011

Francisco Daniel Coutinho Medeiros

ARQUITETURA ORIENTADA A SERVIÇOS EXPANSÍVEL PARA ADAPTAÇÃO DE INTERFACES ACESSÍVEIS NA WEB – IMPLEMENTAÇÃO PARA DEFICIÊNCIA VISUAL

Projeto Final de Curso submetido à Coordenação do programa de Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para a obtenção do grau de **Engenheiro de Teleinformática**.

Autor: Francisco Daniel Coutinho Medeiros
Orientador: Prof. Dr. José Marques Soares

Fortaleza – Ceará

Dezembro/2011

Título do Trabalho: ARQUITETURA ORIENTADA A SERVIÇOS EXPANSÍVEL PARA ADAPTAÇÃO DE INTERFACES ACESSÍVEIS NA WEB – IMPLEMENTAÇÃO PARA DEFICIÊNCIA VISUAL

Autor: Francisco Daniel Coutinho Medeiros

Defesa em: 07 / 12 / 2011 Nota Obtida: _____

Banca Examinadora

_____

Prof. Dr. José Marques Soares

Orientador

_____

Prof. Dr. Danielo Gonçalves Gomes

Universidade Federal do Ceará

_____

Prof. Dr. Giovanni Cordeiro Barroso

Universidade Federal do Ceará

Fortaleza, 07 de Dezembro de 2011

Dedico este trabalho aos meus pais, Marcelo e Raimunda, pelos valiosos ensinamentos que levarei por toda a vida.

AGRADECIMENTOS

A Deus, por ter sempre me guiado no caminho da verdade. Sem Ele, a conclusão deste trabalho teria sido impossível.

À minha mãe Raimunda Maria Coutinho Medeiros, por ter me ensinado ser tudo que sou hoje. Ao meu pai Marcelo de Queiroz Medeiros, por ter me dado a maturidade necessária para enfrentar as adversidades. Aos meus avós pelo carinho e pelas palavras de sabedoria. Aos meus irmãos Thiago, Rafael e Lucas, pela companhia, apoio e divertimento. À minha madrasta Maria Abreu, pelo carinho, paciência, acolhimento e apoio.

À minha noiva, Idalina, pelo carinho, confiança, compreensão e grande apoio nos momentos difíceis.

Aos amigos, pelo divertimento, pela força e pelos sinceros votos de sucesso neste trabalho.

Ao meu orientador, Prof. Dr. José Marques, pelos ensinamentos durante minha vida acadêmica e principalmente neste trabalho.

Aos demais professores do Departamento de Engenharia de Teleinformática, pelos conhecimentos transmitidos durante a minha graduação como Engenheiro.

À Universidade Federal do Ceará, pela formação de nível superior de qualidade.

Resumo

Com a popularização da Internet, cada vez mais o cliente está acessando os serviços das empresas por meio de sistemas Web. No entanto, surge uma pergunta inerente a esta evolução: “Como fazer com que os sítios web sejam realmente acessíveis, independente das suas necessidades e limitações?”. É fato que a parcela de usuários com algum tipo de deficiência está crescendo e a maioria dos sistemas web não está atendendo às suas necessidades, aumentando cada vez mais a exclusão digital destes tipos de usuários. Tecnicamente, alguns pontos ainda dificultam o desenvolvimento acessível na Web. Um deles é o fato de que o tema de acessibilidade é pouco abordado no meio acadêmico, tornando a comunidade de desenvolvedores Web menos capacitada no assunto. Por conta disso, existe um esforço maior na fase de desenvolvimento de um sistema acessível. Isso faz com que as corporações fujam do desenvolvimento acessível. Este trabalho visa, além de disseminar o tema de acessibilidade no meio acadêmico, propor uma arquitetura aberta e flexível, que seja adaptável através de interfaces propriamente definidas pelas diferentes necessidades do usuário com o mínimo de retrabalho. Arquitetura proposta possui módulos que proporciona a adaptação de interfaces não acessíveis para interfaces acessíveis. Deste modo, mesmo num cenário com um sistema já existente e não acessível, a arquitetura pode adaptar sua interface de diversas formas de acordo com a necessidade do usuário. Com esta solução, as corporações poderão desenvolver sistemas acessíveis rápidos, modulares e expansíveis e, consequentemente, proporcionarão inclusão digital dos usuários deficientes.

PALAVRAS-CHAVE: Sistemas Web, Acessibilidade.

Abstract

With the popularization of Internet, more and more customers are accessing the services of companies by means of Web systems, however, a question arises inherent in this development: "How to make web sites are actually accessible, regardless of their needs and limitations? ". It is a fact that the share of users with a disability is growing and most web systems are not meeting their needs, thereby increasing the digital divide more of these types of users. Technically, some issues still hinder the development accessible on the Web One of them is the fact that the issue of accessibility is little explored in academic circles, making the Web developer community less suitable in the subject. Because of this, there is a greater effort in the development phase of an affordable system. This makes development corporations flee accessible. This work aims to spread beyond the issue of accessibility in the academy, to propose an open and flexible architecture that is adaptable through strictly defined interfaces for different user needs with minimal rework. Proposed architecture has modules that provide the adaptation of not accessible interfaces to accessible interfaces. Thus, even in a scenario with an existing system and not accessible, the architecture can adapt the interface in various ways according to user need. With this solution, corporations can develop accessible systems fast, modular and expandable and thus provide digital inclusion of disabled users.

KEYWORDS: Web Systems, Accessibility.

SUMÁRIO

1. Introdução 9
   1.1. Motivação 9
   1.2. Objetivos 10
   1.3. Metodologia 10
   1.4. Organização do Texto 11
2. Deficiência Visual 12
   2.1. Conceitos 12
   2.2. Dados Estatísticos 12
   2.3. Dificuldades Encontradas pelos Deficientes Visuais no Acesso Web 13
   2.4. Considerações Finais 14
3. Acessibilidade na Web 16
   3.1. Organização das Diretrizes de Acessibilidade 16
   3.2. Validadores de Acessibilidade 16
      3.2.1. HERA 17
      3.2.2. Examinator 18
      3.2.3. Cynthia Says 18
      3.2.4. DaSilva 19
   3.3. Considerações Finais 20
4. Tecnologias Assistivas para Deficientes Visuais 21
   4.1. Interfaces para Usuários com Baixa Visão 21
      4.1.1. Lente de Aumento LentePro 21
      4.1.2. Magic 22
   4.2. Interfaces para Usuários Cegos 22
      4.2.1. Linha Braille 22
      4.2.2. Impressora Braille 23
   4.3. Sistemas Operacionais 23
      4.3.1. Dosvox 23
   4.4. Leitores de Tela 24
      4.4.1. Monitvox 25
      4.4.2. Virtual Vision 25
   4.5. Considerações Finais 26
5. Arquitetura Orientada a Serviços 27
   5.1. Características 27
   5.2. Descrição dos Papéis em SOA 28
   5.3. Web Service 28
   5.4. Benefícios em adotar Web Services e SOA 30
      5.4.1. Sistemas Legados 30
      5.4.2. Uso de padrões XML 31
   5.5. SOAP 31
   5.6. Considerações Finais 32
6. Projeto 33
   6.1. Descrição Geral 33
   6.2. Ferramentas de Desenvolvimento 34
   6.3. Arquitetura Proposta 34
   6.4. Considerações Finais 36

7. Estudo de Caso 38
7.1. Identificação dos Serviços 40
7.1.1. Serviço de Listagem de Funcionalidades 40
7.1.2. Serviço de Listagem de Alunos 41
7.1.3. Serviço de Inclusão de Alunos 42
7.1.4. Serviço de Consulta de Aluno 45
7.1.5. Serviço de Atualização de Alunos 46
7.1.6. Serviço de Exclusão de Alunos 48
7.2. Estrutura do Projeto 48
7.3. Considerações Finais 50
8. Resultados 52
9. Conclusão e Trabalhos Futuros 61
Referências Bibliográficas 63
Apêndices 66
Apêndice 1. Critérios de Acessibilidade Nível A 66
- Alternativas de Texto 66
- Multimídia Baseada no Tempo 67
- Formulários Adaptáveis 67
- Conteúdos Distinguíveis 67
- Acessibilidade de Teclado 68
- Limite de Tempo de Uso do Sistema 68
- Navegabilidade 69
- Legibilidade 70
- Compatibilidade 70

Apêndice 2. Critérios de Acessibilidade Nível AA 70
- Multimídia Baseada no Tempo 71
- Conteúdos Distinguíveis 71
- Assistência de Entrada de Dados 71

Apêndice 3. Critérios de Acessibilidade Nível AA 71
- Multimídia Baseada no Tempo 71
- Conteúdos Distinguíveis 72

Anexos 73
- add_service.wsdl 73
- all_service.wsdl 75
- delete_service.wsdl 76
- index_service.wsdl 77
- update_service.wsdl 78
- view_service.wsdl 80

LISTA DE FIGURAS

Figura 1.1 – Metodologia do Projeto com Noção Temporal 10
Figura 3.1 – Página Inicial do Validador HERA 17
Figura 3.2 – Página Inicial do Validador Examinator 18
Figura 3.3 – Página Inicial do Validador Cynthia Says 19
Figura 3.4 – Página Inicial do Validador DaSilva 20
Figura 4.1 – Área de Trabalho aumentada com o uso do LentePro 1.4 21
Figura 4.2 – Software Magic 8.0 22
Figura 4.3 – Linha Braille 22
Figura 4.4 – Impressora Braille 23
Figura 4.5 – Interface do Programa Dosvox 24
Figura 4.6 – Painel de Controle do Virtual Vision 26
Figura 5.1 – Modelo primitivo de SOA. Fonte: [3] 28
Figura 6.1 – Diagrama de Fluxo da Arquitetura Proposta sem Sistema Legado 35
Figura 6.2 – Diagrama de Fluxo da Arquitetura Proposta mantendo o Sistema Legado 36
Figura 7.1 – Tela Inicial (Sistema Legado) 38
Figura 7.2 – Tela de Inicial (Sistema Legado) Lida pelo Dosvox 39
Figura 7.3 – Estrutura do Projeto Cliente no Eclipse 49
Figura 7.4 – Estrutura do Projeto Servidor no Eclipse 50
Figura 8.1 – Tela de Seleção de Interface 52
Figura 8.2 – Tela de Seleção de Interface (Dosvox) 53
Figura 8.3 – Tela Inicial para Usuários Deficientes Visuais 54
Figura 8.4 – Tela Inicial para Usuários Deficientes Visuais (Dosvox) 54
Figura 8.5 – Tela de Inclusão de Alunos para Usuários Deficientes Visuais 56
Figura 8.6 – Tela de Inclusão de Alunos para Usuários Deficientes Visuais (Dosvox) 56
Figura 8.7 – Tela de Visualização de Alunos para Usuários Deficientes Visuais 57
Figura 8.8 – Tela de Visualização de Alunos para Usuários Deficientes Visuais (Dosvox) ... 57
Figura 8.9 – Resultado da Tela de Seleção de Interface (validador HERA) 58
Figura 8.10 – Resultado da Tela de Seleção de Interface (validador DaSilva) 59
Figura 8.11 – Resultado da Tela Inicial para Deficientes Visuais (validador HERA) 59
Figura 8.12 – Resultado da Tela Inicial para Deficientes Visuais (validador DaSilva) 60

LISTA DE ABREVIATURAS

CSS	– Cascading Style Sheet
HTTP	– HyperText Transfer Protocol
HTTPS	– HyperText Transfer Protocol Secure
PHP	– HyperText Preprocessor
RPC	– Remote Procedure Calls
SQL	– Service Query Language
SOA	– Service-Oriented Architecture
SOAP	– Simple Object Access Protocol
UDDI	– Universal Description, Discovery, and Integration
XML	– Extensible Markup Language
W3C	– World Wide Web Consortium
WAI	– Web Accessibility Initiative
WS	– Web Services
WSDL	– Web Service Descriptor Language
WSF/PHP	– Web Service Framework for PHP

1. Introdução

Há 50 anos era difícil uma conversa entre uma pessoa com outra de outro lado do planeta. Com o advento e a popularização da Internet, as pessoas passaram a se comunicar de forma instantânea. Podemos trocar informações com qualquer pessoa a qualquer hora e em qualquer lugar. Essa facilidade de acesso à informação fez com que a Internet admitisse diferentes tipos de usuários com diferentes tipos de necessidade. Dentre estes, os usuários que possuem algum tipo de limitação física ou mental. Entretanto, percebeu-se que estes usuários não tinham a mesma facilidade de acesso às informações disponibilizadas na Internet se comparados com usuários sem deficiência.

Daí surgiu o termo "Acessibilidade à Web", que segundo Thatcher *et al.* (p. 2) [1] significa:

> *Acessibilidade à Web significa que pessoas portadoras de necessidades especiais sejam capazes de usar a Web. Mais concretamente, significa uma Web projetada de modo a que estas pessoas possam perceber, entender, navegar e interagir de uma maneira efetiva com a Web, bem como criar e contribuir com conteúdos para a Web.*

1.1. Motivação

Atualmente o desenvolvimento de sistemas Web robustos é necessário para o sucesso das corporações em geral. Isso porque a Internet é um dos principais meios para a divulgação e disponibilização de serviços. Pessoas com necessidades diferentes estão cada vez mais preferindo acessar estes serviços através da Internet. Isso faz com que seja essencial uma forma de desenvolvimento que contemple as diferentes necessidades dos usuários.

Porém dependendo da arquitetura que se adota e das tecnologias que o projeto utiliza, pode ser que o mesmo sistema precise de várias versões, com cada versão atendendo a uma necessidade específica. Entretanto, o baixo custo é um requisito fundamental para um projeto de sistema Web de sucesso. A motivação deste projeto está em reduzir o retrabalho de desenvolvimento definindo uma arquitetura gratuita e com interface adaptável. Desta forma o sistema Web poderá ser preparado para atender às mais diversas necessidades dos usuários através da adaptação, para cada caso, da interface de suas funcionalidades principais.

1.2. Objetivos

O objetivo principal deste trabalho é apresentar uma solução que facilite o desenvolvimento de sistemas Web acessíveis. Para isso, pretende-se definir uma arquitetura que atenda às diversas necessidades dos usuários sem comprometer no funcionamento do sistema e com o mínimo de reprogramação. Impõe-se como requisito inicial, que a arquitetura definida seja gratuita e que utilize tecnologias de fácil integração com sistemas Web já consolidados.

Além disso, este trabalho visa difundir o tema "Acessibilidade à Web" no meio acadêmico, pois percebe-se que os profissionais da área de TI (Tecnologia da Informação) que estão entrando no mercado, em geral, não são capacitados em seus cursos de formação nesta área do conhecimento e a parcela de pessoas com algum tipo de deficiência está crescendo, o que faz com que a requisição de sistemas Web acessíveis seja cada vez maior.

1.3. Metodologia

Figura 1.1 – Metodologia do Projeto com Noção Temporal

Conforme exposto na Figura 1.1, o projeto iniciou na fase de Revisão Bibliográfica, que foi focada no desenvolvimento de competências sobre os temas de acessibilidade, acessibilidade na Web e Arquitetura Orientada a Serviços (SOA).

Em seguida, foi definido um escopo para o projeto com a definição do problema. O escopo deste projeto é a implantação de uma arquitetura que visa, dentre outras coisas, facilitar o desenvolvimento de módulos de interfaces adaptativas. Está também no escopo do projeto a implementação de um sistema de exemplo (Sistema de Cadastro de Alunos com Deficiência) e de um módulo de interface adaptativa para deficiente visual.

Os conhecimentos adquiridos deram a base teórica necessária para a fase principal deste projeto: A Concepção da Arquitetura. Nesta fase, definiu-se uma arquitetura que atenda

aos requisitos de escalabilidade, adaptabilidade e interoperabilidade. Além disso, é necessário que esta arquitetura não exija qualquer necessidade de manutenção do sistema legado (caso haja).

Em seguida, foram escolhidas as ferramentas de desenvolvimento necessárias para desenvolvimento do sistema e de sua interface para deficiente visual. Nesta etapa, priorizou-se ferramentas gratuitas justamente para minimizar custos de desenvolvimento e manutenção do projeto.

Com o intuito de validar a arquitetura proposta, foi desenvolvido um sistema simples de cadastro de alunos e implementado apenas a versão legada (sem o uso de Web Services) e o módulo para deficientes visuais como um exemplo prático do funcionamento da arquitetura.

Por fim, foi possível um ciclo de testes e melhorias. O sistema foi testado com o uso de ferramentas tradicionais de acesso Web para usuários deficientes visuais e um conjunto de melhorias foram identificadas e implementadas.

Vale salientar na Figura 1.1, que a noção temporal de cada bloco mostra que o projeto teve seu esforço igualmente distribuído ao longo de 2 períodos letivos (1 ano). Além disso, percebe-se que a fase de revisão bibliográfica, concepção da arquitetura e implementação do sistema de cadastro (avaliação da arquitetura) exigiram mais tempo do que as demais justamente pela importância destas fases para o sucesso do projeto.

1.4. Organização do Texto

Este trabalho está organizado em sete capítulos da forma como se segue. No Capítulo 2 são apresentadas algumas estatísticas de deficientes visuais no Brasil e no mundo e, for fim, são abordados os tipos de deficiência visual (desde a baixa visão até a cegueira), bem como as dificuldades encontradas por cada um destes usuários quando do acesso Web. No Capítulo 3 são definidas as diretrizes para auxiliar o desenvolvimento de sistemas Web acessíveis e apresentados alguns validadores de acessibilidade. No Capítulo 4 são apresentadas as diversas ferramentas que auxiliam o deficiente visual no uso do computador. No Capítulo 5 são detalhadas as tecnologias utilizadas na concepção da arquitetura e no desenvolvimento do sistema piloto acessível e, por fim, apresentada a arquitetura proposta. No Capítulo 6 são apresentados os resultados e experimentações para a avaliação do trabalho. Finalmente, no Capítulo 7 são destacadas as conclusões principais deste trabalho, bem como são discutidas as perspectivas e as sugestões para trabalhos futuros.

2. Deficiência Visual

2.1. Conceitos

De acordo com o Decreto n° 3.298 de 21 de Dezembro de 1999 art. 2° do Capítulo I [16]:

> *Considera-se deficiência toda restrição física, mental ou sensorial, de natureza permanente ou transitória, que limita a capacidade de exercer uma ou mais atividades essenciais da vida diária e/ou atividade remunerada.*

Este mesmo artigo deste Decreto define cegueira como "acuidade visual[1] igual ou menor que 0,05 no melhor olho, com a melhor correção óptica", baixa visão é definida como "acuidade visual entre 0,5 e 0,05 no melhor olho e com a melhor correção óptica".

Já de acordo com a Rede Entre Amigos [19]:

> *O termo deficiência visual refere-se a uma situação irreversível de diminuição da resposta visual, em virtude de causas congênitas ou hereditárias, mesmo após tratamento clínico e/ou cirúrgico e uso de óculos convencionais. A diminuição da resposta visual pode ser leve, moderada, severa, profunda (que compõe o grupo de visão subnormal ou baixa visão) e ausência total da resposta visual (cegueira).*

2.2. Dados Estatísticos

Resnikoff relata em [20] que cerca de 314 milhões de pessoas, o que resulta em torno de 4% da população mundial, possuem algum grau de deficiência visual. Segundo este estudo, as principais causas da cegueira no mundo são:

Catarata – 39,1%

Erros Refrativos Não Corrigidos – 18,2%

Glaucoma – 10,1%

Degeneração Macular – 7,1%

[1] "Em termos perceptuais, a Acuidade Visual pode ser melhor definida como sendo a capacidade de resolução do sistema visual como um todo. Em outras palavras, a resposta de detecção e reconhecimento de um padrão se dá a partir da imagem projetada na retina, codificada e processada através das conexões entre as estruturas nervosas que compõem o sistema visual." ([17] p. 1)

Opacidades de Córnea – 4,2%

Retinopatia Diabética – 3,9%

Cegueira Infantil – 3,2%

Tracoma – 2,9%

Oncocerquíase – 0,7%

Outros – 10,6%

O IBGE aponta no Censo 2010 cerca de 128.000 deficientes visuais no Brasil (cerca de 0,075% da população). Entretanto, estes números estão muito aquém da média mundial para os países em desenvolvimento (aproximadamente 0,6% da população). Logo, o Censo 2010, bem como no Censo 2000, ainda não conseguiu levantar dados condizentes com a realidade brasileira.

2.3. Dificuldades Encontradas pelos Deficientes Visuais no Acesso Web

Com base no que foi exposto nos tópicos anteriores deste capítulo, não se pode negar que os usuários deficientes compõem uma parcela significativa da sociedade e que se faz necessário incorporar estes usuários nos critérios de usabilidade para que os mesmos possam navegar nos sistemas Web tão bem quanto os demais.

Mas quando se pensa na apresentação de páginas, seus efeitos visuais atraem muitas pessoas, aumentando o número de visitas nas mesmas, evento imprescindível no plano comercial. Por outro lado, a Internet é uma potente ferramenta para videntes, já que lhes permite ascender ao mundo da informação. Assim, eles deveriam ter acesso a todas as páginas, inclusive às que contém os mais diversos tipos de imagens.

Entretanto, o que se vê é que infelizmente muitos sítios não são acessíveis à pluralidade dos seus usuários. Deste modo, o grande desafio, atualmente, é desenvolver páginas que incorporem essa diversidade (ainda que nem sempre seja possível).

De acordo com a SERPRO [22], as dificuldades encontradas pelos deficientes visuais quando do acesso Web são:

Cegueira:

- Imagens que não possuem texto alternativo;
- Vídeos que não possuem descrição textual ou sonora;
- Uso de tabelas que não fazem sentido se lidas célula por célula;

- Uso de paginação sem descrição textual (apenas numérica);
- Uso de frames, pois não são lidos por alguns leitores de tela;
- Formulários não rotulados ou que não podem ser navegados em sequência lógica;
- Documentos formatados sem seguir os padrões web especificados pela W3C[2]

Baixa Visão:

- Páginas com tamanhos de fonte absoluta, dificultando o redimensionamento;
- Páginas que, devido ao layout inconsistente, são difíceis de navegar, quando ampliadas, devido à perda de conteúdos adjacentes;
- Páginas ou imagens que possuem pouco contraste;
- Textos que são apresentados como imagens, pois não quebram a linha quando ampliadas;
- Se o grau residual de visão for muito baixo, as barreiras podem ser as mesmas das já citadas no item anterior (relativa à cegueira).

Daltonismo:

- Cor utilizada como único recurso para enfatizar o texto;
- Contrastes inadequados entre cores de fonte e cores de fundo;
- Navegadores que não suportam a opção para o usuário utilizar sua própria folha de estilo[3].

2.4. Considerações Finais

Neste capítulo foram apresentados alguns dados estatísticos sobre a deficiência visual nos seus mais diversos estágios. Além disso, este capítulo abordou algumas das principais dificuldades encontradas por estes usuários ao acessar os sistemas Web vigentes. Tais dificuldades não podem ser ignoradas tendo em vista o constante crescimento da população deficiente visual.

Apesar deste trabalho se concentrar na concepção de uma arquitetura para auxiliar no desenvolvimento de interfaces para sistemas Web acessíveis, ele possui uma motivação secundária, porém não menos importante, que é a de apresentar boas práticas de

[2] World Wide Web Consortium. Órgão internacional sem fins lucrativos e sem nenhuma relação com empresas desenvolvedoras de software, cujo propósito é definir padrões para a criação e a interpretação dos conteúdos para a Web.

[3] Conjunto de declarações que especificam a apresentação do documento. Folha de estilo CSS trata-se de uma marcação que proporciona efeitos de formatação (e não estruturais) na página HTML.

desenvolvimento de sistemas Web acessíveis e tornar mais claro o entendimento das necessidades dos deficientes visuais.

No próximo capítulo, serão apresentadas algumas diretrizes definidas pela W3C para auxiliar a comunidade de desenvolvimento Web na criação de sistemas Web acessíveis. Além disso, serão apresentados alguns validadores de acessibilidade que são ferramentas de teste imprescindíveis para o desenvolvedor Web.

3. Acessibilidade na Web

3.1. Organização das Diretrizes de Acessibilidade

Com o intuito de criar padrões de acessibilidade na Web, o W3C criou um grupo, chamado WAI (*Web Accessibility Initiative*). O WAI publicou diretrizes para acessibilidade de sítios Web, navegadores e ferramentas de criação de conteúdo na Internet, a fim de tornar mais fácil o uso da Web para pessoas portadoras de necessidades especiais.

No que tange à organização do conteúdo, o WAI definiu diretrizes chamadas de Recomendações para Acessibilidade do Conteúdo Web (WCAG – *Web Content Accessibility Guidelines*). Este documento já está na sua segunda versão e é seguido por todos os desenvolvedores Web que se preocupam com o desenvolvimento Web respeitando alguns critérios de acessibilidade.

Existem basicamente 3 níveis de acessibilidade:

- **Nível A:** Definem critérios que os desenvolvedores de conteúdo Web devem satisfazer inteiramente;
- **Nível AA:** Definem critérios que os desenvolvedores de conteúdo Web deveriam satisfazer;
- **Nível AAA:** Definem critérios que os desenvolvedores de conteúdo Web podem satisfazer.

Estes 3 níveis são discutidos mais detalhadamente nos Apêndices deste trabalho.

3.2. Validadores de Acessibilidade

Os avaliadores ou validadores de acessibilidade são ferramentas automáticas que verificam o código de uma página emitindo relatórios que indicam inconformidades de acessibilidade segundo as prioridades sugeridas pela WAI-W3C.

O número de avisos em relatórios de acessibilidade normalmente supera em muito a quantidade de inconformidades listadas. Isso ocorre em razão da capacidade limitada das regras que podem ser testadas automaticamente por esses softwares.

Existem diferenças relevantes entre as ferramentas de avaliação de acessibilidade, principalmente na sua aderência aos padrões Web. Portanto, para obter um bom resultado, é aconselhável testar a página Web em mais de um desses validadores.

Vale salientar que a metodologia para se fazer uma boa acessibilidade em uma página não se resume na aprovação desses avaliadores automáticos, eles são tão somente referência

para se chegar a uma boa acessibilidade e para descobrir erros muitas vezes imperceptíveis em uma avaliação manual. Uma avaliação também feita somente por pessoas com deficiência, embora de elevada relevância, pode se limitar a um ou alguns tipos de deficiência. O conceito de acessibilidade é abrangente e deve ser o mais universal possível, aderente às necessidades de todas as pessoas com deficiência, para todos os tipos de acesso (rápidos ou lentos, banda larga ou discada) e para todos os tipos de dispositivos (laptops, celulares, dispositivos que possuem tecnologias assistivas, etc.).

Os tópicos a seguir apresentam os validadores de acessibilidade mais conhecidos e utilizados.

3.2.1. HERA

HERA[4] (Figura 3.1) é uma ferramenta para rever a acessibilidade das páginas Web de acordo com as recomendações das Diretrizes de Acessibilidade para o Conteúdo Web 1.0 (WCAG 1.0)[5]. O HERA efetua uma análise automática da página e disponibiliza informação dos erros encontrados (detectáveis de forma automática) e quais os pontos de verificação que devem ser revistos manualmente.

Figura 3.1 – Página Inicial do Validador HERA

[4] HERA – http://www.sidar.org/hera/
[5] WCAG 1.0 – http://www.w3.org/TR/WCAG10/

3.2.2. Examinator

Permitindo a avaliação da acessibilidade, assim como o HERA, com o objetivo de incentivar a qualidade na concepção das páginas, o Examinator[6] (Figura 3.2) atribui uma nota relativa à cada avaliação. Espera-se que o profissional se constranja em produzir páginas com notas baixas. Além disso, a ferramenta possui um relatório qualitativo, ensinando como produzir a acessibilidade que falta e parabenizando pelos itens de acessibilidade implementados na página.

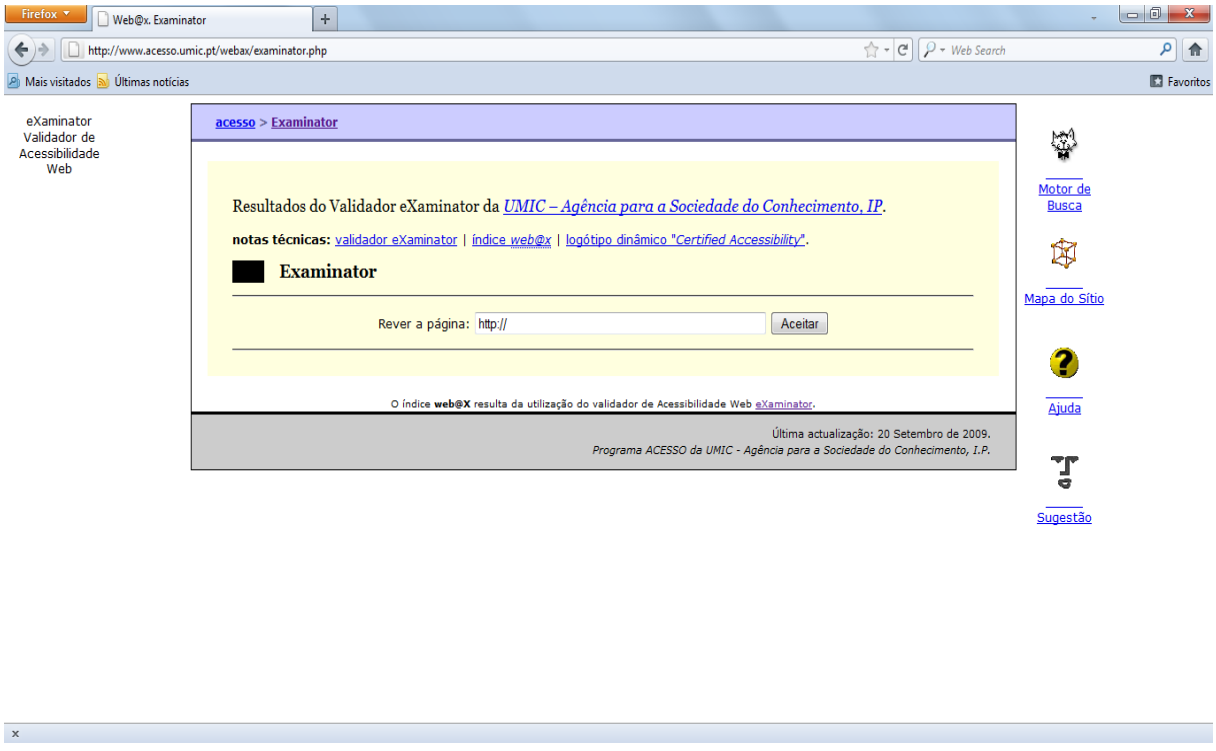

Figura 3.2 – Página Inicial do Validador Examinator

3.2.3. Cynthia Says

O avaliador Cynthia Says[7] (Figura 3.3) possui um relatório que exige maior esforço para ser entendido por desenvolvedores que não estão familiarizados com as diretrizes da WAI-W3C. Porém, dentre desenvolvedores mais experientes, este validador é bastante conhecido e utilizado internacionalmente.

6 Examinator – http://www.acesso.umic.pt/webax/examinator.php
7 Cynthia Says – http://www.cynthiasays.com/

O Cynthia Says traz um validador da HiSoftware[8] que, além de avaliar os sítios Web de acordo com a WCAG, avalia os sites de acordo com o portal Section 508[9]. Este portal americano de acessibilidade se caracteriza por possuir diretrizes mais rigorosas que as contidas na WCAG e por trazer muitas informações tanto para desenvolvedores quanto para pessoas com necessidades especiais sobre leis, fóruns, ferramentas etc.

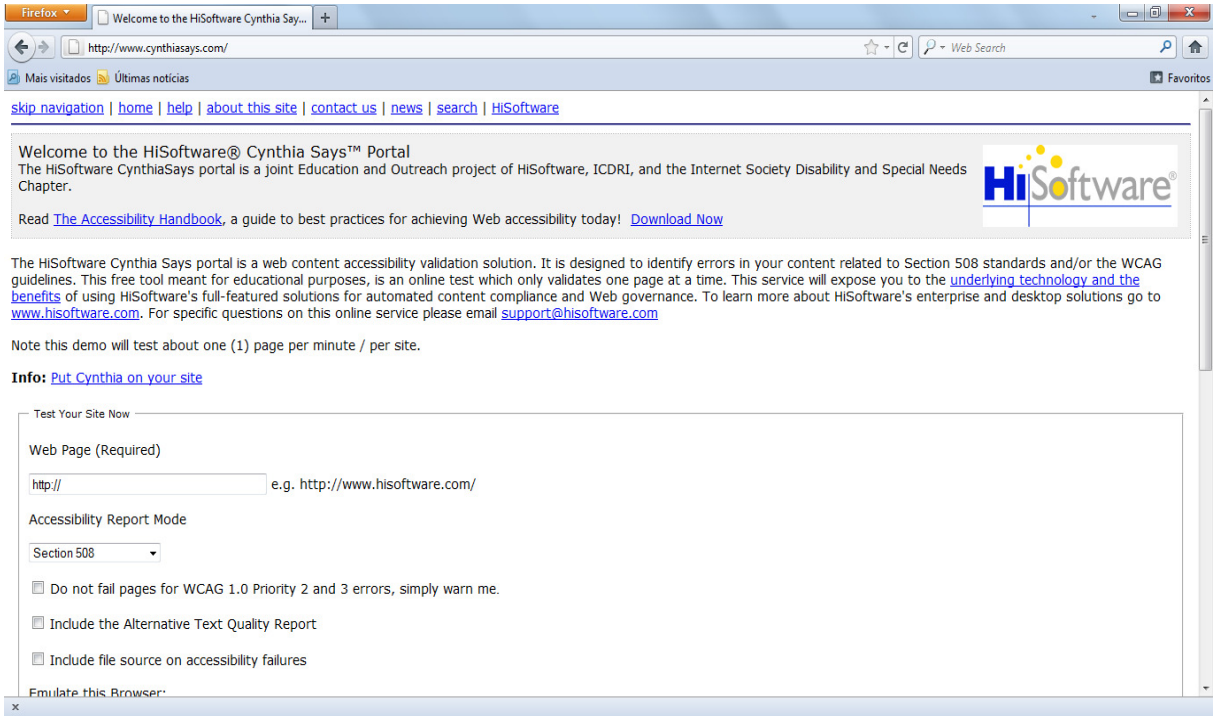

Figura 3.3 – Página Inicial do Validador Cynthia Says

3.2.4. DaSilva

O avaliador DaSilva[10] (Figura 3.4) possui uma versão para a avaliação segundo as diretrizes brasileiras de acessibilidade do governo Eletrônico - EMAG[11], mas tem também a possibilidade de uma avaliação segundo o W3C, baseado no WCAG, que pode ser interessante para as pessoas que estejam iniciando no entendimento de como se faz acessibilidade.

8 HiSoftware – http://www.hisoftware.com/
9 Lei Section 508 – http://www.section508.gov/
10 DaSilva – http://www.dasilva.org.br/
11 EMAG – http://governoeletronico.gov.br/acoes-e-projetos/e-MAG

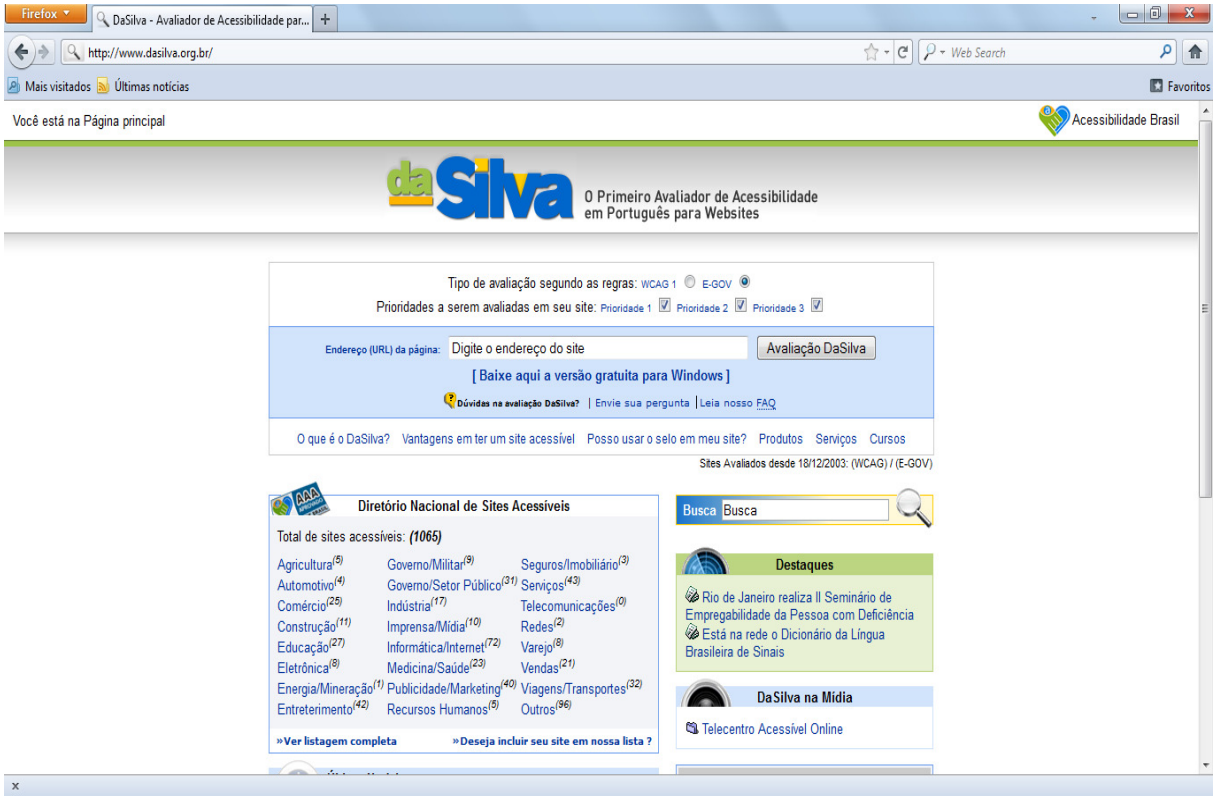

Figura 3.4 – Página Inicial do Validador DaSilva

3.3. Considerações Finais

Neste capítulo foram apresentados alguns dos principais validadores de acessibilidade. Os validadores de acessibilidade são ferramentas importantes na busca de tornar os sítios Web de acordo com os padrões de acessibilidade. Logo, num projeto Web que possui o intuito de contemplar os critérios de acessibilidade, é fortemente recomendado que o sistema seja submetido continuamente aos validadores de acessibilidade.

Neste trabalho, na fase de desenvolvimento, o sistema foi continuamente submetido a 2 validadores: Hera e DaSilva. O motivo desta escolha deveu-se ao fato de que ambos são os mais utilizados dentre a comunidade de desenvolvedores (DaSilva no âmbito nacional e Hera no âmbito mundial).

No próximo capítulo serão listadas algumas tecnologias assistivas bastante usadas pelos deficientes visuais.

4. Tecnologias Assistivas para Deficientes Visuais

4.1. Interfaces para Usuários com Baixa Visão

4.1.1. Lente de Aumento LentePro

O LentePro[12] (Figura 4.1) é um programa ampliador de telas desenvolvido por meio do Projeto *Dosvox,* pelo Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro (NCE-UFRJ). Permite o uso do computador por pessoas que possuem visão subnormal. Por meio dele, o que aparece na tela é ampliado em uma janela (como se fosse uma lupa). O índice de ampliação da imagem dessa janela pode variar de 1 a 9 vezes, permitindo assim que todos os detalhes sejam percebidos mesmo por aqueles com grau muito baixo de acuidade visual. O programa é simples de ser utilizado, ocupa pouco espaço de memória, além de permitir várias alternativas de configuração.

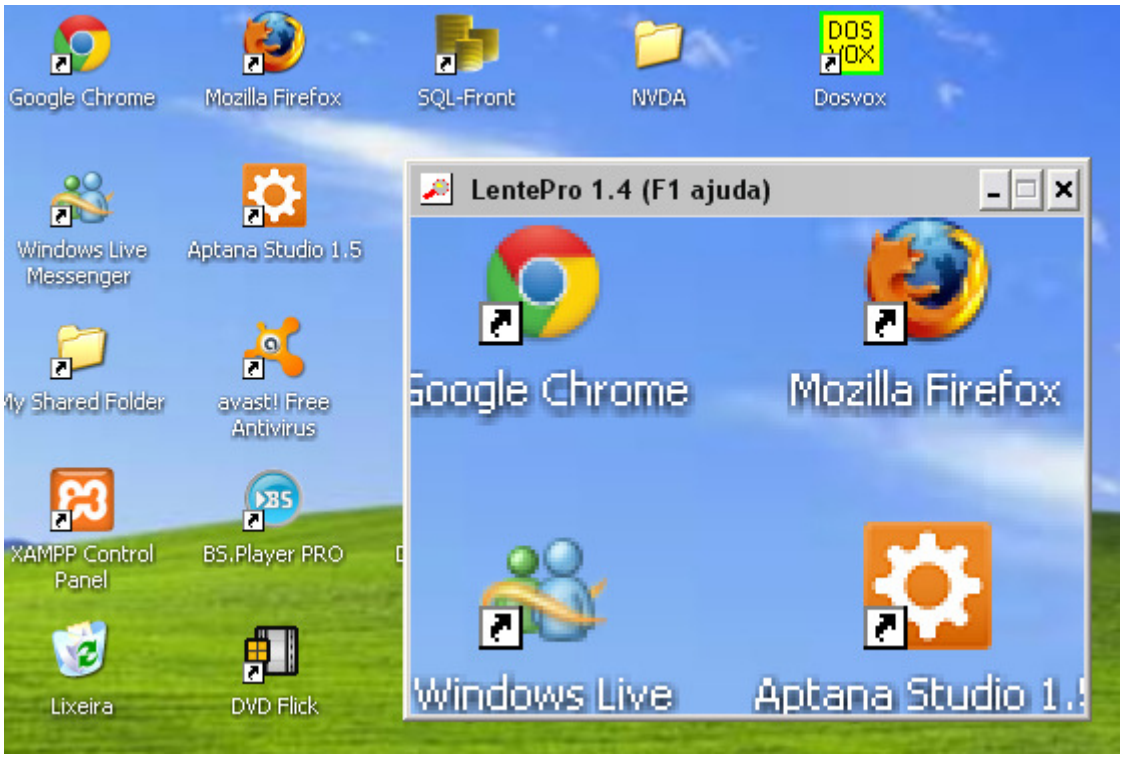

Figura 4.1 – Área de Trabalho aumentada com o uso do LentePro 1.4

[12] LentePro 1.4 (Manual de Operação) – http://www.redespecial.org.br/_doc/lentepro.txt

4.1.2. Magic

Outro exemplo de ampliador de telas é o software Magic[13] (Figura 4.2), da empresa Freedom Scientific, (EUA). Esse programa tem uma capacidade de ampliação de 2 a 16 vezes para ambiente Windows e todos os aplicativos compatíveis. Suas ferramentas permitem alteração de cores e contrastes, rastreamento do cursor ou do mouse, localização do foco do documento e personalização da área da tela antes ou após a ampliação. O aplicativo também pode fazer a leitura da tela por meio de voz sintetizada.

Figura 4.2 – Software Magic 8.0

4.2. Interfaces para Usuários Cegos

4.2.1. Linha Braille

A Linha Braille (Figura 4.3) é um dispositivo de saída ligado ao computador por cabo, que possui uma linha régua de células Braille, cujos pinos se movem para cima e para baixo e que representam uma linha de texto da tela do computador. Trabalha em sincronia com um software leitor de tela, que seleciona os textos e os traduz para o Braille. O número de células braille da régua pode ir de 20 a 80. Os terminais de acesso em Braille geralmente são encaixados a um teclado comum de computador, podendo ser manipulados como se fossem uma linha a mais de teclas, na parte superior ou inferior do teclado. Esse dispositivo pode ser utilizado inclusive por usuários surdo-cegos.

Figura 4.3 – Linha Braille

[13] Magic – http://www.freedomscientific.com/products/lv/magic-bl-product-page.asp

4.2.2. Impressora Braille

As impressoras braile (Figura 4.4) seguem o mesmo conceito das impressoras de impacto comuns e podem ser ligadas ao computador através de portas paralelas ou seriais. Há no mercado uma grande variedade de tipos: de pequeno ou grande porte, com velocidade variada, com impressão em ambos os lados do papel (braille interponto) ou não. Algumas imprimem também desenhos e já existem modelos que imprimem simultaneamente caracteres Braille e comuns em linhas paralelas.

Figura 4.4 – Impressora Braille

4.3. Sistemas Operacionais

4.3.1. Dosvox

O Dosvox [23] (Figura 4.5) é um sistema operacional que vem sendo desenvolvido desde 1993 pelo NCE - Núcleo de Computação Eletrônica da UFRJ (Universidade Federal do Rio de Janeiro), sob a coordenação do professor José Antônio dos Santos Borges. A idéia de desenvolver tal sistema operacional evoluiu a partir do trabalho de um de seus alunos com deficiência visual. O Dosvox possui uma interface especializada que se comunica com o usuário (em Português) por meio de síntese de voz, viabilizando, desse modo, o uso de computadores por deficientes visuais.

O Dosvox disponibiliza um sistema completo, incluindo desde edição de textos, jogos, *browser* para navegação na Internet e utilitários. Na Figura 4.5 é mostrada a forma de acesso à Internet do sistema.

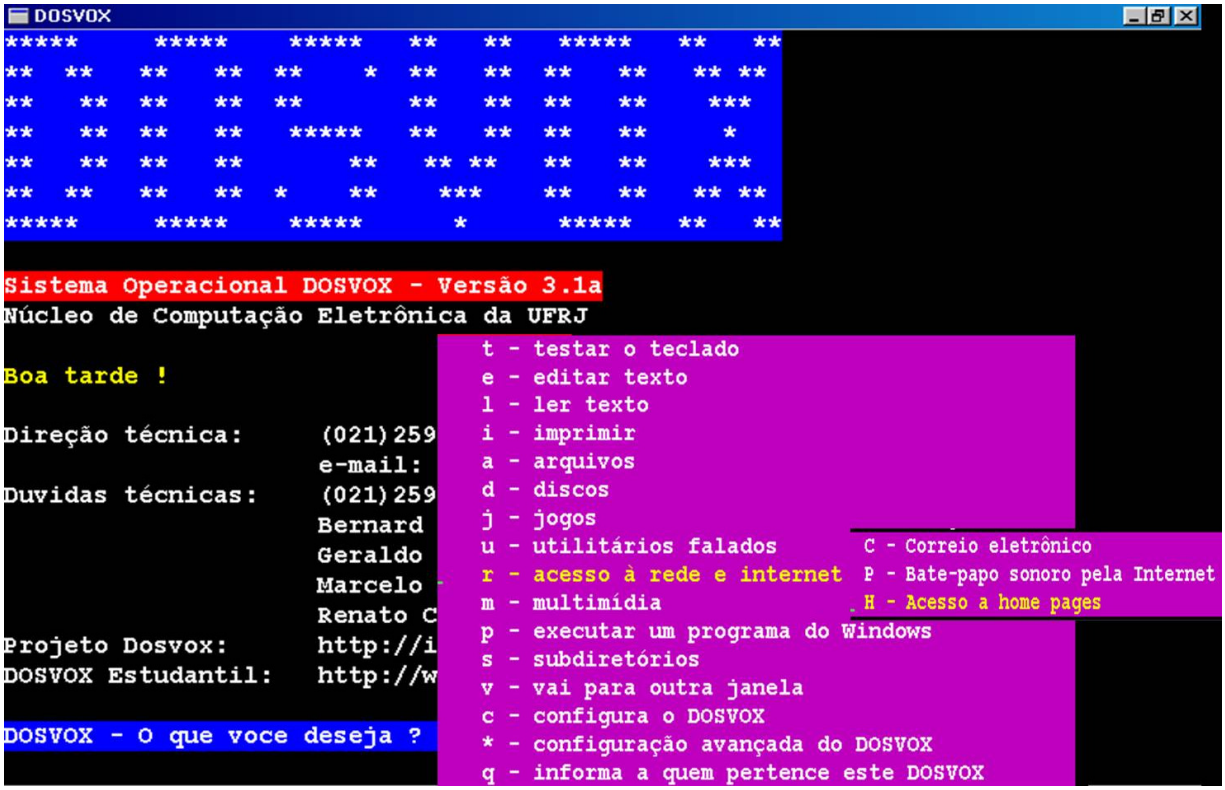

Figura 4.5 – Interface do Programa Dosvox

4.4. Leitores de Tela

Os Leitores de Tela são programas que interagem com o Sistema Operacional, reproduzindo, de forma sonora, os eventos ocorridos no computador. Essas interfaces *lêem* para o usuário as informações, botões, enfim, todos os eventos que se apresentam em forma de texto ou equivalente (imagens etiquetadas) na tela do computador. *Virtual Vision*[14] e *Jaws*[15] são os leitores de tela mais utilizados no Brasil.

Das tecnologias assistivas presentes do mercado, os leitores de tela são as mais usadas pelos deficientes visuais, pois, além de possibilitarem o uso dos computadores, eles possibilitam o acesso aos sistemas Web. Entretanto, este acesso pode ser comprometido caso as páginas Web não sejam produzidas de forma a possibilitar a leitura completa e correta destas ferramentas. Daí a importância dos desenvolvedores Web conhecerem a fundo as características dos principais leitores de tela.

[14] Virtual Vision – http://www.virtualvision.com.br/index.html
[15] Jaws – http://www.freedomscientific.com/products/fs/jaws-product-page.asp

4.4.1. Monitvox

O Monitvox[16] é um utilitário do sistema DOSVOX destinado a ajudar no acesso das pessoas com deficiência visual às janelas do sistema Windows. Ele não pretende ser um leitor de telas com qualidade profissional como o Virtual Vision ou similares. Entretanto, tem a habilidade de ler automaticamente a maior parte das informações usuais acessadas por deficientes visuais, e através de alguns truques, consegue acessar outras informações.

Como o Monitvox é um programa de código aberto, espera-se que a comunidade deficiente visual aproveite as informações contidas no programa para criar opções inovadoras de acessibilidade para programas específicos.

Ele é capaz de exibir na forma de voz as informações que são apresentadas na tela com diversas possibilidades:

- ler em síntese de voz o objeto selecionado (em foco);
- ler o objeto sobre o qual o mouse está passando;
- ler as informações completas sobre a janela do programa ativo.

O Monitvox permite também que se leia e (com algumas limitações) edite através das facilidades usuais do Dosvox diversos objetos, incluídos aí textos gerados nos editores mais populares. permite ainda um controle efetivo do mouse, dando ao usuário a opção de registrar e acessar pontos específicos da tela, saber o conteúdo desses pontos e clicar sobre eles, sem usar o mouse.

4.4.2. Virtual Vision

Desenvolvido pela MicroPower (empresa de Ribeirão Preto – SP). A primeira versão do Virtual Vision (Figura 4.6) foi lançada em janeiro de 1998. Pode ser adaptado em qualquer programa do Windows. É um leitor de telas que interage com o sistema operacional Windows e é capaz de informar aos usuários quais os controles (botão, lista, menu, etc) estão ativos em determinado momento.

[16] Monitvox (Manual de Operação) – http://intervox.nce.ufrj.br/dosvox/manuais/Monitvox.txt

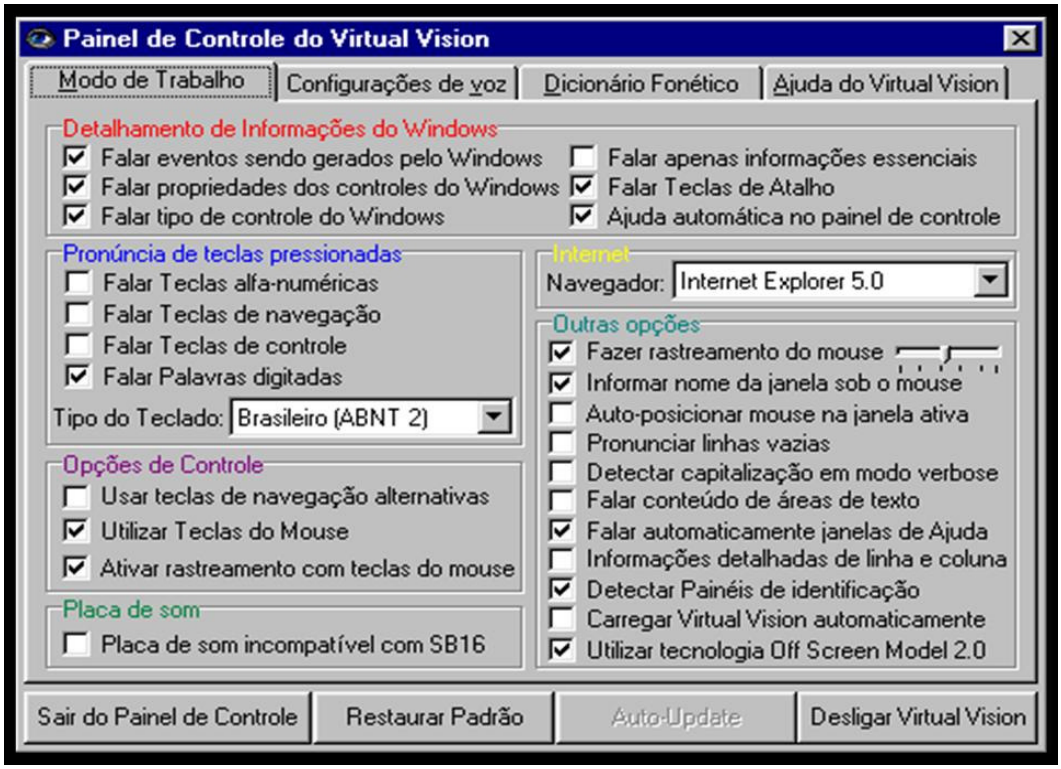

Figura 4.6 – Painel de Controle do Virtual Vision

4.5. Considerações Finais

Neste capítulo, detalhou-se algumas tecnologias assistivas tanto para deficientes visuais quanto para usuários com baixa visão. Esta arquitetura não foi testada exaustivamente em todas as tecnologias assistivas apresentadas neste trabalho. A realização de testes concentrou-se no sistema operacional Dosvox (na versão 4.4 que é a mais atual no momento em que foi produzido este trabalho) e no leitor de tela NVDA.

No próximo capítulo, serão abordados alguns conceitos introdutórios de uma Arquitetura Orientada a Serviços. Este tema é uma das principais bases teóricas da arquitetura proposta neste trabalho.

5. Arquitetura Orientada a Serviços

A Arquitetura Orientada a Serviços (SOA) é um paradigma de desenvolvimento de software que visa permitir que os componentes de um processo de negócio sejam integrados mais facilmente. Um componente de um processo de negócio é uma atividade comum, algo que é realizado frequentemente naquele processo de negócio específico. Assim, o objetivo é implementar um sistema que represente o negócio do cliente, dividindo este negócio em processos e estes em atividades. Algumas características importantes da arquitetura serão detalhadas nos tópicos a seguir.

5.1. Características

Em SOA cada atividade é implementada como um serviço, ou seja, um componente independente que poderá ser utilizado quantas vezes forem necessárias em partes diversas do sistema. Assim, vemos que SOA possui uma característica de ser, essencialmente, uma arquitetura baseada em componentes, onde cada componente preserva a sua independência e interage, apenas, através de interfaces bem definidas. Logo, podemos dizer que a arquitetura orientada a serviços promove o baixo acoplamento.

Outro ponto importante a se ressaltar é que não há limitações prévias em relação ao modo de implementar um serviço. Não há limitações em relação às tecnologias, linguagens ou plataformas a serem utilizadas, por exemplo. O desenvolvedor deve, apenas, especificar adequadamente o que seu serviço irá fazer e definir informações como tipos de dados de entrada e saída. Assim, outros desenvolvedores poderão fazer o uso adequado deste serviço baseado na interface que foi especificada e que será a única parte realmente visível do serviço. Então, os detalhes de implementação específicos de um componente (serviço) não são importantes para o restante do sistema e, por isso, não devem estar disponíveis. Isso significa que os serviços são encapsulados numa arquitetura orientada a serviços.

Para compor um serviço, um conjunto de regras e padrões devem ser seguidos. Tais padrões e regras foram desenvolvidos para especificar tudo que é importante para que um serviço torne-se acessível e utilizável através da Web. Sendo assim, vemos que a arquitetura propõe uma quase completa liberdade de desenvolvimento e, com isso, alcança-se a interoperabilidade, que é a possibilidade de sistemas coexistirem e comunicarem-se independente de fabricantes ou tecnologias. A interoperabilidade é uma característica muito importante, pois permite produzir soluções muito mais flexíveis e de melhor qualidade já que

não encontramos dificuldades de comunicação entre sistemas diversos, implementados de acordo com necessidades e limitações igualmente diversas.

5.2. Descrição dos Papéis em SOA

Um modelo inicial, inspirado no conjunto primário de padrões do WS, definiu SOA como uma arquitetura moldada sobre três componentes básicos: o consumidor de serviços, o provedor de serviços e o registro de serviços.

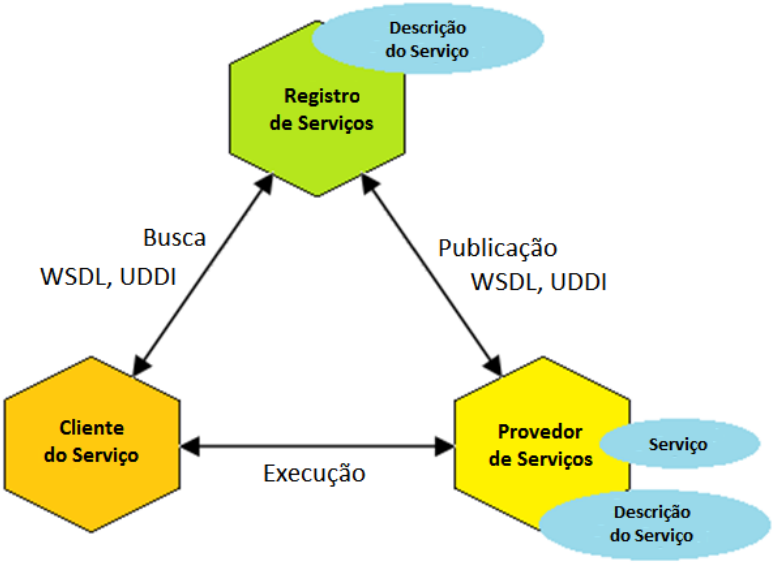

Figura 5.1 – Modelo primitivo de SOA. Fonte: [3]

Na Figura 5.1 é apresentada a dinâmica entre os participantes e o registro: o Provedor de Serviços dá visibilidade ao seu serviço, publicando-o no Registro de Serviços. Estando o serviço visível para o Consumidor de Serviços, poderá ser utilizado, ou consumido.

O Registro de Serviços é um diretório de rede que contém as descrições dos serviços acessíveis. É uma entidade que guarda os contratos dos Provedores de Serviços e os tornam disponíveis para consumidores. O contrato especifica a forma que um consumidor deve interagir com o provedor. Descrevendo o formato da requisição e da resposta do serviço.

5.3. Web Service

Com base na definição do Word Wide Web Consortium (W3C), *web services* são aplicações autocontidas, que possuem interface baseadas em XML (*Extensible Markup Language*) e que descrevem uma coleção de operações acessíveis através de rede, independentemente da tecnologia usada na implementação do serviço.

Os Web Services (WS) são elementos de software que não dependem de plataforma e cujas competências podem ser publicadas, descritas e invocadas via rede. Um WS pode ser acessado por diferentes sistemas através do uso de linguagens e protocolos padronizados, como XML (*Extensible Markup Language*) e HTTP (*Hyper Text Transfer Protocol*).

Para a implementação dos WS é necessário o conhecimento das seguintes tecnologias: WSDL (*Web Service Description Language*), SOAP (*Simple Object Access Protocol*), UDDI (*Universal Description, Discovery, and Integration*).

WSDL é a linguagem de descrição de WS baseada em XML. Ela permite, através da definição de um vocabulário em XML, a possibilidade de descrever serviços e a troca de mensagens. Mais especificamente é responsável por prover as informações necessárias para a invocação do WS, como sua localização, operações disponíveis e suas assinaturas.

SOAP é um protocolo para troca de informações em ambiente distribuído. É baseado em definições XML e utilizado para acessar WS. Esse protocolo encapsula as chamadas e retornos aos métodos dos WS, sendo utilizado, principalmente, sobre HTTP.

UDDI é uma das tecnologias que possibilitam o uso de WS. Uma implementação de UDDI corresponde a um registro *Web Service*, que provê um mecanismo para busca e publicação WS. Um registro UDDI contém informações categorizadas sobre os serviços e as funcionalidades que eles oferecem, e permite a associação desses serviços com suas informações técnicas, geralmente definidas usando WSDL. Como dito anteriormente, o arquivo de descrição em WSDL descreve as funcionalidades do WS, a forma de comunicação e sua localização.

Devido ao modo de acesso, um registro UDDI também pode ser entendido como um WS. A especificação UDDI define uma API baseada em mensagens SOAP, com uma descrição em WSDL do próprio WS do servidor de registro. A maioria dos servidores de registro UDDI também provê uma interface de navegação por browser.

A arquitetura de WS se baseia na interação de três entidades: Provedor de Serviços Cliente do Serviço e Registro de Serviços. De uma forma geral, as interações são para publicação, busca e execução de operações. A Figura 5.1 ilustra estas operações, os componentes envolvidos e suas interações.

O Provedor de Serviços representa a plataforma que hospeda WS permitindo que os clientes acessem o serviço. O Cliente do Serviço é a aplicação que está procurando, invocando ou iniciando uma interação com o WS. O cliente do serviço pode ser uma pessoa acessando através de um browser ou uma aplicação realizando uma invocação aos métodos descritos na interface do WS. O Registro de Serviços busca os WS baseados em arquivos de

descrição de serviços que foram publicados pelo Provedor de Serviço. O Cliente do Serviço busca por serviços no Registro de Serviços e recupera informações referentes a interface de comunicação para os WS durante a fase de desenvolvimento ou durante a execução do cliente, denominados *static binding* e *dinamic binding*, respectivamente.

5.4. Benefícios em adotar Web Services e SOA

5.4.1. Sistemas Legados

Para [24], muitos dos primeiros adeptos da tecnologia de Web Services concentraram seus esforços em interconectar sistemas legados (estabelecidos, seguros, operacionais) uns aos outros. Ainda afirmam que suas razões para isso são as seguintes:

- Riscos: tecnicamente, trabalhar em sistemas internos envolve menos riscos que trabalhar na implantação de novas entidades externas.
- A existência de um cliente para consumir um WS permite aos desenvolvedores o controle tanto do cliente quanto do serviço.
- O fato de soluções mais antigas serem problemáticas. A maioria dos novos projetos substitui programas mais antigos que foram escritos usando outros métodos. Por exemplo, há uma fragilidade nas transferências de arquivos simples. Outros projetos de desenvolvimento costumam ser caros de manter e aprimorar. "Por essa razão, eles são alvos óbvios para substituição" [24].

"Os WS envolvem perfeitamente os sistemas legados, independentemente da linguagem em que são escritos" [24]. Muitos desenvolvedores são contrários à idéia de trabalhar em antigos sistemas COBOL ou PL/1, por terem a idéia que sistemas antigos significam sistemas ruins. Gerencialmente, um sistema de software mais antigo que vem executando seu processamento perfeitamente, é mais confiável que qualquer sistema novo, por mais ultrapassada que seja a tecnologia utilizada no primeiro. Os clientes pagam por produtos que sejam entregues corretamente e, por esse motivo, a tecnologia utilizada é irrelevante.

"A idéia de escrever, testar e implementar um novo sistema para substituir um que já funciona bem é difícil de ser vendida para a gerência de uma empresa" [3]. O motivo mais simples é o questionamento de por que pagar para substituir um sistema estável, por outro que certamente apresentará problemas e necessidade de constantes ajustes por certo tempo, que pode durar meses ou mesmo anos.

A vantagem que o mercado vê nos WS é que eles são escritos de uma maneira que exige pouca ou nenhuma mudança na base do código legado, resultando num moderno sistema distribuído que conserva toda a equidade que a empresa construiu em sua base de código.

5.4.2. Uso de padrões XML

Na arquitetura proposta neste trabalho, as trocas de mensagens são baseadas no padrão de linguagem de marcação *Extensible Markup Language* (XML) e os arquivos de configuração das políticas de segurança também a usam este padrão.

Sendo uma linguagem de marcação de dados especificada pela W3C, o padrão XML é empregado em comunicações dos WS. A linguagem XML descreve os objetos, atributos, métodos e os parâmetros de maneira que as informações sejam interpretadas pelos softwares. Na diversidade dos modos de uso da XML podemos citar: documentação, desenvolvimento de banco de dados e desenvolvimento *Web*. Os principais objetivos da linguagem XML são:

- Prover um padrão para a troca de documentos através da web independente de sistemas operacionais;
- Possuir uma linguagem em alto nível para entendimento humano, simplificando a análise de documentos XML por programadores, facilitando a manutenção das aplicações;
- Separar conteúdo e formatação;
- Criação de *tags* com a possibilidade de tamanhos dinâmicos;
- Fornecer uma especificação formal para marcação de documentos;

Nos padrões segundo a W3C, um documento XML formatado possui um elemento raiz e um prólogo[17]. O prólogo contém metadados correspondentes ao resto do documento e é composto da declaração XML, das instruções de processamento das definições de XML *Schema*[18] ou DTD[19].

5.5. SOAP

SOAP é um protocolo leve destinado à troca de informações estruturadas em um ambiente distribuído e descentralizado. Ele utiliza tecnologias XML para definir uma estrutura de mensagens extensíveis proporcionando uma construção de mensagem que pode

[17] O prólogo é definido no início de um documento XML. Um prólogo pode conter uma declaração XML, comentários, instruções de processamento, e declarações de tipo de documento (DTD).

[18] XML *Schema* expressam vocabulários compartilhados e permitem que as máquinas executem regras feitas por pessoas. Eles fornecem uma forma de definir a estrutura, conteúdo e semântica de documentos XML em mais detalhes.

[19] Um *Document Type Definition* (DTD) define os blocos de construção legal de um documento XML. Ele define a estrutura do documento com uma lista de elementos e atributos.

ser trocados por uma variedade de protocolos subjacentes. SOAP foi projetado para ser independente de qualquer modelo de programação específica [25].

Como o SOAP é uma tecnologia independente de plataforma e linguagem de programação, ele é amplamente usado em WS. O SOAP foi, inicialmente, um protocolo definido para fornecer o recurso de invocação remota por meio da Internet. Esta invocação se dá por meio de XML-RPC[20] ou troca de mensagens. Desta forma, o SOAP pretende garantir a interoperabilidade e intercomunicação entre diferentes sistemas, através da utilização de uma linguagem (XML) e mecanismo de transporte (HTTP) padrões.

5.6. Considerações Finais

Neste capítulo, foram abordados alguns conceitos de SOA e SOAP, bem como os benefícios de se utilizar Web Services. Conforme exposto no sub-tópico 5.4.1, uma Arquitetura Orientada a Serviços não exige que um sistema que já esteja consolidado em produção seja refeito. Será visto no próximo capítulo, onde finalmente será apresentada a arquitetura proposta, que foi feito um sistema simples de cadastro sem o uso de Web Services e ele não foi comprometido ao ser inserido numa Arquitetura Orientada a Serviços. Simplesmente, novos módulos independentes foram inseridos.

[20] É uma Chamada de Procedimento Remoto usando HTTP como transporte e XML como a codificação.

6. Projeto

6.1. Descrição Geral

É necessário pensar em uma forma produtiva de desenvolvimento Web de modo a atender às necessidades tanto dos usuários com deficiência quanto dos usuários sem deficiência. Para isso, deve-se conceber uma arquitetura flexível para o desenvolvimento de sistemas Web acessíveis, sendo este o principal objetivo deste trabalho. Através de uma arquitetura expansível e projetada para considerar diferentes tipos de interface, o sistema poderá se ajustar melhor às necessidades dos usuários, fazendo com que tanto usuários não deficientes quanto usuários deficientes possam manipular todas as funcionalidades do sistema, satisfazendo, assim, os critérios de navegabilidade e usabilidade.

Entretanto, viu-se nos capítulos anteriores que os critérios de navegabilidade e usabilidade para os deficientes visuais em particular, são completamente diferentes dos critérios para usuários que não possuem deficiência. Por conta disto, é realmente necessária uma codificação diferente resultando na apresentação do conteúdo com o uso de diferentes interfaces. Esta codificação diferenciada torna a etapa de implementação do projeto de sistema Web mais complexa, aumentando o custo e o tamanho do projeto e representando a principal dificuldade para a disseminação da prática do desenvolvimento de aplicações acessíveis.

Para avaliar a arquitetura proposta, foi desenvolvido um sistema simples de cadastro de alunos com deficiência, visando a sua expansão e utilização pela Secretaria de Acessibilidade UFC Inclui[21]. Este sistema está disponível no servidor do DETI e foi projetado para atender apenas aos usuários não deficientes e deficientes visuais. Suas funcionalidades são listadas abaixo:

- Inserir Alunos;
- Alterar Alunos;
- Visualizar Alunos;
- Deletar Alunos.

Estas funcionalidades são executadas e apresentadas de acordo com as necessidades do usuário. No que tange às necessidades dos usuários deficientes visuais, este sistema visa

[21] http://www.ufc.br/portal/index.php?option=com_content&task=view&id=10726&Itemid=71

satisfazer os critérios de acessibilidade de Nível AAA definidos pela WAI-W3C (critérios estes que foram expostos nos Apêndices 1, 2 e 3 deste trabalho).

Vale salientar que este trabalho visa atender apenas às necessidades dos usuários deficientes visuais, o que não significa que este sistema não seja acessível para usuários com deficiência motora ou auditiva. A limitação restringe apenas as situações que exigem um tratamento mais específico para outras deficiências, como a surdez, que poderia exigir, por exemplo, apresentação de descrição em símbolos ou LIBRAS. Contudo, a arquitetura proposta permite a expansão dos tipos de interface através de módulos independentes para atender às outras necessidades dos usuários.

6.2. Ferramentas de Desenvolvimento

Para o desenvolvimento da solução foi usado uma série de ferramentas que foram escolhidas sempre visando o aumento da flexibilidade da arquitetura. Todas as ferramentas são gratuitas, facilmente configuradas e de fácil acesso na Internet.

A linguagem de programação escolhida para o desenvolvimento foi o PHP. O motivo da escolha é pela difusão no meio acadêmico, pela rapidez e facilidade no uso de funcionalidades que, para outras linguagens, seriam um tanto quanto complexas (como acesso a Web Services, manipulações de variáveis de sessão, manipulações de cookies, etc).

Entretanto, a arquitetura pode ser expandida para agregar subsistemas em outras linguagens perfeitamente. Isto porque toda a comunicação entre os subsistemas é feita por meio de Web Services. Para implementar e consumir Web Services em PHP, foi usado um framework da WSO2 chamado WSF/PHP[22]. Foi escolhido este framework por ser gratuito, dar suporte à segurança da informação e a mensagens de erro, geração automática de WSDL[23] e pela facilidade de implementação e consumo de Web Services em PHP.

6.3. Arquitetura Proposta

A arquitetura da solução proposta visa o desenvolvimento desacoplado para cada necessidade do usuário, de modo que o atendimento de uma necessidade específica não comprometa o funcionamento de outras.

Independente do tipo de usuário, o sistema possui uma única página de acesso. Esta página (chamada index.php) atende aos padrões de acessibilidade definidos pela W3C

[22] Baixado em http://wso2.org/downloads/wsf/php

[23] Formato XML usado para descrever serviços de rede como um conjunto de terminais que operam em mensagens contendo um ou outro documento ou procedimento orientado a informações (http://www.w3.org/TR/wsdl).

(expostos nos apêndices 1, 2 e 3) acessível independente de qualquer necessidade do usuário e nela ele define a forma de apresentação do conteúdo do sistema. Uma vez escolhida a forma de apresentação, o sistema grava um *cookie*[24] na máquina do usuário para que o sistema futuramente possa ser sempre redirecionado para o subsistema que foi desenvolvido priorizando atender apenas à necessidade escolhida.

Uma vez escolhida a necessidade a ser atendida, o usuário acessa subsistemas que foram programados a despeito das demais necessidades que o sistema como um todo atende. Cada subsistema possui suas telas personalizadas de acordo com a necessidade a ser atendida. Estas telas são construídas se comunicando por meio de Web Services que são implementados em outro sistema servidor.

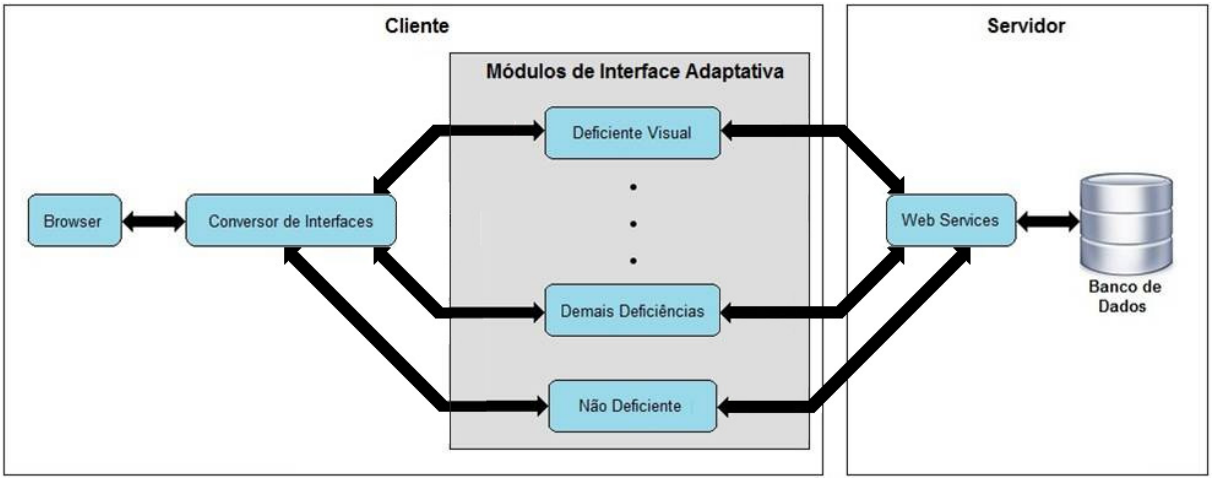

Figura 6.1 – Diagrama de Fluxo da Arquitetura Proposta sem Sistema Legado

Na figura acima, percebe-se que todas as interfaces são desenvolvidas dentro do Módulo de Interface adaptativa, que deve ser implementado seguindo a filosofia de uma Arquitetura Orientada a Serviços. Neste caso, o módulo de interface para não deficientes, assim como os demais, precisa consumir os serviços disponibilizados no Web Service para acessar os conteúdos relevantes para o sistema. Este cenário é o mais simples e, por ser totalmente desenvolvido na Arquitetura Orientada a Serviços, é o de mais fácil manutenção.

Entretanto, numa configuração de ambiente onde já existe um sistema consolidado e que retirá-lo irá gerar um custo elevado (os chamados sistemas legados), são necessários apenas dois ajustes para que a arquitetura proposta possa ser implantada. O módulo para não deficientes deve ser colocado "em paralelo" com os módulos orientados a serviço, tornando,

[24] Um cookie é apenas uma das partes de uma informação armazenada como cadeia de texto em sua máquina. Um servidor da Web envia um cookie para você e seu navegador o armazena. O navegador retorna o cookie para o servidor da próxima vez que a página for consultada (http://informatica.hsw.uol.com.br/dentro-do-cookie.htm).

assim, a arquitetura proposta em uma parte orientada a serviços e uma parte não orientada a serviços. Por fim, é necessário que se elenque todas as funcionalidades do sistema legado e as codifiquem no paradigma da orientação a serviços. Assim os demais módulos poderão consumir os Web Services e o sistema legado irá funcionar a despeito dos módulos implementados.

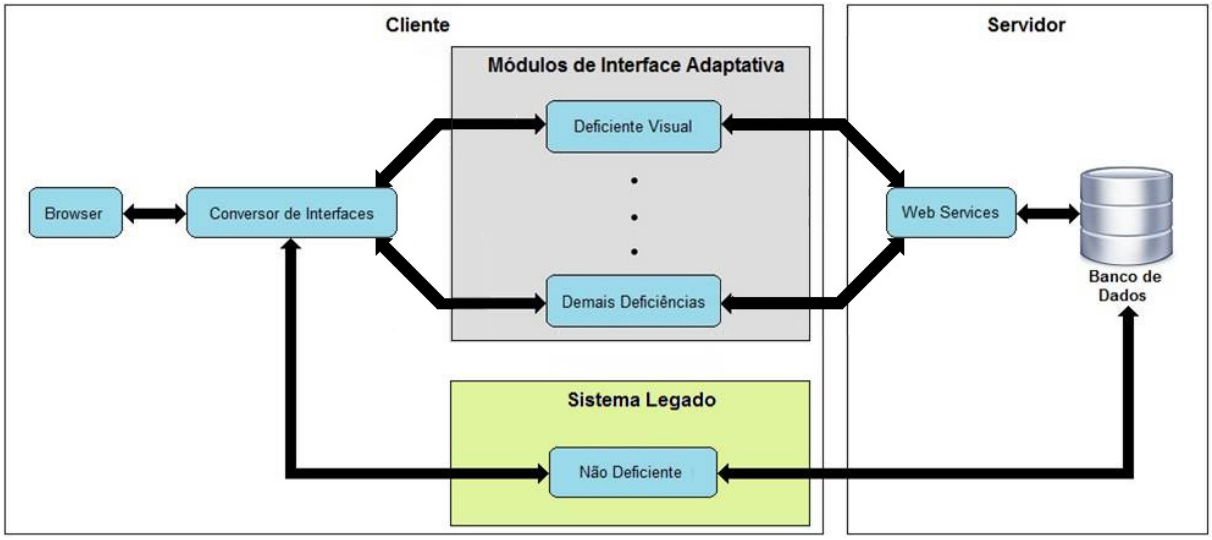

Figura 6.2 – Diagrama de Fluxo da Arquitetura Proposta mantendo o Sistema Legado

Percebe-se na Figura 6.2, que a arquitetura proposta se dividiu, no lado cliente, em um bloco orientado a serviços, que são os módulos de interface adaptativa, e outro bloco não orientado a serviços, que seria o sistema legado. Vale salientar que o sistema legado não necessariamente se comunica diretamente com o banco de dados; pode existir um sistema servidor que realiza algum processamento antes ou depois de realizar alguma operação no banco de dados. Conforme foi visto no Capítulo 5, onde foram abordados os benefícios de se ter um sistema totalmente orientado a serviços, pode-se dizer que este cenário é o mais complexo e de mais difícil manutenção.

6.4. Considerações Finais

Neste capítulo foi apresentada a arquitetura concebida. É possível que existam ambientes onde implantar esta arquitetura represente um alto custo de projeto. Porém, como a arquitetura utiliza apenas tecnologias *open source*[25], em sistemas Web desenvolvidos de

[25] Definição – http://www.free-soft.org/mirrors/www.opensource.org/docs/osd-portuguese.html

acordo com padrões de projeto bem aceitos no mercado, esta arquitetura não exigirá muito esforço para ser implantada.

Foram analisados dois cenários onde a arquitetura poderia ser implantada (Figuras 6.1 e 6.2). Como o mais comum, bem como o mais complexo, é o cenário onde já existe um sistema legado (Figura 6.2), o mesmo foi adotado na implementação do sistema de cadastro deste trabalho.

No próximo capítulo, será feito um estudo de caso da arquitetura proposta. A arquitetura será inserida num ambiente com um sistema legado (Sistema de Cadastro de Alunos).

7. Estudo de Caso

Com base na arquitetura proposta no capítulo anterior, foi desenvolvido um sistema simples de Cadastro de Alunos com Deficiência. Este sistema foi codificado usando as ferramentas descritas no capítulo anterior.

Conforme exposto anteriormente, a arquitetura permite que exista um sistema legado que não foi implementado numa Arquitetura Orientada a Serviços. Sabendo que este é o cenário mais crítico e mais comum, foi desenvolvido um Sistema de Cadastro de Deficientes sem a preocupação de que este sistema atenda aos critérios de acessibilidade. Na Figura 7.1 a tela principal do sistema legado, bem como o resultado da leitura de tela apresentada pelo Dosvox.

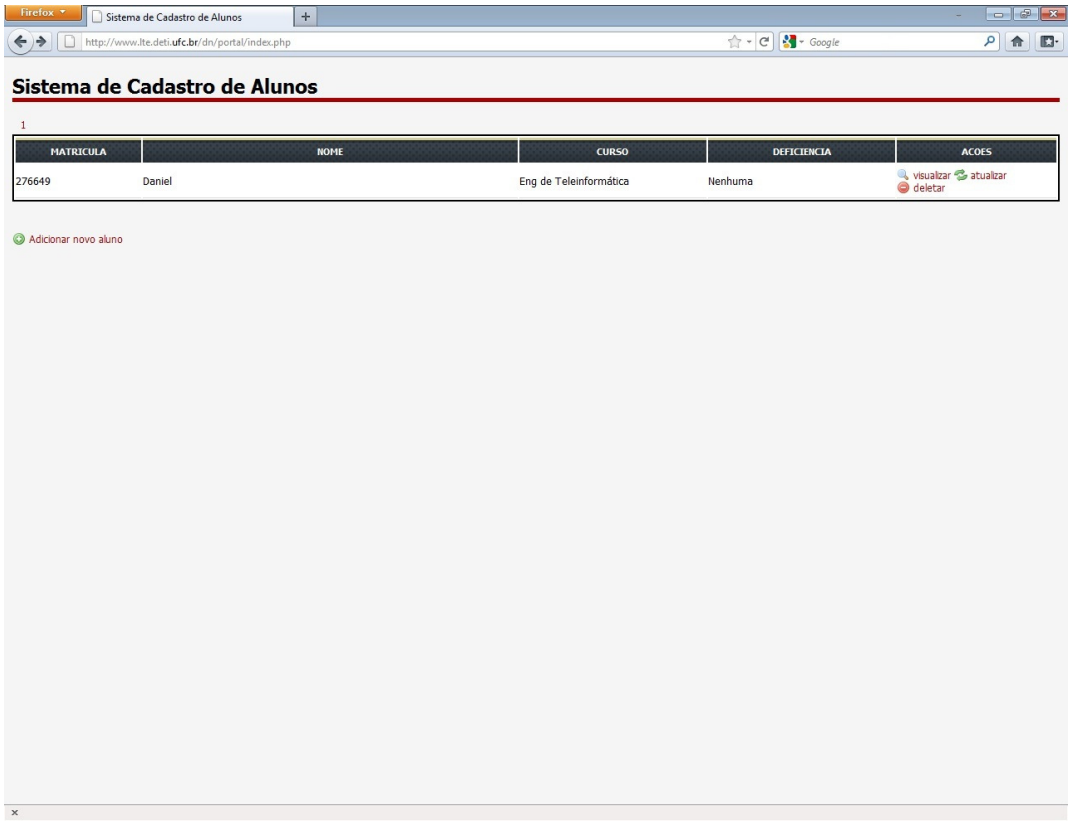

Figura 7.1 – Tela Inicial (Sistema Legado)

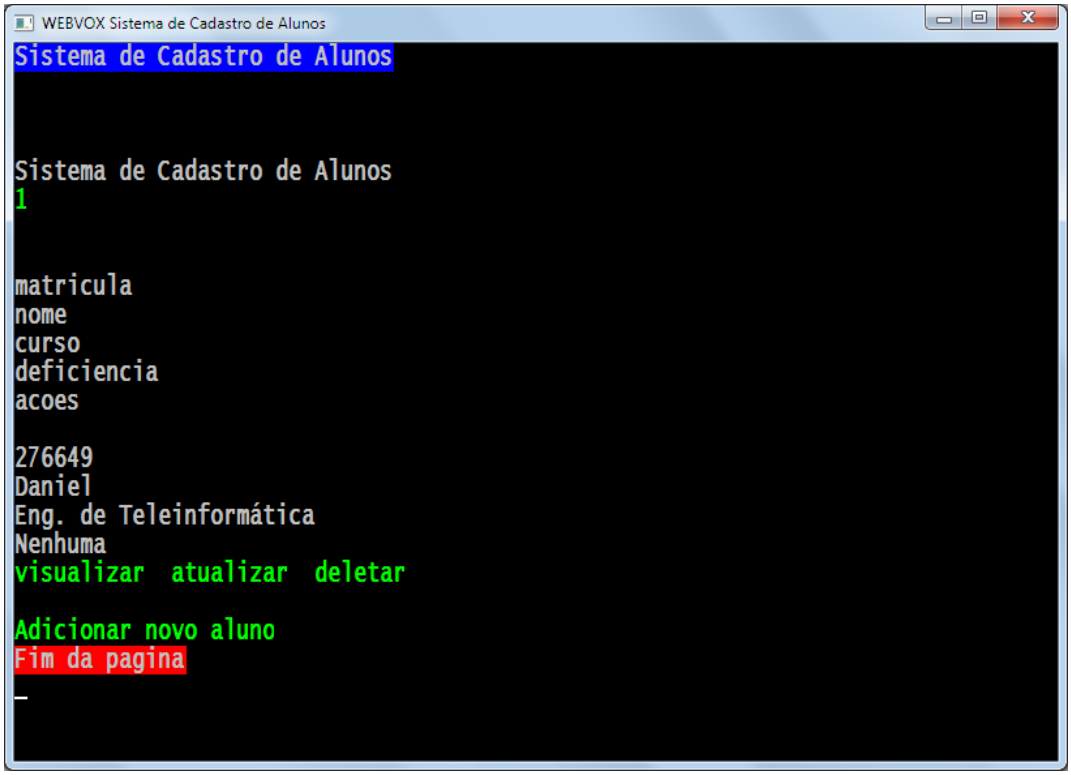

Figura 7.2 – Tela de Inicial (Sistema Legado) Lida pelo Dosvox

Porém, percebe-se pela Figura 7.2 que, apesar de o sistema ser funcional para usuários não deficientes, a tela principal de manipulação do Sistema de Cadastro de Alunos com Deficiência não é acessível. Isto porque ela não foi lida corretamente por um leitor de tela.

Entretanto, é possível identificar o que está tornando esta interface inacessível. Conforme visto no tópico 2.3, este sistema apresenta duas das principais dificuldades apresentadas pelos deficientes visuais no acesso Web, pois tornam sua leitura complexa para um leitor de tela, são elas:

- Uso de tabelas que não fazem sentido se lidas célula por célula;
- Uso de paginação sem descrição textual (apenas numérica);

Como não será feita nenhuma alteração no sistema legado, é necessário desenvolver uma interface adaptativa para deficientes visuais de modo que o Sistema de Cadastro de Alunos com Deficiência possa ser lido corretamente pelos leitores de tela.

7.1. Identificação dos Serviços

Sabendo que a arquitetura proposta exige que os módulos acessíveis sejam implementados seguindo a filosofia da Arquitetura Orientada a Serviços e sabendo da necessidade de desenvolver um módulo para tornar possível do uso do Sistema de Cadastro de Alunos com Deficiência pelos usuários deficientes visuais, é preciso, primeiramente, identificar os serviços necessários para o funcionamento do sistema. Estes serviços serão gerenciados por um sistema servidor que se comunicará com a base de dados.

De acordo com as funcionalidades expostas no Tópico 6.1, foram listados alguns serviços que serão detalhados nos tópicos a seguir.

7.1.1. Serviço de Listagem de Funcionalidades

O sistema deve ser desenvolvido de modo que o servidor possa ser expandido e agregar mais funcionalidades, para isso foi criado um serviço chamado **index** que, basicamente, elenca todas as funcionalidades do sistema. Este serviço será consumido, do lado cliente, por uma tela que antecederá a tela principal de manipulação do sistema. O efeito disto para o usuário é que o leitor de tela irá apresentar a ele todas as funcionalidades que o sistema em questão possui.

Para consumir este serviço, é necessário que o cliente envie um código em XML com a seguinte estrutura:

```
<index></index>
```

Desta forma, o servidor irá retornar um código, também em XML, com a estrutura do exemplo a seguir:

```
<indexResponse>
  <title>Sistema de Cadastro de Alunos</title>
  <table>alunos</table>
  <description>
    <function>Inserir alunos</function>
  </description>
  <description>
    <function>Atualizar alunos</function>
  </description>
  <description>
    <function>Visualizar alunos</function>
  </description>
  <description>
    <function>Deletar alunos</function>
  </description>
</indexResponse>
```

Desta forma, o servidor pode expandir suas funcionalidades sem a necessidade de ajustes no lado cliente.

Após a implementação deste serviço, o framework WSF/PHP (exposto no Tópico 6.2) automaticamente gerou o XML de descritor do serviço **index** (vide anexos).

7.1.2. Serviço de Listagem de Alunos

Uma das funcionalidades da tela principal do sistema é a de apresentar uma determinada quantidade de alunos usando o recurso de paginação. É preciso ajustar este recurso de modo que seja possível sua leitura usando um leitor de tela.

A entidade de dados deste sistema é uma tabela chamada Alunos. Atualmente esta tabela possui os seguintes campos:

- Matrícula;
- Nome;
- Curso;
- Deficiência;

Com base nestas informações, foi criado um serviço chamado **all** que se encarrega de listar os alunos limitados por uma página passada na requisição. Para consumir este serviço, basta que o cliente envie um XML com a seguinte estrutura:

```
<all>
  <table>alunos</table>
  <page>
    <pageNum>0</pageNum>
    <pageSize>5</pageSize>
  </page>
</all>
```

No XML de requisição acima, o cliente está solicitando o serviço **all** passando como parâmetros a tabela (alunos), o número da página (neste exemplo o cliente está requisitando a primeira página) e a quantidade de elementos de cada página (no caso, cada página contém 5 alunos).

Desta forma, o servidor irá retornar um XML com a seguinte estrutura, por exemplo:

```
<allResponse>
  <table name="alunos">
    <record>
      <field>
        <fieldName>id</fieldName>
        <fieldValue>1</fieldValue>
```

```
      </field>
      <field>
        <fieldName>matricula</fieldName>
        <fieldValue>276649</fieldValue>
      </field>
      <field>
        <fieldName>nome</fieldName>
        <fieldValue>Daniel</fieldValue>
      </field>
      <field>
        <fieldName>curso</fieldName>
        <fieldValue>Engenharia de Teleinformática</fieldValue>
      </field>
      <field>
        <fieldName>deficiencia</fieldName>
        <fieldValue>Nenhuma</fieldValue>
      </field>
    </record>
    <record>
      <field>
        <fieldName>id</fieldName>
        <fieldValue>7</fieldValue>
      </field>
      <field>
        <fieldName>matricula</fieldName>
        <fieldValue>679322</fieldValue>
      </field>
      <field>
        <fieldName>nome</fieldName>
        <fieldValue>Rebeca</fieldValue>
      </field>
      <field>
        <fieldName>curso</fieldName>
        <fieldValue>Administração</fieldValue>
      </field>
      <field>
        <fieldName>deficiencia</fieldName>
        <fieldValue>Visual</fieldValue>
      </field>
    </record>
  </table>
</allResponse>
```

Pode-se perceber que o XML de resposta foi estruturado de modo a suportar expansões tanto de o sistema manipular diversas tabelas quanto de aumentar a quantidade de campos de cada tabela.

Após a implementação deste serviço, o framework WSF/PHP (exposto no Tópico 6.2) automaticamente gerou o XML de descritor do serviço **all** (vide anexos).

7.1.3. Serviço de Inclusão de Alunos

Para o desenvolvimento da interface de inclusão de alunos, é necessário que se gere um formulário que respeite os critérios de acessibilidade definidos pela WAI-W3C (vide

anexos). Porém, como o cliente não possui acesso direto à base de dados (como no sistema legado anteriormente implementado), então o sistema cliente não conhece os campos nem as tabelas que estão sendo manipuladas. Desta forma, não basta criar um serviço que realize a inclusão do aluno, deve-se criar um serviço auxiliar para a renderização do formulário de inclusão.

7.1.3.1.Serviço de Questionamento de Campos de Tabela

Este serviço auxiliar consiste em "perguntar" ao servidor quais são os campos das entidades envolvidas no sistema para, em seguida, construir dinamicamente o formulário de inclusão de alunos. A este serviço foi dado o nome de **addQuestion** e para consumi-lo, basta que o cliente envie um XML com a seguinte estrutura:

```
<addQuestion>
      <table>alunos</table>
</addQuestion>
```

Desta forma, o servidor irá retornar um XML com a seguinte estrutura, por exemplo:

```
<addQuestionResponse>
  <table name="alunos">
    <field>
      <fieldName>matricula</fieldName>
      <fieldType>integer</fieldType>
      <fieldSize>10</fieldSize>
    </field>
    <field>
      <fieldName>nome</fieldName>
      <fieldType>string</fieldType>
      <fieldSize>50</fieldSize>
    </field>
    <field>
      <fieldName>curso</fieldName>
      <fieldType>string</fieldType>
      <fieldSize>50</fieldSize>
    </field>
    <field>
      <fieldName>deficiencia</fieldName>
      <fieldType>string</fieldType>
      <fieldSize>50</fieldSize>
    </field>
  </table>
</addQuestionResponse>
```

Analisando o XML de resposta, percebe-se que ele foi estruturado para carregar quaisquer informações adicionais sobre os campos de qualquer entidade além de nome, tipo e tamanho do campo. Além disso, esta estrutura de XML permite que, futuramente, este mesmo serviço envie informações de mais de uma entidade de dados.

7.1.3.2.Serviço de Solicitação de Inclusão de Alunos

Neste serviço, chamado na implementação em questão de **add**, é feita a solicitação de inclusão de aluno. Como a solicitação pode ser atendida ou não (em caso de a base de dados estar em manutenção, por exemplo), é necessário especificar uma estrutura de retorno para que o sistema possa "entender" se a solicitação foi atendida ou não.

Como existem dois tipos de respostas, será exemplificado um caso de sucesso na inclusão. Para consumir este serviço corretamente, basta que o cliente envie um XML com a seguinte estrutura, por exemplo:

```
<add>
  <table name="alunos">
    <field>
      <fieldName>matricula</fieldName>
      <fieldValue>276649</fieldValue>
    </field>
    <field>
      <fieldName>nome</fieldName>
      <fieldValue>Daniel</fieldValue>
    </field>
    <field>
      <fieldName>curso</fieldName>
      <fieldValue>Eng. Teleinformática</fieldValue>
    </field>
    <field>
      <fieldName>deficiencia</fieldName>
      <fieldValue>Nenhuma</fieldValue>
    </field>
  </table>
</add>
```

Neste caso, o servidor irá retornar um XML com a seguinte estrutura:

```
<addResponse>
  <msg type="success">O aluno foi inserido.</msg>
</addResponse>
```

Percebe-se que existe um atributo identificando que trata-se de uma mensagem de sucesso. A descrição da mensagem de retorno fica a cargo da forma de implementação no lado servidor. No caso desta implementação foi dito apenas que o aluno foi inserido.

Foi visto no tópico anterior que o serviço **addQuestion** traz informações que podem ser usadas pelo desenvolvedor do sistema cliente nas validações de formulário. Porém, não é exigido que todas as validações sejam feitas no lado cliente. Logo, o servidor deve possuir uma forma de enviar alguma resposta ao cliente em caso de algum erro. Caso o serviço seja consumido erroneamente, o servidor irá retornar um XML com a seguinte estrutura:

```
<addResponse>
  <msg type="error">Não foi possível inserir o aluno.</msg>
</addResponse>
```

Similar à mensagem de sucesso, percebe-se que existe um atributo identificando que trata-se de uma mensagem de erro. Novamente, a descrição da mensagem de retorno fica a cargo da forma de implementação no lado servidor. No caso desta implementação foi dito apenas que o aluno não foi inserido.

Após a implementação destes dois serviços, o framework WSF/PHP (exposto no Tópico 6.2) automaticamente gerou o XML de descritor de cada serviço (vide anexos).

7.1.4. Serviço de Consulta de Aluno

Este serviço, chamado de **view**, faz-se necessário quando o usuário desejar ver maiores informações sobre o aluno. No caso desta implementação em questão, onde foram definidos poucos campos para o aluno, este serviço não será muito necessário, pois trará poucas informações. Entretanto, quando este sistema for expandido, utilizando, por exemplo, mais de uma tabela para compor as informações do aluno, este serviço será cada vez mais necessário.

Para consumir este serviço, basta que o cliente envie um XML com a seguinte estrutura, por exemplo:

```
<view>
  <table name="alunos">
    <field>
      <fieldName>matricula</fieldName>
      <fieldValue>276649</fieldValue>
    </field>
  </table>
</view>
```

Percebe-se que a estrutura do XML de requisição foi definida de tal forma que o cliente pode consultar por qualquer campo (mesmo que o mesmo não seja chave primária da tabela). Desta forma, o servidor irá retornar um XML com a seguinte estrutura, por exemplo:

```
<viewResponse>
  <table name="alunos">
    <field>
      <fieldName>matricula</fieldName>
      <fieldValue>276649</fieldValue>
    </field>
    <field>
      <fieldName>nome</fieldName>
      <fieldValue>Daniel</fieldValue>
    </field>
    <field>
      <fieldName>curso</fieldName>
      <fieldValue>Eng. Teleinformática</fieldValue>
    </field>
    <field>
      <fieldName>deficiencia</fieldName>
      <fieldValue>Nenhuma</fieldValue>
    </field>
  </table>
```

```
</viewResponse>
```

Vale salientar que o sistema foi implementado de forma que esta requisição é proveniente de um link para maiores informações de um aluno já existente. Logo, não foi necessário criar um serviço de verificação de existência do registro na base de dados.

Após a implementação deste serviço, o framework WSF/PHP (exposto no Tópico 6.2) automaticamente gerou o XML de descritor do serviço **view** (vide anexos).

7.1.5. Serviço de Atualização de Alunos

Analogamente ao que fora exposto no início do sub-tópico 7.1.3, antes de consumir o serviço que de fato realiza a atualização dos dados do aluno, é necessário um serviço que torne possível a renderização de um formulário já contendo os dados do aluno.

7.1.5.1. Serviço de Questionamento de Campos de Tabela com Dados de um Registro

Este serviço, chamado na implementação em questão de **updateQuestion**, possui uma resposta bastante similar à resposta do serviço implementado no sub-tópico 7.1.3.1, porém, internamente sua implementação (além de seu propósito) é totalmente diferente. Logo, pela filosofia da Arquitetura Orientada a Serviços, faz-se necessário a implementação deste serviço, uma vez que seu propósito é diferente.

Para que o cliente possa consumir este serviço, basta que ele envie um XML com a seguinte estrutura, por exemplo:

```
<updateQuestion>
  <table name="alunos">
    <field>
      <fieldName>matricula</fieldName>
      <fieldValue>276649</fieldValue>
    </field>
  </table>
</updateQuestion>
```

Desta forma, o servidor irá retornar um XML com a seguinte estrutura, por exemplo:

```
<updateQuestionResponse>
  <table name="alunos">
    <field>
      <fieldName>matricula</fieldName>
      <fieldValue>276649</fieldValue>
      <fieldType>integer</fieldType>
      <fieldSize>10</fieldSize>
    </field>
    <field>
      <fieldName>nome</fieldName>
      <fieldValue>Daniel</fieldValue>
      <fieldType>string</fieldType>
      <fieldSize>50</fieldSize>
```

```
    </field>
    <field>
      <fieldName>curso</fieldName>
      <fieldValue>Eng. de Teleinformática</fieldValue>
      <fieldType>string</fieldType>
      <fieldSize>50</fieldSize>
    </field>
    <field>
      <fieldName>deficiencia</fieldName>
      <fieldValue>Nenhuma</fieldValue>
      <fieldType>string</fieldType>
      <fieldSize>50</fieldSize>
    </field>
  </table>
</updateQuestionResponse>
```

Analisando a estrutura do XML de resposta, percebe-se a semelhança com o serviço **addQuestion**, porém com a diferença que o serviço **updateQuestion** traz os dados de um registro específico passado na requisição.

7.1.5.2.Serviço de Solicitação de Atualização

Neste serviço, chamado na implementação em questão de **update**, é feita a solicitação de atualização do aluno. Analogamente ao serviço de inclusão, a solicitação pode ser atendida ou não, tornando necessária a especificação uma estrutura de retorno que faça com que o sistema possa "entender" se a solicitação foi atendida ou não.

Como a estrutura do XML de resposta é semelhante a do serviço de solicitação de inclusão, não é necessário detalhar situações de falha ou sucesso na atualização dos dados. Logo, em casos de sucesso, o servidor retorna um XML com a seguinte estrutura:

```
<updateResponse>
  <msg type="success">O aluno foi atualizado.</msg>
</updateResponse>
```

E em casos de erro na atualização, o servidor retorna um XML com a seguinte estrutura:

```
<updateResponse>
  <msg type="error">Não foi possível atualizar o aluno.</msg>
</updateResponse>
```

Após a implementação destes dois serviços, o framework WSF/PHP (exposto no Tópico 6.2) automaticamente gerou o XML de descritor de cada serviço (vide anexos).

7.1.6. Serviço de Exclusão de Alunos

Este serviço, chamado de **delete**, faz-se necessário quando o usuário desejar deletar um aluno. Entretanto, assim como nos serviços de inclusão e atualização, a exclusão do aluno pode ser efetuada ou não pelos mesmos motivos expostos nos seus respectivos tópicos.

Deste modo, para consumir corretamente este serviço, basta que o cliente envie um XML com a seguinte estrutura, por exemplo:

```
<delete>
  <table name="alunos">
    <field>
      <fieldName>matricula</fieldName>
      <fieldValue>276649</fieldValue>
    </field>
  </table>
</delete>
```

Percebe-se que a estrutura do XML de requisição foi definida de tal forma que o cliente pode deletar por qualquer campo (mesmo que o mesmo não seja chave primária da tabela).

Como a estrutura do XML de resposta é semelhante a dos serviços de solicitação de inclusão e atualização, não é necessário detalhar situações de falha ou sucesso na exclusão dos dados. Logo, em casos de sucesso, o servidor retorna um XML com a seguinte estrutura:

```
<updateResponse>
  <msg type="success">O aluno foi deletado.</msg>
</updateResponse>
```

E em casos de erro na exclusão, o servidor retorna um XML com a seguinte estrutura:

```
<updateResponse>
  <msg type="error">Não foi possível deletar o aluno.</msg>
</updateResponse>
```

Após a implementação deste serviço, o framework WSF/PHP (exposto no tópico 6.2) automaticamente gerou o XML de descritor do serviço **delete** (vide anexos).

7.2. Estrutura do Projeto

Uma vez identificado e detalhado todos os serviços necessário para o funcionamento do Sistema de Cadastro de Alunos com Deficiência, foi feita a codificação dos serviços (lado servidor) bem como a codificação do módulo para deficientes visuais (lado cliente). Nesta etapa, foi utilizado a IDE Eclipse[26] juntamente com o *plugin* para criação de projetos PHP[27].

[26] Eclipse – http://www.eclipse.org/
[27] Plugin Eclipse PHP – http://www.eclipse.org/projects/project.php?id=tools.pdt

No lado cliente, foi seguida a arquitetura proposta. Desta forma, foi desenvolvido o Conversor de Interfaces (vide Figura 6.2) que, na estrutura do projeto cliente desta implementação (Figura 7.3), consiste apenas no arquivo *index.php*. Esta tela, trata de redirecionar para o módulo do sistema para deficientes visuais ou para o sistema legado (de acordo com a escolha do usuário). Nada impede que o conversor de interfaces seja um sistema separado do sistema que contém os módulos de interfaces acessíveis e que ele faça algum processamento relevante, porém sua função final é a de redirecionar a apresentação do sistema de acordo com a necessidade escolhida pelo usuário. Como esta implementação possui apenas o caráter de avaliar a arquitetura proposta, o conversor de interfaces foi incorporado no próprio sistema que contém os módulos acessíveis.

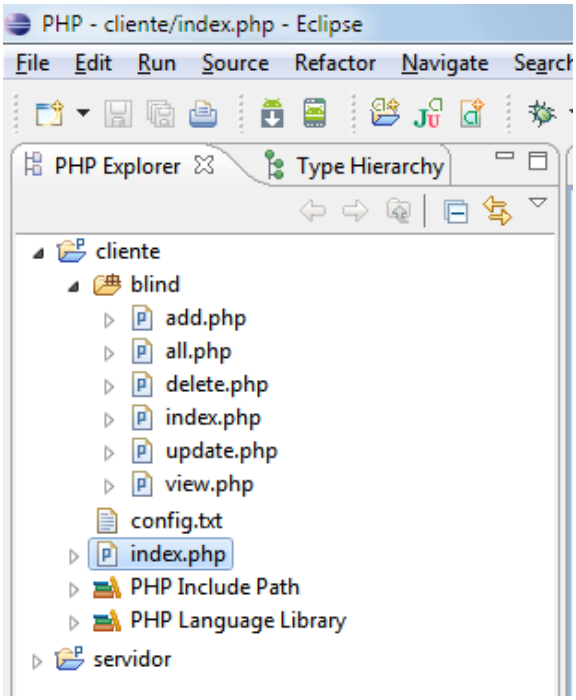

Figura 7.3 – Estrutura do Projeto Cliente no Eclipse

Em seguida, foram desenvolvidas as telas que consumirão os serviços detalhados anteriormente. Percebe-se a organização modular presente na estrutura do projeto, isto é uma exigência da arquitetura proposta exatamente para tornar as implementações dos módulos acessíveis flexíveis e desacopladas.

Por fim, foi criado um arquivo de configuração, chamado *config*.txt, para facilitar futuras expansões e alterações do sistema. Este arquivo possui apenas o endereço do *Web Service*, a tabela que ele manipula e os módulos acessíveis que ele atualmente suporta. Entretanto, futuramente, informações adicionais serão incluídas neste arquivo de configuração.

No lado servidor (Figura 7.4), foram implementados todos os servidos necessários para o funcionamento do sistema. Como o servidor não é um sistema “visível” pelo usuário, ele possui apenas o acesso à base de dados e implementa os serviços necessários para o sistema de cadastro.

Para facilitar futuras expansões e manutenções do sistema servidor, foi criado também um arquivo de configuração chamado *config.txt*. Este arquivo possui, basicamente, as informações para acessar a base de dados.

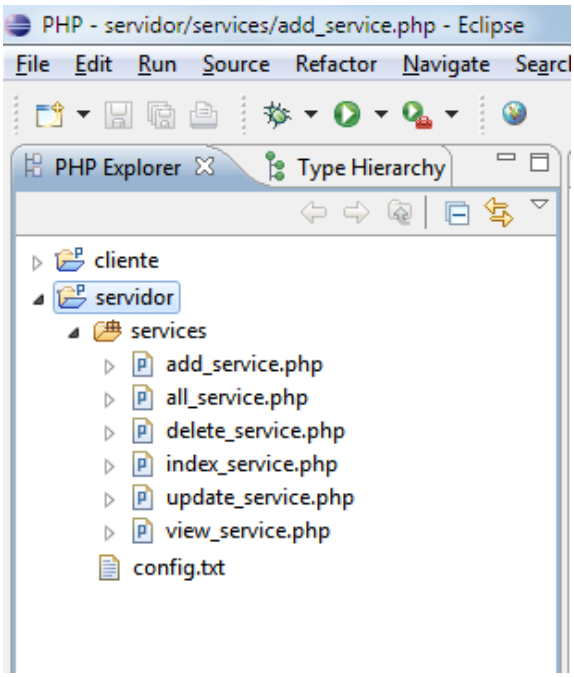

Figura 7.4 – Estrutura do Projeto Servidor no Eclipse

7.3. Considerações Finais

Neste capítulo, foi abordada uma implementação de um Sistema de Cadastro de Alunos com Deficiência com um módulo de interface acessível para deficientes visuais. Vale salientar que esta implementação trata-se apenas de um estudo de caso da arquitetura proposta. Desta forma não foi, portanto, uma preocupação deste trabalho tornar o Sistema de Cadastro robusto de tal forma que ele pudesse atender a várias necessidades dos usuários deficientes.

Certamente, que existem alguns tipos de deficiências que exigirão uma reestruturação dos serviços detalhados neste capítulo. Além disso, o sistema foi construído de forma a ser o mais simples possível, usando o mínimo de formulários possíveis e concentrando a navegação e o acesso às funcionalidades (no caso, incluir, atualizar, visualizar e deletar) por meio de links. Caso a consulta de alunos fosse, por exemplo, por meio de um formulário de consulta,

seria necessária uma reestruturação do serviço de consulta de alunos. Logo, a metodologia de implementação dos serviços, bem como as definições de requisição de resposta de serviços pode ser reajustada em face a inevitáveis mudanças de requisito no projeto.

No próximo capítulo, serão apresentadas as telas produzidas pelo Sistema de Cadastro de Alunos com Deficiência. Além disso, serão apresentados os resultados destas telas ao serem analisados pelos principais validadores de acessibilidade.

8. Resultados

Após um período de desenvolvimento de cada bloco definido na arquitetura detalhada no Capítulo 6, obteve-se um sistema funcional tanto para o módulo destinado aos usuários não deficientes quanto para o módulo destinado aos usuários deficientes visuais. As telas foram testadas nos principais navegadores (Chrome, Firefox e Internet Explorer) e todas apresentaram a mesma interface acessível. Entretanto, os testes da interface se concentraram na ferramenta Webvox do sistema operacional Dosvox versão 4.4 (exposta no sub-tópico 4.3.1) pois é o sistema mais usado pela comunidade de deficientes visuais e motores. As telas resultantes do desenvolvimento dos módulos são apresentadas nas figuras a seguir.

Primeiramente, é apresentado o Conversor de Interfaces (Figura 8.1) que, conforme exposto no capítulo anterior, possui a funcionalidade de redirecionar o fluxo do sistema para os módulos suportados de acordo com as necessidades do usuário. Como se trata de uma tela de acesso, sua interface deve ser a mais acessível possível para que ele possa ser manipulado por usuários deficientes e não deficientes.

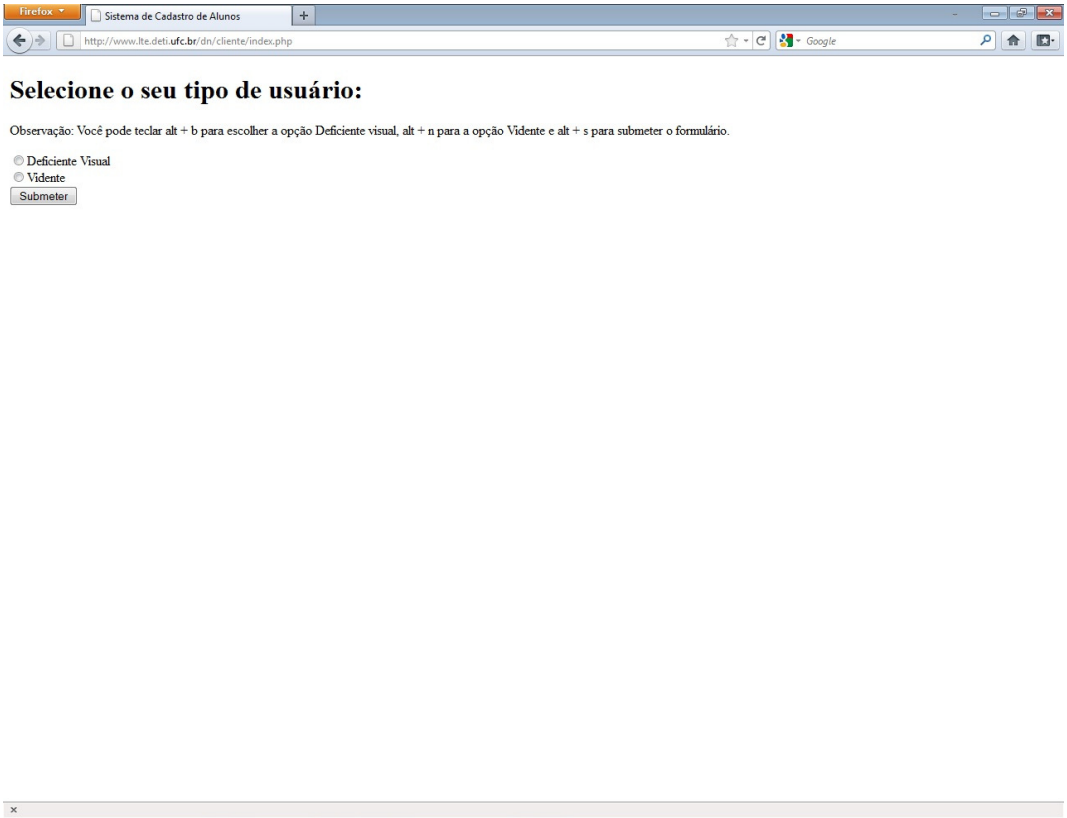

Figura 8.1 – Tela de Seleção de Interface

Entretanto, o usuário deficiente visual manipula as telas do sistema Web com o uso de leitores de tela. Como no caso deste trabalho foi usado o leitor de telas do sistema Dosvox, segue na Figura 8.2, a tela do Conversor de Interfaces sendo lida pelo Dosvox. Percebe-se que a tela em questão foi lida corretamente, proporcionando ao deficiente visual a possibilidade de escolher a interface do sistema de acordo com sua necessidade.

Conforme exposto no sub-tópico 4.3.1, o Dosvox se comunica com o deficiente visual por meio de síntese de voz. Com sua ferramenta Webvox a comunicação continua sendo por meio de síntese de voz. Deste modo, tanto a tela do Conversor de Interfaces quanto as demais telas apresentadas pelo Webvox neste capítulo foram faladas para o usuário deficiente visual.

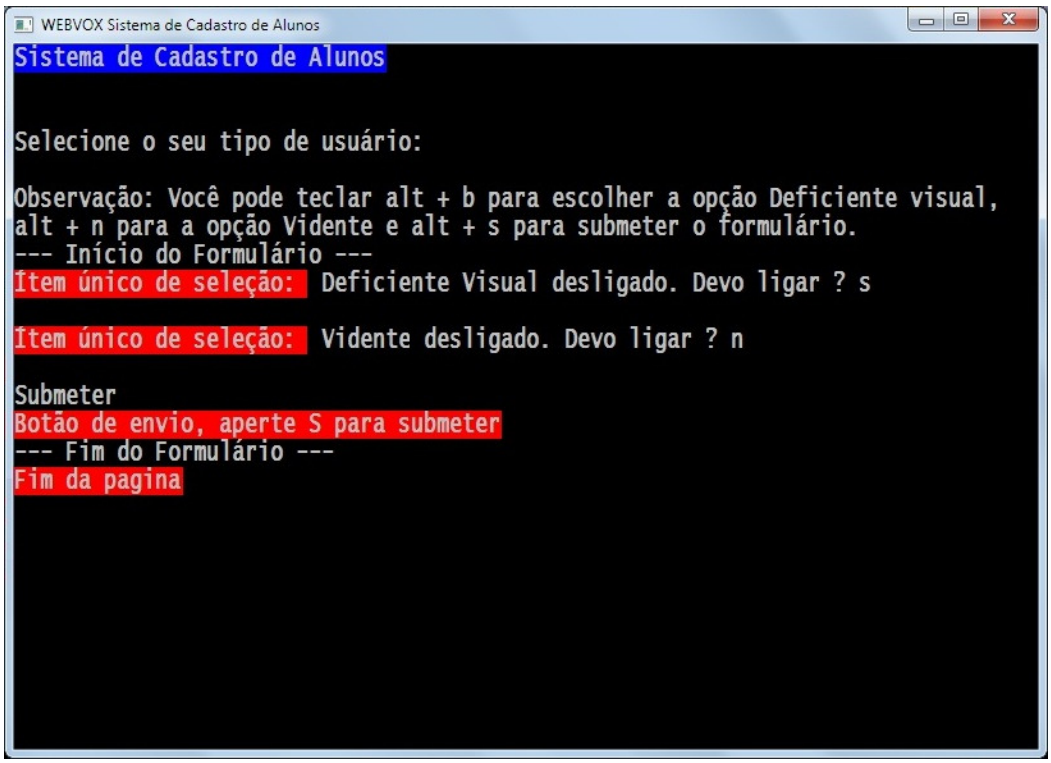

Figura 8.2 – Tela de Seleção de Interface (Dosvox)

Uma vez escolhida a interface de acesso, o sistema passa a apresentar as informações de acordo com a necessidade. No caso do Sistema de Cadastro de Alunos com Deficiência, o usuário poderá visualizar, inserir, atualizar e deletar alunos.

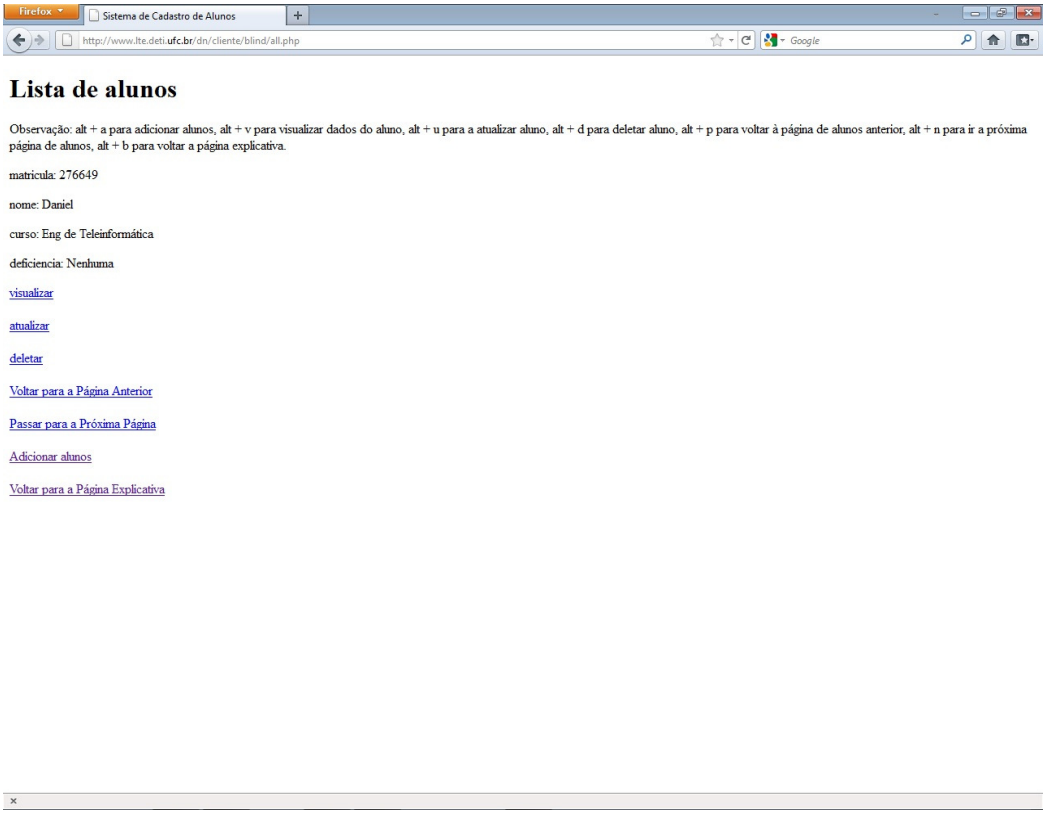

Figura 8.3 – Tela Inicial para Usuários Deficientes Visuais

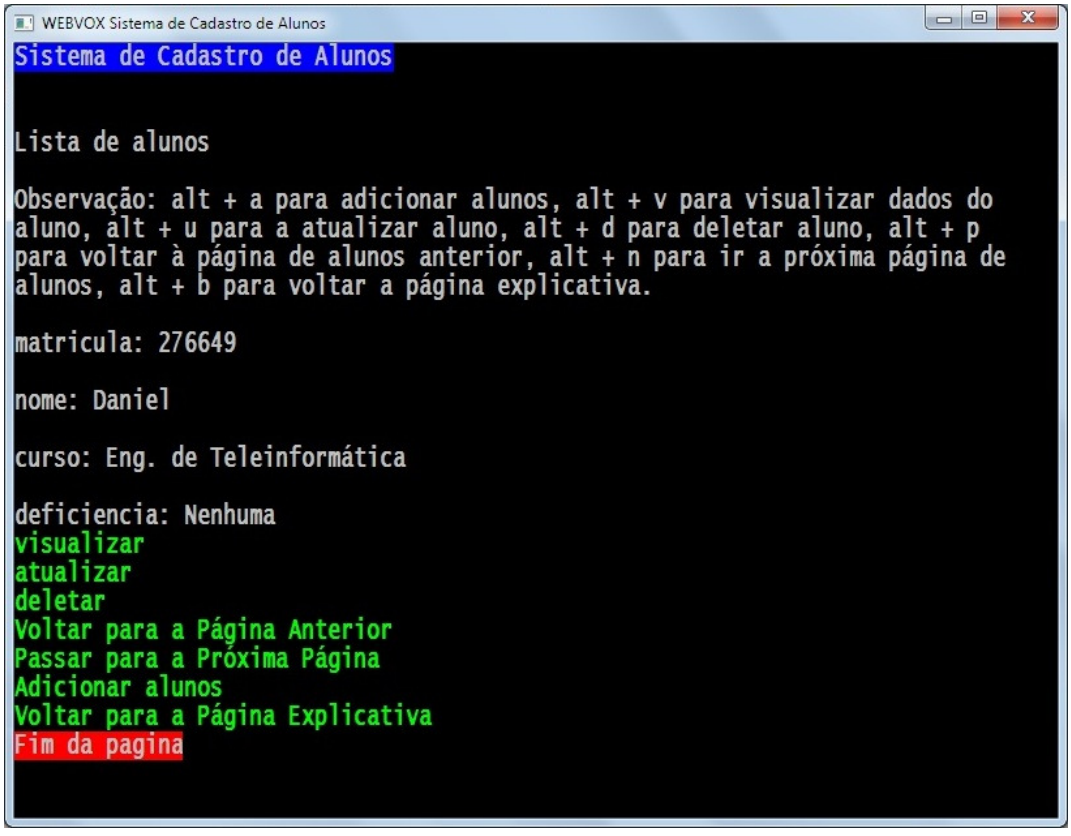

Figura 8.4 – Tela Inicial para Usuários Deficientes Visuais (Dosvox)

Na realidade a tela de listagem de alunos (Figura 8.3) não é a tela apresentada para o deficiente visual. Ela foi apresentada neste trabalho apenas para que se possa perceber a diferença de interface em comparação com a do sistema legado. Na interface presente no módulo acessível para deficientes visuais, percebe-se que a estrutura antiga de interface na forma de tabela, apresentada no sistema legado (vide Figura 7.1), foi adaptada para uma estrutura linear de apresentação do conteúdo.

A tela que realmente apresentada ao usuário deficiente visual (Figura 8.4) foi lida corretamente por um leitor de tela, tornando assim possível para este usuário a manipulação das funcionalidades do sistema.

Além disso, nada impede que sejam apresentadas informações adicionais e distintas nos módulos acessíveis, como a observação com o conteúdo descritivo de atalhos de teclado para acesso aos *links*. Futuramente, na implementação de novos módulos, será necessária, além de uma particular apresentação de conteúdo, a geração de conteúdos auxiliares com o intuito de tornar o sistema mais intuitivo.

Nas demais telas do sistema, percebe-se também que a adaptação da interface do sistema legado, que era disposta na forma de tabela, tornou correta a leitura de tela de todas as telas do sistema.

Vale salientar que as imagens de tela apresentadas no browser tradicional para videntes possuem apenas o caráter de ilustrar a diferença estrutural na adaptação da interface legada. Na prática, o que é apresentado para o usuário deficiente visual são as imagens de telas lidas pelo Dosvox.

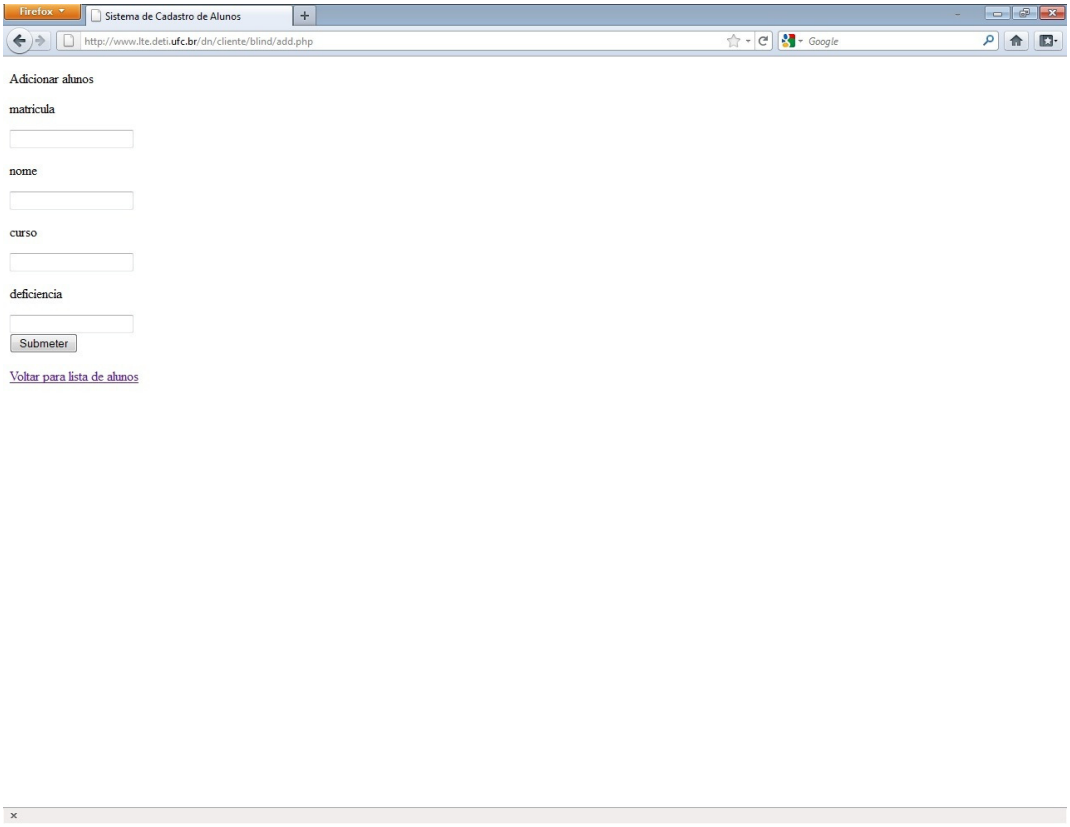

Figura 8.5 – Tela de Inclusão de Alunos para Usuários Deficientes Visuais

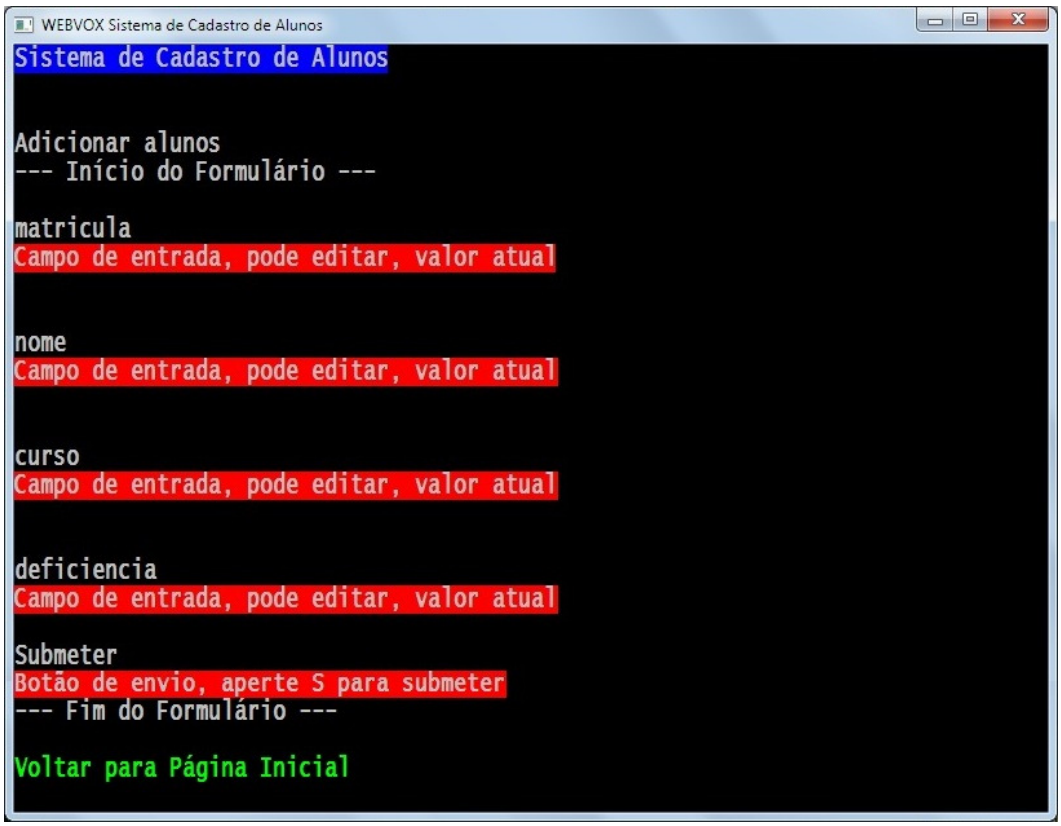

Figura 8.6 – Tela de Inclusão de Alunos para Usuários Deficientes Visuais (Dosvox)

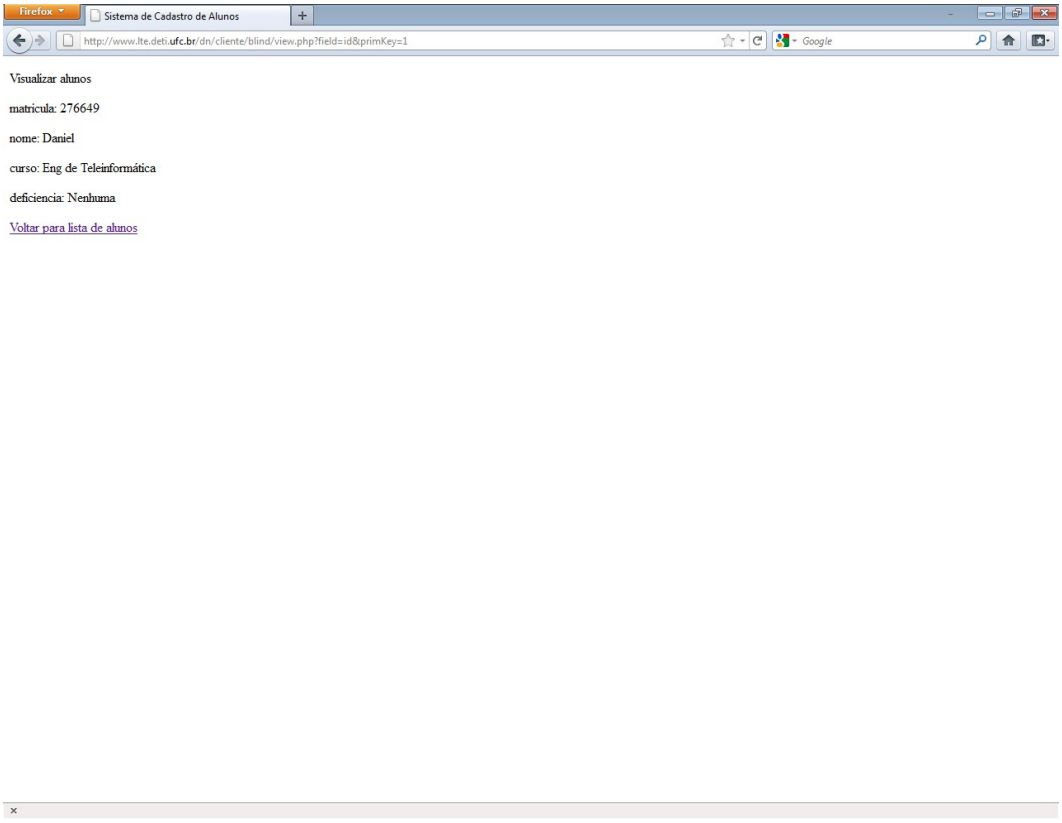

Figura 8.7 – Tela de Visualização de Alunos para Usuários Deficientes Visuais

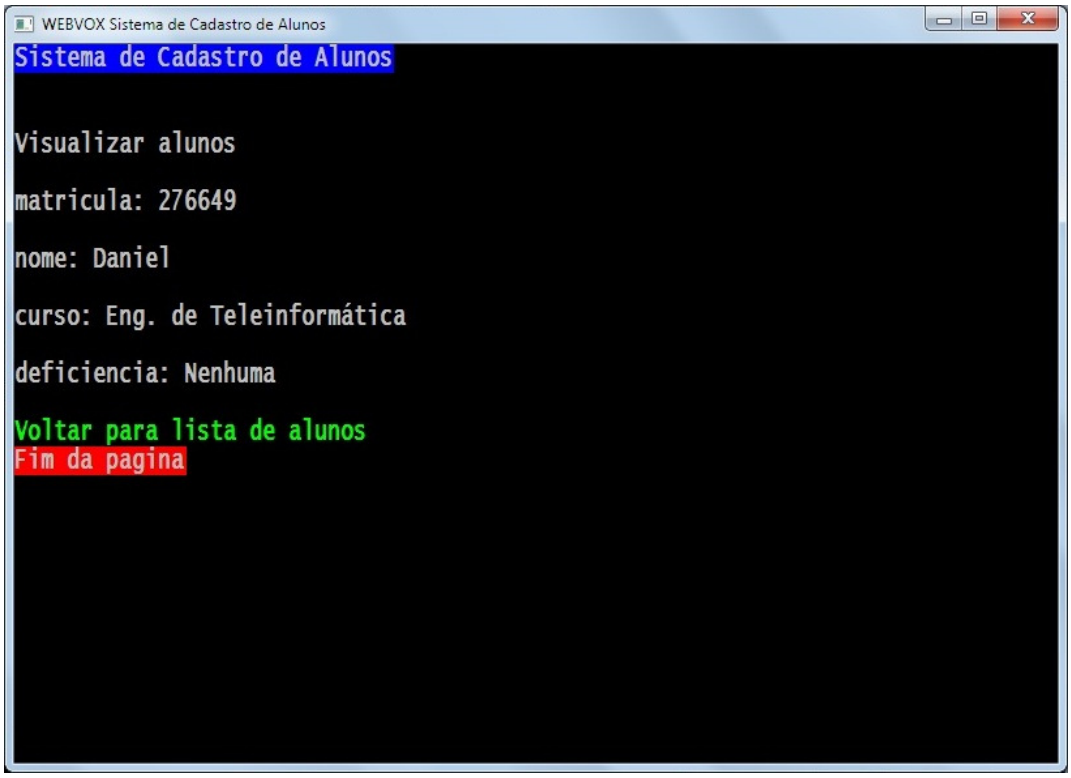

Figura 8.8 – Tela de Visualização de Alunos para Usuários Deficientes Visuais (Dosvox)

Entretanto, conforme exposto no Tópico 3.2, é necessário que o sistema seja submetido aos validadores de acessibilidade afim de rotular o sistema como satisfatório para algum nível de acessibilidade definido pela WAI-W3C. Logo, o sistema implementado na arquitetura proposta foi submetido aos validadores de acessibilidade (expostos no tópico 3.2). Dos quatro validadores apresentados neste trabalho, foi escolhido 2 deles, Hera e DaSilva (detalhados nos subtópicos 3.2.1 e 3.2.4, respectivamente). O motivo desta escolha deveu-se ao fato de que ambos são os mais utilizados dentre a comunidade de desenvolvedores (DaSilva no âmbito nacional e Hera no âmbito mundial).

O sistema foi submetido a estes validadores e os resultados são apresentados nas figuras a seguir.

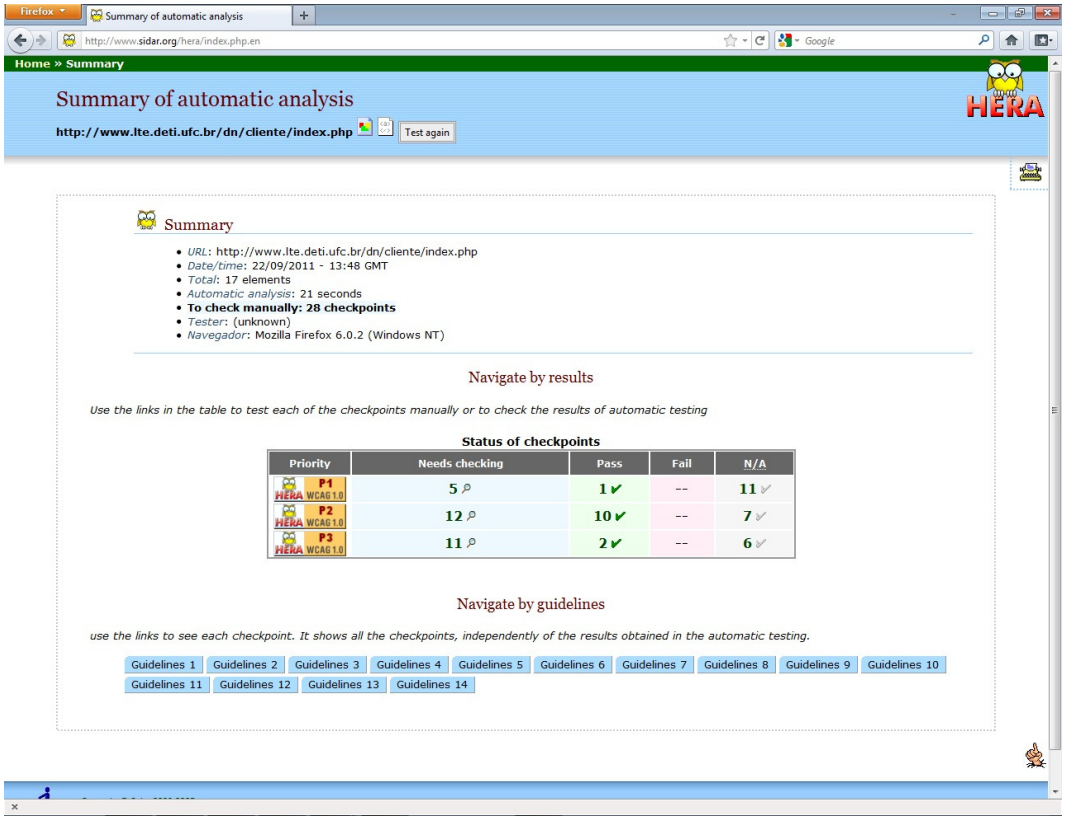

Figura 8.9 – Resultado da Tela de Seleção de Interface (validador HERA)

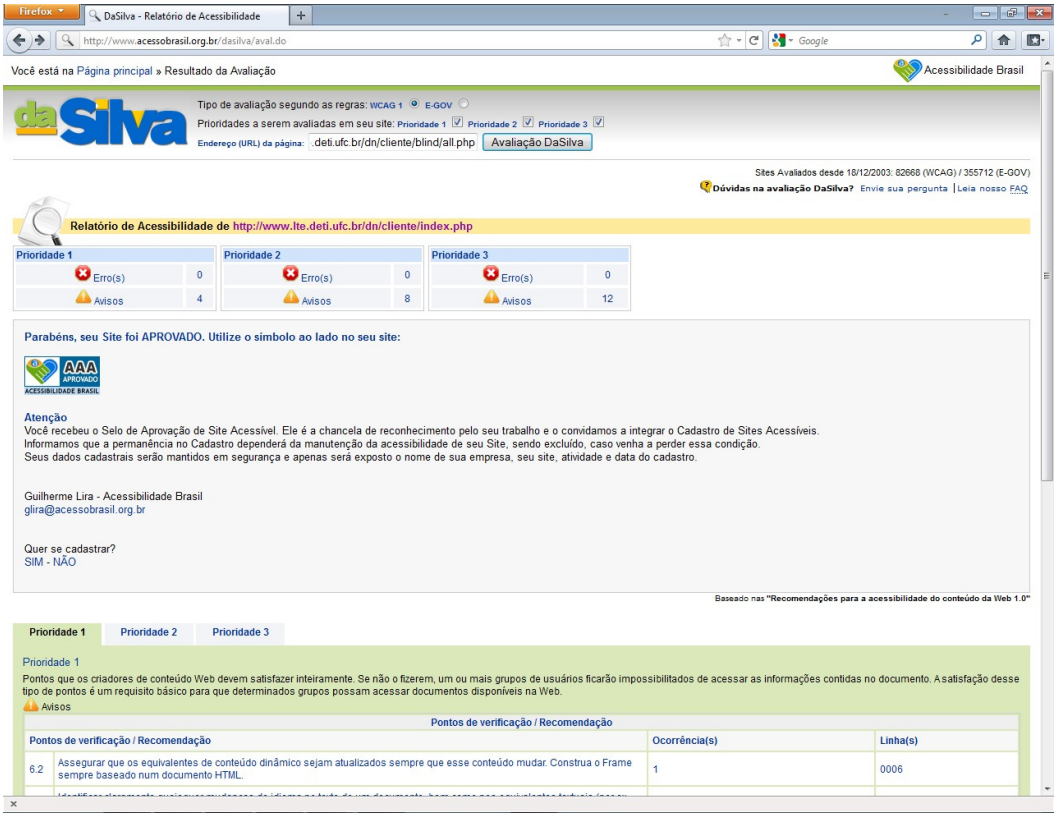

Figura 8.10 – Resultado da Tela de Seleção de Interface (validador DaSilva)

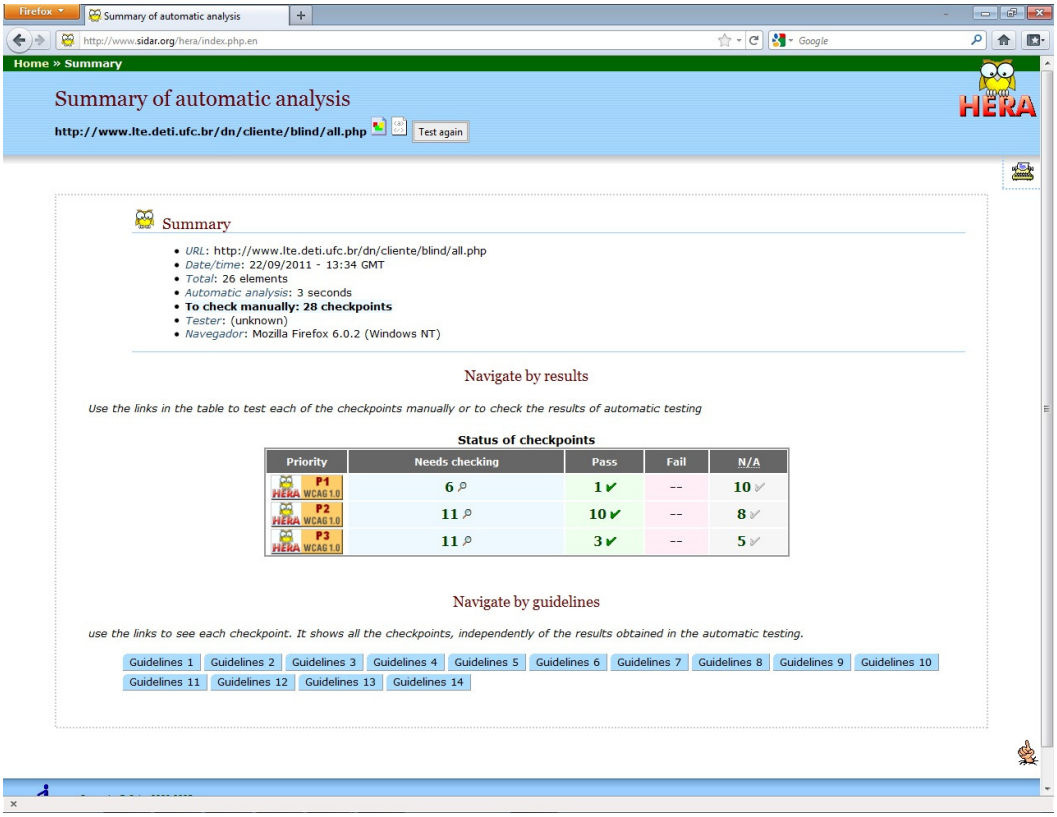

Figura 8.11 – Resultado da Tela Inicial para Deficientes Visuais (validador HERA)

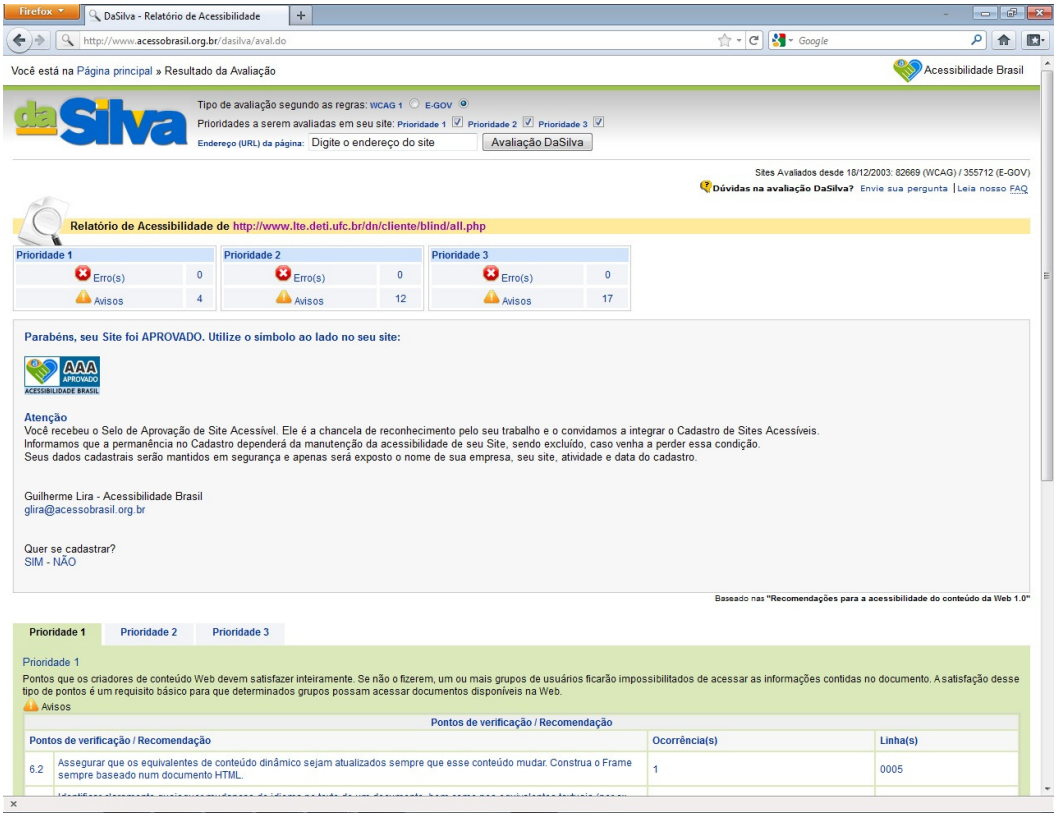

Figura 8.12 – Resultado da Tela Inicial para Deficientes Visuais (validador DaSilva)

As Figuras 8.9 e 8.11 mostram todas as informações presentes no momento da validação da tela de seleção de interfaces no validador HERA. Na área **Summary** é apresentada informações gerais do acesso (URL, data de acesso, total de elementos que foram analisados no código, tempo de análise, navegador, etc). Na área **Status of checkpoints** são listados os pontos que passaram pela análise, os pontos que falharam na análise e os pontos que não falharam mas que o validador não consegue dizer que está correto.

As Figuras 8.10 e 8.12 mostram todas as informações presentes no momento da validação da tela de seleção de interfaces no validador daSilva. A tela de resultados deste navegador é mais direta e apresenta apenas uma tabela com a numeração dos pontos de avisos e dos pontos de falhas para os 3 níveis de prioridade. Logo abaixo, o validador dá o resultado final através do selo de acessibilidade. Por fim, o daSilva apresenta uma descrição dos pontos de aviso e de falhas (se houverem) separados por nível de acessibilidade.

A partir dos resultados acima expostos, percebe-se que as telas desenvolvidas no módulo para deficiente visual estão de acordo com todos os critérios de acessibilidade especificados pela W3C, fazendo com que o sistema desenvolvido na arquitetura proposta possa carregar em suas páginas o Selo de Aprovação de Site Acessível (Nível AAA).

9. Conclusão e Trabalhos Futuros

Tomando como base a fase de desenvolvimento (que foi feito de forma desacoplada e sem maiores problemas de comunicação entre subsistemas) e a fase de testes (na qual os subsistemas foram testados com o intuito de satisfazer os critérios de usabilidade e navegabilidade dos usuários deficientes visuais), conclui-se que este trabalho atingiu aos objetivos especificados inicialmente. A arquitetura projetada proporcionou a adaptação da interface de um sistema inacessível para uma interface compreensível por um leitor de tela, tornando, assim, possível sua manipulação por um usuário deficiente visual. Além disso, a arquitetura proporcionou expansibilidade e interoperabilidade ao sistema de exemplo deste trabalho (Sistema de Cadastro de Alunos de Acessibilidade), fazendo com que este sistema possa agregar outras plataformas de desenvolvimento em sua expansão sem comprometer o funcionamento dos seus diversos módulos.

Tendo em vista o fato de que as páginas web da Universidade Federal do Ceará (UFC) ainda não satisfazem aos principais critérios de acessibilidade, sugere-se que arquitetura semelhante possa ser integrada às mesmas visando a redução do trabalho em futuras necessidades de adaptação. Para isso, seria necessário que os módulos dos portais da universidade fossem desenvolvidos com base na Arquitetura Orientada a Serviços. Sob esse paradigma, serviços podem ser consumidos por diferentes módulos que, por sua vez, podem ser desenvolvidos separadamente. A arquitetura proposta facilitaria a adaptação para apresentação de conteúdo de acordo com a necessidade escolhida.

Naturalmente, nem todos os recursos Web foram usados no sistema de exemplo deste trabalho. Um exemplo disso é que não foi realizada manipulação de conteúdos multimídia e CAPTCHA[28] (recursos muito conhecidos na Internet). Para isso, é necessário que seja definida uma forma de transmissão da informação para que o conteúdo multimídia seja apresentado de modo síncrono com a descrição textual ou outro conteúdo multimídia com descrição em símbolos ou LIBRAS, de modo a atender também aos usuários deficientes auditivos.

Esta arquitetura não foi testada exaustivamente em todas as tecnologias assistivas apresentadas neste trabalho. Assim, é importante que ela seja amadurecida e enriquecida com novos testes e com o desenvolvimento de módulos específicos que ofereçam suporte a outras tecnologias assistivas ou mesmo a outros tipos de deficiência. Os testes se concentraram em

[28] *Completely Automated Public Turing Test To Tell Computers and Humans Apart*

leitores de tela e no sistema operacional Dosvox 4.4. Futuramente, as funcionalidades do sistema devem ser testadas quando o usuário utilizar uma interface de entrada ou saída diferentes, por exemplo, como uma linha Braile (exposto no sub-tópico 4.2.1) ou uma impressora braile. Assim o sistema não só será adaptável na apresentação do conteúdo, mas também no suporte a diferentes dispositivos de entrada e saída de dados.

Espera-se ter contribuído para que, num futuro próximo, os portais e demais sistemas Web da UFC sejam totalmente acessíveis para todos os tipos de deficiência, fazendo com que a UFC seja de fato uma universidade de todos.

Referências Bibliográficas

[1] THATCHER, Jim et al. **Web Accessibility: Web Standards and Regulatory Compliance**, friends of ED, 2006.

[2] HOGETOP, L e SANTAROSA, L.M.C, **Tecnologias Adaptiva/Assistiva Informáticas na Educação Especial: viabilizando a acessibilidade ao potencial individual.** Revista de Informática na Educação: Teoria, Prática – PGIE/UFRGS.

[3] ERL, Thomas. **SOA: Princípios de Design de Serviços.** Editora Pearson Prentice Hall, 2009

[4] CARVALHO, José Oscar Fontanini de **Interfaces para o Deficiente Visual.** Revista Informédica 1(1): 5-11, 1993. Disponível em <http://www.informaticamedica.org.br/informed/defic.htm>, Acesso em Agosto de 2011.

[5] THE INTERNET SOCIETY. RFC 2616 **Hypertext Transfer Protocol - HTTP/1.1.** United States, 1999.

[6] THE PHP GROUP **PHP5 Manual**, Disponível em <http://www.php.net/manual/en/index.php>, Acesso em Junho de 2011.

[7] FLANAGAN, D. **JavaScript: The Definitive Guide.** 5th. ed. O'Reilly Media, 2006.

[8] JAYAWARDANA, N. **Installing WSO2 WSF/PHP with XAMPP on MS Windows**, Disponível em < http://wso2.org/library/3076>, Acesso em Junho de 2011.

[9] Apache Friends **Installing XAMPP Beta Version 1.7.5**, Disponível em <http://www.apachefriends.org/en/xampp-beta.html>, Acesso em Maio de 2011.

[10] WSF/PHP **Improved WSDL mode support in new WSF/PHP release**, Disponível em: <http://wso2.com/about/news/wsf-php-1-3-0-release/>. Acesso em Julho de 2011.

[11] CHRISTENSEN, Erik; CURBERA, Francisco; MEREDITH, Greg; WEERAVARANA, Sanjiva **Web Services Description Language (WSDL) 1.1. W3C Notes, Mar, 2001.**, Disponível em <http://www.w3.org/TR/wsdl>, Acesso em Julho de 2011.

[12] WSO2 Oxygen Tank **WSO2 WSF/PHP API**, Disponível em <http://wso2.org/project/wsf/php/1.2.0/docs/api.html>, Acesso em Junho de 2011.

[13] World Wide Web Consortium **Extensible Markup Language (XML) 2004**, Disponível em: <http://www.w3.org/XML/>, Acesso em Julho de 2011.

[14] World Wide Web Consortium **HTML 4.01 Specification, 1999**, Disponível em: <http://www.w3.org/TR/html401/>, Acesso em Maio de 2011.

[15] BRASIL, Decreto nº 5.296 de 2 de Dezembro de 2004, **Regulamenta as Leis nos 10.048, de 8 de novembro de 2000, que dá prioridade de atendimento às pessoas que especifica, e 10.098, de 19 de dezembro de 2000, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida, e dá outras providências.**, Disponível em <http://www.planalto.gov.br/ccivil_03/_Ato2004-2006/2004/Decreto/D5296.htm>, Acesso em Agosto de 2011.

[16] BRASIL, Decreto n° 3.298 de 21 de Dezembro de 1999, **Estatuto da Pessoa com Deficiência**, Disponível em <http://www.senado.gov.br/senadores/Senador/PauloPaim/pages/vida/publicacoes/textos/Estatuto%20da%20Pessoa%20com%20Defici%C3%AAncia%20Novo.pdf>, Acesso em Agosto de 2011.

[17] SANTOS, Natanael Antônio dos. **Tópicos em Percepção e Processamento Visual da Forma: Acuidade Visual Versus Sensibilidade ao Contraste**, Disponível em < http://www.revispsi.uerj.br/v3n1/artigos/Artigo%206%20-%20V3N1.pdf>, Acesso em Agosto de 2011.

[18] World Wide Web Consortium **Web Accessibility Initiative**, Disponível em <http://www.w3.org/WAI/>, Acesso em Maio de 2011.

[19] Entreamigos **Informações Básicas sobre Deficiência Visual**, Disponível em <http://www.entreamigos.com.br/sites/default/files/textos/Informações%20basicas%20sobre%20deficiencia%20visual.pdf>, Acesso em Julho de 2011.

[20] RESNIKOFF, Serge et al. **Global magnitude of visual impairment caused by uncorrected refractive errors in 2004**, Disponível em <http://www.who.int/bulletin/volumes/86/1/07-041210.pdf>, Acesso em Agosto de 2011.

[21] CONFORTO, Débora e SANTAROSA, Lucila M. C. **Acessibilidade à Web: Internet para Todos.**, Disponível em <http://pan.nied.unicamp.br/~proinesp/material/arquivos/Semana%203%20-%20Internet%202/Leituras/Leitura%20Complementar%20-%20Acessibilidade/acessibilidade.pdf>, Acesso em Julho de 2011.

[22] SERPRO, **Acesso à Web e Tecnologia Assistiva**, Disponível em <http://www.serpro.gov.br/acessibilidade/acesso.php>, Acesso em Julho de 2011.

[23] BORGES, Antônio. **Projeto Dosvox. Núcleo de Computação Eletrônica da UFRJ – Universidade Federal do Rio de Janeiro.**, Disponível em <http://intervox.nce.ufrj.br/dosvox>, Acesso em Junho de 2011.

[24] POTTS, Stephen; KOPACK, Mike. **Aprenda em 24 horas Web Services.** Rio de Janeiro: Elsevier, 2003.

[25] World Wide Web Consortium **Simple Object Access Protocol (SOAP) 1.2**, Disponível em < http://www.w3.org/TR/soap12-part1/>, Acesso em Setembro de 2011.

Apêndices

Apêndice 1. Critérios de Acessibilidade Nível A

- **Alternativas de Texto**

O sítio Web deve fornecer alternativas de texto para qualquer conteúdo não-textual. Desta forma, o mesmo poderá ser transformado em descrição por leitores de tela. Além disso, deve-se disponibilizar alteração de tamanho de fonte (para mais e para menos) para que possam ser lidos por usuários com baixa visão.

Por fim, o desenvolvedor deve disponibilizar a mesma informação em Braille, em áudio, símbolos ou linguagem mais simples.

Entretanto existem algumas situações que exigem um tratamento diferenciado. Estas situações serão especificadas nos tópicos abaixo.

- **Controles de Entrada**

Se uma página de um sítio Web tiver um controle que permita entrada do usuário, então deve ser apresentado próximo ao controle, um texto indicando seu propósito.

- **Sensorial**

Se o conteúdo se destina a criar uma experiência sensorial específica, então o sistema deve descrever de forma textual a sensação que se pretende passar com este conteúdo.

- **CAPTCHA**

Se o propósito do conteúdo não textual for o de confirmar que o sistema está sendo acessado por uma pessoa ao invés de uma máquina, então formas textuais alternativas que identificam e descrevem o propósito do conteúdo devem ser apresentadas. Formas alternativas de CAPTCHA usando modos de saída para diferentes tipos de percepção sensorial (como texto e voz) devem ser fornecidas para acomodar diferentes tipos de deficiência.

- **Formatação e Campos Invisíveis**

Se o conteúdo não-textual é meramente decorativo ou se for um campo de uma página Web que não deve ser apresentado, então ele deve ser implementado de um forma que ele possa ser ignorado pelas diversas tecnologias assistivas.

- **Multimídia Baseada no Tempo**

  - **Multimídia Somente Áudio ou Somente Vídeo (Pré-Gravada)**

Para multimídia pré-gravada somente áudio (música, por exemplo) e multimídia pré-gravada somente vídeo (filme mudo, por exemplo) é necessário que todo o conteúdo seja apresentado com a mesma temporização sonora, porém na forma textual. Isso, porque o conteúdo textual pode ser traduzido em qualquer modalidade sensorial pelas tecnologias assistivas (visual, auditiva ou tátil, por exemplo).

  - **Multimídia com Áudio e Vídeo (Pré-Gravada)**

Para conteúdos multimídia pré-gravada tradicionais com áudio e vídeo, é necessário que seja disponibilizada a apresentação de legendas para o áudio e legendas de descrição de vídeo para usuários deficientes visuais.

- **Formulários Adaptáveis**

  - **Informações e Relações**

As informações, estrutura e relações transmitidas através da apresentação de um formulário devem ser determinadas de forma flexível (de preferência através da escolha do usuário).

  - **Seqüência Significativa**

A seqüência de leitura do formulário deve ser reajustada em face às diferentes necessidades do usuário. Logo, várias opções de leitura seqüencial devem ser programadas no sistema.

  - **Características Sensoriais**

As instruções fornecidas para compreender e utilizar o formulário não devem, de forma alguma, depender das características sensoriais dos componentes, tais como: forma, tamanho, localização visual, orientação ou som.

- **Conteúdos Distinguíveis**

  - **Uso de cores**

A cor não deve ser usada como única forma para transmitir informação, indicando uma ação, pedindo uma resposta ou distinguindo um elemento visual.

Este critério é importante, pois muitos sistemas usam formulários que dizem, por exemplo, que os campos em vermelho são obrigatórios. Entretanto, para deficientes visuais,

esta informação é inútil. Logo, o sistema deve fornecer outras formas, além da visual, para distinguir a informação da obrigatoriedade do preenchimento de um campo de formulário.

- **Controle de Áudio**

Se qualquer áudio na página Web for executado por mais de 3 segundos, é necessário disponibilizar mecanismos de manipulação deste áudio (para que o usuário possa parar o som ou escutá-lo pausadamente ou até mesmo controlar o nível sonoro do áudio).

Vale salientar que, neste caso, não se pode esquecer de atender ao critério de **Multimídia Somente Áudio ou Somente Vídeo** acima citado.

- **Acessibilidade de Teclado**

- **Todas Funcionalidades no Teclado**

É necessário que todas as funcionalidades no sistema possam ser acessadas via teclado.

- **Sem Bloqueio de Teclado**

Muitas aplicações acabam abrindo caixa de diálogos modais (aquelas com botão Cancelar e OK, por exemplo). Quando isto acontece o foco do teclado estará preso na caixa de diálogo e, por padrão, o usuário não consegue acessar o restante dos controles do diálogo, impossibilitando a manipulação do mesmo.

Por conta disso, é necessário que se o foco de teclado pode ser movido para um componente de página usando uma interface de teclado, então ele deve ser afastado desse componente usando apenas uma interface de teclado.

- **Limite de Tempo de Uso do Sistema**

O sistema deve fornecer tempo suficiente para que o usuário possa ler e manipular o conteúdo.

Além disso, toda vez que utilizar limite de tempo de utilização, para obter sucesso neste critério de acessibilidade, o sistema deve satisfazer, pelo menos, um dos seguintes pontos especificados abaixo.

- **Desligar Limite de Tempo**

O usuário tem permissão para desligar o limite de tempo.

- **Ajustar Limite de Tempo**

O usuário pode ajustar o limite de tempo, podendo ampliá-lo em, pelo menos, 10 vezes o valor default.

- **Aviso e Solicitação de Extensão do Limite de Tempo**

O usuário é avisado antes do tempo expirar e tem, pelo menos, 20 segundos para estender o limite de tempo com uma ação simples (pressionando a tecla de espaço, por exemplo) e o usuário tem permissão para prorrogar o limite de tempo em, pelo menos, 10 vezes o valor default.

- **Exceção para Casos de Tempo Real**

Em aplicações em que o limite de tempo é uma parte necessária de um evento de tempo real (aplicações de leilão, por exemplo), nenhuma alternativa para o limite de tempo é possível.

- **Exceção para Transações Baseadas em Tempo**

Em aplicações em que a ampliação do limite de tempo iria invalidar alguma transação do sistema, nenhuma alternativa para o limite de tempo é possível.

- **Exceção para Limite de Tempo Superior a 20 Horas**

Em aplicações que possuem limite de tempo superior a 20 horas, nenhuma alternativa para o limite de tempo é possível.

- **Informações que Causam Ataques Epiléticos**

Não criar conteúdo de uma forma que possa gerar ataques epiléticos.

- **Três Flashes ou Abaixo do Limite**

As páginas Web não devem conter nenhum conteúdo que pisque mais de 3 vezes por segundo.

Além disso, o efeito de flash não pode ocupar um bloco de pixels maior do que 341 x 256 (isso tomando como referência uma tela de resolução 1024 x 768).

- **Navegabilidade**

O sistema deve fornecer formas de ajudar os usuários a navegar, localizar conteúdo e determinar onde os mesmos estão.

- **Blocos de Bypass**

É necessário disponibilizar mecanismos que ignorem blocos de conteúdo que se repetem em várias páginas Web.

- **Títulos das Páginas**

Os títulos das páginas Web devem ser claros e resumir seu propósito.

– **Ordem do Foco**

Se uma página Web puder ser navegada em seqüência e esta seqüência afetar seu significado ou operação, os componentes devem assumir a ordem de foco de um modo tal que a página preserve seu significado ou operação.

– **Links**

Cada link numa página Web deve conter em seu texto a sua finalidade, exceto quando o ato de especificar a finalidade do link seja ambíguo para os usuários em geral.

- **Legibilidade**

O sistema deve apresentar seu conteúdo sempre de forma legível e compreensível.

– **Idioma**

O sistema deve ser disponibilizado para ser visualizado em, pelo menos, dois idiomas.

- **Compatibilidade**

É necessário que os sistemas sejam desenvolvidos de modo a maximizar a compatibilidade com as tecnologias atuais e futuras, incluídos as tecnologias assistivas.

– **Organização dos Elementos de Marcação**

Em páginas Web implementadas utilizando linguagens de marcação, os elementos devem conter o início e o fim da tag. Além disso, os elementos devem estar encaixados de acordo com suas especificações originais. Por fim, os elementos não podem conter atributos duplicados ou ID's repetidos (exceto quando as especificações originais permitirem estas características).

– **Páginas aninhadas**

Não utilizar páginas Web aninhadas (ou a tag *frame* para linguagens de marcação). Esta estrutura impossibilita a leitura da página para alguns leitores de tela utilizados por deficientes visuais.

Para definir blocos dentro de páginas Web, deve-se usar a tag *div*, pois a mesma é bem entendida pelos leitores de tela.

Apêndice 2. Critérios de Acessibilidade Nível AA

Para que um sistema atinja o nível AA, ele deve, além de satisfazer os critérios do nível A, satisfazer todos os critérios listados abaixo.

- **Multimídia Baseada no Tempo**

  - **Multimídia em Tempo Real (Strem)**

Para multimídia em tempo real (stream) em áudio, as legendas devem ser disponibilizadas de modo sincronizado.

Para multimídia em tempo real (stream) em vídeo, as legendas de descrição de vídeo devem ser disponibilizadas de modo sincronizado.

Para multimídias em tempo real (stream) em áudio e vídeo, devem ser disponibilizadas tanto as legendas de áudio quanto as legendas de descrição de vídeo.

- **Conteúdos Distinguíveis**

  - **Contraste**

A apresentação visual de texto e imagens de texto numa página Web devem ter uma relação de contraste de, pelo menos, 4 : 1.

  - **Redimensionar o Texto**

Exceto para legendas e para imagens de texto, a página Web deve disponibilizar a opção de redimensionar seu conteúdo em até 200% sem perda de conteúdo e funcionalidade e sem apoio de alguma tecnologia assistiva.

- **Assistência de Entrada de Dados**

  - **Sugestão de Erro**

Se um erro de entrada de dados é automaticamente detectado e sugestões de correção são conhecidas, então estas sugestões devem ser fornecidas aos usuários (a menos que isso coloque em risco a segurança e a finalidade do conteúdo).

Apêndice 3. Critérios de Acessibilidade Nível AA

Para que um sistema atinja o nível AAA, ele deve, além de satisfazer os critérios dos níveis A e AA, satisfazer todos os critérios listados abaixo.

- **Multimídia Baseada no Tempo**

  - **Linguagem de Sinais**

Deve-se satisfazer todos os critérios listados nos níveis A e AA disponibilizando a mesma versão do conteúdo na forma de linguagem de sinais.

- **Conteúdos Distinguíveis**

  - **Contraste**

A apresentação visual de texto e imagens de texto numa página Web devem ter uma relação de contraste de, pelo menos, 7 : 1.

  - **Áudio de Background Inexistente ou Baixo**

Para áudio pré-gravado que não seja nem fala em primeiro plano, nem áudio CAPTCHA, o sistema deve disponibilizar a opção de desligar o áudio de background. Além disso, este tipo de áudio deve ter, no mínimo, 20 dB a menos que os possíveis conteúdos de áudio presentes do sistema, exceto para áudios de background ocasionais que duram apenas 1 ou 2 segundos.

  - **Apresentação Visual**

Para a apresentação visual de qualquer bloco de texto, um mecanismo de ajuste de layout deve ser disponibilizado para alcançar os itens abaixo:

- Cores de primeiro plano e de background podem ser selecionadas pelo usuário;
- Larguras não maiores do que 80 caracteres;
- Texto não deve possuir layout justificado;
- O espaçamento entre os parágrafos deve ser, pelo menos, 1.5 vezes maior que o espaçamento entre linhas.

Anexos

- **add_service.wsdl**

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.wso2.org/php"
             xmlns:tnx="http://www.wso2.org/php/xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:http="http://www.w3.org/2003/05/soap/bindings/HTTP/"
             xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
             targetNamespace="http://www.wso2.org/php">
  <types>
    <xsd:schema elementFormDefault="qualified"
                targetNamespace="http://www.wso2.org/php/xsd">
      <xsd:element name="addQuestion">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="message" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="addQuestionResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="returnVal" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="add">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="message" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="addResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="returnVal" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <message name="addQuestion">
    <part name="parameters" element="tnx:addQuestion" />
  </message>
  <message name="addQuestionResponse">
    <part name="parameters" element="tnx:addQuestionResponse" />
  </message>
  <message name="add">
    <part name="parameters" element="tnx:add" />
  </message>
  <message name="addResponse">
    <part name="parameters" element="tnx:addResponse" />
```

```
  </message>
  <portType name="portal_services_add_service.phpPortType">
    <operation name="addQuestion">
      <input message="tns:addQuestion" />
      <output message="tns:addQuestionResponse" />
    </operation>
    <operation name="add">
      <input message="tns:add" />
      <output message="tns:addResponse" />
    </operation>
  </portType>
  <binding name="portal_services_add_service.phpSOAPBinding"
           type="tns:portal_services_add_service.phpPortType">
    <soap:binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                  transport="http://schemas.xmlsoap.org/soap/http"
                  style="document" />
    <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
               name="addQuestion">
      <soap:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                      soapAction=" http://www.lte.deti.ufc.br/dn/portal/
                      services/add_service.php/addQuestion"
                      style="document" />
      <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </input>
      <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </output>
    </operation>
    <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
               name="add">
      <soap:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                      soapAction=" http://www.lte.deti.ufc.br/dn/portal/
                      services/add_service.php/add"
                      style="document" />
      <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </input>
      <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </output>
    </operation>
  </binding>
  <service name="portal_services_add_service.php">
    <port xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
          name="portal_services_add_service.phpSOAPPort_Http"
          binding="tns:portal_services_add_service.phpSOAPBinding">
      <soap:address xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                    location="http://www.lte.deti.ufc.br/dn/portal/
                    services/add_service.php" />
    </port>
  </service>
</definitions>
```

- **all_service.wsdl**

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.wso2.org/php"
             xmlns:tnx="http://www.wso2.org/php/xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:http="http://www.w3.org/2003/05/soap/bindings/HTTP/"
             xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
             targetNamespace="http://www.wso2.org/php">
  <types>
    <xsd:schema elementFormDefault="qualified"
                targetNamespace="http://www.wso2.org/php/xsd">
      <xsd:element name="all">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="message" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="allResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="returnVal" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <message name="all">
    <part name="parameters" element="tnx:all" />
  </message>
  <message name="allResponse">
    <part name="parameters" element="tnx:allResponse" />
  </message>
  <portType name="portal_services_all_service.phpPortType">
    <operation name="all">
      <input message="tns:all" />
      <output message="tns:allResponse" />
    </operation>
  </portType>
  <binding name="portal_services_all_service.phpSOAPBinding"
           type="tns:portal_services_all_service.phpPortType">
    <soap:binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                  transport="http://schemas.xmlsoap.org/soap/http"
                  style="document" />
    <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
               name="all">
      <soap:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                      soapAction=" http://www.lte.deti.ufc.br/dn/portal/
                      services/all_service.php/all"
                      style="document" />
      <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </input>
      <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
```

```
      </output>
    </operation>
  </binding>
  <service name="portal_services_all_service.php">
    <port xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
          name="portal_services_all_service.phpSOAPPort_Http"
          binding="tns:portal_services_all_service.phpSOAPBinding">
      <soap:address xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                    location="http://www.lte.deti.ufc.br/dn/portal/
                    services/all_service.php" />
    </port>
  </service>
</definitions>
```

- **delete_service.wsdl**

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.wso2.org/php"
             xmlns:tnx="http://www.wso2.org/php/xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:http="http://www.w3.org/2003/05/soap/bindings/HTTP/"
             xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
             targetNamespace="http://www.wso2.org/php">
  <types>
    <xsd:schema elementFormDefault="qualified"
                targetNamespace="http://www.wso2.org/php/xsd">
      <xsd:element name="delete">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="message" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="deleteResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="returnVal" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <message name="delete">
    <part name="parameters" element="tnx:delete" />
  </message>
  <message name="deleteResponse">
    <part name="parameters" element="tnx:deleteResponse" />
  </message>
  <portType name="portal_services_delete_service.phpPortType">
    <operation name="delete">
      <input message="tns:delete" />
      <output message="tns:deleteResponse" />
    </operation>
  </portType>
```

```
  <binding name="portal_services_delete_service.phpSOAPBinding"
           type="tns:portal_services_delete_service.phpPortType">
    <soap:binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                  transport="http://schemas.xmlsoap.org/soap/http"
                  style="document" />
    <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
               name="delete">
      <soap:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                      soapAction="http://www.lte.deti.ufc.br/dn/portal/
                      services/delete_service.php/delete"
                      style="document" />
      <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </input>
      <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </output>
    </operation>
  </binding>
  <service name="portal_services_delete_service.php">
    <port xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
          name="portal_services_delete_service.phpSOAPPort_Http"
          binding="tns:portal_services_delete_service.phpSOAPBinding">
      <soap:address xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                    location="http://www.lte.deti.ufc.br/dn/portal/
                    services/delete_service.php" />
    </port>
  </service>
</definitions>
```

- **index_service.wsdl**

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.wso2.org/php"
             xmlns:tnx="http://www.wso2.org/php/xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:http="http://www.w3.org/2003/05/soap/bindings/HTTP/"
             xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
             targetNamespace="http://www.wso2.org/php">
  <types>
    <xsd:schema elementFormDefault="qualified"
                targetNamespace="http://www.wso2.org/php/xsd">
      <xsd:element name="index">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="message" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="indexResponse">
        <xsd:complexType>
          <xsd:sequence>
```

```
            <xsd:element name="returnVal" type="xsd:anyType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <message name="index">
    <part name="parameters" element="tnx:index" />
  </message>
  <message name="indexResponse">
    <part name="parameters" element="tnx:indexResponse" />
  </message>
  <portType name="portal_services_index_service.phpPortType">
    <operation name="index">
      <input message="tns:index" />
      <output message="tns:indexResponse" />
    </operation>
  </portType>
  <binding name="portal_services_index_service.phpSOAPBinding"
           type="tns:portal_services_index_service.phpPortType">
    <soap:binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                  transport="http://schemas.xmlsoap.org/soap/http"
                  style="document" />
    <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
               name="index">
      <soap:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                      soapAction=" http://www.lte.deti.ufc.br/dn/portal/
                      services/index_service.php/index"
                      style="document" />
      <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </input>
      <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </output>
    </operation>
  </binding>
  <service name="portal_services_index_service.php">
    <port xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
          name="portal_services_index_service.phpSOAPPort_Http"
          binding="tns:portal_services_index_service.phpSOAPBinding">
      <soap:address xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                    location="http://www.lte.deti.ufc.br/dn/portal/
                    services/index_service.php" />
    </port>
  </service>
</definitions>
```

- **update_service.wsdl**

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.wso2.org/php"
             xmlns:tnx="http://www.wso2.org/php/xsd"
```

```
            xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:http="http://www.w3.org/2003/05/soap/bindings/HTTP/"
            xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
            targetNamespace="http://www.wso2.org/php">
<types>
  <xsd:schema elementFormDefault="qualified"
              targetNamespace="http://www.wso2.org/php/xsd">
    <xsd:element name="update">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="message" type="xsd:anyType" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="updateResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="returnVal" type="xsd:anyType" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="updateQuestion">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="message" type="xsd:anyType" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="updateQuestionResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="returnVal" type="xsd:anyType" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</types>
<message name="update">
  <part name="parameters" element="tnx:update" />
</message>
<message name="updateResponse">
  <part name="parameters" element="tnx:updateResponse" />
</message>
<message name="updateQuestion">
  <part name="parameters" element="tnx:updateQuestion" />
</message>
<message name="updateQuestionResponse">
  <part name="parameters" element="tnx:updateQuestionResponse" />
</message>
<portType name="portal_services_update_service.phpPortType">
  <operation name="update">
    <input message="tns:update" />
    <output message="tns:updateResponse" />
  </operation>
  <operation name="updateQuestion">
    <input message="tns:updateQuestion" />
    <output message="tns:updateQuestionResponse" />
  </operation>
</portType>
```

```
  <binding name="portal_services_update_service.phpSOAPBinding"
           type="tns:portal_services_update_service.phpPortType">
    <soap:binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                  transport="http://schemas.xmlsoap.org/soap/http"
                  style="document" />
    <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
               name="update">
      <soap:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                      soapAction=" http://www.lte.deti.ufc.br/dn/portal/
                      services/update_service.php/update"
                      style="document" />
      <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </input>
      <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </output>
    </operation>
    <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
               name="updateQuestion">
      <soap:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                      soapAction="http://www.lte.deti.ufc.br/dn/portal/
                      services/update_service.php/updateQuestion"
                      style="document" />
      <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </input>
      <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
        <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                   use="literal" />
      </output>
    </operation>
  </binding>
  <service name="portal_services_update_service.php">
    <port xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
          name="portal_services_update_service.phpSOAPPort_Http"
          binding="tns:portal_services_update_service.phpSOAPBinding">
      <soap:address xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                    location="http://www.lte.deti.ufc.br/dn/portal/
                    services/update_service.php" />
    </port>
  </service>
</definitions>
```

- **view_service.wsdl**

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.wso2.org/php"
             xmlns:tnx="http://www.wso2.org/php/xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:http="http://www.w3.org/2003/05/soap/bindings/HTTP/"
            xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
            targetNamespace="http://www.wso2.org/php">
<types>
  <xsd:schema elementFormDefault="qualified"
              targetNamespace="http://www.wso2.org/php/xsd">
    <xsd:element name="view">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="message" type="xsd:anyType" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="viewResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="returnVal" type="xsd:anyType" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</types>
<message name="view">
  <part name="parameters" element="tnx:view" />
</message>
<message name="viewResponse">
  <part name="parameters" element="tnx:viewResponse" />
</message>
<portType name="portal_services_view_service.phpPortType">
  <operation name="view">
    <input message="tns:view" />
    <output message="tns:viewResponse" />
  </operation>
</portType>
<binding name="portal_services_view_service.phpSOAPBinding"
         type="tns:portal_services_view_service.phpPortType">
  <soap:binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                transport="http://schemas.xmlsoap.org/soap/http"
                style="document" />
  <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
             name="view">
    <soap:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                    soapAction="http://www.lte.deti.ufc.br/dn/portal/
                    services/view_service.php/view"
                    style="document" />
    <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
      <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                 use="literal" />
    </input>
    <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
      <soap:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                 use="literal" />
    </output>
  </operation>
</binding>
    <service name="portal_services_view_service.php">
  <port xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
        name="portal_services_view_service.phpSOAPPort_Http"
        binding="tns:portal_services_view_service.phpSOAPBinding">
```

```
      <soap:address xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                    location="http://www.lte.deti.ufc.br/dn/portal/
                    services/view_service.php" />
    </port>
  </service>
</definitions>
```