UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CURITIBA
ESPECIALIZAÇÃO EM TECNOLOGIA JAVA E DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

NILSON ANDRÉ MACHNIEWICZ

ACOMPANHAMENTO DE RENDIMENTO ESCOLAR MOBILE

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2013

NILSON ANDRÉ MACHNIEWICZ

ACOMPANHAMENTO DE RENDIMENTO ESCOLAR MOBILE

Monografia de especialização apresentada ao curso de Especialização em Tecnologia Java e Desenvolvimento para Dispositivos Móveis da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de especialista.

Orientador: Prof.ª Dra. Maria Claudia Figueiredo Pereira Emer.

CURITIBA
2013

AGRADECIMENTOS

Agradeço a minha família que sempre esteve do meu lado me ajudando e apoiando em todos os momentos. Aos meus colegas de turma e aos professores, mestres e doutores que contribuíram para meu conhecimento. Também agradeço aos gestores da instituição de ensino que permitiram utilizar a ferramenta da escola para desenvolver minha análise apresentada nesse trabalho.

Os agradecimentos também se estendem aos meus colegas de trabalho que de maneira indireta contribuíram com opiniões e sugestões durante o processo de desenvolvimento.

RESUMO

Informatizando os processos das instituições de ensino é possível disponibilizar recursos com os quais são extraídas informações importantes para alunos, pais/responsáveis, professores e coordenadores. O objetivo deste trabalho é apresentar um protótipo de sistema e compatível com aparelhos *smartphones* que utilizam o sistema operacional *Android* e aprimorar o sistema web de uma escola de ensino fundamental para uma versão móvel. Alunos, pais, professores e coordenadores poderão consultar informações importantes a qualquer momento por meio de um aparelho celular em qualquer lugar. Com base no sistema *web* utilizado pela instituição de ensino, será apresentada uma avaliação de usabilidade, usando as técnicas de avaliação heurística e *checklist*. O protótipo do aplicativo móvel será apresentado levando em consideração as possíveis melhorias de interação usuário-sistema para possibilitar uma melhor usabilidade do *software*.

Palavras-chaves: *Android*. Mobilidade. Acompanhamento Escolar. Dispositivos Móveis. Usabilidade.

ABSTRACT

Computerizing the processes of educational institutions can provide resources that are extracted important information for students, parents/guardians, teachers and coordinators. The objective of this paper is to present a prototype system compatible with mobile devices smartphones that use the Android operating system and improve the system of a web elementary school for a mobile version. Students, parents, teachers and coordinators may consult important information at any time via a mobile device anywhere. Web based system used by the school, will be filed a usability evaluation, using the techniques of heuristic evaluation. The prototype of the mobile application will be presented taking into account the possible improvements of user-system interaction to enable better usability of the software.

Keywords: Android. Mobility. Monitoring School. Mobile devices. Usability.

LISTA DE ILUSTRAÇÕES

LISTA DE DIAGRAMAS

LISTA DE TABELAS

LISTA DE TELAS

LISTA DE INTERFACES WEB

LISTA DE ABREVIATURAS E SIGLAS

ADT - *Android Development Tools*
API - *Application Programming Interface*
APK - *Android Package File*
APP - *Application (Abreviação de Aplicação)*
DEX - *Dalvik Executable*
DOM – *Document Object Model*
HTTP - *Hypertext Transfer Protocol*
IDE - *Integrated Development Environment*
IHC - Interação Humano-Computador
JDK - *Java Development Kit*
JVM - *Java Virtual Machine*
OHA - *Open Handset Alliance*
PDA - *Personal digital assistant*
SDK - *Software Development Kit*
SOAP - *Simple Object Access Protocol*
UC - Use Case
XML - *Extensible Markup Language*
WSDL - *Web Services Description Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Ter as informações centralizadas em um sistema informatizado facilita e agiliza diversas atividades simples como notas dos alunos, matérias dos alunos, frequência dos alunos, trabalhos e avaliações, boletim e informações pessoais dos alunos e professores até atividades complexas como transferências, situação financeira, folha de pagamento dos funcionários e impressão de histórico escolar. Oferecer um completo apoio à secretaria da instituição de ensino, imprimindo todos os documentos (boletim, histórico escolar, ficha de aproveitamento individual, ata de resultados finais, diário de classe, etc.) de forma personalizada e tirando qualquer preocupação com relação aos prazos de entrega dos documentos na secretaria da instituição de ensino, além de fazê-lo em bom estilo é um benefício que pode ser alcançado com a informatização dos processos de uma secretaria da instituição de ensino. Esse é um cenário já encontrado em varias instituições de ensino, entre elas uma que será objeto de aperfeiçoamento.

Atualmente a escola conta com um sistema informatizado, em que essas informações são disponibilizadas para pais e/ou responsáveis, alunos e professores por meio da *web*, junto ao site da escola e mediante autenticação de informações. Usando essa massa de dados, é possível prover informações por meio de um *web service* que serão consumidas por um aplicativo móvel. Uma vez que o sistema seja alimentado pela secretaria da instituição de ensino, será possível disponibilizar informações como deveres de casa, notas, faltas, horário, situação financeira, ocorrências, boletim e avisos, acessíveis também em celulares, fazendo com que a instituição de ensino seja mais presente para pais/responsáveis e alunos.

Atualmente observa-se um crescimento em soluções para aplicativos móveis. Diversas empresas buscam incorporar aplicações móveis a seu dia-a-dia para agilizar seus negócios. As escolas também podem se beneficiar de soluções para aplicativos moveis.

Acompanhando o avanço da tecnologia este trabalho apresenta o protótipo da *app* para Acompanhamento de Rendimento Escolar – *ARE Mobile*. O propósito desse protótipo ARE *Mobile* é aperfeiçoar o acesso a informações da instituição de ensino, por meio de um aparelho *smartphone*. Com uma interface intuitiva e fácil de usar, o ARE *Mobile* será desenvolvido em plataforma *Android* e integrado a uma base de dados por meio de um *web service*. O ARE *Mobile* contará com um menu para os alunos, um menu para os pais, um menu para professores e também um menu para coordenação da instituição de ensino.

Neste trabalho, também será realizada a avaliação de usabilidade do sistema atual, para que os resultados dessa avaliação sejam utilizados no desenvolvimento do protótipo do aplicativo móvel como melhorias de interface. A interface é um dos elementos mais importantes para se obter um *software* de qualidade, pois o usuário qualifica-o de acordo com a parte visível do *software* e com a qual ele interage, ou seja, a interface. A área de Interação Humano-Computador (IHC) preocupa-se com a interação usuário-sistema e seus resultados práticos

para o projeto de interface humano-computador [1]. Esses estudos buscam o desenvolvimento de interfaces de qualidade, bem como a avaliação da sua usabilidade.

A usabilidade é sinônimo de facilidade de uso. Se o uso de um produto é fácil, o usuário tem maior produtividade: aprende mais rápido a utiliza-lo, memoriza as operações e comete menos erros [1]. A Avaliação de usabilidade deve ocorrer durante o desenvolvimento do *software*, assim, quanto mais cedo o problema é descoberto e reparado, menor será o custo das alterações necessárias ao sistema, consequentemente, oferecendo ao usuário uma interação de qualidade.

Por meio do aplicativo ARE *Mobile*, o acesso a essas informações pode acontecer em qualquer momento, seja ele durante o horário do almoço, preso no trânsito ou aguardando embarque num aeroporto. O aplicativo móvel não se destina apenas para alunos e pais/responsáveis. O aplicativo pode ser usado também por professores e coordenadores de instituição de ensino, sempre como ferramenta de consulta e/ou auxílio.

1.2 Objetivo

1.2.1 Objetivo Geral

O objetivo deste trabalho é desenvolver um protótipo de um aplicativo móvel para acompanhamento de rendimento escolar baseado em um sistema *web* já implantando na instituição de ensino.

1.2.2 Objetivos Específicos

Esse trabalho será um aperfeiçoamento do atual cenário da instituição, fornecendo assim uma ferramenta de consulta móvel para acompanhamento de rendimento escolar. Tem como objetivos específicos:

- Realizar um estudo sobre o sistema operacional *Android,* usabilidade em dispositivos móveis e *web service*;
- Realizar uma analise de usabilidade no sistema *web* da instituição de ensino propondo alguma melhoria de interface;
- Criar um *web service* que forneça informações em XML para consulta por meio de um *smartphone*;
- Criar um aplicativo em *Android* que busque e carregue todas essas informações XML de modo que o usuário possa utilizar para consultas em seu aparelho *smatphone*. Essas informações serão apresentadas no aparelho *smartphone* mediante validação de *login* e senha.

1.2.3 Justificativa

Os sistemas têm mudado a vida de muitas pessoas assim como as pessoas também tem modificado muito os sistemas. As interfaces têm um papel importante nesse aspecto. É por meio das interfaces que ocorre a interação do usuário com o computador. A área de Interfaces Humano-Computador é importante, pois uma mudança ou falha pode resultar em erros e, consequentemente, gerar resultados indesejados [14].

Seguindo os padrões de usabilidade *checklist* e avaliação heurística, foi feito uma releitura nas interfaces do sistema *web* atual da instituição de ensino e desenvolvido o protótipo de um *app*, o ARE *Mobile*. O desenvolvimento do ARE *Mobile* é um aperfeiçoamento do sistema atual o qual já fornece todas essas informações por meio da *web* site. Em sistemas *móveis, deve-se tomar mais cuidado com a interface, pois o meio de interação e o contexto são bem variados.* Nesses casos a usabilidade tornasse necessário para uma interface de qualidade.

1.2.4 Escopo e Delimitação do Trabalho

No universo do desenvolvimento para dispositivos móveis existem diferentes categorias de dispositivos como *Smartphones*, *PDAs* e *Tablets*. Existe também uma variada disponibilidade de plataformas de desenvolvimento para estes dispositivos, dentre as quais podemos citar *Android*, *iOS, Symbiam, webOS* entre outras. O fato do escopo deste projeto focar no desenvolvimento de um protótipo da *app* para funcionar em um dispositivo específico (s*martphone*) baseado em uma plataforma específica (*Android*) constitui-se em uma das restrições deste trabalho. Outra restrição é relacionada ao fato de que o a *app* resultante deste trabalho não pretende ser um aplicativo de software completo, limitando-se ao perfil de aluno. O *app* será uma ferramenta de consulta móvel para informações contidas em uma base de dados localizada na instituição de ensino.

1.2.5 Organização do Trabalho

Este trabalho está organizado como segue:

A Seção 2 aborda uma contextualização do sistema operacional *Android*, usabilidade e *web service*. Na Seção 3 será apresentado o Desenvolvimento do trabalho, abordando os resultados obtidos com a avaliação de usabilidade, o web service implementado, os protótipos de tela desenvolvidos e o ARE *Mobile*. Na Seção 4 é apresentada a validação do protótipo. Na Seção 5 apresenta a conclusão e algumas contribuições para trabalhos futuros ou mesmo melhorias no material apresentado nesse trabalho. E por fim, os apêndices, nos quais podem ser observadas telas do sistema Web atual, informações sobre a avaliação de usabilidade, código fonte *Java* referente ao *web service* e ARE *Mobile Android*.

2 REVISÃO BIBLIOGRÁFICA

2.1 Plataforma de Desenvolvimento - *Android*

São diversas as plataformas e linguagens disponíveis para a criação de aplicativos móveis cada qual com a sua peculiaridade. A Tabela 1 exibe as principais plataformas na atualidade.

	Android	iOS	Symbian	BlackBerry	Linux	Windows Mobile
Sistema Operacional	**Android**	**iOS**	**Symbian**	**BlackBerry**	**Linux**	**Windows Mobile**
Mantenedora	Google / OHA	Apple	Nokia	RIM	LiMo Foundation	Microsoft
Kernel	Linux	Mac OSX	EPOC	Blackberry OS	Linux	Windows
Desenvolvimento	Java, C, C++, Javascript	Objective C	C++, QT, Python, Ruby, .NET	JavaME, BB API	C, C++	C#
SDK	Sim	Sim	Sim	Sim	Não	Sim
Codigo Aberto	Sim	Não	Sim	Não	Para Membros	Não
Loja de Aplicativos Oficial	Android Market	App Store	OVI	App World	Não Disponivel	Market Place
Fabricante de aparelhos	Diversos	Apple	Diversos, Nokia	RIM	Diversos	Diversos

Tabela 1 - Sistemas Operacionais *Mobile* (adaptado[2])

Atualmente as atenções estão voltadas para o *Android*, que segundo pesquisa realizada pela *International Data Corporation* [3], apresentada na Tabela 2, o sistema operacional *Android* vem liderando a preferência do mercado e crescendo em larga escala, assumindo grande fatia do mercado de *smartphones* a nível mundial.

Baseado nestas pesquisas é possível identificar o quanto é promissor o mercado de desenvolvimento de aplicativos para a plataforma *Android*, e este foi um dos motivos da escolha desta plataforma para o desenvolvimento do projeto que será apresentado neste trabalho.

Ranking Top 6 Sistemas Operacionais Smartphones (Unidade em milhões)					
Sistema Operacional Mobile	**1Q12 Vendas Unitárias**	**1Q11 Vendas Unitárias**	**1Q11 Fatia do Mercado**	**1Q12 Fatia do Mercado**	**Crescimento Anual**
Android	89,9	36,7	36,10%	59,00%	145,00%
iOS	35,1	18,6	18,30%	23,00%	88,70%
Symbian	10,4	26,4	26,00%	6,80%	-60,60%
BlackBerry OS	9,7	13,8	13,60%	6,40%	-29,70%
Linux	3,5	3,2	3,10%	2,30%	9,40%
Windows Mobile	3,3	2,6	2,60%	2,20%	26,90%
Outro	0,4	0,3	0,30%	0,30%	33,30%
Total	152,3	101,6	100,00%	100,00%	49,90%

Tabela 2 - Venda Mundial de *Smartphones* - Milhões de Unidades (adaptado [3])

Outros motivos não menos importantes:

- É uma plataforma *open source*, baseada em um núcleo *Linux*.

- O desenvolvimento é baseado em *Java* que conta com uma grande comunidade de desenvolvedores, e possui documentação de acesso livre.

- Está disponível em uma vasta quantidade de marcas e modelos de dispositivos móveis, com várias opções de preços, podendo com um custo relativamente baixo ter acesso a um dispositivo para teste real da aplicação, sem ficar dependente apenas de emuladores.

2.1.1 O *Android*

Consiste em uma plataforma de desenvolvimento para aplicativos móveis, baseada em um sistema operacional *Linux*, com diversas aplicações já instaladas e, ainda, um ambiente de desenvolvimento bastante flexível [4]. Inicialmente desenvolvido por uma *startup* americana do Vale do Silício chamada *Android Inc*. Esta pequena empresa foi adquirida pelo *Google* no ano de 2005, que por sua vez tratou de amadurecer o projeto e o tornou público em meados de 2007 com o objetivo de apresentar a primeira plataforma open source de desenvolvimento para dispositivos móveis. Atualmente o *Android* é mantido por um grupo denominado *Open Handset Alliance* (OHA), que é formado por mais de 40 empresas das 20 quais figuram o próprio *Google* e outras de importância nos ramos de telefonia (Telefônica), fabricação de semicondutores (*Intel*) e fabricação de celulares (*Motorola*), dentre outras conforme Tabela 3.

Operadoras Telefonia	Fabricantes Semicondutores	Fabricantes Celulares	Fabricantes Softwares	Distribuidores
Telecom	AKM	acer	ACCESS	accenture
China Unicom	ARM	ASUS	andago	Aplix
vodafone	Atheros	CCI	Ascender Corporation	BORQS
中国电信	Audience	Alcatel	cooliris	sasken
Telefónica	Broadcom	Dell	eBay	Intrinsyc
中国移动	csr	FIH	Google	L&T Infotech
T-Mobile	Cypress	Fujitsu	Living Image	embinux
Bouygues	freescale	Garmin	Motoya	noser
Telus	gemalto	htc	myriad	Teleca
Sprint	intel	Huawei	Nuance	tat
SoftBank	Marvell	Kyocera	NXP Software	Wipro
docomo	MediaTek	lenovo	omron	Wind River
KDDI	MIPS	LG	pv	
	nVidia	NEC	skype	
	Qualcomm	Motorola	SONiVOX	
	Renesas	Pantech	SVOX	
	ST Ericsson	Samsung	VisualOn	
	Synaptics	Sharp		
	Texas Instruments	Sony Ericsson		
	VIA Telecom	Toshiba		
		ZTE中兴		

Tabela 3 - *Open Handset Alliance* [5]

2.1.2 A plataforma *Android*

O *Android* é a plataforma de desenvolvimento para aplicativos móveis como *smartphone* e contém um sistema operacional baseado em *Linux*, uma interface visual rica, GPS, diversas aplicações já instaladas e ainda um ambiente de desenvolvimento bastante poderoso e flexível. A linguagem *Java* pode ser utilizada para desenvolver as aplicações, usufruindo-se de todos os recursos nativos da linguagem. O fato de o *Android* ser de código aberto contribui para seu aperfeiçoamento, uma vez que desenvolvedores de todos os lugares do mundo podem contribuir para seu código-fonte, adicionando novas funcionalidades ou simplesmente corrigindo falhas.

Outro ponto forte do *Android* é que seu sistema operacional é baseado em *Linux*, e ele mesmo se encarrega de gerenciar memória e os processos. Isso permite que diversas aplicações possam ser executadas ao mesmo tempo, permitindo que aplicações em segundo plano consigam executar sem que o usuário perceba, enquanto ele esta acessando internet ou atendendo uma ligação. O *Android* oferece recursos visuais e aquelas funcionalidades que impressionam o usuário comum, e nesse quesito, o *Android* certamente atende as expectativas. A navegação de telas por meio da *touch screen* permite que o usuário tenha uma ótima experiência ao usar o aparelho.

O *Android* também tem suporte a gráficos 3D baseados na especificação 1.0 da *OpenGL ES* e, dessa forma, é possível criar jogos com uma qualidade excelente de resolução.

Toda a segurança do *Android* é baseada na segurança do *Linux*. No *Android* cada aplicação é executada em um único processo e cada processo por sua vez possui uma *thread* dedicada. Para cada aplicação instalada no celular é criado um usuário no sistema operacional para ter acesso a sua estrutura de diretórios. Dessa forma nenhum outro usuário pode ter acesso a essa aplicação

A linguagem *Java* é utilizada para construir as aplicações para o *Android*. O fato é que em seu sistema operacional não existe uma máquina virtual *Java (JVM)*. Em *Android* a maquina virtual chama-se *Dalvik,* que é otimizada para execução em dispositivos móveis. Ao desenvolver as aplicações para o *Android* será utilizado a linguagem *Java* e todos os seus recursos , mas depois que o *bycode* (*.class*) é compilado ele é convertido para o formato .dex, que representa a aplicação do *Android* compilada. Depois disso, os arquivos .dex e outros recursos como imagens são compactados em um único arquivo .apk, que representa a aplicação final, pronta para ser distribuída e instalada [4].

"O Android é a primeira plataforma para dispositivos móveis realmente aberta e abrangente. Ela inclui um sistema operacional, interface com o usuário e aplicações – todos os softwares necessários para rodar um telefone móvel, mas sem os obstáculos tem impedido a inovação da tecnologia móvel.

Andy Rubin, Diretor da Plataforma Móvel da Google, Novembro/2007 "

2.1.3 Visão Geral da Arquitetura

O *Android* é uma pilha de softwares para dispositivos móveis que inclui um sistema operacional, *softwares* intermediários e aplicações chaves conforme apresentado na Figura 1.

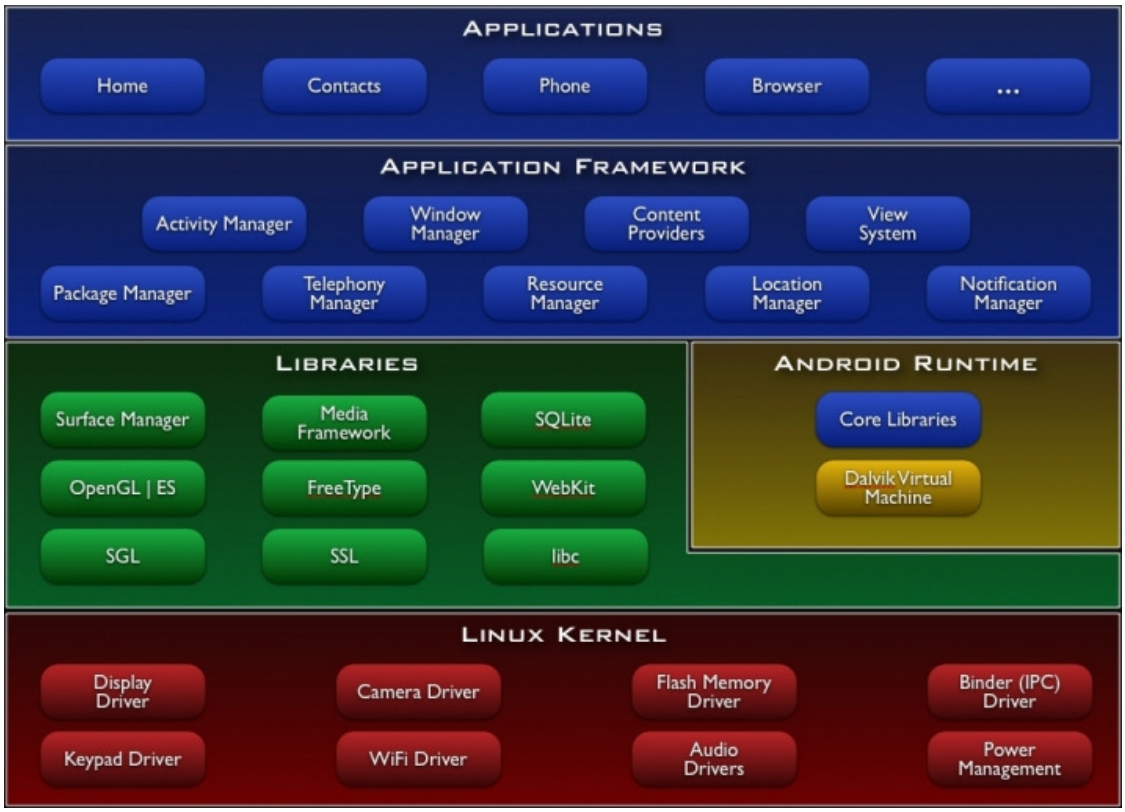


Figura 1 - Arquitetura *Android* [6]

2.1.3.1 Linux Kernel

O *Android* usa o *kernel* do *Linux* versão 2.6 para os serviços fundamentais como segurança, gerenciamento de memória, gerenciamento de processo, pilha de rede e *drives*. O *kernel* também atua como uma camada entre o *hardware* e o resto da pilha de *softwares* [4].

2.1.3.2 Libraries

O *Android* inclui um conjunto de bibliotecas C/C++ usadas por vários componentes do sistema *Android*. Estes recursos são expostos para os desenvolvedores por meio de *framework* de aplicação [4].

2.1.3.3 Android Runtime

As aplicações *Android* são executadas sobre uma máquina virtual chamada *Dalvik*, construída pelo engenheiro Dan Bornstein, cujo nome deve-se a uma pequena vila de pescadores da Islândia.

O *Android* incluir um conjunto de bibliotecas que oferecem as principais funcionalidades disponíveis na linguagem de programação *Java.* As aplicações são compiladas para o formato *Dalvik Executable* (.dex) e cada aplicação *Android* executa em um único processo com sua própria instância da máquina virtual *Dalvik*. A máquina virtual *Dalvik* depende do *Linux Kernel* para funcionalidades como gerenciamento de *threads* e memória [4].

2.1.3.4 Application Framework

O *framework* de aplicações é um conjunto de serviços e sistemas que, de forma padronizada, possibilita a criação de telas, acesso a dados, gerenciamento de recursos, telefonia, atividades, pacotes, localização e notificações. Por ser uma plataforma aberta, o *Android* dá livre acesso as mesmas *APIs* usadas no núcleo do sistema. A arquitetura foi desenvolvida para simplificar o reuso de componentes e permite ao usuário substituir as aplicações do sistema por versões de outro fornecedor [4].

2.1.4 Estrutura das Aplicações *Android*

Aplicações desenvolvidas em *Android* são baseadas em uma arquitetura de componentes chaves demonstrada pela Figura 2, porém, não necessariamente uma aplicação deve obrigatoriamente utilizar-se de todos estes componentes, no geral as aplicações são compostas por uma combinação destes. Em conjunto com estes componentes existe um arquivo XML denominado *AndroidManifest.xml* de existência obrigatória, e no qual são feitas configurações gerais da aplicação e dos componentes utilizados por ela. Juntam-se a esta estrutura dois outros itens importantes que fazem estes quatro componentes chaves funcionarem que são as *Intents* e as *Views*.

Figura 2 - Componentes de uma aplicação *Android* [7]

Activities são as representantes das telas da aplicação. Associada a uma *activity* normalmente existe uma *view*, que define como será feita a exibição visual para o usuário. As *activities* são responsáveis por gerenciar os eventos de tela e também coordenam o fluxo da aplicação.

Os ***Services*** são códigos que executam em segundo plano. Normalmente são utilizados para tarefas que demandam um grande tempo de execução.

Os ***Content Providers*** (provedores de conteúdos) são a maneira utilizada pela plataforma para compartilhar dados entre as aplicações que executam no dispositivo. Um exemplo bem claro disto é a aplicação de gerenciamento de contatos do *Android*, que é nativa. Aplicações desenvolvidas por terceiros podem utilizar um *content provider* a fim de ler os contatos armazenados no dispositivo de forma simples.

Os ***Broadcast Receivers*** são componentes que ficam "escutando" a ocorrência de determinados eventos, que podem ser nativos ou disparados por aplicações. Uma aplicação pode, por exemplo, utilizar um *broadcast receiver* para ser avisada quando o dispositivo estiver recebendo uma ligação e, com base nessa informação, realizar algum tipo de processamento.

Junto os estes componentes, existe o arquivo de manifesto *AndroidManifest.xml*. Ele é obrigatório e único para cada aplicação. É nele que são feitas as configurações gerais da aplicação e dos componentes que fazem parte dela. E, juntando tudo isto, existe a figura do ***Android Core****,* que na verdade não é um componente específico, mas sim a plataforma Android propriamente dita. É ele quem proporciona a interação entre os componentes e as aplicações e torna possível a execução do código [7].

Intents são mensagens responsáveis por ativar os componentes *Service*, *Activity* e *BroadcastReceiver* de uma aplicação. Essas mensagens são utilizadas para facilitar a ligação entre os componentes da aplicação ou de aplicações diferentes, em tempo de execução.

Views são elementos utilizados para definir objetos gráficos exibidos na tela, com o objetivo de prover interação com o usuário. Exemplo destes elementos são botões, caixas de diálogo, mapas entre outros.

2.1.5 Ciclo de Vida

O ciclo de vida de um aplicativo é representado pelos seguintes estados [4]: executando, temporariamente interrompida em segundo plano ou completamente destruída. O importante é entender que o sistema operacional cuida desse ciclo de vida, mas ao desenvolver aplicações é importante levar em consideração cada possível estado para garantir uma aplicação robusta.

Segundo a documentação do *Android* há três subníveis do ciclo principal, que por sua vez ficam se repetindo durante a execução da aplicação. Esses três ciclos são chamados de *entire lifetime, visibile lifetime* e *foreground lifetime*.

Entire Lifetime: Este ciclo ocorre apenas uma vez e define o tempo de vida completo de uma *Activity*. Ele acontece entre as chamadas do método *onCreate(bundle)* e *onDestroy()*, os quais são chamados uma única vez.

Visible Lifetime: Ocorre entre os métodos *onStart()* e *onStop()*. Durante esse período é possível visualizar a *Activity* na tela, mas ela pode não estar em primeiro plano e interagindo com o usuário, ou seja, pode ocorrer situações onde outra *Activity* esteja no topo da pilha de execução. Uma *Activity* só ocupa o topo da pilha de execução quando o método é executado.

Foreground Lifetime: Este ciclo ocorre entre os métodos *onResume()* e *onPause()* e, durante esse tempo, a *Activity* está no topo da pilha de execução e interagindo com o usuário.

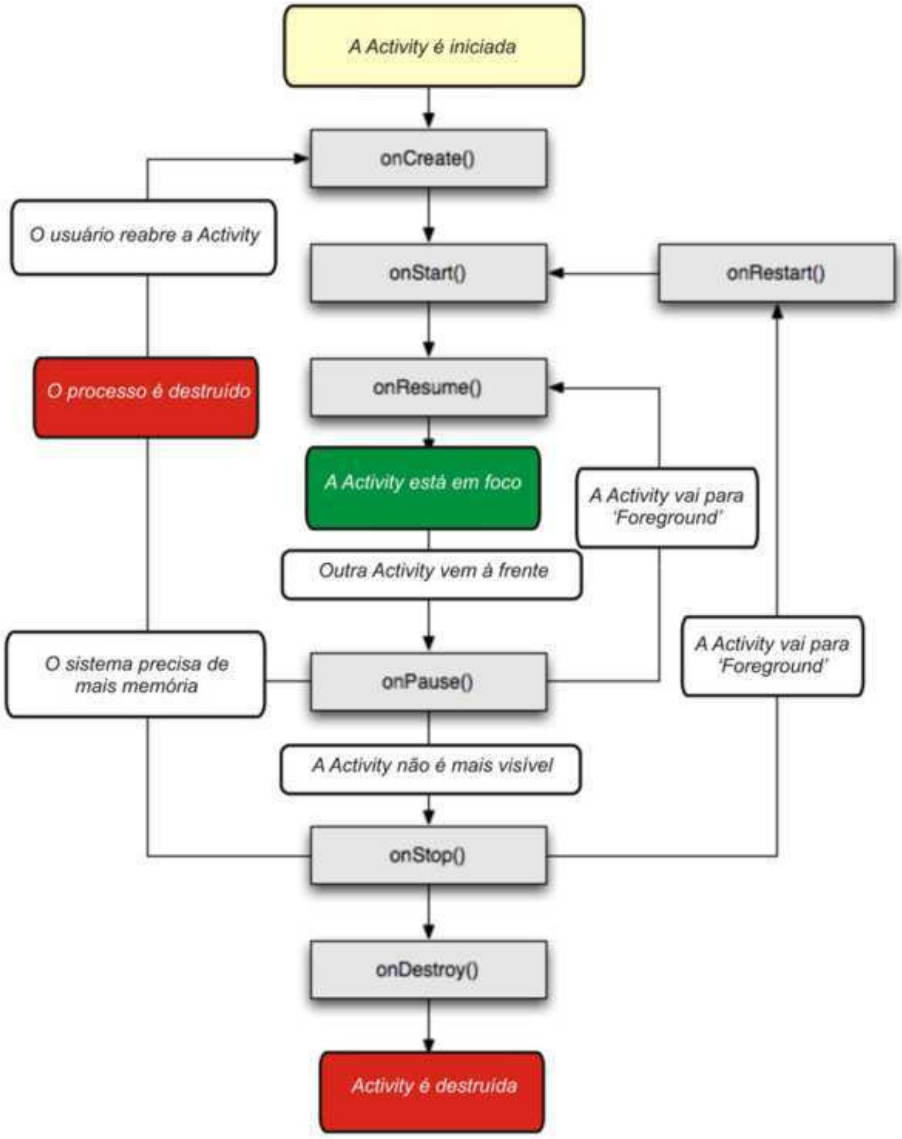


Figura 3 - Ciclo de vida de uma *activity* [8]

A *Activity* possui métodos que auxiliam o desenvolvedor a controlar o estado da aplicação, são eles: *onCreate(), onStart(), onRestart(), onResume(), onPause(), onStop()* e o *onDestroy()*.

Vamos à definição de cada método [8].

onCreate – Executado apenas 1 vez quando a *Activity* é iniciada, logo após sua execução o método *onStart()* é chamado.

onStart() – Pode ser executado após o *onCreate* ou *onRestart()*, após sua execução o método *onResume* é chamado.

onResume() – É chamada quando a *activity* está executando. É chamado após o *onStart()*.

onRestart() – Esse método é chamado quando a *Activity* estava no estado de interrompida em segundo plano e volta a ser executada.

onPause() – É executado quando a *Activity* é interrompida, seja pelo recebimento de uma ligação, um *SMS*, etc.

onStop() – É chamado quando uma *Activity* está sendo encerrada, por exemplo com uma chamada a outra *Activity* ou ao método *finish().* Caso a *Activity* seja reiniciada, o método *onResume()* será executado.

onDestroy – Aqui a *Activity* é encerrada. Ele é executado quando o sistema operacional necessita liberar memória ou quando o método *finish()* é executado.

2.2 Usabilidade

A usabilidade é definida por Pressman [9] como: “O esforço para aprender, operar, preparar a entrada e interpretar a saída de um programa”. A usabilidade é o esforço necessário, de um determinado conjunto de usuários, para utilização e julgamento de um software. Também pode ser entendida com a preocupação com a interação do usuário em um sistema por meio da interface, fazendo parte da Área de Interfaces Humano-Computador.

A norma *ISO* 9241 define usabilidade como uma medida, na qual um produto pode ser utilizado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso [10]. Segundo a Associação Brasileira de Normas Técnicas – ABNT [10], é por meio da incorporação de características e atributos capazes de beneficiar os usuários em um contexto particular de uso que ocorre uma melhor usabilidade. É muito importante medir a usabilidade de um produto para poder visualizar a complexidade das interações entre o usuário, os objetivos, as características da tarefa e os outros elementos do contexto de uso. Um mesmo produto, quando usado em diferentes contextos, apresenta diferentes níveis de usabilidade.

2.2.1 Usabilidade *Mobile*

A Usabilidade nos dispositivos móveis não tem o mesmo peso que a usabilidade *desktop*, seu peso é muito, mas muito maior. É importante enfatizar que em frente ao *desktop* o usuário tem foco total, ele senta em frente ao seu computador, abre a aplicação e começar a mexer, ou mesmo começa a navegar na *internet*, ele pode minimizar fazer outra atividade e então voltar. Já no *mobile* não é a mesma coisa, o foco do usuário está prejudicado. Ele geralmente está fazendo outra atividade principal, assistindo tv, comendo com os amigos em algum restaurante, almoço, em algum transporte público, e por aí vai. O seu celular é foco secundário.

Portanto, a importância de se ter uma boa usabilidade para prender o foco do usuário não é algo importante, é algo vital para a sua aplicação.

Fazer com o que o usuário baixe já é o primeiro esforço, porém, o segundo será realmente ter a atenção do usuário e isso está muito ligado a usabilidade do *app* [11].

2.2.2 Recomendações da Usabilidade *Mobile*

Desenvolver sites e aplicações para *mobile* requer atenção para alguns critérios que tem um grande impacto na forma com que as pessoas interagem com estes dispositivos [12].

2.2.2.1 Reduzir *clicks*

Esta parece ser uma recomendação óbvia para ambiente *mobile*. Em um projeto usual de interface, as melhores práticas indicam que seria mais adequado disponibilizar toda a informação necessária em uma única tela e poupar cliques do usuário. Deixar o conteúdo mais conciso é crucial para que a informação possa ser apresentada de modo objetivo e o menos fragmentada possível [12].

2.2.2.2 Reduzir funcionalidades

Restringir a quantidade de funcionalidades, mantendo as que são necessárias ao ambiente mobile, diminui a chance dos usuários se confundirem diante de todas as possibilidades e opções oferecidas.

2.2.2.3 Reduzir conteúdo

Devido ao tamanho das telas, o conteúdo para *mobile* exige uma carga cognitiva maior e, portanto, pode ser até duas vezes mais difícil de compreender. Como a memória de curto prazo é fraca, quanto mais os usuários tiverem que “rolar a tela” para se lembrar de um conteúdo, menos eles o farão.

2.2.2.4 Dar escolhas ao usuário

Textos mais concisos e funcionalidades mais restritas são necessários. Mas é importante manter um *link* para a versão convencional do site, caso o usuário precise acessar algum recurso que não esteja na versão *mobile.* O usuário deve ter o direito de escolha sobre como ele deseja visualizar o site.

2.2.2.5 Outras práticas importantes que herdamos da usabilidade convencional

- **Integridade estética**

O quanto o *design* da sua aplicação se integra com a função da mesma. É o casamento entre forma e função, interface com boa qualidade estética e funcional.

- **Consistência**

A consistência de interface permite que o usuário transfira seus conhecimentos e habilidades de uso de uma aplicação para outra. É preciso frisar que uma aplicação consistente não é aquela que copia outras aplicações. Pelo contrário, é uma aplicação que tira proveito dos padrões e paradigmas de interface com os quais as pessoas se sentem mais confortáveis durante a interação.

- **Metáforas**

Fácil reconhecimento e memorização de palavras, símbolos e imagens.

- **Contexto do Usuário**

Especificação do ambiente do usuário, incluindo também a modelagem e análise de tarefa e objetivos de negócio.

- **Modelo Mental**

Organização apropriada de dados, funções, tarefas, papéis e pessoas de acordo com o modo com que o usuário compreende e reconhece estes elementos.

- **Navegação**

Navegação adequada considerando o modelo mental por meio de janelas, menus, caixas de diálogos e painéis de controle em formato compreensível.

- **Interação e *Feedback***

Input efetivo e *feedback* do *output* de informações para assegurar ao usuário que uma ação está em processamento.

- **Aparência e *Design***

Qualidade visual e atenção ao *design* com relação à escala, proporção, ritmo, simetria e balanceamento de componentes.

- **Visualização de Informações**

Apresentação de informações por tabelas, gráficos, mapas e diagramas. Uma vez que a tela destes dispositivos ainda é pequena em comparação aos computadores comuns (mesmo se tratando de *tablets*), é preciso se valer de componentes coringas que são capazes de

apresentar uma boa quantidade de informação de modo compacto, conciso, de fácil visualização e acessível.

- **Convenções de interface e interação em plataformas**

Compatibilidade de ícones, comandos e ações não é uma diretriz recente em ergonomia de *software*. Mas em ambientes *mobile*, este é um ponto que ainda não está bem adequado. Basta observarmos como cada plataforma disponibiliza convenções distintas de interface, sem contar a variação de navegadores, que geralmente são nativos de cada plataforma, o que acaba "amarrando" o usuário.

- **Comportamento do usuário no ambiente**

Dispositivos móveis geralmente não são utilizados em um ambiente estável, como uma mesa. O ambiente externo onde o usuário se encontra influencia a sua interação com o sistema. É dia ou noite? O usuário está parado ou em movimento? Com as duas mãos livres? Ocupado? Todos estes fatores externos têm impacto na interação e é da alçada da equipe de desenvolvimento analisar todas estas variáveis para compor o cenário de uso da sua aplicação. O ambiente do usuário influencia na qualidade da interação.

2.2.3 Avaliação da Usabilidade

Para Rocha [14], a avaliação da usabilidade deve ocorrer durante todo o processo de desenvolvimento do *software* e os resultados devem ser utilizados para melhorar a interação entre usuário e o sistema.

A avaliação de usabilidade é o método de analisar a facilidade de uso do sistema e se ela cumpre os requisitos dos usuários. Ela deve acontecer conforme perfil do usuário e resultará nos três aspectos de usabilidade: eficiência, eficácia e satisfação.

Para avaliação de usabilidade, deve-se considerar qual técnica ser utilizada, quem serão os avaliadores e em que nível de projeto se encontra. *Cybis [15]* dividiu as técnicas em três grupos:

- Técnicas Prospectivas: baseadas na aplicação de questionários e entrevistas que servem para avaliar a satisfação do usuário.

-Técnicas Preditivas ou Diagnosticadas: preveem erros de projeto sem a participação direta do usuário. São classificadas em: avaliações analíticas, nas quais a técnica é aplicada nas primeiras etapas de desenvolvimento da interface; avaliação heurística consiste em realizar uma inspeção do sistema, detectado os problemas de interação e já fornecem possíveis soluções; e *checklist*, técnica baseada em lista de verificação, por meio das quais, qualquer profissional faz o diagnostico rápido de problemas gerais e repetitivos das interfaces.

- Técnicas Objetivas ou Empíricas: baseiam-se diretamente com o usuário, por meio de ensaios de interações e sistema de monitoramento.

Como a instituição de ensino já possui sistema implantado, a ferramenta ARE *Mobile* será um aperfeiçoamento, as técnicas escolhidas para fazer a avaliação foram: avaliação heurística e *checklist* por ser baseado em uma *ISO*.

2.2.3.1 Avaliação heurística

Conforme Rocha [14], a avaliação heurística é um método básico de inspeção da interface. Ele serve para encontrar os problemas de usabilidade nas interfaces para que esses possam ser atendidos durante a etapa de projeto ou reprojeto.

A avaliação heurística deve ser feita por, no mínimo, dois avaliadores, pois é difícil uma única pessoa conseguir identificar todos os problemas de usabilidade que o sistema apresenta. Cada avaliador examina a interface várias vezes, inspecionam os elementos de diálogo e faz comparações com os princípios de usabilidade. Para ajudar a identificar os problemas, os avaliadores utilizam regras heurísticas como as Heurísticas de Nielsen [14]:

1 – Visibilidade do status do sistema: O usuário tem de estar informado sobre o que está acontecendo no sistema, por meio de *feedback*, dentro de um determinado tempo;

2 – Compatibilidade do sistema com o mundo real: O sistema deve utilizar a linguagem que o usuário conhece e não a linguagem técnica;

3 – Controle do usuário e liberdade: Quando o usuário escolher por engano uma função, o sistema deve mostrar, de maneira clara e rápida, a forma de voltar para estado anterior;

4 – Consistência e padrões: Seguir as convenções de plataforma computacional utilizada;

5 – Prevenção de erros: Desenvolver um bom *design* para prevenir o erro antes de ele acontecer;

6 – Reconhecimento ao invés de relembrança: Tornar objetos, ações e opções visíveis para que o usuário não precise lembrar informações de uma parte ou outra do diálogo;

7 – Flexibilidade e eficiência de uso: O sistema tem que oferecer aos usuários mais experientes possibilidades de acelerar ações frequentes;

8 – Estética e projeto minimalista: os diálogos devem apresentar apenas informações relevantes e a visibilidade deve ser favorecida;

9 – Ajuda aos usuários para reconhecer, diagnosticar e corrigir erros: As mensagens de erro devem ser expressas na linguagem do usuário, haverá indicação do problema e, construtivamente, a solução para uma solução.

10 – Ajuda e documentação: O sistema deve fornecer ajuda e documentação de forma acessível e não muito extensa.

Após os avaliadores aplicarem as regras heurísticas, as listas de problemas encontrados devem ser consolidadas em uma única lista. A avaliação é feita individualmente e dura, em média, duas horas. Depois do trabalho individual, o ideal é fazer uma reunião para discutir os problemas que cada avaliador encontrou no sistema. Cada um apresenta um relatório, contendo a localização do problema, a heurística violada, a explicação do problema, possíveis soluções e a gravidade. A gravidade do problema é definida pela seguinte escala [14]:

0 – Não concordo que isto seja um problema.

1 – Problema cosmético: não precisa ser consertado a menos que haja tempo extra no projeto.

2 – Problema pequeno: o conserto desse é desejável, mas deve receber baixa prioridade.

3 – Problema grande: deve ser consertado. Deve receber alta prioridade.

4 – Catastrófico: é imperativo consertar o problema antes do lançamento do produto.

Como qualquer outro método da engenharia de usabilidade, a avaliação heurística não garante resultados perfeitos e nem o encontro de todos os problemas de usabilidade da interface. Ela é mais utilizada na fase de projeto, porém, não substitui uma avaliação mais completa nas próximas etapas de desenvolvimento do *software*.

2.2.3.2 Checklist

As inspeções por *checklist* constituem uma técnica de avaliação capaz de identificar vários problemas gerais e repetitivos da interface. Conforme Cybis [15], as vantagens de avaliação realizada por meio de *checklist* são: rapidez na aplicação, causando redução de custo da avaliação; facilidade de identificação de problema de usabilidade, devido à especificidade das questões do *checklist*, sistematização da avaliação, que garante resultados mais estáveis, mesmo quando aplicada separadamente por diferentes avaliadores e, para realizar a avaliação, não há necessidade de profissionais especializados em ergonomia, pois o conhecimento ergonômico está embutido no próprio *checklist*.

As inspeções de usabilidade por *checklist* são desenvolvidas por meio de uma lista de questões e podem estar acompanhadas de glossários explicativos, que servem para esclarecer dúvidas relacionadas às questões. O instrumento oficial de inspeção ergonômica é a norma internacional *ISO* 9241. Ela é composta por um conjunto de 18 partes, cada uma tratando de diferentes aspectos do trabalho em escritório informatizados.

Para essa técnica, foi utilizada a de *checklistergolist*. Esta ferramenta foi criada pelo LabiUtil – Laboratório de Utilizabilidade da Universidade Federal da Santa Catarina – UFSC [16] e é baseada na norma *ISO* 9241 – parte 11. Ela possui como objetivo conceber, projetar, desenvolver e disponibilizar, via *web*, a ferramenta para a avaliação autônoma da facilidade de uso de dispositivos de software interativo.

A *ergolist* oferece varias recomendações, sendo cada uma delas especializada em um aspecto ou critério que determina a ergonomia das Interfaces Humano-Computador. São 18 as recomendações ergonômicas da *ergolist* que são definidas como segue:

- Presteza: verifica a compatibilidade do sistema com as expectativas e necessidades do usuário em sua tarefa. É composta por 17 questões.

- Agrupamento por localização: verifica se a distribuição espacial dos itens traduz as relações entre as informações. É composta por 11 questões.

- Agrupamento por formato: verifica os formatos dos itens como meio de transmitir associações e diferenças. É composta por 17 questões.

- *Feedback*: avalia a qualidade do *feedback* imediato às ações do usuário. É composto por 12 questões.

- Legibilidade: verifica a legibilidade das informações apresentadas nas telas do sistema. É composta por 27 questões.

- Concisão: verifica o tamanho dos códigos e termos apresentados e introduzidos no sistema. É composto por 14 questões.

- Ações Mínimas: verifica a extensão dos diálogos estabelecidos para realização dos objetivos do usuário. É composto por 5 questões.

- Densidade Informacional: avalia a densidade informacional das telas apresentadas pelo sistema. É composta por 9 questões.

- Ações Explicitas: verifica se é o usuário quem comanda, explicitamente, as ações do sistema. É composta por 4 questões.

- Controle do Usuário: avalia as possibilidades de o usuário controlar o encadeamento e a realização das ações. É composta por 4 questões.

- Flexibilidade: verifica se o sistema permite personalizar as apresentações e os diálogos. É composto por 3 questões.

- Experiência do Usuário: avalia se usuários, com diferentes tipos de experiência, têm iguais possibilidades de obter sucesso em seus objetivos. É composta por 6 questões.

- Proteção contra erros: verifica se o sistema oferece as oportunidades para o usuário prevenir eventuais erros. É composta por 7 questões.

- Mensagens de erro: avalia a qualidade das mensagens de erro enviadas aos usuários em dificuldades. É composta por 9 questões.

- Correção de erros: verifica as facilidades oferecidas para que o usuário possa corrigir erros cometidos. É composta por 5 questões.

- Consistência: avalia se é mantida uma coerência no projeto de códigos, telas e diálogos com o usuário. É composta por 11 questões.

- Significados: avalia se os códigos e denominações são claros e significativos para os usuários do sistema. É composta por 12 questões.

- Compatibilidade: verifica a compatibilidade do sistema com as expectativas e necessidades do usuário em sua tarefa. É composta por 21 questões.

Cada questões da *ergolist* contém com resposta opções de “sim”, “não”, “não aplicável” e “adiar resposta”, oferecendo, também, um espaço para o usuário inserir comentários, quando for o caso, sobre a questão.

2.3 Web Service

Define-se *Web Service* como a disponibilização de um serviço que pode ser acessado por meio da *internet*. Representa uma lógica de negócio bem definida permitindo a interação com clientes, os quais enviam requisições bem definidas e recebem respostas

síncronas, na qual o cliente fica bloqueado esperando o serviço completar sua execução, e assíncronas não havendo necessidade de esperar o término da execução.

Serviços expostos como *Web Services* tem lógicas de negócio que devem encapsular regras ou funcionalidades que representem regras de negócio. Devem ser publicados e acessados por intermédio de uma linguagem padrão de publicação e um protocolo específico de comunicação.

Um *Web Service* possui uma estrutura interna fracamente acoplada, ou seja, a implementação do serviço pode mudar a qualquer momento sem afetar sua utilização pelo cliente, e, ao adotar esse tipo de tecnologia, conseguimos fazer com que os sistemas sejam mais bem modularizados e atinjam níveis mais adequados de integração. Suporta chamadas remotas de procedimento (RPC), o que permite que o cliente invoque as operações de objetos remotos usando um protocolo baseado em XML, e cada operação deve, opcionalmente, expor parâmetros de entrada e definir seu tipo de retorno [18].

2.3.1 Componentes de um *web service*

Protocolo HTTP: protocolo padrão para transmissão de dados pela *internet*.

XML: formato padrão para troca de informações. Possui sintaxe rigorosa e leveza no armazenamento de dados. Principal elemento da tecnologia de *Web Services*.

Simple Object Acess Protocol(SOAP): prove uma estrutura padrão de empacotamento para transporte de documentos XML pela *internet*.

Web Service Description Language (WSDL): tecnologia XML que descreve de forma padronizada a interface de um *Web Service*. Determina como são representados os parâmetros de entrada e saída de uma chamada externa, a estrutura das funções, a natureza da chamada (somente entrada / entrada e saída) e o protocolo de conexão com o serviço. Também define a forma como os clientes interagem e executam os serviços.

***Universal Description, Discovery, and Integration(UDDI)*:** descreve um registro mundial de serviços e serve como integração, propaganda e descoberta de serviços. Pode-se utilizar este registro para descobrir serviços disponíveis na *internet* [18].

A Figura 4 exemplifica o funcionamento de um *web service*:

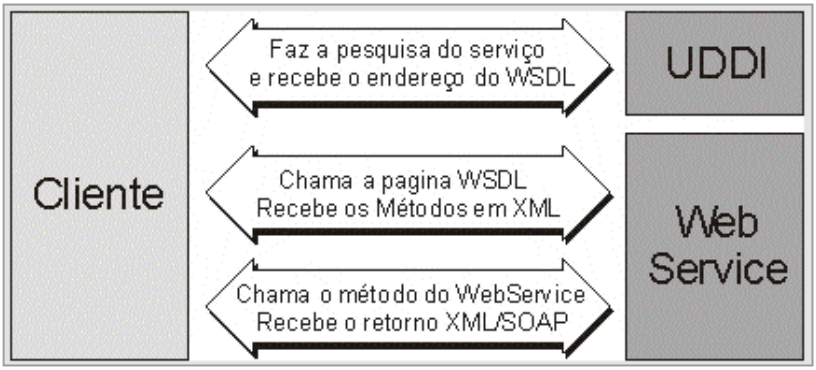


Figura 4 - Funcionamento do *Web Service* [19]

3 DESENVOLVIMENTO

3.1 AVALIAÇÃO DE USABILIDADE *WEB*

As técnicas de Avaliação Heurística e *CheckList* foram aplicadas ao sistema *web* da instituição de ensino. A avaliação heurística foi preenchida por três pontos de vista diferentes e podem ser consultadas no APÊNDICE C – Avaliação de Usabilidade nas Figura 43,Figura 44 eFigura 45.

Como essa avaliação heurística visa apontar possíveis melhorias de usabilidade e as questões tratam informações mais técnicas, as avaliações foram realizadas por analistas e programadores. Na Figura 5 é possível visualizar o resultado final obtido por meio das avaliações heurísticas.

Resultado Avaliação - Perfil Aluno							
	Login Aluno	Avisos Aluno	Boletim Aluno	Materiais Aluno	Provas Aluno	Horarios Aluno	Conteúdo Aluno
Avaliador 01 - Analista de Sistemas	3	7	6	9	9	9	7
Avaliador 02 - Programador Pleno	5	9	11	15	12	8	8
Avaliador 03 - Programador Júnior	2	5	8	13	7	5	5
Total	10	21	25	37	28	22	20

Figura 5 - Resultado Avaliação - Perfil Aluno

Cada falha ou melhoria identificada pelo avaliador foi contabilizado e ao final apresentado um total. Quanto maior for a pontuação, mais falhas foram identificadas.

Com base no resultado é possível apontar quais as telas precisam ser melhoradas ou corrigidas.

Com o uso da ferramenta *ErgoList* [16] e levando em consideração a interface da instituição de ensino, foi obtido resultado apresentado no APÊNDICE C – Avaliação de Usabilidade na Figura 46.

3.1.1 Conclusão sobre a avaliação de usabilidade

Após executar a avaliação de usabilidade nas interfaces da instituição de ensino usando os métodos de avaliação heurística e *checklist,* é possível afirmar que existem alguns aspectos de melhorias de interface. Também foi possível identificar algumas falhas pontuais consideradas graves, porém não devendo refletir no desenvolvimento do *app mobile*.

Mesmo sendo uma ferramenta de consulta, onde as informações são extraídas facilmente, foi possível observar que:

- A interface não apresenta mensagens de confirmação, o que pode causar certa dúvida para o usuário ao realizar uma consulta.

- Ao tentar acessar informações como Rematrícula, a mesma não funciona e apresenta um erro de acesso ao banco de dados.

- Os erros não são tratados pela interface de maneira customizada. Eles são apresentados contendo informações técnicas, dificultando o entendimento.

Para desenvolvimento do protótipo, o resultado da avaliação foi considerado e as melhorias identificadas foram aplicadas a fim de oferecer uma boa usabilidade.

3.2 DESENVOLVIMENTO DO *WEB SERVICE* – ARE *MOBILE*

Para desenvolvimento do *web service* foi utilizado o *framework Apache Axis*. O *Apache Axis* é um *framework* de código aberto, baseado na linguagem *Java* e no padrão XML, utilizado para construção de *web services* no padrão *SOAP.* Por meio do *Axis* os desenvolvedores podem criar aplicações distribuídas. Além da versão para *Java*, existe uma implementação baseada na linguagem C++. O projeto *Apache Axis* é suportado pela *Apache Software Foundation*.

O Axis disponibiliza dois modos para "expor" os métodos de uma classe por meio de *web services*. O modo mais simples utiliza os arquivos *JWS (Java Web Service)* do *Axis*. O outro método utiliza um arquivo *WSDD (Web Service Deployment Descriptor)*, que descreve com detalhes como serão criados os *web services* a partir dos recursos (classes *Java*) existentes.

Também é possível, por meio do *Axis*, gerar automaticamente o arquivo *WSDL (Web Service Description Language)*. O WSDL contém a definição da interface dos *web services* [20]. A Figura 6 mostra a interface gráfica do WSDL desenvolvido para utilização do ARE *Mobile.*

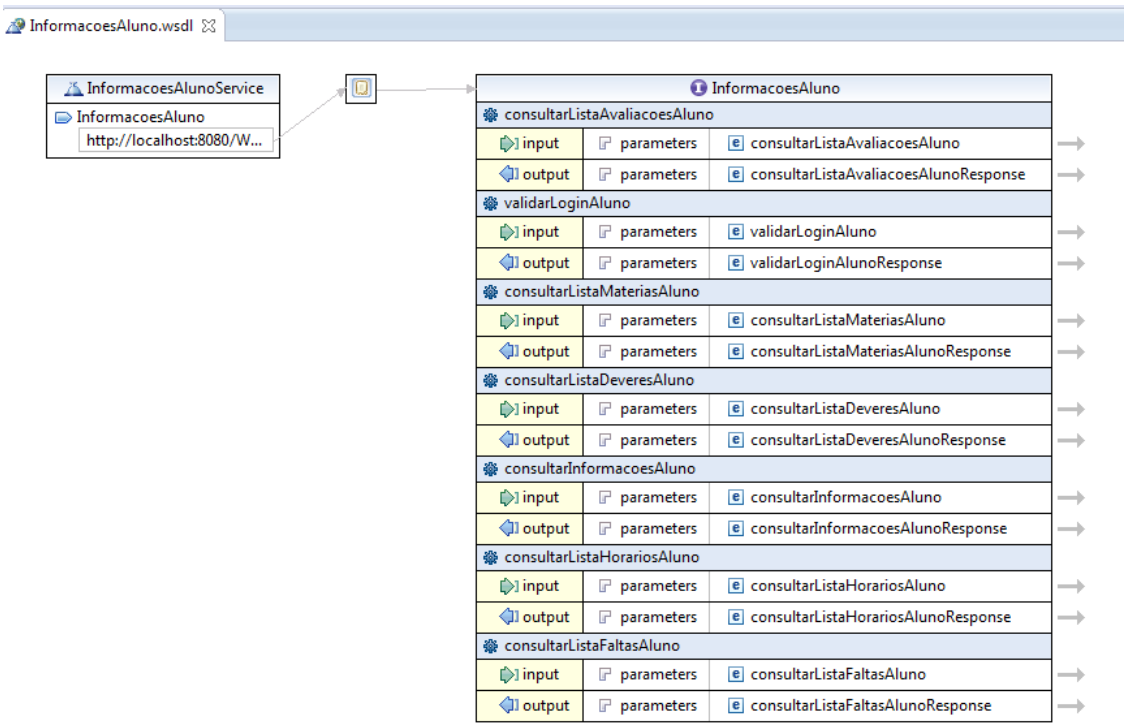


Figura 6 - *Design WSDL* Desenvolvido

A Figura 7 mostra parte da estrutura criada para desenvolvimento do *web service*.

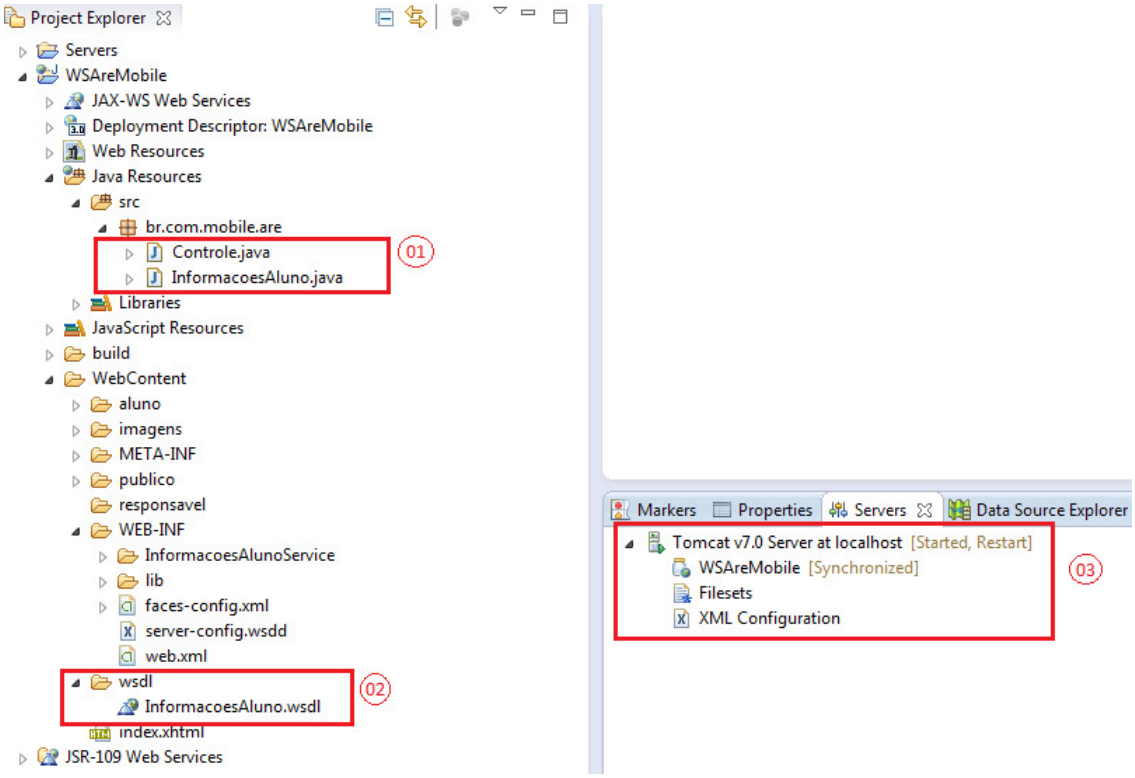


Figura 7 - Estrutura Eclipse para *Web Service*

Os destaques marcados na Figura 7 representam:

1. Classes responsáveis pela conexão com banco de dados e obtenção das informações;
2. Arquivo WSDL gerado para *web service*;
3. Servidor *Web TomCat* (*container*) para publicação do *web service*.

Para realizar os testes com *web service* do ARE *Mobile* foi desenvolvido uma interface conforme apresentada nas Figura 8 e Figura 9.

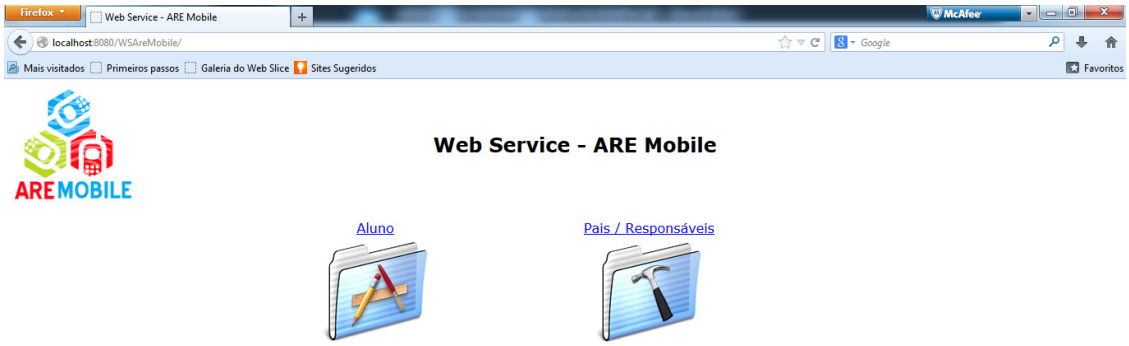


Figura 8 - Interface Principal *Web Service*

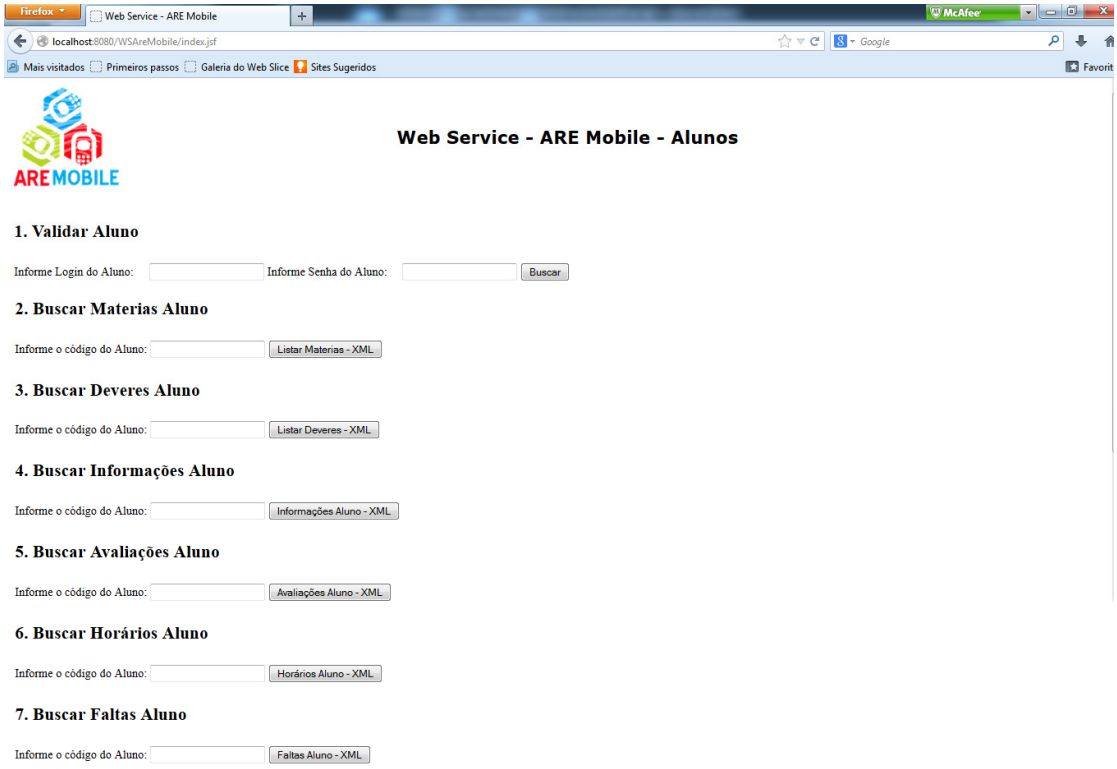


Figura 9 - Interface *Web Service* – Aluno

O *web service* está publicado e disponível para consulta no seguinte endereço:

HTTP://189.11.37.163/WSAreMobile

No Apêndice F são apresentados mais detalhes da implementação *Java* referente ao web service.

3.3 PROTOTIPAÇÃO

3.3.1 Protótipo de Tela Principal

O protótipo de tela Principal do ARE Mobile apresenta as opções de navegabilidade do *web app*. É a primeira interface que o usuário encontrará ao iniciar o aplicativo em seu dispositivo móvel. Nela temos quatro opções de acesso, onde cada uma representa um perfil diferente de usuário.

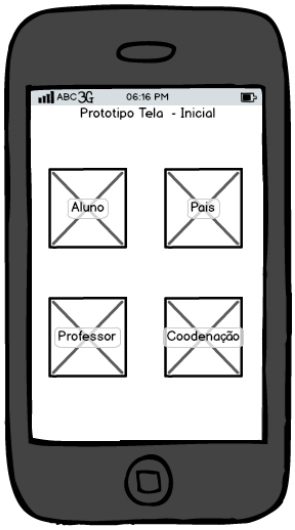


Figura 10 – Protótipo Tela Principal ARE *Mobile*

3.3.2 Protótipos de Telas Aluno:

Nos protótipos de telas do Aluno ARE *Mobile*, temos seis possíveis telas. Após autenticação na tela de *Login*, o aluno conseguirá consultar informações de seu desempenho escolar. De maneira simples e objetiva, o aluno consegue extrair informações referentes a deveres de casa registrados pelos professores, nota de avaliações já realizadas, data de futuras avaliações, horários de aulas e frequência.

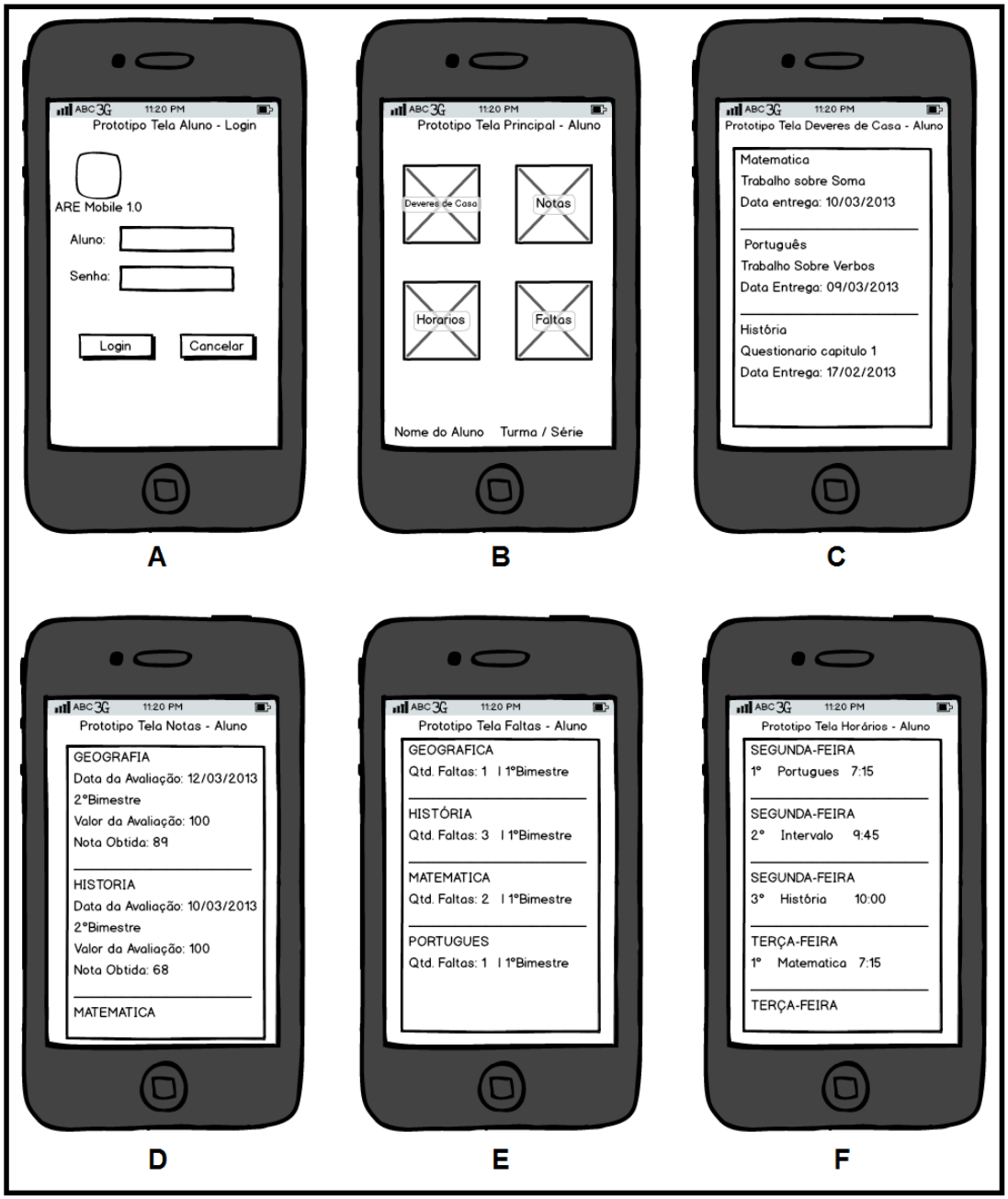


Figura 11 - Protótipos de Telas Perfil Aluno

3.3.3 Protótipos de Tela Professor

Nos protótipos de tela do Professor ARE *Mobile*, após autenticação, temos a possibilidade de consultar as informações e avisos encaminhados pelos coordenadores e pedagogos. Também é possível consultar os protocolos entregues na secretaria. Toda vez que os professores entregam um caderno de notas ou médias da turma, é registrado um protocolo que formaliza essa entrega junto à secretaria da instituição de ensino.

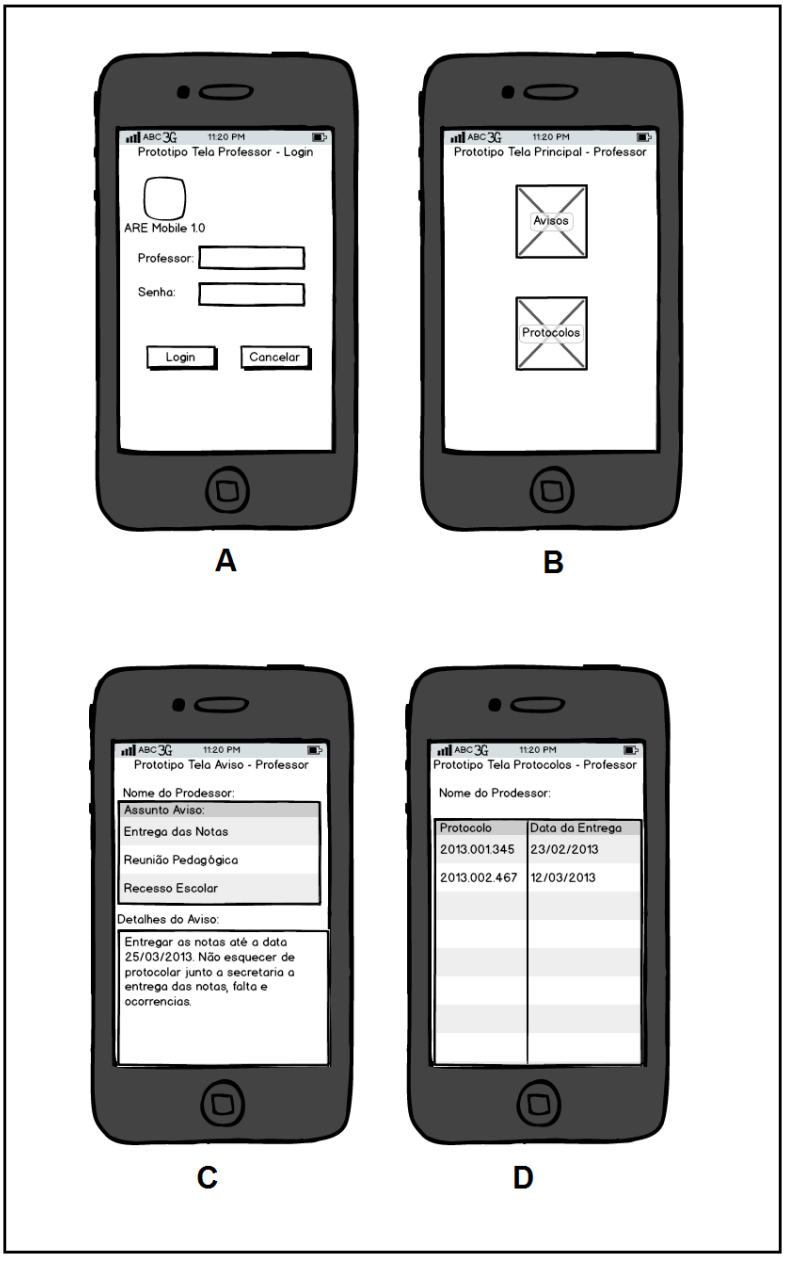


Figura 12 - Protótipos Telas Perfil Professor

3.3.4 Protótipo de Tela Pais / Responsáveis

Após a autenticação dos pais/responsáveis, o ARE *Mobile* oferece a opção de consulta de avaliações, situação financeira, ocorrências, boletim, horários e avisos. Na consulta de avaliações é possível verificar a nota do aluno e comparar se a nota ficou dentro ou abaixo da média geral da turma.

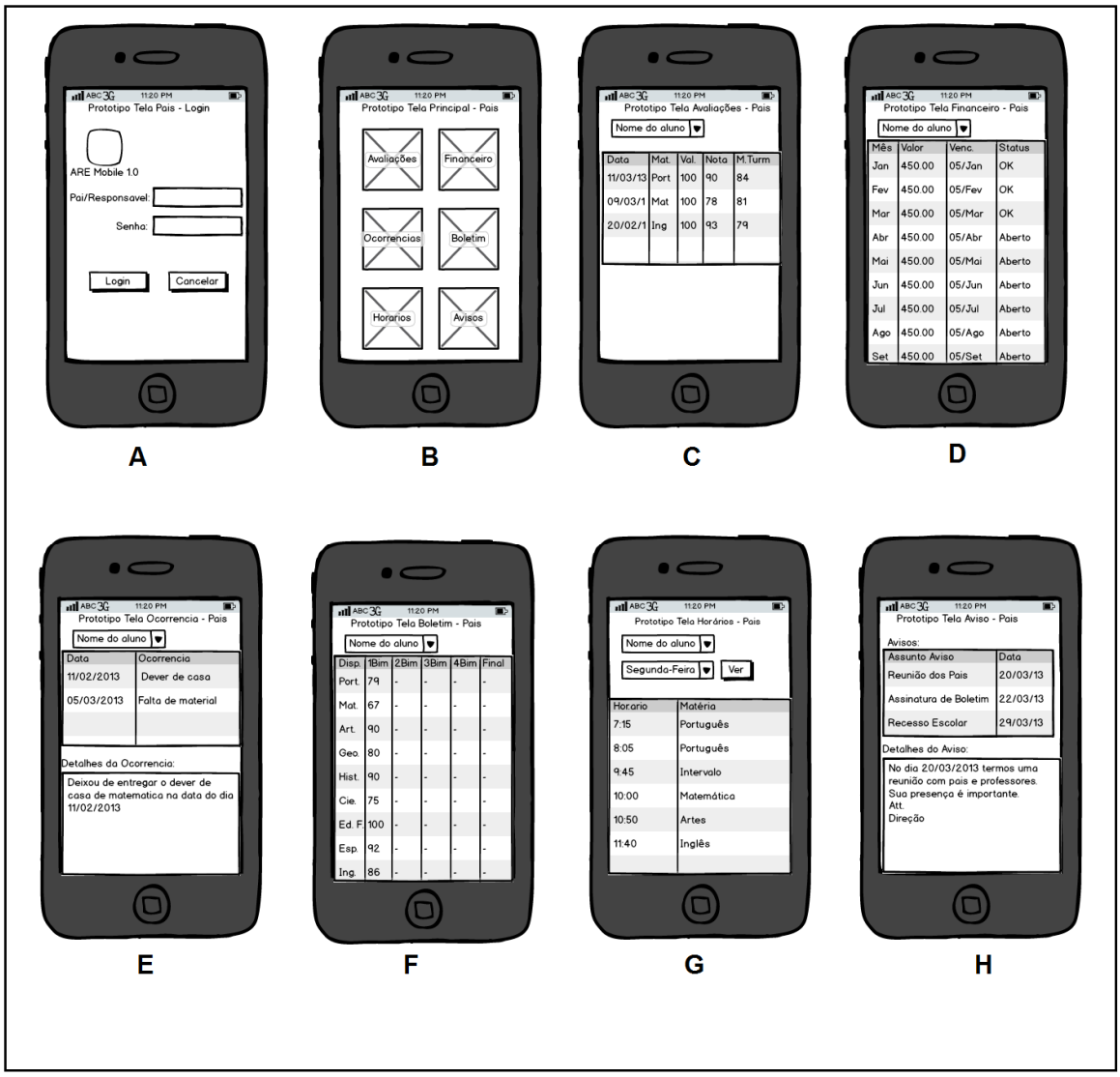


Figura 13 - Protótipos de Telas Perfil Pais/Responsáveis

3.3.5 Protótipo de Tela Coordenação

A coordenação da instituição de ensino precisa ter acesso a informações de todos os alunos, bem como avaliações, ocorrências, situações financeiras e boletim. Depois de passar pela autenticação, a coordenação poderá ter acesso a essas informações de maneira rápida e simples usando o ARE *Mobile*.

Figura 14 - Protótipos Telas Perfil Coordenação

3.4 DESENVOLVIMENTO ARE *MOBILE*

3.4.1 Descrição do Problema

Usando o sistema atual da instituição de ensino, é possível acompanhar o desempenho escolar dos alunos por meio de um *web site*. Com base nas interfaces do atual sistema *web* da instituição, é apresentada uma avaliação de usabilidade utilizando avaliação heurística e *checklist*. Com base nessas avaliações de usabilidade, foi apresentado o protótipo de um aplicativo móvel chamado ARE *Mobile*. Esse protótipo do aplicativo móvel será desenvolvido para *smartphone* que fazem uso do sistema operacional *Android* e é uma ferramenta a mais para a instituição de ensino aproximar os pais/responsáveis da vida estudantil dos alunos.

O ARE *Mobile* tem acesso às informações por meio de um *web service*. Foi implementado um *web service* e publicado na *web*, de forma que por meio de uma URL seja possível obter as informações necessárias a quatro perfis diferentes de usuários: aluno, pais/responsáveis, professores e coordenadores.

ARE *Mobile* oferece para os pais/responsáveis um meio mais prático de se comunicar com a escola e acompanhar o rendimento/resultado de seus filhos. Por meio desta *app* os pais podem verificar a sua situação financeira, verificar as notas, faltas, ocorrências e tarefas de casa dos seus filhos, além de receber comunicados diretamente da secretária da instituição de ensino por meio de avisos. Dessa forma fica muito mais prático, principalmente para aqueles pais que vivem viajando ou que não têm tempo para ir à escola acompanhar diretamente a vida escolar dos seus filhos.

Os alunos podem verificar seus deveres de casa, suas notas e faltas sem ter que ficar perguntando para os professores e anotando. Por meio do ARE *Mobile* os alunos podem acessar os horários e suas matérias diariamente, oferece muito mais controle sobre sua própria vida estudantil.

Para os professores o ARE *Mobile* oferece a possibilidade de verificar os avisos encaminhados pela coordenação da instituição de ensino. Outra funcionalidade interessante é consultar os protocolos registrados junto à secretaria. Sempre que o professor entrega as notas na coordenação, será registrado um protocolo formalizando a entrega dessas notas e esse protocolo poderá ser consultado pelo aplicativo móvel.

O principal problema de ser coordenador é ter que fazer a “ponte” entre pais e professores e muitas vezes entre os pais e os próprios filhos. O coordenador precisa ter sempre os dados à mão para poder se reportar aos pais dos alunos com as informações de que precisa tais como ocorrências em que o aluno esteja envolvido, notas de provas, situação financeira e boletim. O ARE *Mobile* facilita a vida dos coordenadores, trazendo-lhes a informação certa no momento certo.

3.4.2 Requisitos

3.4.2.1 Requisitos Não Funcionais

- O protótipo do ARE *Mobile* foi desenvolvido para funcionar em *smartphones* com plataforma *Android*.
- Como o *app* Are *Mobile* necessita consultar informações em um *web service* publicado na *internet*, para seu funcionamento é necessário que o *smartphone* tenha acesso a *internet*.
- O *web service* será a "ponte" entre o *smartphone* com *Android* e o repositório de dados da instituição de ensino, como demonstra a Figura 15, o seu funcionamento é necessário para obter as informações.

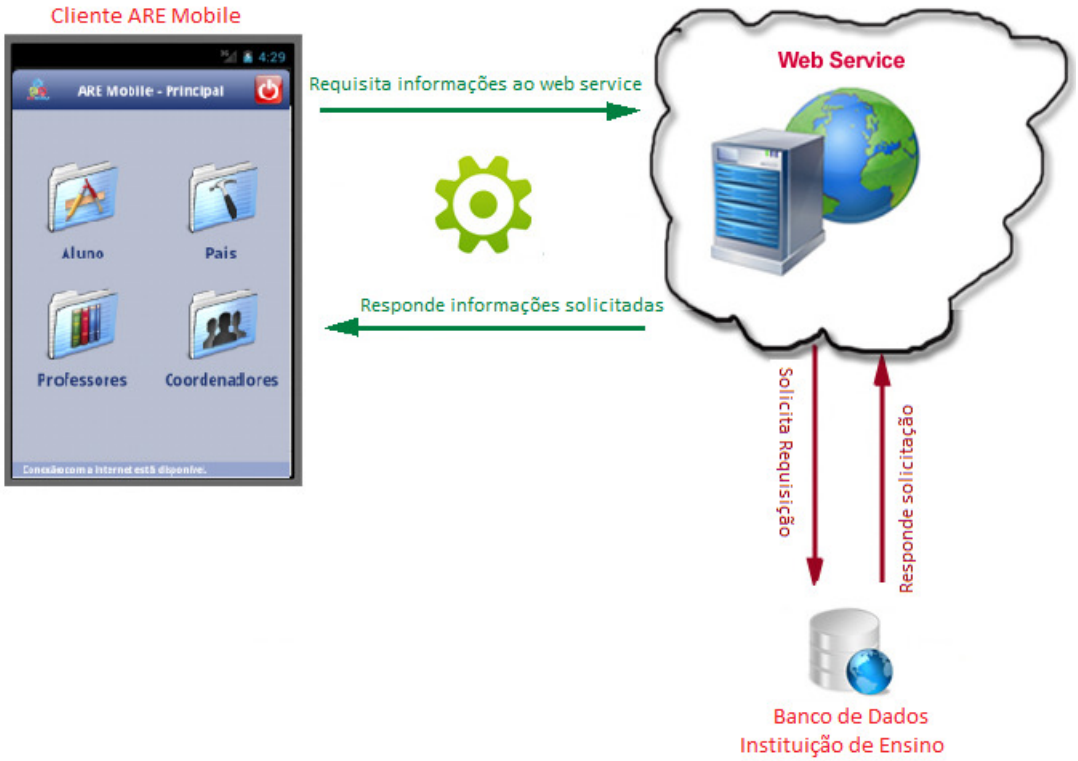


Figura 15 - Funcionamento Cliente ARE *Mobile* - *Web Service* - Banco de Dados

3.4.2.2 Requisitos Funcionais

- O protótipo visa disponibilizar informações para alunos devidamente matriculados na instituição de ensino, pais e/ou responsáveis, professores e coordenadores pedagógicos.
- Para realizar as consultas usando o *app* ARE *Mobile*, é necessário validação de usuário e senha. Feita essa validação de usuário e senha, o ARE *Mobile* será

capaz de listar informações para os perfis alunos, responsáveis, professores e coordenadores.

- As informações serão consultadas em um repositório de dados que a instituição de ensino mantém e apresentadas pelo ARE *Mobile*.

Uma vez que esses requisitos sejam atendidos, é possível acompanhar o rendimento escolar do aluno como, por exemplo: notas, faltas, situação financeira, boletim, avaliações e horário escolar.

É importante ressaltar que o ARE *Mobile* é uma ferramenta de apoio, que visa prover informações apenas para consulta, portanto todas as informações fornecidas não poderão ser alteradas ou excluídas por meio do *app*.

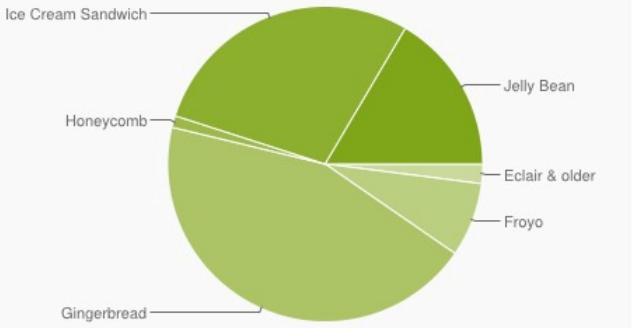
3.4.3 Ambiente de Desenvolvimento

Para o desenvolvimento de aplicativos em *Android* é necessário que esteja configurado um ambiente contendo a última versão do *Java Development Kit* (JDK) juntamente com o *Android* SDK. Também é altamente recomendável que se utilize uma IDE para a codificação. Uma IDE indicada é o Eclipse [21], o qual será utilizada nesse projeto.

Para o desenvolvimento do protótipo ARE *Mobile* apresentado nesse trabalho, foi utilizada como IDE o Eclipse versão 4.3 denominado *Kepler.* O Eclipse é uma IDE de código aberto que possibilita a instalação de diversos *plugins*, entre eles o ADT que é um *plugin* que estende as funcionalidades do Eclipse, possibilitando a criação de projetos voltados para *Android*.

O projeto *Android* foi implementado utilizando AVD (*Android Virtual Device)* com API *Level* 10 que representa a versão 2.3.3 do *Android*, denominada *Gingerbread* e que domina grande fatia no mercado, como mostra a Figura 16.

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	1.9%
2.2	Froyo	8	7.6%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	44%
3.1	Honeycomb	12	0.3%
3.2		13	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	28.6%
4.1	Jelly Bean	16	14.9%
4.2		17	1.6%



Data collected during a 14-day period ending on March 4, 2013

Figura 16 - Versão *Android* [22]

3.4.4 Caso de Uso

Nesta seção serão apresentados os diagramas de casos de uso relacionados ao perfil aluno e ao perfil pais/responsáveis. Os outros diagramas de casos de uso referentes ao perfil professores e perfil coordenadores não foram desenvolvidos nesta etapa do trabalho. As especificações detalhadas dos casos de uso relacionados ao perfil do aluno e ao perfil de pais ou responsáveis podem ser observadas no Apêndice D e Apêndice E.

3.4.4.1 Caso de Uso Perfil Aluno

Este caso de uso tem como objetivo autenticar o aluno e após validação, disponibilizar as opções: deveres de casa, avaliações, faltas e horários do aluno autenticado. Sua representação gráfica pode ser vista no Diagrama 1.

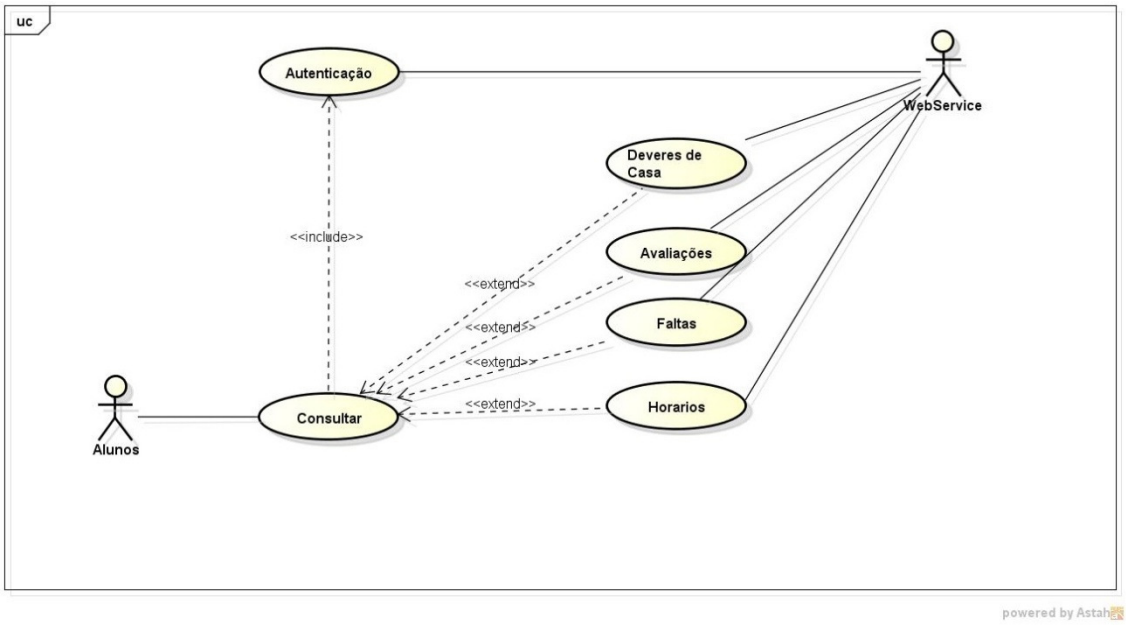


Diagrama 1 - Caso de Uso Perfil Aluno

No Apêndice D serão apresentados detalhes de cada UC que compõem o caso de uso do perfil aluno.

3.4.4.2 Caso de Uso – Perfil Pais / Responsáveis

Este caso de uso tem como objetivo autenticar o usuário pais ou responsável e após validação disponibilizar para consulta informações com avaliações, situação financeira, ocorrências, horários, boletim e avisos.

Sua representação gráfica pode ser vista no Diagrama 2.

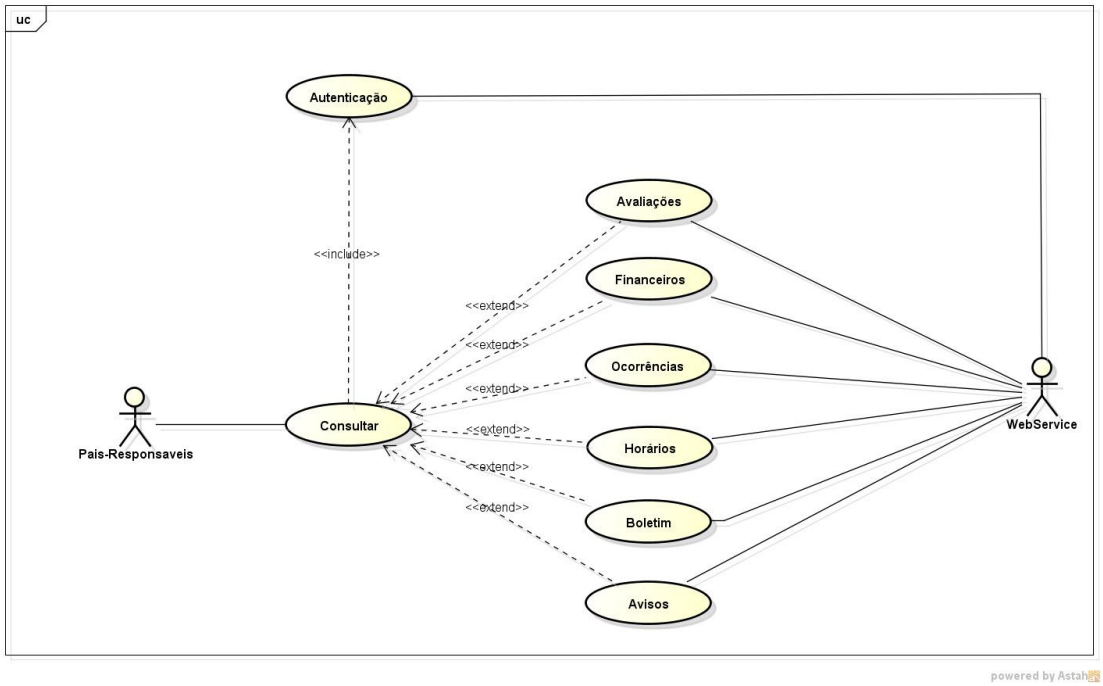


Diagrama 2 - Caso de Uso Perfil Pais/Responsáveis

No Apêndice E serão apresentados detalhes de cada UC que compõem o caso de uso do perfil pais/responsáveis.

3.4.5 Diagrama de Classes

O diagrama de classes é uma representação da estrutura e das relações das classes que servem de modelo para objetos. É uma modelagem muito útil para o desenvolvimento de sistemas, pois define todas as classes que o sistema necessita possuir [25].

O Diagrama 3 representa as classes existentes no sistema principal da instituição, o qual fornecerá as informações para o *web service* e para o *app* ARE *Mobile*.

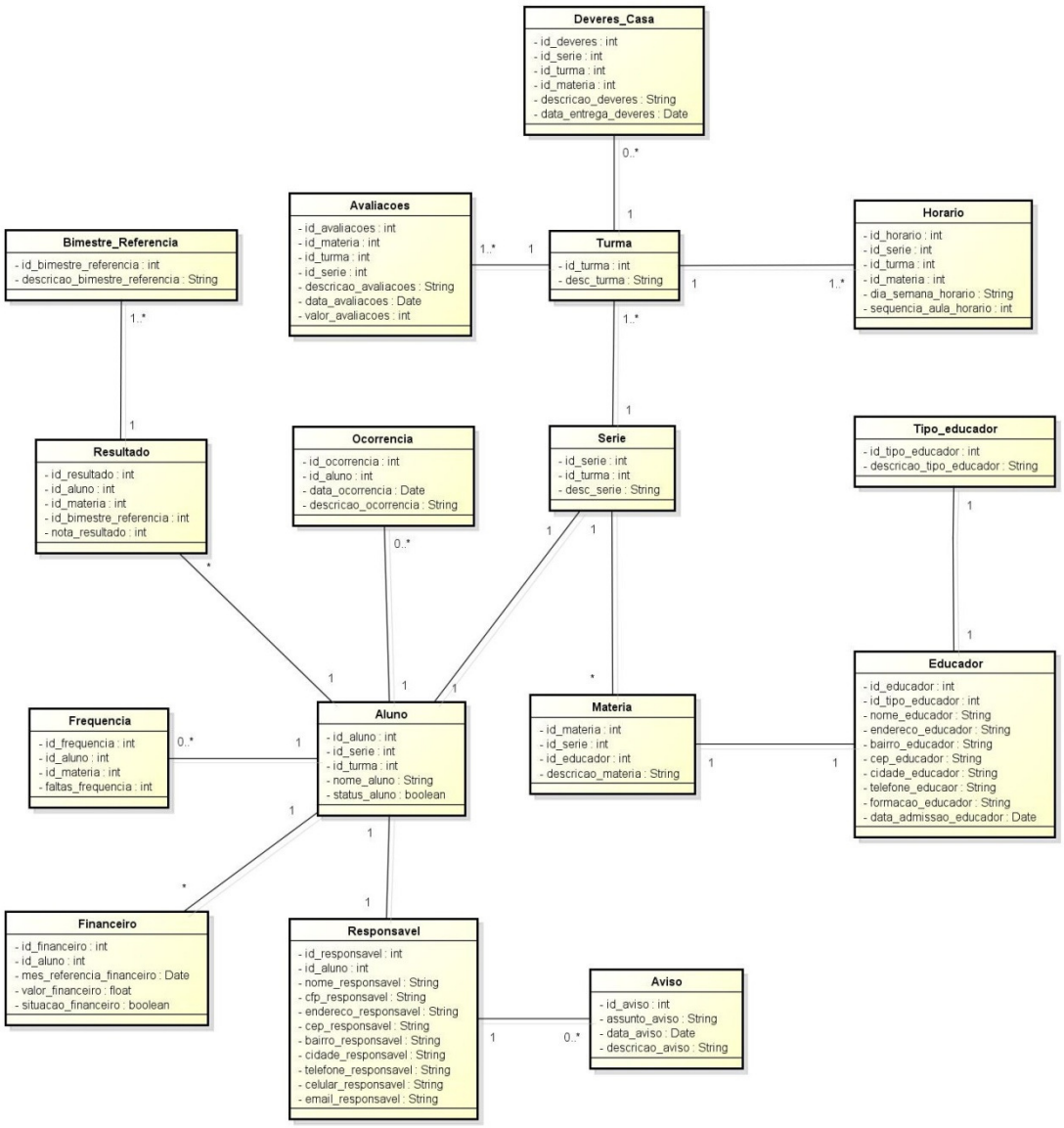


Diagrama 3 - Diagrama de Classes - ARE *Mobile*

3.4.6 Interface

O protótipo do ARE *Mobile* – Perfil Aluno é composto pelo total de seis telas as quais cada uma representa uma atividade (*activity*) no *Android*.

Ao acessar o protótipo do ARE *Mobile*, será exibida a tela principal do *app* conforme *Activity* ARE *Mobile* 1.

Activity* ARE *Mobile* 1 - Tela Principal ARE *Mobile

Na *Activity* ARE *Mobile* 1 - Tela Principal ARE *Mobile* é possível inicializar as seguintes consultas:

- Consulta Perfil Aluno;
- Consulta Perfil Pais / Responsáveis;
- Consulta Perfil Professores;
- Consulta Perfis Coordenadores.

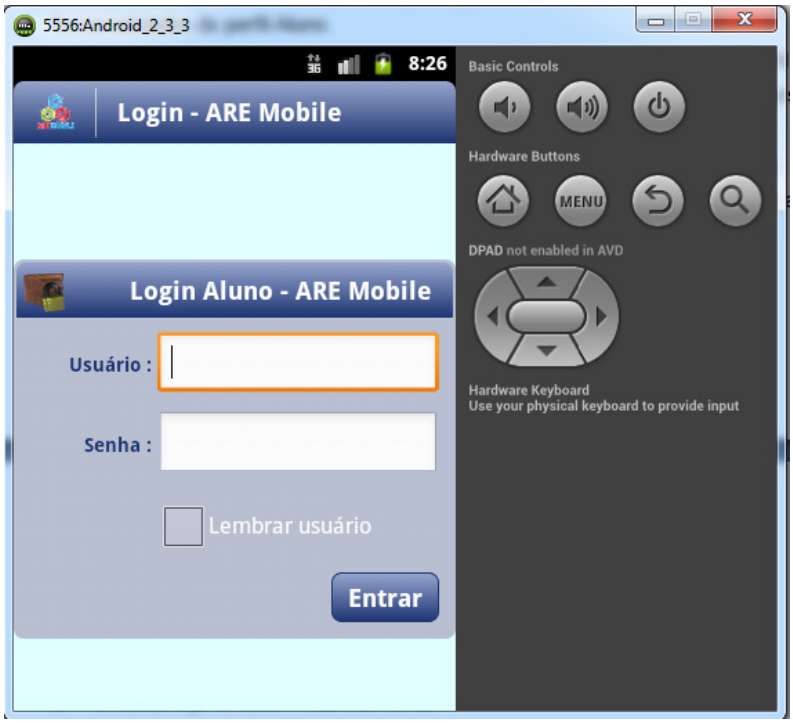
Como o ARE *Mobile* utiliza internet para acessar *web service* e exibir as consultas, na tela principal é exibido o *status* da conexão com internet.

Coforme apresentado no item 1.2.4 - Escopo e Delimitação do Trabalho, esse trabalho não visa o desenvolvimento total da *app* ARE *Mobile*. Nessa seção serão apresentadas as funcionalidades do perfil Aluno.

Ao ser acionado o botão Aluno, será apresentado a Tela de *Login* do Perfil Aluno conforme *Activity* ARE *Mobile* 2. Nela o aluno deverá informar dois parâmetros para realizar obter o acesso:

- *Login*: O *login* do aluno é representado pelo código de matrícula.

- Senha: A senha para validação do aluno é composta por apenas números que correspondem à data de nascimento do mesmo. Nesse caso não é permitido a digitação de letras ou caracteres especiais.



***Activity* ARE *Mobile* 2 - Tela *Login* Aluno**

Após validação do aluno, será exibida a *Activity* ARE *Mobile* 3.

***Activity* ARE *Mobile* 3 - Tela Principal Aluno**

Na *Activity* ARE *Mobile* 3 - Tela Principal Aluno são apresentada outras quatro opções de telas:

- Tela Deveres de Casa é representada pela *Activity* ARE *Mobile* 4. Nela serão listados todos os deveres de casa do aluno.

***Activity* ARE *Mobile* 4 - Tela Deveres de Casa Aluno**

- Tela de Notas é representada pela *Activity* ARE *Mobile* 5. Nela é possível obter uma lista contendo todas as notas de avaliações realizadas pelo aluno.

***Activity* ARE *Mobile* 5 - Tela de Notas Aluno**

- Tela de Horários é representada pela *Activity* ARE *Mobile* 6. Nela é possível listar todos os horários de aula do aluno.

***Activity* ARE *Mobile* 6 - Tela Horários Aluno**

- Tela de Faltas é representada pela *Activity* ARE *Mobile* 7. Nela o aluno consulta todas as faltas por matéria e bimestre.

***Activity* ARE *Mobile* 7 - Tela Faltas Aluno**

3.4.7 Implementação

Para implementação do ARE *Mobile* foi utilizada a linguagem de programação *Java* na IDE Eclipse juntamente com o *plug-in* ADT.

A Figura 17 exibe a estrutura do projeto no que se refere às classes implementadas para uso das *Activities.*

As classes são representadas pelos arquivos com extensão .java e, para cada classe existe uma *activity* representada pelos arquivos de extensão .xml, conforme apresentado na Figura 18.

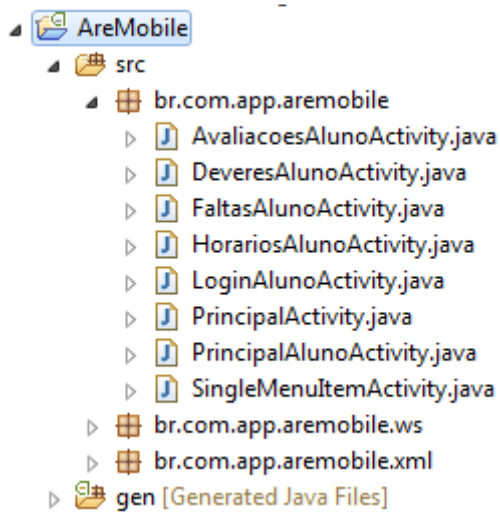


Figura 17 - Classes ARE *Mobile*

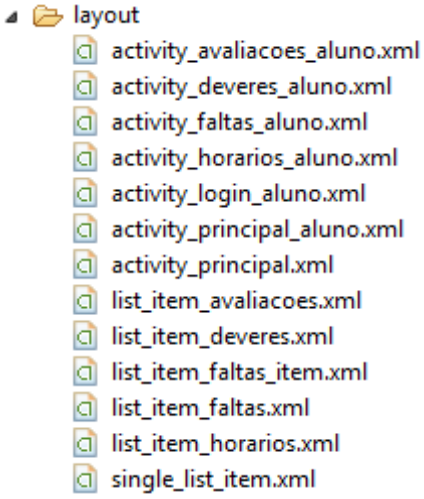


Figura 18 - XML *Activity* ARE *Mobile*

Ainda existem outras duas classes que são extremamente necessárias para funcionamento do *app*, apresentadas na Figura 19 e Figura 20.

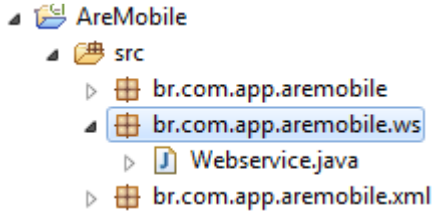


Figura 19 - Classe *WebService* ARE *Mobile*

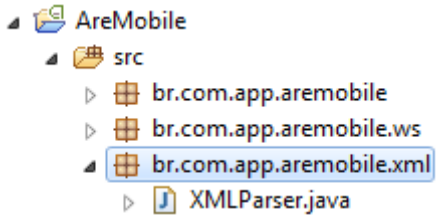


Figura 20 - Classe *XMLParser* ARE *Mobile*

A classe *Webservice.java* apresentada na Figura 19 tem como finalidade conectar-se ao *web services*, acessa-lo e realizar a consulta solicitada pelo *Android*.

Para que o acesso ocorra sem erros, é necessário usar a biblioteca *ksoap.* O *ksoap* permite transformar o XML em uma cadeia de caracteres literal para a comunicação *SOAP.* A biblioteca *ksoap* foi concebida para permitir a comunicação entre aplicações que rodam na plataforma *Android* com *web services* que utilizam o protocolo SOAP [23].

A Figura 21 mostra uma fração do código da classe *Webservice.java* .

Webservice.java

```
package br.com.app.aremobile.ws;

import java.util.List;

import org.ksoap2.HeaderProperty;
import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapPrimitive;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;

public class Webservice {

    public String NAMESPACE = "http://are.mobile.com.br";
    public String url = "http://189.11.37.163:8080/WSAreMobile/services/InformacoesAluno?wsdl";
    public int timeOut = 60000;

    public void setTimeOut(int seconds) {
        this.timeOut = seconds * 1000;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    /*
     * Informações do Aluno - Webservice
     *
     */
     public String consultarInformacoesAluno(String codigoAluno, List<HeaderProperty> headers) {
         SoapSerializationEnvelope soapEnvelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
         soapEnvelope.implicitTypes = true;
         soapEnvelope.dotNet = true;
         SoapObject soapReq = new SoapObject("http://are.mobile.com.br","consultarInformacoesAluno");
         soapReq.addProperty("codigoAluno", codigoAluno);
         soapEnvelope.setOutputSoapObject(soapReq);
         HttpTransportSE httpTransport = new HttpTransportSE(url, timeOut);
         try {

             if (headers != null) {
                 httpTransport.call("http://are.mobile.com.br/consultarInformacoesAluno",soapEnvelope, headers);
             } else {
                 httpTransport.call("http://are.mobile.com.br/consultarInformacoesAluno",soapEnvelope);
             }
             SoapObject result = (SoapObject) soapEnvelope.bodyIn;
             if (result.hasProperty("consultarInformacoesAlunoReturn")) {
                 Object obj = result.getProperty("consultarInformacoesAlunoReturn");
                 if (obj.getClass().equals(SoapPrimitive.class)) {
                     SoapPrimitive j4 = (SoapPrimitive) result.getProperty("consultarInformacoesAlunoReturn");
                     String resultVariable = j4.toString();
                     return resultVariable;
                 }
             }
         } catch (Exception e) {
             e.printStackTrace();
         }
         return null;
     }
```

Figura 21 - Classe *Webservice.java*

Já a classe *XMLParser.java* conforme apresentada na Figura 20 tem como função obter o conteúdo XML retornado pelo *web service,* analisar esse conteúdo XML e receber o elemento DOM do XML (criado pelo W3C, o DOM é uma multi-plataforma que representa como as marcações em HTML, XHTML e XML são organizadas e lidas pelo navegador). Uma vez indexadas, estas marcações se transformam em elementos de uma árvore que pode ser manipulada via API [24]. Feito isso, é possível obter cada valor do elemento XML passando o nome do nó do elemento.

Nas Figura 22 e Figura 23 é apresentado o código *Java*.

XMLParser.java

```
package br.com.app.aremobile.xml;

import java.io.IOException;

public class XMLParser {


    public XMLParser() {

    }

    /**
     * Getting XML from URL making HTTP request
     * @param url string
     * */
    public String getXmlFromUrl(String url) {
        String xml = null;

        try {
            // defaultHttpClient
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new HttpPost(url);

            HttpResponse httpResponse = httpClient.execute(httpPost);
            HttpEntity httpEntity = httpResponse.getEntity();
            xml = EntityUtils.toString(httpEntity);

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        // return XML
        return xml;
    }

    /**
     * Getting XML DOM element
     * @param XML string
     * */
    public Document getDomElement(String xml){
        Document doc = null;
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        try {

            DocumentBuilder db = dbf.newDocumentBuilder();

            InputSource is = new InputSource();
                is.setCharacterStream(new StringReader(xml));
                doc = db.parse(is);

            } catch (ParserConfigurationException e) {
                Log.e("Error: ", e.getMessage());
                return null;
            } catch (SAXException e) {
                Log.e("Error: ", e.getMessage());
                return null;
            } catch (IOException e) {
                Log.e("Error: ", e.getMessage());
                return null;
            }

            return doc;
    }
```

Figura 22 - *XMLParser.java* (Parte 1)

```
    /** Getting node value
      * @param elem element
      */
     */
    public final String getElementValue( Node elem ) {
        Node child;
        if( elem != null){
            if (elem.hasChildNodes()){
                for( child = elem.getFirstChild(); child != null; child = child.getNextSibling() ){
                    if( child.getNodeType() == Node.TEXT_NODE  ){
                        return child.getNodeValue();
                    }
                }
            }
        }
        return "";
    }

    /**
     * Getting node value
     * @param Element node
     * @param key string
     * */
    public String getValue(Element item, String str) {
           NodeList n = item.getElementsByTagName(str);
           return this.getElementValue(n.item(0));
       }
}
```

Figura 23 - *XMLParser.java* (Parte 2)

Para funcionamento da *app*, no arquivo *AndroidManifest.xml* foram declaradas as permissões necessários para acesso a internet, conforme apresentada na Figura 24.

AreMobile Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.app.aremobile"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="10" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Figura 24 - Permissões *AndroidManifest.xml*

Nesse mesmo arquivo, foram declaradas as *Activities* desenvolvidas para funcionamento do *app* ARE *Mobile*, apresentados na Figura 25.

AreMobile Manifest

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/HomeText" >
    <activity
        android:name="br.com.app.aremobile.PrincipalActivity"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.NoTitleBar" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name="br.com.app.aremobile.LoginAlunoActivity"
        android:theme="@android:style/Theme.NoTitleBar" />
    <activity
        android:name="br.com.app.aremobile.NotasActivity"
        android:theme="@android:style/Theme.NoTitleBar" />
    <activity
        android:name="br.com.app.aremobile.PrincipalAlunoActivity"
        android:theme="@android:style/Theme.NoTitleBar" />
    <activity
        android:name="br.com.app.aremobile.DeveresAlunoActivity"
        android:theme="@android:style/Theme.NoTitleBar" />
    <activity
        android:name="br.com.app.aremobile.AvaliacoesAlunoActivity"
        android:theme="@android:style/Theme.NoTitleBar" />
    <activity
        android:name="br.com.app.aremobile.HorariosAlunoActivity"
        android:theme="@android:style/Theme.NoTitleBar" />
    <activity
        android:name="br.com.app.aremobile.FaltasAlunoActivity"
        android:theme="@android:style/Theme.NoTitleBar" />
    <activity android:name="br.com.app.aremobile.SingleMenuItemActivity" >
    </activity>
</application>
```

Figura 25 - *AndroidManifest.xml*

Para garantir o funcionamento do *app* é necessário assegurar que o *smartphone tem acesso à internet.* O trecho de código *Java* apresentado na Figura 26 é responsável por verificar se existe alguma conexão com a internet, seja ela 3G ou Wi-Fi.

```
/*
 * Are Mobile
 * Autor: Nilson André
 * Metodo: Verificar se smartphone tem acesso a internet
 *
*/

public static boolean VerificaConexao(Context contexto) {
    ConnectivityManager cm = (ConnectivityManager) contexto
            .getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo netInfo = cm.getActiveNetworkInfo();
    if ((netInfo != null) && (netInfo.isConnectedOrConnecting())
            && (netInfo.isAvailable()))
        return true;
    else
        return false;

}
```

Figura 26 - Verifica conexão com *internet*

O método que verifica a conexão com internet, apresentado na Figura 26 é chamado / invocado dentro do método *onCreate* da *PrincipalActivity.java* e apresentado na Figura 27.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_principal);
    txtInternet = (TextView) findViewById(R.id.tvRodape);
    if (VerificaConexao(this)){
        msgInternet = "Conexão com a internet está disponível.";
    }else{
        msgInternet= "Internet indisponível no momento. Tente mais tarde.";
    }
    txtInternet.setText(msgInternet);

}
```

Figura 27 - Chamada do método que verificar conexão com *internet*

Seguindo os padrões de desenvolvimento *Android*, todos os *layouts* das *activities* foram implementados utilizando-se XML. Uma fração do código XML responsável pelo *layout* da activity_deveres_aluno.xml é apresentado na Figura 28.

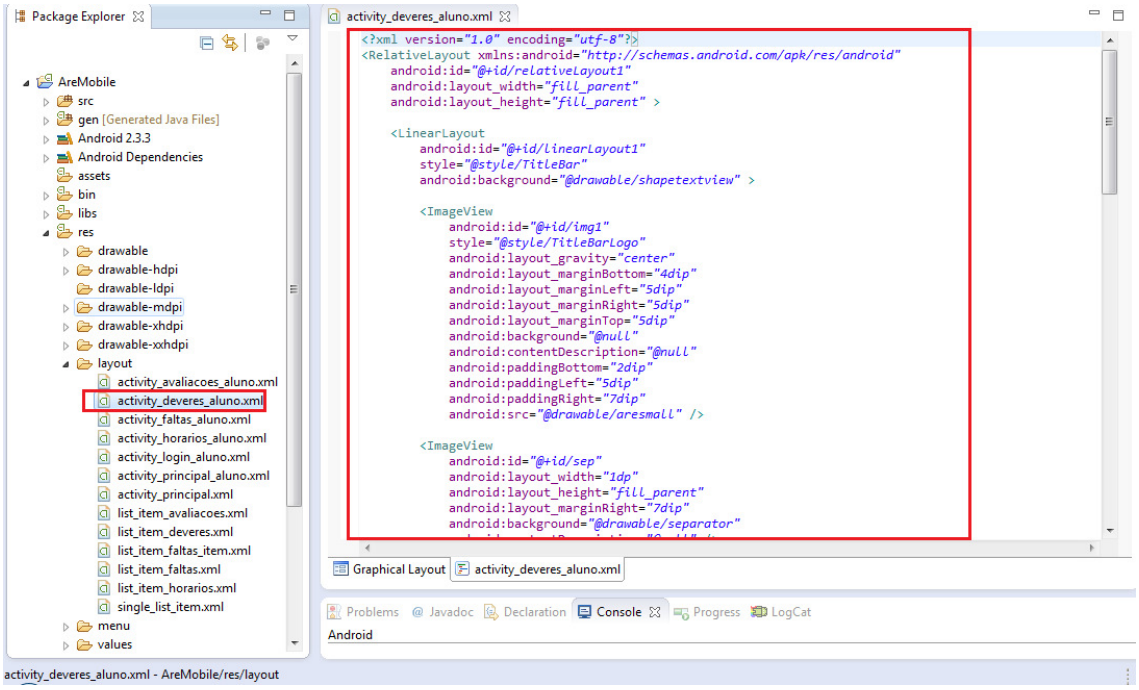


Figura 28 - Código XML *Activity* Deveres Aluno

No Apêndice G são apresentados mais detalhes de implementação *Java*.

4 VALIDAÇÃO DO PROTÓTIPO

Para validação do protótipo foi necessário a implementação do *web service* e preenchimento dos dados no banco de dados. Tendo uma massa de dados, foi possível iniciar a validação do protótipo para apresentar nessa seção.

Durante a fase de desenvolvimento todos os testes foram realizados usando o emulador SDK do *Android*. Para uma melhor validação do protótipo, foi gerado uma APK do ARE *Mobile* e instalado em um aparelho *smartphone Samsung Galaxy Y Duos* conforme apresentado na Figura 29.

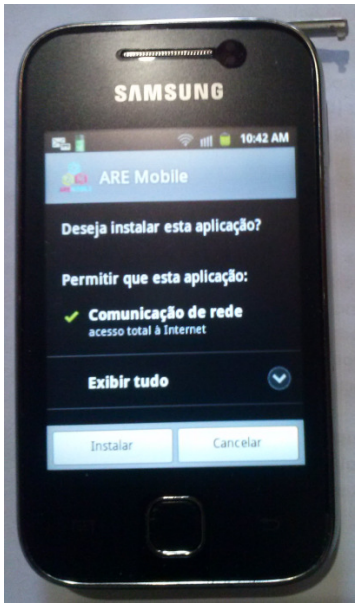


Figura 29 - Instalação APK ARE *Mobile*

Após a instalação do ARE *Mobile*, pode ser observada a tela principal do *app* em que é possível iniciar a validação propriamente dita e já identificar algumas características do aplicativo.

A Figura 30 apresenta a tela principal. Nela foram destacados dois itens:

- Item 01: Nota-se que o aparelho está conectado em uma rede sem fio;
- Item 02: O *app* identificou que existe uma conexão de internet disponível.

Na Figura 31 é destacado o item 01, mostrando a mensagem exibida pela *activity* quando não existe uma conexão com *internet* disponível.

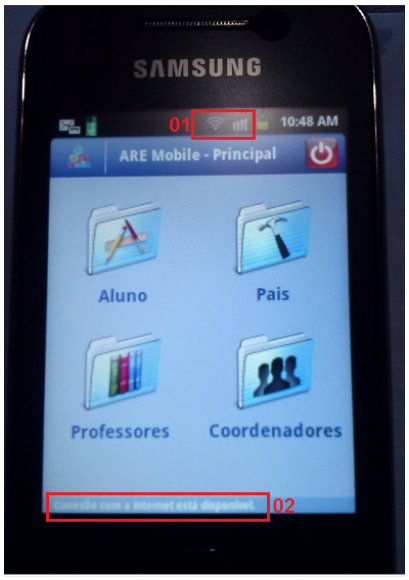


Figura 30 - Tela Principal ARE *Mobile*

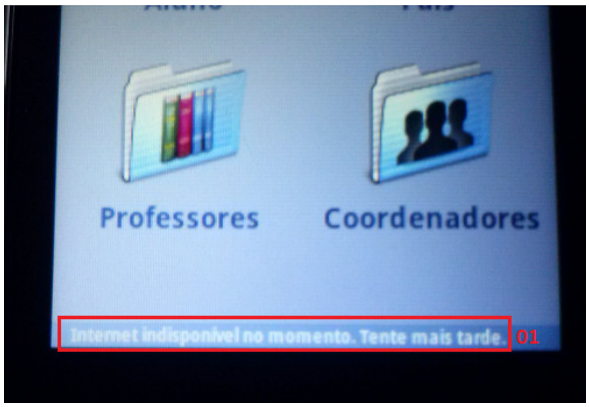


Figura 31 - Tela Principal ARE *Mobile* Sem *Internet*

Ao tocar no perfil Aluno, é solicita validação de usuário e senha. Essa tela de requisição é apresentada na Figura 32. Ao informar um usuário e senha correto é apresentada a Figura 33.

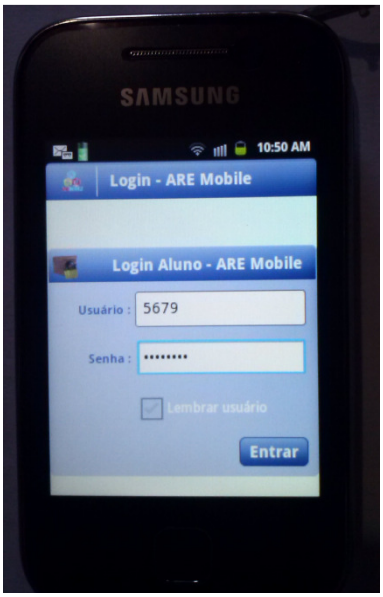


Figura 32 - Autenticação ARE *Mobile*

Figura 33 - Usuário e Senha Validados com sucesso

Com o *web service* publicado na *internet* e acessível por meio do *link* http://189.11.37.163/WSAreMobile é possível passar os parâmetros para consulta do XML que o *smartphone* receberá. Dessa forma é possível comparar o XML obtido por meio do *web service* com o conteúdo apresentado no aparelho *smartphone*.

Na Figura 34 é apresentado o XML de retorno do *web service* ao passar como parâmetro o aluno cujo código de matrícula é 5679.

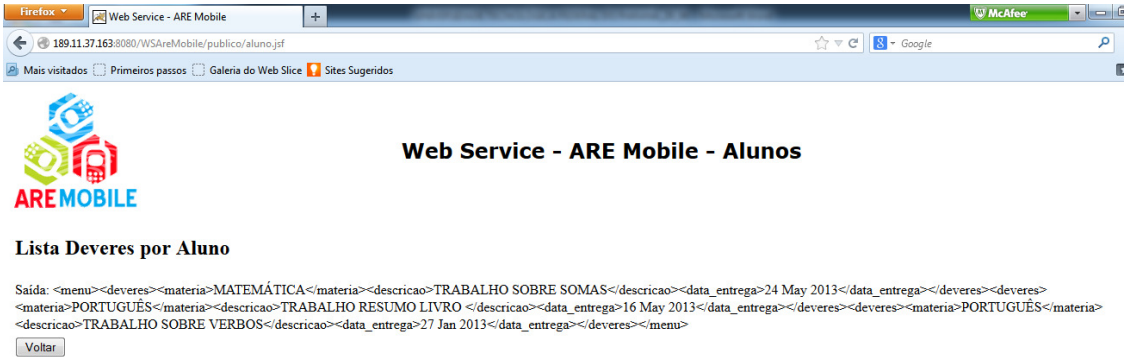


Figura 34 - *Response* Deveres Aluno

Para comparação, é apresentada a Figura 35. Nesta figura é possível verificar que o ARE *Mobile* recebeu o mesmo XML, tratou corretamente e apresentou na *activity* Deveres de Casa.

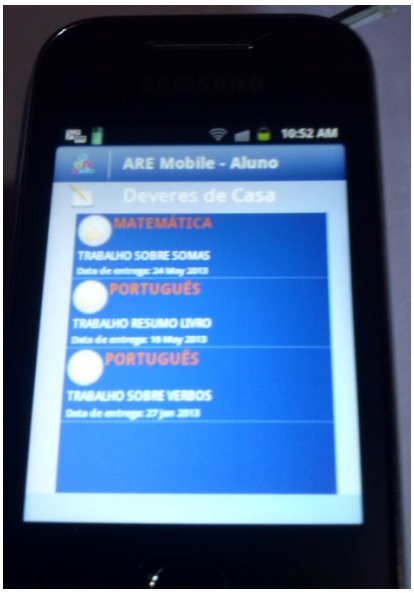


Figura 35 - Tela Deveres de Casa

O mesmo acontece com as demais *activities*. Na sequência será apresentada essa mesma comparação para as *activities* do perfil Aluno.

A Figura 36 mostra o XML referente aos horários do aluno.

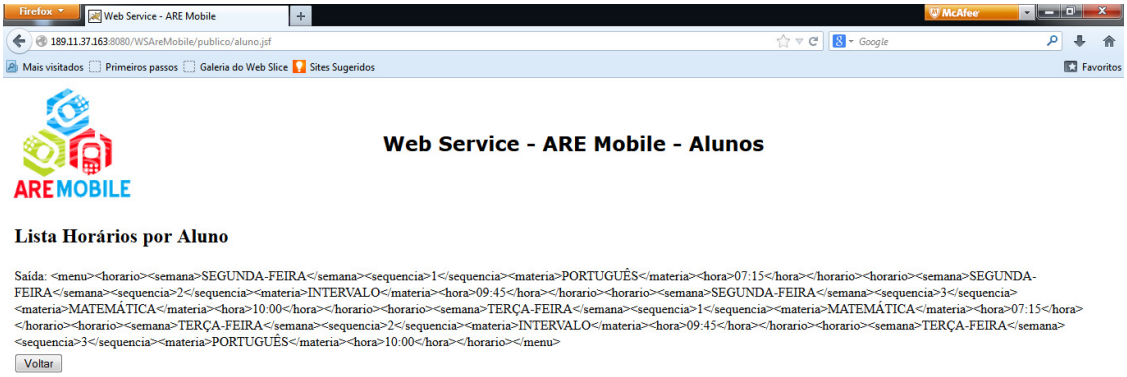


Figura 36 - *Response* Horários Aluno

Na Figura 37 é possível verificar como as informações contidas no XML são apresentadas na *activity*.

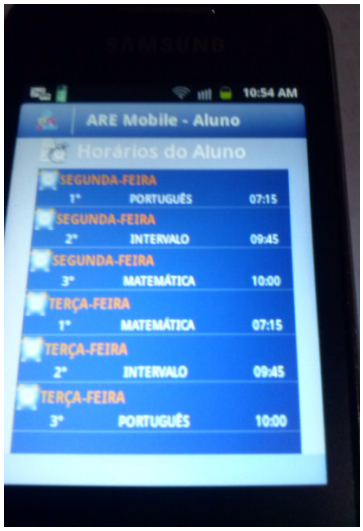


Figura 37 - Tela Horários Aluno

A Figura 38 apresenta o XML referente à consulta de avaliações do aluno.

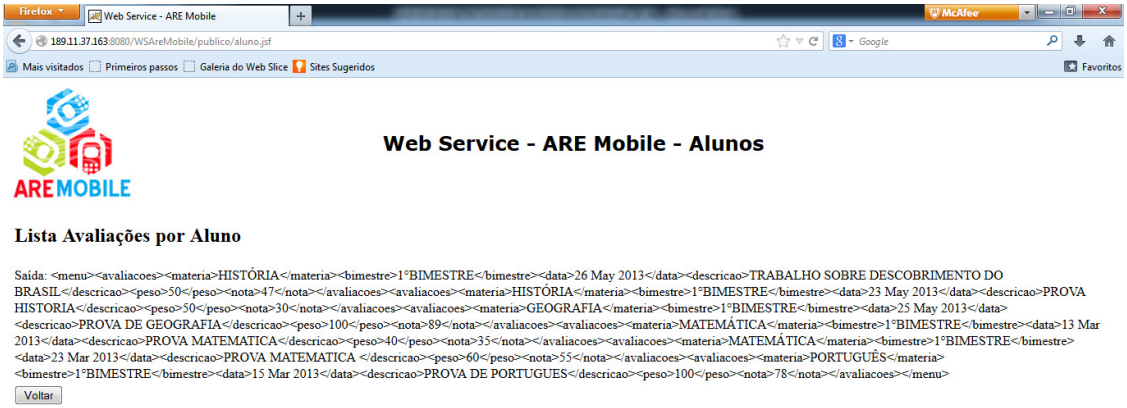


Figura 38 - *Response* Avaliações Aluno

Na Figura 39 é apresentado o conteúdo do XML já tratado pela *activity*.

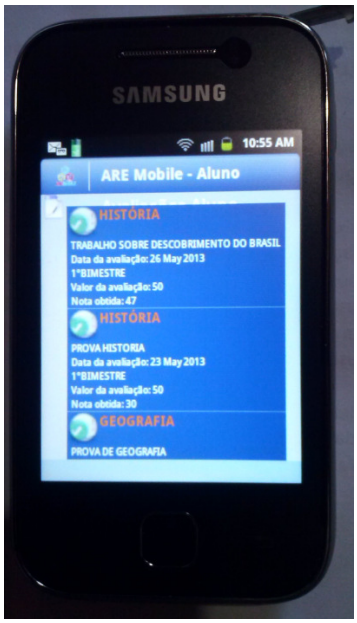


Figura 39 - Tela Avaliações Aluno

A Figura 40 mostra o XML referente à consulta faltas do aluno.

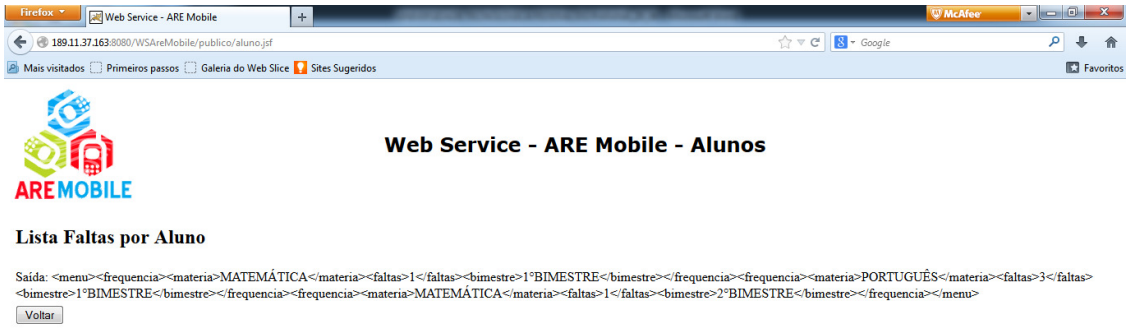


Figura 40 - *Response* Faltas Aluno

Na Figura 41 é apresentado o conteúdo do XML já tratado pela *activity*.

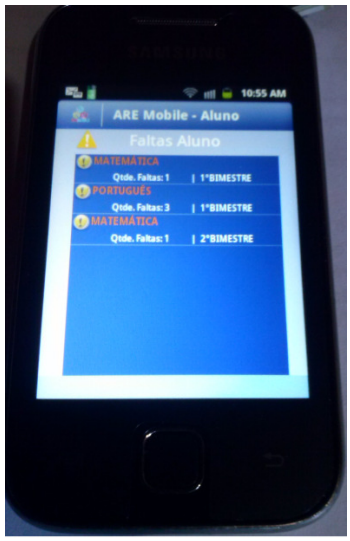


Figura 41 - Tela Faltas Aluno

Para finalizar a validação do protótipo, na Figura 42 é apresentado mensagem de confirmação de saída do ARE *Mobile*.

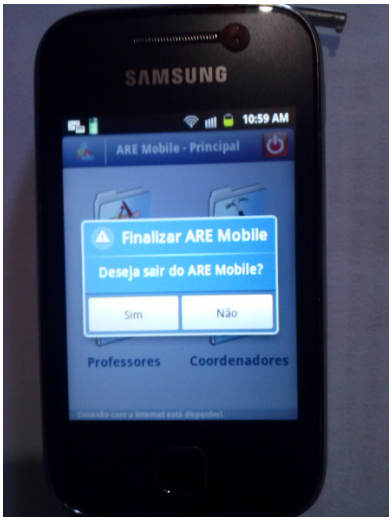


Figura 42 - Sair ARE *Mobile*

As funcionalidades do ARE *Mobile*, referentes ao perfil do aluno, desenvolvidas no protótipo, foram validadas diversas vezes, sendo no emulador SDK do *Android* como também em alguns modelos de aparelhos. Em todos os testes realizados, foram retornados os comportamentos esperados.

Como o *web service* está publicado em um *link* dedicado onde existem outras aplicações sendo requisitadas em paralelo, em alguns momentos é possível observar uma certa lentidão ao realizar as consultas. Uma solução para essa situação é utilizar um *link* exclusivo para esse serviço.

Mesmo com essa ressalva, é possível assegurar que o ARE *Mobile* atendeu as expectativas esperadas.

5 CONCLUSÃO

Muitas instituições de ensino oferecem uma ferramenta para consulta e acompanhamento escolar. Geralmente é uma ferramenta web, disponível na internet e, em sua grande maioria acessível por meio do site da instituição de ensino, que permite aos pais e/ou responsáveis o acompanhamento do rendimento escolar do aluno, ao aluno o controle de sua vida escolar, aos professores que possam interagir com a secretaria por meio do sistema e aos coordenadores que consigam extrair informações referentes ao aluno ou professor, enfim, tendo um equipamento com acesso a internet é possível obter informações de acompanhamento escolar de uma maneira fácil e simples.

Pensando em uma possível melhoria e buscando ampliar a quantidade de usuários do sistema, uma ferramenta que possibilita consultar as mesmas informações disponíveis na *web* por meio de um aplicativo para *smartphone Android* foi desenvolvida.

Para o desenvolvimento deste trabalho foram utilizadas diversas características da plataforma *Android*, procurando contribuir com um conteúdo de leitura aos interessados em explorar o desenvolvimento nesta plataforma, bem como aos que se interessam pelo desenvolvimento para dispositivos móveis em geral. Também foram abordadas características de desenvolvimento de *web service* e também avaliação de usabilidade.

Com o desenvolvimento do protótipo, espera-se contribuir para que instituições de ensino possam oferecer uma ferramenta *mobile* para consulta e acompanhamento de rendimento escolar.

A escolha da plataforma *Android* foi motivada principalmente por ser um projeto de código aberto com disponibilidade de vasta documentação e códigos de exemplo, bem como de ferramentas gratuitas que auxiliam o desenvolvimento, aspectos que seguramente facilitam e aceleram o processo de criação de aplicações para dispositivos móveis.

Com base na atual interface do sistema da instituição de ensino, foi realizada uma análise de usabilidade aplicando os métodos de avaliação heurística e *checklist*. Essa avaliação apontou como principal falha à falta de retorno de informações (falta de *feedback*) no sistema atual, falhas essas que foram corrigidas e aplicadas no protótipo *mobile* apresentado nesse trabalho.

O protótipo ARE *Mobile* foi desenvolvido contemplado o perfil Aluno e oferecendo as mesmas funcionalidades do sistema web em um *smartphone* com sistema operacional *Android*.

O protótipo desenvolvido nesse trabalho comprovou cumprir seus objetivos, conforme as validações realizadas, dentre os quais permitir a consulta de informações disponibilizadas a partir de um *web service* e apresentar em um *smartphone.* Dessa forma a instituição de ensino passa a oferecer uma nova ferramenta para consulta e acompanhamento de rendimento escolar seguindo a tendência de mobilidade.

5.1 CONTRIBUIÇÕES E TRABALHOS FUTUROS

O material apresentando nesse trabalho pode ser utilizando com fonte de pesquisa para outros trabalhos que contemplam sistema operacional *Android, web service* ou usabilidade. O código fonte do *web service* ou mesmo do aplicativo ARE *Mobile* disponível nesse trabalho, pode contribuir para estudos, ampliações ou melhorias.

Como a aplicação apresentada no trabalho é um protótipo, no qual foi desenvolvido o perfil dos alunos, as outras funcionalidades previstas no trabalho podem ser adicionadas em trabalhos futuros. Essas funcionalidades estão relacionadas ao:

- Perfil dos Pais/Responsável;
- Perfil do Professor;
- Perfil do Coordenador.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MANDEL, Theo. **Elements of user interface design**. New York: John Willey & Son, 1997

[2] FAEBER, N., HILZINGER, M. Sistemas Operacionais Móveis. **Linux Magazine**. São Paulo: LNMB, n.75, p. 32-37, fev. 2011.

[3] IDC: **Android- and iOS-Powered Smartphones Expand Their Share of the Market in the First Quarter, According to IDC**. Disponível em: http://www.idc.com/getdoc.jsp?containerId=prUS23503312 . Acesso em 03 mai. 2013.

[4] LECHETA, R. R. Google Android: **Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. Novatec, 2ª edição, 2010.

[5] ALLIANCE MEMBERS: **Open Handset Alliance**. Disponível em: http://www.openhandsetalliance.com/oha_members.html. Acesso em 03 de mai. 2013.

[6] ANDROID DEVELOPERS: **Android Developers.** Disponível em: http://developer.android.com. Acesso em 03 de mai. 2013.

[7] CONHECENDO O ANDROID: **Dicas Conhecendo Android.** Disponível em: http://agata.codigolivre.org.br/arquivo/conhecendo_o_android.php. Acesso em 03 de mai. 2013.

[8] CICLO DE VIDA ACTIVITY: **Entenda o ciclo de vida de uma Activity Android.** Disponível em: http://www.petherjose.com/2011/07/entenda-o-ciclo-de-vida-de-uma-activity-android/. Acesso em 05 de mai. 2013.

[9] PRESSMAN, Roger S. **Engenharia de software.** Rio de Janeiro: McGraw-Hill, 2002.

[10] ABNT – Associação Brasileira de Normas Técnicas. **Requisitos Ergonômicos para Trabalho de Escritórios com Computadores.** 2002. NBR 9241 Parte 11 – Orientações sobre Usabilidade. Disponível em: http://www.inf.ufsc.br/~cybis/pg2003/iso9241-11F2.pdf. Acesso em 30 de mai. 2013.

[11] USABILIDADE MOBILE: **Usabilidade Mobile – Uma visão de Jakob Nielsen.** Disponível em: http://www.dclick.com.br/2012/01/19/usabilidade-mobile-uma-visao-de-jakob-nielsen/. Acesso em 30 de mai. 2013.

[12] TABLELESS: **Usabilidade de interfaces para dispositivos movéis – parte 1.** Disponível em: http://tableless.com.br/usabilidade-de-interfaces-para-dispositivos-moveis-parte1. Acesso em 30 de mai. 2013.

[13] TABLELESS: **Usabilidade de interfaces para dispositivos movéis – parte 2.** Disponível em: http://tableless.com.br/usabilidade-interfaces-dispositivos-moveis-parte2. Acesso em 30 de mai. 2013.

[14] ROCHA, Heloisa Vieira da, **Design e avaliação de interfaces humano-computador.** Campinas: NIED/UNICAMP,2003.

[15] CYBIS, Walter de Abreu. 2003. **Engenharia de usabilidade:** uma abordagem ergonômica. Disponível em: http://www.labiutil.inf.ufsc.br/hiperdocumento/conteudo.html. Acesso em 31 de mai. 2013.

[16] LABIUTIL. 2011. **ErgoList.** Disponível em: http://www.labiutil.inf.ufsc.br/ergolist/ . Acesso em 31 de mai.2013.

[17] BALSAMIQ MOCKUPS: **Balsamiq Mockups Web Demo.** Disponível em: http://builds.balsamiq.com/b/mockups-web-demo/. Acesso em 03 de jun. 2013.

[18] WEB SERVICE – CONCEITO: **ITnerante Estudos de TI para Concursos Públicos.** Disponível em: http://www.itnerante.com.br/profiles/blogs/web-service-conceito. Acesso em 03 de jun. 2013.

[19] VISÃO GERAL SOBRE WEBSERVICES: **Visão geral sobre webservices – iMaster.** Disponível em: http://imasters.com.br/artigo/1680/web-services/visao-geral-sobre-webservices/. Acesso em 03 de jun. 2013.

[20] WEBSERVICE - AXIS: **Apache Axis.** Disponível em: http://axis.apache.org/axis/ . Acesso em 04 de jun. 2013.

[21] ANDROID DEVELOPERS: **System Requirements**. Disponível em: http://developer.android.com/sdk/requirements.html. Acesso em 04 jun. 2013

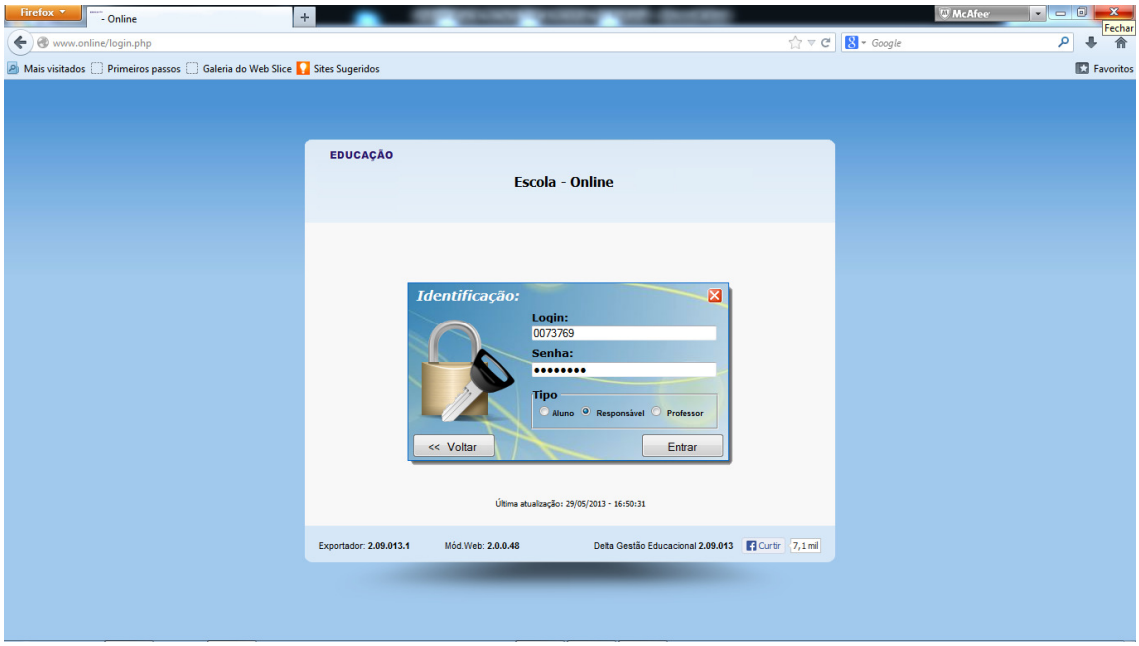
[22] ANDROID DEVELOPERS: **Dashboards and Platform Versions.** Disponível em: http://developer.android.com/about/dashboards/index.html. Acesso em 05 jun. 2013.

[23] TECNOLOGIAS, INFRAESTRUTURA E SOFTWARE: **Desenvolvimento de API Android para validador WSDL de WebServices SOAP em Java.** Disponível em: http://revistatis.dc.ufscar.br/index.php/revista/article/download/38/41 . Acesso em 10 jul. 2013.

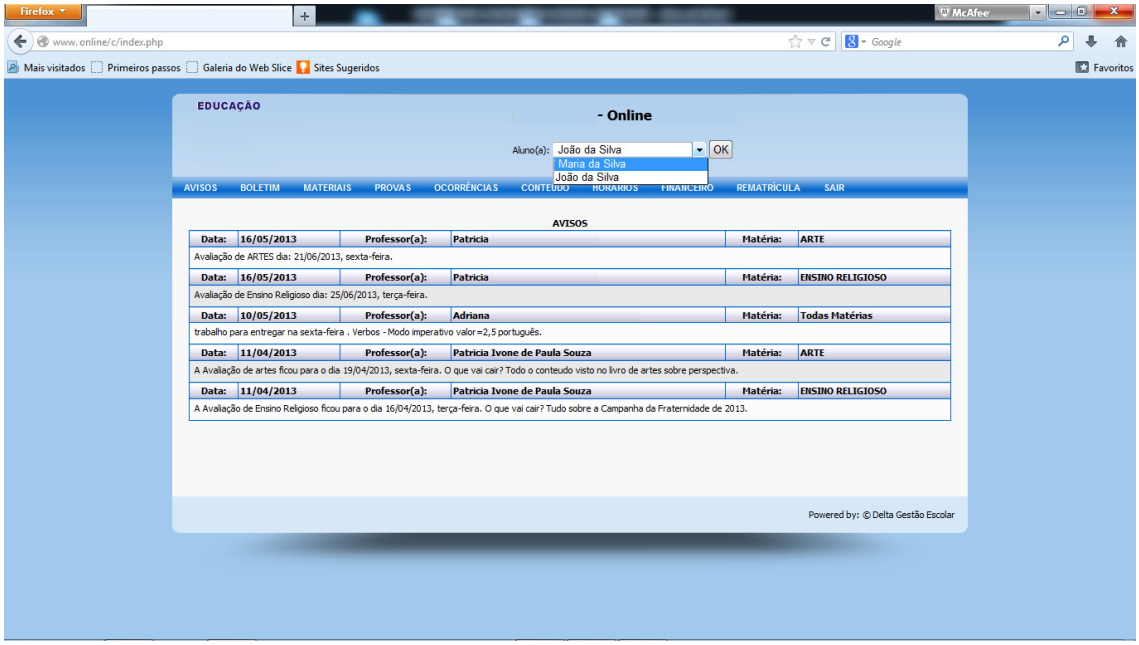
[24] TABLELESS: **Entenda o que é o Document Object Model e tenha o DOM.** Disponível em: http://tableless.com.br/tenha-o-dom/. Acesso em 12 de jul. 2013

[25] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do Usuário.** Rio de Janeiro: Campus, 2006.

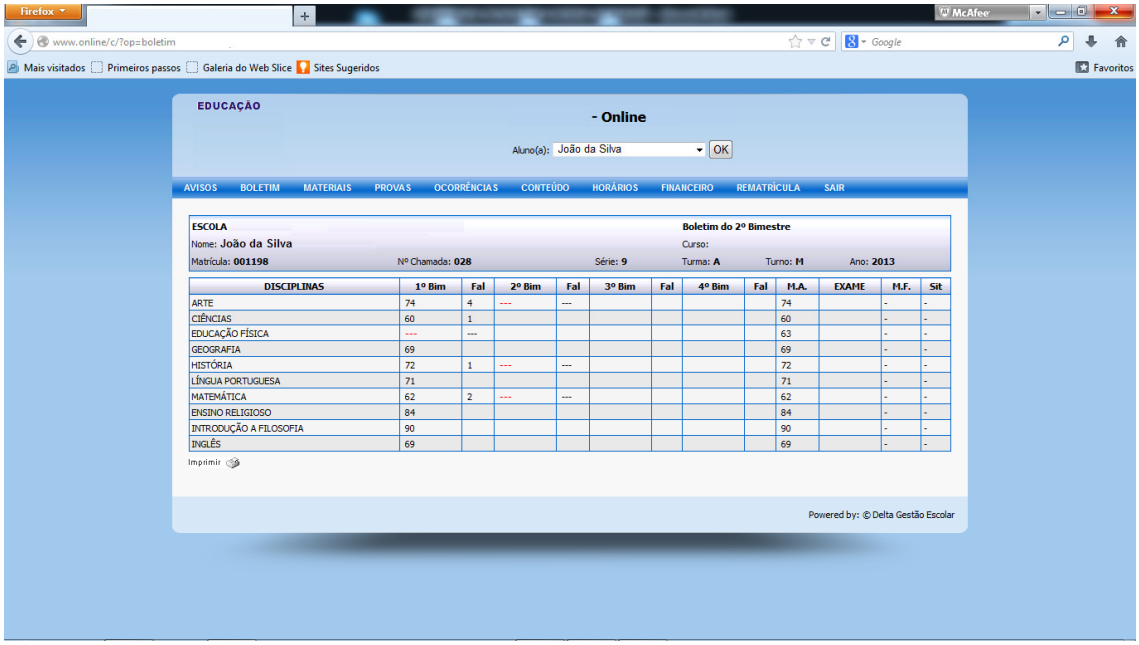
APÊNDICE A – Telas sistema Web atual – Perfil Responsável



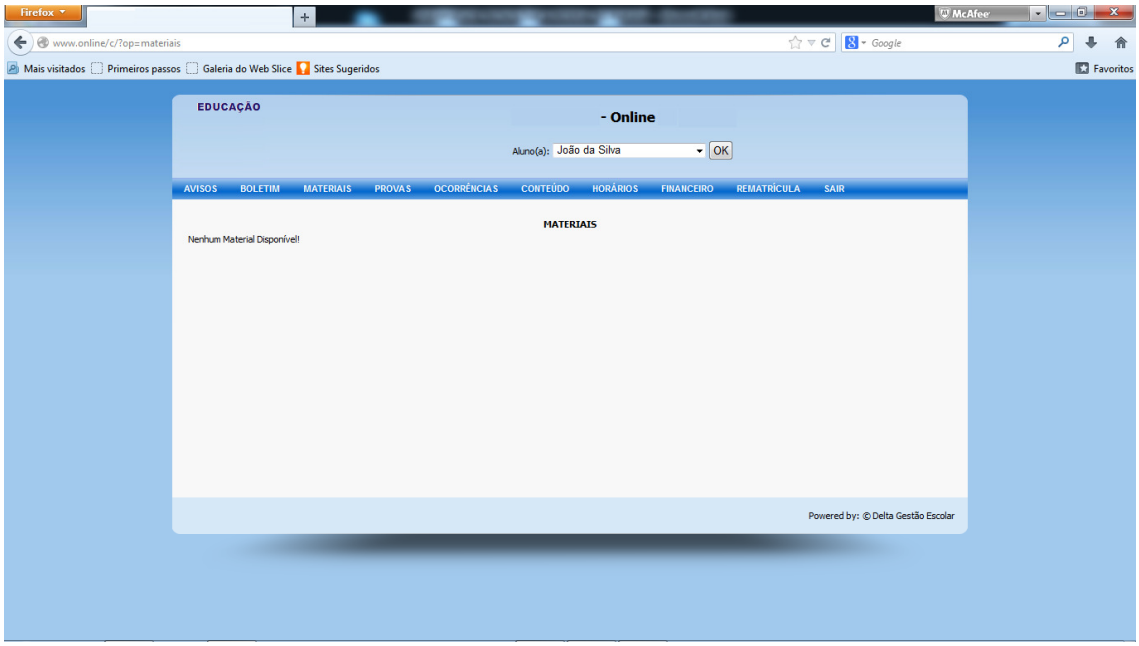
Interface Web 1 - *Login* Perfil Responsável



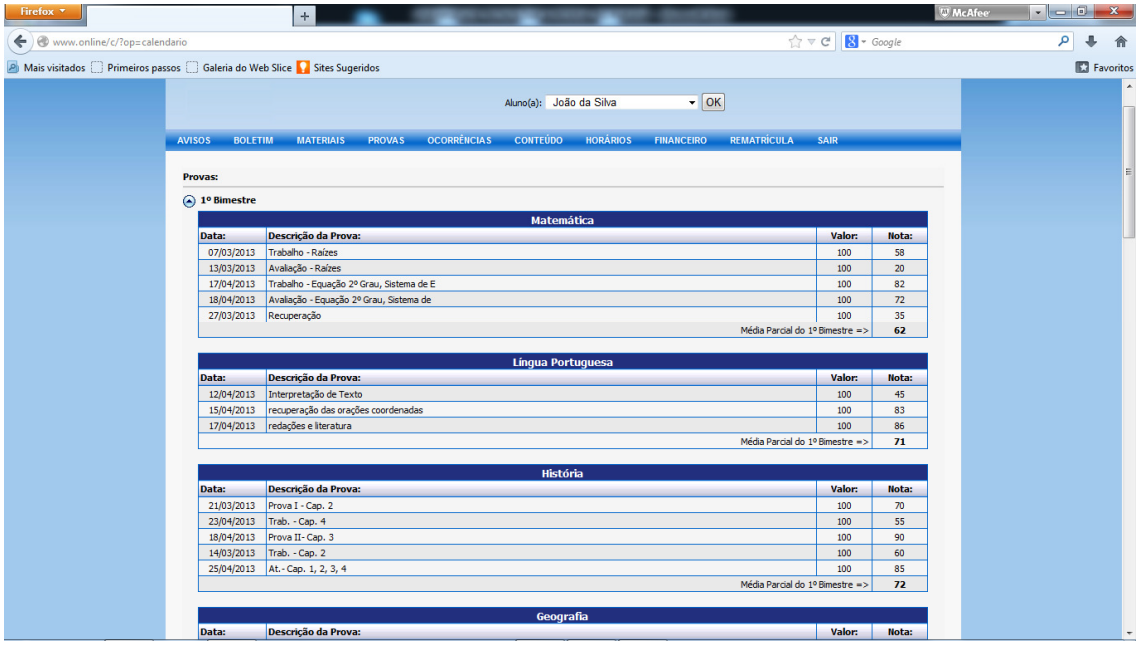
Interface Web 2 - Consulta Avisos Perfil Responsável



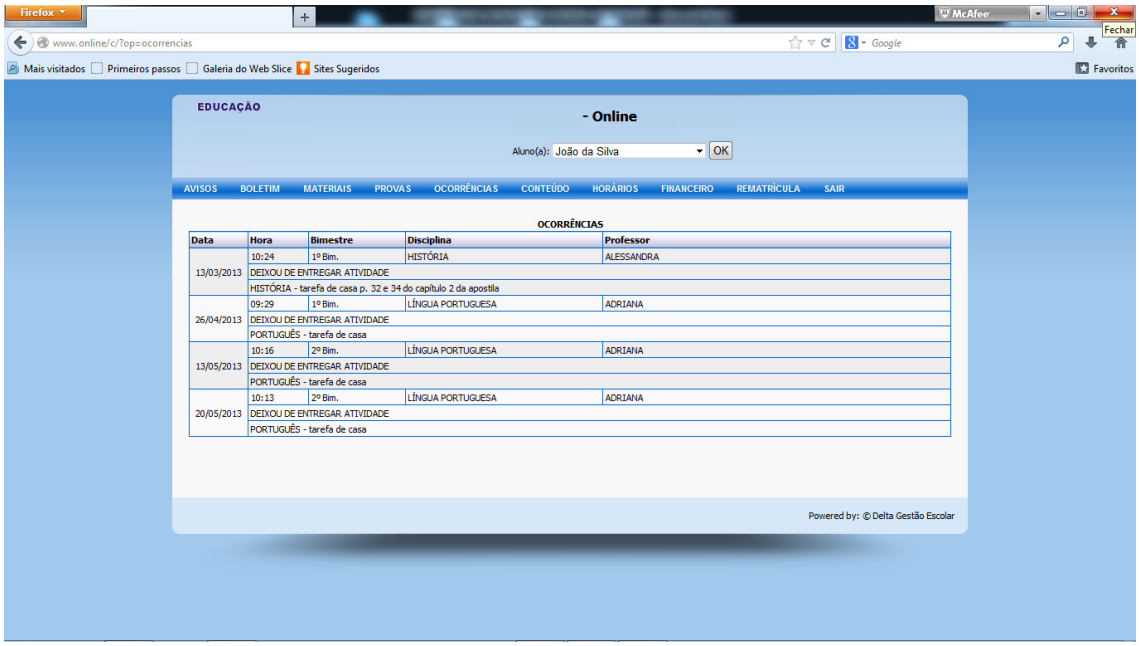
Interface Web 3 - Consulta Boletim Perfil Responsável



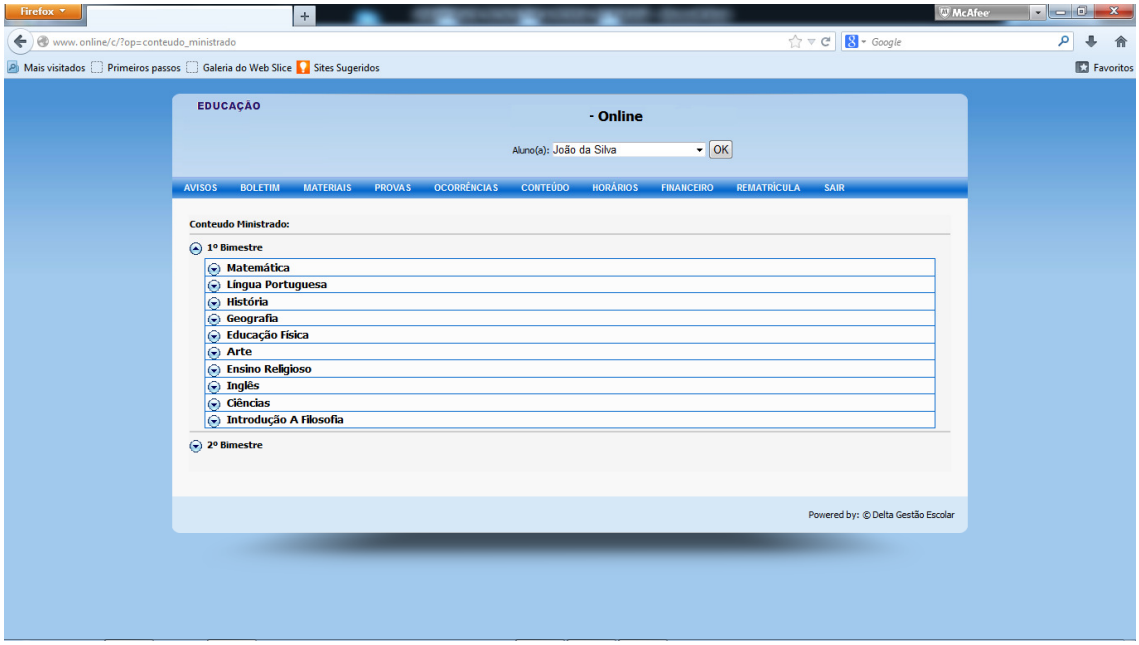
Interface Web 4 - Consulta Materiais Perfil Responsável



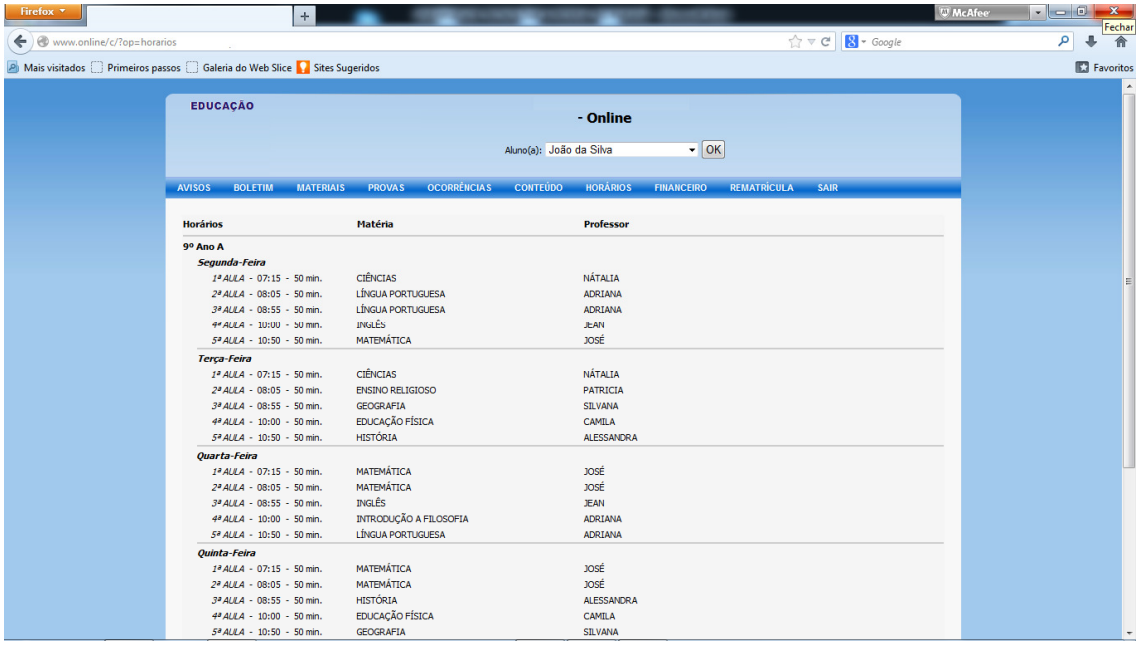
Interface Web 5 - Consulta Provas Perfil Responsável



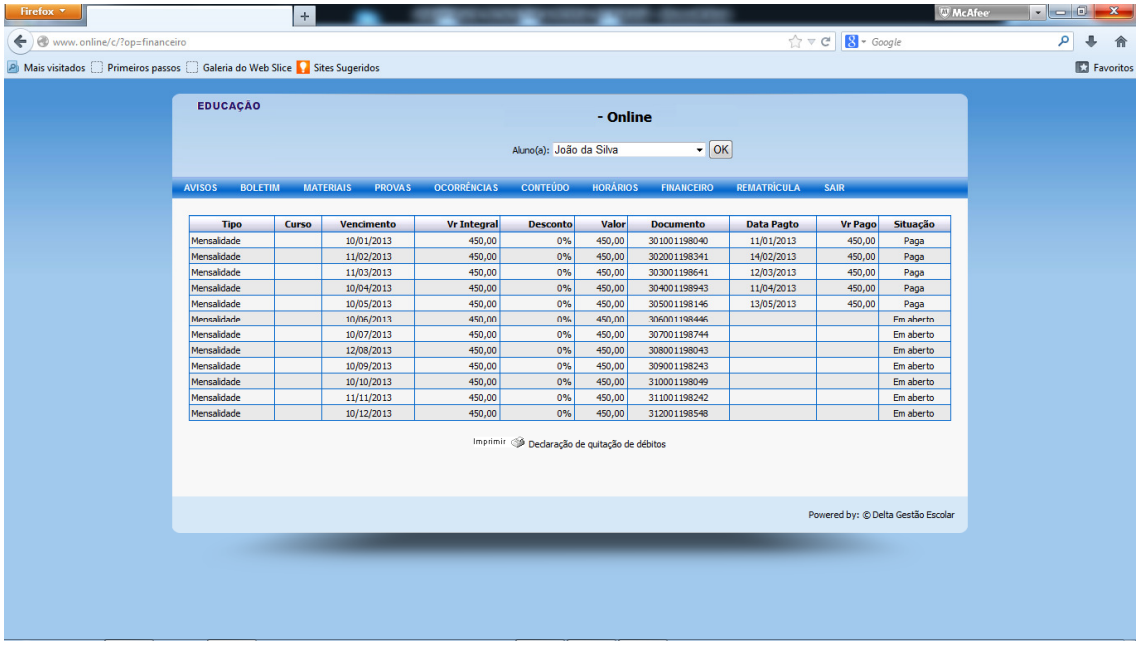
Interface Web 6 - Consulta Ocorrências Perfil Responsável



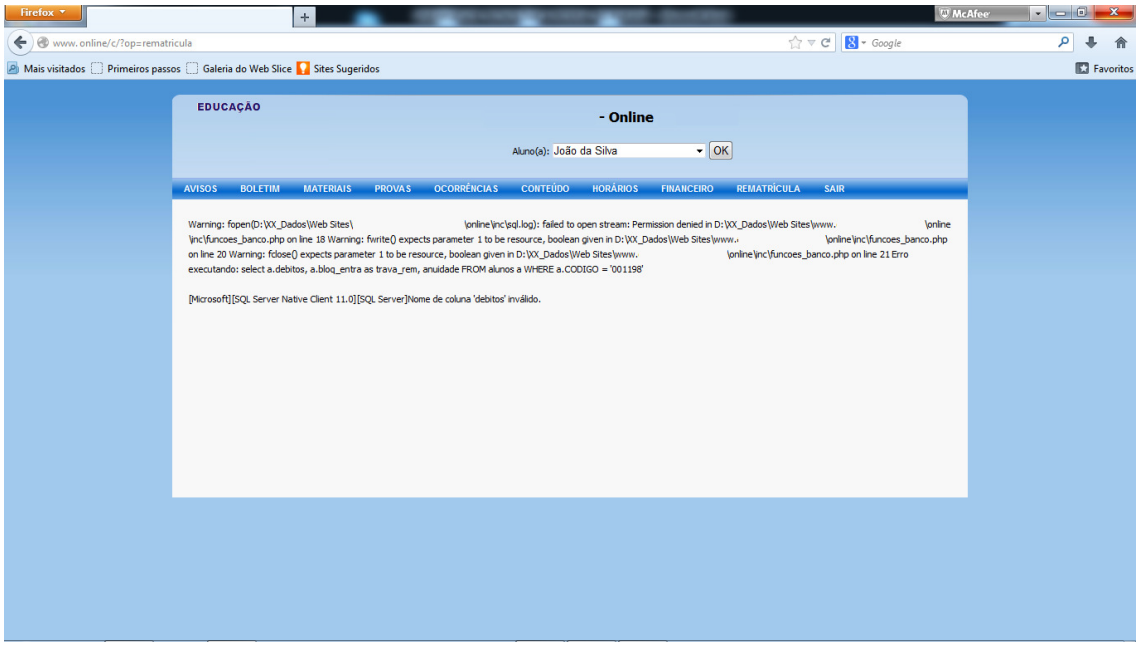
Interface Web 7 - Consulta Conteúdo Perfil Responsável



Interface Web 8 - Consulta Horários Perfil Responsável

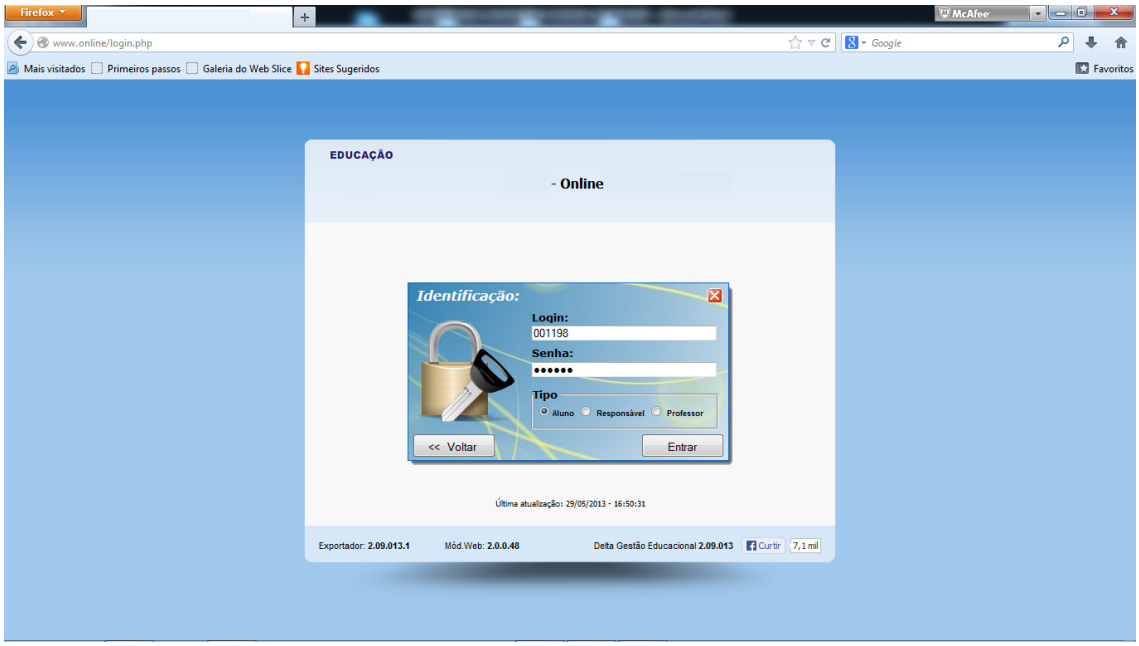


Interface Web 9 - Consulta Financeiro Perfil Responsável

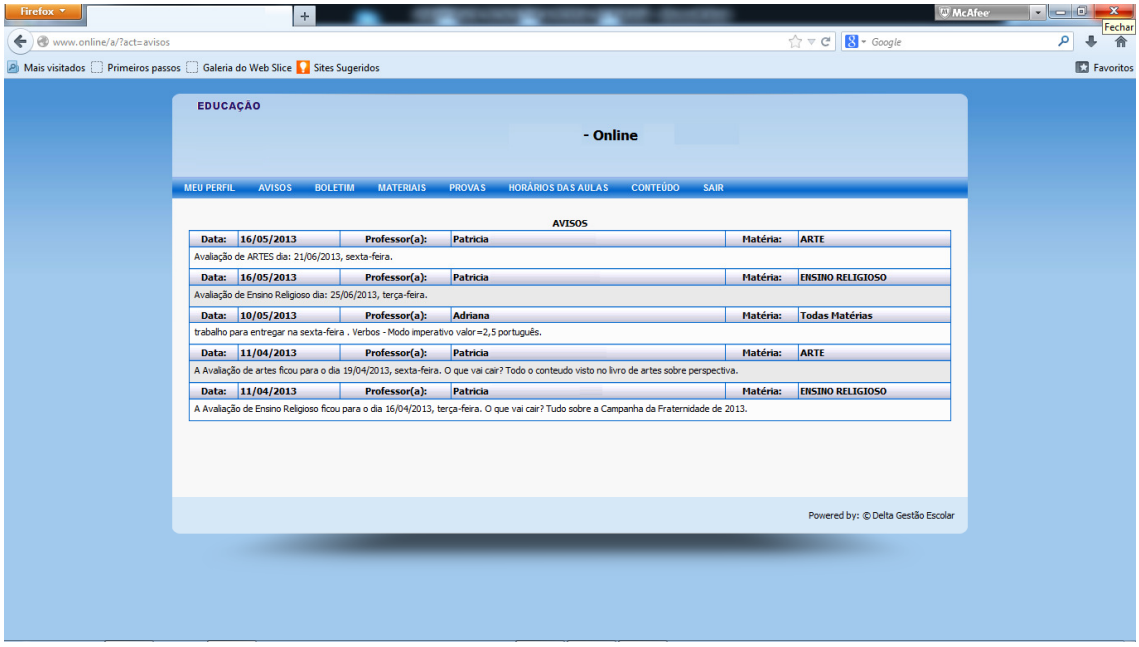


Interface Web 10 - Rematrícula Perfil Responsável

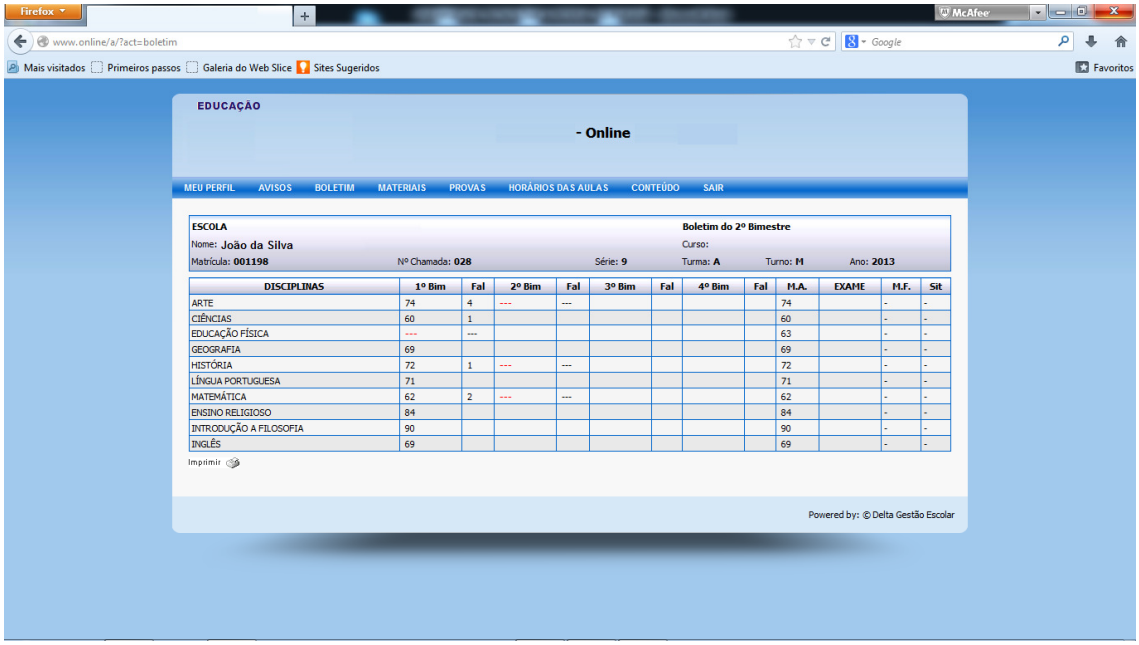
APÊNDICE B – Telas sistema Web atual – Perfil Aluno



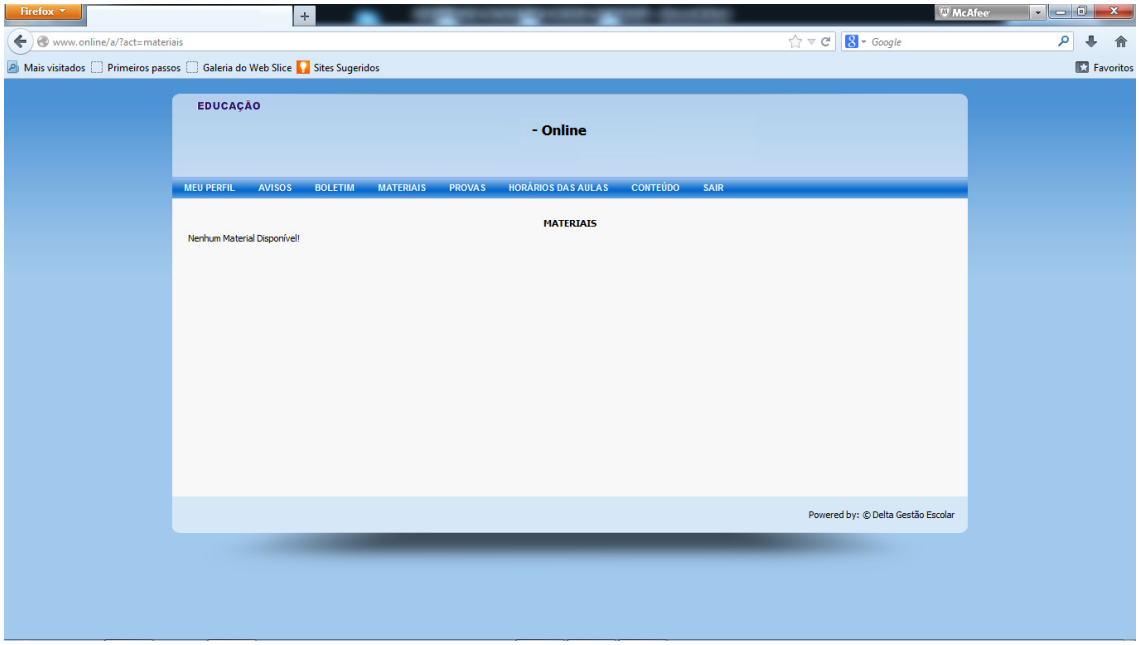
Interface Web 11 - *Login* Perfil Aluno



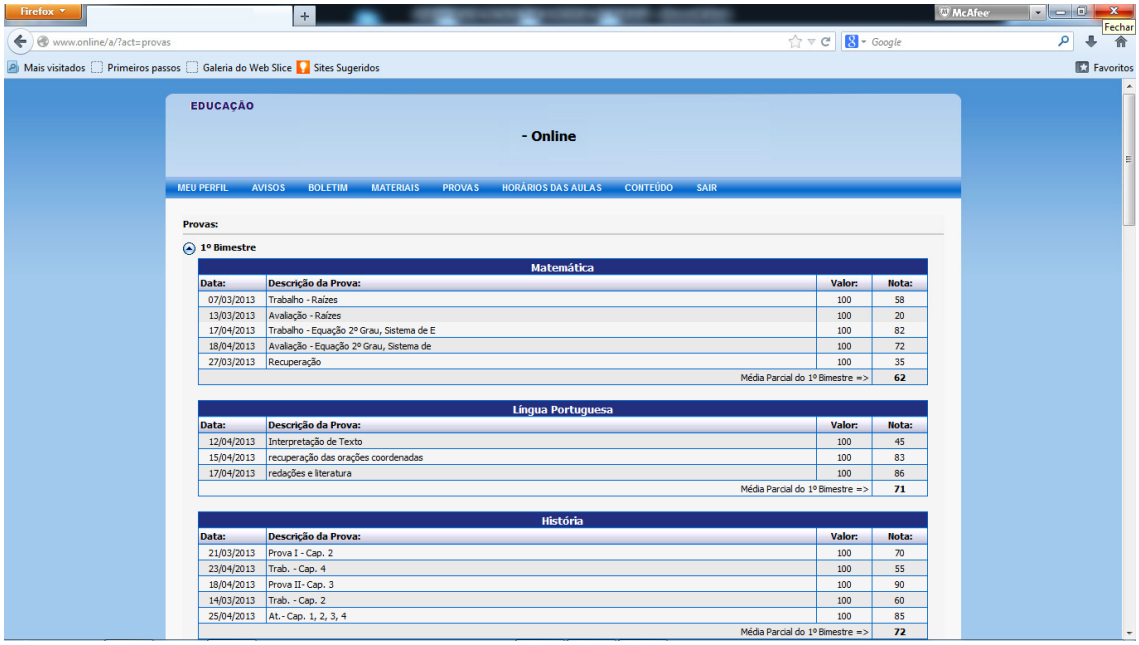
Interface Web 12 - Consulta Avisos Perfil Aluno



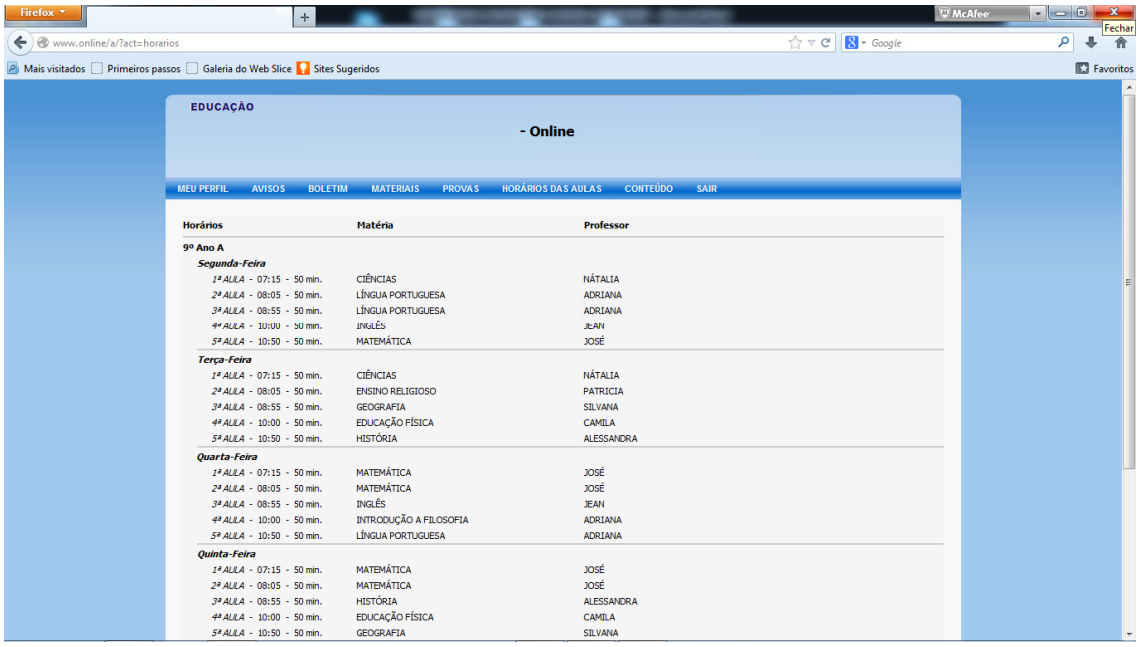
Interface Web 13 - Consulta Boletim Perfil Aluno



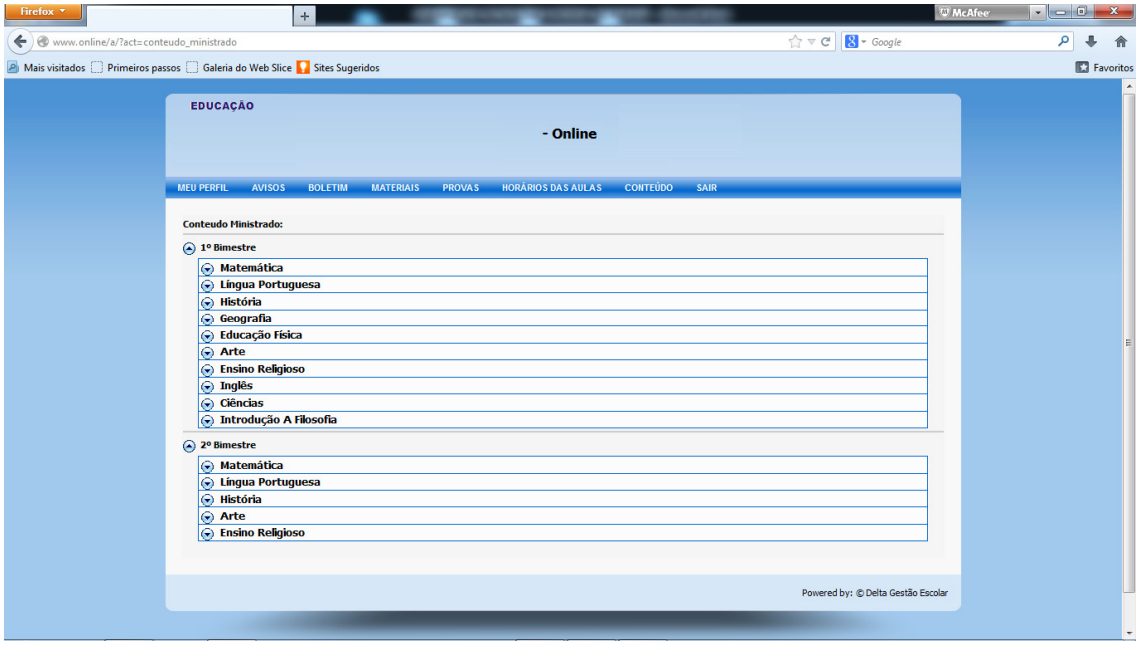
Interface Web 14 - Consulta Matérias Perfil Aluno



Interface Web 15 - Consulta Provas Perfil Aluno



Interface Web 16 - Consulta Horários Perfil Aluno



Interface Web 17 - Consulta Conteúdo Perfil Aluno

APÊNDICE C – Avaliação de Usabilidade

Avaliação Heuristica - Perfil Aluno							
Questões Avaliação Heuristica	**Telas sistema web - Instituição de Ensino**						
	Login Aluno	**Avisos Aluno**	**Boletim Aluno**	**Materiais Aluno**	**Provas Aluno**	**Horarios Aluno**	**Conteúdo Aluno**
Visibilidade do status do sistema	0	0	0	1	2	0	0
Compatibilidade do sistema com o mundo real	0	0	0	0	0	0	0
Controle do usuário e liberdade	0	1	1	1	1	1	1
Consistência e padrões	1	1	1	1	1	1	1
Prevenção de erros	1	0	0	2	1	2	1
Reconhecimento ao invés de relembrança	0	2	2	2	2	2	2
Flexibilidade e eficiência de uso	0	0	0	0	0	0	0
Estética e projeto minimalista	1	1	1	0	0	2	0
Ajuda aos usuários para reconhecer, diagnosticar e corrigir erros	0	0	0	0	0	0	0
Ajuda e documentação	0	2	1	2	2	1	2
	3	7	6	9	9	9	7

Escala de gravidade do problema
0 – Não concordo que isto seja um problema
1 – Problema cosmético
2 – Problema pequeno
3 – Problema grande
4 – Catastrófico

Avaliador 01
Analista de Sistemas PL

Figura 43 - Avaliação Heurística 1

Avaliação Heuristica - Perfil Aluno							
Questões Avaliação Heuristica	Telas sistema web - Instituição de Ensino						
	Login Aluno	Avisos Aluno	Boletim Aluno	Materiais Aluno	Provas Aluno	Horarios Aluno	Conteúdo Aluno
Visibilidade do status do sistema	0	0	0	2	1	0	0
Compatibilidade do sistema com o mundo real	0	0	0	0	0	0	0
Controle do usuário e liberdade	0	1	1	0	0	1	0
Consistência e padrões	0	0	0	0	0	0	0
Prevenção de erros	0	0	0	3	1	0	0
Reconhecimento ao invés de relembrança	0	1	1	1	1	1	1
Flexibilidade e eficiência de uso	0	0	1	3	0	0	1
Estética e projeto minimalista	1	0	2	0	1	0	0
Ajuda aos usuários para reconhecer, diagnosticar e corrigir erros	0	1	1	2	1	1	1
Ajuda e documentação	1	2	2	2	2	2	2
	2	5	8	13	7	5	5

Escala de gravidade do problema
0 – Não concordo que isto seja um problema
1 – Problema cosmético
2 – Problema pequeno
3 – Problema grande
4 – Catastrófico

Avaliador 02
Programador Pleno

Figura 44 - Avaliação Heurística 2

Avaliação Heuristica - Perfil Aluno							
Questões Avaliação Heuristica	Telas sistema web - Instituição de Ensino						
	Login Aluno	Avisos Aluno	Boletim Aluno	Materiais Aluno	Provas Aluno	Horarios Aluno	Conteúdo Aluno
Visibilidade do status do sistema	0	0	2	4	2	0	0
Compatibilidade do sistema com o mundo real	0	0	0	0	0	0	0
Controle do usuário e liberdade	0	1	1	1	1	1	1
Consistência e padrões	0	0	0	0	0	0	0
Prevenção de erros	0	1	1	1	1	0	0
Reconhecimento ao invés de relembrança	0	2	2	2	2	2	2
Flexibilidade e eficiência de uso	1	1	1	1	1	1	1
Estética e projeto minimalista	1	0	0	2	1	0	0
Ajuda aos usuários para reconhecer, diagnosticar e corrigir erros	0	1	1	1	1	1	1
Ajuda e documentação	3	3	3	3	3	3	3
	5	9	11	15	12	8	8

Escala de gravidade do problema
0 – Não concordo que isto seja um problema
1 – Problema cosmético
2 – Problema pequeno
3 – Problema grande
4 – Catastrófico

Avaliador 03
Programador JR

Figura 45 - Avaliação Heurística 3

Figura 46 - Resultado *Checklist*

APÊNDICE D – Detalhamento Casos de Uso Perfil Aluno

Caso de Uso Autenticação Aluno

Caso de uso responsável pela validação do usuário e senha. Após validação do aluno, é possível realizar as consultas por meio do *smartphone.*

1 Ator

Aluno e WebService

2 Pré-Condição

- O ator Aluno deverá estar devidamente cadastrado no sistema da instituição de ensino;
- O ator Aluno precisa estar devidamente matriculado e com seu *status* ativo;
- O ator WebService deverá estar on-line e sempre disponível para retornar a validação do aluno

3 Pós-Condição

- Resposta do WebService com validação bem sucedida ou não.

4 Requisitos associados

- Fornecer informações se aluno foi autenticado corretamente no sistema da instituição.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – O caso de uso é iniciado quando o Ator Aluno acessa o *app* em um celular e escolhe o perfil “Aluno” na tela principal.
P.2 – O *app* apresenta a tela de *Login* de Aluno.
P.3 – O ator aluno informa o *login* e senha (RA1), (RN1) e (RN2).
P.4 – Clica no botão “Entrar” (A1) e (A2).

b. Fluxo Alternativo

A.1 – Validar Aluno

A.1.1 – O Ator Aluno informa "*login*" e "senha"

A.1.2 – Os parâmetros "*login*" e "senha" informados pelo usuário é repassado para o Ator *Web Service*.

A.1.3 – O *Web Service* faz a validação dos dados e retorna para o *smartphone.*

A.1.4 – Caso o Ator Aluno seja validado, será apresentando uma mensagem "Aluno validou com sucesso" e a tela Consultar será apresentada.

A.1.5 – Caso o Ator Aluno não seja validado, será apresentado uma mensagem "Usuário e/ou senha inválida! Favor verificar."

A.2 – *Web Service* indisponível

A.2.1 – Mensagem "Falha ao consultar *WebService*." é exibida quando não é possível conectar com Ator *Web Service*.

6 Regras

a. Regra de Aplicação

R.A.1 – Os campo "*login*" e "senha" aceitam apenas dígitos numéricos entre 0 e 9.

b. Regra de Negocio

R.N.1 – O "*login*" é composto pelo código de matricula do Aluno, único durante toda a vida escolar do mesmo.

R.N.2 – A "senha" é composta pelos dígitos numéricos da data de nascimento do aluno, desconsiderando a barra (/).

Caso de Uso Consultar Aluno

O caso de uso Consultar é responsável por executar outros casos de uso como deveres de casa, avaliações, faltas e horários, mediante escolha do usuário.

1 Ator

Aluno

2 Pré-Condição

- O ator Aluno escolhe qual consulta deseja realizar no sistema da instituição.

3 Pós-Condição

- É executado o caso de uso que o ator Aluno escolher no *smartphone*.

4 Fluxo de Eventos

a. Fluxo Principal

P.1 – O caso de uso é iniciado após validação de usuário e senha e apresentando a tela consultar do ARE *Mobile* com as opções “Deveres de Casa”, “Avaliações”, “Faltas” e “Horários”.
P.2 – O ator deve escolher a opção que deseja realizar a consulta.

b. Fluxo Alternativo

A.1 – Retorna para caso de uso de autenticação.

Caso de Uso Deveres de Casa Aluno

Caso de uso responsável pela consulta aos deveres de casa do aluno autenticado.

1 Ator

Aluno e *WebService*

2 Pré-Condição

- O ator aluno precisa estar autenticado;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com todos os deveres do aluno, ordenados em ordem cronológica.

4 Requisitos associados

- Fornecer lista de deveres do aluno autenticado.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Deveres de Casa” na tela consultar, o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo a lista de deveres do aluno autenticado.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar
A.2 – *Web Service* indisponível
A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

Caso de Uso Avaliações Aluno

Caso de uso responsável pela consulta das avaliações realizadas ou a serem realizadas, bem como seu valor e nota obtida.

1 Ator

Aluno e *WebService*

2 Pré-Condição

- O ator aluno precisa estar autenticado;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com todas as avaliações realizadas ou a serem realizadas, ordenados em ordem cronológica juntamente com valor da avaliação e nota obtida.

4 Requisitos associados

- Fornecer lista de avaliações do aluno autenticado.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Avaliações” na tela consultar, o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo a lista de avaliações do aluno autenticado.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar.
A.2 – *Web Service* indisponível
A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

Caso de Uso Faltas Aluno

Caso de uso responsável pela consulta das faltas do aluno.

1 Ator

Aluno e *WebService*

2 Pré-Condição

- O ator aluno precisa estar autenticado;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com todas as faltas por matéria em cada bimestre.

4 Requisitos associados

- Fornecer lista de faltas do aluno autenticado.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Faltas” na tela consultar, o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo a lista de faltas do aluno autenticado.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar.
A.2 – *Web Service* indisponível
A.2.1 – Mensagem “Falha ao consultar *WebService.*” é exibida quando não é possível conectar com Ator *Web Service*.

Caso de Uso Horários Aluno

Caso de uso responsável pela consulta dos horários do aluno.

1 Ator

Aluno e *WebService*

2 Pré-Condição

- O ator aluno precisa estar autenticado;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com os horários e matérias para cada dia da semana.

4 Requisitos associados

- Fornecer lista de horários do aluno autenticado.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Horários” na tela consultar o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo a lista de horários do aluno autenticado.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar.

A.2 – *Web Service* indisponível

A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

APÊNDICE E – Detalhamento Casos de Uso Perfil Pais/Responsáveis

Caso de Uso Autenticação Pais/Responsáveis

Este caso de uso é responsável pela validação dos pais ou responsáveis para posteriormente disponibilizar as consultas e acompanhamento escolar do aluno.

1 Ator

Pais/Responsáveis e *WebService*

2 Pré-Condição

- O ator Pai/Responsável deverá estar devidamente cadastrado no sistema da instituição de ensino;
- O ator *WebService* deverá estar on-line e sempre disponível para retornar as informações consultadas.

3 Pós-Condição

- Resposta do *WebService* com validação do responsável bem sucedida ou não.

4 Requisitos associados

- Fornecer para pais/responsável autenticado um acompanhamento do rendimento escolar do aluno do qual ele é responsável.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – O caso de uso é iniciado quando o Ator Pais/Responsável acessa o *app* em um celular e escolhe o perfil “Pais” na tela principal.
P.2 – O *app* apresenta a tela de *Login* de Pais/Responsável.
P.3 – O ator informa o *login* e senha (RA1), (RA2), (RN1) e (RN2).
P.4 – Clica no botão “Entrar” (A1) e (A2).
P.5 – O *app* apresenta a tela Consultar Pais/Responsável com as opções “Avaliações”, “Financeira”, “Ocorrências”, “Horários”, “Boletim” e “Avisos”.

b. Fluxo Alternativo

A.1 – Validar Pais/Responsáveis

A.1.1 – O Ator Pais/Responsáveis informa "*login*" e "senha"

A.1.2 – Os parâmetros "*login*" e "senha" informados pelo usuário é repassado para o Ator *Web Service*.

A.1.3 – O *Web Service* faz a validação dos dados e retorna para o *smartphone.*

A.1.4 – Caso o Ator Pais/Responsável seja validado, será apresentando uma mensagem "Responsável validou com sucesso" e a tela Consultar Pais/Responsável será apresentado.

A.1.5 – Caso o Ator Pais/Responsável não seja validado, será apresentada uma mensagem "Usuário e/ou senha inválida! Favor verificar".

A.2 – *Web Service* indisponível

A.2.1 – Mensagem "Falha ao consultar *WebService*." é exibida quando não é possível conectar com Ator *Web Service*.

6 Regras

a. Regra de Aplicação

R.A.1 – O campo "*login*" aceita apenas dígitos numéricos entre 0 e 9 composto de 11 dígitos.

R.A.2 - O campo "senha" aceita apenas dígitos numéricos entre 0 e 9.

b. Regra de Negocio

R.N.1 – O "*login*" é composto pelo CPF do responsável desconsiderando qualquer pontuação com ponto (.) e hífen / traço (-).

R.N.2 – A "senha" é composta pelo número de matricula do aluno.

Caso de Uso Consultar Pais/Responsáveis

O caso de uso Consultar é responsável por executar outros casos de uso pertinentes ao perfil pais/responsáveis, como: avaliações, financeiros, ocorrências, horários, boletim e avisos, mediante escolha do usuário.

1 Ator

Pais/Responsáveis

2 Pré-Condição

- O ator Pais/Responsáveis deve escolher qual consulta deseja realizar no sistema da instituição.

3 Pós-Condição

- É executado o caso de uso que o ator Pais/Responsáveis escolher no *smartphone*.

4 Fluxo de Eventos

a. Fluxo Principal

P.1 – O caso de uso é iniciado após validação de usuário e senha e apresentando a tela consultar do ARE *Mobile* com as opções “Avaliações”, “Financeiros”, “Ocorrências”, “Horários”, “Boletim” e “Avisos”.
P.2 – O ator deve escolher a opção que deseja realizar a consulta.

b. Fluxo Alternativo

A.1 – Retorna para caso de uso de autenticação.

Caso de Uso Financeiros Pais/Responsáveis

Caso de uso responsável pela consulta financeira do aluno. Por meio dessa consulta é possível acompanhar a situação financeira junto à instituição de ensino.

1 Ator

Pais/Responsáveis e *WebService*

2 Pré-Condição

- O ator pais/responsáveis precisa estar autenticado;
- O ator pais/responsáveis precisar ter um aluno devidamente matriculado na instituição de ensino;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Lista com a situação financeira do aluno, ordenados por mês de referencia.

4 Requisitos associados

- Fornecer lista de pendências financeiras do aluno.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Financeiros” na tela consultar, o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo a lista de situação financeira do aluno.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar
A.2 – *Web Service* indisponível
A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

Caso de Uso Avaliações Pais/Responsáveis

Caso de uso responsável pela consulta das avaliações realizadas ou a serem realizadas, bem como seu valor e nota obtida.

1 Ator

Pais/Responsáveis e *WebService*

2 Pré-Condição

- O ator pais/responsáveis precisa estar autenticado;
- O ator pais/responsáveis precisar ter um aluno devidamente matriculado na instituição de ensino;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com todas as avaliações realizadas ou a serem realizadas, ordenados em ordem cronológica juntamente com valor da avaliação e nota obtida.

4 Requisitos associados

- Fornecer lista para pais/responsáveis das avaliações do aluno.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Avaliações” na tela consultar, o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo a lista de avaliações do aluno.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar.
A.2 – *Web Service* indisponível
A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

Caso de Uso Ocorrências Pais/Responsáveis

Caso de uso responsável pela consulta das ocorrências do aluno. Nela é possível obter informações como não entrega de deveres, descumprimento das normas da instituição, chamadas na secretaria e comportamento indevido.

1 Ator

Pais/Responsáveis e *WebService*

2 Pré-Condição

- O ator pais/responsáveis precisa estar autenticado;
- O ator pais/responsáveis precisar ter um aluno devidamente matriculado na instituição de ensino;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com todas as ocorrências do aluno, registradas durante o ano letivo na instituição.

4 Requisitos associados

- Fornecer lista de ocorrências do aluno para o pais/responsáveis autenticado.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Ocorrências” na tela consultar, o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo a lista de ocorrências do aluno.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar.
A.2 – *Web Service* indisponível
A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

Caso de Uso Horários Pais/Responsáveis

Caso de uso responsável pela consulta dos horários do aluno.

1 Ator

Pais/Responsáveis e *WebService*

2 Pré-Condição

- O ator pais/responsáveis precisa estar autenticado;
- O ator pais/responsáveis precisar ter um aluno devidamente matriculado na instituição de ensino;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com os horários e matérias para cada dia da semana.

4 Requisitos associados

- Fornecer lista de horários do aluno para pais/responsáveis autenticado.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Horários” na tela consultar, o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo a lista de horários do aluno.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar.
A.2 – *Web Service* indisponível
A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

Caso de Uso Boletim Pais/Responsáveis

Caso de uso responsável pela consulta das médias obtidas pelo aluno.

1 Ator

Pais/Responsáveis e *WebService*

2 Pré-Condição

- O ator pais/responsáveis precisa estar autenticado;
- O ator pais/responsáveis precisar ter um aluno devidamente matriculado na instituição de ensino;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com todas as matérias e médias obtidas durante o bimestre pelo aluno.

4 Requisitos associados

- Fornecer lista com as matérias e médias do aluno obtidas durante o bimestre.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Boletim” na tela consultar, o caso de uso é executado.
P.2 – O *app* apresenta a tela contendo o boletim do aluno.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar.
A.2 – *Web Service* indisponível
A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

Caso de Uso Avisos Pais/Responsáveis

Caso de uso responsável pela exibição dos avisos encaminhados para os pais/responsáveis.

1 Ator

Pais/Responsáveis e *WebService*

2 Pré-Condição

- O ator pais/responsáveis precisa estar autenticado;
- O ator pais/responsáveis precisar ter um aluno devidamente matriculado na instituição de ensino;
- O ator *webservice* deverá estar on-line e disponível para acesso.

3 Pós-Condição

- Listagem com os avisos encaminhados pela secretária da instituição.

4 Requisitos associados

- Fornecer aos pais/responsáveis autenticados uma lista de avisos encaminhados pela secretária da instituição.

5 Fluxo de Eventos

a. Fluxo Principal

P.1 – Ao selecionar a opção “Avisos” na tela consultar, o caso de uso é executado.

P.2 – O *app* apresenta a tela contendo a lista de avisos aos pais/responsáveis.

b. Fluxo Alternativo

A.1 – Voltar à tela Consultar.

A.2 – *Web Service* indisponível

A.2.1 – Mensagem “Falha ao consultar *WebService*.” é exibida quando não é possível conectar com Ator *Web Service*.

APÊNDICE F – Código Fonte *Web Service* – Java

InformacoesAluno.java

```
package br.com.mobile.are;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

public class InformacoesAluno {
    Connection connection = null;

    public InformacoesAluno() {
        this.connect();
    }

    void close() {
        try {
            this.connection.close();
        } catch (SQLException ex) {
            Logger.getLogger(InformacoesAluno.class.getName()).log(
                    Level.SEVERE, null, ex);
        }
    }

    void connect() {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        } catch (ClassNotFoundException e) {
            System.out.println("Where is your SQLServer 2008 JDBC Driver? "
                    + "Include in your library path!");
            e.printStackTrace();
            return;
        }
        this.connection = null;
        try {
            this.connection = DriverManager
                    .getConnection(
                            "jdbc:sqlserver://192.168.0.110:1433;databaseName=ARE_MOBILE_ANDROID;instanceName=AGF",
                            "userRoot", "@dmintesteare");
        } catch (SQLException e) {
            System.out.println("Connection Failed! Check output console");
            e.printStackTrace();
            return;
        }
        if (this.connection != null) {
        } else {
            System.out.println("Failed to make connection!");
        }
    }
```

Figura 47 - WS Parte 1 - Conexão Banco de Dados

InformacoesAluno.java

```
    /*
     * Metodo para Validar Aluno
     *
     */
    public String validarLoginAluno(String loginAluno, String senhaAluno){
        String retorno = "";
        try {
            Statement sq_stmt = this.connection.createStatement();
            String sql_str = "select COUNT(*) as valida "
                    + "from ALUNO "
                    + "where ID_ALUNO ='"+loginAluno+"'"
                    + " and DATA_NASCIMENTO_ALUNO = '"+senhaAluno+"' and STATUS_ALUNO = 1";
            ResultSet rs = sq_stmt.executeQuery(sql_str);
            rs.next();
            retorno = rs.getString("valida");
            System.out.println("--------------- Cliente Android Acesso Web Service ARE Mobile -----------------");
            System.out.println("---------------------------- Validação Aluno ---------------------------------");
            System.out.println("------->> Consulta no banco de dados: "+sql_str );

        } catch (SQLException e) {
            e.printStackTrace();
        }
        System.out.println("   ----------------------- Condição / Resultado da Validação ------------");
        System.out.println("   ------>> Se resultado é 1 [True ]-->Aluno validou com sucesso! -------");
        System.out.println("   ------>> Se resultado é 0 [False]-->Validação de Aluno falhou! -------");
        System.out.println("   ------>> Resultado da consulta: " +retorno +" <<------------");
        return retorno;
    }
```

Figura 48 - WS Parte 2 - Método Validar Aluno

InformacoesAluno.java

```
    /*
     * Metodo Listar Materias
     *
     */
    public String consultarListaMateriasAluno(String codigoAluno) {
        String retorno = "";
        try {
            Statement sq_stmt = this.connection.createStatement();
            String sql_str = "select m.ID_MATERIA as codigo, m.DESCRICAO_MATERIA as descricao "
                    + "from ALUNO a, MATERIA m "
                    + "where a.ID_SERIE = m.ID_SERIE and  m.ANO_REFERENCIA_MATERIA = YEAR(GETDATE()) and a.ID_ALUNO ="
                    + codigoAluno;
            System.out.println("--------------- Cliente Android Acesso Web Service ARE Mobile -----------------");
            System.out.println("------------- Lista Materias do Aluno (XML) ----------------");
            System.out.println("-------->> Consulta no banco de dados: "+sql_str);
            ResultSet rs = sq_stmt.executeQuery(sql_str);
            retorno += "<materias>";
            while (rs.next()) {
                retorno += "<item>" + rs.getString("descricao") + "</item>";
            }
            retorno += "</materias>";
        } catch (SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }
        System.out.println("----------->> Resultado (XML) da consulta - Materias Aluno: -------------------------- ");
        System.out.println(retorno);
        System.out.println("-------------------- Final do XML ----------------------");
        return retorno;
    }
```

Figura 49 - WS Parte 3 - Método Listar Matérias Aluno

InformacoesAluno.java

```
    /*
     * Metodo Informações do Aluno
     *
     */
    public String consultarInformacoesAluno(String codigoAluno) {
        String retorno = "";
        try {
            Statement sq_stmt = this.connection.createStatement();
            String sql_str = "select a.NOME_ALUNO as nome, s.DESCRICAO_SERIE as serie, t.DESCRICAO_TURMA as turma "
                    + "from ALUNO a, SERIE s, TURMA t  "
                    + "where s.ID_SERIE=a.ID_SERIE and t.ID_TURMA = a.ID_TURMA and a.ID_ALUNO ="
                    + codigoAluno;
            System.out.println("--------------- Cliente Android Acesso Web Service ARE Mobile -----------------");
            System.out.println(" ---------------- Consulta Lista (XML) Informações do Aluno --------------------");
            System.out.println("-------->> Select no banco de dados: "+sql_str);
            ResultSet rs = sq_stmt.executeQuery(sql_str);
            retorno +="<menu>";
            while (rs.next()) {
                retorno += "<aluno>";
                retorno += "<nome>" + rs.getString("nome") + "</nome>";
                retorno += "<serie>" + rs.getString("serie") + "</serie>";
                retorno += "<turma>" + rs.getString("turma") + "</turma>";
                retorno += "</aluno>";
            }
            retorno +="</menu>";
        } catch (SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }
        System.out.println("------------>> Resultado (XML) da consulta - Informações Aluno: --------------------------- ");
        System.out.println(retorno);
        System.out.println("--------------------- Final do XML ------------------------");
        return retorno;
    }
```

Figura 50 - WS Parte 4 - Método Informações do Aluno

InformacoesAluno.java

```
    /*
     *  Metodo Listar Avaliações
     */
    public String consultarListaAvaliacoesAluno(String codigoAluno) {
        String retorno = "";
        try {
            Statement sq_stmt = this.connection.createStatement();
            String sql_str = "select m.DESCRICAO_MATERIA as materia, b.DESCRICAO_BIMESTRE as bimestre, (CONVERT(VARCHAR(11),a.DATA_AVALIACOES, 106)) as data, "
                    + "a.DESCRICAO_AVALIACOES as descricao, a.PESO_AVALIACOES as valor, a.VALOR_AVALIACOES as nota "
                    + "from AVALIACOES a, BIMESTRE_REFERENCIA b, MATERIA m "
                    + "where m.ID_MATERIA = a.ID_MATERIA and b.ID_BIMESTRE = a.ID_BIMESTRE and a.ID_ALUNO ="
                    + codigoAluno + " order by a.ID_MATERIA desc";
            System.out.println("--------------- Cliente Android Acesso Web Service ARE Mobile -----------------");
            System.out.println(" ---------------- Consulta Lista (XML) de Avaliações do Aluno --------------------");
            System.out.println("-------->> Select no banco de dados: "+sql_str);
            ResultSet rs = sq_stmt.executeQuery(sql_str);
            retorno +="<menu>";
            while (rs.next()) {
                retorno += "<avaliacoes>";
                retorno += "<materia>" + rs.getString("materia") + "</materia>";
                retorno += "<bimestre>" + rs.getString("bimestre") + "</bimestre>";
                retorno += "<data>" + rs.getString("data") + "</data>";
                retorno += "<descricao>" +rs.getString("descricao")+ "</descricao>";
                retorno += "<peso>" +rs.getString("valor")+ "</peso>";
                retorno += "<nota>" +rs.getString("nota")+ "</nota>";
                retorno += "</avaliacoes>";
            }
            retorno +="</menu>";
        } catch (SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }
        System.out.println("------------>> Resultado (XML) da consulta - Avaliação Aluno: --------------------------- ");
        System.out.println(retorno);
        System.out.println("--------------------- Final do XML ------------------------");
        return retorno;
    }
```

Figura 51 - WS Parte 5 - Método Listar Avaliações Aluno

InformacoesAluno.java

```
    /*
     * Metodo Listar Horarios
     *
     */
    public String consultarListaHorariosAluno(String codigoAluno) {
        String retorno = "";
        try {
            Statement sq_stmt = this.connection.createStatement();

            String sql_str = "select s.DESCRICAO_SEMANA as semana, h.SEQUENCIA_DIA_HORARIO as sequencia, CONVERT(VARCHAR(5),h.HORARIO_HORARIO,108) as hora, "
                    + "m.DESCRICAO_MATERIA as materia "
                    + "from HORARIO h, ALUNO a, SEMANA s, MATERIA m "
                    + "where a.ID_SERIE = h.ID_SERIE and a.ID_TURMA = h.ID_TURMA "
                    + "and s.ID_SEMANA = h.DIA_SEMANA_HORARIO and m.ID_MATERIA = h.ID_MATERIA and a.ID_ALUNO = "
                    + codigoAluno;
            System.out.println("--------------- Cliente Android Acesso Web Service ARE Mobile ----------------");
            System.out.println(" ---------------- Consulta Lista (XML) de Avaliações do Aluno --------------------");
            System.out.println("-------->> Select no banco de dados: "+sql_str);
            ResultSet rs = sq_stmt.executeQuery(sql_str);
            retorno +="<menu>";
            while (rs.next()) {
                retorno += "<horario>";
                retorno += "<semana>" + rs.getString("semana") + "</semana>";
                retorno += "<sequencia>" + rs.getString("sequencia") + "</sequencia>";
                retorno += "<materia>" +rs.getString("materia")+ "</materia>";
                retorno += "<hora>" + rs.getString("hora") + "</hora>";
                retorno += "</horario>";
            }
            retorno +="</menu>";
        } catch (SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }
        System.out.println("----------->> Resultado (XML) da consulta - Avaliação Aluno: -------------------------- ");
        System.out.println(retorno);
        System.out.println("-------------------- Final do XML ----------------------");
        return retorno;
    }
```

Figura 52 - WS Parte 6 - Método Listar Horários Aluno

InformacoesAluno.java

```
     * Metodo Listar Faltas
     *
     */
      public String consultarListaFaltasAluno(String codigoAluno) {
        String retorno = "";
        try {
            Statement sq_stmt = this.connection.createStatement();

            String sql_str = "select SUM(f.FALTAS_FREQUENCIA) as faltas, m.DESCRICAO_MATERIA as materia, b.DESCRICAO_BIMESTRE as bimestre "
                    + "from FREQUENCIA f, BIMESTRE_REFERENCIA b, MATERIA m "
                    + "where f.ID_BIMESTRE_REFERENCIA = b.ID_BIMESTRE and m.ID_MATERIA = f.ID_MATERIA and f.ID_ALUNO = "
                    + codigoAluno
                    + " group by b.DESCRICAO_BIMESTRE, m.DESCRICAO_MATERIA "
                    + "order by b.DESCRICAO_BIMESTRE asc";
            System.out.println("--------------- Cliente Android Acesso Web Service ARE Mobile ----------------");
            System.out.println(" ---------------- Consulta Lista (XML) de Faltas do Aluno --------------------");
            System.out.println("-------->> Select no banco de dados: "+sql_str);
            ResultSet rs = sq_stmt.executeQuery(sql_str);
            retorno +="<menu>";
            while (rs.next()) {
                retorno += "<frequencia>";
                retorno += "<materia>" + rs.getString("materia") + "</materia>";
                retorno += "<faltas>" + rs.getString("faltas") + "</faltas>";
                retorno += "<bimestre>" + rs.getString("bimestre") + "</bimestre>";
                retorno += "</frequencia>";
            }
            retorno +="</menu>";
        } catch (SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }
        System.out.println("----------->> Resultado (XML) da consulta - Faltas Aluno: -------------------------- ");
        System.out.println(retorno);
        System.out.println("-------------------- Final do XML ----------------------");
        return retorno;
    }
}
```

Figura 53 - WS Parte 7 - Método Listar Faltas Aluno

Controle.java

```
package br.com.mobile.are;

import java.util.ArrayList;

import javax.faces.bean.ManagedBean;

@ManagedBean
public class Controle {

    private String codigoAluno;
    private String senhaAluno;
    private String loginAluno;
    private String saida;
    private ArrayList<String> s;

    public String validaAluno() {
        InformacoesAluno login = new InformacoesAluno();
        saida = login.validarLoginAluno(loginAluno, senhaAluno);
        System.out.println(" -------> Controle:" + saida);
        return "/aluno/resultado_login_aluno";
    }

    public String consultarListaMateriasAluno() {
        if(codigoAluno != null && codigoAluno.length()>0){
        InformacoesAluno lista = new InformacoesAluno();
            saida = lista.consultarListaMateriasAluno(codigoAluno);
            return "/aluno/resultado_lista_materias";
        } else {
            return "/publico/falha";
        }
    }

public String consultarListaDeveresAluno() {
    if(codigoAluno != null && codigoAluno.length()>0){
    InformacoesAluno listaDeveres = new InformacoesAluno();
        saida = listaDeveres.consultarListaDeveresAluno(codigoAluno);
        return "/aluno/resultado_lista_deveres";
    } else {
        return "/publico/falha";
    }
}

public String consultarInformacoesAluno() {
    if(codigoAluno != null && codigoAluno.length()>0){
    InformacoesAluno listaAluno = new InformacoesAluno();
        saida = listaAluno.consultarInformacoesAluno(codigoAluno);
        return "/aluno/resultado_informacoes_aluno";
    } else {
        return "/publico/falha";
    }
}

public String consultarListaAvaliacoesAluno() {
    if(codigoAluno != null && codigoAluno.length()>0){
    InformacoesAluno avaliacaoAluno = new InformacoesAluno();
        saida = avaliacaoAluno.consultarListaAvaliacoesAluno(codigoAluno);
        return "/aluno/resultado_avaliacoes_aluno";
    } else {
        return "/publico/falha";
    }
}
```

Figura 54 - WS Controle Parte 1

```
    public String consultarListaHorariosAluno() {
        if(codigoAluno != null && codigoAluno.length()>0){
        InformacoesAluno avaliacaoAluno = new InformacoesAluno();
            saida = avaliacaoAluno.consultarListaHorariosAluno(codigoAluno);
            return "/aluno/resultado_horarios_aluno";
        } else {
            return "/publico/falha";
        }
    }

    public String consultarListaFaltasAluno() {
        if(codigoAluno != null && codigoAluno.length()>0){
        InformacoesAluno faltasAluno = new InformacoesAluno();
            saida = faltasAluno.consultarListaFaltasAluno(codigoAluno);
            return "/aluno/resultado_faltas_aluno";
        } else {
            return "/publico/falha";
        }
    }

    public String principalAluno(){
        return "/publico/aluno";
    }

    public String principalPais(){
        return "/publico/responsavel";
    }

    public String voltarAluno() {
        return "/publico/aluno";
    }

    public String voltarResponsavel() {
        return "/publico/responsavel";
    }


   public String voltarAluno() {
       return "/publico/aluno";
   }

   public String voltarResponsavel() {
       return "/publico/responsavel";
   }

   public String voltarPrincipal() {
       return "/index";
   }

   public String getLoginAluno() {
       return loginAluno;
   }

   public void setLoginAluno(String loginAluno) {
       this.loginAluno = loginAluno;
   }

   public String getCodigoAluno() {
       return codigoAluno;
   }

   public void setCodigoAluno(String codigoAluno) {
       this.codigoAluno = codigoAluno;
   }

   public String getSenhaAluno() {
       return senhaAluno;
   }

   public void setSenhaAluno(String senhaAluno) {
       this.senhaAluno = senhaAluno;
   }

  public String getSaida() {
      return saida;
  }

  public void setSaida(String saida) {
      this.saida = saida;
  }


  public ArrayList<String> getS() {
      return s;
  }


  public void setS(ArrayList<String> s) {
      this.s = s;
  }
}
```

Figura 55 - WS Controle Parte 2

APÊNDICE G – Código Fonte *Android – Java*

```java
package br.com.app.aremobile;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PrincipalAlunoActivity extends Activity {
    static final String KEY_CODIGO_ALUNO = "codigoAluno";
    static final String KEY_INFO_ALUNO = "infoAluno";
    private String codigo;
    private String infoAluno;
    private TextView txtInfoAluno;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal_aluno);
        Intent in = getIntent();
        codigo = in.getStringExtra(KEY_CODIGO_ALUNO);
        infoAluno = in.getStringExtra(KEY_INFO_ALUNO);
        txtInfoAluno = (TextView) findViewById(R.id.tvRodape);
        txtInfoAluno.setText(infoAluno);
 }

    public void onClickDeveres(View v) {
        Intent deveres = new Intent(this, DeveresAlunoActivity.class);
        deveres.putExtra(KEY_CODIGO_ALUNO, codigo);
        startActivity(deveres);
    }

    public void onClickNotas(View v){
        Intent notas = new Intent(this, AvaliacoesAlunoActivity.class);
        notas.putExtra(KEY_CODIGO_ALUNO, codigo);
        startActivity(notas);
    }

    }

    public void onClickHorarios(View v){
        Intent horarios = new Intent(this, HorariosAlunoActivity.class);
        horarios.putExtra(KEY_CODIGO_ALUNO, codigo);
        startActivity(horarios);
    }

    public void onClickFaltas(View v){
        Intent faltas = new Intent(this, FaltasAlunoActivity.class);
        faltas.putExtra(KEY_CODIGO_ALUNO, codigo);
        startActivity(faltas);
    }

}
```

Figura 56 - ARE *Mobile Android - Activity* Principal Aluno

```
package br.com.app.aremobile;
import java.util.ArrayList;

public class AvaliacoesAlunoActivity extends ListActivity {

    // XML node keys
        static final String KEY_ITEM = "avaliacoes"; // parent node
        static final String KEY_MATERIA = "materia";
        static final String KEY_DESC = "descricao";
        static final String KEY_DATA = "data";
        static final String KEY_BIMESTRE = "bimestre";
        static final String KEY_PESO = "peso";
        static final String KEY_NOTA = "nota";
        static final String KEY_CODIGO_ALUNO = "codigoAluno";

    private Webservice ws;
    private String xml;
    private String codigo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_avaliacoes_aluno);
        ws = new Webservice();
        Intent in = getIntent();
        codigo = in.getStringExtra(KEY_CODIGO_ALUNO);
        xml = ws.consultarListaAvaliacoesAluno(codigo);
        ArrayList<HashMap<String, String>> menuItems = new ArrayList<HashMap<String, String>>();
        XMLParser parser = new XMLParser();
        Document doc = parser.getDomElement(xml);
        NodeList nl = doc.getElementsByTagName(KEY_ITEM);
        for (int i = 0; i < nl.getLength(); i++) {
            HashMap<String, String> map = new HashMap<String, String>();
            Element e = (Element) nl.item(i);
            map.put(KEY_MATERIA, parser.getValue(e, KEY_MATERIA));
            map.put(KEY_DESC, parser.getValue(e, KEY_DESC));
            map.put(KEY_DATA, "Data da avaliação: " + parser.getValue(e, KEY_DATA));
            map.put(KEY_BIMESTRE, parser.getValue(e, KEY_BIMESTRE));
            map.put(KEY_PESO, "Valor da avaliação: "+parser.getValue(e, KEY_PESO));
            map.put(KEY_NOTA, "Nota obtida: "+parser.getValue(e, KEY_NOTA));
            menuItems.add(map);
        }
        ListAdapter adapter = new SimpleAdapter(this, menuItems,
                R.layout.list_item_avaliacoes,
                new String[] { KEY_MATERIA, KEY_DESC, KEY_DATA, KEY_BIMESTRE, KEY_PESO, KEY_NOTA },new int[]{
                        R.id.lbl_materia_avaliacoes, R.id.lbl_descricao_avaliacoes, R.id.lbl_data_avaliacoes, R.id.lbl_bimestre_avaliacoes,
                        R.id.lbl_peso_avaliacoes, R.id.lbl_nota_avaliacoes });
        setListAdapter(adapter);
        ListView lv = getListView();
        lv.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                    int position, long id) {
                String materia = ((TextView) view.findViewById(R.id.lbl_materia)).getText().toString();
                String data = ((TextView) view.findViewById(R.id.lbl_data)).getText().toString();
                String descricao = ((TextView) view.findViewById(R.id.lbl_descricao)).getText().toString();
                Intent in = new Intent(getApplicationContext(), SingleMenuItemActivity.class);
                in.putExtra(KEY_MATERIA, materia);
                in.putExtra(KEY_DATA, data);
                in.putExtra(KEY_DESC, descricao);
                startActivity(in);
            }
        });


    }
```

Figura 57 - ARE *Mobile Android - Activity* Avaliações Aluno

```
package br.com.app.aremobile;

import java.util.ArrayList;

public class DeveresAlunoActivity extends ListActivity {

    static final String KEY_ITEM = "deveres";
    static final String KEY_MATERIA = "materia";
    static final String KEY_DESC = "descricao";
    static final String KEY_DATA = "data_entrega";
    static final String KEY_CODIGO_ALUNO = "codigoAluno";
    private Webservice ws;
    private String xml;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_deveres_aluno);
        ws = new Webservice();
        Intent in = getIntent();
        String codigo = in.getStringExtra(KEY_CODIGO_ALUNO);
        xml = ws.consultarListaDeveresAluno(codigo);
        ArrayList<HashMap<String, String>> menuItems = new ArrayList<HashMap<String, String>>();
        XMLParser parser = new XMLParser();
        Document doc = parser.getDomElement(xml);
        NodeList nl = doc.getElementsByTagName(KEY_ITEM);
        for (int i = 0; i < nl.getLength(); i++) {
            HashMap<String, String> map = new HashMap<String, String>();
            Element e = (Element) nl.item(i);
            map.put(KEY_MATERIA, parser.getValue(e, KEY_MATERIA));
            map.put(KEY_DESC, parser.getValue(e, KEY_DESC));
            map.put(KEY_DATA, "Data de entrega: "+parser.getValue(e, KEY_DATA));
            menuItems.add(map);
        }
        ListAdapter adapter = new SimpleAdapter(this, menuItems,
                R.layout.list_item_deveres,
                new String[] { KEY_MATERIA, KEY_DESC, KEY_DATA },new int[]{
                        R.id.lbl_materia_deveres, R.id.lbl_descricao_deveres, R.id.lbl_data_deveres });
        setListAdapter(adapter);
        ListView lv = getListView();
        lv.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                    int position, long id) {
                String materia = ((TextView) view.findViewById(R.id.lbl_materia_deveres)).getText().toString();
                String data = ((TextView) view.findViewById(R.id.lbl_data_deveres)).getText().toString();
                String descricao = ((TextView) view.findViewById(R.id.lbl_descricao_deveres)).getText().toString();
                Intent in = new Intent(getApplicationContext(), SingleMenuItemActivity.class);
                in.putExtra(KEY_MATERIA, materia);
                in.putExtra(KEY_DATA, data);
                in.putExtra(KEY_DESC, descricao);
                startActivity(in);

            }
        });


    }

}
```

Figura 58 - ARE *Mobile Android - Activity* Deveres Aluno

```
package br.com.app.aremobile;

import java.util.ArrayList;

public class FaltasAlunoActivity extends ListActivity {

    static final String KEY_ITEM = "frequencia";
    static final String KEY_FALTA = "faltas";
    static final String KEY_MATERIA = "materia";
    static final String KEY_BIMESTRE = "bimestre";
    static final String KEY_CODIGO_ALUNO = "codigoAluno";
    private Webservice ws;
    private String xml;
    private String codigo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_faltas_aluno);
        ws = new Webservice();
        Intent in = getIntent();
        codigo = in.getStringExtra(KEY_CODIGO_ALUNO);
        xml = ws.consultarListaFaltasAluno(codigo);
        ArrayList<HashMap<String, String>> menuItems = new ArrayList<HashMap<String, String>>();
        XMLParser parser = new XMLParser();
        Document doc = parser.getDomElement(xml);
        NodeList nl = doc.getElementsByTagName(KEY_ITEM);
        for (int i = 0; i < nl.getLength(); i++) {
            HashMap<String, String> map = new HashMap<String, String>();
            Element e = (Element) nl.item(i);
            map.put(KEY_MATERIA, parser.getValue(e, KEY_MATERIA));
            map.put(KEY_FALTA, "Qtde. Faltas: "+parser.getValue(e, KEY_FALTA));
            map.put(KEY_BIMESTRE, "|    "+parser.getValue(e, KEY_BIMESTRE));
            menuItems.add(map);
        }
        ListAdapter adapter = new SimpleAdapter(this, menuItems,R.layout.list_item_faltas,new String[] { KEY_MATERIA, KEY_FALTA, KEY_BIMESTRE},new int[]{
                        R.id.lbl_materia_faltas, R.id.lbl_falta_faltas, R.id.lbl_bimestre_faltas});
                setListAdapter(adapter);
        }


}
```

Figura 59 - ARE *Mobile Android - Activity* Faltas Aluno

```
package br.com.app.aremobile;

import java.util.ArrayList;

public class HorariosAlunoActivity extends ListActivity {
    static final String KEY_ITEM = "horario";
    static final String KEY_SEMANA = "semana";
    static final String KEY_SEQUENCIA = "sequencia";
    static final String KEY_MATERIA = "materia";
    static final String KEY_HORA = "hora";
    static final String KEY_CODIGO_ALUNO = "codigoAluno";
    private Webservice ws;
    private String xml;
    private String codigo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_horarios_aluno);
        ws = new Webservice();
        Intent in = getIntent();
        codigo = in.getStringExtra(KEY_CODIGO_ALUNO);
        xml = ws.consultarListaHorariosAluno(codigo);
        ArrayList<HashMap<String, String>> menuItems = new ArrayList<HashMap<String, String>>();
        XMLParser parser = new XMLParser();
        Document doc = parser.getDomElement(xml);
        NodeList nl = doc.getElementsByTagName(KEY_ITEM);
        for (int i = 0; i < nl.getLength(); i++) {
            HashMap<String, String> map = new HashMap<String, String>();
            Element e = (Element) nl.item(i);
            map.put(KEY_SEMANA, parser.getValue(e, KEY_SEMANA));
            map.put(KEY_SEQUENCIA, parser.getValue(e, KEY_SEQUENCIA)+"°");
            map.put(KEY_MATERIA, parser.getValue(e, KEY_MATERIA));
            map.put(KEY_HORA, parser.getValue(e, KEY_HORA));
            menuItems.add(map);
        }
        ListAdapter adapter = new SimpleAdapter(this, menuItems,
                R.layout.list_item_horarios,
                new String[] { KEY_SEMANA, KEY_SEQUENCIA, KEY_MATERIA, KEY_HORA},new int[]{
                        R.id.lbl_semana_horario, R.id.lbl_sequencia_horario, R.id.lbl_materia_horario, R.id.lbl_hora_horario});
        setListAdapter(adapter);
        }

}
```

Figura 60 - ARE *Mobile Android - Activity* Horários Aluno

```
package br.com.app.aremobile;

import org.w3c.dom.Document;

public class LoginAlunoActivity extends Activity {
    static final String KEY_CODIGO_ALUNO = "codigoAluno";
    static final String KEY_INFO_ALUNO = "infoAluno";
    static final String KEY_ITEM = "aluno";
    static final String KEY_NOME = "nome";
    static final String KEY_SERIE = "serie";
    static final String KEY_TURMA = "turma";
    private Webservice ws;
    private String xml;
    private String descAluno;
    private EditText txtUsuario;
    private EditText txtSenha;
    private String resultado;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login_aluno);
        ws = new Webservice();
        txtUsuario = (EditText) findViewById(R.id.etUsuario);
        txtSenha = (EditText) findViewById(R.id.etSenha);
    }

    public void doValidarAluno(View v) {
        resultado = null;
        if (txtUsuario != null && txtSenha.length() > 0) {
            String login = txtUsuario.getText().toString();
            String senha = txtSenha.getText().toString();
            resultado = ws.validarLoginAluno(login, senha);
            if (resultado.endsWith("1")) {
                descAluno = getAluno(login);
                Intent in = new Intent(this,PrincipalAlunoActivity.class);
                in.putExtra(KEY_CODIGO_ALUNO, login);
                in.putExtra(KEY_INFO_ALUNO, descAluno);
                startActivity(in);
                resultado = "Aluno validado com sucesso!!";
                this.finish();
            }
            if (resultado.endsWith("0")){
                resultado = "Usuário e/ou senha inválida! Favor verificar.";
            }
        }
        if (resultado== null) {
                resultado = "Falha ao consultar webservice.";
        }
        trace(resultado);
    }

    public String getAluno(String codigo){
        xml = ws.consultarInformacoesAluno(codigo);
        XMLParser parser = new XMLParser();
        Document doc = parser.getDomElement(xml);
        NodeList nl = doc.getElementsByTagName(KEY_ITEM);
        Element e = (Element) nl.item(0);
        String infoAluno =  "Olá "+parser.getValue(e, KEY_NOME)+ "| "+ parser.getValue(e, KEY_SERIE)+" '"+parser.getValue(e, KEY_TURMA)+"'";
        return infoAluno;
    }

    public void toast(String msg) {
        Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
    }

    private void trace(String msg) {
        toast(msg);
    }
}
```

Figura 61 - ARE *Mobile Android - Activity Login* Aluno

```java
package br.com.app.aremobile;

import android.app.Activity;

public class PrincipalActivity extends Activity {
    private TextView txtInternet;
    String msgInternet = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);
        txtInternet = (TextView) findViewById(R.id.tvRodape);
        if (VerificaConexao(this)){
            msgInternet = "Conexão com a internet está disponível.";
        }else{
            msgInternet= "Internet indisponível no momento. Tente mais tarde.";
        }
        txtInternet.setText(msgInternet);
    }

    public void onClickAluno(View v) {
        trace("Aluno.");
        Intent loginAluno = new Intent(this, LoginAlunoActivity.class);
        startActivity(loginAluno);
    }

    public void onClickResponsavel(View v) {
        trace("Pais / Responsavel.");
    }

    public void onClickProfessor(View v) {
        trace("Professor.");
    }

    public void onClickCoordenador(View v) {
        trace("Coordenador");
    }

    public void onClickSair(View v){
        Mensangens();
    }

    public void toast(String msg) {
        Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
    }

    private void trace(String msg) {
        toast(msg);
    }

    private void Mensangens(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Finalizar ARE Mobile").setMessage("Deseja sair do ARE Mobile?")
        .setIcon(android.R.drawable.ic_dialog_alert).setPositiveButton("Sim", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                finish();
                Toast.makeText(PrincipalActivity.this, "ARE Mobile finalizado!", Toast.LENGTH_SHORT).show();
            }
        }).setNegativeButton("Não", null).show();

    }

    public static boolean VerificaConexao(Context contexto) {
        ConnectivityManager cm = (ConnectivityManager) contexto
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if ((netInfo != null) && (netInfo.isConnectedOrConnecting())
                && (netInfo.isAvailable()))
            return true;
        else
            return false;
    }
}
```

Figura 62 - ARE *Mobile Android - Activity* Principal