UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CORDENAÇÃO DE LICENCIATURA EM INFORMÁTICA

DESEMVOLVIMENTO DE SISTEMAS PARA INTERNET E DISPOSITIVOS MÓVEIS

HENRIQUE MANDUCA DA SILVA

COMUNICAÇÃO ENTRE PLATAFORMAS PHP E JAVA ANDROID: DESENVOLVENDO UM SISTEMA PARA AGENDAMENTO ONLINE DE SERVIÇOS

MONOGRAFIA DE ESPECIALIZAÇÃO

FRANCISCO BELTRÃO

2014

HENRIQUE MANDUCA DA SILVA

COMUNICAÇÃO ENTRE PLATAFORMAS PHP E JAVA ANDROID: DESENVOLVENDO UM SISTEMA PARA AGENDAMENTO ONLINE DE SERVIÇOS

Trabalho de Conclusão de Curso apresentada a Coordenação de Licenciatura em Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de Especialista em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis.

Orientador: Prof. Géri Natalino Dutra.

FRANCISCO BELTRÃO

2014

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Francisco Beltrão
Diretoria de Pesquisa e Pós-Graduação
Curso de Especialização em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis

TERMO DE APROVAÇÃO

Dia 22 do mês de novembro de 2014 às: 22 horas, na sala COLIN – anfiteatro do Campus Francisco Beltrão, realizou-se a apresentação pública da monografia pelo estudante Henrique Manduca da Silva, intitulada “Comunicação entre Plataformas PHP e Java Android: Desenvolvendo um sistema para agendamento online de serviços.” Finalizada a apresentação e arguição, a Banca Examinadora declarou aprovada a monografia do estudante, como requisito parcial para obtenção do título de Especialização em Desenvolvimento e Sistemas para Internet e Dispositivo Móveis.

_____
Professor Ademir Roberto Freddo - UTFPR
(Orientador)

_____
Professor Edson dos Santos Cordeiro - UTFPR
(Convidado)

_____
Professor Andrei Carniel - UTFPR
(Convidado)

_____
Professor Dr. Ademir Roberto Freddo - UTFPR
(Coordenação)

A folha de aprovação com as rubricas encontram-se disponíveis na Diretoria de Pesquisa e Pós-Graduação, UTFPR, Francisco Beltrão.

AGRADECIMENTOS

Em primeiro agradeço a minha família, em especial a meus pais Claudio Cesar Manduca da Silva e Solange Aparecida da Silva, por sempre me incentivarem ao estudo e não se conformar com o pouco que sabemos. Agradeço aos meus colegas de sala pelo conhecimento compartilhado através de longas discussões sobre os vários temas da especialização.

Agradeço ao meu orientador Prof. Géri N. Dutra, pelo auxílio na elaboração e conclusão desde documento.

A Secretaria do Curso, pela cooperação.

Enfim, a todos os que por algum motivo contribuíram para a realização deste trabalho.

Resumo

MANDUCA DA SILVA, Henrique. Comunicação entre Plataformas PHP e Java Android: Desenvolvendo um Sistema para Agendamento Online de Serviços. (Curso de Especialização em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis) – Universidade Tecnológica Federal do Paraná. Francisco Beltrão, 2014.

Por meio da presente monografia serão apresentados conceitos para elaboração de uma solução que visa integrar duas tecnologias distintas, sendo estas, a parte web, desenvolvida com linguagem de programação PHP e layout HTML, e a outra sendo um aplicativo Java para dispositivos com sistema operacional Android, afim que estes possam se comunicar e efetuar uma tarefa com satisfatória eficiência. Assim sendo, será desenvolvido um projeto denominado AgendaFix, um produto capaz de se adequar a qualquer empresa que queria automatizar o processo de agendamento e prestação de serviços, visando a integração e comunicação remota de vários dispositivos móveis com um servidor central, disponibilizando ao usuário uma experiência mais rica deste processo.

Palavras-chave: Sistema Online, Android, PHP, HTML, Servidores Web, JSON.

Resumo

MANDUCA DA SILVA, Henrique. Communication between Android Java and PHP Platforms: Developing a System for Online Scheduling Service. (Specialization in Systems Development for Internet and Mobile Devices) – Federal Technological University of Paraná. Francisco Beltrão, 2014.

Through this monograph concepts for developing a solution that aims to integrate two different technologies, these being, the web part developed with the PHP programming language and HTML for the layout, and the other being a Java application for devices with Android operating system will be presented, in order that they can communicate and carry out a task with satisfactory efficiency. Therefore, it will be developed a project called AgendaFix, a product able to suit any company that wanted to automate the scheduling process and services, to integrate various remote communication and mobile devices with a central server, providing the user a richer experience of the process.

Palavras-chave: Online system, Android, PHP, HTML, Web Servers, JSON.

LISTA DE SIGLAS

ADT	*Android Development Toolkit*
CSS	*Cascading Style Sheets*
DER	*Diagrama de Entidade e Relacionamento*
HTML	*HyperText Markup Language*
HTTP	*HypertText Transfer Protocol*
HTTPS	*HypertText Transfer Protocol Secure*
IDE	*Integrated Development Enviroment*
IP	*Internet Protocol*
JDK	*Java Devlopment Kit*
JMS	*Java Messaging Service*
JSON	*JavaScript Objetct Notation*
PHP	*Preprocessor Hypertext Preprocessor*
REST	*Representational State Transfer*
SDK	*Software Development Kit*
SGBD	*Sistema Gerenciador de Banco de Dados*
SMTP	*Simple Mail Transfer Protocol*
SOAP	*Simple Object Access Protocol*
SQL	*Standart Query Language*
TCP	*Transmission Control Protocol*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
URI	*Uniform Resource Identifier*
XML	*Extensible Markup Language*

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, com uma visão geral do trabalho, os objetivos, a justificativa e a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

Nota-se que a evolução da Internet vem mudando nossas vidas. A cada ano surgem novas tecnologias, serviços e oportunidades provenientes ou feitas para a grande rede de computadores.

Outra constante nesse tema é a informatização de processos, ações e/ou etapas antes feitas por um indivíduo, agora automatizadas, substituindo o trabalho manual, aumentando a eficiência e minimizando erros ocasionais.

Para aquelas empresas que buscam fazer algum investimento e posteriormente conseguir algum retorno, estar atento e saber escolher a melhor solução para determinado problema é essencial para sair na frente e não ter que enfrentar ocasiões inesperadas, podendo levar tudo a perder (SEBRAE, 2013).

Entre empresas prestadoras de serviços, tal qual oficinas mecânicas ou auto elétricas por exemplo, o cenário comercial é o mesmo, clientes mais exigentes, que pesquisam preços e buscam interagir mais com as companhias para ganhar a confiança do mesmo ou apenas garantir a qualidade do serviço que estão adquirindo. Estes são exemplos de estabelecimentos que ainda não apresentam grandes mudanças no processo de agendamento da prestação de seus serviços.

Uma nova maneira de interagir e ao mesmo tempo disponibilizar novas soluções para os clientes é por meio dos celulares, *smartphones* e *tablets*, que a cada geração se tornam mais potentes e flexíveis, juntamente com a atual expansão da produção e distribuição de aplicativos para estas plataformas, de modo que tornou-se fácil construir sistemas personalizados capazes de atender praticamente qualquer demanda por automação, muito devido ao custo benefício que melhora a cada ano, trazendo dispositivos mais potentes por valores reduzidos.

As vendas de smartphones no Brasil durante o segundo trimestre de 2014 somaram 13 milhões de aparelhos comercializados, um crescimento de 22% em relação ao mesmo período de 2013, número recorde, de acordo com estudo da

International Data Corporation (IDC, empresa de consultoria). Entre abril e junho, foram vendidos mais de 100 smartphones por minuto. O valor diz respeito às vendas para o varejo, número que não corresponde à venda direta para o consumidor [...]. (G1, 2014).

1.2 JUSTIFICATIVA

O aumento da demanda de softwares gestores de informação pertinentes a diversas áreas de negócio é cada vez maior, logo as opções e tecnologias disponíveis para suprir essa demanda também se tornou vasta e ampla.

Assim surgiu a ideia de desenvolver uma solução que auxiliasse no processo de agendamento para certos tipos de empresa, que melhore a organização interna da mesma, garantindo confiabilidade e segurança nas informações a um baixo custo e que na mesma proporção consiga chegar a um grande número de clientes, trazendo uma nova experiência ao agendar um serviço, que atualmente se resume a estar presente no local ou através de ligações telefônicas.

Para ambos os casos, o mínimo de organização são anotações feitas planilhas eletrônicas com pouca organização, quando não, feitas em cadernos que estão sujeitos a extravios.

As aplicações aqui elaboradas visam facilitar a utilização sobre o funcionamento de um web service baseado em tecnologias de fácil acesso, rápido desenvolvimento e custo baixo.

1.3 OBJETIVOS

O objetivo geral do trabalho representa a proposta de desenvolvimento fornecendo uma visão simplificada e os objetivos específicos detalham citando a características do produto.

1.3.1 Objetivo Geral

Desenvolver uma aplicação móvel que efetue requisições de agendamento de serviços, permitindo ao usuário escolher entre serviços e datas que lhe interessam, bem como consultar valores de cada serviço.

1.3.2 Objetivo Específico

Seguem os objetivos específicos:

- Realizar uma breve introdução as tecnologias utilizadas no desenvolvimento;
- Demonstrar a interação e estrutura de uma interface web para gerenciamento dos agendamentos;
- Construir um web service que irá receber as requisições dos aplicativos móveis;
- Desenvolver um aplicativo cliente com a tecnologia Android, capaz de se comunicar com o web service.

1.5 ESTRUTURA DO TRABALHO

Este trabalho foi estruturado em capítulos, no qual o primeiro capítulo apresenta o assunto e a ideia inicial, além dos objetivos a alcançar e a justificativa.

No capítulo 2 será mostrado o referencial teórico que fundamentou a proposta do sistema desenvolvido. Este trata de sistemas de agendamento online, utilização de novos serviços oferecidos a usuários finais e sobre as tecnologias de integração cliente móvel e web services.

O capítulo 3 descreve os matérias e métodos utilizados para o desenvolvimento dos aplicativos propostos.

O capítulo 4 destaca-se a modelagem e implementação dos sistemas, bem como é feita comunicação entre eles.

No capítulo 5 é apresentado as considerações finais do trabalho, tal qual as dificuldades encontradas e as sugestões de trabalhos futuros, afim de complementar o projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentado os referenciais teóricos que ajudaram na concepção e posterior elaboração dos sistemas propostos. Apresenta-se inicialmente dois sistemas de agendamento, reservas e consultas online, as tecnologias disponíveis para o desenvolvimento de novas ferramentas, com foco para a tecnologia Android e comunicação com servidores.

2.1 SISTEMAS DE AGENDAMENTO ONLINE

A concorrência entre as empresas sempre foi um gatilho para que estas buscassem novidades para atrair novos clientes. Possuir apenas sistemas informatizados apenas para controle de estoque, tarefas e gerenciamento financeiro é deixar de usufruir de todas as novas soluções e tecnologias disponíveis, que podem facilitar a interação entre empresário e cliente.

Aproveitando esse cenário de integração que a Internet disponibiliza de forma cada vez mais acessível, foi verificado como referência o sistema Cucco, da em empresa de mesmo nome Cucco (2014), possuindo recursos de integração com sites das empresas que o utilizam, disponibilizam uma completa interface para acesso do cliente, permitindo de forma fácil o agendamento de uma determinada tarefa.

Lembrando que este conceito pode ser aplicado a diversos ramos de atividade, bem como salões de beleza, psicólogos, aluguel de equipamentos e oficinas mecânicas que será aqui demonstrado.

Na Figura 1 pode ser visto uma das telas do sistema web Cucco, onde são demostrados a escolha de datas e horários para se efetuar um agendamento.

Figura 01 – Agenda Online Cucco
Fonte: Canal Cucco

Outro sistema verificado foi o SimDoctor, este sistema de agendas tem caráter mais específico, de modo que atende apenas consultórios e clínicas, diferente do sistema proposto, que visa se adequar em qualquer área de prestação de serviço.

Através do SimDoctor é possível que um médico possa gerenciar seus pacientes e visualizar as consultas médicas da semana, também dispõe da possibilidade de se criar um site, por onde os pacientes interessados em agendar uma consulta podem estar visualizando a agenda do médico em questão e posteriormente solicitar um agendamento.

Tanto para o sistema Cucco quanto o sistema SimDoctor, o usuário cliente deve ter a seu dispor um microcomputador e a acessá-lo através do navegador de Internet, ainda que seja possível acessar por outros dispositivos com acesso a Internet porem de forma mais custosa e menos eficiente, esse acesso poderia ser facilitado expandindo o sistema através de aplicativos dedicados para outras plataformas, deixando mais prático a utilização da ferramenta.

A extensão do sistema *web* para outras plataformas é outro diferencial em relação ao sistema proposto, os sistemas Cucco e SimDoctor não apresentam comunicação com dispositivos portáteis como *smartphones* e *tablets* por exemplo.

Na Figura 2 é exibido umas das telas que são visíveis apenas pelos médicos que utilizam o sistema, nela é mostrado a agenda em formato de semana, onde cada dia apresenta uma lista de horários.

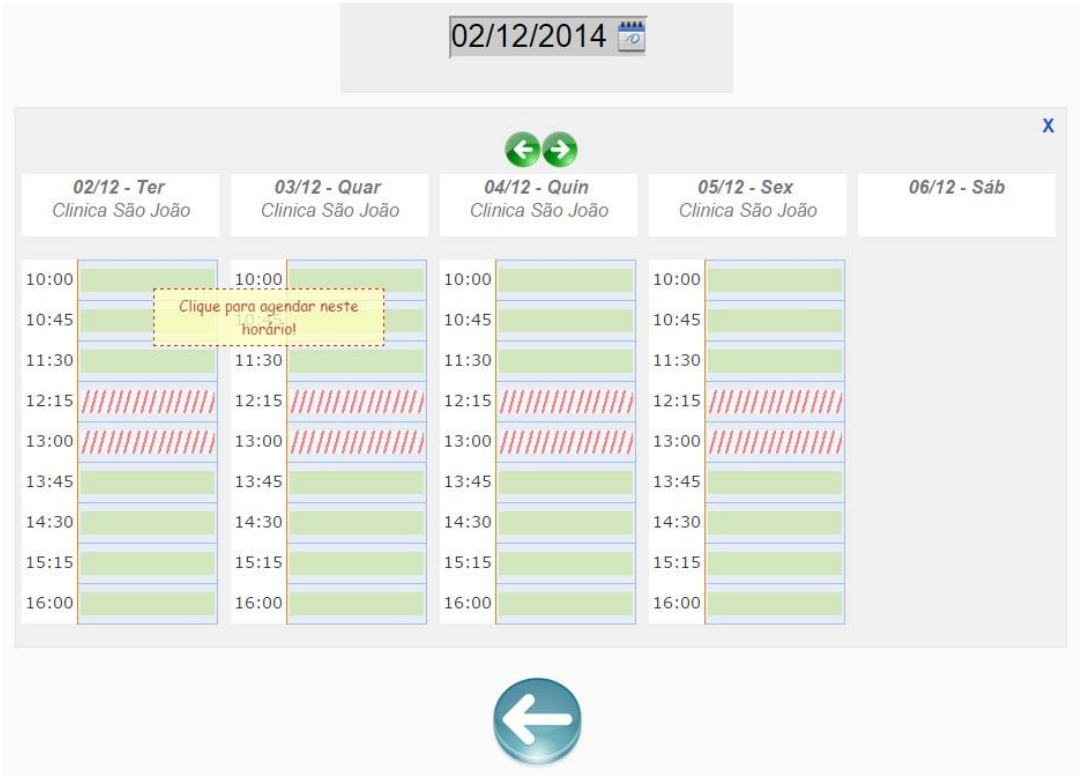

Figura 02 – Agenda Online SimDoctor
Fonte: http://www.simdoctor.com.br/como-funciona/agenda-onlineco

2.2 SISTEMA OPERACIONAL ANDROID

O Android é o sistema operacional móvel da Google Inc., este possui a função e gerenciar todos os recursos de hardware e software de um *smartphone,* sendo possível executar boa parte das tarefas que seriam feitas em um computador de mesa, tais como, navegar na Internet, gerenciar e-mails e acessar redes sociais.

Com o avanço no que diz respeito em desenvolvimento de hardware para pequenos dispositivos como celulares e a chegada dos *tablets*, em 2007, a Google Inc. decidiu investir no desenvolvimento de um sistema operacional para estes aparelhos, foi então que formou e passou a liderar a *Open Handset Alliance (OHA),*

conglomerado de empresas de software, operadoras de telefonia, fabricantes de celulares e chips, cujo objetivo era construir padrões abertos para dispositivos móveis.

Baseado em código Linux, atualmente o Android possui código aberto, possibilitando que vários desenvolvedores participem do projeto, também facilita a distribuição do programa para diferentes hardwares, já que as fabricantes podem altera-los conforme necessário, “o que chama a atenção é o grande impacto e a adoção dessa plataforma nos smartphones e nos tablets hoje. Em 2009, por exemplo, menos de 1% dos smartphones possuía suporte ao Android; hoje, em 2013, esse número ultrapassa 60%.” (OGLIARI; BRITO, 2013, p.5).

Em novembro de 2008 chegou ao mercado o HTC G1, mostrado na Figura 3, primeiro dispositivo com sistema Android.

Figura 03 – HTC G1
Fonte: http://mobiles.sulekha.com/htc_g1_mobile.htm

Mas somente um bom sistema operacional não consegue fornecer todos os recursos para o desenvolvimento de grandes aplicações, por vezes se faz

necessário ir além do dispositivo e fornecer meios para que este se conecte a outros programas e soluções.

2.3 WEB SERVICES

Web services são softwares que ficam hospedados em máquinas conectadas a Internet com o intuito de enviar e receber mensagens contendo dados de qualquer gênero, tratar esses dados conforme necessário e dar um retorno a quem iniciou a requisição.

Cada aplicação pode ter sua própria estrutura de dados que é traduzida para uma linguagem universal, como o formato XML ou JSON, assim disponibilizam um serviço, com o intuito de funcionar constantemente.

Desta forma é possível que aplicações diferentes interajam entre si, não importando a linguagem que ela foi desenvolvida e tornando compatíveis sistemas de plataformas diferentes.

Para que as informações trafeguem corretamente na Internet é necessário a utilização de protocolos de comunicação, o mais conhecido e comentado é o protocolo IP (*Internet Protocol*), através dele que é possível que os pacotes de dados percorram a rota correta entre uma máquina e outra, ou ainda, entre vários roteadores (MENDES, 2007).

O endereço IP é composto por duas partes: primeira identifica a rede em que o equipamento está conectado, enquanto a segunda identifica o próprio equipamento na rede. O IP trabalha em conjunto com o TCP (*Transmission Control Protocol*) formando o modelo de referência TCP/IP, padrão de comunicação adotado pela Internet (MENDES, 2007).

O protocolo TCP garante que os dados encapsulados pelo protocolo IP sejam entregues com eficiência, ou seja, ele permite que as duas máquinas comunicantes possam controlar o estado da conexão, tal como o início e o termino desta conexão (MENDES, 2007).

Outra camada de protocolo e o HTTP (*HyperText Transfer Protocol*), este é destinado a transferência de ficheiros pela rede, tornando possível por exemplo que um navegador de Internet possa exibir um site, cujo conteúdo está salvo em uma

pasta acessível a rede, está por sua vez ligada entre o cliente e o servidor por uma cadeia de caracteres denominada URL ou endereço do site (MENDES, 2007).

Web services podem trabalhar basicamente de duas formas para transmitir seus dados e informações, por meio de texto puro ou estruturado como Json, ou ainda por meio de arquivos no formato XML, deste modo pode caracteriza-se respectivamente as arquiteturas REST (*Representational State Transfer*) e SOAP (*Simple Object Access Protocol*) (STACKOVERFLOW.COM, 2014).

REST é uma especificação criada no ano de 2000 por Roy Fielding em uma dissertação de pós-doutorado. Trata-se de um estilo composto, derivado de outros estilos de arquitetura baseado em rede, de modo que não se trata de uma tecnologia ou algum protocolo, mas sim uma maneira de organizar e utilizar recursos que já existia na *web* (FIELDING, 2000).

O REST usa o protocolo HTTP para troca de mensagens, assim utiliza além dos métodos clássicos como POST e GET, métodos menos comum como o PUT e o DELETE, que faz uma analogia com CRUD (Create, Read, Update e Delete) para as quatro operações básicas utilizadas em bancos de dados. Os métodos de acesso do protocolo são os que dão a semântica ao REST, o acesso assim como a identificação do serviço por sua vez é dado por uma URI (Uniform Resource Identifier) (FIELDING, 2000).

O SOAP surgiu da ideia de um mecanismo de RPC (Remote Procedure Call), originalmente proposto por Dave Winer em 1998, o SOAP foi desenvolvido e descrito pela IBM, Lotus Development Corporation, Microsoft, Development-Mentor e Userland (FREIRE, 2002).

SOAP é um protocolo baseado em XML que permite que aplicativos se comuniquem pela Internet de forma estruturada, utilizando documentos XML encapsulados chamados de SOAP. Portanto, SOAP é independente tanto de plataforma como de software e pode ser implementado em qualquer linguagem de programação, suportando métodos de transporte genéricos, indo desde HTTP, HTTPS (*HypertText Transfer Protocol Secure*), SMTP (*Simple Mail Transfer Protocol*) a JMS (*Java Messaging Service*) (DEITEL, 2003).

3 MATERIAIS E MÉTODOS

Neste capítulo serão apresentados os materiais e métodos que foram utilizados na realização deste trabalho. Os materiais consistem em destacar as tecnologias estudadas, tal como ferramentas de modelagem, posteriormente as linguagens de programação e a implementação do sistema. O método contém as etapas e procedimentos seguidos para o desenvolvimento da aplicação.

3.1 MATERIAIS

As soluções utilizadas para realização da análise, estruturação do projeto e desenvolvimento da aplicação foram: Astah Community 6.9.0 para modelagem UML dos sistemas cliente e servidor, banco de dados MySql para persistência dos dados no servidor modelado com a ferramenta MySql Workbench 6.1, linguagem de marcação HTML, linguagem de programação PHP, Eclipse Juno integrado com Android Development Tools e notação JSON.

3.1.1. Banco de dados MySQL

O MySQL é um Sistema Gerenciador de Banco de Dados (*SGBD*) com versões gratuitas, para uso comercial ou privado, através da licença pública GPL, é um dos mais rápidos programas para servidores de SQL (*Structured Query Language*), uma solução robusta para qualquer tipo de aplicação, muito utilizada em aplicações online, como sites e programas web, suporta várias conexões simultâneas, trabalha com sistema de gerenciamento de dados relacional, ou seja, armazena seus dados em tabelas separadas, o que garante maior velocidade e flexibilidade nas buscas de dados dentro dos arquivos do banco (WELLING; THOMSON, 2005).

Dentre as funcionalidades do SGBD MySQL destacam-se:

- Multiplataforma, suportando: Windows, Linux e Unix.
- Sofisticado sistema de senhas.
- Suporte a API’s de diferentes linguagens: PHP, Java, Pynthon, entre outros.

- Cliente conectado através de TCP/IP.
- Suporte a procedimentos armazenados.
- Suporte a gatilhos.

O MySQL foi escolhido por ser um banco de dados multiplataforma, moderno, de desempenho equivalente a outros SGBD's pagos e alta distribuição entre os desenvolvedores de softwares para Internet. Usuários deste banco de dados podem encontrar suporte e conteúdo para desenvolvimento a partir da página do MySql (http://dev.mysql.com/), conforme mostra Figura 4.

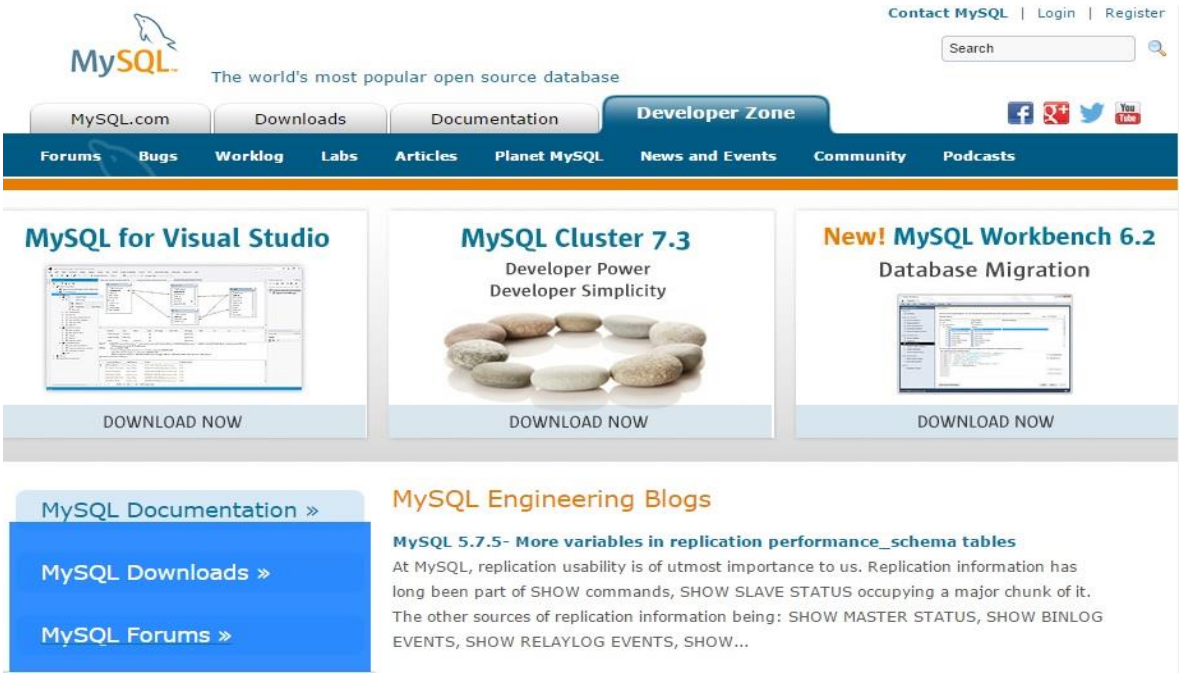

Figura 04 – Página do MySQL voltado a suporte a desenvolvedores
Fonte: Desenvolvido pelo autor (2014)

3.1.2. MySQL Workbench

Trata-se de uma ferramenta que oferece interface amigável para criação e manipulação de bancos de dados para o SGBD MySQL, a partir dele é possível conectar em instâncias do serviço MySQL na própria máquina do desenvolvedor ou ainda em ambientes online.

Após conectado em uma instância é possível executar operações cruciais como anexar novas bases, excluir e fazer *backups,* criar diagramas, ou representações gráficas de como será a estrutura banco.

Esse processo agiliza o desenvolvimento pois trabalha-se com objetos que se relacionam visualmente, como as tabelas que irão receber as informações oriundas de programas, tipos de campos das tabelas, procedimentos armazenados, funções, e deixamos de lado a escrita de longos documentos contendo scripts de SQL.

A versão utilizada foi a 6.1, e foi escolhido por ser dos mesmos desenvolvedores do SGBD MySQL, facilitando a integração entre as ferramentas. A Figura 5 mostra a tela principal do Workbench, exibindo a conexão local disponível, e alguns diagramas que podem ser abertos e editados. O programa pode ser obtido no site: http://www.mysql.com/products/workbench/.

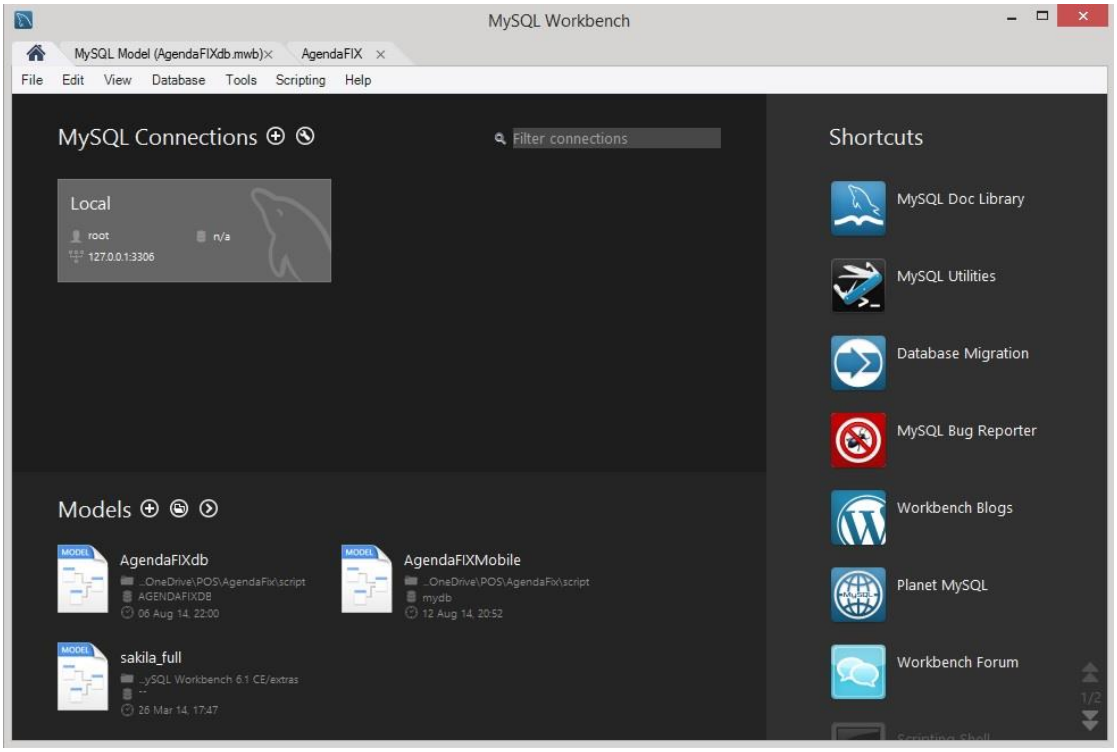

Figura 05 – Tela principal do MySQL Workbench
Fonte: Desenvolvido pelo autor (2014)

3.1.3. Astah Community

O Astah Community, disponível em http://jude.change-vision.com/jude-web/product/community.html é um software que permite entre outras funcionalidades, criar modelos de UML, que por sua vez é uma linguagem para modelagem de projetos, capaz de dar uma visão total dos objetos que compõem o

mesmo e a comunicação entre eles. Este tipo de ferramenta auxilia nas atividades de concepção de um projeto de software, tal como análise de requisitos, diagramas de caso de uso e interação de objetos e processos.

A Figura 6 exibe a tela principal do Astah, exemplificando a elaboração de um caso de uso do sistema proposto.

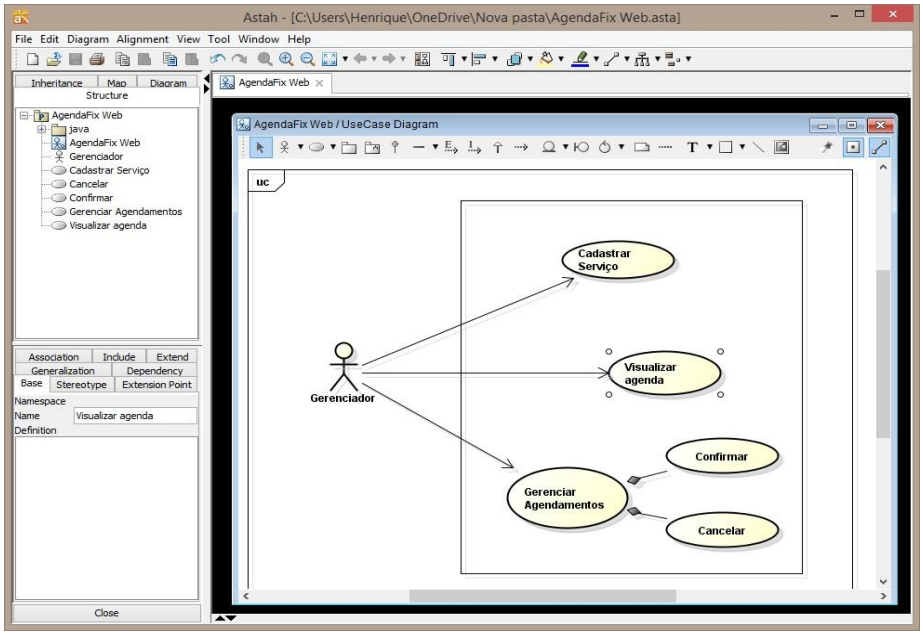

Figura 06 – Tela principal do Astah Community
Fonte: Desenvolvido pelo autor (2014)

3.1.4. HTML

Para a criação de um site que ofereça grande interação com o usuário se faz necessário a integração de diversas tecnologias, onde cada uma deve atender determinada função. Diferente de um software instalado junto aos computadores convencionais, por exemplo, onde que com apenas uma linguagem de programação dispõe de recursos e soluções suficientes para que o programa seja construído e disponibilizado com apenas um arquivo executável (MACORATTI.NET, 2014).

Para aplicações de Internet o programador deverá ter um conhecimento mais diversificado, noções sobre configurações de servidores, sistemas de base de dados, entendimento sobre alguma linhagem de *scripting* como o Java Scrip, protocolos como HTTP, sintaxe de HTML (*Hyper Text Markup Language*) e CSS (*Cascading Style Sheets*) são de suma importância, além de entender as diferentes camadas de programação que compõem uma aplicação web, sendo estas divididas

em linguagens de apresentação, programação do lado cliente e programação do lado servidor.

A Figura 7 mostra algumas das linguagens de estrutura e programação para Internet mais utilizadas e conhecidas atualmente.

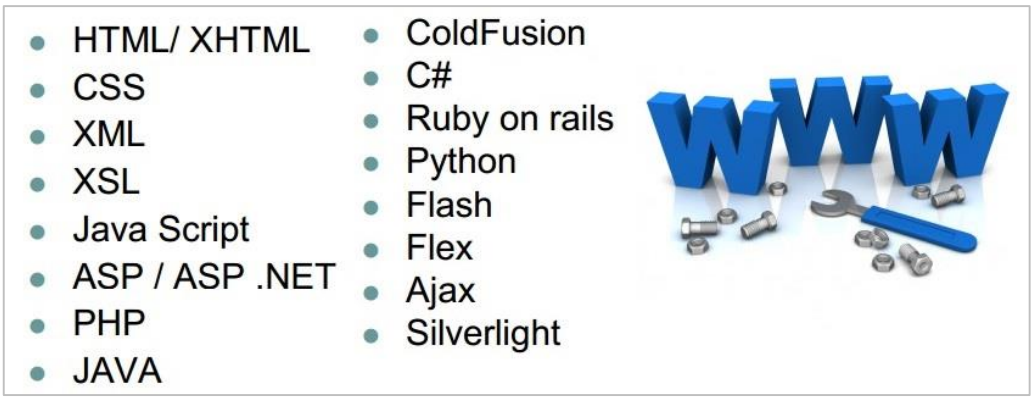

Figura 07 – Linguagens de estrutura e programação para Internet
Fonte: Desenvolvido pelo autor (2014)

O HTML vem a ser um arquivo cuja sua extensão é *.html*, que pode ser alterado por qualquer editor de texto. Em seu conteúdo é possível encontrar textos e códigos específicos dessa estrutura denominados *Tags*, esses marcadores serão interpretados para a construção de uma página web e posteriormente apresentado ao usuário pelo navegador de Internet.

Um arquivo HTML pode conter além de textos e parágrafos, referências para imagens, sons, vídeos e até possibilitar que outras linguagens façam animações dentro da página web, possibilita que seja incorporado ligações ou *links* para outras páginas formando assim documentos com o conceito de hipertexto.

O HTML se torna assim, um elemento indispensável para a construção de uma página web de qualidade.

3.1.5. PHP

O PHP é uma linguagem de programação com código-fonte aberto, implementada e executada do lado servidor da aplicação, ficou altamente difundida entre os desenvolvedores web devido ao fato de possuir sintaxe similar a C, C++ e Perl, possui desempenho satisfatório mesmo em servidores de pequeno porte e de baixo custo.

Também possui facilidade de interagir com páginas HTML por exemplo, assim permite que estas páginas que devido a sua linguagem estruturada são estáticas, tornem-se mais dinâmicas e agradáveis ao usuário, isso porque elas são criadas ou alteradas em tempo de execução pelo código PHP.

Alguns exemplos de páginas dinâmicas são os sistemas de buscas online Google (www.google.com.br) ou Yahoo (www.yahoo.com.br).

Outro aspecto que deve ser levando em conta são as bibliotecas nativas da linguagem, devido ao fato de o PHP ter sido criado para Internet possui muitas tarefas e/ou implementações relacionadas ao tema, inclusive com banco de dados.

Segundo Welling e Thomson (2005, p.28):

> "O PHP tem conexões nativas disponíveis para muitos sistemas de banco de dados. Além do MySQL, você pode conectar-se diretamente a banco de dados PostgreSQL, Oracle, dbm, FilePro, HyperWave, Informix, InterBase e Sybase, entre outros. O PHP 5 também tem uma interface integrada SQL para um arquivo simples, chamada SQLite. Utilizando o Open DataBase Connectivity Standard (ODBC), você pode conectar-se a qualquer banco de dados que forneça um driver ODBC. Isso inclui produtos da Microsoft e muitos outros."

Foi escolhido o PHP como linguagem de desenvolvimento, por ser gratuito, com sintaxe de fácil aprendizagem como mostra o pequeno exemplo da Figura 8, também por possuir integração simplificada com o bando de dados MySQL. Os códigos PHP apresentados nos próximos capítulos foram desenvolvidos com a versão 5 da linguagem, mais informações e documentação podem ser acessados pelo endereço http://php.net/.

```
<html>
 <head>
  <title>PHP Teste</title>
 </head>
 <body>
 <?php echo "<p>Olá Mundo</p>"; ?>
 </body>
</html>
```

Figura 08 – Exemplo de código PHP dentro de uma página HTML
Fonte: Site php.net

3.1.6. Servidores

Para se implementar um web service, além da programação do software, é necessário dispor ou de um computador, solução exclusiva para redes locais, ou a contratação de um serviço online, que possibilitaria o acesso de qualquer lugar via Internet. Tendo uma destas opções devidamente configuradas obtém-se suporte para que à aplicação trabalhe dentro de um ambiente dedicado e controlado pelo desenvolvedor.

Aplicações desenvolvidas em PHP e sistema operacional Windows, contão com uma variedade de pacotes de instalação que visam facilitar esta etapa, sendo eles o pacote WAMP (*Windows Apache MySQL PHP*), XAMPP (*Apache MySQL PHP PERL para qualquer Sistema Operacional)* e EasyPHP. Todos esses pacotes podem ser instalados em forma de serviços no sistema operacional, permitindo que o software desenvolvido fique ativo e execute seus métodos a qualquer momento desde que haja conexão com a rede específica de trabalho.

Para implementação e teste do projeto AgendaFix Web, foi utilizado o pacote XAMPP, por ser melhor difundido entre os desenvolvedores e por ser fácil encontrar documentação que auxilie no desenvolvimento, a versão utilizada é a 1.8.6, posterior ao desenvolvimento e já em processo de finalização, foi contratado um servidor online para que a aplicação trabalha-se complemente fora de uma rede local tornando-se acessível de qualquer dispositivo conectado à Internet.

3.1.7. Plataforma Eclipse

O Eclipse é uma plataforma de desenvolvimento IDE (*Integrated Development Enviroment*) baseada em Java, possui código-fonte aberto o que possibilita a alteração de sua estrutura, adicionado ou removendo recursos dando origem a novas IDEs derivadas, mais completas e com focos de desenvolvimento diferenciados.

O IDE mais utilizado no desenvolvimento de aplicativos Java é o Eclipse, que é de código aberto [...]. (ANDROIDZ, 2014)

O motivo da utilização do Eclipse é por ser um software livre, sem custos, garante um desenvolvimento estável, também recebe atualizações periódicas garantindo a qualidade das aplicações desenvolvidas nele.

Atualmente faz parte do kit de desenvolvido de aplicativos recomendado para desenvolvedores Android possuindo uma versão dedicada a este, sendo esta a versão utilizada para a construção do aplicativo AgendaFix denominada Eclipse Juno 4.2.1, disponível na página https://developer.android.com/sdk/index.html.

A Figura 9 ilustra o ambiente Eclipse Juno para desenvolvimento Android.

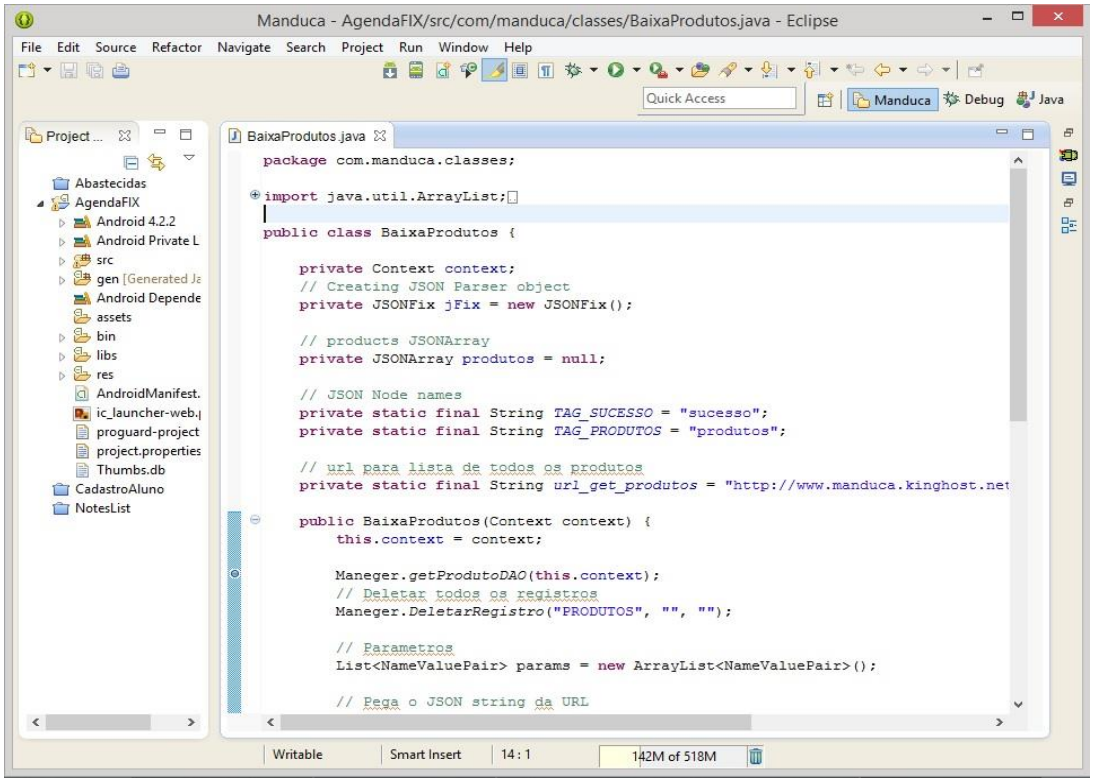

Figura 09 – Ambiente Eclipse Juno
Fonte: Desenvolvido pelo autor (2014)

3.1.8. Android Development toolkit e SDK

Tanto o ADT (*Android Development Toolkit*) como o SDK (*Software Developers Kit*) para Android constituem-se em pacotes de programas e soluções menores voltados ao desenvolvimento de aplicações Android.

Um SDK, em geral, provê os recursos necessários para desenvolvedores criarem softwares que rodem em uma plataforma específica, no caso o Android.

Várias empresas distribuem suas próprias SDK, assim como a Apple para desenvolvimento do IPhone e IPad, e a Microsoft, que também liberou uma SDK para criação de aplicativos do Windows Phone, logo vem a ser a base para o desenvolvimento, sendo que sem esse pacote devidamente configurado na máquina

junto a IDE de programação não há a possibilidade de iniciar a criação de um novo projeto de aplicativo.

A ADT tem a forma de um conjunto de *plugins,* específico para a interface de desenvolvido Elipse citada anteriormente. A versão Juno do Eclipse já vem com uma versão pré-instalada do pacote, a ADT contém diversos componentes, ferramentas, bibliotecas, comandos entre outras particularidades desta tecnologia.

Assim os desenvolvedores podem dedicar mais tempo a suas aplicações.

3.1.9. JSON

O JSON (*JavaScript Object Notation*) é uma maneira de se escrever estruturas de dados em JavaScript, com o objetivo de facilitar o transporte desses dados entre diferentes aplicações e sistemas operacionais. Nativo do Javascript, está disponível na linguagem a bastante tempo e veem ganhando destaque entre os desenvolvedores, por ser extremamente leve e simples de se analisar visualmente.

Não é obrigatório o uso do Javascript para a utilização do JSON, basta apenas que a linguagem utilizada ofereça bibliotecas que deem suporte a semântica dos objetos JSON.

A estrutura do JSON é escrita em duas partes, a primeira recebe o nome do elemento a identificar, também conhecido como atributo, na segunda parte da estrutura é informado o valor desse atributo, ou seja, o conteúdo que se deseja transmitir entre as aplicações, cada objeto JSON pode conter um ou vários desses pares formados por atributos e valores.

A Figura 10 apresenta uma estrutura JSON com dois pares de dados, note que a interpretação visual é altamente intuitiva, contendo dois atributos identificadores e seus respectivos valores.

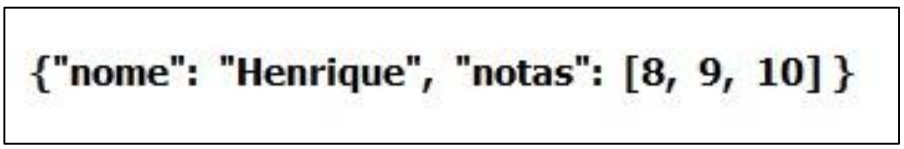

Figura 10 – JSON com dois pares de identificadores
Fonte: Desenvolvido pelo autor (2014)

A linguagem de programação Java que será utilizada no aplicativo Android e o PHP para a programação do servidor e aplicação web oferecem bibliotecas de desenvolvimento para JSON tornando o trabalho com esse formato de comunicação muito mais prático e fácil.

3.2 MÉTODOS

Para o desenvolvimento dos aplicativos propostos foram seguidas as seguintes etapas:

1) Análise de negócio: Etapa de estudos, afim de elaborar uma maneira prática para se fazer agendamentos de serviços, algo que se torne mais atrativo e interessante para o adquirente do serviço e melhore a organização da empresa prestadora. Desta forma permitindo levantar os requisitos funcionais e não funcionais para o desenvolvimento dos aplicativos, bem com a comunicação entre si.

2) Modelagem de sistema: Foi identificada a necessidade de um servidor de banco de dados que possa ser acessado de qualquer lugar via Internet. Para o controle das requisições foi proposto um sistema web implementado junto ao servidor de dados, possibilitando que este seja acessado sem a necessidade de programas adicionais. Na outra ponta, encontra-se elaboração do aplicativo móvel para *tablets* ou *smartphones*, que por sua vez envia as requisições de agendamento ao servidor central já citado. Com base nesta estrutura foi possível visualizar os requisitos funcionais e não funcionais para ambos os sistemas, modelagem de banco de dados, confecção de diagramas de caso de uso, diagramas de comunicação entre servidor e cliente. Com estas informações foi possível iniciar o desenvolvimento.

3) Desenvolvimento: Iniciou-se o desenvolvimento do sistema pela elaboração do banco de dados da aplicação AgendaFix Web, e para o aplicativo móvel. Com a finalização das estruturas de dados, deu-se início a criação do AgendaFix Web com a estruturação das páginas e telas do programa, e a parte servidor, que fará a comunicação externa, e conexão com banco de dados. Tendo a parte servidor finalizada, inicia a implementação da parte cliente para Android no Eclipse, respeitando os diagramas de caso de uso e ajustando a interface para que fique amigável ao usuário.

3.2.1. Levantamento de Requisitos - AgendaFix Web

Para esta parte do projeto foram levantados cinco requisitos funcionais, Efetuar *Login* (Quadro 1), Gerenciar Usuários (Quadro 2), Gerenciar Serviços (Quadro 3), Gerenciar Clientes (Quadro 4) e Visualização da Agenda (Quadro 5).
Para cada requisito funcional foi estabelecido os respectivos requisitos não funcionais.

F1 Efetuar *Login*			
Descrição: Após aberto o site do AgendaFIX, deve-se efetuar *login* para iniciar uma seção no sistema.			
Requisitos Não-Funcionais			
Nome	**Restrição**	**Categoria**	**Permanente**
NF1.1 Segurança	Informar credenciais para liberar entrada	Segurança	(X)
NF1.2 Nível de acesso	O usuário deve estar cadastrado previamente.	Usabilidade	(X)

Quadro 1 – Requisito Funcional: Efetuar *Login*
Fonte: **Desenvolvido pelo autor (2014).**

F2 Gerenciar Serviços			
Descrição: Após iniciado seção no sistema, deve-se cadastrar ou editar, se necessário, os serviços disponíveis aos clientes.			
Requisitos Não-Funcionais			
Nome	**Restrição**	**Categoria**	**Permanente**
NF2.1 Acesso	Usuário possuir nível de acesso compatível.	Segurança	(X)
NF2.2 Dados cadastrais	O serviço deve ter uma descrição e preço definidos.	Usabilidade	(X)
NF2.3 Listagem	Todos os serviços devem ser listados	Usabilidade	(X)
NF2.4 Edição	Escolher facilmente um serviço a ser editado	Usabilidade	(X)

Quadro 2 - Requisito Funcional: Gerenciar Serviços
Fonte: Desenvolvido pelo autor (2014).

F3 Gerenciar Usuários			
Descrição: Cadastro, listagem e edição dos usuários do sistema			
Requisitos Não-Funcionais			
Nome	**Restrição**	**Categoria**	**Permanente**
NF3.1 Acesso	Usuário possuir nível de acesso compatível.	Segurança	(X)
NF3.2 Dados cadastrais	O usuário deve ter nome, um *login* e uma senha.	Usabilidade	(X)
NF3.3 Nível	Informar nível conforme função do usuário.	Segurança	(X)
NF1.4 Listagem	Todos os usuários devem ser listados	Usabilidade	(X)

Quadro 3 - Requisito Funcional: Gerenciar Usuários
Fonte: Desenvolvido pelo autor (2014).

F4 Gerenciar Clientes			
Descrição: Listagem dos dados cadastrais clientes do sistema			
Requisitos Não-Funcionais			
Nome	**Restrição**	**Categoria**	**Permanente**
NF4.1 Acesso	Usuário possuir nível de acesso compatível.	Segurança	(X)
NF4.2 Dados cadastrais	Inclusão apenas por dispositivo móvel.	Segurança	(X)
NF4.3 Listagem	Todos os clientes devem ser listados	Usabilidade	(X)

Quadro 4 - Requisito Funcional: Gerenciar Clientes
Fonte Desenvolvido pelo autor (2014).

F5 Visualizar Agenda			
Descrição: Listar todos os itens que compõem um agendamento, identificados por cliente e data.			
Requisitos Não-Funcionais			
Nome	**Restrição**	**Categoria**	**Permanente**
NF5.1 Acesso	Usuário possuir nível de acesso compatível.	Segurança	(X)
NF5.2 Configuração do serviço	O usuário pode alterar o STATUS do serviço.	Usabilidade	(X)

Quadro 5 - Requisito Funcional: Visualizar Agenda
Fonte: Desenvolvido pelo autor (2014).

3.2.2. Levantamento de Requisitos - AgendaFix Móvel

No projeto foram levantados 3 requisitos funcionais, Cadastrar Cliente (Quadro 6), Visualizar Serviços (Quadro 7), Efetuar Agendamento (Quadro 8). Como anteriormente, para cada requisito funcional foram estabelecidos respectivos requisitos não funcionais.

F6 Cadastrar Cliente			
Descrição: Após aberto o aplicativo, verificar o registro do cliente.			
Requisitos Não-Funcionais			
Nome	**Restrição**	**Categoria**	**Permanente**
NF6.1 Configuração de cliente	Não liberar a interface do agendamento sem efetuar registro.	Segurança	(X)
NF6.2 Dados pessoais	O cliente deve informar dados para contato (Nome, telefone e e-mail).	Usabilidade	(X)

Quadro 6 - Requisito Funcional: Cadastrar Cliente
Fonte: Desenvolvido pelo autor (2014).

F7 Visualizar Serviços			
Descrição: Todos os serviços devem estar disponíveis para consulta e atualização.			
Requisitos Não-Funcionais			
Nome	**Restrição**	**Categoria**	**Permanente**
NF7.1 Informação atualizada	Os serviços devem ser atualizados automaticamente	Usabilidade	(X)
NF7.2 Listagem dos serviços	Possibilidade de consulta de disponibilidade e valor de um serviço	Usabilidade	(X)

Quadro 7 - Requisito Funcional: Visualizar Serviços
Fonte: Desenvolvido pelo autor (2014).

F8 Efetuar Agendamento			
Descrição: Os serviços são agendados a partir da escolha da data e itens pertinentes.			
Requisitos Não-Funcionais			
Nome	**Restrição**	**Categoria**	**Permanente**
NF8.1 Configuração do agendamento	Escolha da data pertinente conforme disponibilidade.	Usabilidade	(X)
NF8.2 Configuração dos itens agendados	Listagem para escolha de pelo menos um serviço.	Usabilidade	(X)
NF8.3 Envio do agendamento	Os itens devem ser enviados e ficarão pendentes até a data escolhida.	Segurança	( )

Quadro 8 - Requisito Funcional: Efetuar Agendamento
Fonte: Desenvolvido pelo autor (2014).

3.2.3. O Software Proposto

A solução consiste na elaboração de dois softwares operando em plataformas diferentes uma da outra, mas ainda assim capazes de se comunicar sem dificuldades ou perca de informação.

A aplicação AgendaFix Web foi elaborado para ser um software de gerenciamento e controle dos dados online, podendo assumir o caráter de sistema de retaguarda ou um sistema integrador com algum outro já existente e mais complexo.

Foi desenvolvido com base na estrutura de páginas HTML, juntamente com o PHP para conexão com banco de dados e programação dos métodos implementados no servidor, sendo este o responsável por disponibilizar os dados e receber os agendamentos requisitados pelos aplicativos clientes.

Para correto funcionamento, o software deve estar devidamente configurado em uma máquina servidor com acesso ininterrupto a Internet, ou em algum serviço de hospedagem online, com suporte à linguagem PHP e banco de dados MySQL, como por exemplo as empresas de hospedagem KingHost (www.kinghost.com.br) ou Hostinger (www.hostinger.com.br), desta maneira torna-se possível a comunicação entre servidor e cliente mesmo estando em redes diferentes.

No servidor também ficam armazenados todos os dados provenientes de cadastrados, sendo estes os registos de usuários como *logins*, senhas e nível de acesso, gerenciamento dos serviços disponíveis pela empresa que utiliza o sistema, dados de contato a cliente, e todo o histórico de agendamentos gerados e enviados pelos aplicativos clientes.

Na Figura 11 o diagrama do banco de dados principal do sistema AgendaFix Web, desenvolvido para MySQL, este banco deve ficar instalado ou hospedado na mesma máquina servidor do sistema, com intuito de armazenar todos os dados referentes a cadastros e registros oriundos da comunicação com o aplicativo cliente com a maior facilidade e desempenho possíveis.

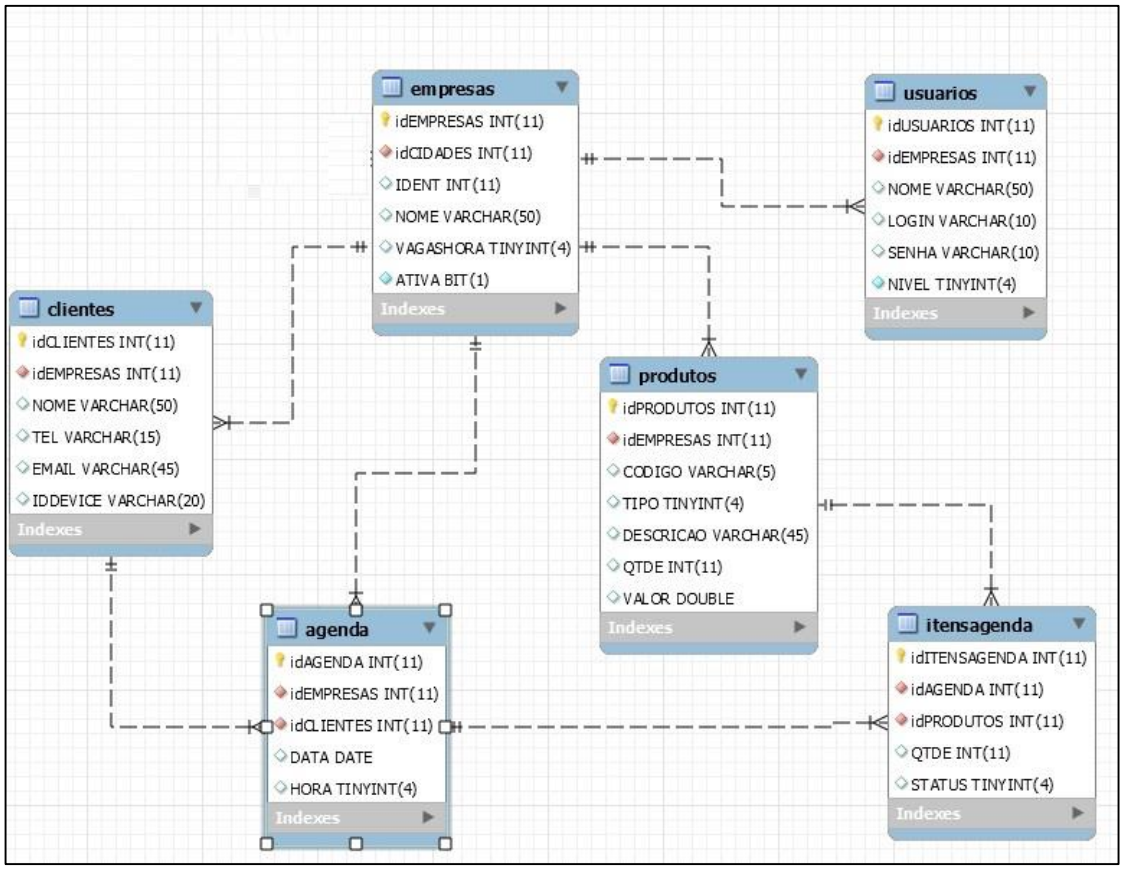

Figura 11 – Diagrama Entidade Relacionamento do Banco de Dados
Fonte: Desenvolvido pelo autor (2014)

O Diagrama Entidade Relacional (DER) da Figura 12 apresenta estrutura simples e mostra as tabelas que foram criadas para o completo funcionamento do sistema e persistência dos dados. A tabela Produtos pode receber facilmente de outro sistema os serviços que irão ser agendados, e para armazenar os agendamentos foram criadas as tabelas Agenda e ItensAgenda que relaciona as informações de itens agendados com o respectivo cliente e data escolhida.

As Figuras 11 e 12 exibem diagramas de casos de usos, que mostram os dois tipos principais de usuários dos sistemas AgendaFix, onde cada um é tratado como ator e cada interação de usuário é mostrada com uma associação aos casos de uso.

Pode ser visto na Figura 12 um retangulo maior, este representa o sistema, já as circunferências representam os casos de uso, que são as funcionalidades presentes no sistema, o ator Usuário (Funcionário) que está cadastrado no sistema e pode acessar todas as funções disponíveis, demonstrando uma parte restrita do programa.

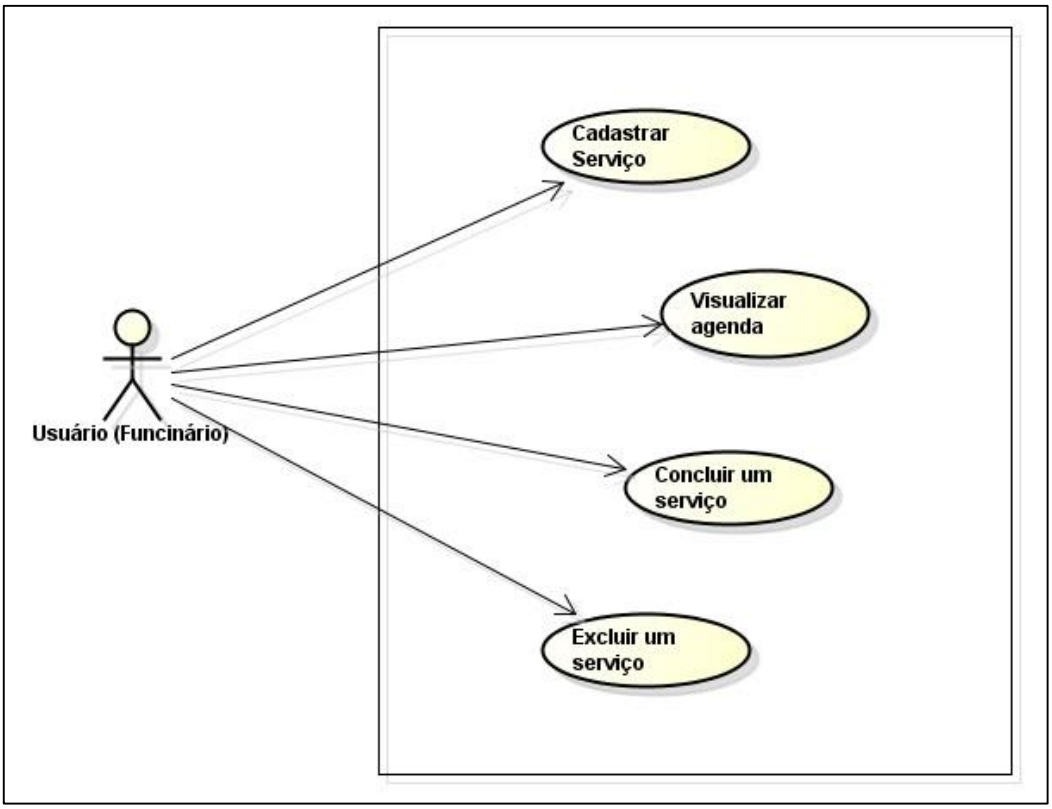

Figura 12 – Caso de Uso AgendaFix Web
Fonte: Desenvolvido pelo autor (2014)

O aplicativo cliente AgendaFix Móvel foi desenvolvido com Eclipse juntamente com o SDK Android para dispositivos que possuem sistema operacional Android, sendo estes smartphones ou tablets. Tanto o PHP na programação do servidor quanto o Java na aplicação cliente possuem facilidades na utilização de protocolos HTTP, suportando utilização de notação JSON para transferência dos dados que será utilizado na integração de ambos os sistemas por qualquer tipo de rede.

Na Figura 13, é exibido outro caso de uso, agora mostrando as funções específicas do AgendaFix Móvel, este mostra o ator Usuário (Cliente) com possibilidade de executar as três funcionalidades principais desta parte do projeto, ele pode efetuar seu registro no AgendaFix Web, posteriormente visualizar todos os serviços que estão disponíveis para ele, e em seguida efetuar um agendamento caso o queira.

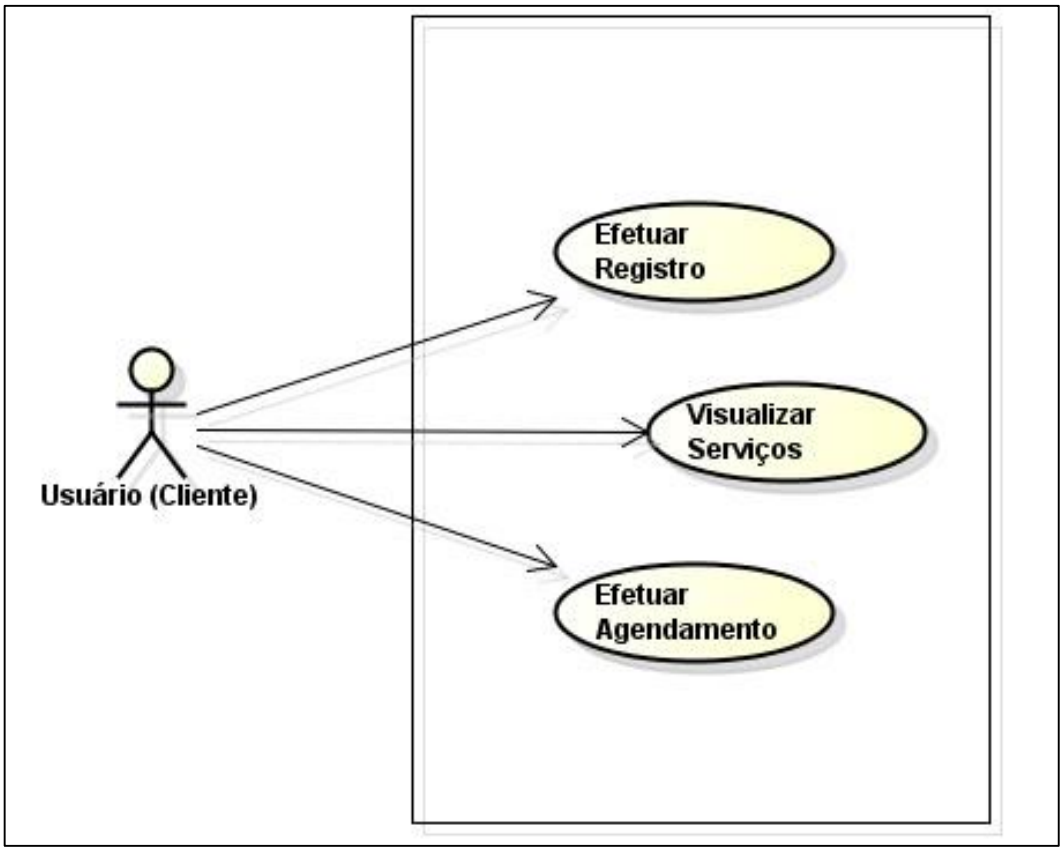

Figura 13 – Caso de Uso AgendaFix Móvel
Fonte: Desenvolvido pelo autor (2014)

O AgendaFix Móvel funcionará como uma extensão, que poderá ser instalado em qualquer dispositivo com Android respeitando as especificações já citadas. Depois de instalado e aberto pela primeira vez, o aplicativo redireciona obrigatoriamente para a tela de registro, assim o usuário cliente pode informar alguns dados para contato e concluir seu cadastro no sistema AgenaFix Web. Após este processo o aparelho irá receber todos os serviços disponíveis e enviados pelo servidor, para tal o aparelho deve dispor de algum tipo de conexão com a Internet podendo ser wi-fi, 3g ou 4g.

Efetuado as configurações iniciais, o cliente já pode requisitar um agendamento de prestação de serviço, chamando esta função, o sistema irá fazer um pedido ao servidor para lhe informar quantos horários por dias estão disponíveis para os próximos 30 dias, após a escolha da data será exibido uma lista com os serviços para que seja informado qual é necessário, podendo ser escolhido mais de um serviço por agendamento.

A Figura 14 apresenta uma visão de como é feita a comunicação entre os elementos que fazem parte do projeto, onde destaca-se o sistema AgendaFix Web rodando e gerenciando os dados a partir do próprio servidor de aplicação.

A aplicação cliente AgendaFix Móvel recupera e envia dados ao servidor por meio da Internet via protocolo HTTP, assim ambos não precisam estar na mesma rede para conseguirem trocar dados.

Juntamente ao servidor de aplicação encotra-se o gerenciamento do banco de dados, este fica encarregado de armazenar todas as informações pertinentes ao projeto, o aplicativo cliente não gera ou sustenta nenhuma informação que não esteja salva primeiramente salva no banco do servidor.

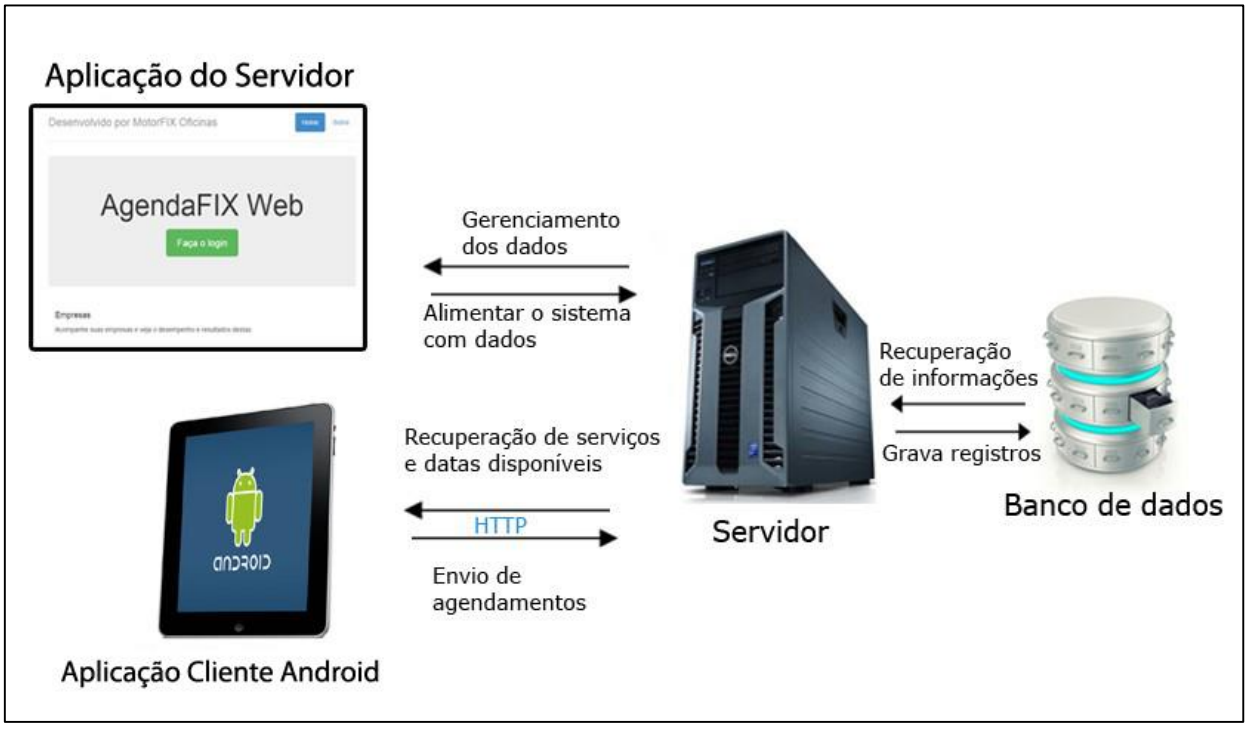

Figura 14 – Visão geral de comunicação: cliente, servidor e banco de dados
Fonte: Desenvolvido pelo autor (2014)

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os sistemas que foram desenvolvidos. Através deste descreve-se as aplicações mostrando suas funcionalidades visuais e lógicas, a comunicação cliente-servidor, bem como exibição de telas e trechos de códigos relevantes a solução proposta.

4.1 IMPLEMENTAÇÃO SISTEMA AGENDAFIX WEB

Como trata-se de uma aplicação para Internet e a mesma deve ficar hospedada em uma máquina servidor, os arquivos que compõem o projeto devem estar devidamente organizados afim de facilitar o processo de implementação e manutenção.

A Figura 15 mostra a organização dos arquivos do projeto dentro do servidor a partir de uma estrutura de pastas, a pasta css contém os arquivos com extensão *.css*, estes arquivos possuem *Tags* que controlam e atribuem características puramente visuais a páginas web, ou seja, são um complemento as *Tags* encontradas nos arquivos *.html*, assim como as pastas fonts e img que apenas fornecem arquivos, como tipos de fontes para os textos exibidos e ilustrações de qualquer gênero. A pasta script guarda um ou mais arquivos *.sql*, possuem a estrutura do banco de dados que a aplicação se conecta em linguagem SQL apenas para fim de consulta.

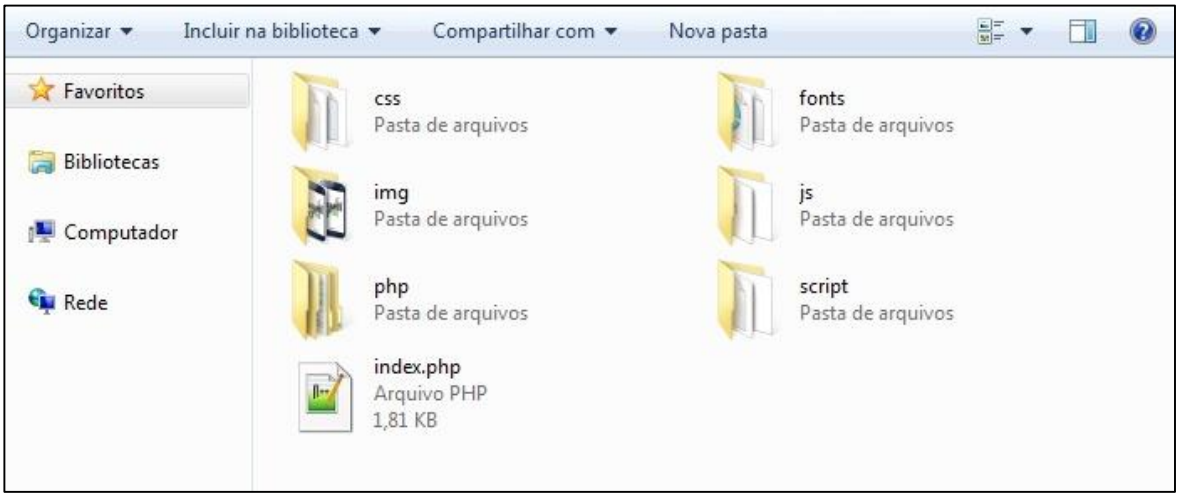

Figura 15 – Visão organizacional dos arquivos no servidor
Fonte: Desenvolvido pelo autor (2014)

A pasta js possui os arquivos com extensão *.js* que contém códigos na linguagem Java Script, de forma simples as funções dessa linguagem fornecem níveis de interatividade maior em páginas HTML. No pasta php encontra-se os a aplicação em si, escrita com linguagem PHP, toda a programação dos métodos de criar, editar, visualizar e excluir dados, assim como as funções dos serviços que estarão disponíveis online, tudo isso em vários arquivos *.php*.

Já o arquivo index.php, toda vez que um navegador acessa um endereço eletrônico este é o primeiro arquivo a ser interpretado e carregado dando inicialização a aplicação *web* ou *site*.

O sistema é iniciado por meio de uma página principal exibindo uma ligeira apresentação comercial do sistema. A Figura 16 mostra a página inicial visível a qualquer tipo de usuário cadastrado previamente ou não. Também apresenta um botão que redireciona para uma área restrita onde somente usuários cadastrados podem acessar via *login* e senha.

Figura 16 – Página inicial do sistema AgendaFix Web
Fonte: Desenvolvido pelo autor (2014)

Após efetuar o *login*, o usuário é redirecionado para o sistema AgendaFix Web, conforme mostra a Figura 17, nesta página do sistema ficam visíveis a barra lateral com os menus Agendas, Usuários, Produtos e Clientes. A página acessada pelo menu Agendas já estará visível. Nela encontrar-se a listagem de todos os serviços pendentes e passíveis a edição.

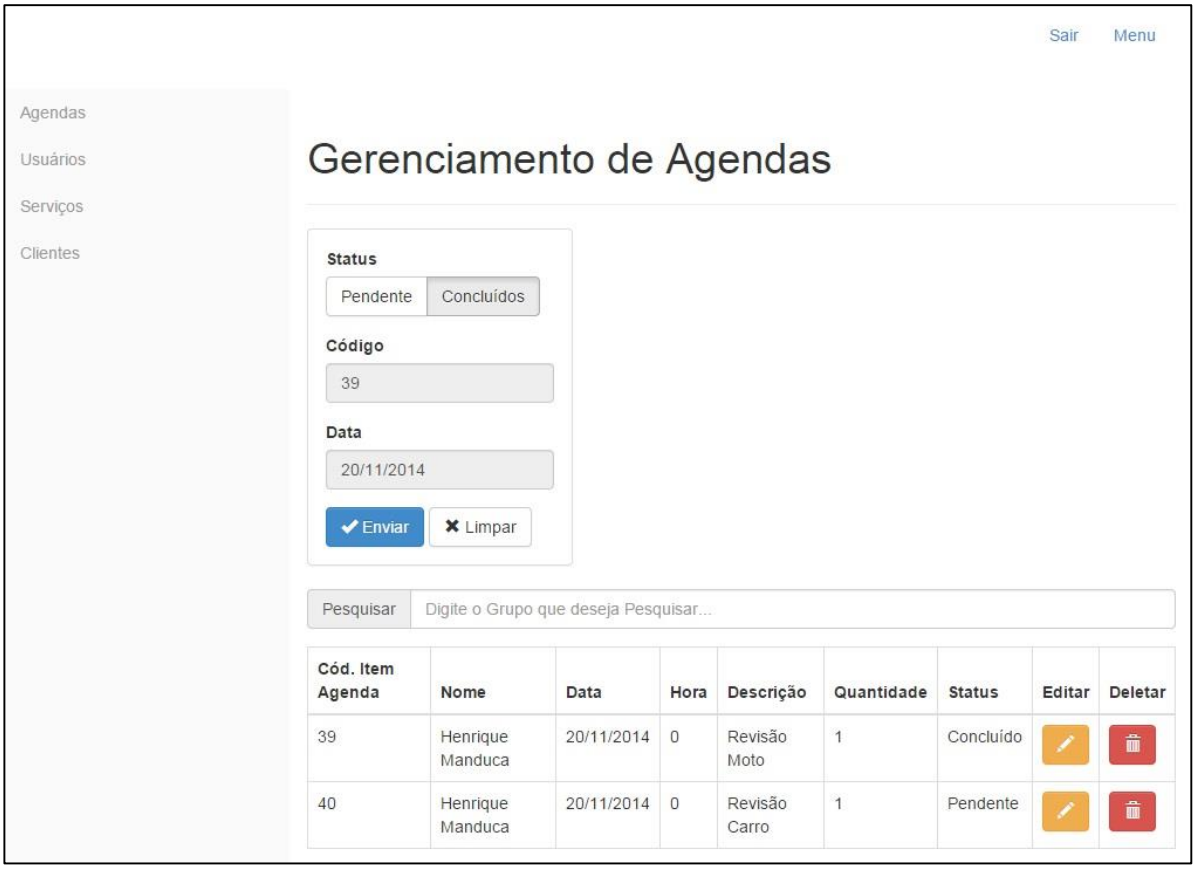

Figura 17 – Página de acesso a agenda
Fonte: Desenvolvido pelo autor (2014)

Para que um agendamento possa ser realizado pela aplicação cliente, primeiro é necessário fazer o cadastro dos serviços disponíveis na empresa prestadora, para isso basta acessar o menu Produtos mostrado na Figura 17.

Na sequência será exibida a página *web* mostrada na Figura 18, contendo o formulário para cadastro bem como a opção de inserir um serviço ou produto, mais a baixo a lista com os serviços já cadastrados.

Esta mesma lógica de cadastro pode e deve ser aplicada às páginas de cadastro de Usuários e Clientes, alterando apenas os dados pertinentes a cada um.

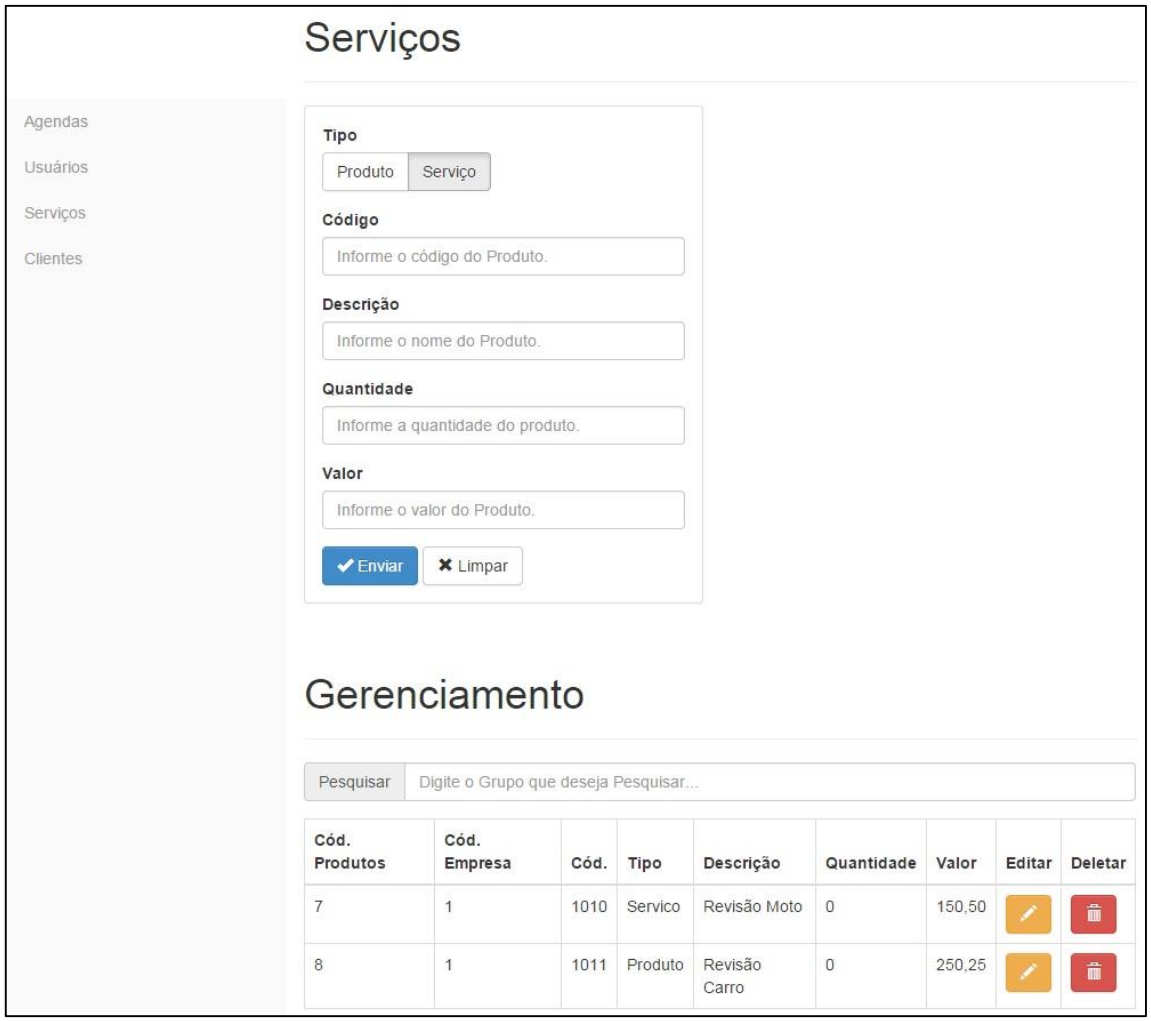

Figura 18 – Página de cadastro de serviços
Fonte: Desenvolvido pelo autor (2014)

Para que as informações sejam armazenadas corretamente no banco de dados do servidor e possam fluir entre as telas para a aplicação cliente é necessário construir uma classe em PHP que faça a conexão e disponibilize métodos de inicialização, finalização da conexão e execução de comandos de SQL.

Toda vez que é necessário inserir ou recuperar informações do banco de dados são chamados os métodos connect(), abrindo a conexão, select_sql() quando se faz uma requisição de informações, execute_sql() quando é necessário injetar dados nas tabelas, o comando disconnect() é chamado para fechar a conexão, as implementações desses métodos podem ser vistas na Listagem 1.

```
<?php
class DataBase
{
	private static $myServer = "caminho para o branco de dados";
	private static $myUser = "usuario";
	private static $myPass = "senha";
	private static $myDB = "nome banco";
	private static $dbhandle;

	private function connect() {
		DataBase::$dbhandle = mysql_connect(DataBase::$myServer,
							DataBase::$myUser,
							 DataBase::$myPass)
			or die("Couldn't connect to SQL Server on DataBase::$myServer");
		$selected = mysql_select_db(DataBase::$myDB, DataBase::$dbhandle)
			or die("Couldn't open database DataBase::$myDB");
	}

	private function disconnect() {
		mysql_close(DataBase::$dbhandle);
	}

	private function query($sql='', $rows=false, $organize=true) {
		$this->connect();
		$result = mysql_query( $sql );
		$numrows = @mysql_num_rows( $result );

		if ($numrows > 0 ) {
			$collection = array();
			while ( $row = mysql_fetch_assoc($result) ) {
				$collection[] = $row;
			}
		} else {
			$this->disconnect();
			return false;
		}

		$this->disconnect();
		return $collection;
	}
```

```
        public function select_sql($sql, $debug=false) {
                if ( $debug ) return var_dump( $sql );
                return $this->query($sql);
        }

        public function execute_sql($sql='', $debug=false) {
                if ( $debug ) return var_dump( $sql );
                $this->connect();
                $result = mysql_query($sql);
                $this->disconnect();
                return $result;
        }
}
```

Listagem 1 – Classe DataBase
Fonte: Desenvolvido pelo autor (2014)

Pela Figura 19 é apresentado uma visão dos arquivos que compõem toda a parte de conexão com banco de dados como o já mostrado DataBase.php, e os arquivos que implementam os serviços que ficam disponíveis pelo servidor, garantindo a comunicação com a aplicação cliente, são estes get_agenda.php, get_produtos.php, set_agenda.php e set_cliente.php.

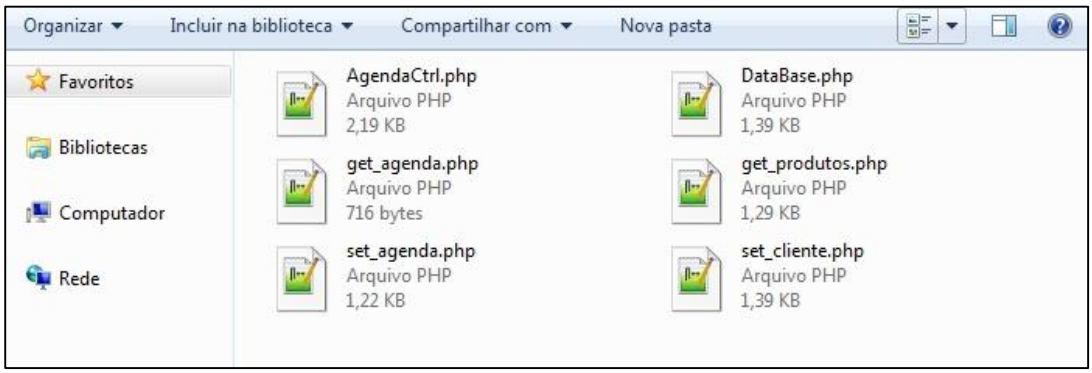

Figura 19 – Visão organizacional dos arquivos de serviço
Fonte: Desenvolvido pelo autor (2014)

A implementação dos arquivos set_cliente.php e set_agenda.php pode ser visto respectivamente na Listagem 2 e Listagem 3, eles são responsáveis por receber e salvar no banco de dados as informações de registro do usuário cliente como nome, e-mail e telefone e os dados de uma agendamento como a data e a lista de serviços, esses arquivo são executados quando o servidor recebe uma

mensagem via HTTP e através da operação POST presente para este protocolo decompõe a mensagem, restando apenas os dados enviados pelo aplicativo cliente.

Complementado a Listagem 3, na Listagem 4 é possível observar a implementa de uma classe auxiliar que está presente no arquivo AgendaCtrl.php, essa classe é responsável por tratar os dados do agendamento e inseri-las no banco de dados.

```
<?php

require_once("DataBase.php");
$response = array();

if (isset($_POST['device']) && isset($_POST['idemp']) &&
            isset($_POST['nome']) && isset($_POST['email']) && isset($_POST['tel'])) {

        $device= $_POST['device'];
        $idemp = $_POST['idemp'];
        $nome  = $_POST['nome'];
        $email = $_POST['email'];
        $tel   = $_POST['tel'];

        $database = new DataBase();
        $result = $database->select_sql("SELECT * FROM CLIENTES WHERE
IDDEVICE='$device'");

        if ($result) {
                $result = $database->execute_sql("UPDATE CLIENTES SET
idEMPRESAS=$idemp, NOME= '$nome', TEL='$tel', EMAIL='$email'");
        } else {
                $result = $database->execute_sql("INSERT INTO CLIENTES(idEMPRESAS,
NOME, TEL, EMAIL, IDDEVICE) VALUES ($idemp, '$nome', '$tel', '$email', '$device')");
        }
}
```

Listagem 2 – Implementação do arquivo set_cliente.php
Fonte: Desenvolvido pelo autor (2014)

```
<?php
require_once("DataBase.php");
require_once("AgendaCtrl.php");

$response = array();

if (isset($_POST['device']) && isset($_POST['data']) && isset($_POST['ser'])) {
        $device= $_POST['device'];
        $data  = $_POST['data'];
                $sercodigos = $_POST['ser'];
                $agenda = new AgendaCtrl();
                $infoCliente  = $agenda->getInfoCliente($device);

                $itens = array();
                $index = 0;

                for ($i = 0; $i < ((strlen($sercodigos)*2)-1); $i++) {
                        if ($sercodigos[$i] == ";") {
                                $itens[$index] = $agenda->getIdProduto( $codigo );
                                ++$indext;
                        } else {
                                $codigo .= $sercodigos[$i];
                        }
                }

                $result = $agenda->salvarAgenda($infoCliente['idCLIENTES'], $data, $itens);
                if ($result) {
                        $response["sucesso"] = 1;
                        $response["mensagem"] = "Agendamento criado.";
                } else {
                        $response["sucesso"] = 0;
                        $response["mensagem"] = "Erro de insert.";
                }

        // echoing JSON response
        echo json_encode($response);
}
```

Listagem 3 – Implementação do arquivo set_agenda.php
Fonte: Desenvolvido pelo autor (2014)

```
<?php
require_once("DataBase.php");

class AgendaCtrl
{
		var $database;
		 public function __construct() {
			$this->database = new DataBase();
		}

		 public function getInfoCliente($device) {
			 $sql = " SELECT idCLIENTES, NOME, EMAIL FROM CLIENTES WHERE
IDDEVICE='$device'";
			 $result = $this->database->select_sql($sql);
			 return $result[0];
		}

		 public function getIdProduto($codigo) {
			$sql = " SELECT idPRODUTOS FROM PRODUTOS WHERE
CODIGO='$codigo'";
			$result = $this->database->select_sql($sql);
			return $result[0];
		}

		 public function salvarAgenda($idCliente, $data, $Itens) {
			$sql = " INSERT INTO `AGENDA`(`idEMPRESAS`, `idCLIENTES`, `DATA`)
VALUES (1, $idCliente, '$data'); ";

			$this->database->execute_sql( $sql );
			$sql = "SELECT MAX(idAGENDA) AS ID FROM AGENDA";
			$qry = $this->database->select_sql( $sql );
			$idAgenda = 0;

			if (!empty($qry)) {
				foreach ($qry as $key => $value) {
					$idAgenda = $qry[ $key ][ 'ID' ];
				}
			}

			foreach ($Itens as $key => $value) {
```

```
                                $sql = " INSERT INTO ITENSAGENDA(idAGENDA, idPRODUTOS,
QTDE, STATUS) VALUES ($idAgenda, $value, 1, 0)";
                                $this->database->execute_sql( $sql );
                        }
                        return true;
                }

                public function getDatasHorarios() {
                        $response = array();
                        $sql = "SELECT idAGENDA, idCLIENTES, DATA, VAGASHORA,
COUNT(idAGENDA) AS SER_D FROM AGENDA INNER JOIN EMPRESAS ON
AGENDA.idEMPRESAS=EMPRESAS.idEMPRESAS WHERE DATA >= curdate() GROUP BY
DATA";
                $result = $this->database->select_sql( $sql );
                if (!empty($result)) {
                        $response["sucesso"] = 1;
                        $response["agenda"] = array();

                        foreach ($result as $key => $value) {
                                $agenda = new stdClass;
                                $agenda->data = $result[ $key ][ 'DATA' ];
                                $date = new DateTime($agenda->data );
                                $agenda->data = $date->format('d/m/Y');
                                $agenda->vagas  = $result[ $key ][ 'VAGASHORA' ];
                                $agenda->ser_d = $result[ $key ][ 'SER_D' ];
                                array_push($response["agenda"], $agenda);
                        }
                } else {
                        $response["sucesso"] = 0;
                }
                return $response;
        }
}
```

Listagem 4 – Implementação do arquivo AgendaCtrl.php
Fonte: Desenvolvido pelo autor (2014)

Também via HTTP, porém agora utilizando o operador GET do protocolo é possível responder ao aplicativo cliente com tabelas de dados completas, a Listagem 5 e Listagem 6 mostram respectivamente a implementação dos arquivos

get_produtos.php que retorna uma lista com os serviços cadastrados previamente no sistema AgendaFix Web, e get_agenda.php que retorna uma lista com as datas e vagas por data para a prestação de serviços.

As informações são organizadas em várias listas e depois convertidas para notação JSON através do método json_encode() e enviadas para dispositivo com a utilização do método echo.

```
<?php

require_once("DataBase.php");
$response = array();

        if (isset($_GET["pAll"])) {
            $database = new DataBase();
            $result = $database->select_sql("SELECT * FROM PRODUTOS");

            if (!empty($result)) {
                    // success
                    $response["sucesso"] = 1;
                    $response["produtos"] = array();

                    foreach ($result as $key => $value) {
                            $prod = new stdClass;
                            $prod->idprod = $result[ $key ][ 'idPRODUTOS' ];
                            $prod->idemp  = $result[ $key ][ 'idEMPRESAS' ];
                            $prod->codigo = $result[ $key ][ 'CODIGO' ];
                            $prod->tipo   = $result[ $key ][ 'TIPO' ];
                            $prod->descr  = $result[ $key ][ 'DESCRICAO' ];
                            $prod->valor  = $result[ $key ][ 'VALOR' ];

                            array_push($response["produtos"], $prod);
                    }
                    // echoing JSON response
                    echo json_encode( $response );
            }
}
```

Listagem 5 – Implementação do arquivo get_produtos.php
Fonte: Desenvolvido pelo autor (2014)

```
<?php

require_once("DataBase.php");
require_once("AgendaCtrl.php");

        if (isset($_GET["pAll"])) {
                $agenda = new AgendaCtrl();
                $response = $agenda->getDatasHorarios();

            if (!empty($response)) {
                 echo json_encode( $response );
            } else {
                $response["sucesso"] = 0;
                $response["mensagem"] = "Agenda não encontrada";

                // echo no users JSON
                echo json_encode($response);
            }
}
```

Listagem 6 – Implementação do arquivo get_agenda.php
Fonte: Desenvolvido pelo autor (2014)

4.2 IMPLEMENTAÇÃO SISTEMA AGENDAFIX PARA ANDROID

Desenvolvido no Eclipse, o aplicativo AgendaFix Móvel vem a ser a parte cliente do projeto, é ele que irá acessar e consumir todos os métodos programados e disponíveis pelo *web service* do AgendaFix Web. Na sequência será apresentado telas do aplicativo, capturadas a partir de um dispositivo que recebia a instalação do aplicativo durante todo o desenvolvido do mesmo, bem como códigos-fonte utilizados para completar a comunicação com o servidor de dados.

Pela Figura 20 é possível observar a estrutura do aplicativo AgendaFix Móvel dentro do Eclipse, as classes de objetos são organizadas em pacotes para melhor organização, outro item de destaque é a pasta *layout* mostrada no lado direito, cada arquivo .xml contido nesta pasta representa uma tela do aplicativo proposto. Os demais itens e pastas presentes são criados por padrão para qualquer projeto Android dentro da IDE Eclipse.

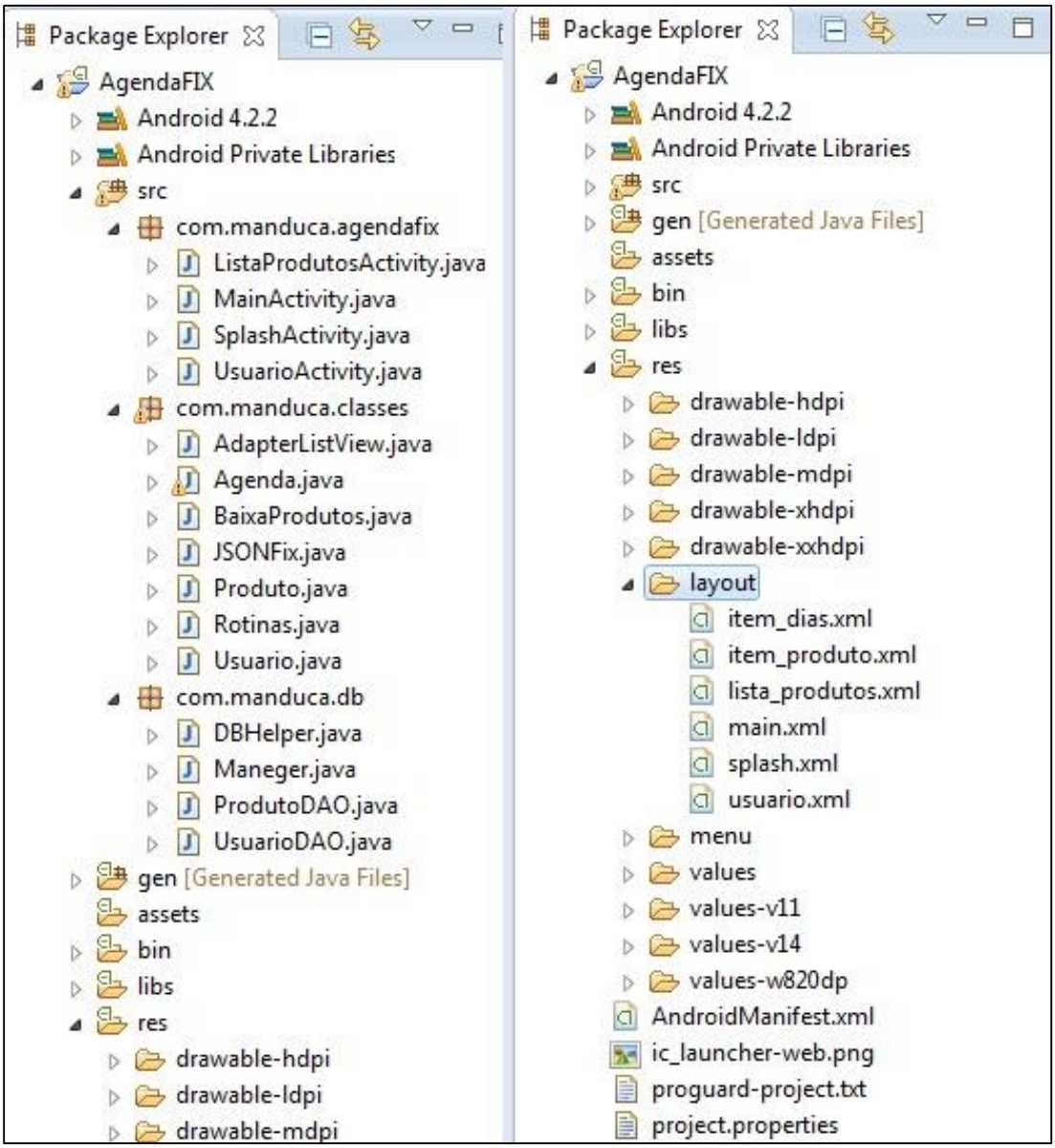

Figura 20 – Organização do projeto AgendaFix Móvel
Fonte: Desenvolvido pelo autor (2014)

A Figura 21 apresenta a tela principal, nele estão contidas todas as funções disponíveis no aplicativo, esta tela é construída a partir do arquivo main.xml (Figura 20), seus métodos estão implementados na classe MainActivity.java, todas as classes que irão receber os métodos e comandos de um arquivo .xml de tela, devem obrigatoriamente estender e/ou herdar da classe Activity.class.

Figura 21 – Tela principal do aplicativo AgendaFix móvel
Fonte: Desenvolvido pelo autor (2014)

Quando o aplicativo é aberto pela primeira vez no dispositivo em que está instalado a tela de registro de usuário é exibida automaticamente, posteriormente ela pode ser acessada utilizando o botão Meu Contato da tela principal. A Figura 22 mostra a tela de registro de usuário que é composta pela classe UsuarioActivity.java e tem sua estrutura salva no arquivo usuário.xml.

Quando o usuário cadastra seus dados utilizando o botão Registrar desta tela, as informações são recebidas pela classe SetUsuario exibida na Listagem 7, que está programada para se comunicar com o arquivo set_cliente.php do *web service*.

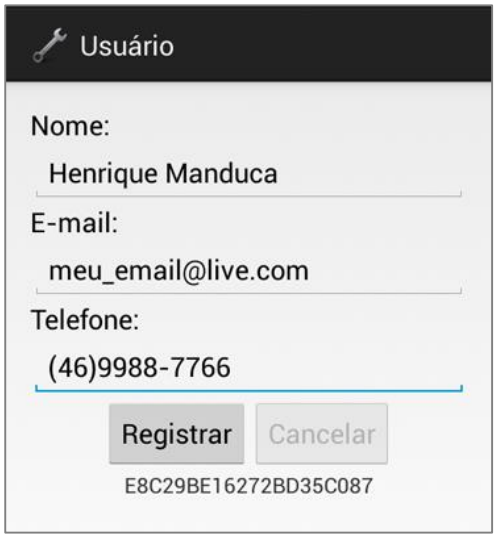

Figura 22 – Tela de registro de cliente por dispositivo
Fonte: Desenvolvido pelo autor (2014)

```
class SetUsuario extends AsyncTask<String, String, String> {

        private static final String TAG_SUCESSO = "sucesso";
        private ProgressDialog pDialog;
        private JSONFix jFix = new JSONFix();
        private static final String url_set_cliente =
"http://meu_servidor/agendafix/php/service/set_cliente.php";

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(UsuarioActivity.this);
            pDialog.setMessage("Registrando cliente...");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }

        protected String doInBackground(String... args) {
            // Building Parameters
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("device",
tvID.getText().toString()));
            params.add(new BasicNameValuePair("idemp", "1"));
            params.add(new BasicNameValuePair("nome",
edtNome.getText().toString()));
            params.add(new BasicNameValuePair("email",
edtEmail.getText().toString()));
            params.add(new BasicNameValuePair("tel",
edtTel.getText().toString()));

            jFix.makeHttpRequest(url_set_cliente, "POST", params);

            return null;
        }

        protected void onPostExecute(String file_url) {
            // dismiss the dialog once done
            pDialog.dismiss();
            onBackPressed();
        }
    }
}
```

Listagem 7 – Implementação da classe setUsuario
Fonte: Desenvolvido pelo autor (2014)

Depois que as informações são organizadas na variável params, que se caracteriza por ser uma lista de pares, onde encontra-se um identificador e o seu valor correspondente, em seguida os dados são enviados para o objeto jFIx da classe JSONFix exibida na Listagem 8.

Este objeto é usado em todas as operações de comunicação com o *web service*, pois ele contém toda a implementação necessária para fazer uma requisição HTTP com o servidor, tanto para enviar dados através de um método

POST e recuperar informações com o método GET, desde que estas estejam em formato JSON.

```
public class JSONFix {

	static InputStream inputStream = null;
	static JSONObject jsonObject = null;
	static String jsonStr = "";

	// construtor
	public JSONFix() {

	}

	// função para pegar json da url
	// por metodo HTTP POST ou GET
	public JSONObject makeHttpRequest(String url, String method,
			List<NameValuePair> params) {

		try {
			// check for request method
			if (method.equals("POST")) {
				// request method is POST
				// defaultHttpClient
				DefaultHttpClient httpClient = new
DefaultHttpClient();

				HttpPost httpPost = new HttpPost(url);
				httpPost.setEntity(new
UrlEncodedFormEntity(params));

				HttpResponse httpResponse =
httpClient.execute(httpPost);
				HttpEntity httpEntity = httpResponse.getEntity();
				inputStream = httpEntity.getContent();

			} else if (method.equals("GET")) {
				// request method is GET
				DefaultHttpClient httpClient = new
DefaultHttpClient();

				if (params.size()>0) {
					String paramString =
URLEncodedUtils.format(params, "utf-8");
					url += "?" + paramString;
				} else {
					url += "?pAll";
				}

				HttpGet httpGet = new HttpGet(url);

				HttpResponse httpResponse =
httpClient.execute(httpGet);
				HttpEntity httpEntity = httpResponse.getEntity();
				inputStream = httpEntity.getContent();

			}

		} catch (UnsupportedEncodingException e) {
```

```
                e.printStackTrace();
            } catch (ClientProtocolException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }

            try {
                BufferedReader reader = new BufferedReader(new
InputStreamReader(
                        inputStream, "UTF-8"), 8);

                StringBuilder strBuilder = new StringBuilder();
                String line = null;

                while ((line = reader.readLine()) != null) {
                    strBuilder.append(line + "\n");
                }

                inputStream.close();
                jsonStr = strBuilder.toString();

            } catch (Exception e) {
                Log.e("Buffer Error",
                        "Erro ao converter resultado. " +
e.toString());
            }

            // Converte a string para JSON object
            try {
                jsonObject = new JSONObject(jsonStr);
            } catch (JSONException e) {
                Log.e("JSON Fix", "Erro ao converter dados. " +
e.toString());
            }

            // retorna o JSON
            return jsonObject;
        }
}
```

Listagem 8 – Implementação da classe JSONFix
Fonte: Desenvolvido pelo autor (2014)

A tela de consulta de serviços mostra os itens que podem ser agendados é mostrado pela Figura 23, pode ser acessada a partir do botão Lista Serviços da tela principal, é composta pelos arquivos lista_produtos.xml que forma a base da lista e o arquivo item_produtos.xml que configura os itens da lista permitindo a exibição de duas linhas de texto por item da lista.

O arquivo ListaProdutosActivity.java fica responsável por implementar as funções e métodos desta tela. Toda vez que o usuário consulta os serviços disponíveis é feito uma atualização dos mesmos.

Figura 23 – Lista de serviços ofertados para consulta de valores
Fonte: Desenvolvido pelo autor (2014)

A Listagem 9 apresenta a implementação da classe BaixaProdutos, que é responsável por fazer a requisição ao servidor, ler a resposta no formato JSON e inserir as informações no banco de dados interno do aplicativo.

```
public class BaixaProdutos {

	private Context context;
	// Creating JSON Parser object
	private JSONFix jFix = new JSONFix();

	// products JSONArray
	private JSONArray produtos = null;

	// JSON Node names
	private static final String TAG_SUCESSO = "sucesso";
	private static final String TAG_PRODUTOS = "produtos";

	// url para lista de todos os produtos
	private static final String url_get_produtos = "http://
meu_servidor/agendafix/php/service/get_produtos.php";

	public BaixaProdutos(Context context) {
		this.context = context;

		Maneger.getProdutoDAO(this.context);
		// Deletar todos os registros
		Maneger.DeletarRegistro("PRODUTOS", "", "");

		// Parametros
		List<NameValuePair> params = new ArrayList<NameValuePair>();

		// Pega o JSON string da URL
		JSONObject json = jFix.makeHttpRequest(url_get_produtos,
"GET", params);

		try {
			// Checking for SUCCESS TAG
			int sucesso = json.getInt(TAG_SUCESSO);

			if (sucesso == 1) {
				// products found
				// Getting Array of Products
```

```
                        produtos = json.getJSONArray(TAG_PRODUTOS);

                        // looping through All Products
                        for (int i = 0; i < produtos.length(); i++) {
                              JSONObject item = produtos.getJSONObject(i);

                              int cod = item.getInt("codigo");
                              String desc = item.getString("descr");
                              Integer tipo = item.getInt("tipo");
                              Double valor = item.getDouble("valor");

                              Maneger.getProdutoDAO(this.context)
                                          .Insert(cod, desc, tipo, valor);
                        }
                  } else {
                        // Sem produtos
                  }
            } catch (JSONException e) {
                  e.printStackTrace();
            }
      } // baixaprodutos
}
```

Listagem 9 – Implementação da classe BaixaProdutos
Fonte: Desenvolvido pelo autor (2014)

Para realizar um agendamento o usuário deve clicar no botão Agendar Serviço da tela principal, na sequência será feito uma requisição ao servidor para ser informado quais datas dos próximos trinta dias possuem horários disponíveis, essas datas serão exibidas através de uma caixa de diálogo, a Figura 24 demonstra esta etapa, onde o usuário pode navegar por uma lista, e escolher uma data apenas clicando em um dos itens desta lista.

Figura 24 – Lista com datas disponíveis para agendamentos
Fonte: Desenvolvido pelo autor (2014)

Após escolhido uma data da lista, é exibida uma nova caixa de diálogo mostrando agora uma lista com os serviços disponíveis para agendamento, a interação nesta última etapa é maior, pois é necessário permitir ao usuário múltiplas escolha de serviços, para tal, e exibido um *checkbox* por item da lista. Após escolhido os serviços o usuário pode concluir o agendamento clicando no botão confirmar, conforme mostra a Figura 25.

Figura 25 – Lista para escolha de serviços
Fonte: Desenvolvido pelo autor (2014)

Em um processo similar ao da Listagem 7, alterando apenas o destino no servidor e os dados o agendamento é enviado e concluído, podendo já ser consultado na página de acesso a agenda do AgendaFix Web.

5 CONCLUSÃO

O trabalho desenvolvido teve como objetivo elaborar e construir duas aplicações em plataformas diferentes e possibilitar a comunicação segura e sem perca de dados entre elas, afim de disponibilizar uma nova maneira de requisitar e efetuar agendamentos, por meio de um aplicativo instalado em um dispositivo móvel os clientes de uma determinada empresa prestadora de serviço podem consultar preços e datas disponíveis para uma futura prestação deste serviço.

Por outro lado, a empresa prestadora consulta os agendamentos enviados pelos seus clientes através de uma aplicação para Internet, registrar novos serviços, gerenciar o que estiver em andamento e verificar dados para contato com os clientes.

Seguindo a ordem dos objetivos apresentados foram descritas as ferramentas que tiveram seus recursos explorados conforme necessário, juntamente com estudo teórico, modelagem e desenvolvimento para disponibilizar duas aplicações que interagem entre si, possibilitando gerenciar e efetuar agendamentos de maneira prática e intuitiva.

Para o desenvolvimento da aplicação de gerenciamento *web* denominada aqui como AgendaFix Web, foram escolhidas as tecnologias HTML, PHP e notação JSON e gerenciador de banco de dados MYSQL para construção, todas as tecnologias utilizadas são livres para desenvolvimento e possuem ótima aceitação entre os desenvolvedores *web*.

A elaboração dos recursos de comunicação disponibilizados pelo servidor, também foram todos desenvolvidos com linguagem de programação PHP, e juntamente disponibilizados com a interface de gerenciamento na máquina servidor.

O uso das ferramentas para desenvolvimento de aplicativos voltado ao sistema operacional Android possibilitaram a conclusão do último objetivo que consistia na elaboração de uma aplicação nomeada AgendaFix Móvel, sendo estas capaz de se conectar a um *web service* online, consumindo todos os recursos disponibilizados por este.

REFERÊNCIAS

MENDES, Douglas Rocha. **Redes de computadores: teoria e prática.** Ed. São Paulo: Novatec, 2007

WELLING, Luke; THOMSON, Laura**. PHP e MySQL Desenvolvimento Web.** 3ª ed. Rio de Janeiro: Vozes Ltda, 2005.

PEREIRA, Lucio Camilo Oliva; DA SILVA, Michel Lourenço. **Android para Desenvolvedores.** Ed. Rio de Janeiro: Brasport, 2009.

OGLIARI, Ricardo da silva; BRITO, Robison Cris. **Android: Do Básico ao Avançado.** Ed. Rio de Janeiro: Ciência Moderna, 2014.

FIELDING, Roy Thomas. **Architetural Styles and Design of Network-based Software Architetures.** Universidade da Califórnia: Dissertação (Doutorado em Filosofia da Computação), 2000.

OGLIARI, Ricardo da silva; BRITO, Robison Cris. **Android: Do Básico ao Avançado.** Ed. Rio de Janeiro: Ciência Moderna, 2003.

DEITEL, H.M. **Internet & World Wide Web – Como Programar**. Ed. Porto Alegre: Bookman, 2003.

FREIRE, H. **Web Services: A Nova Arquitetura da Internet**. Ed. São Paulo Developer's CIO Magazine, 2002.

SEBRAE. Disponível em:
<http://www.sebraesp.com.br/index.php/76-noticias/multissetorial/9274-informatizar-a-empresa-e-uma-questao-de-necessidade>. Acesso em: 25 jul 2014.

CUCCO. Disponível em:
<http://www.cucco.com.br/>. Acesso em: 1 aug. 2014.

SIMDOCTOR. Disponível em:
< http://www.simdoctor.com.br/>. Acesso em: 1 aug. 2014.

G1. Disponível em:
<http://g1.globo.com/tecnologia/noticia/2014/09/venda-de-smartphones-passam-de-13-milhoes-no-brasil-no-2-trimestre.html>. Acesso em: 3 aug. 2014.

ANDROIDZ. Disponível em:
<http://www.androidz.com.br/portal/o-que-e-o-sdk-do-android.html>. Acesso em: 15 aug. 2014.

STACKOVERFLOW.COM. Disponível em:
<http://pt.stackoverflow.com/questions/11183/diferen%C3%A7as-de-tipos-de-web-service-soap-rest-xml>. Acesso em: 5 set. 2014.

CASSIO SANTOS. Disponível em:
<http://cassiosantos.wordpress.com/2009/09/16/criando-um-web-service-basico-usando-php-mysql-xml-e-json>. Acesso em: 18 jun. 2014.

IMASTERS. Disponível em:
<http://imasters.com.br/artigo/4535/php/construindo-web-services-em-php>. Acesso em: 9 jun. 2014.

JSON. Disponível em:
<http://json.org/json-pt.html>. Acesso em: 17 jun. 2014.

XAMPP. Disponível em:
<http://www.baixaki.com.br/download/xampp.htm>. Acesso em: 15 mar. 2014.

MACORATTI.NET. Disponível em:
<http://www.macoratti.net/vbn_dkwb.htm >. Acesso em: 12 jun. 2014.