



CLOUDPOWERSAVE - SISTEMA DE MONITORAMENTO REMOTO DE ENERGIA ELÉTRICA BASEADO NAS TECNOLOGIAS CLOUD E MOBILE

Rodrigo Pinheiro Carvalho

Projeto de Graduação apresentado ao Curso de Computação e Informação da Escola Politécnica da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

Orientador: Sérgio Barbosa Villas-Boas

Rio de Janeiro
Dezembro de 2013

CLOUDPOWERSAVE - SISTEMA DE MONITORAMENTO REMOTO DE
ENERGIA ELÉTRICA BASEADO NAS TECNOLOGIAS CLOUD E MOBILE

Rodrigo Pinheiro Carvalho

PROJETO SUBMETIDO AO CORPO DOCENTE DO CURSO DE
COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE
DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE
ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinadores:

Prof. Sérgio Barbosa Villas-Boas, Ph.D.

Prof. Aloysio de Castro Pinto Pedroza, Dr.

Prof. Antônio Carlos Moreirão de Queiroz, Dr.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2013

Carvalho, Rodrigo Pinheiro

CloudPowerSave - Sistema de Monitoramento Remoto de Energia Elétrica Baseado nas Tecnologias Cloud e Mobile/Rodrigo Pinheiro Carvalho. – Rio de Janeiro: UFRJ/POLI, 2013.

XII, 51 p.: il.; 29,7cm.

Orientador: Sérgio Barbosa Villas-Boas

Projeto (graduação) – UFRJ/ Escola Politécnica/ Curso de Computação e Informação, 2013.

Referências Bibliográficas: p. 50 – 51.

1. Web Services.
 2. Arquitetura Orientada a Serviços.
 3. SOA-MC.
- I. Villas-Boas, Sérgio Barbosa. II. Universidade Federal do Rio de Janeiro, Escola Politécnica/ Curso de Computação e Informação. III. Título.

Agradecimentos

Agradeço à minha família, que sempre me apoiou em todos os momentos importantes. Agradeço à minha namorada Martha e aos amigos Daniel, Marcelo e Pedro, pela motivação e auxílio para realização deste trabalho.

Por fim, agradeço ao povo brasileiro por pagar meu curso superior.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

**CLOUDPOWERSAVE - SISTEMA DE MONITORAMENTO
REMOTO DE ENERGIA ELÉTRICA BASEADO NAS
TECNOLOGIAS CLOUD E MOBILE**

Rodrigo Pinheiro Carvalho

Dezembro/2013

Orientador: Sérgio Barbosa Villas-Boas

Curso: Engenharia de Computação e Informação

Visando o monitoramento remoto de energia elétrica, foi desenvolvido um sistema usando a linguagem de programação Java. Foram construídos um web service, um programa que interage com medidores e um aplicativo para dispositivos móveis com o sistema operacional Android. Monitorar a energia elétrica é altamente relevante para indivíduos e empresas que desejam reduzir seus custos e agir de forma sustentável.

Palavras-Chave: Web Services, Arquitetura Orientada a Serviços, SOA-MC.

Abstract of the Undergraduate Project presented to Poli/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

**CLOUDPOWERSAVE - ELECTRIC ENERGY REMOTE
MONITORING SYSTEM BASED ON CLOUD AND MOBILE
TECHNOLOGIES**

Rodrigo Pinheiro Carvalho

December/2013

Advisor: Sérgio Barbosa Villas-Boas

Course: Computer and Information Engineering

Aiming the remote electric energy monitoring, a system was developed using the Java programming language. A web service, a program that interacts with meters and an application for mobile devices with the Android operating system were constructed. To monitor electric energy is highly relevant for individuals and enterprises who desire to reduce their costs and act in a sustainable way.

Keywords: Web Services, Service-Oriented Architecture, SOA-MC.

Siglas

ADT Android Development Tools

CPU Central Processing Unit

CSV Comma-Separated Values

DAO Data Access Object

DOUA Database Oriented Usecase Authorization

GMT Greenwich Mean Time

GPS Global Positioning System

GPU Graphical Processing Unit

GUI Graphical User Interface

HDMI High-Definition Multimedia Interface

HTTP HyperText Transfer Protocol

IDE Integrated Development Environment

IP Internet Protocol

JAR Java Archive

JDBC Java Database Connectivity

JSON JavaScript Object Notation

RCA Radio Corporation of America

SD Secure Digital

SDHC Secure Digital High Capacity

SDK Software Development Kit

SGBD Sistema de Gerenciamento de Banco de Dados

SOA Service Oriented Architecture

SOA-MC Service Oriented Architecture - Multiple Client

SOAP Simple Object Access Protocol

SQL Structured Query Language

SSH Secure Shell

THD Total Harmonic Distortion

URL Uniform Resource Locator

USB Universal Serial Bus

WSD Web Service Description

WSDL Web Services Description Language

XML eXtensible Markup Language

Sumário

Lista de Figuras	xi
Lista de Tabelas	xii
1 Introdução	1
1.1 Objetivo	1
1.2 Metodologia	1
1.3 Estrutura do Documento	2
2 O Sistema CloudPowerSave	3
2.1 Motivação e Concepção do Sistema	3
2.2 Arquitetura do CloudPowerSave	4
2.2.1 CloudDNS	5
2.3 SOA-MC	6
2.4 Os Medidores Powersave	8
2.5 Sequência de Evolução do Hardware	10
3 Ferramentas Utilizadas no Desenvolvimento do Sistema	13
3.1 Módulo Cloud	13
3.1.1 A Escolha do SGBD	13
3.1.2 Axis2	14
3.2 Módulo Bundler (CloudPowerSave Box)	15
3.3 Módulo Mobile	15
3.3.1 aChartEngine	17
3.3.2 db4o	19
4 O Módulo Cloud	20
4.1 <i>Web Services</i>	20
4.2 O Banco de Dados CloudPowerSave	21
4.2.1 <i>Data Access Objects</i> (DAOs)	22
4.2.2 As Entidades do Sistema	22
4.2.3 Modelagem	23

4.3	Principais Métodos	25
5	O Módulo Bundler (CloudPowerSave Box)	27
5.1	Principais Classes	27
5.1.1	A Classe SerialCom	27
5.1.2	A Classe PowerSaveMeterDriver	28
5.1.3	A Classe RealSystemDriver	29
5.1.4	A Classe ServiceProvider	29
5.2	Funcionamento do Módulo	30
5.2.1	O Arquivo de Configuração	31
6	O Módulo Mobile	34
6.1	A Aplicação Android	34
6.1.1	Tela Inicial	35
6.1.2	Tela de <i>Login</i>	36
6.1.3	Tela de Seleção de Meter Group	37
6.1.4	Tela de Seleção de Meter	38
6.1.5	Tela de Seleção de Intervalo de Tempo	39
6.1.6	Tela de Gráfico	41
6.1.7	Tela de Seleção de Gráfico Salvo	43
6.1.8	Tela de Preferências	44
6.1.9	Tela de Informações da Aplicação	45
7	Conclusões	46
8	Trabalhos Futuros	47
8.1	Medição de Outros Parâmetros de Energia Elétrica	47
8.2	Publicação no Google Play	48
8.3	Aplicação iOS	48
8.4	Página <i>Web</i>	48
8.5	Cortar a Energia Remotamente	48
8.6	DOUA (<i>Database Oriented Usecase Authorization</i>)	49
	Referências Bibliográficas	50

Lista de Figuras

2.1	<i>Arquitetura do CloudPowerSave.</i>	5
2.2	<i>Arquitetura SOA-MC.</i>	7
2.3	<i>Medidor Powersave.</i>	9
2.4	<i>Estrutura de um CloudPowerSave Box.</i>	11
2.5	<i>CloudPowerSave Box.</i>	12
3.1	<i>Editor de interface gráfica do plugin ADT.</i>	16
3.2	<i>Emulador de dispositivo móvel Android.</i>	17
3.3	<i>Gráfico gerado com a biblioteca aChartEngine.</i>	18
3.4	<i>Outro gráfico gerado com a biblioteca aChartEngine.</i>	18
4.1	<i>Modelo do banco de dados.</i>	23
6.1	<i>Tela inicial.</i>	35
6.2	<i>Tela de login.</i>	36
6.3	<i>Tela de seleção de Meter Group.</i>	37
6.4	<i>Tela de seleção de Meter.</i>	38
6.5	<i>Parte superior da tela de seleção de intervalo de tempo.</i>	39
6.6	<i>Parte inferior da tela de seleção de intervalo de tempo.</i>	40
6.7	<i>Tela de gráfico.</i>	41
6.8	<i>Dialog box da tela de gráfico.</i>	42
6.9	<i>Tela de seleção de gráfico salvo.</i>	43
6.10	<i>Tela de preferências.</i>	44
6.11	<i>Tela de informações da aplicação.</i>	45

Lista de Tabelas

3.1	Ferramentas utilizadas para desenvolvimento do módulo Cloud. . . .	13
3.2	Ferramentas utilizadas para desenvolvimento do módulo Bundler. . .	15
3.3	Ferramentas utilizadas para desenvolvimento do módulo Mobile. . . .	15

Capítulo 1

Introdução

1.1 Objetivo

Este trabalho é a reportagem do trabalho de projetar e implementar um sistema de monitoramento remoto de energia elétrica chamado de CloudPowerSave.

O sistema é composto por 3 módulos de *software*: um *Web Service*, um programa que interage com aparelhos medidores, e uma aplicação para dispositivos móveis baseados no sistema operacional Android, do Google. Estes módulos serão chamados de módulo Cloud, módulo Bundler e módulo Mobile respectivamente.

1.2 Metodologia

Para guiar o desenvolvimento do projeto, foi necessário escolher uma metodologia de desenvolvimento de software. Diversas opções foram estudadas durante a graduação, porém, por sugestão do orientador, a escolhida foi o *Scrum Academic* [11].

O *Scrum Academic* é uma versão simplificada do *Scrum*, possuindo um escopo menor e sendo aplicada em projetos acadêmicos. São previstas várias reuniões com o usuário, representado pelo orientador no caso deste projeto. Nestas reuniões são discutidas as prioridades, as mudanças e as melhorias que podem ser feitas. Devem ser apresentadas versões cada vez mais próximas da versão final, que são avaliadas pelo usuário até a conclusão do projeto. O código fonte é compartilhado com o orientador através de um repositório.

Por fim, a modelagem do sistema seguiu o padrão de orientação a objetos, com o objetivo de facilitar o entendimento do problema e obter uma maior organização de todas as partes do projeto. Além disso, a orientação a objetos é uma forma eficiente de implementar o sistema, e que torna mais fácil a manutenção do mesmo.

1.3 Estrutura do Documento

O capítulo 2 apresenta uma visão geral do sistema, incluindo sua arquitetura e o hardware utilizado. Apresenta também uma breve teoria envolvendo a arquitetura SOA-MC [6] (*Service Oriented Architecture - Multiple Client*), utilizada no sistema.

O capítulo 3 apresenta as ferramentas e tecnologias utilizadas para o desenvolvimento do sistema.

O capítulo 4 apresenta o módulo Cloud, que corresponde ao *Web Service* do sistema. Ele apresenta uma breve teoria envolvendo *Web Services*, o banco de dados do sistema e os principais métodos implementados.

O capítulo 5 apresenta o módulo Bundler, que corresponde ao programa que interage com os aparelhos medidores. Ele explica o funcionamento do módulo, a comunicação com os medidores e as classes mais importantes.

O capítulo 6 apresenta o módulo Mobile, que corresponde à aplicação para dispositivos móveis Android. Ele explica o funcionamento das telas presentes na aplicação.

O capítulo 7 apresenta as conclusões sobre este trabalho, e possíveis trabalhos futuros serão explicados no capítulo 8.

Capítulo 2

O Sistema CloudPowerSave

2.1 Motivação e Concepção do Sistema

A concepção inicial deste projeto era criar um sistema que utilizasse as tecnologias *cloud* e *mobile*, em virtude do interesse e da prática adquiridos através de disciplinas da graduação, em especial a de *Software* para *Smartphone* ministrada pelo orientador. A alta relevância dos *smartphones* atualmente também foi um fator importante, uma vez que o número de usuários cresce cada vez mais, junto do poder de processamento dos aparelhos. Além disso, a capacidade dos *smartphones* de se conectarem à *internet* abre a possibilidade da criação de inúmeras aplicações inovadoras.

Sendo assim, através de diversas conversas com o orientador, decidiu-se que a parte do projeto que utilizaria tecnologia *cloud* seria um *Web Service* e funcionaria como um depósito de dados, que seria alimentado de alguma forma a ser pensada. Já a parte que envolveria tecnologia *mobile* seria uma aplicação para dispositivos móveis usando o sistema operacional Android, e consumiria estes dados, faria algum processamento, e exibiria algum tipo de estatística ou gráfico para o usuário. Finalmente, o usuário poderia salvar este resultado na memória interna de seu dispositivo, permitindo o acesso *offline* às informações. Foi cogitada a ideia de fazer um projeto genérico: os dados poderiam ser quaisquer, de acordo com o interesse do usuário.

Após certo tempo, porém, o orientador sugeriu a utilização de um produto desenvolvido por dois de seus amigos, da empresa Powersave. A Powersave produz um aparelho, o medidor Powersave, capaz de medir e armazenar diversas informações de aparelhos eletrônicos que estejam conectados, e enviar estas informações para um computador via porta USB.

Por fim, decidiu-se que este projeto usaria estes medidores da Powersave, e trabalharia com dados referentes à energia elétrica. A nova concepção do projeto seria utilizar estes dados para monitoramento remoto, através da geração de gráficos em

um dispositivo móvel. A possibilidade de monitorar remotamente a energia elétrica é altamente relevante, tanto para empresas quanto para pessoas que desejam cortar gastos e, ao mesmo tempo, agir de forma sustentável e preservar o meio ambiente. O projeto recebeu o nome CloudPowerSave, e pode ser descrito como um sistema de monitoramento remoto de energia elétrica, envolvendo as tecnologias *cloud* e *mobile*.

A seção seguinte apresenta a arquitetura do sistema.

2.2 Arquitetura do CloudPowerSave

O sistema CloudPowerSave é composto por um *Web Service* e duas aplicações clientes.

O *Web Service*, chamado de módulo Cloud, interage com um banco de dados remoto desenvolvido pelo aluno, que poderá ser acessado por ambas aplicações clientes. Este banco de dados armazenará dados dos usuários e medidas referentes à energia elétrica.

Um dos clientes é um programa que interage com medidores, chamado de módulo Bundler. Ele executa em um aparelho desenvolvido sob encomenda para este projeto, chamado CloudPowerSave Box, composto por 3 aparelhos medidores e um mini computador. Este aparelho será detalhado mais adiante. O programa, executando no mini computador, se comunica com os medidores e alimenta o *Web Service* com medidas.

O outro cliente é uma aplicação Android, chamado de módulo Mobile. Esta aplicação consome o *Web Service* para obter as medidas efetuadas e gerar gráficos, permitindo o monitoramento remoto. Ele também permite o armazenamento dos gráficos na memória interna do dispositivo móvel.

A figura 2.1 exibe a arquitetura do sistema, mostrando o *Web Service* e seus clientes.

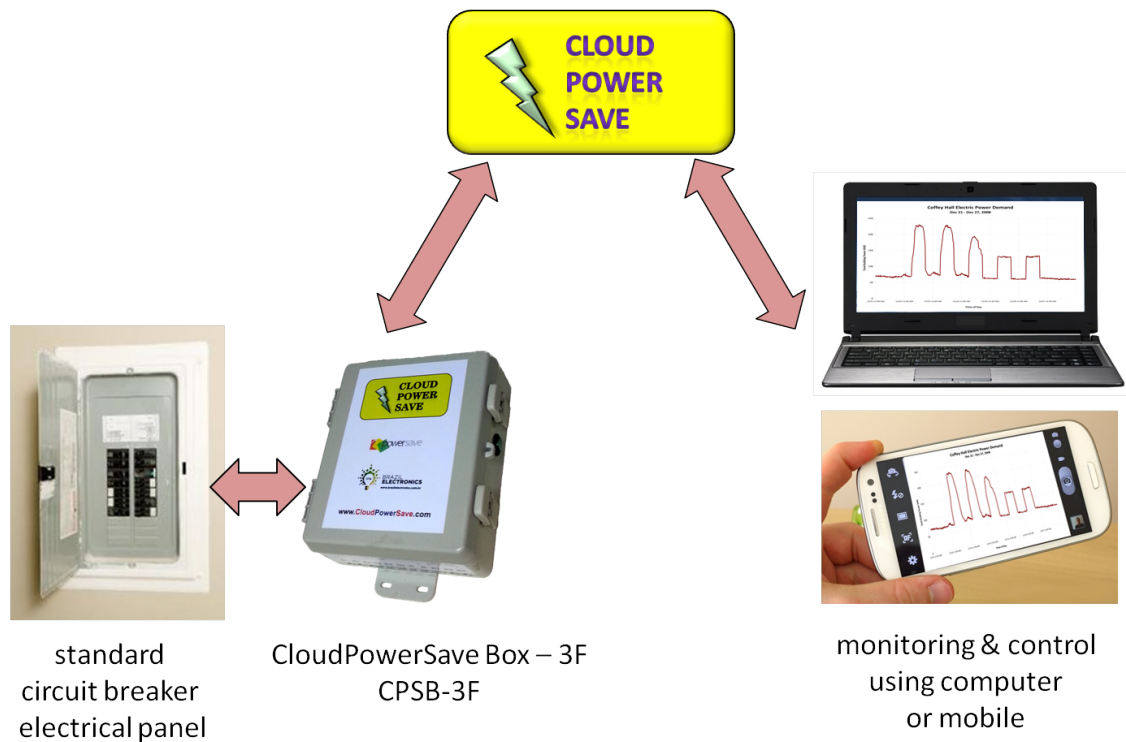


Figura 2.1: Arquitetura do CloudPowerSave [8].

Na parte de cima da figura temos o *Web Service* do CloudPowerSave, que funciona como um depósito de dados.

Do lado esquerdo da figura temos um painel de circuitos elétricos, cujas fases são ligadas a um CloudPowerSave Box (atualmente suporta 3 fases). O módulo Bundler é executado no CloudPowerSave Box e coleta medidas, enviando-as para o *Web Service*.

Do lado direito da figura temos um *smartphone* e um *laptop*. A aplicação Android, executando no *smartphone*, recupera medidas armazenadas no *Web Service* e gera gráficos. Um trabalho futuro será permitir que esta funcionalidade possa ser acessada também a partir de uma página *web*, através de um *laptop* por exemplo. Trabalhos futuros serão discutidos no capítulo 8.

Na versão atual do sistema, o cadastro de usuário está sendo feito pelo módulo Bundler, como será explicado no capítulo 5. Esta é uma funcionalidade provisória do módulo, que deverá ser desempenhada pela página *web* a ser desenvolvida futuramente.

2.2.1 CloudDNS

Para as aplicações clientes se comunicarem com o *Web Service*, primeiro é feita uma consulta à página CloudDNS: <http://www.cloudpowersave.com/clouddns.php>.

O conteúdo desta página é o endereço IP e a porta do servidor *web* que executa o *Web Service* do sistema. Dessa forma, caso seja necessário alterar a máquina remota que executa o *Web Service*, bastaria atualizar o conteúdo desta página.

A seção seguinte apresenta uma breve teoria envolvendo a arquitetura SOA-MC [6], que serviu de base para o desenvolvimento do projeto.

2.3 SOA-MC

Como visto em [1], uma arquitetura orientada a serviços (SOA - *Service Oriented Architecture*) é uma estrutura que define como duas entidades computacionais - por exemplo, dois programas - interagem de modo a permitir que uma entidade realize uma unidade de trabalho para a outra entidade. Estas interações são definidas usando uma linguagem de descrição.

Outra definição de SOA, vista em [2], seriam as políticas, práticas e *frameworks* que permitem que funcionalidades de uma aplicação sejam providas e consumidas como conjuntos de serviços publicados numa granularidade relevante ao consumidor do serviço. Serviços podem ser invocados, publicados e descobertos, e são abstraídos da implementação usando uma única forma de interface baseada em padrões. Dessa forma, as interfaces providas por uma arquitetura orientada a serviços devem ser independentes de plataforma, permitindo que ambientes e plataformas totalmente distintos possam usufruir de suas funcionalidades.

Além disso, uma arquitetura orientada a serviços deve promover a reutilização. Serviços podem ser reutilizados para gerar serviços compostos, reaproveitando as funcionalidades já desenvolvidas e evitando o retrabalho, que prejudica o tempo de desenvolvimento de uma nova aplicação. As funcionalidades de um serviço precisam ser implementadas apenas uma vez.

Web Services baseados no protocolo SOAP (*Simple Object Access Protocol*) estão se tornando a implementação de SOA mais comum. SOAP é um protocolo leve para troca de informação estruturada em um ambiente descentralizado e distribuído [4], baseado em XML (*eXtensible Markup Language*). Seus principais objetivos são simplicidade e extensibilidade, e suas mensagens podem ser transportadas por diversos protocolos da *internet*.

No caso deste projeto, foi utilizada uma implementação do SOA chamada SOA-MC (*Service-Oriented Architecture - Multiple Client*) [6]. O SOA-MC [6] trabalha com 2 camadas, que se comunicam enviando dados através do protocolo HTTP (*HyperText Transfer Protocol*). Os formatos de dados mais comuns são XML, JSON (*JavaScript Object Notation*) e CSV (*Comma-Separated Values*). A figura 2.2 ilustra esta arquitetura.

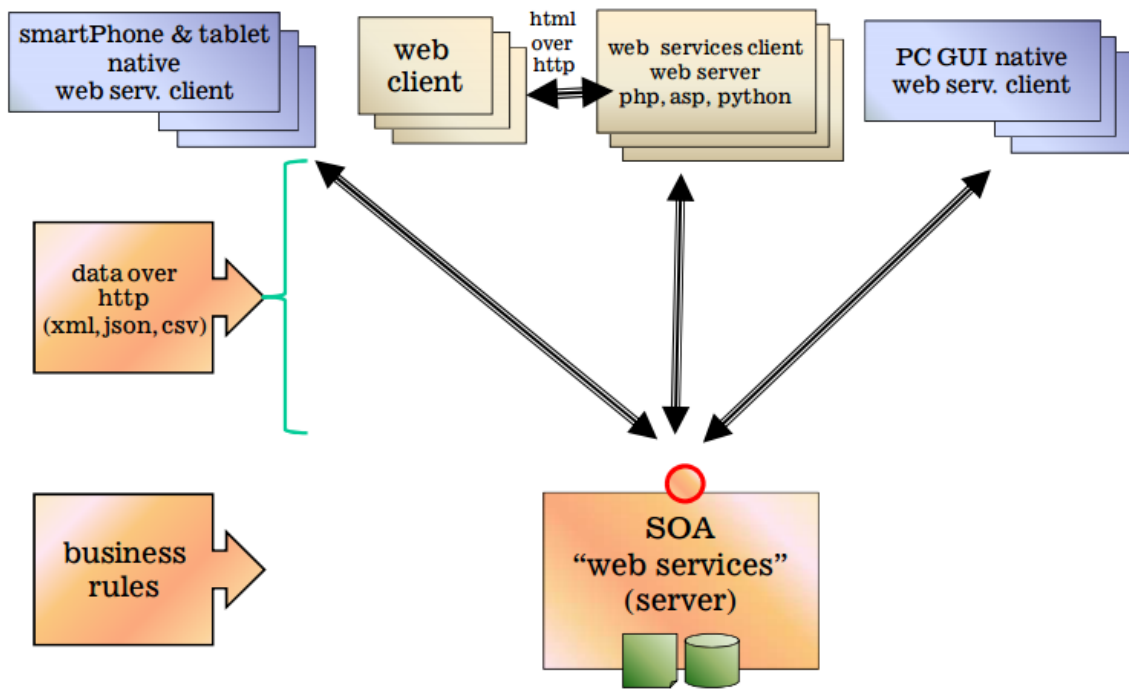


Figura 2.2: Arquitetura SOA-MC [6].

A camada superior contém os clientes, que podem ser de múltiplas plataformas, justificando o nome da arquitetura. A figura mostra, como exemplo, que os clientes poderiam ser dispositivos móveis nativos (*smartphones* e *tablets*), *web servers* (acessados pelo cliente através de um *web client*) ou aplicações GUI nativas. A camada inferior implementa as regras de negócio, oferecendo funcionalidade como *Web Services*.

As principais vantagens do SOA-MC [6] são:

- As regras de negócio são escritas uma única vez para todos os múltiplos clientes.
- Não há necessidade de escrever todos os clientes de uma só vez. Eles podem ser desenvolvidos de acordo com a necessidade e com a capacidade de desenvolvimento.
- O mesmo sistema pode ser acessado pelo mesmo usuário por diferentes clientes de *software*, dependendo da necessidade do usuário. Por exemplo, um usuário poderia utilizar um sistema através de um *web browser* em um computador *desktop* em casa, e quando estivesse fora de casa, poderia acessar o mesmo sistema através de um cliente *smartphone* nativo.
- Esta arquitetura separa completamente as regras de negócio e a camada de apresentação, para o cliente *web* do sistema. Isto permite separar a equipe

de regras de negócio (geralmente mais experiente e mais cara) da equipe de apresentação (geralmente menos experiente e mais barata). A equipe de apresentação pode usar linguagens de *script* fáceis de se aprender como PHP, e a estabilidade do sistema não será comprometida, pois as regras de negócio são mantidas pela outra equipe.

- É possível ter diversas páginas *web* como clientes do mesmo sistema, e ter diferentes apresentações para cada uma. Por exemplo, o sistema pode ser desenvolvido e vendido para diversas firmas, cada uma tendo sua própria página *web*.
- A separação de camadas implicitamente provida pelo SOA-MC [6] pode gerar um processo de desenvolvimento mais barato do sistema de *software*.

2.4 Os Medidores Powersave

Para realizar as medidas necessárias para o monitoramento remoto da energia elétrica, escolheu-se trabalhar com os medidores da empresa Powersave. Como descrito na página da empresa [7], o medidor Powersave é capaz de registrar a curva de consumo de um equipamento para promover a eficiência energética. Um de seus diferenciais é a facilidade de uso, pois para instalá-lo é necessário simplesmente inseri-lo em uma tomada e em seguida conectar a tomada do aparelho a ser medido. Este método suporta corrente máxima de 10 amperes. Para equipamentos de alto consumo, o barramento lateral é usado, suportando uma única fase com corrente máxima de 50 amperes. A tensão suportada é de 100 a 240VAC.

A figura 2.3, exibida na página da empresa, mostra um medidor.



Figura 2.3: Medidor Powersave [7].

Este medidor é capaz de informar medidas instantâneas de consumo, tensão, corrente, potência, fator de potência e frequência. Para obter as medidas, é possível acessar a memória interna do medidor ou fazer pedidos em tempo real. O medidor possui memória interna para *log* com intervalo de 5 ou 15 minutos, e capacidade de 3 meses de armazenamento. O CloudPowerSave trabalha com pedidos em tempo real. Estes dados e outras funções podem ser acessados por comandos via porta USB.

Nesta primeira versão do sistema, escolheu-se realizar o monitoramento através da geração de gráficos de potência (em Watts). A vantagem de utilizar a potência é a facilidade de leitura dos gráficos gerados: momentos onde houve alto consumo de energia aparecem como picos, e podem ser facilmente diferenciados dos momentos onde houve baixo consumo de energia. Futuras versões do sistema permitirão a geração de gráficos das outras medidas também.

2.5 Sequência de Evolução do Hardware

Inicialmente, o programa responsável pela comunicação com os medidores funcionava em um computador conectado a um ou mais medidores via USB. Este modo de trabalho não permitia flexibilidade no uso do sistema, pois o conjunto formado pelo computador e pelos medidores não era de fácil transporte e instalação. Além disso, para usar o sistema em um novo computador, havia a necessidade de realizar uma configuração inicial. Porém, esta primeira montagem serviu como base importante para o desenvolvimento.

Como primeiro aprimoramento, foi projetado um protótipo chamado Cloud-PowerSave Box, que foi desenvolvido sob encomenda pela Powersave. Este protótipo é composto por um mini computador Raspberry Pi modelo B e 3 medidores Powersave, todos armazenados dentro de uma caixa. Os medidores são conectados internamente ao mini computador, que executa o programa. Dessa forma, 3 fases podem ser monitoradas, cada uma com corrente máxima de 50 amperes. As especificações do Raspberry Pi modelo B, como visto em [18], são as seguintes:

- CPU ARM1176JZF-S core, com clock de 700MHz
- GPU Broadcom VideoCore IV, com clock de 250MHz
- Memória de 512MB (compartilhada com a GPU)
- 2 portas USB
- Saídas de vídeo RCA Composto e HDMI
- Saída de áudio HDMI
- Armazenamento onboard com slot para cartão SD ou SDHC
- Rede onboard Ethernet
- Potência de 3.5W
- Fonte de energia de 5V via MicroUSB
- Sistema operacional Raspbian, uma distribuição Linux específica para o Raspberry Pi

A figura 2.4 mostra a estrutura do protótipo.

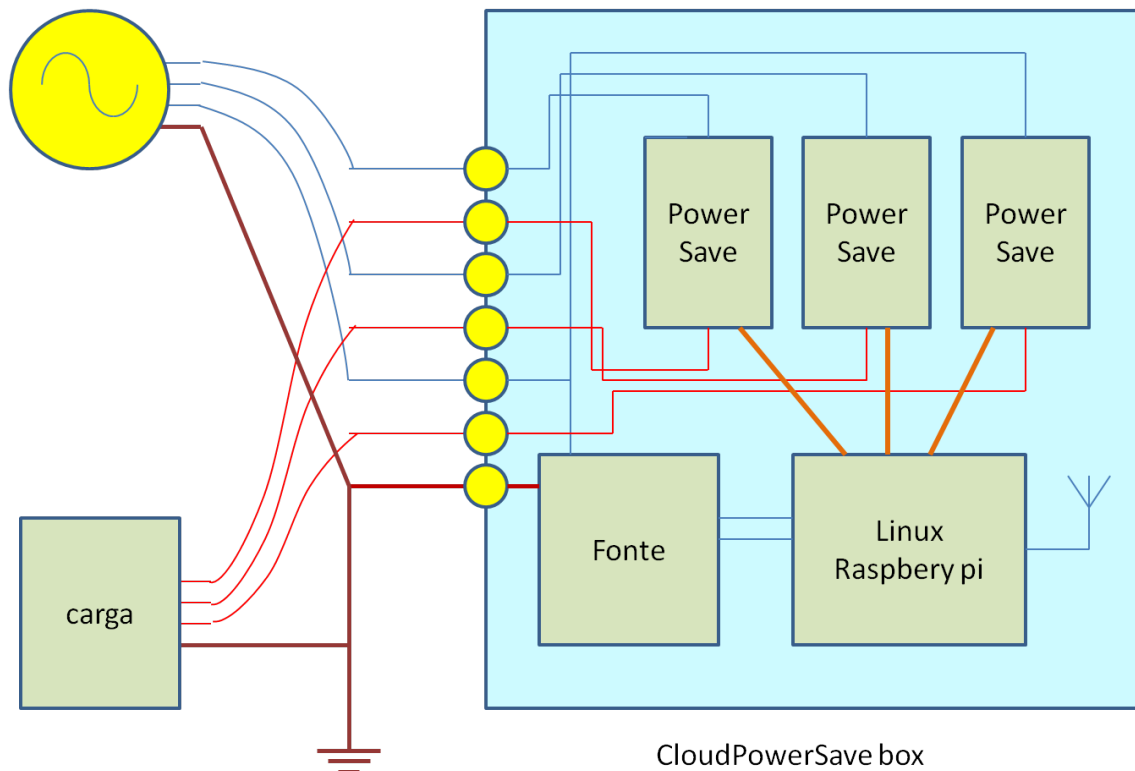


Figura 2.4: Estrutura de um CloudPowerSave Box.

Nesta montagem, os medidores não possuem sua carcaça e são conectados a um *hub* USB, que por sua vez é conectado ao mini computador. Um adaptador *wireless* USB é conectado também ao mini computador, permitindo a conexão à *internet* e, conseqüentemente, acesso ao *Web Service*. Este adaptador deve ser configurado para que se conecte automaticamente a alguma rede *wireless*. Existem 3 pares de entrada e saída (indicados no CloudPowerSave Box por E1 e S1, E2 e S2, E3 e S3), cada par ligado a um medidor. Em cada entrada, liga-se a fonte de energia da fase a ser monitorada (que alimentará também o medidor), e na saída a carga a ser alimentada. Existem também dois pontos para ligar o neutro (indicados no CloudPowerSave Box por N). Uma das entradas alimenta também uma fonte, que por sua vez alimenta o mini computador. O CloudPowerSave Box possui também uma chave liga/desliga. Ao ligá-la, o mini computador automaticamente é ligado.

A figura 2.5 mostra um CloudPowerSave Box.



Figura 2.5: CloudPowerSave Box.

Capítulo 3

Ferramentas Utilizadas no Desenvolvimento do Sistema

Este capítulo descreve as ferramentas e tecnologias escolhidas para o desenvolvimento de cada um dos módulos do sistema. As ferramentas e tecnologias mais importantes serão explicadas nas subseções.

3.1 Módulo Cloud

A tabela 3.1 contém as ferramentas utilizadas no desenvolvimento do módulo Cloud.

Tabela 3.1: Ferramentas utilizadas para desenvolvimento do módulo Cloud.

Ferramenta	Explicação
Netbeans IDE 7.3	IDE para desenvolvimento do módulo Cloud
Java SE 7 Development Kit (JDK)	Kit de desenvolvimento para aplicações Java
Apache Tomcat 7.0.34	Servidor para hospedagem do serviço
Apache Axis2 1.6.2	Ferramenta para criação de <i>Web Services</i>
MySQL Server 5.5.34	Sistema de gerenciamento de banco de dados

3.1.1 A Escolha do SGBD

Para manipular dados de forma confiável e eficiente, é fundamental a escolha de um bom SGBD (Sistema de Gerenciamento de Banco de Dados). Durante a graduação, diversos SGBDs foram apresentados e estudados. Porém, houve contato

com apenas três de forma prática, todos utilizando o modelo relacional e a linguagem SQL. São eles: MySQL, PostgreSQL e Microsoft SQL Server.

Tanto as consultas quanto as entidades do banco de dados do CloudPowerSave são simples. Nesta versão inicial do sistema, o volume de dados e frequência das consultas são pequenos. Dessa forma, os principais critérios para a escolha do SGBD foram a experiência prévia e a facilidade de implementação. Tendo em vista estes critérios, o SGBD escolhido foi o MySQL. O Microsoft SQL Server necessitaria de um computador com o sistema operacional Windows, e como o servidor inicial que executa o *Web Service* do CloudPowerSave usa Linux, esta opção foi descartada. O PostgreSQL, assim como o MySQL, é de uso bastante difundido. Porém, pela experiência prévia maior, popularidade em aplicações *web*, facilidade de obtenção de informações a respeito de solução de problemas na *internet* e recomendação do orientador, a opção escolhida foi o MySQL.

3.1.2 Axis2

Por sugestão do orientador, para o desenvolvimento do *Web Service* do CloudPowerSave foi utilizada a ferramenta Axis2. O projeto Apache Axis2 é uma implementação baseada em Java tanto do lado cliente quanto do lado servidor de um *Web Service*, e funciona com o servidor *web* da Apache, o Tomcat. Ele permite realizar tarefas como: enviar mensagens SOAP, receber e processar mensagens SOAP, criar um *Web Service* a partir de uma classe Java, criar classes de implementação tanto do servidor quanto do cliente usando WSDL (*Web Services Description Language*), enviar e receber mensagens SOAP com anexos, entre outras.

Para trabalhar com esta ferramenta, foi utilizado um *plugin* disponível para a IDE Netbeans chamado Axis2 Support. Este plugin automatiza boa parte da criação do serviço, e agilizou bastante o desenvolvimento desta parte do projeto. Para criar um novo serviço, basta seguir um *wizard*, que cria uma classe Java. Os métodos desta classe serão as funcionalidades do *Web Service*.

O *plugin* também faz o *deploy* do serviço para o Tomcat, e permite que as funcionalidades sejam testadas em qualquer *web browser*. Como o IDE Netbeans também possui compatibilidade com o Tomcat, todo o desenvolvimento do *Web Service* foi realizado no IDE e testado localmente. Para fazer o *deploy* do serviço na máquina remota, basta enviar o arquivo (formato *.aar*) gerado pelo Axis2 por SSH (*Secure Shell*) para a pasta adequada do Tomcat. O serviço então está pronto para ser utilizado.

3.2 Módulo Bundler (CloudPowerSave Box)

A tabela 3.2 contém as ferramentas utilizadas no desenvolvimento do módulo Bundler.

Tabela 3.2: Ferramentas utilizadas para desenvolvimento do módulo Bundler.

Ferramenta	Explicação
Netbeans IDE 7.3	IDE para desenvolvimento do módulo Bundler
Java SE 7 Development Kit (JDK)	Kit de desenvolvimento para aplicações Java

3.3 Módulo Mobile

A tabela 3.3 contém as ferramentas utilizadas no desenvolvimento do módulo Mobile.

Tabela 3.3: Ferramentas utilizadas para desenvolvimento do módulo Mobile.

Ferramenta	Explicação
Eclipse 4.3	IDE para desenvolvimento do módulo Mobile
Java SE 7 Development Kit (JDK)	Kit de desenvolvimento para aplicações Java
Android SDK Tools	Ferramentas de desenvolvimento para Android
Android 2.3.3	Versão do sistema operacional utilizada
aChartEngine 1.0.0	Biblioteca para geração de gráficos
db4objects 8.0	Banco de dados para Android

Como o aluno possui experiência com desenvolvimento para Android, e não possui nenhuma experiência com desenvolvimento para o iOS da Apple, foi feita a escolha de desenvolver o módulo Mobile para Android. Uma possível expansão para o projeto é o desenvolvimento do módulo também para iOS, como será citado no capítulo 8.

O IDE Eclipse foi escolhido para desenvolver o módulo Mobile, em função de seu suporte ao *plugin* ADT (*Android Development Tools*). Este *plugin* traz inúmeras facilidades para o desenvolvimento de aplicações Android, como por exemplo: criação

de projetos Android rapidamente através de um *wizard*, editor de interface gráfica, *debug* da aplicação usando ferramentas do SDK do Android, exportação de arquivos (formato *.apk*) para a distribuição da aplicação e diversas outras ferramentas integradas.

O *plugin* ADT também acompanha um emulador de dispositivo móvel Android, facilitando o desenvolvimento e permitindo que testes sejam realizados rapidamente no próprio computador do desenvolvedor. A Google disponibiliza um pacote com o IDE e o *plugin* juntos, e recomenda o uso do mesmo para o desenvolvimento de aplicações Android. As figuras 3.1 e 3.2 mostram o editor de interface gráfica do IDE e o emulador de dispositivo móvel, respectivamente.

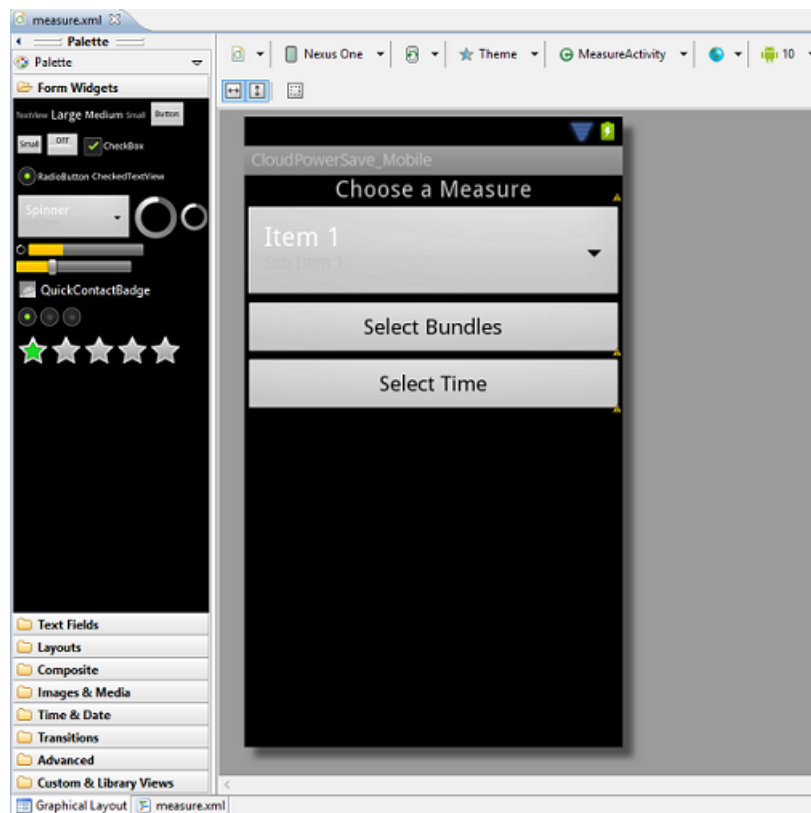


Figura 3.1: Editor de interface gráfica do *plugin* ADT.

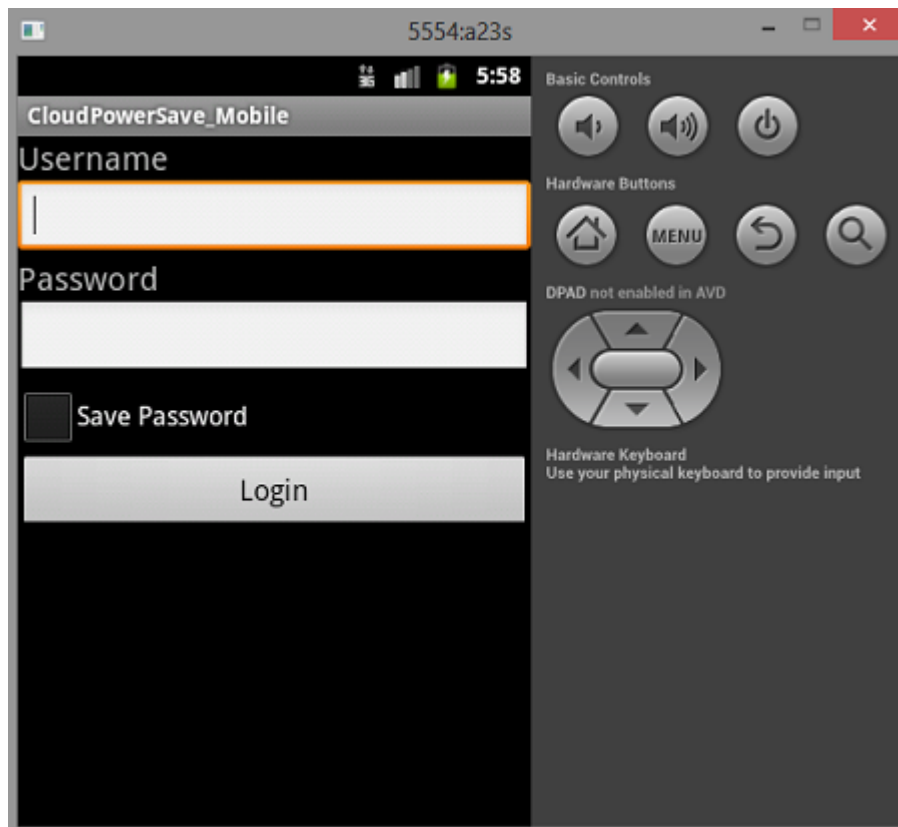


Figura 3.2: Emulador de dispositivo móvel Android.

A aplicação foi desenvolvida para a versão 2.3.3 do Android, também funcionando em versões posteriores, suportando tanto *smartphones* quanto *tablets*.

3.3.1 aChartEngine

O aChartEngine é uma biblioteca de criação de gráficos para aplicações Android, que suporta diversos tipos de gráficos: linha, área, espalhamento, tempo, barra, combinações dos anteriores, etc. Todos estes tipos podem conter múltiplas séries, podem ser exibidos com o eixo X na horizontal ou na vertical e suportam muitas outras funcionalidades.

Além disso, inúmeras customizações são suportadas, como por exemplo: seleção da cor de fundo, seleção série a série da cor dos pontos e exibição dos mesmos como círculos, triângulos, quadrados ou outros formatos.

As figuras 6.3 e 6.4 mostram exemplos de gráficos gerados com a biblioteca.

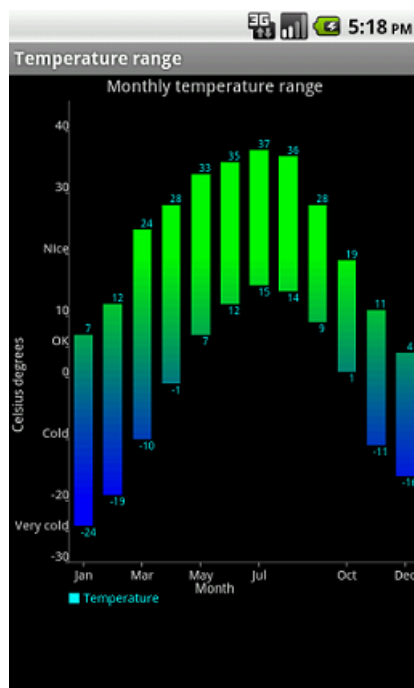


Figura 3.3: Gráfico gerado com a biblioteca aChartEngine [9].

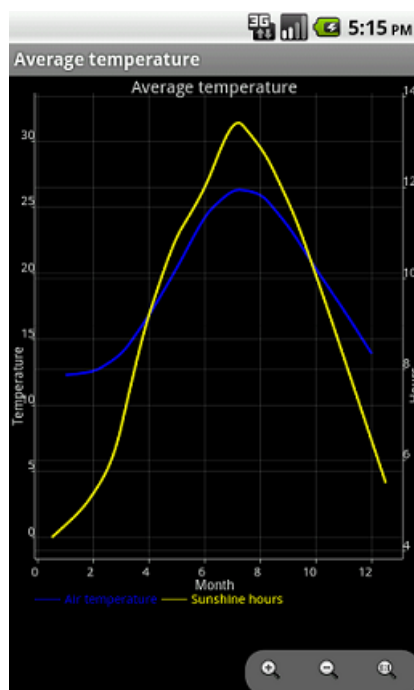


Figura 3.4: Outro gráfico gerado com a biblioteca aChartEngine [9].

A biblioteca acompanha diversos códigos de exemplo, que foram bastante esclarecedores para o aprendizado e auxiliaram no desenvolvimento dos gráficos da aplicação Android do CloudPowerSave.

3.3.2 db4o

Para armazenar gráficos na memória interna do dispositivo móvel e permitir a visualização *offline* dos mesmos, foi usado o banco de dados orientado a objetos db4objects (db4o). Este é um banco de dados bem leve e permite que desenvolvedores Java e .NET armazenem e recuperem qualquer objeto da aplicação com poucas linhas de código, eliminando a necessidade de definir e manter um modelo rígido separadamente. Além disso, é compatível com Android.

Como os objetos a serem armazenados pela aplicação são bem simples, o db4o foi escolhido pela facilidade de uso. Apenas olhando um único código de exemplo, já foi possível desenvolver a persistência local da aplicação. Uma das vantagens de um banco de dados orientado a objeto é que, ao invés de desmontar o objeto para o armazenamento em estruturas de tabelas, como no modelo relacional, o objeto é diretamente armazenado como objeto.

Capítulo 4

O Módulo Cloud

O módulo Cloud do CloudPowerSave corresponde ao *Web Service* do sistema. Sua finalidade é agir como um repositório de dados para as entidades do sistema, provendo serviços de criação, recuperação, atualização e deleção destes dados.

A seção seguinte apresenta brevemente a teoria envolvendo *Web Services*. Em seguida, o banco de dados do CloudPowerSave e os principais métodos do *Web Service* são apresentados.

4.1 *Web Services*

Um *Web Service* é um sistema de *software* desenvolvido para suportar interações máquina a máquina interoperáveis por uma rede. Possui uma interface descrita em um formato processável por máquina (mais especificamente, WSDL). Outros sistemas interagem com o *Web Service* de uma maneira definida por sua descrição, usando normalmente mensagens SOAP, tipicamente enviadas usando HTTP com serialização XML em conjunção com outros padrões relacionados a *web*.

WSDL é uma linguagem de descrição de interface baseada em XML usada para descrever a funcionalidade oferecida por um *Web Service*. Um arquivo WSDL fornece uma descrição de como o serviço pode ser chamado, que parâmetros ele espera e que estruturas de dados ele retorna. Dessa forma, ele funciona como uma assinatura de um método em uma linguagem de programação.

Um *Web Service* é uma noção abstrata que deve ser implementada por um agente concreto, ou seja, o agente é a peça de *software* ou *hardware* concreta que envia e recebe mensagens, e o serviço é o recurso caracterizado pelo conjunto abstrato de funcionalidades que ele provê. Por exemplo, um *Web Service* em particular pode ser implementado por um agente num dado dia, escrito em uma linguagem de programação, e no próximo dia por outro agente, escrito em uma linguagem de programação diferente. Apesar da mudança de agente, o *Web Service* continua o mesmo [13].

A finalidade de um *Web Service* é prover funcionalidade em nome de seu dono - uma pessoa ou organização, por exemplo um indivíduo ou um negócio. A entidade provedora é a pessoa ou organização que fornece um agente adequado para implementar um serviço em particular. A entidade consumidora é a pessoa ou organização que deseja fazer uso do *Web Service* da entidade provedora. A entidade consumidora usará um agente consumidor para trocar mensagens com o agente provedor da entidade provedora.

As mecânicas da troca de mensagem são documentadas numa descrição de *Web Service* (WSD - *Web Service Description*). WSD é uma especificação da interface do *Web Service*, escrita usando WSDL. Ela define formato de mensagens, tipos de dados, protocolos de transporte, e formatos de serialização de transporte que devem ser usados entre o agente do consumidor e o agente do provedor. Além disso, também especifica uma ou mais localidades de rede onde o agente provedor pode ser invocado, e pode fornecer alguma informação sobre o padrão de troca de mensagem que é esperado. Resumidamente, a descrição do serviço representa um acordo governando as mecânicas de interação com um serviço específico.

A semântica de um *Web Service* é a expectativa compartilhada sobre o comportamento do serviço, em particular em resposta às mensagens que são enviadas a ele. Na prática, este é o “contrato” entre a entidade consumidora e a entidade provedora no que diz respeito ao objetivo e às consequências da interação. Apesar deste contrato representar uma concordância entre a entidade consumidora e a entidade provedora sobre como e por que seus respectivos agentes irão interagir, ele não é necessariamente escrito ou explicitamente negociado. Ele pode ser explícito ou implícito, oral ou escrito, processável por máquina ou orientado a humanos, e pode ser um acordo legal ou informal (não-legal). Enquanto a descrição do serviço representa um contrato que governa as mecânicas de interação com um serviço particular, a semântica representa um contrato governando o significado e objetivo desta interação.

4.2 O Banco de Dados CloudPowerSave

Os dados do CloudPowerSave consistem de informações dos usuários, informações dos medidores e medidas propriamente ditas. Esses dados são acessados pelas duas aplicações clientes, o módulo Bundler e o módulo Mobile. A comunicação com o banco de dados foi implementada usando *Data Access Objects*, que serão explicados adiante.

4.2.1 *Data Access Objects* (DAOs)

Os acessos ao banco de dados do CloudPowerSave são feitos através de *Data Access Objects*. Um *Data Access Object* (DAO) tem como objetivo isolar a conexão ao banco de dados, permitindo que os dados somente sejam acessados através destes objetos e abstraindo a implementação da conexão com o banco do resto do projeto. Dessa forma, torna-se mais simples alterar a implementação da conexão com o banco se necessário, ou até mesmo alterar o SGBD utilizado, sem modificações no resto do projeto. Para isso, basta que a assinatura dos métodos das classes DAO não sejam alteradas. Dessa forma, caso versões futuras do sistema possuam diferentes necessidades de banco de dados, bastará alterar as classes DAO e o *driver* JDBC (Java Database Connectivity) responsável pela conexão com o banco de dados.

As classes DAO possuem o endereço do banco de dados, a porta utilizada, o nome do banco e as informações de autorização (usuário e senha). Para cada entidade do sistema, existe uma classe DAO.

4.2.2 As Entidades do Sistema

A seguir são detalhadas as entidades do sistema. São elas: Customer, Meter Group, Meter e Meter Data Bundle. Em certas partes deste texto, serão feitas referências a essas entidades.

Customer representa o cliente, ou usuário do sistema. Este usuário possui um ou vários aparelhos CloudPowerSave Box, e deseja monitorar remotamente sua energia elétrica.

Meter representa um único medidor Powersave. O medidor estará constantemente gerando medições, e estas serão enviadas pelo módulo Bundler para o *Web Service*.

Um Meter Group é um conjunto de Meters. O usuário deve organizar seus Meters como desejar, e agrupá-los em Meter Groups. Como exemplo, um usuário poderia definir um Meter Group chamado “Minha Casa”, para agrupar os medidores (Meters) de sua residência, e outro chamado “Minha Empresa”, para agrupar os de sua empresa. É importante notar que, apesar de um CloudPowerSave Box conter 3 medidores (ou seja, 3 Meters), não existe nenhuma obrigação de que eles pertençam ao mesmo Meter Group.

Por fim, um conjunto de medições feitas por um Meter são representadas por um Meter Data Bundle. Este conjunto pode conter um número variável de medidas, e deve ter uma data/hora inicial e uma taxa de amostragem definidas. As medidas são guardadas por uma única *string*, separadas por vírgulas.

4.2.3 Modelagem

A modelagem do banco de dados foi feita pelo aluno, e todo o processo de criação do banco, das tabelas e dos usuários do banco foi feito pela linha de comando do MySQL Server.

O banco de dados foi modelado em inglês e possui 4 tabelas, representando as entidades: *tb_customer*, *tb_metergroup*, *tb_meter* e *tb_meterdatabundle*. A figura 4.1 exibe o diagrama entidade-relacionamento do banco de dados do CloudPowerSave, gerado com a ferramenta Erwin.

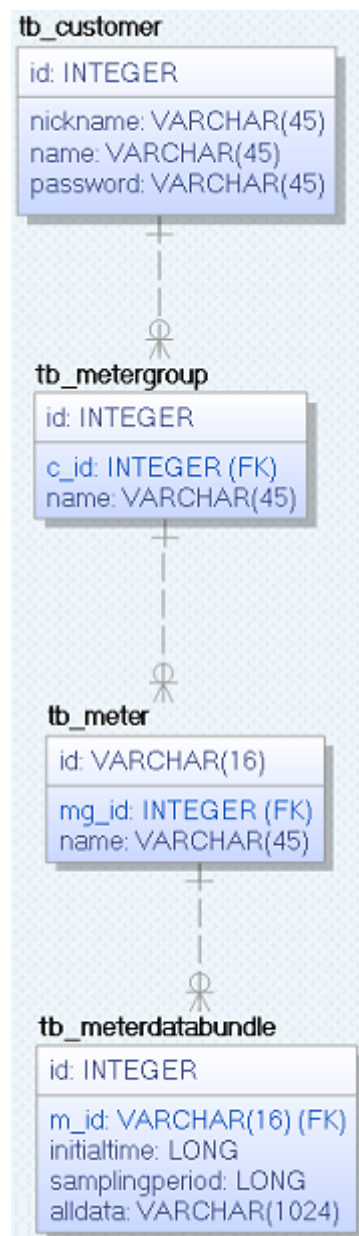


Figura 4.1: Modelo do banco de dados.

Abaixo segue uma breve descrição de cada entidade, com seus respectivos atributos. O tipo de cada atributo é informado entre parêntesis. Todas as entidades

possuem um campo *id*, um identificador que funciona como chave primária.

- Customer

Representa um usuário do CloudPowerSave. Possui 4 atributos: *id* (inteiro); *nickname* (*string*), o nome único do usuário no sistema; *name* (*string*), o nome real do usuário; e *password* (*string*), a senha do usuário. O *nickname* deve ser único.

- MeterGroup

Representa um conjunto de medidores Powersave pertencentes a um usuário. Possui 3 atributos: *id* (inteiro); *name* (*string*), o nome escolhido pelo usuário para o grupo; e *c_id* (inteiro), uma referência para a tabela Customer. Um usuário não pode ter dois Meter Groups com o mesmo nome, então a combinação de nome e *c_id* deve ser única.

- Meter

Representa um medidor Powersave, e pertence a um Meter Group. Possui 4 atributos: *id* (*string*), o identificador do medidor; *name* (*string*), o nome escolhido pelo cliente para o medidor; e *mg_id* (inteiro), uma referência para a tabela MeterGroup. É importante notar que, diferente das outras entidades, o atributo *id* do Meter é do tipo *string*, e corresponde ao seu identificador real. Cada medidor possui um identificador que pode ser lido ou escrito através de comandos via USB, como será mostrado no capítulo 5. Será assumido que os identificadores são únicos, e não serão alterados pelos usuários. Além disso, um usuário não pode ter dois Meters com o mesmo nome no mesmo Meter Group, então a combinação de nome e *mg_id* também deve ser única.

- MeterDataBundle

Representa um conjunto de medidas de potência (em Watts) realizadas por um Meter. Possui 5 atributos: *id* (inteiro); *alldata* (*string*), representando as medidas realizadas pelo medidor; *initialtime* (*long*), representando a data/hora da primeira medida; *samplingperiod* (*long*), representando a taxa de amostragem das medidas em milissegundos; e *m_id* (*string*), uma referência para a tabela Meter. A representação da data/hora pelo tipo *long* corresponde ao número de milissegundos desde 01/01/1970, 00:00:00 GMT (*Greenwich Mean Time*).

4.3 Principais Métodos

Para atender as necessidades das aplicações clientes, diversos métodos foram implementados no *Web Service*. Estes métodos correspondem às operações de criação, recuperação, atualização e deleção para cada uma das entidades do sistema, e representam os serviços que o *Web Service* disponibiliza. Os principais métodos são descritos abaixo:

- createCustomer

Este método insere um novo Customer no sistema, e seus parâmetros são o nome, o *nickname* e a senha do usuário. Ele retorna o *id* do Customer criado. É usado pelo módulo Bundler.

- createMeterGroup

Este método insere um novo Meter Group no sistema, e seus parâmetros são o nome do MeterGroup e o *id* do cliente dono deste Meter Group. Ele retorna o *id* do Meter Group criado. É usado pelo módulo Bundler.

- createMeter

Este método insere um novo Meter no sistema, e seus parâmetros são o nome do Meter, o *id* do medidor representado por este Meter e o *id* do Meter Group a que este Meter pertence. Ele retorna o *id* do Meter criado. É usado pelo módulo Bundler.

- createMeterDataBundle

Este método insere um novo Meter Data Bundle no sistema, e seus parâmetros são a data/hora inicial, a taxa de amostragem, a *string* contendo as medidas separadas por vírgula e o *id* do Meter que fez as medidas. Ele retorna o *id* do Meter Data Bundle criado.

Este é o método usado pelo módulo Bundler para fazer o envio de Meter Data Bundles para o *Web Service*.

- retrieveCustomer

Este método retorna um Customer. Seus parâmetros são o *nickname* e a senha do usuário.

Este método é usado para o *login* do usuário na aplicação Android.

- retrieveMeterGroup

Este método retorna uma lista de Meter Groups pertencentes a um Customer. Seu parâmetro é o *id* do Customer. É usado pela aplicação Android.

- retrieveMeter

Este método retorna uma lista de Meters pertencentes a um Meter Group. Seu parâmetro é o *id* do Meter Group. É usado pela aplicação Android.

- retrieveMeterDataBundle

Este método retorna uma lista de Meter Data Bundles pertencentes a um Meter, que estejam dentro de um certo intervalo de tempo. Seus parâmetros são o *id* do Meter, data/hora inicial e data/hora final. Meter Data Bundles que contenham pelo menos uma medida no intervalo informado são incluídos na lista de retorno.

Este é o método usado pela aplicação Android para recuperar as medidas armazenadas no *Web Service*.

Capítulo 5

O Módulo Bundler (CloudPowerSave Box)

O programa que executa no CloudPowerSave Box e interage com os medidores, chamado módulo Bundler, é responsável por se comunicar com os medidores Powersave via porta USB, obter as medidas de potência feitas pelos mesmos, e enviá-las para o *Web Service*. Ele lê de um arquivo de texto de configuração diversos parâmetros de execução, que serão explicados no final deste capítulo.

Além disso, na versão corrente do sistema, o arquivo de configuração também contém os dados do usuário a serem inseridos no banco de dados do sistema, como por exemplo seu nome, *nickname* e senha. Um trabalho futuro será permitir o cadastramento destes dados através de uma página *web*, ao invés de usar este arquivo.

Este programa não possui interface gráfica. Após ligar a CloudPowerSave Box, ele executa até que ela seja desligada.

Como explicado no capítulo 2, os medidores Powersave suportam uma série de comandos via porta USB, por onde também enviam a resposta. Para permitir a comunicação do programa os medidores, várias classes Java foram desenvolvidas. Também foi necessário criar uma classe para invocar os serviços do *Web Service*. A seção seguinte explica as principais classes criadas.

5.1 Principais Classes

5.1.1 A Classe SerialCom

No Linux, os medidores podem ser tratados como arquivos a serem escritos para o envio de comandos, e lidos para o recebimento das respostas. Ao conectar os medidores no computador, os arquivos correspondentes podem ser encontrados na pasta “dev”, e possuem nomes no formato “ttyACM0”, “ttyACM1”, “ttyACM2”, e assim por diante. É importante notar que, para que a leitura e a escrita nestes

arquivos funcionem corretamente, eles devem ter as permissões adequadas de leitura e escrita no sistema de arquivos.

A classe `SerialCom` é responsável pela leitura e escrita dos arquivos referentes aos medidores. As escritas são sempre de uma linha, correspondendo aos comandos suportados pelos medidores Powersave. As respostas dos medidores podem ter um número arbitrário de linhas.

5.1.2 A Classe `PowerSaveMeterDriver`

Esta classe representa um único medidor Powersave, e guarda seu identificador, o caminho de seu arquivo no sistema de arquivos e uma lista com os comandos suportados pelo medidor. Para enviar um comando e obter a resposta, a classe `SerialCom` é utilizada. A classe `PowerSaveMeterDriver` possui métodos para tratar as respostas de cada comando.

A seguir são mostrados os comandos dos medidores Powersave e suas respectivas descrições. Alguns comandos necessitam de parâmetros, outros não. Por questões de propriedade intelectual referente a empresa Powersave, os formatos dos comandos e respostas não são mostrados.

- Medição em tempo real

Responde a medição em tempo real, incluindo data/hora, consumo, potência, fator de potência, tensão, corrente e frequência. Este é o comando usado pelo programa para obter as medidas de potência.

- Ligar energia

Liga a energia do medidor.

- Desligar energia

Corta a energia do medidor.

- Zerar contador de consumo

Zera o contador de consumo.

- Ler *string* de identificação

Retorna a *string* de identificação do medidor.

- Escrever *string* de identificação

Escreve a *string* de identificação do medidor com a *string* passada por parâmetro.

- Reiniciar o medidor

Reinicia o medidor e as informações armazenadas.

- Acessar memória

O medidor constantemente armazena medidas em sua memória interna de forma automática, mas o programa não usa estas medidas. Este comando responde todas estas medições previamente armazenadas em memória.

- Apagar memória

Apaga as medições armazenadas em memória.

- Modificar intervalo de gravação

Altera o intervalo de tempo com que as medidas automáticas são feitas e armazenadas em memória. O novo intervalo é passado por parâmetro, as opções são 5 ou 15 minutos.

- Atualizar data/hora

Atualiza a data/hora do medidor com a data/hora passada por parâmetro.

A classe `PowerSaveMeterDriver` implementa a interface `MeterDriver`, criada com o intuito de suportar outros tipos de medidores semelhantes no futuro, que podem implementar comandos diferentes.

5.1.3 A Classe `RealSystemDriver`

Esta classe é responsável por detectar os medidores conectados ao sistema, instanciando um objeto da classe `PowerSaveMeterDriver` para cada um deles e mantendo uma lista contendo estes objetos. Ela possui ainda um método para verificar se houve alteração nos medidores conectados, como um medidor sendo desconectado ou um novo medidor sendo conectado.

Ela implementa a interface `SystemDriver`, criada com o intuito de suportar outros sistemas operacionais, e também uma versão simulada do sistema. Estas outras versões, porém, foram descontinuadas no decorrer do desenvolvimento do projeto, mas podem ser retomadas de acordo com a necessidade.

5.1.4 A Classe `ServiceProvider`

Esta classe é usada tanto pelo módulo `Mobile` quanto pela aplicação `Android`, e seu intuito é se comunicar com o *Web Service* através de uma conexão `HTTP`. Para cada método existente no *Web Service*, ela possui um método próprio, que envia os parâmetros por `URL` e trata a resposta `SOAP` do *Web Service*. Os métodos do *Web Service* foram explicados no capítulo 4.

5.2 Funcionamento do Módulo

Inicialmente, o programa lê um arquivo de configuração que definirá diversos parâmetros para sua execução. O programa é executado a partir de um arquivo JAR (Java Archive) automaticamente quando o CloudPowerSave Box é ligada. O arquivo de configuração se localiza na mesma pasta do arquivo JAR, e deve ter o nome “config.txt”. Seu formato será exibido no fim do capítulo.

Como explicado, na versão corrente do sistema, este módulo também faz o cadastramento dos dados do usuário, usando serviços do *Web Service* para criar as entidades necessárias no banco de dados. Estes dados também serão lidos do arquivo de configuração, e são os seguintes:

- *nickname*, nome e senha do usuário
- nome dos Meter Groups que o usuário deseja criar
- nome dos Meters presentes em cada Meter Group, juntamente com o *id* do medidor representado pelo Meter em questão

Caso os dados do usuário já estejam cadastrados, existe uma linha no arquivo de configuração que define que este cadastramento não precisa ser feito.

Como já foi mencionado no capítulo 2, será desenvolvida uma página *web* para agregar esta funcionalidade de cadastro, que deixará de ser feita pelo módulo Bundler.

Após ler o arquivo de configuração e, se necessário, criar as entidades, o programa cria duas *threads* que funcionarão indefinidamente, chamadas de Getter Thread e Sender Thread.

A tarefa da Getter Thread é se comunicar com os medidores Powersave e obter as medidas de potência. Para cada medidor, ela envia um comando de obter medida em tempo real. Após obter a resposta e extrair a medida de potência, a *thread* espera por um intervalo de tempo referente à taxa de amostragem (definida no arquivo de configuração) e repete o processo de obter uma medida de potência. Quando ela obtém de cada medidor um número de medidas igual ao tamanho máximo de um Meter Data Bundle (definido pelo arquivo de configuração), ela cria um novo Meter Data Bundle e armazena em uma lista de Meter Data Bundles a serem enviados. Finalmente, a *thread* dorme por um tempo definido pelo arquivo de configuração, para depois acordar novamente e repetir o processo.

A lista de Meter Data Bundles a serem enviados possui um tamanho máximo definido pelo arquivo de configuração. Caso esse tamanho máximo seja atingido, novos Meter Data Bundles que forem criados serão perdidos. A lista de Meter Data Bundles só esvazia conforme os mesmos são enviados pela Sender Thread.

A tarefa da Sender Thread é enviar para o *Web Service* os Meter Data Bundles presentes na lista, e caso sejam enviados com sucesso, removê-los da lista. É importante lembrar que, além das medidas, um Meter Data Bundle também armazena a data/hora inicial (data/hora da primeira medida), a taxa de amostragem e o *id* do medidor que fez as medidas. Estes dados são parâmetros do método *createMeterDataBundle*, usado para enviar um Meter Data Bundle para o *Web Service*. Esta *thread* também dorme por um tempo definido no arquivo de configuração, para depois acordar e repetir o processo.

5.2.1 O Arquivo de Configuração

O arquivo de configuração define as entidades a serem criadas no banco de dados do sistema (cadastramento de usuário) e parâmetros de execução do programa. Atualmente, este arquivo deve ser preenchido previamente, e colocado na pasta adequada do mini computador Raspberry Pi. Idealmente, este processo deve ser feito por um técnico com as devidas instruções. O arquivo de configuração possui os seguintes campos (os valores entre colchetes devem ser preenchidos):

- Usuário: define os dados do usuário que serão criados no banco de dados. Possui uma linha, no seguinte formato:

```
customer | [nickname] | [nome do usuário] | [senha do usuário]
```

- Meter Groups: define os Meter Groups e Meters que serão criados para o usuário no banco de dados. Possui uma linha para cada Meter Group, incluindo todos os seus Meters e os respectivos identificadores dos medidores correspondentes, no seguinte formato:

```
meterGroup | [nome do Meter Group] | [nome do Meter 1] : [id  
do Meter 1] | [nome do Meter 2] : [id do Meter 2] | ...
```

- Tamanho máximo de Meter Data Bundle: Define o tamanho máximo de um Meter Data Bundle, ou seja, a quantidade máxima de medidas de potência que ele irá conter. Possui uma linha, no seguinte formato:

```
maxBundleSize | [tamanho máximo de um Meter Data Bundle]
```

- Tamanho máximo da lista de Meter Data Bundle: Define o máximo de Meter Data Bundles que serão armazenados, ou seja, o tamanho máximo da lista de Bundles usada pelas *threads* da classe Bundler. Possui uma linha, no seguinte formato:

`maxContainerSize | [tamanho máximo da lista]`

- Taxa de amostragem: Define a taxa de amostragem dos Meter Data Bundles que serão criados, ou seja, o intervalo de tempo entre duas medidas sucessivas de um mesmo Meter Data Bundle. Possui uma linha, no seguinte formato:

`samplingPeriod | [taxa de amostragem (milissegundos)]`

- Tempo de dormir da Getter Thread: Define por quanto tempo a Getter Thread dorme. Possui uma linha, no seguinte formato:

`getterThreadSleepTime | [tempo de dormir (milissegundos)]`

- Tempo de dormir da Sender Thread: Define por quanto tempo a Sender Thread dorme. Possui uma linha, no seguinte formato:

`senderThreadSleepTime | [tempo de dormir (milissegundos)]`

- Executar PreBundler: Define se será feito o cadastramento dos dados do usuário informados no arquivo de configuração. Seu valor será *true* caso os dados precisem ser cadastrados, ou *false* caso os dados já estejam cadastrados. Possui uma linha, em um dos seguintes formatos:

`executePreBundler | true`

`executePreBundler | false`

Abaixo é exibido um exemplo de arquivo de configuração.

```
customer | paulino | Carlos Paulino | paul123
meterGroup | Minha Casa | Med1: METERAAAAAAAAAAAA0 | Med2: METERAAAAAAAAAAAA1
maxBundleSize | 15
maxContainerSize | 100
samplingPeriod | 30000
getterThreadSleepTime | 30000
senderThreadSleepTime | 60000
executePreBundler | true
```

Usando este arquivo de configuração e executando o módulo Bundler, será criado um usuário de *nickname* “paulino”, nome completo “Carlos Paulino” e senha “paul123”. Para este usuário, será criado um único Meter Group de nome “Minha Casa”, contendo dois Meters: “Med1” e “Med2”. “Med1” possui o *id* “METERAAAAAAA0”, “Med2” possui o *id* “METERAAAAAAAAAAAA1”. Um Meter Data Bundle conterà no máximo 15 medidas. A lista de Meter Data Bundle armazenará no máximo 100 Meter Data Bundles. A taxa de amostragem será 30000 milissegundos. Os tempos de dormir da Getter Thread e da Sender Thread serão 30000 e 60000 milissegundos, respectivamente. Os dados do usuário foram cadastrados pois a linha que indica que os dados devem ser cadastrados está com o valor *true*.

Capítulo 6

O Módulo Mobile

O mercado de aparelhos móveis cresce cada vez mais, e junto com ele o número de empresas e desenvolvedores que trabalham com o ramo. Novas ferramentas e serviços que usam dispositivos móveis surgem diariamente.

Dispositivos móveis abrangem uma vasta quantidade de aparelhos: celulares, smartphones, tablets, computadores portáteis com tela sensível ao toque, etc. Os sistemas operacionais mais famosos que executam nestes aparelhos são o Android da Google e o iOS da Apple. O Android é o mais popular, sendo utilizado por diversas empresas em seus dispositivos. Já o iOS é utilizado apenas nos dispositivos da própria Apple, como o iPhone e o iPad.

Um dos principais motivos do sucesso dos dispositivos móveis é a possibilidade de conexão com a *internet*. Outro motivo é a possibilidade de desenvolver novos aplicativos e disponibilizá-los em uma loja de aplicativos. Tanto Android como iOS possuem sua própria loja, com aplicações gratuitas ou pagas que podem ser habilitadas através de *download*. Estes aplicativos adicionam novas funcionalidades ao dispositivo, e podem usar recursos presentes no próprio dispositivo, como câmera, GPS ou Bluetooth. Isto abre a possibilidade de criação de inúmeras aplicações inovadoras, que podem se tornar cada vez mais complexas, uma vez que o poder de processamento dos dispositivos móveis evolui continuamente.

A facilidade de conexão com a *internet* permitiu aos aplicativos móveis tornarem-se clientes de *Web Services*. Este é o caso do módulo Mobile do CloudPowerSave: uma aplicação Android que utiliza *Web Services* para prover suas funcionalidades.

6.1 A Aplicação Android

Esta seção explica as funcionalidades da aplicação Android do CloudPowerSave. Foram desenvolvidas 9 atividades, que correspondem a telas com as quais o usuário interage. A navegação nas atividades é feita tocando os botões na tela, usando o botão retornar do dispositivo móvel para retornar à atividade anterior, ou usando o

botão menu do dispositivo para exibir opções extras. Os serviços usados por estas atividades foram descritos no capítulo 4. As informações de cada atividade são detalhadas a seguir.

6.1.1 Tela Inicial

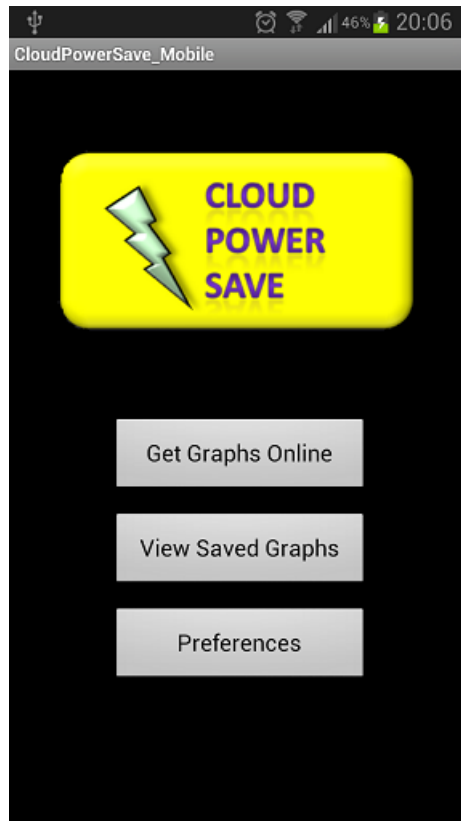


Figura 6.1: Tela inicial.

Esta tela possui 3 botões, “Get Graphs Online”, “View Saved Graphs” e “Preferences”.

O botão “Get Graphs Online” permite ao usuário obter Meter Data Bundles para gerar novos gráficos, e requer conexão com a *internet*. Ao tocá-lo, o usuário é redirecionado para a tela de *login*.

O botão “View Saved Graphs” permite que o usuário visualize ou delete gráficos previamente salvos, sem necessitar de conexão com a *internet*. Ao tocá-lo, o usuário é redirecionado para a tela de seleção de gráfico salvo.

O botão “Preferences” permite ao usuário acessar o menu de preferências da aplicação. Ao tocá-lo, o usuário é redirecionado para a tela de preferências.

6.1.2 Tela de *Login*

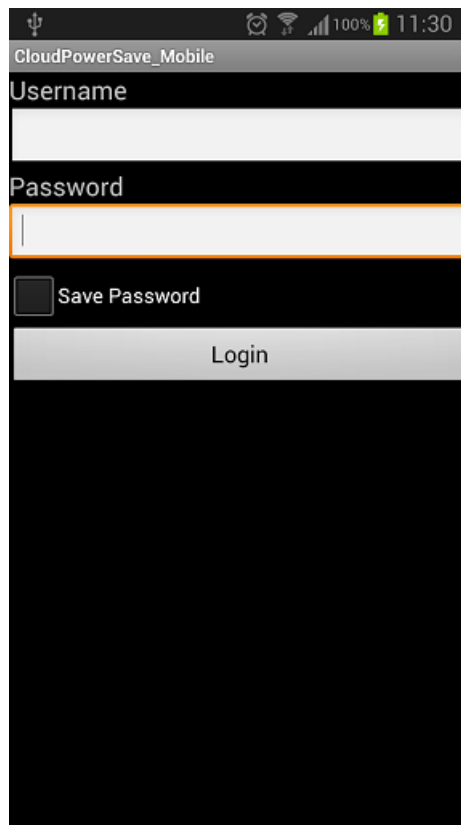


Figura 6.2: Tela de *login*.

Esta tela possui 2 campos de texto, um *checkbox* e um botão, “Login”. O usuário deve digitar seu *nickname* e senha nos campos de texto correspondentes. Caso queira salvar estas informações, ele marca a *checkbox*. Em seguida, o usuário toca no botão e seus dados são enviados para o *Web Service*. Caso seus dados sejam válidos, ele é redirecionado para a tela de seleção de Meter Group. Caso contrário, uma mensagem de erro é exibida.

6.1.3 Tela de Seleção de Meter Group

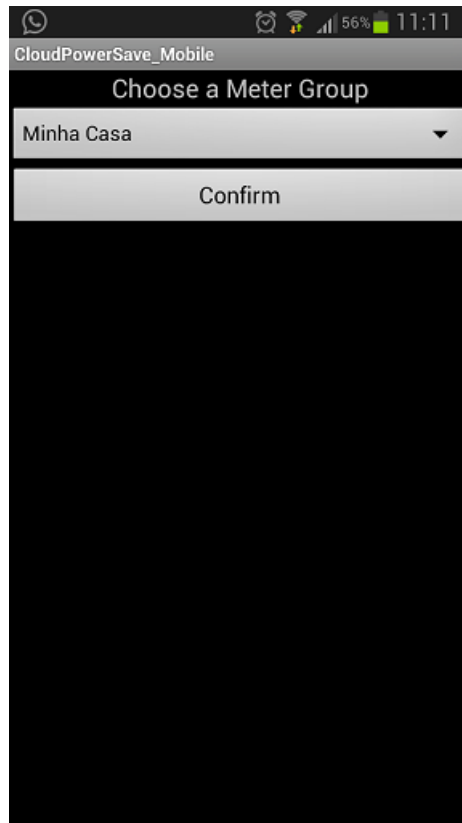


Figura 6.3: Tela de seleção de Meter Group.

Esta tela possui um *spinner* e um botão, “Confirm”. Ao ser carregada, os Meter Groups referentes ao usuário são recuperados a partir do *Web Service*. O usuário deve selecionar no *spinner* o Meter Group do qual deseja obter Meter Data Bundles e tocar no botão. Ele é então redirecionado para a tela de seleção de Meter.

6.1.4 Tela de Seleção de Meter

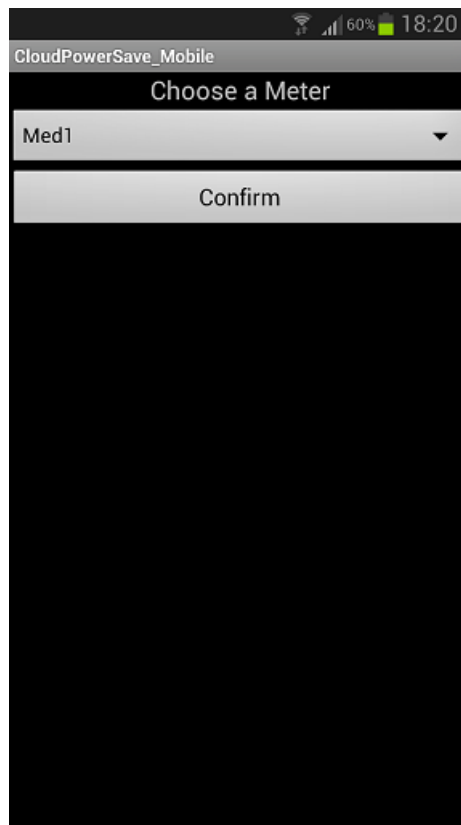


Figura 6.4: Tela de seleção de Meter.

Esta tela possui um *spinner* e um botão, “Confirm”. Ao ser carregada, os medidores (Meters) referentes ao Meter Group previamente selecionado são recuperados a partir do *Web Service*. O usuário deve selecionar no *spinner* o medidor (Meter) do qual deseja obter Meter Data Bundles e tocar no botão. Ele é então redirecionado para a tela de seleção de intervalo de tempo, permitindo a escolha de data/hora inicial e final para a geração de um gráfico.

6.1.5 Tela de Seleção de Intervalo de Tempo

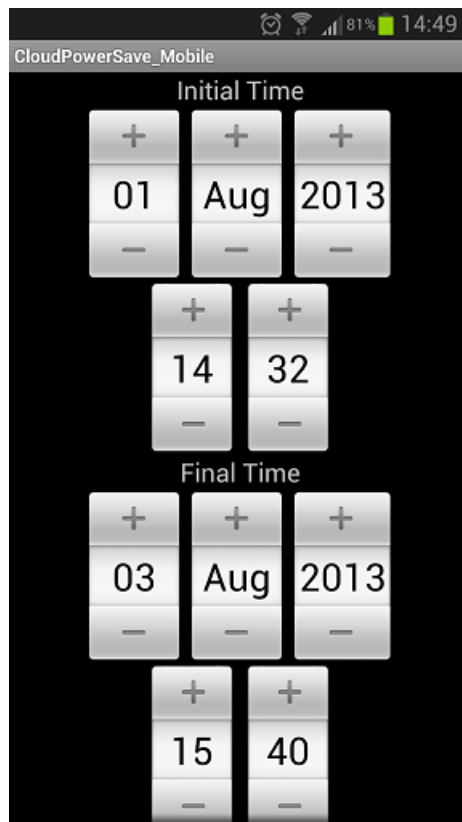


Figura 6.5: Parte superior da tela de seleção de intervalo de tempo.

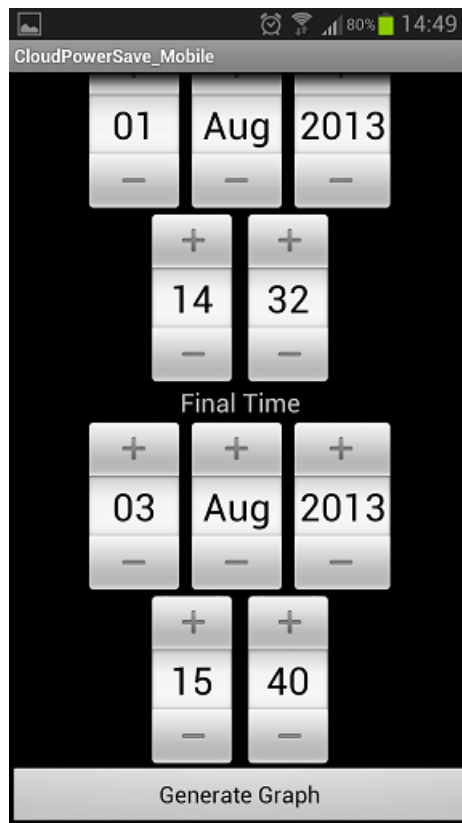


Figura 6.6: Parte inferior da tela de seleção de intervalo de tempo.

Esta tela contém 2 *widgets* de seleção de tempo e um botão, “Generate Graph”.

O *widget* de cima corresponde à data/hora inicial, e o de baixo à data/hora final. O usuário deve inserir neles o intervalo de tempo das medidas que deseja visualizar. Dessa forma, o gráfico resultante incluirá as medidas de Meter Data Bundles que estejam dentro daquele intervalo.

Ao tocar no botão “Generate Graph”, o usuário é redirecionado para a tela de gráfico, desde que existam medidas no intervalo de tempo informado. Caso não existam, uma mensagem de erro é exibida.

6.1.6 Tela de Gráfico

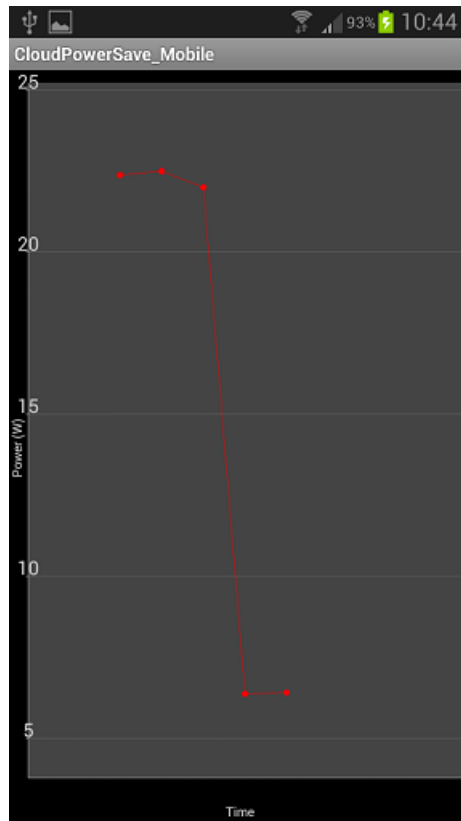


Figura 6.7: Tela de gráfico.

Quando esta tela é carregada, os Meter Data Bundles referentes ao medidor (Meter) selecionado previamente são recuperados a partir do *Web Service*, de acordo com o intervalo de tempo escolhido na tela anterior. Esta tela exibe um gráfico de potência (Watts) por tempo com as medidas presentes nestes Meter Data Bundles que respeitem a data/hora inicial e a data/hora final escolhidas pelo usuário. É possível utilizar os gestos de “expandir” e “contrair” do Android, tanto na horizontal quanto na vertical, para expandir ou contrair o gráfico naquela direção.

Tocando no botão “menu” de seu aparelho Android, o usuário tem a opção “Save”, que permite salvar o gráfico para posterior visualização, sem necessidade de conexão com a *internet*. Esta opção exibe um *dialog box* com um botão de “OK”, que permite ao usuário confirmar a gravação daquele gráfico na memória interna, e um botão de “Cancel”, que cancela a ação.

Ao tocar em um ponto no gráfico, um *dialog box* com os valores da potência e data/hora naquele ponto é exibido, como mostrado na figura 6.8.

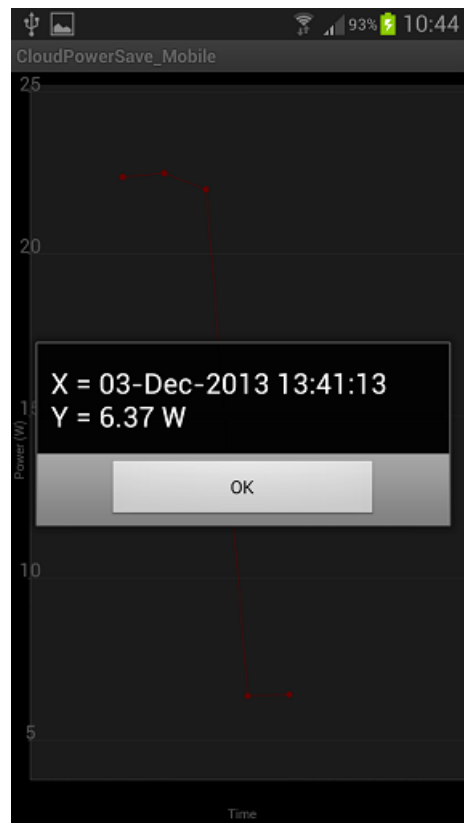


Figura 6.8: Dialog box da tela de gráfico.

6.1.7 Tela de Seleção de Gráfico Salvo

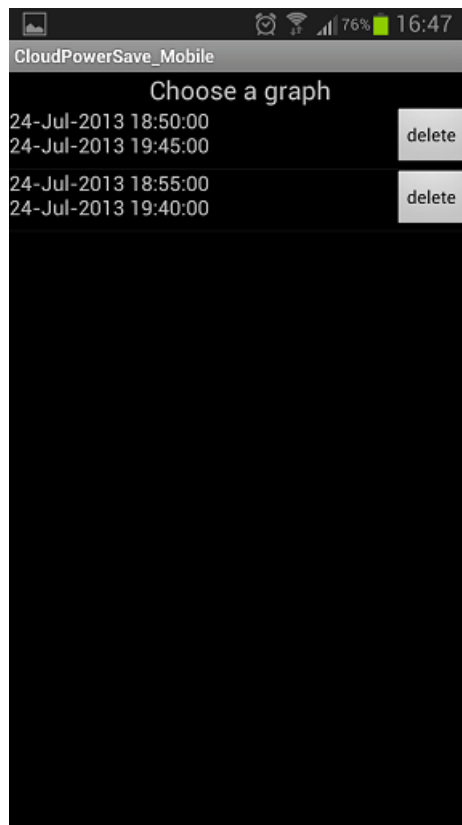


Figura 6.9: Tela de seleção de gráfico salvo.

Esta tela exibe uma lista contendo os gráficos salvos na memória interna. Cada gráfico é identificado pela sua data/hora inicial e data/hora final, representando os tempos de seu primeiro e último ponto respectivamente.

Ao lado do nome do gráfico, existe um botão “delete”, que permite remover o gráfico da memória interna. Ao tocá-lo, um *dialog box* surge para que o usuário confirme a deleção. Caso deseje confirmar, ele toca no botão “OK”, caso contrário, toca no botão “Cancel”.

Para exibir um gráfico salvo, basta tocar no nome do mesmo. O usuário então é redirecionado para a tela de gráfico.

6.1.8 Tela de Preferências

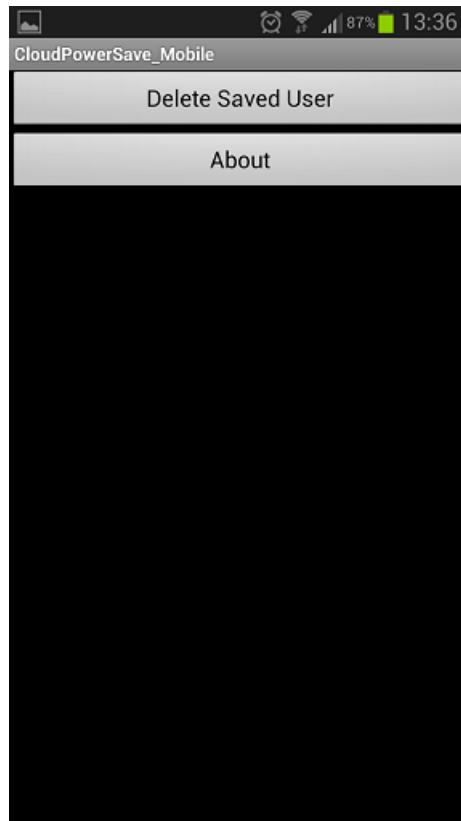


Figura 6.10: Tela de preferências.

Esta tela possui 2 botões, "Delete Saved User" e "About".

O botão "Delete Saved User" permite deletar as informações de *login* salvas na memória interna do dispositivo. Ao tocá-lo, é exibido um *dialog box* com um botão de "OK", que permite ao usuário confirmar a deleção, e um botão de "Cancel", que cancela a ação.

O botão "About" permite acessar as informações sobre a aplicação. Ao tocá-lo, o usuário é redirecionado para a tela de informações da aplicação.

6.1.9 Tela de Informações da Aplicação



Figura 6.11: Tela de informações da aplicação.

Esta tela exhibe informações sobre os criadores da aplicação.

Capítulo 7

Conclusões

Este projeto reportou o projeto e a implementação do sistema CloudPowerSave, que permite monitoramento remoto de dados de energia elétrica. O sistema foi concebido e analisado de forma a permitir sua escalabilidade (uso massivo) e eventual uso empresarial dessa tecnologia.

As tecnologias usadas foram baseadas na arquitetura SOA_MC [6]. Seguindo essa arquitetura, foram desenvolvidos 3 módulos de software - um para interagir com medidores, um para a *cloud* e um para o cliente *mobile*. Para o primeiro, comandado o desenvolvimento de um *hardware* específico, baseado no produto Powersave.

A realização do projeto foi engrandecedora em diversos sentidos, pois houve a oportunidade de desenvolver diversos módulos de um sistema de *software* e lidar com tecnologias novas, o que garantiu experiência para o futuro. Isto é muito importante para um engenheiro em formação, pois o conhecimento adquirido certamente será utilizado em empregos e projetos futuros e pode constituir um importante diferencial. Isto é ainda mais válido para o caso das tecnologias *cloud* e *mobile*, que estão presentes em cada vez mais aplicações.

Capítulo 8

Trabalhos Futuros

A versão inicial do CloudPowerSave desenvolvida para este projeto exibe as principais funcionalidades desejadas para o sistema. Porém, diversas melhorias e novas funcionalidades podem ser implementadas futuramente, e algumas delas serão citadas neste capítulo.

8.1 Medição de Outros Parâmetros de Energia Elétrica

Na versão corrente, o CloudPowerSave está trabalhando apenas com medidas de potência. Além da potência, os medidores Powersave - e portanto o CloudPowerSave Box - permitem a medição de diversos parâmetros de energia elétrica:

- Consumo
- Corrente
- Tensão
- Potência
- Fator de Potência
- Frequência

Além disso, embora a versão corrente do medidor Powersave não tenha a capacidade de medir THD (*Total Harmonic Distortion*), isso será possível em versões futuras do hardware. Quando isso se tornar possível, e o software implementar a obtenção de todas estas medidas, o sistema estará automaticamente preparado para executar o trabalho de monitoramento remoto completo.

8.2 Publicação no Google Play

Pretende-se publicar a aplicação Android do CloudPowerSave no Google Play, a loja de aplicativos da Google para dispositivos Android.

Antes da publicação, é desejável implementar alguns dos trabalhos futuros sugeridos neste capítulo. Serão realizadas discussões com o orientador para definir um momento ideal para publicação.

8.3 Aplicação iOS

Uma interessante expansão para o projeto seria o desenvolvimento de uma aplicação para dispositivos móveis que usem o iOS da Apple. Esta possibilidade será estudada, de acordo com a demanda que pode existir no futuro.

8.4 Página Web

A criação de uma página *web*, com funções semelhantes à aplicação Android, é prevista para depois do término do projeto. Ela seria um segundo cliente para o *Web Service*, e como foi explicado no capítulo 2, isto poderia ser feito sem alterações no *Web Service*, uma vez que o sistema utiliza a arquitetura SOA-MC [6].

A página *web* permitirá, além das funções já disponíveis na aplicação Android, o cadastramento de usuários. Esta é uma funcionalidade importante para que o sistema possa ser usado no futuro.

A criação de uma página *web* permitirá também maior flexibilidade no uso do CloudPowerSave. Um usuário poderá usar a página *web* enquanto estiver em casa, e o seu dispositivo móvel quando estiver fora de casa.

8.5 Cortar a Energia Remotamente

Uma das funções mais importantes e que ainda não foram implementadas é a de cortar a energia remotamente. Os medidores possuem um comando de cortar energia, o que pode ser usado para permitir que um usuário corte remotamente a energia da fase ligada ao medidor pelo seu dispositivo móvel. Isto adicionará um fator de controle ao sistema, e aumentará seu apelo comercial.

8.6 DOUA (*Database Oriented Usecase Authorization*)

Para manter a segurança do sistema, é preciso definir que casos de uso (recursos) do sistema um dado usuário pode acessar. DOUA [19] consiste de uma camada de um sistema que lida com toda esta parte de autorização, guardando a relação entre atores (grupos de usuários que compartilham as mesmas características) e casos de uso (operações ou métodos que um determinado usuário pode executar no sistema). A utilização do DOUA [19] é um trabalho futuro de alta prioridade, pois o sistema precisa ser seguro para ser comercialmente viável.

Referências Bibliográficas

- [1] ROUSE, M. “service-oriented architecture”. . <http://searchsoa.techtarget.com/definition/service-oriented-architecture>. Acessado em agosto de 2013.
- [2] SPROTT, D., WILKES, L. “Understanding Service-Oriented Architecture”. . <http://msdn.microsoft.com/en-us/library/aa480021.aspx>. Acessado em agosto de 2013.
- [3] HE, H. “What is Service-Oriented Architecture”. . <http://www.xml.com/pub/a/ws/2003/09/30/soa.html>, sep. 2003. Acessado em agosto de 2013.
- [4] “SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)”. . <http://www.w3.org/TR/soap12-part1>, apr. 2007. Acessado em agosto de 2013.
- [5] THE APACHE SOFTWARE FOUNDATION. “Apache Axis2 User’s Guide”. . <http://axis.apache.org/axis2/java/core/docs/userguide.html>, apr. 2012. Acessado em agosto de 2013.
- [6] VILLAS-BOAS, S. B. “SOA_MC Service Oriented Architecture - Multiple Client”. . <http://www.sbv.com.br/proposals/SOA-MC-sbv.com.br.pdf>, mar. 2013. Acessado em agosto de 2013.
- [7] POWERSAVE. “powersave”. . <http://powersave.com.br>. Acessado em agosto de 2013.
- [8] VILLAS-BOAS, S. B. “www.CloudPowerSave.com”. . <http://cloudpowersave.com>. Acessado em agosto de 2013.
- [9] THE 4VIEWSOFT COMPANY. “achartengine”. . <https://code.google.com/p/achartengine>. Acessado em agosto de 2013.
- [10] VERSANT CORP. “db4objects”. . <http://www.db4o.com/about/productinformation/db4o>.

- [11] VILLAS-BOAS, S. B. “Scrum Academic”. . <http://www.sbv.com.br/proposals/Scrum-Academic-sbv.com.br.pdf>, apr. 2013. Acessado em agosto de 2013.
- [12] “Apache Tomcat”. . <http://tomcat.apache.org>. Acessado em agosto de 2013.
- [13] “Web Services Architecture”. . <http://www.w3.org/TR/ws-arch>, feb. 2004. Acessado em agosto de 2013.
- [14] THE AXIS DEVELOPMENT TEAM. “Web Services - Axis”. . <http://axis.apache.org/axis>. Acessado em agosto de 2013.
- [15] JAYASINGHE, D. “Apache Axis2 architecture”. . <http://careerride.com/Apache-Axis2-Architecture.aspx>, 2007. Acessado em agosto de 2013.
- [16] “Web Services Description Language (WSDL) 1.1”. . <http://www.w3.org/TR/wsdl>, mar. 2001. Acessado em agosto de 2013.
- [17] ENGLANDER, R. *Java and SOAP*. O’Reilly Media, 2002.
- [18] WIKIPEDIA. “Raspberry Pi”. . http://en.wikipedia.org/wiki/Raspberry_Pi, aug. 2013. Acessado em agosto de 2013.
- [19] VILLAS-BOAS, S. B. “DOUA Database Oriented Usecase Authorization”. . <http://www.sbv.com.br/proposals/DOUA-sbv.com.br.pdf>, mar. 2013. Acessado em dezembro de 2013.