UNIVERSIDADE FEDERAL DE SANTA CATARINA

INTEGRAÇÃO ENTRE TERMÔMETROS E UM SISTEMA ERP VIA WEB SERVICE, COM DISPONIBILIDADE DOS DADOS VIA DISPOSITIVO MÓVEL

Mayco Amorim da Rocha

FLORIANÓPOLIS - SC

2014/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

INTEGRAÇÃO ENTRE TERMÔMETROS E UM SISTEMA ERP VIA WEB SERVICE, COM DISPONIBILIDADE DOS DADOS VIA DISPOSITIVO MÓVEL

Mayco Amorim da Rocha

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Federal de Santa Catarina.
Orientador: Prof. Fernando Augusto da Silva Cruz

Florianópolis – SC

2014/2

Mayco Amorim da Rocha

INTEGRAÇÃO ENTRE TERMÔMETROS E UM SISTEMA ERP VIA WEB SERVICE, COM DISPONIBILIDADE DOS DADOS VIA DISPOSITIVO MÓVEL

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

_____

Prof. Fernando Augusto da Silva Cruz, Dr.
Orientador
Universidade Federal de Santa Catarina

Banca Examinadora:

_____

Prof. João Bosco Mangueira Sobral, Dr.
Membro da Banca
Universidade Federal de Santa Catarina

_____

Prof. Laércio Lima Pilla, PhD.
Membro da Banca
Universidade Federal de Santa Catarina

Agradecimentos

Este trabalho é fruto de muito empenho e dedicação, porém não teria condições de realizá-lo se não tivesse o apoio de pessoas fundamentais, e que vieram em momentos muito importantes.

Agradeço muito a Deus, pelo dom da vida, pela inspiração e a vontade de querer ser cada vez melhor.

Agradeço a minha esposa Nayane, pela compreensão nos momentos de ausência, pelo incentivo nos momentos difíceis, pelos puxões de orelha nos momentos em que relaxei, e por todo o companheirismo ao logo de todos esses anos de faculdade.

Ao meu orientador, Professor Fernando Augusto Cruz. Sem ele este trabalho não teria sido realizado. Ele foi fundamental e me guiou com maestria no desenvolvimento do mesmo.

Agradeço também aos amigos que tive o prazer de conhecer ao longo dessa jornada universitária e que pretendo levar para o resto da vida. Marlon Mafra e Ygor Gasparin. Com vocês dividi momentos de superação, de incentivo e apoio, e claro, de muitas alegrias, principalmente quando estávamos nos bares ao redor da UFSC. Vocês também foram fundamentais na concretização deste trabalho.

Por último e a quem dedico este trabalho, aos meus pais, Laura Amorim e Valdir Rocha (*in memorian*). Eles me trouxeram a este mundo e me deram todo o carinho. Sempre me incentivaram a buscar novos rumos e me tornaram a pessoa que sou hoje. A eles serei eternamente grato.

Pedi força e vigor, Deus me mandou dificuldades para me fazer forte
Pedi sabedoria, Deus me mandou problemas para resolver
Pedi prosperidade, Deus me deu energia e cérebro para trabalhar
Pedi coragem, Deus me mandou situações para superar
Pedi amor, Deus me mandou pessoas com problemas para eu ajudar
Pedi favores, Deus me deu oportunidades
Não recebi nada do que queria,
Mas, recebi tudo o que precisava!

(Oração Conversando com Deus)

RESUMO

A necessidade de monitorar a temperatura de ambientes e equipamentos em um hospital ou qualquer outra organização é um processo muito importante e fundamental para o seu perfeito funcionamento. Nesse processo, é muito importante que o monitoramento seja realizado o mais automaticamente possível, reduzindo assim o tempo de ação e consequentemente evitando um problemas.

Sabemos que no mercado existem diversos sistemas e equipamentos que realizam este monitoramento. Porém a grande maioria trabalha de forma isolada, não havendo integração com os sistemas de gestão.

Analisando esta necessidade, o objetivo deste trabalho é realizar a integração entre termômetros conectados a rede e um sistema ERP para automatizar o processo de controle de temperaturas de equipamentos e ambientes em uma organização de saúde, com disponibilidade de consulta através de um dispositivo móvel.

Palavras-chave:

Web service, termômetro, integração, ERP, aplicativo, dispositivo móvel, android.

ABSTRACT

The need to monitor the temperature of rooms and equipment in a hospital or any other organization is a very important and fundamental to its perfect functioning. In this process, it is very important that the monitoring it performed automatically when possible, thus reducing the time for action and consequently avoiding problems.

We know the market there are various systems and equipment on the market that perform this monitoring. Nevertheless most systems work in isolation, with no integration with the management systems.

In this context, the purpose of this work is the integration between online thermometers and an ERP system to automate the process of temperature control of equipment and environments in a healthcare organization, with availability for consultation via a mobile device.

Keywords:

Web service, thermometer, integration, ERP, application, mobile, android.

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES

APK: Android Package
ERP: Enterprise Resource Planning
HP: Hewlett-Packard
HTTP: Hypertext Transfer Protocol
IBM: International Business Machines
IDE: Integrated Development
MEP: Messages Exchange Patterns
MRP: Manufacturing Resource Planning
SDK: Software Development Kit
SOAP: Simple Object Access Protocol
URI: Uniform Resource Identifier
URL: Uniform Resource Locator
W3C: Word Wide Web Consortium
XML: eXtensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

1.1. Descrição do problema

Os hospitais, assim como outras organizações possuem a necessidade de controlar a temperatura de ambientes e equipamentos, como geladeiras de medicamentos controlados, de bolsas de sangue, câmara fria de alimentos, Data Center, entre outros. O controle da temperatura desses equipamentos e ambientes é um processo muito importante e indispensável para que tudo funcione perfeitamente e que imprevistos sejam evitados.

Para que o monitoramento seja eficaz é necessário um controle periódico de consulta e armazenamento dos dados obtidos em cada instante, fazendo assim com que ações sejam tomadas caso haja alguma divergência do padrão estabelecido, e que se tenha também um histórico das informações obtidas. Geralmente, este controle é realizado de forma manual, onde um colaborador responsável por coletar as informações se desloca até o termômetro e confere periodicamente os dados, preenchendo-os em planilhas ou em um sistema de gestão. Caso o mesmo identifique alguma divergência realiza a comunicação ao responsável para tomar alguma ação para identificar a causa e resolver o problema.

Num modelo ideal, este controle pode ser totalmente automatizado, onde as informações das temperaturas dos equipamentos são enviadas a um servidor que, ao receber as informações, identificam as divergências com base nos padrões e automaticamente insere as informações na base de dados, e consequentemente também comunicam os devidos responsáveis sobre a divergência. Seguindo a tendência do mercado, este monitoramento também pode ser feito através de aplicativos para dispositivos móveis. O uso de um aplicativo reduziria ainda mais o tempo entre a identificação do incidente e a ação a ser realizada para a solução do mesmo.

1.2. Objetivos

1.2.1. Objetivo Geral

Desenvolver um *web service* que irá buscar as temperaturas de termômetros que estarão conectados na rede e atualizar automaticamente os dados da temperatura em um sistema ERP (Enterprise Resource Planning). Desenvolver também um aplicativo na plataforma Android para consulta da temperatura de um determinado equipamento a qualquer momento e de qualquer lugar com a premissa de estar conectado na Internet.

1.2.2. Objetivos Específicos

- Desenvolver um *web service* que irá buscar as temperaturas via protocolo SOAP (Simple Object Access Protocol), dos termômetros que estarão conectados a rede da empresa e atualizar os dados na base do ERP.
- Parametrizar no ERP, os dados dos responsáveis para que o ERP envie um alerta via e-mail e/ou SMS para o devido responsável, em caso de divergências de temperaturas.
- Desenvolver um aplicativo para a plataforma Android que possibilite a consulta da temperatura atual de um determinado equipamento, assim como também a geração de notificações caso a temperatura fique fora de um padrão.

1.3. Delimitação do escopo

O sistema web service não contemplará o desenvolvimento da funcionalidade de envio dos alertas tanto por e-mail como por SMS. Esta customização esta presente no sistema ERP.

1.4. Justificativa

Atualmente, na maioria dos hospitais o controle de temperaturas dos equipamentos e ambientes é totalmente manual. Transformar este monitoramento numa forma automatizada é o desafio. Fazer com que se tenha um histórico das

temperaturas coletadas em um banco de dados para análise, assim como a geração de alertas e/ou notificações em caso de divergências, é o grande benefício de automatizar o processo, pois desta maneira, acreditasse que imprevistos sejam evitados e que tenham uma solução mais rápida.

1.5. Organização do trabalho

Além deste primeiro capítulo introdutório, este trabalho está organizado em mais quatro capítulos. O segundo capítulo traz fundamentação dos conceitos abordados. O terceiro capítulo faz uma visão geral do projeto do sistema desenvolvido, trazendo os requisitos e diagramas. No quarto capítulo é abordada a metodologia de desenvolvimento, assim como o ambiente e ferramentas utilizadas. A implementação do projeto esta descrita no capítulo 5, onde foi dividida em três partes: na primeira temos a descrição da construção do termômetro virtual, o desenvolvimento deste foi necessário, pois a empresa ainda não adquiriu o modelo do termômetro testado na construção do projeto. Na segunda parte temos o projeto do *web service*, que foi chamado de WSTermometro. Este é o responsável por consumir o termômetro, inserir a temperatura na base de dados e disponibilizá-la para o aplicativo. Por fim temos o projeto do aplicativo para dispositivos móveis, denominado de AppTermometro, onde irá consumir o *web service* e exibir na tela as temperaturas de forma textual e gráfica. O sexto capítulo traz as considerações finais e o sétimo e último capitulo apresenta as possibilidades de trabalhos futuros.

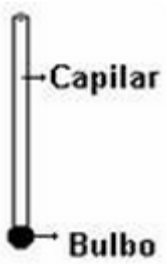
2. FUNDAMENTAÇÃO TEÓRICA

2.1. Medidas de temperaturas

Termômetro é um aparelho composto por uma substância que possua uma propriedade termométrica, ou seja, uma propriedade que varia com a temperatura. É utilizado para medir a temperatura ou as variações de temperaturas. A temperatura é medida através do grau de agitação das moléculas, sendo assim não podemos medi-la diretamente. É necessário estabelecer padrões observando as alterações dos objetos analisados, como os efeitos da dilatação térmica e a resistência elétrica (GONCALVES, 2014).

O termômetro funciona com o princípio de equilíbrio térmico, ou seja, ao ser colocado em contato com um corpo, ao passar do tempo, ele atinge o equilíbrio térmico com esse corpo, fazendo com que a substância termométrica se dilate ou contraia. Quando isso ocorrer ela indicará um valor. Mas para ter esse valor é necessário ter escalas numéricas no Capilar. Por esta razão, os termômetros são feitos baseados em dois pontos de fácil marcação, o Bulbo e o Capilar (COLEGIO WEB, 2014).

Figura 1: Arquitetura de um termômetro



Fonte: Colégio Web (http://www.colegioweb.com.br)

2.1.1. Escalas termométricas

Uma escala termométrica é definida por conjunto de valores atribuídos a um mesmo corpo em diferentes temperaturas, mediante a correspondência de uma equação termométrica ou uma lei de dependência.

2.1.1.1. Celsius

A escala Celsius (°C) é a mais usada na maioria dos países do mundo, foi desenvolvida pelo físico sueco Anders Celsius (1701-1744). Nessa escala, o parâmetro considerado é a água, com 0°C para o ponto de congelamento e 100°C para o ponto de ebulição, observados a uma pressão atmosférica padrão, também chamada de pressão normal.

2.1.1.2. Fahrenheit

Fahrenheit (ºF) é uma escala de temperatura proposta por Daniel Gabriel Fahrenheit em 1724. Esta escala foi utilizada principalmente pelos países colonizados pelos britânicos, mas seu uso atualmente se restringe a poucos países como Estados Unidos e Belize. Nessa escala o parâmetro também é a água, que tem seu ponto de fusão em 32ºF e o ponto de ebulição em 212 ºF.

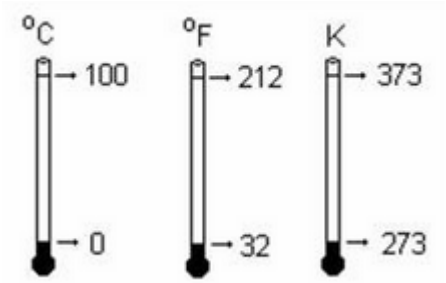
2.1.1.3. Kelvin

Kelvin (K) é o nome da unidade de base do sistema internacional de unidades (SI) para a grandeza temperatura dinâmica. É uma escala muito utilizada na física e química, principalmente utilizada para medir a temperatura absoluta de um objeto. A escala kelvin recebeu este nome em homenagem ao físico e engenheiro irlandês William Thomson, que se tornou o primeiro Lorde

Kelvin quando foi feito par do Reino Unido, que escreveu sobre a necessidade de uma "escala termométrica absoluta".

O zero absoluto, na escala Kelvin, é a temperatura mais baixa que um sistema pode atingir. Nessa temperatura, o hélio se torna líquido.

Figura 2: Escalas termométricas e seus valores



Fonte: Colégio Web (http://www.colegioweb.com.br/trabalhos-escolares/fisica/termometria/termometro.html)

2.2. ERP

Enterprise Resource Planning (ERP), de acordo com (LOPES & PIERRE, 2009) é um conjunto de sistemas que tem como objetivo agregar e estabelecer relações de informação entre todas as áreas de uma empresa. De modo geral, pode ser visto como uma arquitetura de software que facilita o fluxo de informações entre todas as atividades de uma empresa, como fabricação, logística, finanças e recursos humanos. Geralmente possui um único banco de dados interagindo com um conjunto de aplicações através de alguma plataforma comum, possibilitando a automação e armazenamento de todas as informações de negócios.

A história do ERP teve seu inicio na década de 50, quando se iniciou os conceitos modernos de controle tecnológico e gestão corporativa. Naquela época, a tecnologia utilizada era a dos gigantescos *mainframes*, que rodavam os primeiros sistemas de controle de estoques – pioneiro na interseção entre tecnologia e gestão. Na década de 70, com a expansão econômica e a maior disseminação computacional, gerou o antecessor do ERP, o MRP (*Manufacturing Resource Planning*). Estes surgiram já no formato de conjunto de sistemas, também conhecidos como pacotes, que por conversarem entre si, possibilitavam o

planejamento do uso de insumos e a administração das mais diversas etapas dos processos produtivos. Com o inicio das redes de computadores ligadas a servidores na década de 80, e a revolução nas atividades de logística e gerenciamento de produção, o MRP evolui e se transformou no ERP, que agora passou também a controlar outras atividades como mão-de-obra e maquinário, e também as atividades administrativas como finanças, compras, vendas e recursos humanos. A nomenclatura ERP ganhou força nos anos 90, devido principalmente pela evolução das redes de comunicação entre computadores e a disseminação da arquitetura cliente/servidor (ERP, 2014).

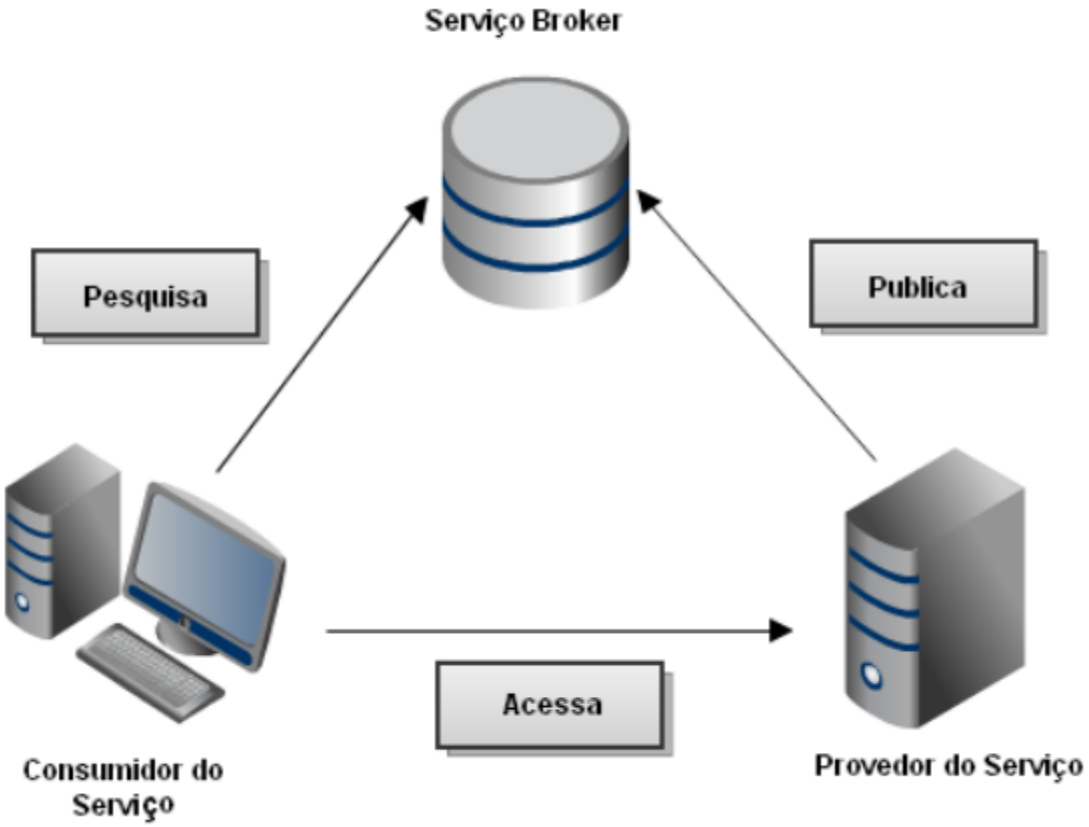
É muito notável que o ERP transformou a forma como as empresas trabalham atualmente. Sua importância foi fundamental para a evolução tanto da forma de gestão, como de produção, logística e todas as outras áreas. O ERP proporcionou às empresas uma maior confiabilidade dos dados, o monitoramento em tempo real, a diminuição do retrabalho e o auxílio à tomada de decisões, entre tantos outros benefícios.

2.3. SOA

A arquitetura orientada a serviços (SOA) é um conjunto de funcionalidades bem definidas em forma de serviços disponibilizados na rede. É importante destacar que esta arquitetura estabelece a implementação de componentes como serviços modulares que podem ser descobertos e usados pelos clientes, além de fornecer um repositório para publicação e descoberta de serviços, possibilitando a transparência de localização (OLIVEIRA, 2012).

A adoção de uma arquitetura orientada a serviços facilita a adaptabilidade de sistemas, permitindo a construção de sistemas altamente dinâmicos na medida em que os serviços podem ser substituídos ou melhorados. SOA promove também a reusabilidade de seus serviços, e permite que tais serviços sejam compostos, formando lógicas de processos mais elaborados, e evitando desperdício de recursos ao mesmo tempo. O baixo acoplamento da arquitetura oferece vantagens como flexibilidade, escalabilidade e tolerância a falhas (OLIVEIRA, 2012).

Figura 3: Modelo básico do SOA



Fonte: OLIVEIRA, 2012

De acordo com ERL (ERL, 2011) os serviços em SOA apresentam as seguintes características:

I. **Contrato de serviço padronizado:** o contrato especifica o que o serviço faz e qual o procedimento para utilizá-lo.
II. **Autonomia e baixo acoplamento:** serviços devem ser autônomos e com baixo acoplamento, ou seja, não devem ter dependências funcionais e/ou técnicas.
III. **Abstração:** serviços devem ocultar detalhes de sua implementação, praticando o conceito de “caixa-preta”, que prega que para uso e reuso de um serviço não é necessário conhecer detalhes além do que é especificado no contrato de serviço.
IV. **Granularidade e Reutilização:** A granularidade do serviço pode ser definida como grossa ou fina, dependendo do nível de detalhe do

serviço. Quanto maior o nível de detalhe, mais fina é a granularidade do serviço.

V. **Compatibilidade:** serviços devem ser capazes de participar como membros efetivos na composição.

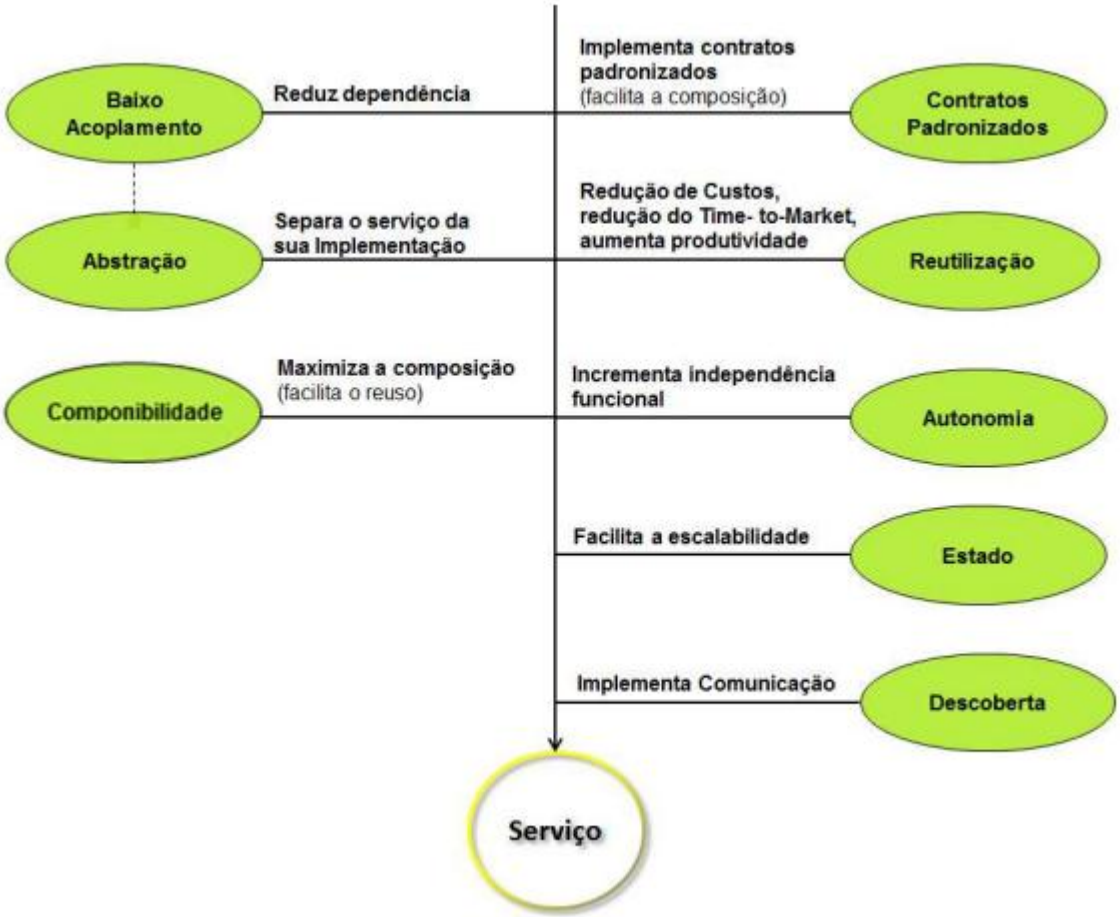
VI. **Estado:** serviços minimizam o armazenamento de informações especificas de uma atividade, visando não comprometer sua disponibilidade e seu potencial de escalabilidade.

VII. **Descoberta:** serviços devem ser facilmente identificados e interpretados à medida que as oportunidades de reuso se apresentam.

VIII. **Interoperabilidade:** a interoperabilidade integra diferentes sistemas por meio dos serviços, independentes dos ambientes em quais tais sistemas operam ou linguagens de programação utilizadas na sua composição.

A figura 4 resume os princípios do SOA.

Figura 4: Princípios da arquitetura SOA



Fonte: KUROIWA, 2011.

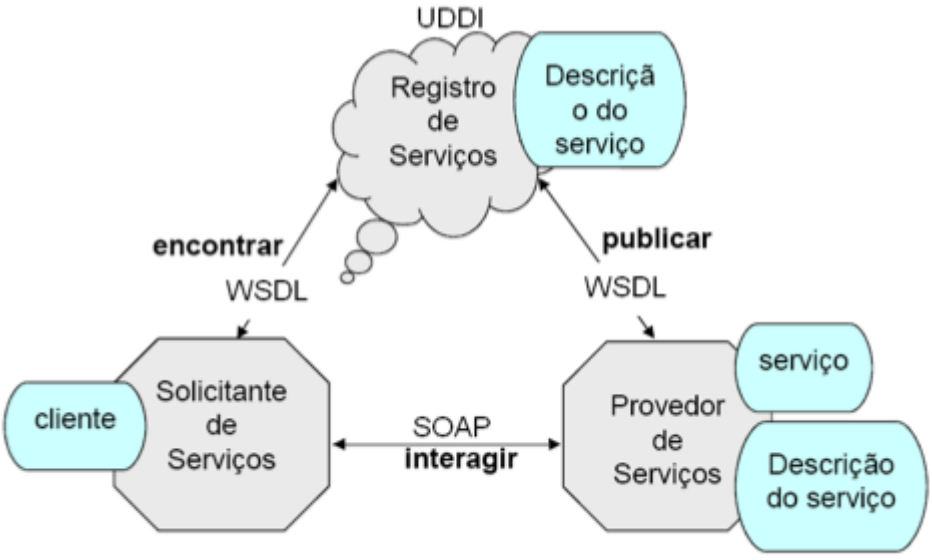
2.4. Web Services

Web services disponibilizam uma maneira de diferentes tipos de aplicação, possivelmente rodando em diversas plataformas e sistemas operacionais interagirem. Originalmente foram criados como uma interface padronizada baseada em tecnologias já conhecidas, basicamente a especificação XML (eXtensible Markup Language) e o protocolo HTTP (Hypertext Transfer Protocol).

Essa padronização foi desenvolvida pela W3C (Word Wide Web Consortium), um consórcio de empresas de tecnologia que tem como o objetivo criar padrões comuns para conteúdo da Web, apoiada por grandes empresas como a Microsoft, IBM (International Business Machines), HP (Hewlett-Packard) entre outras.

A W3C define *web service* como um sistema de software projetado para suportar a interoperabilidade entre máquinas sobre a rede. É uma tecnologia que se constitui de outras tecnologias e é utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Esta integração pode ser realizada entre aplicações desenvolvidas em plataformas distintas, pois *web service* utiliza como padrão o formato XML para a troca de informações.

Figura 5: Arquitetura básica dos Web Services



Fonte: Diego Macedo (http://www.diegomacedo.com.br/web-services/)

2.5. XML

XML (eXtensible Markup Language) é uma linguagem de marcação recomendada pela W3C para a representação de documentos de forma estruturada, independente da plataforma onde é utilizada. Descreve uma classe de objetos de dados chamados documentos XML e parcialmente descreve o comportamento dos programas de computador que os processa. Uma grande vantagem da linguagem XML, além de oferecer interoperabilidade, tanto de plataforma como de linguagem, é que ela não se limita a certo número de tags (marcações), ou seja, você pode criar suas próprias *tags* [VARGAS, 2008].

Um documento XML é composto de marcações (tags) e dentro destas tags é possível à existência de dados e atributos. Os atributos são usados para fornecer uma informação adicional sobre os elementos, e como já dito anteriormente, o conjunto de *tags* é criado pelos usuários conforme suas necessidades e sua semântica advêm de um acordo entre as partes que utilizarão estes dados, visto que a especificação da linguagem não define necessariamente a sintaxe e a semântica a serem utilizadas. Devido a estas características, a linguagem XML é um padrão para troca de dados através da Internet.

2.6. SOAP

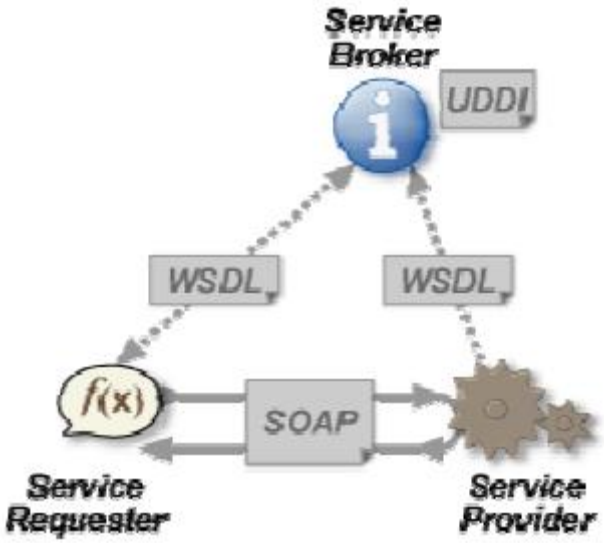
Concebido e mantido pela W3C, SOAP *web services* adotam protocolos formais para comunicação e descrição dos serviços, além de um conjunto de especificações derivadas que padronizam questões como endereçamento, segurança e transacionalidade [GOBBO, 2012].

É um protocolo utilizado em web services para a troca de informações de forma estruturada em uma plataforma descentralizada e distribuída. Seu formato de mensagem é baseado em XML.

O protocolo SOAP possui duas vantagens principais:

- Facilidade de tráfego na rede: suas mensagens trafegam através de requisições HTTP, com cabeçalhos adicionais e conteúdos especiais. A dificuldade de difusão pela Internet foi um grande problema para tecnologias como o CORBA.
- Uso de XML para as mensagens: esta é sem dúvida a vantagem principal. OXML é um formato de dados de baixo acoplamento por excelência (SCHROEDER & SENRA, 2007), por duas razões: é auto descrito e transformável.

Figura 6: Arquitetura de SOAP de acordo com a W3C



Fonte: GOBBO, Leandro. UFSC 2012.

Os dois principais objetivos do projeto para o SOAP, segundo o W3C são simplicidade e extensibilidade. O projeto SOAP procura atingir esses objetivos e omitir, do *framework* de mensagens, características que sejam encontradas com frequência em sistemas distribuídos. Tais características incluem, mas não se limitam a “confiabilidade”, “segurança”, “correlação” e “distribuição”, além das SOAP Messages Exchange Patterns (MEPs), ou em português “Padrão para troca de mensagens SOAP”. Sendo esse último o principal foco do documento de especificação do protocolo SOAP.

2.7. REST

Representational State Transfer ou somente REST se referia, originalmente, a um conjunto de princípios de arquitetura que definem Web Standards como HTTP e URIs (Uniform Resource Identifier) devem ser usados. Atualmente REST é utilizado em um sentido mais amplo para descrever qualquer *interface web* simples que utiliza XML e HTTP, sem as abstrações adicionais dos protocolos que são baseados nos padrões de troca de mensagem, como por exemplo, Web SOAP. Foi definido por Roy T. Fielding em sua tese de PhD. Ele foi um dos principais desenvolvedores de muitos protocolos Web essenciais, incluindo HTTP e URIs.

3. ESPECIFICAÇÃO DO SISTEMA

Este capítulo inicia-se abordando uma visão geral do projeto deste trabalho, além de mostrar os requisitos do projeto.

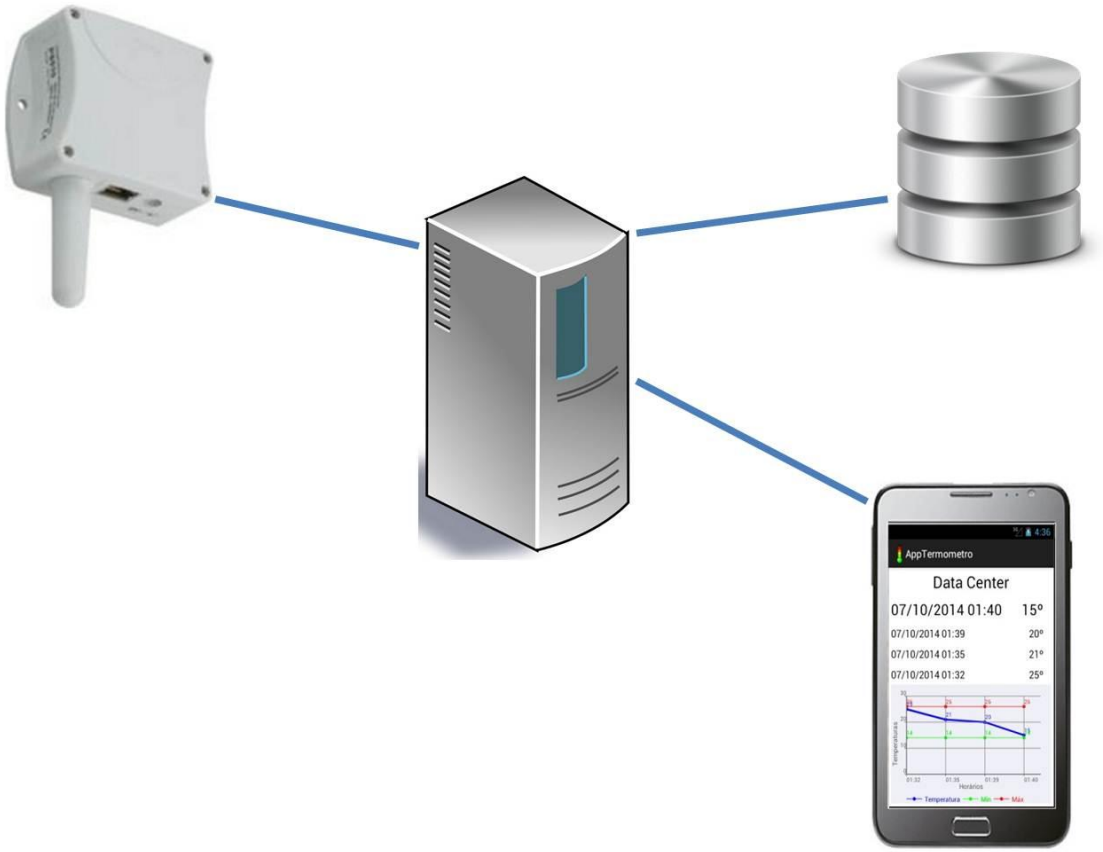
Na seção seguinte temos os diagramas UML do projeto, e por fim uma descrição do formato que o trabalho esta organizado.

3.1. Visão geral

O objetivo deste trabalho é desenvolver um *web service* orientado a serviço, que irá consumir as temperaturas de um termômetro e inseri-las na base de um sistema ERP, além de disponibilizar estas temperaturas para uma aplicação Android.

A figura 7 proporciona uma sião geral do projeto.

Figura 7: Visão geral do projeto



Fonte: Elaborado pelo autor (2014).

Como podemos observar na figura acima, o termômetro é conectado na rede, onde disponibiliza as temperaturas através de SOAP. O *web service* irá obter estas temperaturas, salvar no sistema ERP, e manter um histórico das últimas três (3) temperaturas obtidas para disponibilizá-las para a aplicação Android, além da temperatura atual, que também será exibida na aplicação.

3.2. Requisitos

Nos próximos itens, serão apresentados os requisitos funcionais e não funcionais do projeto. Esta etapa visa buscar todas as informações sobre as funções que o projeto deve executar, além das suas restrições de funcionamento.

3.2.1. Requisitos Funcionais

Web service

RF1. O WS deverá consultar a temperatura atual do termômetro num intervalo de tempo pré-definido. Esta funcionalidade deverá executar desde a inicialização do servidor até seu encerramento. A frequência ideal, conforme informação do setor responsável pela manutenção do ambiente deve ser de 30 minutos.

RF2. A temperatura obtida do termômetro deverá ser inserida na base do sistema ERP, no cadastro do respectivo cadastro, que se refere ao termômetro consultado.

RF3. O WS deverá disponibilizar, via REST, as temperaturas para serem consumidas pelo aplicativo.

Aplicativo

RF4. O aplicativo deve permitir selecionar o termômetro desejado (este trabalho está considerando apenas um termômetro) para consultar a temperatura do mesmo.

RF5. Exibir a temperatura atual, além de exibir as últimas três temperaturas obtidas anteriormente e inseridas na base do ERP, permitindo assim visualizar a variação das temperaturas. Estas temperaturas serão exibidas na forma textual e também na forma de gráfico, facilitando a visualização.

3.2.2. Requisitos Não Funcionais

Web service

RNF1. O WS deverá consumir o termômetro via SOAP

RNF2. O WS deverá disponibilizar as temperaturas para o aplicativo via REST

Aplicativo

RNF3. O aplicativo deve ser de fácil usabilidade.

3.3. Diagramas

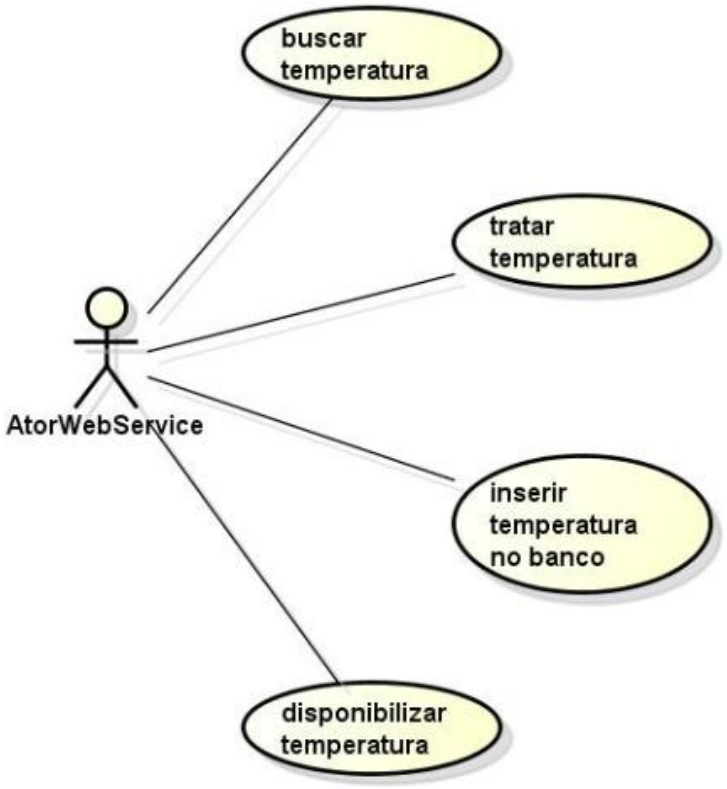
Uma vez definido os objetivos do projeto e seus requisitos, podemos fazer uma análise através da modelagem via UML do sistema.

3.3.1. Diagramas de Casos de Uso

Os diagramas de caso de uso fazem uma modelagem dinâmica do software em alto nível de abstração, modelando o software com os atores (elementos externos) que interagem com o mesmo. (MAFRA & GASPARIN, 2013).

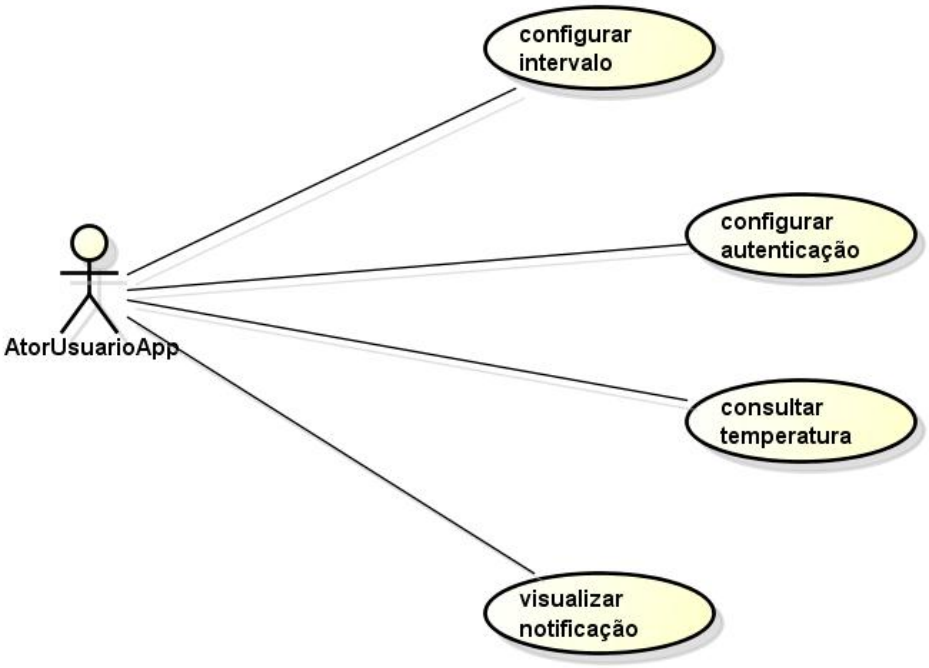
Nas figuras 8 e 9 temos os diagramas de casos de uso do sistema proposto. O primeiro diagrama representa as ações a serem realizadas pelo *web service* a ser desenvolvido. A segunda figura representa o diagrama das ações que serão realizadas pelo autor.

Figura 8: Diagrama de caso de uso do *web service*.



Fonte: Elaborado pelo autor (2014).

Figura 9: Diagrama de caso de uso do aplicativo.



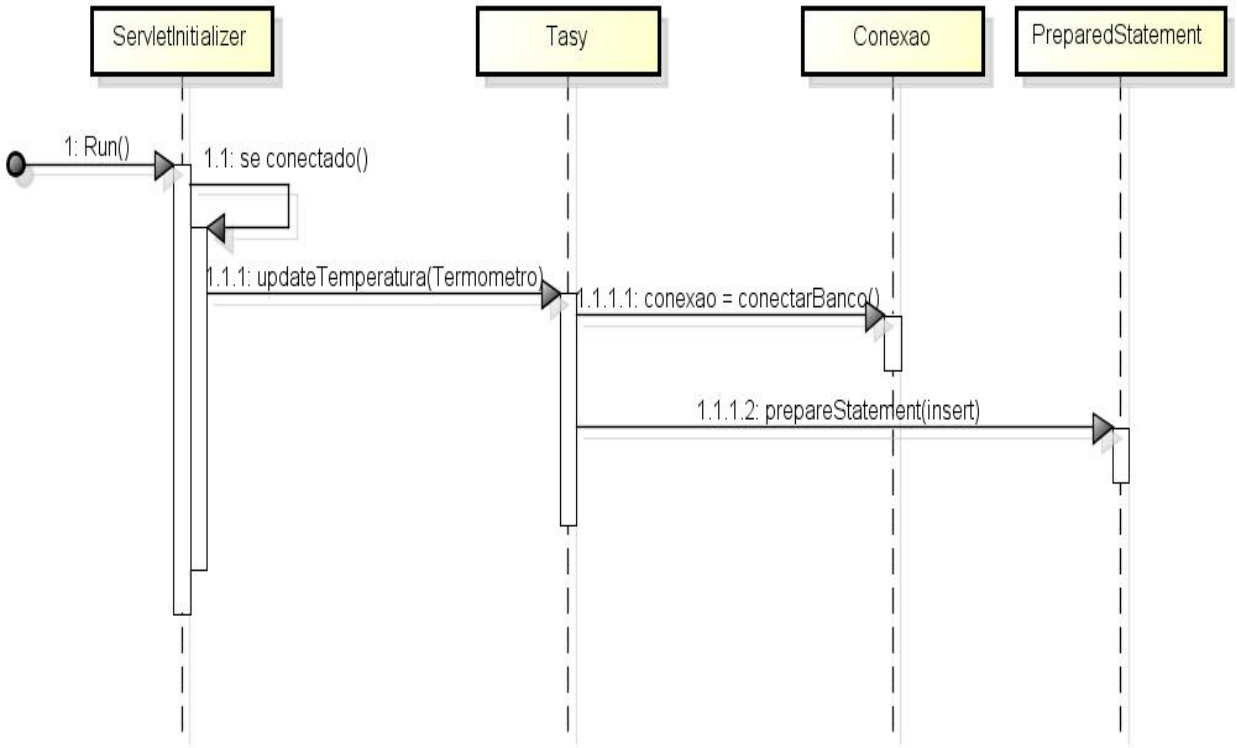
Fonte: Elaborado pelo autor (2014).

3.3.2. Diagramas de Sequência

O diagrama de sequência é um diagrama temporal, que tem por objetivo representar a ordem em que as mensagens são passadas de objeto a objeto. Ele visa facilitar a compreensão de um determinado comportamento do sistema, tendo em vista que este comportamento pode estar distribuído em vários métodos e objetos ao longo do sistema. (MAFRA & GASPARIN, 2013).

Para representar uma determinada funcionalidade do sistema, a figura 10 representa a sequência de atividades da funcionalidade do caso de uso de inserir a temperatura no banco de dados do ERP.

Figura 10: Diagrama de sequência - inserir dados no banco do ERP.



Fonte: Elaborado pelo autor (2014).

4. METODOLOGIA DE DESENVOLVIMENTO

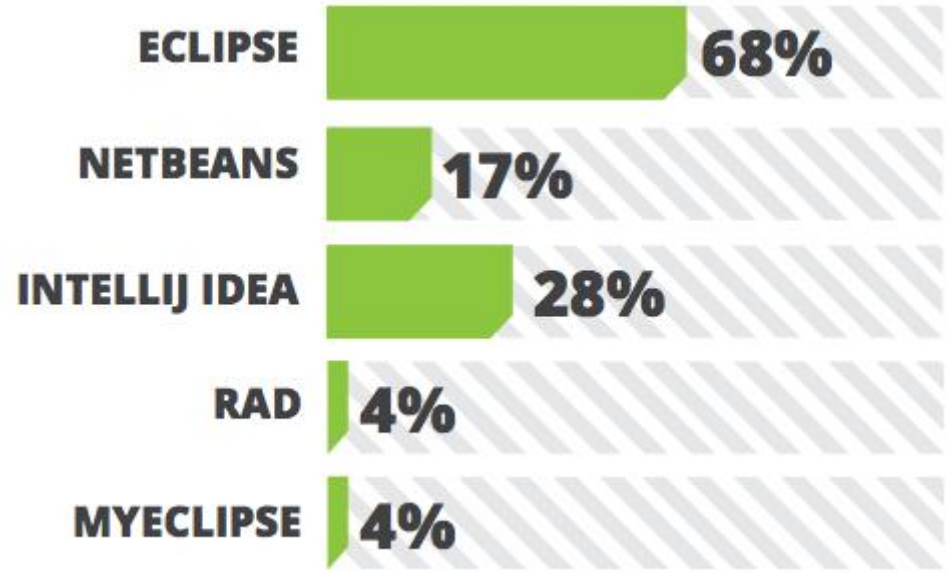
Nesta seção serão apresentadas as tecnologias utilizadas para o desenvolvimento do projeto, tanto para o *web service* como para o aplicativo Android.

4.1. Eclipse

Eclipse é uma IDE (Integrated Development) desenvolvida em Java para o desenvolvimento em Java, porém não exclusivamente. Ela suporta a partir de plugins outras linguagens de programação como C/C++, PHP, Android, Python. Seu desenvolvimento segue o modelo O*pen Source* (código Aberto). O projeto

Eclipse foi originalmente criado pela IBM em novembro de 2001, apoiado por um consórcio de fornecedores de software. Em janeiro de 2004 foi criada a Fundação Eclipse. Uma organização sem fins lucrativo e independente, com o objetivo de administrar a comunidade Eclipse. Hoje a comunidade é mantida por indivíduos e organizações da indústria de software. Atualmente o Eclipse é a IDE Java mais utilizada no mundo (ECLIPSE 2013).

Figura 11: IDEs Java mais utilizadas no mundo



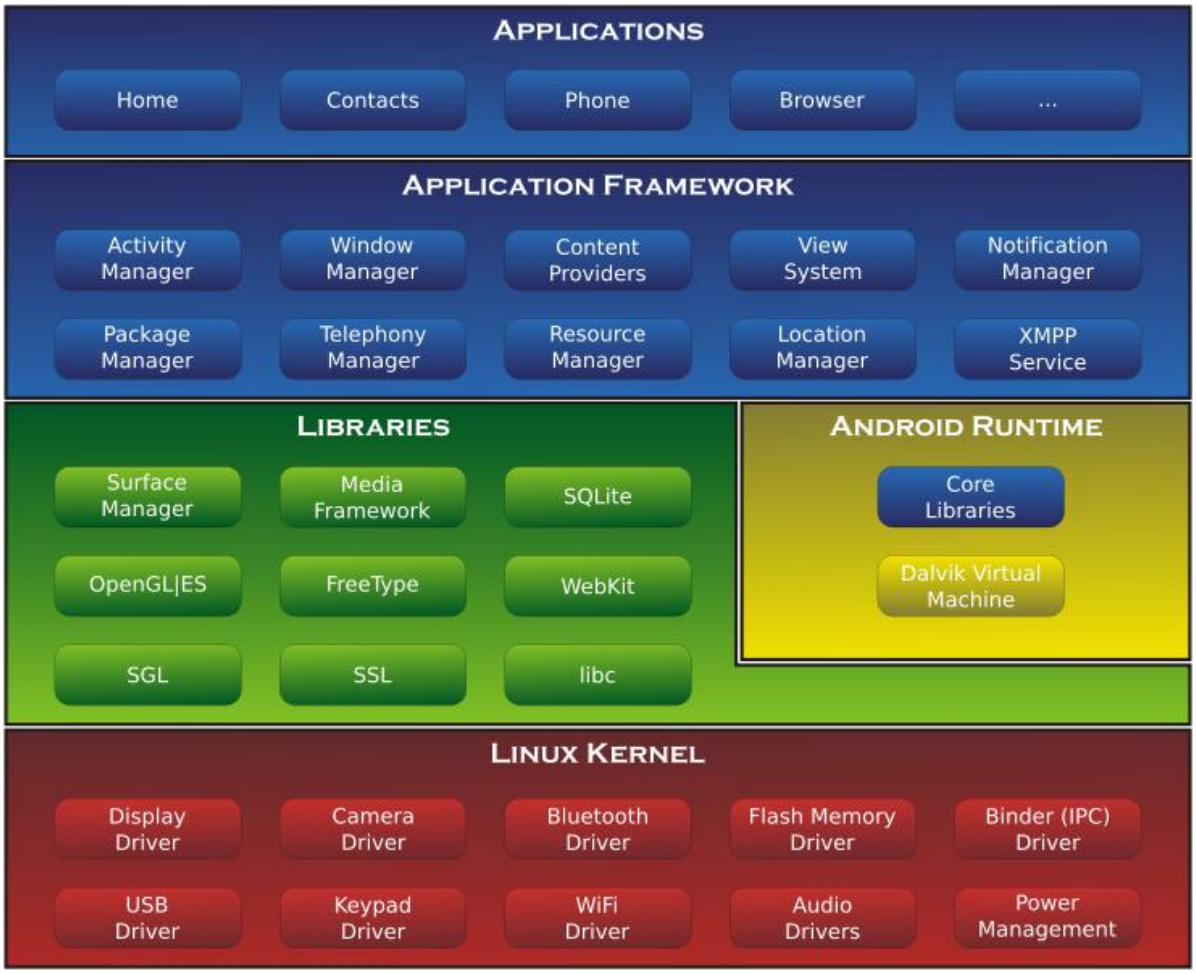
4.2. Java

Java é uma linguagem de programação orientada a objetos lançada em 1995 pela Sun Microsystems. Em 13 de novembro de 2006, a Sun Microsystems lançou a maior parte do Java como Software Livre sob os termos da GNU General Public License (GPL). Em oito de maio de 2007 a Sun finalizou o processo, tornando praticamente todo o código Java como software de código aberto, menos uma pequena porção da qual a Sun não possui copyright.

4.3. Android

Android é um sistema operacional baseado em Linux, mais precisamente sobre o kernel 2.6. Ele é desenvolvido pela Google, juntamente com a Open Handset Alliance (OHA). Esta última formada por um conjunto de empresas. Ele foi criado com o objetivo de ser uma plataforma móvel de código aberto que permitisse ao desenvolvedor total liberdade para tirar o máximo de proveito que os dispositivos têm a oferecer. Por exemplo, uma aplicação pode invocar qualquer uma das principais funcionalidades do dispositivo, como fazer chamadas, enviar mensagens de texto, ou utilizar a câmera, permitindo aos desenvolvedores criar experiências mais ricas e mais coesas para os usuários [MAFRA & GASPARIN, 2013].

Figura 12: Arquitetura do Sistema Operacional Android



Fonte: Wikipédia (http://pt.wikipedia.org/wiki/Ficheiro:Android-System-Architecture.svg)

O Android está disponível como código aberto desde 21 outubro de 2008. A Google publicou todo o código sob a licença Apache. No entanto ele depende de uma autorização do próprio Google para poder acessar a biblioteca de aplicativos, Play Store.

4.4. Plugin ADT

Android Development Tools (ADT) é um plugin para a IDE Eclipse que foi desenvolvido para proporcionar um ambiente integrado para construir aplicativos para Android. Este plugin amplia os recursos do Eclipse, permitindo criar rapidamente novos projetos, criar interfaces de aplicações, adicionar pacotes baseados no framework de API do Android, depurar aplicações, exportar os projetos no formato *apk* (Android Package), entre outras opções.

Para se começar a desenvolver para Android, o caminho mais rápido e fácil é através do plugin ADT para Eclipse, pois ele é altamente recomendado, além de permitir integração com ferramentas, editores de XML, painel de saída para depuração, além de proporcionar uma configuração guiada do projeto.

4.5. SDK ANDROID

SDK (Software Development Kit) Android é um kit disponibilizado pelo Google que contém documentação, código e utilitários para o desenvolvimento de aplicações de acordo com padrões de desenvolvimento para o sistema Android. Os SDKs normalmente são disponibilizados por empresas para que programadores externos possam ter uma melhor integração com o software proposto.

4.6. DISPOSITIVO VIRTUAL (AVD)

O sistema operacional Android roda em uma ampla variedade de dispositivos, com diferentes tamanhos de telas e diferentes capacidade de hardware e recursos. Para que os desenvolvedores possam programar, testar e verificar seus aplicativos em vários dispositivos de versões distintas, os mesmos precisam de uma ferramenta que viabilize este processo de forma simples. O AVD é este dispositivo que irá facilitar os desenvolvedores realizarem estas tarefas. O mesmo é um emulador presente no Android SDK que permite simular um dispositivo móvel no computador. O AVD pode ser visto como um conjunto de

arquivos de configurações que especificam diferentes recursos de hardware e software do dispositivo Android.

4.7. TOMCAT

O Tomcat é um servidor web Java que permite a execução de aplicações para web. Tem como característica principal a linguagem Java, centrada nas tecnologias de *Servlets* e *Java Server Pages*. Neste projeto o Tomcat foi utilizado na publicação do termômetro virtual e na publicação do *web service*.

5. IMPLEMENTAÇÃO

5.1. Termômetro Virtual

Como durante a realização deste trabalho o termômetro avaliado para o desenvolvimento do projeto e para realização da integração com o ERP não foi adquirido devido ao seu valor financeiro, criou-se a necessidade de desenvolver um termômetro virtual que pudesse fazer o papel de um termômetro verdadeiro, e que possibilitasse o desenvolvimento do projeto. Sendo assim, foi desenvolvido um projeto na linguagem Java que simula um termômetro virtual, baseado no mesmo protocolo de comunicação do termômetro original, o protocolo SOAP. Para simular a temperatura, foi utilizado o método *random*, onde são gerados aleatoriamente valores inteiros entre 15 e 30. Assim como no termômetro original, no termômetro virtual, a comunicação para obtenção das temperaturas é feita através do protocolo SOAP.

Figura 13: WSDL do termômetro virtual

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
 xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://p8510/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
 targetNamespace="http://p8510/" name="TermServerImplService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://p8510/" schemaLocation="http://189.90.55.226:3001/P8510/P8510?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="temperatura"/>
  <message name="temperaturaResponse">
    <part name="return" type="tns:termometro"/>
  </message>
  <portType name="TermServer">
    <operation name="temperatura">
      <input wsam:Action="http://p8510/TermServer/temperaturaRequest" message="tns:temperatura"/>
      <output wsam:Action="http://p8510/TermServer/temperaturaResponse" message="tns:temperaturaResponse"/>
    </operation>
  </portType>
  <binding name="TermServerImplPortBinding" type="tns:TermServer">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="temperatura">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal" namespace="http://p8510/"/>
      </input>
      <output>
        <soap:body use="literal" namespace="http://p8510/"/>
      </output>
    </operation>
  </binding>
  <service name="TermServerImplService">
    <port name="TermServerImplPort" binding="tns:TermServerImplPortBinding">
      <soap:address location="http://189.90.55.226:3001/P8510/P8510"/>
    </port>
  </service>
</definitions>
```

Fonte: Elaborado pelo autor (2014).

5.2. WSTermometro

WSTermometro foi o nome dado ao projeto referente ao *web service* desenvolvido. O mesmo foi desenvolvido baseado na arquitetura orientada a serviço (SOA), sendo cliente do termômetro e servidor do aplicativo móvel. Ele possui três atividades principais: a primeira é consumir o termômetro obtendo sua temperatura; a segunda é atualizar a temperatura na base de dados do ERP, e a terceira é disponibilizar estas temperaturas para o aplicativo do dispositivo móvel.

O consumo da temperatura no termômetro ocorre através do protocolo SOAP, sendo que esta atividade ocorre em paralelo, ou seja, assim que o servidor é inicializado ele já começa a consumir o termômetro sem a necessidade de ter nenhuma requisição por parte do usuário, sendo que fará isto numa frequência de tempo pré-determinada. Este processo em paralelo é necessário, pois um dos objetivos é consultar a temperatura constantemente e inseri-la na base do ERP,

independe do consumo do aplicativo. Para que fosse possível executar este processo em paralelo junto ao inicio do servidor, foi necessário criar uma thread que será executada incessantemente.

Figura 14: Inicialização da thread junto com o servidor.

```
public void contextInitialized(ServletContextEvent arg0) {
    ServletContext servletContext = arg0.getServletContext();

    int delay = 1000;
    Timer timer = new Timer();

    timer.scheduleAtFixedRate(new TimerTask() {
        public void run() {
            try {
                if (cliente.conectar()) {
                    TermServerImplService service = new TermServerImplService();
                    TermServer proxy = service.getTermServerImplPort();
                    Termometro termometro = proxy.temperatura();
                    cliente.tratarTemp(termometro, temperaturas);
                    tasy.updateTemperatura(termometro);
                } else {
                    System.out.println("falha na conexão!");
                }
            } catch (ParserConfigurationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (SAXException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InstantiationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }// End of Run
    }, delay, 200000); //30 minutos = 1800000
    servletContext.setAttribute("timer", timer);
}
```

Fonte: elaborado pelo autor (2014).

.A figura 15 destaca um trecho do código do projeto onde ocorre a conexão via SOAP com o termômetro.

Figura 15: Conexão SOAP do WS com o termômetro

```
@WebServiceClient(name = "TermServerImplService", targetNamespace = "http://p8510/",
wsdlLocation = "http://189.90.55.226:3001/P8510/P8510?wsdl")
public class TermServerImplService
    extends Service
{

    private final static URL TERMSERVERIMPLSERVICE_WSDL_LOCATION;
    private final static WebServiceException TERMSERVERIMPLSERVICE_EXCEPTION;
    private final static QName TERMSERVERIMPLSERVICE_QNAME =
            new QName("http://p8510/", "TermServerImplService");

    static {
        URL url = null;
        WebServiceException e = null;
        try {
            url = new URL("http://189.90.55.226:3001/P8510/P8510?wsdl");
        } catch (MalformedURLException ex) {
            e = new WebServiceException(ex);
        }
        TERMSERVERIMPLSERVICE_WSDL_LOCATION = url;
        TERMSERVERIMPLSERVICE_EXCEPTION = e;
    }
```

Fonte: elaborado pelo autor (2014).

Após obter a temperatura, a mesma deve ser inserida na base de dados do ERP, que utiliza o banco de dados Oracle. Para que fosse possível inserir os dados, foi necessário utilizar a classe SQL do Java. A Figura 16 mostra o código de conexão com o banco de dados do ERP.

Figura 16: Conexão com o banco de dados Oracle do ERP

```
public Connection conectarBanco() throws SQLException {

    try {
        Class.forName(JDBC_CLASS);
    } catch (ClassNotFoundException e) {
        throw new SQLException("A classe " + JDBC_CLASS
                + " não foi encontrada no classpath.");
    }

    Connection oracle = DriverManager
            .getConnection("jdbc:oracle:thin:@192.168.5.12:1521:dbprod1",
                    "usuario", "senha");

    return oracle;
}
```

Fonte: elaborado pelo autor (2014).

A Figura 17 mostra o método responsável por inserir os dados na base.

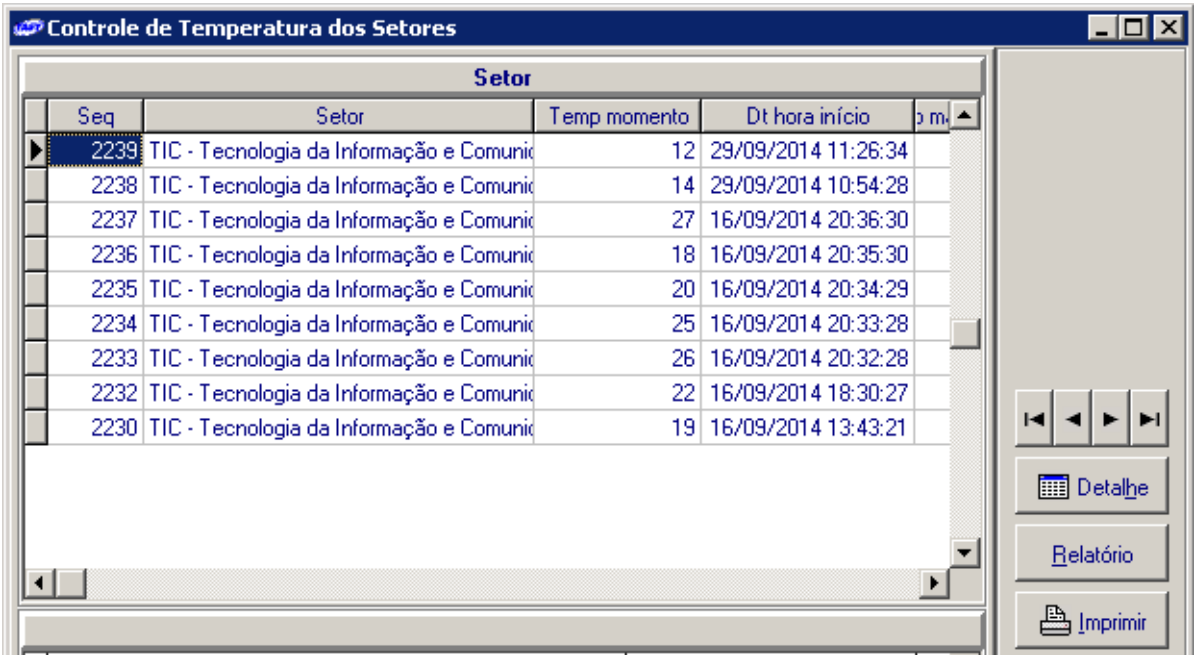
Figura 17: Inserção dos dados no banco de dados do ERP

```
private static final String JDBC_CLASS = "oracle.jdbc.driver.OracleDriver";

public void updateTemperatura(Termometro term) throws SQLException,
        InstantiationException, IllegalAccessException {
    // TODO Auto-generated method stub
    Termometro termometro = term;
    Connection conexao = conectarBanco();


    if (conexao != null) {
        System.out.println("Conectado!");

        String insert = "INSERT INTO SETOR_TEMPERATURA(NR_SEQUENCIA, CD_SETOR_ATENDIMENTO, QT_TEMP_INICIAL,"
                + "DT_HORA_INICIO, DT_ATUALIZACAO, NM_USUARIO)"
                + "VALUES(SETOR_TEMPERATURA_seq.nextval,"+termometro.getId()+","
                + termometro.getTemperature() + ",SYSDATE,SYSDATE,'Mayco')";

        PreparedStatement pstmt = conexao.prepareStatement(insert);
        pstmt.execute(insert);
        conexao.commit();
    }
}
```

Fonte: elaborado pelo autor (2014).

Na imagem da Figura 18, temos a tela da função onde as temperaturas são inseridas no ERP.

Figura 18: Tela ERP - Controle de temperatura



Fonte: Elaborado pelo autor (2014).

Depois que a temperatura é obtida e inserida na base, ela também é salva em variáveis no servidor, pois será utilizada para ser disponibilizada ao aplicativo, e

evitando assim ficar fazendo conexões e consultas na base. O mesmo irá depois obter as três últimas temperaturas salvas juntamente com a temperatura atual do termômetro. As temperaturas serão disponibilizadas para o aplicativo através do protocolo REST, conforme mostra a próxima a Figura 19.

Figura 19: Método que disponibiliza as temperaturas para o App via REST

```
@Path("/getTemp")
public class WSApp {
    private Temperaturas temperaturas = null;
    TempApp tempApp = new TempApp();
    Cliente cliente = new Cliente();

    @GET
    @Path("/{param}")
    @Produces(MediaType.TEXT_XML)
    public TempApp getTemp(@PathParam("param") String id)
            throws MalformedURLException, InstantiationException,
            IllegalAccessException, SQLException, InterruptedException,
            ParserConfigurationException, SAXException {
        temperaturas = Temperaturas.getInstance();
        guardaTempUltimas(temperaturas);

        if (cliente.conectar()) {
            TermServerImplService service = new TermServerImplService();
            TermServer proxy = service.getTermServerImplPort();
            Termometro termometro = proxy.temperatura();
            guardaTempAgora(termometro);
        } else {
            System.out.println("falha na conexão com o termometro!");
        }

        if (tempApp == null)
            throw new RuntimeException("Temperaturas not found");
        return tempApp;
    }
```

Fonte: elaborado pelo autor (2014).

5.2.1. Segurança do *web service*

Os web services são considerados um tanto quanto frágeis em relação à segurança, devido principalmente, ao fato de expor suas funcionalidades através de uma arquitetura orientada a serviços que é bem mais aberta em virtude de sua característica distribuída e de sua natureza heterogênea quanto às plataformas de execução.

Pensando neste quesito de segurança em relação ao *web service*, foi desenvolvido neste projeto um formato de autenticação do aplicativo no servidor.

Foi gerado um *token* no servidor, utilizando a classe *Filter* do Java, e o aplicativo somente consegue consumir o mesmo, caso tenha configurado a senha deste *token* no menu de configurações do aplicativo. Caso esta informação esteja correta, as informações serão obtidas com sucesso. Este filtro funciona como uma barreira de acesso direto ao servidor. Sendo somente possível passar por esta barreira, caso a senha do *token* gerado esteja correta.

5.3. APPTERMOMETRO

Nesta seção será apresentado o desenvolvimento do aplicativo para dispositivos móveis com o sistema operacional Android. Este projeto foi nomeado de AppTermometro e permite a conexão com o *web service* obtendo a temperatura atual do termômetro e as últimas três temperaturas inseridas na base.

5.3.1. Android Manifest

O Eclipse automaticamente gera o arquivo *androidmanifest.xml* no diretório raiz do projeto. Este é o principal arquivo do projeto e contém informações essenciais sobre o aplicativo, como, por exemplo, a versão mínima da API Android, nome do pacote Java, as permissões do aplicativo, o nome das classes de cada Activity que o aplicativo possui, entre outras informações.

Figura 20: Arquivo AndroidManifest.xml

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

    <!-- android:versionName="1.0" -->

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Termometros" >
        </activity>
    </application>

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

</manifest>
```

Fonte: elaborado pelo autor (2014).

5.3.2. Activity

Todo aplicativo Android começa por uma Activity. Uma Activity é basicamente uma classe gerenciadora de interface com o usuário (UI). Ela é uma das mais importantes classes de uma aplicação Android. É importante ressaltar que a Activity é apenas a tela exibida, sem nenhum layout. Para adicionarmos conteúdo nas Activities, geralmente devemos utilizar os arquivos XML de layout do Android, nos quais definimos os elementos visuais como, botões e imagens.

No projeto AppTermometro foram criadas duas Activities. A primeira é a *MainActivity*, que é a tela inicial do aplicativo. Nela o usuário irá selecionar o dispositivo que deseja consultar a temperatura.

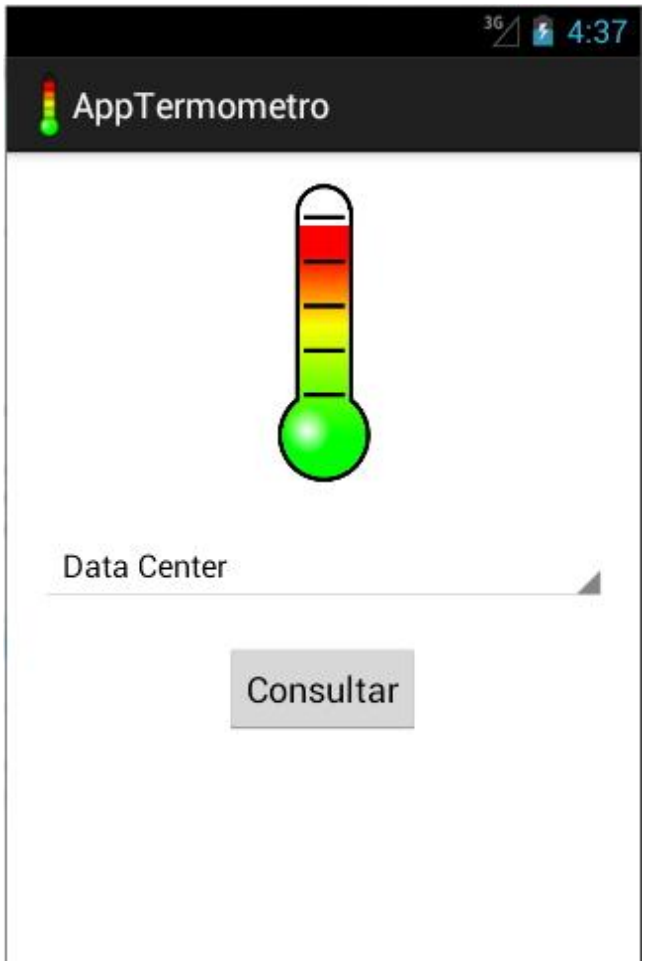
Figura 21: Activity principal - MainActivity

```
@TargetApi(Build.VERSION_CODES.GINGERBREAD)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder()
            .permitAll().build();
    StrictMode.setThreadPolicy(policy);

    setContentView(R.layout.activity_main);

    spinner = (Spinner) findViewById(R.id.dispositivos);
    btnConsultar = (Button) findViewById(R.id.btn_consultar);
    btnConsultar.setOnClickListener(this);

    list.add("Data Center");

    ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, list);

    dataAdapter
            .setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

    spinner.setAdapter(dataAdapter);

    spinner.setOnItemSelectedListener(this);
}
```

Fonte: elaborado pelo autor (2014).

A Figura 22 mostra a tela com o layout da Activity principal do aplicativo.

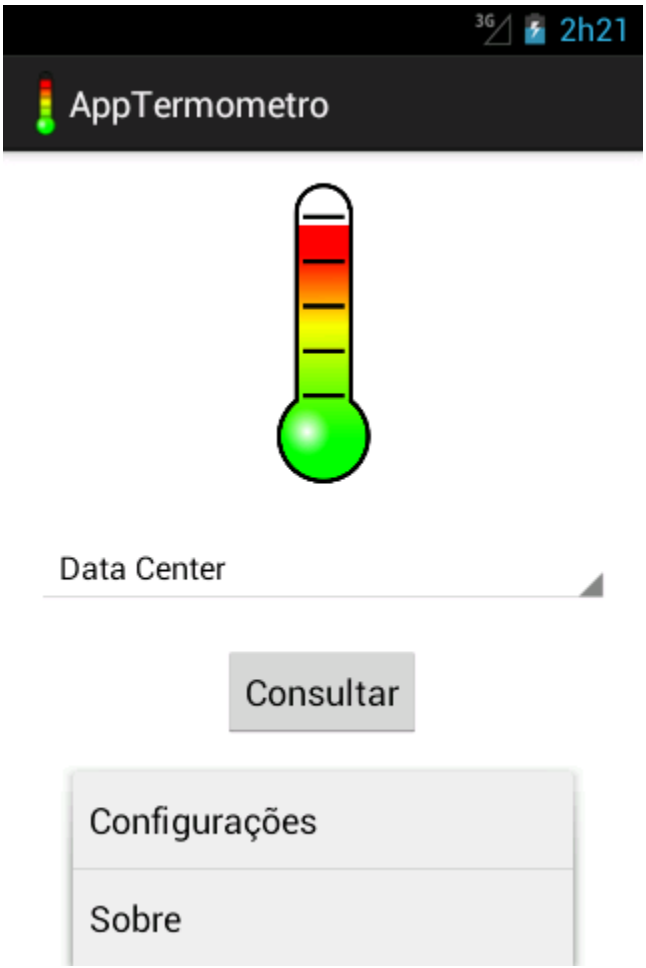
Figura 22: Activity principal – Layout



Fonte: elaborado pelo autor (2014).

Ainda na Activity principal, também foi desenvolvido um menu com duas opções:

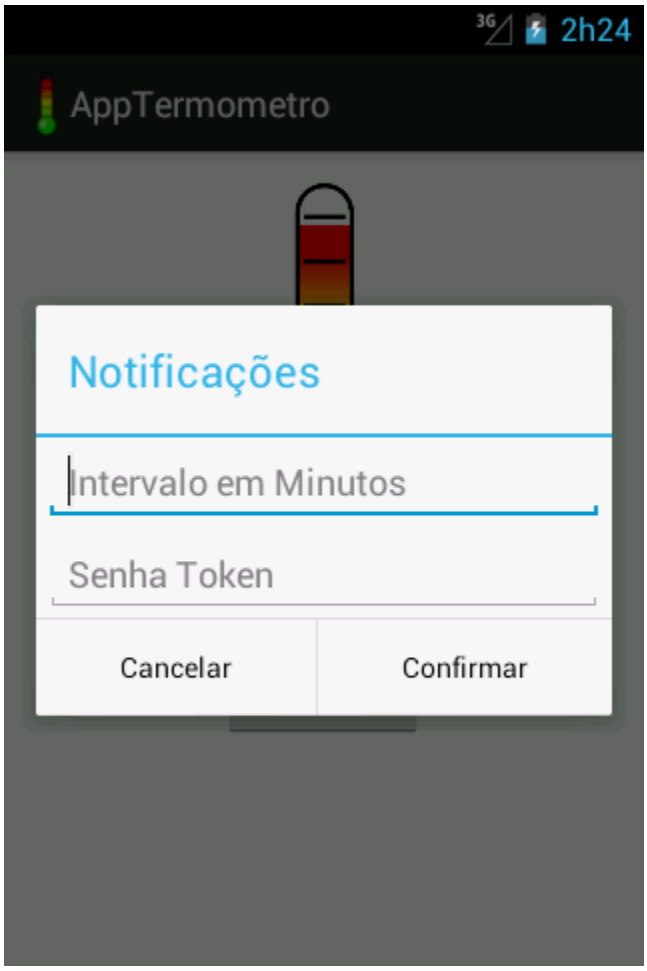
Figura 23: Menu do aplicativo



Fonte: elaborado pelo autor (2014).

- Configurações: onde o usuário deverá fornecer informações do tempo em minutos que deseja que o aplicativo fique consumindo o termômetro, e caso identifique uma temperatura fora do padrão, gere uma notificação informando sobre a mesma.

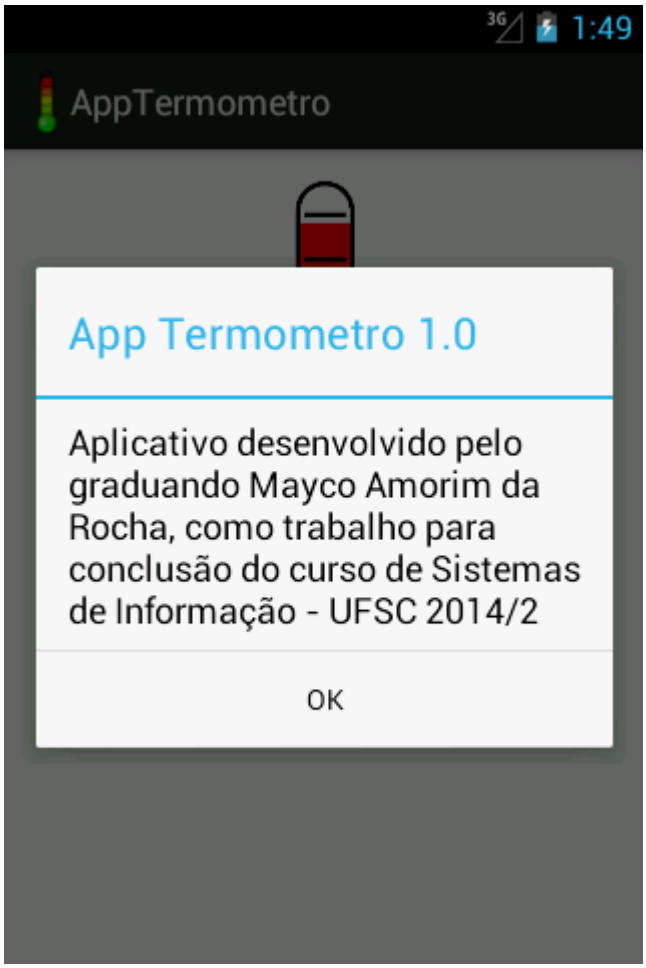
Figura 24: Item menu - Configurações



Fonte: elaborado pelo autor (2014).

- Sobre: neste item foram adicionadas informações referente ao aplicativo, como a versão e o responsável pelo desenvolvimento.

Figura 25: Item menu - Sobre



Fonte: elaborado pelo autor (2014).

A segunda Activity foi chamada de *Termometros*. Nela serão exibidas as temperaturas do dispositivo selecionado anteriormente, tanto de forma textual como de forma gráfica.

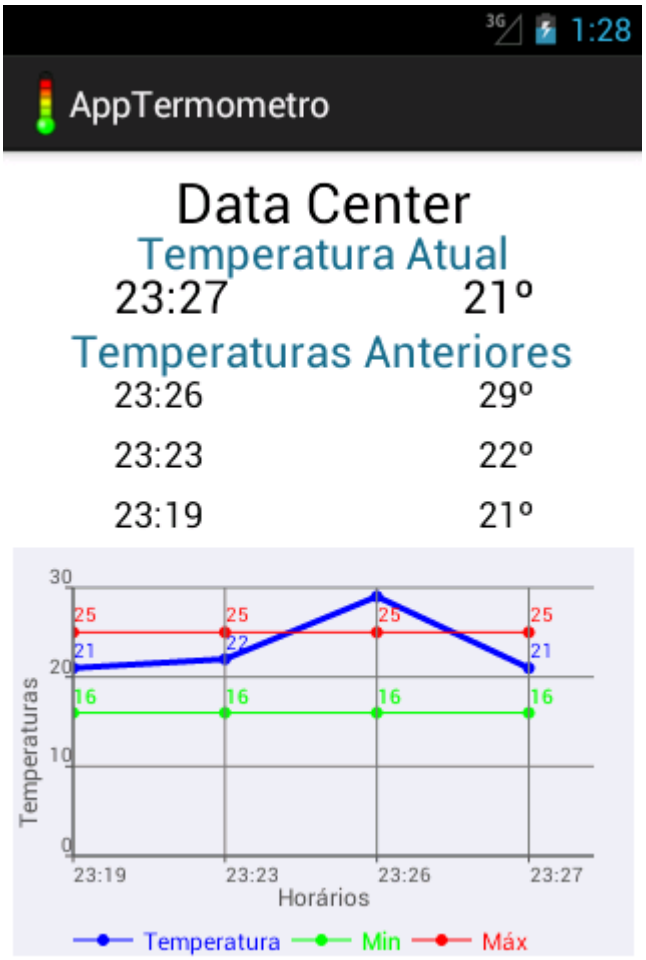
Figura 26: Activity Termometros.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.termometros);

    tvId = (TextView) findViewById(R.id.dispositivo);
    tvDataAgora = (TextView) findViewById(R.id.dataAgora);
    tvData1 = (TextView) findViewById(R.id.data1);
    tvData2 = (TextView) findViewById(R.id.data2);
    tvData3 = (TextView) findViewById(R.id.data3);
    tvTemperaturaAgora = (TextView) findViewById(R.id.temperaturaAgora);
    tvTemperatura1 = (TextView) findViewById(R.id.temperatura1);
    tvTemperatura2 = (TextView) findViewById(R.id.temperatura2);
    tvTemperatura3 = (TextView) findViewById(R.id.temperatura3);

    Intent intent = getIntent();
    Bundle params = intent.getExtras();

    this.temperatura = (Temperatura) params.getSerializable("Termometro");

    salvaLista();
    preencherGrafico();
}
```

Fonte: elaborado pelo autor (2014).

Afigura 27 mostra o layout da tela da segunda Activity do aplicativo, onde as temperaturas são mostradas na tela.

Figura 27: Layout - Activity Termometros.



Fonte: elaborado pelo autor (2014).

5.3.3. Intent

Uma *intent* (intenção) é uma descrição abstrata de uma operação a ser executada. Uma *intent* faz parte da arquitetura do Android e é um conceito que deve ser dominado por todos que desejam programar para Android. Ela pode ser utilizada para iniciar uma Activity, “ativar” um broadcast, enviar uma mensagem para outra aplicação que roda em outro processo, etc. De forma geral, podemos entender que uma *intent* representaria uma “mensagem”, um pedido que é encaminhado ao sistema operacional. O sistema pegará a mensagem, verificará qual é a “intenção da mensagem” e tomará a decisão que pode ser desde abrir uma página web, fazer uma chamada telefônica ou iniciar um Activity.

No AppTermometro, foram criadas algumas I*ntents*, como por exemplo, a chamada de outra Activity ou um chamado de tela flutuante de mensagens, como

as de sem conexão com Internet ou de senha não configurada. A Figura 28 mostra a criação da *intent* principal do aplicativo, na qual sua intenção é chamar a Activity termômetro, a partir da tela inicial, passando como parâmetros as temperaturas obtidas do *web service*.

Figura 28: Declaração e chamada da intent

```
Intent intent = new Intent(this, Termometros.class);
intent.putExtras(params);
startActivity(intent);
this.overridePendingTransition(R.anim.slide_entrando_direita,
        R.anim.slide_saindo_esquerda);
```

Fonte: elaborado pelo autor (2014).

5.3.4. REST

A comunicação do aplicativo com o *web service* é realizada através da RestTemplate. Esta é uma forma simples de comunicação para consumo de web services e é muito recomendada e utilizada na atualidade. Na figura 29 podemos ver a chamada da classe, passando por parâmetro à URL (Uniform Resource Locator) do servidor.

Figura 29: Chamada da classe RestTemplate

```
RestTemplate restTemplate = new RestTemplate();

List<HttpMessageConverter<?>> messageConverters = new ArrayList<HttpMessageConverter<?>>();
messageConverters.add(new SimpleXmlHttpMessageConverter());
restTemplate.setMessageConverters(messageConverters);

restTemplate
        .setRequestFactory(new HttpComponentsClientHttpRequestFactory());
final String url = "http://192.168.0.101:8080/RESTfulExample/rest/getTemp/1";

this.temperatura = restTemplate.getForObject(url, Temperatura.class);
```

Fonte: elaborado pelo autor (2014).

5.3.5. Notificação

Como forma de obter mais informações sobre incidentes e assim ter um tempo de ação mais rápido, foi desenvolvido também notificações no aplicativo. O processo funciona da seguinte forma: o usuário informa no Menu Configurações o intervalo em minutos que deseja que o aplicativo consulte a temperatura. No aplicativo foi criado um service que irá rodar neste intervalo definido, porém irá executar ininterruptamente a partir do boot do sistema operacional Android, ou seja, por mais que o usuário minimize o aplicativo, o service continuará sendo executado em background, consultando a temperatura e verificando se a temperatura está fora do padrão estabelecido, caso esteja fora deste padrão, uma notificação será gerada informando qual foi a temperatura obtida.

Figura 30: Telas da notificação

Fonte: elaborado pelo autor (2014).

6. CONSIDERAÇÕES FINAIS

A realização deste trabalho foi motivada pela necessidade de integrar a um ERP um termômetro que mede a temperatura de um determinado equipamento e/ou setor de uma organização. Através desta integração, seria possível automatizar o processo de controle das temperaturas.

Com este objetivo, o trabalho resultou no desenvolvimento de um *web service* orientado a arquitetura de serviço, que capta a informação da temperatura de um termômetro virtual, também desenvolvido para o projeto. Esta temperatura é inserida na base de dados do ERP, permitindo um maior volume de dados históricos das temperaturas, assim como também facilitando as tarefas dos responsáveis em caso de incidentes, pois caso alguma temperatura fora do padrão seja identificada, o ERP gera automaticamente uma Ordem de Serviço pra o grupo executor responsável, assim como também possibilita o envio de SMS aos responsáveis.

Seguindo a tendência do mercado atual, também foi desenvolvido um aplicativo para dispositivos móveis com o sistema operacional Android. Este aplicativo permite a consulta da temperatura de um determinado termômetro alocado na empresa, com a premissa de possuir acesso a Internet. O aplicativo também possui um processo de notificação, onde a partir de uma frequência de tempo definida pelo usuário, fará consultas ao termômetro e caso também seja identificado uma temperatura fora do padrão, uma notificação será gerada no dispositivo móvel, informando o local do termômetro e a temperatura obtida.

Com base no que foi desenvolvido, o trabalho atingiu os objetivos definidos para o mesmo, além de proporcionar uma maior amplitude dos métodos e ferramentas disponíveis no mercado. Um exemplo a ser destacado, foi a utilização de dois protocolos recomendados pela W3C para web services, o protocolo SOAP e o protocolo REST. O primeiro foi utilizado na comunicação do web service com o termômetro, já o segundo foi utilizado entre o aplicativo e o web service.

Com esta integração, o processo de controle de temperaturas fica mais automatizado, o tempo de resposta a um incidente é reduzido drasticamente, e o

histórico de temperaturas possibilitará uma melhor análise do ambiente que esta sendo monitorado.

7. TRABALHOS FUTUROS

Mesmo atingindo o objetivo inicialmente proposto, surgiram ao longo do desenvolvimento deste trabalho algumas ideias para uma gestão ainda mais eficiente do controle de temperaturas.

Uma delas seria a expansão do controle para outros locais que também necessitam de monitoramento. Como este é um trabalho de caráter tecnológico, abordou apenas o monitoramento de um único local.

Outro detalhe a ser considerado é a segurança para o *web service*, pois pelo fato de estarem expondo informações para troca de mensagens, segurança é um quesito totalmente necessário. Pensando neste quesito, seria ideal implantar outras formas de segurança além da que foi aplicada neste trabalho.

Vislumbrando uma gestão ainda mais eficaz, uma opção a ser considerada e desenvolvida caso seja viável, seria a criação de uma tela que contenha ao mesmo tempo a informação em tempo real de vários ambientes/equipamentos que necessitam de monitoramento. Através desta tela o usuário teria um maior controle de todos os locais, além do fato de a tarefa ser mais simplificada.

Outra opção de trabalhos futuros, seria em relação a tomada de decisões automáticas, onde poderia ser configurado para que uma ação seja realizada em caso de identificação de temperaturas fora do padrão.

REFERÊNCIAS

ACTIVITY. **Android Activity**. Disponível em <http://www.androidnaveia.com.br/2011/02/desenvolvimento-activitys.html>. Acesso em12 de outubro de 2014.

ADT. **Plugin ADT**. Disponível em <http://developer.android.com/tools/sdk/eclipse-adt.html>. Acesso em 06 de outubro de 2014.

ANDROID. **Android**. Disponível em <http://www.ibm.com/developerworks/br/library/os-android-devel/>. Acesso em 31 de outubro de 2013.

ANDROID. **Emulação de Dispositivo Virtual Android para a arquitetura Intel**. Disponível em <https://software.intel.com/pt-br/android/articles/android-virtual-device-emulation-for-intel-architecture?language=es#summary>. Acesso em 07 de outubro de 2014.

COLÉGIO WEB. **Termômetro**. Disponível em<http://www.colegioweb.com.br/trabalhosescolares/fisica/termometria/termometro.html>Acesso em 08 de outubro de 2014.

ECLIPSE. **IDE Eclipse**. Disponível em <http://www.eclipse.org/org/#about>. Acesso em 2 de outubro de 2013.

ERL, Thomas. **Service-Orientation Design Principles**. Disponível em <http://serviceorientation.com/>. Acesso em 26 de outubro de 2014.

GOBBO, Leandro. **Uma proposta de sistema para o registro de informações sobre bovinos via web services visando a rastreabilidade**. Universidade Federal de Santa Catarina, 2012.

GONCALVES, Leila. **Termômetros**. Disponível em <http://www.if.ufrgs.br/~leila/termo.htm>. Acesso em 08 de outubro de 2014.

INTENT. **Android Intent.** Disponível em <http://www.programarandroid.com.br/2013/04/fundamental-trabalhando-com-intent-e.html>. Acesso em 01 de outubro de 2014.

JAVA. **Java**. Disponível em <http://www.java.com/pt_BR/about/>. Acesso em 2 de outubro de 2013.

KUROIWA, Débora Momono Alves. **Reflexões sobre a Arquitetura Orientada a Serviço e o Surgimento de uma Nova Disciplina, a Engenharia de Software de serviço**. Faculdade de Tecnologia de São Paulo, 2011.

LOPES, Juliana Maria; PIERRE, Fabio Roberto. **ERP – Conceito e Evolução**, 2009.

MAFRA, Marlon; GASPARIN, Ygor Henrique. **Uma Aplicação Android para o Portal Minha UFSC**. Universidade Federal de Santa Catarina, 2013.

OLIVEIRA, Ricardo Ramos. **Avaliação de manutenibilidade entre as abordagens de web services RESTful e SOAP-WSDL**. Universidade do Estado de São Paulo, 2012.

REST. **Introdução ao REST**. Disponível em <http://www.infoq.com/br/articles/rest-introduction/>. Acesso em 03 de outubro de 2014.

REST. **RESTful**. Disponível em <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch _style.htm>. Acesso em 01 de outubro de 2014.

SCHROEDER, Douglas; SENRA, Fernando M. **Uma proposta de Integração entre Centros de Saúde e Hospitais Públicos Baseada em Web services**. Universidade Federal de Santa Catarina, 2007.

VARGAS, Roberto Silva. **Usando Web Services para Acesso a Informação de Contexto em um Portal Web**. Universidade Federal de Pelotas, 2008.