FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Análise e Desenvolvimento de Sistemas

Carlos Eduardo Santiago

COMUNICAÇÃO ENTRE UM APLICATIVO ANDROID E UM SISTEMA WEB UTILIZANDO A BIBLIOTECA KSOAP2

Americana, SP

2016

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Análise e Desenvolvimento de Sistemas

Carlos Eduardo Santiago

COMUNICAÇÃO ENTRE UM APLICATIVO ANDROID E UM SISTEMA WEB UTILIZANDO A BIBLIOTECA KSOAP2

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Me. Diógenes de Oliveira.

Área de concentração: Desenvolvimento de Sistemas

Americana, S. P.

2016

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

Santiago, Carlos Eduardo
S226c Comunicação entre um aplicativo Android e um sistema Web utilizando biblioteca KSOAP2. / Carlos Eduardo Santiago. – Americana: 2016.
40f.

Monografia (Graduação em Tecnologia em Análise e Desenvolvimento de Sistemas). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Me. Diógenes de Oliveira

1. Comunicação de dados 2. Android – aplicativo 3. WEB – rede de computadores I. Oliveira, Diógenes de II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.519

Carlos Eduardo Santiago

COMUNICAÇÃO ENTRE UM APLICATIVO ANDROID E SISTEMA WEB UTILIZANDO A BIBLIOTECA KSOAP2

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.

Área de concentração: Análise e Desenvolvimento de Sistemas.

Americana, 23 de junho de 2016.

Banca Examinadora:

Diógenes de Oliveira (Presidente)
Mestre
CEETEPS/Faculdade de Tecnologia – FATEC/ Americana

André de Lima (Membro)
Doutor
CEETEPS/Faculdade de Tecnologia – FATEC/ Americana

Alberto Martins Júnior (Membro)
Mestre
CEETEPS/Faculdade de Tecnologia – FATEC/ Americana

AGRADECIMENTOS

Em primeiro lugar agradeço aos meus queridos pais que sempre me apoiaram, incentivaram nos estudos, amigos que de alguma forma me auxiliaram a desbravar o mundo da programação, ao meu orientador e amigo Mestre Diógenes de Oliveira que me ensinou e mostrou o caminho do conhecimento e todas as pessoas que contribuíram, direta ou indiretamente, para que o seu trabalho tenha sido desenvolvido.

RESUMO

O presente texto conceitua a criação de um aplicativo na plataforma Android que poderá ser executado em versões iguais ou menores que 6.0, baseada em módulos de um sistema web. Neste projeto foi utilizado a linguagem de programação Java a tecnologia *Web Service* utilizando o protocolo *SOAP* e bibliotecas voltadas para o desenvolvimento de aplicativos nativos Android, como a biblioteca *KSOAP2-Android* para a interpretação do *XML* e banco de dados *SQlite*. O projeto EQUINOVET® é um *software* web de gestão desenvolvido, exclusivamente, para o médico veterinário especializado em equinos. Para unificar os dados coletados a partir do aplicativo mobile e garantir a integridade dos dados quando submetidos ao sistema na web, utilizando os recursos de *Web Services* que garantem a comunicação entre sistemas distintos. Este sistema foi projetado com o objetivo de desenvolver uma ferramenta para ser utilizada diretamente no campo, mesmo sem acesso à internet, eliminando papeis e planilhas, com a qual o veterinário irá registrar os atendimentos de forma simples e rápida, assim mantendo todos os dados de trabalho organizados em um único local.

Palavras Chave: *Android;WebService;SOAP*.

ABSTRACT

This paper conceptualizes the creation of an App on Android platform that could be performed in equal or smaller versions that 6.0, based in modules of a Web System. In the project it was used Java programming language and technology using SOAP protocol, and libraries focused on the development of native Android applications with KSOAP2-Android Library to interpretation of XML and also SQLite Data base. EQUINOVET® this project is a management web software developed exclusively for veterinarian specialized in equine.To unify the data colleted from the mobile application and ensure data integrity when subjected to the web system, using the web services features that ensure communication between different system.
This system was projected with the goal in develop a tool to be used directly in the camp even without internet access, eliminating papers and spreadsheets, in 0which the veterinarian will record the attendance simply and quickly, thus keeping all work data organized in one location.

Keywords: *Android;Web Service;SOAP.*

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

Nos últimos anos o mercado de *smartphones*, *tablets* tiveram um crescimento no Brasil, em 2014 cerca de 145,6 milhões de novos usuários. Com essa alta demanda o aumento de processamento de dados, podendo ter até 16 núcleos, com a facilidade de acesso a informação e a rapidez que a rede 4g dispõe, está sendo vantagem ter um sistema, ou módulos de sistemas, na tecnologia *mobile*, trazendo comodidade e facilidades para os usuários acessarem as informações com maior rapidez e segurança (BOUÇAS, 2014).

O Android é um sistema operacional *open source* para dispositivos móveis baseado em Linux desenvolvido pela *Open Handset Alliance* e liderada pelo Google Inc. Em um aparelho com o sistema operacional Android, é possível acessar sites, enviar e-mails, sms, jogar, alguns dos recursos disponíveis são semelhantes ao de um computador.

Como o código do sistema é *open source*, existe uma grande variedade de aplicativos, ou seja, qualquer pessoa que tenha algum conhecimento em programação está apto a criar e desenvolver aplicativos.

Esses aplicativos ficam todos catalogados e à disposição dos usuários na *Google Play*. Segundo levantamentos do Google, a *Google Play* já conta com mais de um milhão de aplicativos e mais de 50 bilhões de downloads.

Atualmente o Android está disponível na versão 6.0: *Marshmallow*, anunciada em 05 de outubro de 2015, cada versão segue em ordem alfabética e com o nome de algum doce.

O objetivo geral deste trabalho é apresentar uma solução de comunicação entre uma aplicação *mobile* e um sistema web, atrelado ao uso de *Web Services*, tomando como exemplo o desenvolvimento de um aplicativo voltado para o mercado veterinário de equinos, posteriormente incluído no estudo de caso deste trabalho.

Com a aplicação *móbile* é possível facilitar a coleta de informações do atendimento que o veterinário coleta em campo (atendimento em um haras, por exemplo) onde antes era feita através de várias planilhas em folhas de papel e

posteriormente inserir esses dados no sistema, detalhando os procedimentos efetuados, isso custa um tempo hábil do profissional, agora esse processo feito uma única vez, através do aplicativo, otimizando o tempo e economizando dinheiro, evitando o retrabalho de lançamentos dos dados no sistema.

O objetivo geral deste trabalho foi apresentar uma arquitetura que provesse a comunicação de dados entre sistemas de plataformas distintas na qual fosse garantir a integridade dos dados, baixa taxa de processamento e desempenho, assim atendendo os requisitos desejados para o desenvolvimento do aplicativo. Para o desenvolvimento do aplicativo foi utilizando a linguagem *JAVA* e o ambiente de desenvolvimento Android, composto pelo *Android Studio* + *SDK Android*, para a sincronia de dados com o sistema *.NET*, foi utilizado *web services* com o protocolo *SOAP(Simple Object Access Protocol),* tecnologias que garantissem a agilidade do desenvolvimento, confiabilidade e segurança dos dados.

Como objetivos específicos foram utilizados para resolução do problema a tecnologia mobile, destacando a importância dos aparelhos móveis, seus benefícios, sistemas suportados, facilitando tarefas cotidianas, devido sua praticidade, usabilidade e interoperabilidade de sistemas de plataformas distintas, levando em conta a melhoria contínua da qualidade e a veracidade dos dados. Especificamente para o desenvolvimento foram utilizados:

- O estudo da tecnologia embasada na pesquisa em livros.
- O desenvolvimento de aplicações nativas Android, utilização da linguagem *JAVA*.
- Aplicações offline, armazenamento de dados através do *SQLite*, flexibilidade de um banco de dados compacto, com suporte nativo do sistema operacional Android.
- Comunicação de dados do aplicativo com o servidor mediante conexão com a internet, utilização da tecnologia de protocolo *SOAP (Simple Object Access Protocol)* para esta tarefa.
- Design das interfaces, utilizando os padrões especificados pela Google no desenvolvimento de aplicações mobiles.

O método científico de pesquisa foi bibliográfico, conteúdo online e entrevistas com colaboradores da empresa envolvida no estudo de caso deste trabalho.

O trabalho foi estruturado em sete capítulos, sendo que o primeiro conceitua a apresentação do Android e toda sua arquitetura, o segundo apresenta o *Web Service* e serviços através do protocolo *http*, o terceiro descreve o uso do protocolo SOAP, seguido pelo quarto explicando mais a fundo a biblioteca*kSOAP2,*o quinto aborda o estudo de caso, como aplicações mobile e web podem ajudar a resolver problemas cotidianos, o sexto capitulo implementa a biblioteca KSOAP2 o seu desenvolvimento, utilizando para receber o protocolo *SOAP* e converter em dados que possam ser interpretados pela linguagem de programação.

Com base nas informações conseguidas a partir dos estudos realizados nos capítulos anteriores, o capítulo sétimo se reserva às considerações finais.

2. O ANDROID

O Android é um sistema operacional *open source* para dispositivos móveis baseado no *Kernel Linux*, desenvolvido pela *Open Handset Aliance* e liderada pelo Google *Inc*, na qual inclui operadoras móveis, fabricantes de aparelhos, fabricantes de componentes e empresas de marketing.

A *Open Handset Aliance* é um grupo formado por grandes empresas do mercado de celulares. Entre alguns integrantes como *HTC, LG, Motorola, Samsung, Sony Ericsson*. O objetivo do grupo é definir uma plataforma moderna, para mudar e melhorar a experiência do usuário mobile. Na figura 1 o logotipo da marca é representado por um robô.

Figura 1 - Logotipo Android

Fonte: Developers Android (2016)

Segundo (2010, Lecheta) o fato de existir uma plataforma única e consolidada é uma grande vantagem para as empresas, pois a mesma possui código aberto e assim sendo torna possível realizar alterações no código-fonte para a customização. Com isso, cada fabricante altera da melhor forma possível e personaliza de acordo com as necessidades de cada uma sem precisar compartilhar ou distribuir o que foi customizado.

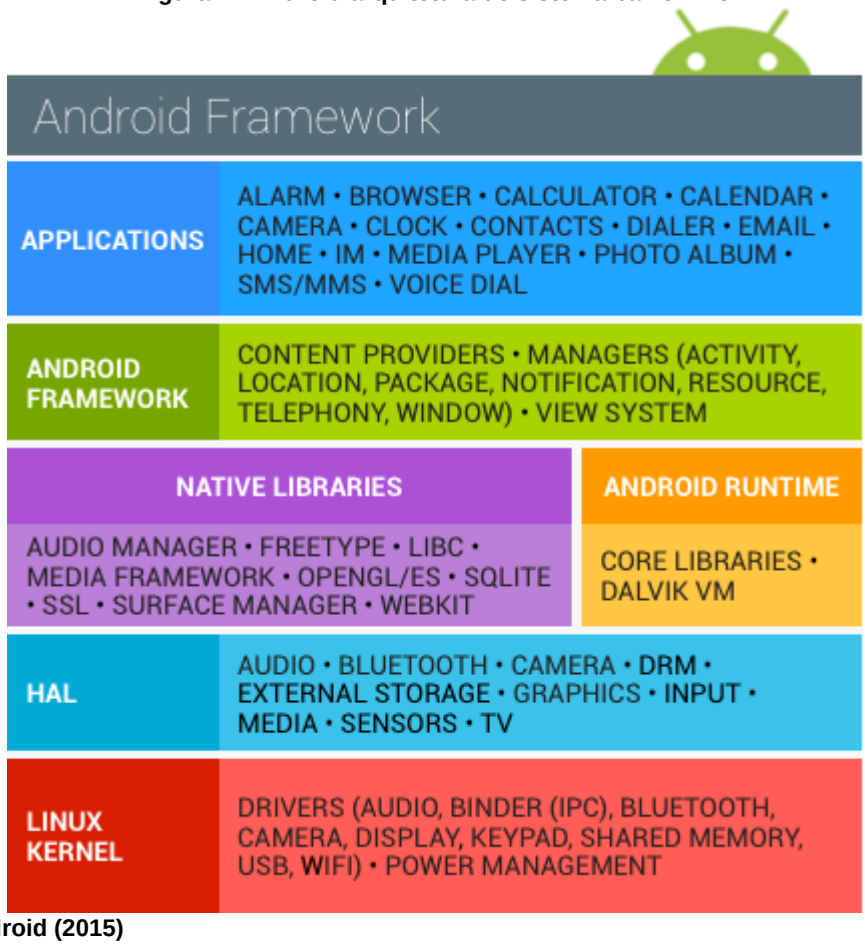
2.1. ARQUITETURA

Android é um conjunto de software de código aberto criado para uma ampla variedade de dispositivos com diferentes formatos e elementos. Os principais objetivos do Android são a criação de uma plataforma de software aberto disponível para as operadoras, *OEMs* (*Original equipment manufacturer*) e desenvolvedores para tornar as suas ideias inovadoras em realidade e para introduzir, um produto que melhora a experiência dos usuários na utilização dos dispositivos móveis.

O Android é uma plataforma que inclui um sistema operacional, *middlewere* e aplicativos. (ANDROID, 2015)

A arquitetura é dividida em núcleo, *runtime*, bibliotecas, *framework* e aplicações conforme detalhado na figura 2.

Figura 2 - Android arquitetura de sistema baixo nível



Fonte: Android (2015)

Para uma compreensão detalhada, as funções das camadas da arquitetura são descritas a seguir.

2.1.1. APPLICATIONS

A camada *Applications* representa as aplicações que executam sobre a plataforma. Elas podem ser tanto aplicações nativas – como o gerenciador de contatos, navegador, calendário, etc. – como aplicações criadas por terceiros. Aliás, para o Android não existe distinção entre aplicações nativas e aplicações desenvolvidas por outras pessoas, e é esta característica que garante a ela o alto grau de flexibilidade e extensibilidade da plataforma. (ANDROID, 2015)

2.1.2. APLICATION FRAMEWORK

Na camada *Application Framework* ficam as *APIs* do Android que são normalmente utilizadas pelas aplicações que são executadas sobre a plataforma, é o nível que a maioria dos desenvolvedores de aplicativos se preocupam. Os gerenciadores de serviços de telefonia, localização, mensagens e notificação são alguns exemplos de aplicativos que utilizam os recursos dessa camada. (ANDROID, 2015)

2.1.3. LIBRARIES

As bibliotecas são um conjunto de programas que fornecem suporte para que as aplicações utilizem os recursos no dispositivo; tais como: Acesso a banco de dados (*SQLite*), recursos gráficos de vídeo, renderização 3D, áudio, etc. (ANDROID, 2015)

2.1.4. ANDROID RUNTIME

Runtime Android (ART) é camada da arquitetura que dá suporte para que as aplicações baseadas na plataforma sejam executadas no qual consiste em uma máquina virtual, que tem a funcionalidade de dar suporte a execução de toda e qualquer aplicação, nessa camada observamos que ela dá condições para que as aplicações baseadas na plataforma sejam executadas. (ANDROID, 2015)

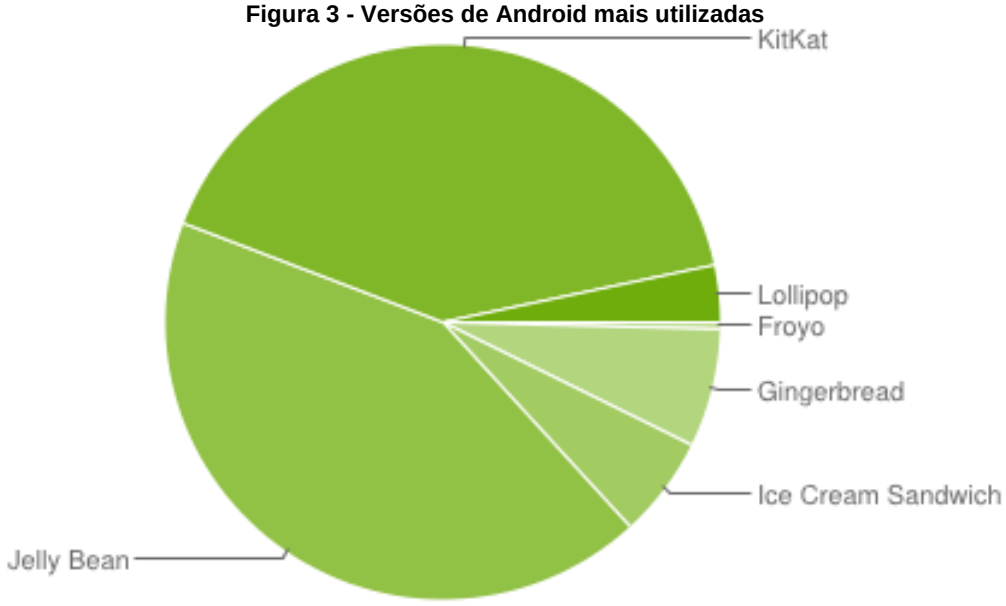
2.1.5. KERNEL LINUX

O sistema operacional do Android foi baseado no kernel 2.6 do Linux, e é responsável por gerenciar a memória, os processos, threads e a segurança dos arquivos e pastas, além de redes, segurança e drivers. (ANDROID, 2015)

2.2. VERSÕES DO ANDROID

O nome da versão do Android procede de uma sequência de codinome ordenado alfabeticamente com nome de doces, na Figura 3. são apresentadas as versões mais utilizadas.

Figura 3 - Versões de Android mais utilizadas



Fonte: Developers Android (2016)

Como podemos observar na figura 3, as versões “*KitKat*” e “*Jelly Bean*” são as mais utilizadas nesta plataforma.

2.2.1. PLATAFORMA CODINOMES, VERSÕES, NÍVEIS DE API E RELEASES NDK

Os codinomes batem com os seguintes números de versão, juntamente com os níveis de *API* e *NDK* fornecidos.

Figura 4 - Versões e APIs do Android

Code name	Version	API level
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5
Donut	1.6	API level 4, NDK 2
Cupcake	1.5	API level 3, NDK 1
(no code name)	1.1	API level 2
(no code name)	1.0	API level 1

Fonte : Android (2015)

Segundo a figura 4, a versão 6.0 codinome “*Marshmallow*”, de nível *API* 23 e *NDK8* foi a última a ser oferecida no primeiro semestre de 2016.

3. WEB SERVICES

Web Services (Serviços Web) é uma tecnologia para comunicar uma linguagem de programação com a outra, por exemplo, a linguagem de programação *Java* pode interagir com *PHP* e *.NET* usando *web services*. Em outras palavras, o *web services* fornece uma maneira de assegurar a interoperabilidade. (JAVA T POINT, 2015)

> Web Services foram criados para resolver a interoperabilidade das aplicações em vários sistemas operacionais, linguagens de programação e modelos de objeto. Web Services podem conseguir isto, baseando-se em padrões da Internet bem suportados, como (HTTP) Hypertext Transfer Protocol e o (*XML*) *Extensible MarkupLanguage*. (TURTSHI et al. 2002, 576).

SegundoIBM (2002) *XML* é uma linguagem de marcação, foi criado *pelo W3C (World Wide Web Consotium)* projetado para a web no intuito de poder criar as próprias tags e superar as limitações do *HTML*. Abaixo um exemplo de *XML*.

```
<XML version="1.0" encoding="UTF-8"?>
<Usuario>
<CodUser>1</CodUser>
<Nome>Carlos Eduardo</Nome>
<Email>Carlos.Santiago@tcc.com</Email>
<Password>511e31c4b9109925b6be03ad651693a9</Password>
</Usuario>
```

Segundo a IBMo*Web Service* é termo técnico de uma função que está disponível em um endereço na rede (web), ou seja, através de um link http conseguimos obter dados, podendo se concluir que*web services* são serviços baseados em nuvem.

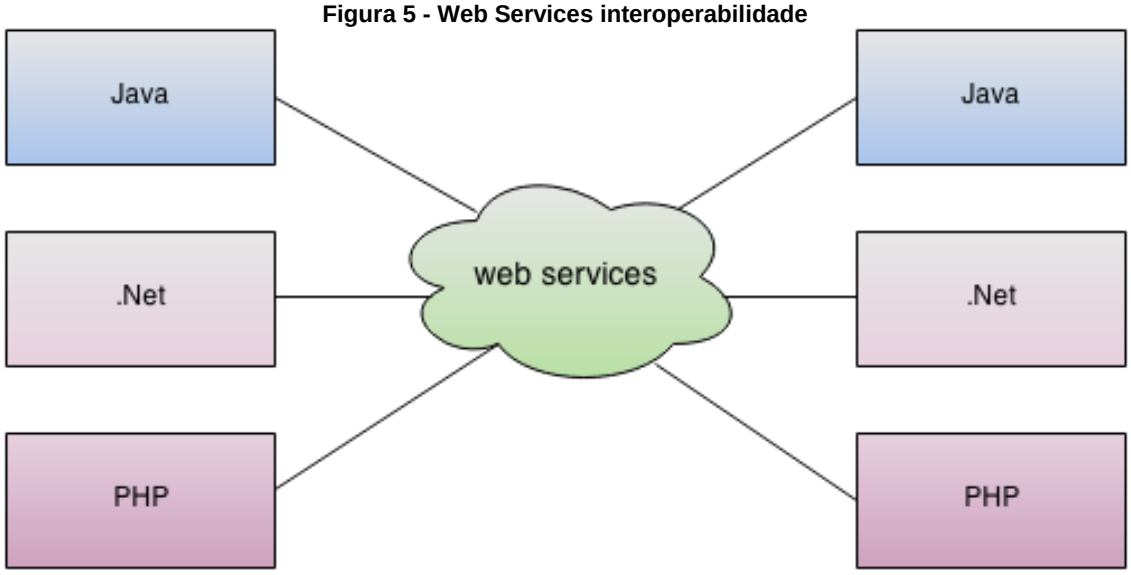
Web Service é uma aplicação lógica de programação e pode ser usada por meio da internet. Os *web services* são aplicativos cliente e servidor que se comunicam por *HyperText Transfer Protocol* da *World Wide Web (HTTP)*. Conforme descrito pelo *World Wide Web Consortium (W3C)*, serviços web fornecem um padrão de interoperabilidade, ou seja, a capacidade de interagir com outro sistema, entre as aplicações de software em execução em uma variedade de plataformas e frameworks. *Web Services* são caracterizados pela sua grande capacidade de troca de informação, graças ao uso de *XML*. *Web services* podem ser combinados de forma fracamente acoplada, que podem possuir dois ou mais sistemas de computação interligados,

sendo que cada sistema possui o seu próprio sistema, gerenciamento, etc., para poder processar operações complexas.

3.1. HOST DE SERVIÇO

O *Host de web Service*, assim é chamada a máquina na qual executa um *Web Service*. A aplicação cliente envia uma solicitação para o *Host de web service*, que executa a solicitação e devolve uma resposta para a aplicação. Esse tipo de computação distribuída beneficia os sistemas de vários modos, por exemplo, uma aplicação sem acesso direto aos dados de outro sistema poderia ser capaz de enviar e receber dados através de um *web service*. Outro exemplo seria uma aplicação que não tem capacidade de processamento suficiente que poderia utilizar um *web service* para enviar os dados que serão processados e receber somente o resultado do que foi processado, podendo tirar vantagem e facilitar o processamento dos dados.(DEITEL, 2000)

Figura 5 - Web Services interoperabilidade



Fonte: JAVAT POINT (2015)

Na Figura 5, pode-se observar a interoperabilidade entre sistemas, que por sua vez possuem diferentes tipos de linguagem, a tarefa de sincronizar todos esses dados é responsabilidade do *Host de Web Services*.

4. PROCOLO SOAP

Segundo Tidwell, Snell e Kulchenko (2001), *SOAP* na tecnologia de *web services* é um protocolo para padronização de troca de mensagens compartilhadas entre aplicações. Define que o SOAP é um envelope com base em *XML* para a troca de informações, contendo um conjunto de regras para ser interpretado quando requisitado.

A utilização do SOAP se torna adequada e ampla, pela sua variedade de padrões, podendo se integrar em diversos tipos de aplicações. Isso contribui para a sua crescente popularidade.

SOAP Simple Object Access Protocol é uma especificação apresentada pela *W3C* e de autoria da *Microsoft, IBM/Lotus Developmentor and UserLand*. A especificação que define o contrato semântico de requisição e resposta, onde o consumidor ou prestador, permite uma poderosa plataforma de troca de dados. (ANDERSON, 2000).

Segundo a documentação W3C, 2004 informa que SOAP é um protocolo para troca de informações em um ambiente distribuído descentralizado. É um protocolo baseado em *XML* que consiste em três partes: um envelope que define um quadro para descrever o que está em uma mensagem e como processá-lo, um conjunto de regras de codificação para expressar exemplos de tipos de dados definidos pelo aplicativo, uma convenção para representar chamadas de procedimento remoto e respostas. O SOAP pode potencialmente ser usado em combinação com uma variedade de outros protocolos.

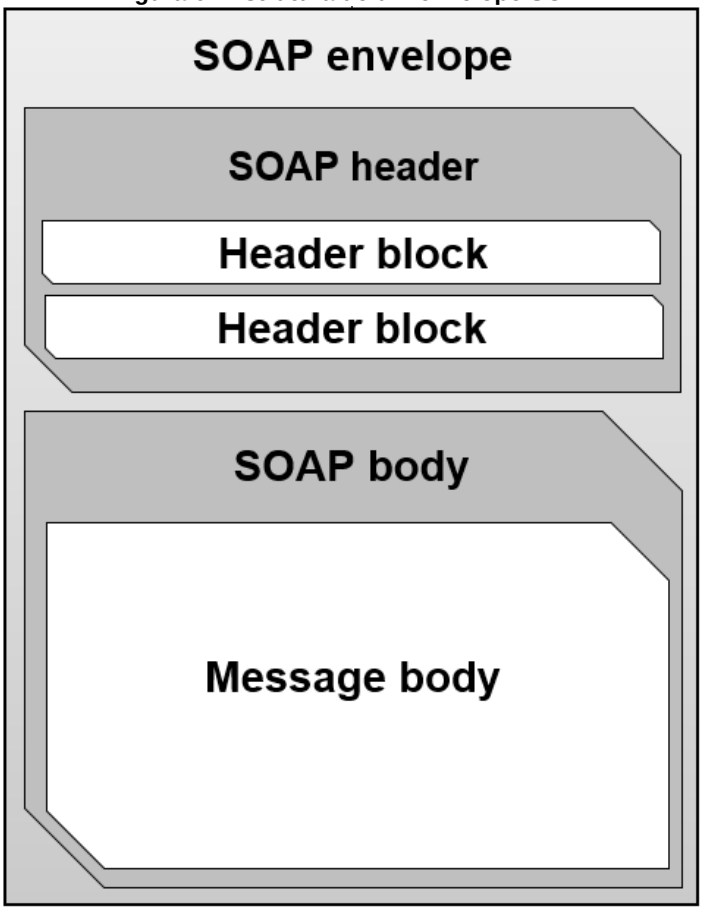
4.1. ESTRUTURA DO PROTOCOLO SOAP

O *SOAP* fornece um mecanismo simples e leve para a troca de informações estruturadas entre os ambientes, descentralizado e distribuídos usando XML. O SOAP serve para ter uma análise das estruturas e conjuntos de dados como parâmetros para um método web. (ANDERSON, 2000).

```
<XML version="1.0" encoding="UTF-8"?>
<Usuario>
<CodUser>1</CodUser>
<Nome>Carlos Eduardo</Nome>
<Email>Carlos.Santiago@tcc.com</Email>
<Password>511e31c4b9109925b6be03ad651693a9</Password>
</Usuario>
```

O protocolo SOAP consiste em um envelope que pode conter um cabeçalho e um corpo caso necessário, como observamos na Figura 6, o envelope e seus atributos.

Figura 6 - Estrutura de um envelope SOAP



Fonte: Adaptada de Tidwell, Snell e Kulchenko (2001)

Toda mensagem *SOAP* contém o envelope. O envelope é o elemento do documento *XML*, podendo conter declarações de *namespaces* e atributos adicionais, todas as informações contidas são representadas no documento *XML*. O *Header* (cabeçalho) pode ser opcional, caso usado, é o primeiro elemento. É ele quem carrega informações, tal como, se a mensagem deve ser processada por um *web service* intermediário (É importante lembrar que, ao trafegar pela rede, a mensagem

normalmente passa por diversos pontos intermediários até alcançar o destino final).O *Body* (corpo) é um elemento obrigatório no envelope, é ele que contém as informações que serão transportadas até o destino. O *Body* ou corpo, além de conter a mensagem, pode conter um elemento que carrega outras mensagens, como mensagens de status, e mensagens de erros, caso ocorra algum em um processo de um *web service*.(DANTAS, 2007)

Em outras palavras, o *SOAP* é um conjunto de dados empacotados em uma *String* como *XML* e transportada no corpo de uma mensagem *HTTP*. Em particular, o envelope e as regras de codificação são definidos em espaços distintos a fim de promover a simplicidade através modularidade. (ANDERSON, 2000)

5. BIBLIOTECA KSOAP2-ANDROID

O *kSOAP2-Android* é uma biblioteca para ser utilizada em dispositivos moveis que utilizam como linguagem de desenvolvimento o *Java*, com pouca capacidade de processamento e que necessitem o consumo do protocolo *SOAP*. O *KSOAP2-Android* é um *fork* (é uma cópia de um repositório que não afeta o projeto original), da biblioteca *Ksoap2* para outras plataformas que utilizam bibliotecas *Java*. *Ksoap2-android* tem sido constantemente melhorado e aperfeiçoado com mais recursos. Ela é mantida ativamente, com atualizações que são executadas semi-regularmente, através de contribuições da comunidade na forma de melhorias. Até a versão 3.4.0, a biblioteca *Ksoap2-android* ainda está usando *Java* 1.3 assim operando em *Java ME, Black Berry* e demais sistemas *Java*. O *Ksoap2-android* está licenciado sob *MIT* e pode, portanto, ser incluído em uma aplicação comercial. (SIMPLIGILITY, 2016)

5.1. COMPOSIÇÃO KSOAP2

Ksoap2-android é composto por um *parser XML*, um de “serializador” e uma camada de transporte. O processo de “serialização” e “deserialização” funciona de modo que os objetos possam ser armazenados ou transferidos e depois recriados. A serialização é o processo de converter um objeto em uma sequência linear de bytes que podem ser armazenados e transferidos. A “deserialização” é o processo inverso, transforma os bytes em objeto, o *parser* analisador utilizado em *kSOAP2* é o *kXML*.(KOBJECTS, 2016)

5.2. DESERIALIZAR / SERIALIZAR

Esta seção é responsável por mapear a representação do objeto para a representação *XML* e vice-versa. Ao usar *complexTypes* em *kSOAP*, os objetos que a camada de serialização pode operar e implementar a interface *KVMSerializable*. Para utilizarmos este serviço de mapeamento devemos "registrar" seus objetos "Serializable", para que o motor consiga mapear o nome encontrado no XML, assim como o nome da classe que irá carregar os valores. Se os objetos não são do tipo

complexo ou os tipos complexos não são registrados, o *serializer* pode quebrar os objetos em *SoapPrimitives* ou *SoapObjects*.(KOBJECTS, 2016)

5.3. TRANSPORTE

A camada de transporte fornece o mecanismo em que as mensagens *SOAP* são trocadas entre cliente e servidor, prosseguindo o trabalho de resumo desta camada para que diferentes tipos de transportes ou outros meios de comunicação. Os mecanismos de transporte básicos, que são atualmente incluídos na biblioteca são denominados *HTTPTransport.*Com uma *flag. dotNet* é possível mudar o *SoapSerializationEnvelope* de comportamento padrão para o tratamento de *namespaces* que é um padrão no *.NET.*(KOBJECTS, 2016)

6. ESTUDO DE CASO

Empresa AgroRoxa® atuante no ramo de Agronegócios, possui diversos sistemas para o auxílio e facilitação de problemas encontrados no ramo.

Em um dos sistemas de veterinário para equinos havia a necessidade de catalogação de matrizes e equinos de clientes, haras, vacinas e outros suprimentos. O sistema deveria ter como foco o acompanhamento veterinário e principalmente um banco de dados, dotado de várias informações desde controle de peso a vacinas de seu rebanho equino.

6.1. PROBLEMA

O principal problema encontrado pela empresa foi a dificuldade que os médicos veterinários encontravam quando efetuavam um atendimento. Eles registravam os dados em agendas, planilhas impressas em papeis ou qualquer outro meio de anotação, posteriormente tinham que recorrer a esses dados para imputar no sistema e gerar a emissão de cobrança dos honorários.

Gastava-se muito tempo na recuperação e organização dessas informações, por não haver uma base de dados em um único local, a emissão do prontuário dos animais tornava-se uma tarefa tediosa e inviável.

A utilização de aparelhos como notebooks era uma tarefa dispendiosa, pois levava-se tempo para ligar, conectar-se e transporta-lo até o local, tornando-se pouco eficiente para gravar os registros.

6.2. SOLUÇÃO ADOTADA

Através de uma reunião com os colaboradores da empresa foram levantadas as principais necessidades do usuário do sistema, são elas:

- Mobilidade;
- Custo relativamente baixo;
- Uma base de dados;

- Eficiência energética;
- Operar em locais onde não há conectividade;
- Sincronizaçãodos dados;

Para atender os requisitos propostos optou-se pela utilização da tecnologia mobile (smartphones/tablets), principalmente por sua eficiência energética, custo benefício e praticidade. A solução se deu ao uso de um aplicativo e que fosse possível registrar os dados coletados em um banco de dados e quando houvesse uma conexão com a internet, podendo realizar a sincronização dos dados armazenados no dispositivo com o sistema principal na web e do sistema web para o dispositivo, garantindo a integridade dos dados nas duas bases de dados.

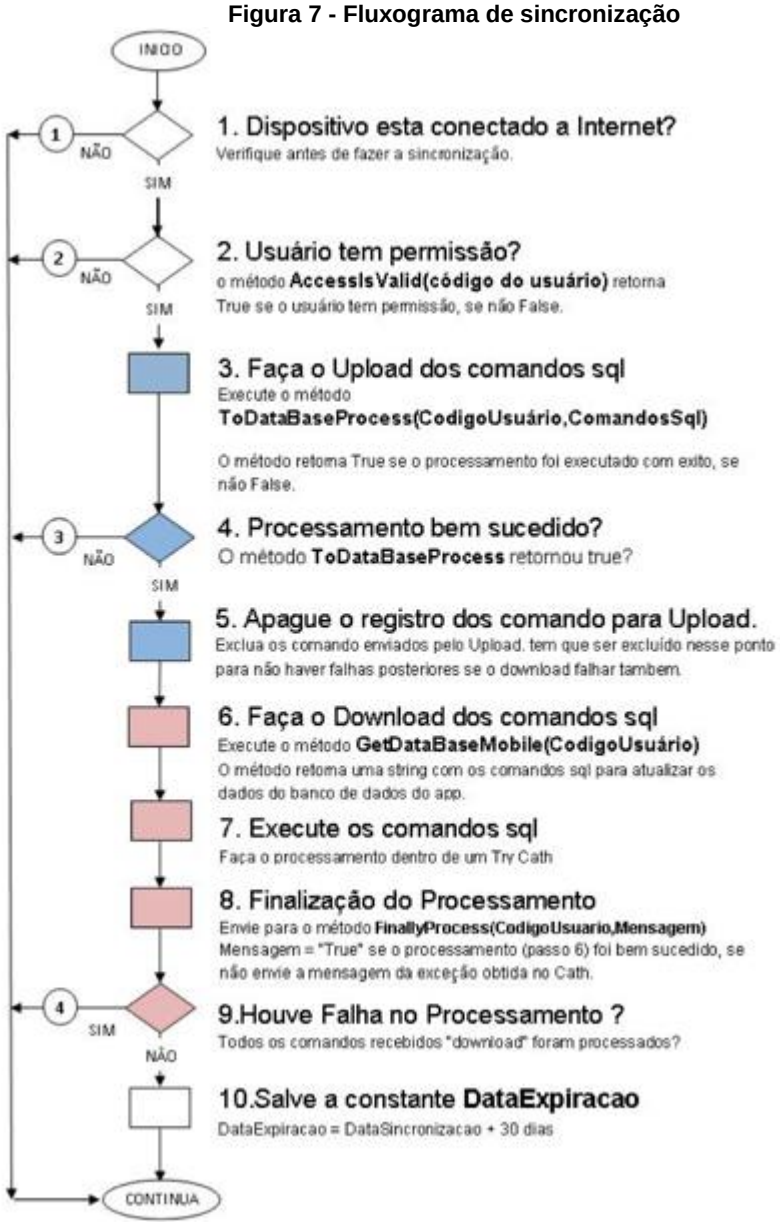
A opção para realizar a sincronização dos dados foi utilizar *web services* e o protocolo *SOAP*, por conveniência da facilidade de programação, tendo em vista que o sistema web utiliza a tecnologia *.NET.* Evidenciando a interoperabilidade resultante do uso de *web services* em qualquer ambiente.

7. IMPLEMENTAÇÃO DA BIBLIOTECA KSOAP2 NO ANDROID

O modelo adotado para o desenvolvimento da aplicação se baseia no modelo cliente-servidor, como dito anteriormente, a integração do sistema mobile com o sistema da web, foi feito através de *web services* e o protocolo *SOAP*.

A sincronização dos dados armazenados no dispositivo móvel e o sistema web segue por uma lógica rígida para evitar que algum dado possa ser perdido. Na Figura 7 é demonstrado um exemplo de fluxograma para a execução da sincronização de dados com o sistema.

Figura 7 - Fluxograma de sincronização



Fonte: Próprio autor

7.1. CONFIGURAÇÃO DO PROJETO

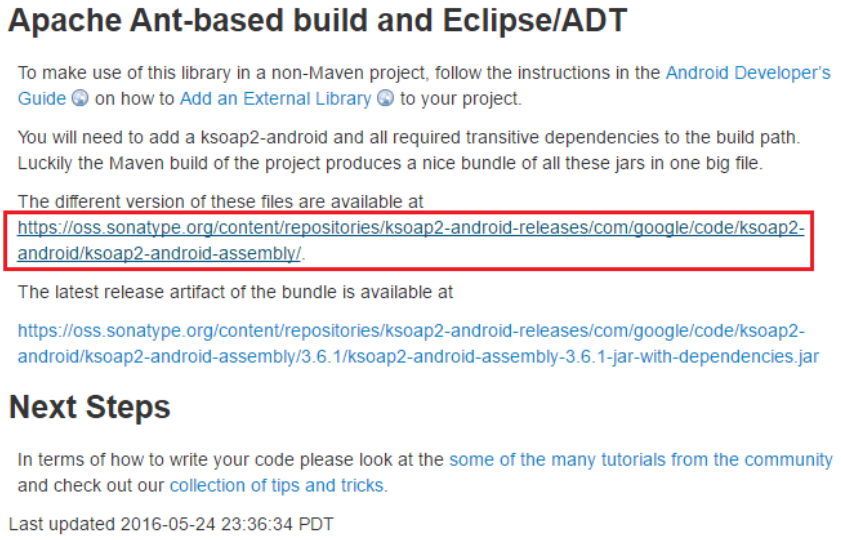
No desenvolvimento do app foi utilizado a ferramenta oferecida pelo Google o *Android Studio*, que é de código aberto, baseado na *IDEA intelij*. Com o ambiente de desenvolvimento configurado com *JDK (Java Development Kit)* e o *SDK (software development kit)*. O *Android SDK* fornece as bibliotecas da *API* e ferramentas de desenvolvimento necessárias para construir, testar e depurar aplicativos para o Android. Para a aplicação poder se conectar na internet é necessário o acréscimo de uma permissão de acesso à INTERNET no arquivo *AndroidManifest.XML*. São essas permissões que definem o que, o aplicativo poderá usar de recursos do aparelho, como demostrado no código de permissão abaixo.

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

7.2. TRANSFERINDO KSOAP

Para a receptação e interpretação do *XML* no Android, devemos utilizar uma biblioteca de terceiros para o consumo de *SOAP*, a *KSOAP2-android* que se encontra no endereço: "http://simpligility.github.io/ksoap2-android/getting-started" é aconselhável sempre utilizar as versões mais estáveis de bibliotecas, para fazer o download do ".jar", clicar no link, conforme é informado na ilustração da Figura 8 abaixo.

Figura 8 - Download biblioteca KSOAP2

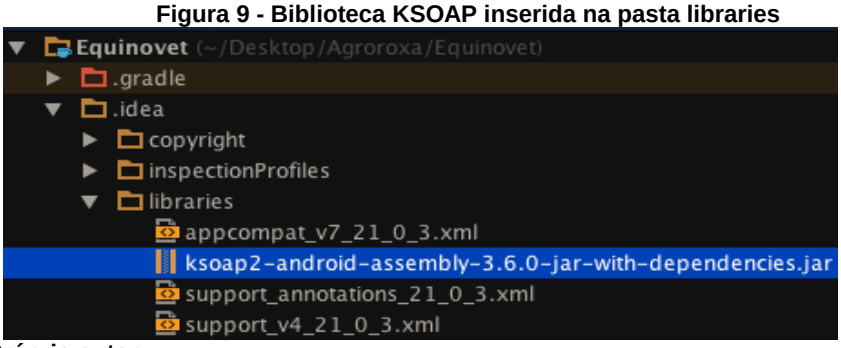


Fonte: Simpligility (2016)

7.3. ADICIONANDO A BIBLIOTECA KSOAP NO PROJETO

Para utilizarmos a biblioteca *kSOAP* no projeto, devemos adicioná-la como demostrado na Figura 9, onde o projeto com a biblioteca adquirida já está inserido na pasta *libraries*.

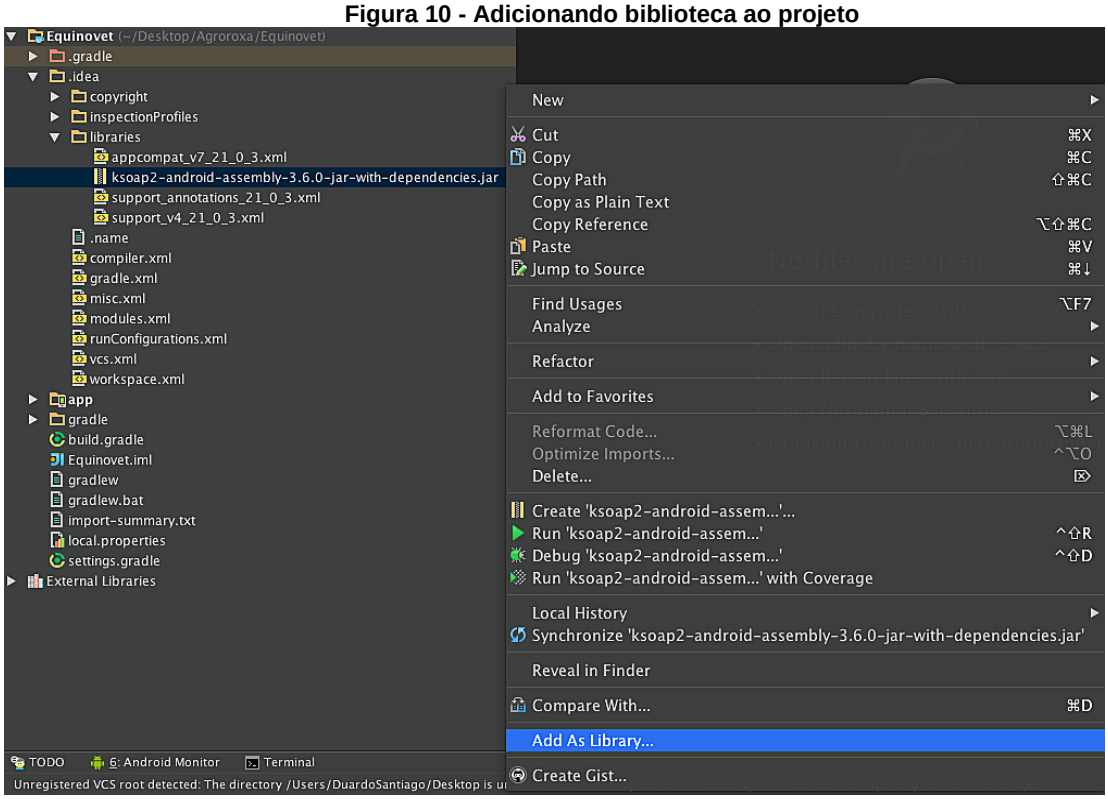
Figura 9 - Biblioteca KSOAP inserida na pasta libraries



Fonte: Próprio autor

A biblioteca *kSOAP* é visível como uma dependência do projeto. Para este reconhecer e utilizar recursos da biblioteca, é necessário referenciá-la no projeto. Clicando com o botão direito sobre o arquivo da biblioteca *kSOAP2* será exibido um menu, posteriormente devemos selecionar "*Add As Library*…" Para adicioná-la ao projeto, conforme exemplo da Figura 10 abaixo.

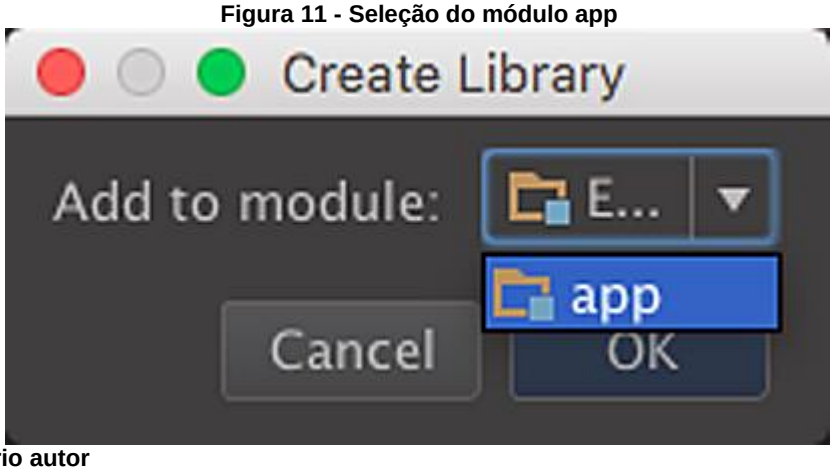
Figura 10 - Adicionando biblioteca ao projeto



Fonte: Próprio autor

Logo após executado o passo descrito na Figura 10, o programa retornará uma caixa de mensagem contendo uma combobox, o valor de “Add to module: ” deverá ser selecionado para “app”, conforme demonstrado na Figura 11.

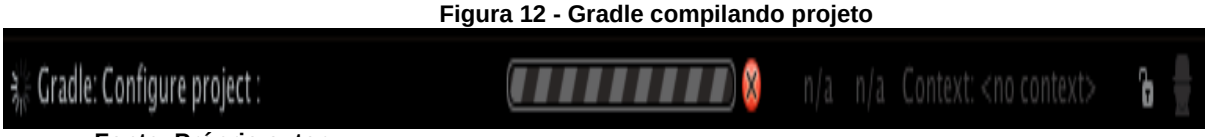
Figura 11 - Seleção do módulo app



Fonte: Próprio autor

Depois da confirmação do módulo como “app”, o *Gradle*(gerenciador de dependências) iniciará a compilação do projeto, como podemos observar na Figura 12, adicionando a referência da biblioteca no projeto.

Figura 12 - Gradle compilando projeto



Fonte: Próprio autor

Concluída a compilação pelo *Gradle*o arquivo *build.gradle*irá adicionar um caminho, informando onde o arquivo da biblioteca está, para poder “versionar” e compilar o código.

```
dependencies {
        compile 'com.android.support:appcompat-v7:21.0.3'
        compile 'com.android.support:support-v4:21.0.3'
        compile files('/TCC/libraries/ksoap2-android-assembly-3.6.0-jar-with-dependencies.jar')
    }
```

Após o caminho ser informado a biblioteca está pronta para ser utilizada.

7.4. HABILITANDO O SERVIÇO WEB

Para utilizar os recursos do *KSOAP*, temos que conhecer o serviço que vai ser consumido. O endereço do *web service* é a URL base que é usado para se conectar ao serviço da web, como descrito no trecho de código abaixo.

```
POST /webservices/ToDataBase.asmx HTTP/1.1
Host: www.tcc.com
Content-Type: text/XML; charset=utf-8
Content-Length: length
SOAPAction: "http://www.tcc.com/webservices/ToDataBase"

<?XML version="1.0" encoding="utf-8"?>
<soap:Envelope XMLns:xsi="http://www.tcc.org/webservices/ToDataBase"
XMLns:xsd="http://www.tcc.org/webservices"
XMLns:soap="http://www.tcc.org/webservices/ToDataBase">
<soap:Body>
<ToDataBase XMLns="http://www.tcc.com/webservices/">
<CodUsuario>string</CodUsuario>
<CmdSQL>string</CmdSQL>
</ToDataBase>
</soap:Body>
</soap:Envelope>
```

O próximo trecho de código mostra a resposta da requisição feita no web service.

```
HTTP/1.1 200 OK
Content-Type: text/XML; charset=utf-8
Content-Length: length

<?XML version="1.0" encoding="utf-8"?>
<soap:Envelope XMLns:xsi="http://www.tcc.org/webservices/ToDataBase"
XMLns:xsd="http://www.tcc.org/webservices"
XMLns:soap="http://www.tcc.org/webservices/ToDataBase">
<soap:Body>
<ToDataBase XMLns="http://www.tcc.com/webservices/">
<ToDataBaseResult>string</CToDataBaseResult>
</ToDataBase>
</soap:Body>
</soap:Envelope>
```

7.5. UTILIZANDO A BIBLIOTECA KSOAP

Segundo Govender, para iniciar a enviar e receber os dados, primeiro precisamos informar os “endereços” para usar durante o código posteriormente, como informa o código abaixo:

```
String NAMESPACE = "http://tcc.org/webservices";
String URL = "http://tcc.org/webservices/ToDataBase.asmx";
String SOAP_ACTION = "http://tcc.org/webservices/ToDataBase";
String METHOD_NAME = "ToDataBaseProcess";
```

Para a configuração do envelope *SOAP*, é necessário definir as informações de pedido, importado a classe *SoapObject (org.ksoap2.serialization.SoapObject).*

Começamos criando uma instância da *SoapObject* classe, que requer dois parâmetros, um *namespace* e uma nome do método.

```
SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
```

Usando o *SoapSerializationEnvelope* classe disponível a partir do *import (org.ksoap2.serialization.SoapSerializationEnvelope)*, é criado o envelope *SOAP*, no trecho de código a seguir uma instância da *SoapSerializationEnvelopeclasse*.

```
SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
```

O parâmetro passado para a classe é usado para informar a biblioteca *kSOAP* que nós estaremos usando *SOAP 1.1.*

Para enviar a solicitação para o *web service*, é preciso criar uma solicitação de transporte *HTTP*. Utilizando a classe *HttpTransportSE* do *import (org.ksoap2.transport.HttpTransportSE)*, precisamos apenas informar a URL base do*web service*, que foi definida em uma variável chamada: *URL* no início do código.

```
HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
```

Podemos adicionar propriedades adicionais para a solicitação usando o *AddProperty*, método da classe *SoapObject*, como no exemplo a seguir, este por sua vezusa os parâmetros passados para o método, o “CodigoUsuario” e o “ComandoSQL”.

O primeiro parâmetro do método *addProperty* é o nome do primeiro parâmetro esperado do *web service*, o segundo parâmetro é o valor.

```
request.addProperty("CodUsuario", CodigoUsuario);
request.addProperty("CmdSQL", ComandoSQL);
```

Utilizando a propriedade. *dotNet = true*, irá preparar o envelope para consumir *web services* em *.NET* da *Microsoft.*

```
envelope.dotNet = true;
```

Com o envelope já definido, a rota e os dados inseridos na mensagem do protocolo, já é possível enviar a solicitação *SOAP* para o *web service*, através de *HTTP,* utilizando os objetos de transporte e do envelope criados anteriormente. O objeto de transporte *HTTP* tem um método o *call()*, que é usado para adicionar o *SOAP* e transportá-lo. O método recebe como parâmetro o *soap action*, o “alvo” da requisição e o envelope,

```
androidHttpTransport.call(SOAP_ACTION, envelope);
```

Após o processamento da requisição no*web service* é enviado uma resposta de volta, será preciso processá-la. Para extrair a resposta usamos método*getResponse*.

```
SoapPrimitive response = (SoapPrimitive) envelope.getResponse ();
```

O trecho de código a seguir é o método *ToDataBase*, implementado todo o conteúdo explicado acima. O método aceita duas variáveis como argumento. As variáveis são adicionadas como uma propriedade para a solicitação SOAP, exemplificado anteriormente. Esses dados são enviados para o *web service ToDataBaseProcess* e retornam se o processamento foi realizado com sucesso na base de dados do sistema web.

```
Package com.app.tcc.webservices;


import org.ksoap2.SoapEnvelope;
import org.ksoap2.SoapFault;
import org.ksoap2.serialization.PropertyInfo;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapPrimitive;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;
import org.XMLpull.v1.XMLPullParserException;

public class WebServiceTcc {

/**
*Método para enviar os dados para sincronizar
* @param cUsuario
* @param strSql
* @return boolean
* @throws InterruptedException
*/
public Boolean ToDataBase(String cUsuario, String strSql){

    String NAMESPACE = "http://tcc.org/webservices";
    String URL = "http://tcc.org/webservices/ToDataBase.asmx";
    String SOAP_ACTION = "http://tcc.org/webservices/ToDataBase";
    String METHOD_NAME = "ToDataBaseProcess";
    boolean boolWebResponse = false;


    SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
    SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);

    HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
    request.addProperty("CodUsuario", CodigoUsuario);
    request.addProperty("CmdSQL", ComandoSQL);

    envelope.dotNet = true;

    try {
 androidHttpTransport.call(SOAP_ACTION, envelope);
       SoapPrimitive result = (SoapPrimitive)envelope.getResponse();

boolWebResponse = Boolean.valueOf(response.toString());

    } catch (SocketTimeoutException t) {
       t.printStackTrace();
    } catch (IOException i) {
       i.printStackTrace();
```

```
            } catch (Exception q) {
                q.printStackTrace();
            }
        return boolWebResponse;
    }
}
```

8. CONSIDERAÇÕES FINAIS

As pesquisas realizadas na área de interoperabilidade evidenciam a importância de armazenar, recuperar e processar dados de maneira clara e a simplicidade dos protocolos que os dados são transmitidos.

O uso de Web Services foi primordial para a execução do projeto, pois garantiu a troca de mensagens dos sistemas e unificando a base de dados e garantindo que as bases sempre permaneçam atualizadas e integras, que é de grande importância para os sistemas da empresa, uma grande vantagem dos Web Services é a facilidade e versatilidade de implementação entre as tecnologias, isso aponta que o uso da tecnologia auxilia na transposição de dados de sistemas que na qual, qualquer alteração de componentes, pode acabar representando um custo muito alto, os Web Services podem ser a solução.

9. REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, Richard et. al. **Active Server Pages+:**a preview of. BIrgmingham: Wrox Press, 2000. p. 160-163.

ANDROID, THE WORLD´S MOST POPULAR MOBILE PLATFORM. Disponível em: <http://developer.android.com/about/index.html>. Acesso em: 30 set. 2014.

ANDROID. Codenames, tags, and build numbers. Disponível em: <https://source.android.com/source/build-numbers.html>.Acesso em: 15 Abril. 2016.

ANDROID. Security. Disponível em: <https://source.android.com/devices/tech/security/index.html>. Acesso em: 26 jul. 2015.

ANDROID. The Android source code. Disponível em: <https://source.android.com/source/index.html>.Acesso em: 15 Abril. 2015.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS - ABNT. NBR 10520: informação e documentação: citações em documentos: apresentação. Rio de Janeiro: ABNT, 2002. 7p.

BOUÇAS, Cibele. Mercado de smartphones crescerá 28% em 2014. **Valor Econômico**, 22 jan 2014. Disponível em: <http://www.valor.com.br/empresas/3403512/mercado-de-smartphones-crescera-28-em-2014-na-al-aponta-emarketer >. Acesso em: 3 nov. 2015.

DANTAS, Daniel Chaves Toscano. Simple object access protocol (SOAP). Disponível em :<http://www.gta.ufrj.br/grad/07_2/daniel/>Acesso em: 3 nov. 2015.

DEITEL, Paul; DEITEL Harvey. **JAVA:**como programar. 8.ed São Paulo: Pearson Prentice Hall. 2010. p. 1019-1020.

DEVELOPER.Brand guidelines.Disponível em: <http://developer.android.com/distribute/tools/promote/brand.html>.Acesso em: 17 mar. 2015.

DEVELOPER. Dashboards.Disponível em: <https://developer.android.com/about/dashboards/index.html>.Acesso em: 17 Abril. 2016.

GOVENDER, Sashen.Consuming web servisse with kSOAP.**Envatotuts+**Disponível em: <http://code.tutsplus.com/tutorials/consuming-web-services-with-ksoap--mobile-21242>.Acesso em: 3 nov. 2015.

IBM. DEVELOPER WORKS. Disponível em: <http://www.ibm.com/developerworks/XML/tutorials/XMLintro/XMLintro.html>Acesso em: 3 nov. 2015.

IBM. Concepts of JSON web services. Disponível em: <https://www.ibm.com/support/knowledgecenter/SSGMCP_5.2.0/com.ibm.cics.ts.webservices.doc/concepts/concepts_json.html>Acesso em: 3 nov. 2015.

KOBJECTS. Disponível em: <http://kobjects.org/ksoap2/index.html>.Acesso em: 3 nov. 2014.

LANCHETA, Ricardo R. **Google Android**: aprenda a criar aplicações para dispositivos movéis com Android SDK. 2.ed.São Paulo: Novatec Editora, 2010.

JAVA T POINT. Web services tutorial. Disponível em: <http://www.javatpoint.com/web-services-tutorial>.Acesso em: 4 nov. 2015.

MACORATTI. Disponível em: <http://www.macoratti.net>. Acesso em: 31 mar. 2015.

MICROSFT. HTTP protocol. Disponível em: <https://msdn.microsoft.com/en-us/library/aa767734(v=vs.85).aspx>. Acesso em: 4 nov. 2015.

MICROSFT. How TCP/IP works. Disponível em: <https://technet.microsoft.com/en-us/library/cc786128(v=ws.10).aspx>.Acesso em: 4 nov. 2015.

OFICINA DA NET. O protocolo HTTP. Disponível em: <https://www.oficinadanet.com.br/artigo/459/o_protocolo_http> Acesso em: 22 maio 2016.

OPEN HANDSET ALLIANCE. Members. Disponível em:<www.openhandsetaliance.com/oha_members.html>Acesso em: 5 nov. 2015.

ORACLE. The Java EE 6 tutorial. Disponível em: <http://docs.oracle.com/javaee/6/tutorial/doc/giqsx.html>.Acesso em: 07 de abril de 2015.

ORACLE. XML DB developer´s guide. Disponível em: <http://docs.oracle.com/database/121/ADXDB/partpgjson.htm>.Acesso em: 07 de abril de 2015.

SNELL, James; TIDWELL, Doug; KULCHENCO, Pavel. Programming web services with SOAP. Disponível em: <https://docs.google.com/file/d/0B4K4VAX1hkTgUlczejJkZXI1Unc/preview>. Acesso em: 6 nov. 2015.

SIMPLIGILITY. Ksoap2-android – lightweight, eficiente SOAP on android. Disponível em: <http://simpligility.github.io/ksoap2-android/index.html>.Acesso em: 07 de abril de 2015.

TIDWELL, D.; SNELL, J.; KULCHENKO, P. **Programing Web Services with SOAP**. 1.ed. [S.l.]: O'Reilly, 2001. 216 p.

TURTSCHI, A. et al. Chapter 11 - Web Services. C#.NET Web Developer's Guide, Burlington, p. 575-668, 2002.

WEBOPEDIA. HTTP – Hyper text transfer protocol. Disponível em: <http://www.webopedia.com/TERM/H/HTTP.html>Acesso em: 4 nov. 2015.

W3. HTTP – Hypertext transfer protocol. Disponível em: <https://www.w3.org/Protocols/>Acesso em: 4 nov. 2014.

W3. Simple object access protocol (SOAP) 1.1. Disponível em: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ >Acesso em: 30 mar. 2015.