UNIVERSIDADE DE ARARAQUARA - UNIARA DEPARTAMENTO DE CIÊNCIAS DA ADMINISTRAÇÃO E TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

EDUARDO DE SANTANA DA SILVA

INTEGRAÇÃO DE SISTEMAS LEGADOS E MOBILE UTILIZANDO UMA API RESTFUL

ARARAQUARA 2016

EDUARDO DE SANTANA DA SILVA

INTEGRAÇÃO DE SISTEMAS LEGADOS E MOBILE UTILIZANDO UMA API RESTFUL

Trabalho de Conclusão de Curso apresentado ao Departamento de Ciências da Administração e Tecnologia, da Universidade de Araraquara UNIARA, como exigência para obtenção do título de Bacharel em Engenharia de Computação. Orientador: Prof Dr. João Henrique Gião Borges

ARARAQUARA 2016

LISTA DE FIGURAS

Figura 1 - Representação de um objeto JSON 10
Figura 2 - Configuração do arquivo web.xml.....13
Figura 3 - Diagrama de sequência da comunicação entre aplicação Android e a API 14
Figura 4 - Dependência da biblioteca GraphView que foi adicionada ao projeto.....15
Figura 5 - Telas da aplicação mobile.....16
Figura 6 - Classe assíncrona genérica responsável por consumir o Web Service.....17
Figura 7 - JSON retornado ao consumir o método /listarTodosPeriodo 18
Figura 8 - Classe utilizada na desserialização do objeto JSON retornado 19
Figura 9 - Gráfico obtido na requisição 19
Figura 10 - Informações detalhadas da Série 20

SUMÁRIO

1 INTRODUÇÃO 1
1.1 Apresentação do Tema.....1
1.2 Objetivo.....1
1.3 justificativa.....2
1.4 Problema e Hipótese 3
1.5Metodologia.....3
2. REVISÃO BIBLIOGRÁFICA.....4
2.1 SOA (Service Oriented Architecture) 4
2.1.2 Web Services.....5
2.1.3 Android.....5
2.1.4 REST (Representational State Transfer) 6
2.2 HTTP.....7
2.2.1 Métodos HTTP.....7
2.2.2 Código de Status.....8
2.1.5 JSON 10
2.1.6 Apache Tomcat.....11
2.1.7 JERSEY.....11
2.1.8 ERP.....11
2.1.9 JAVA.....12
3. DESENVOLVIMENTO DO PROJETO.....12
3.1.1 Desenvolvimento do Web Service.....12
3.1.2 Recursos.....14
3.1.3 Aplicação Mobile.....15
3.1.4 Gráficos.....15
3.1.5 Activitys.....16
3.1.6 Requisições ao Web Service.....16
4. CONCLUSÃO.....20
5. REFERÊNCIAS BIBLIOGRÁFICAS.....21

RESUMO

Devido ao grande poder de processamento de smartphones e *tablets* e a facilidade de se manter conectado à internet por *wi-fi* ou tecnologias de dados móveis, os dispositivos móveis se tornaram uma ferramenta importante para diversas empresas. É possível utilizar os diversos recursos do dispositivo para enviar e receber dados a todo momento e empresas vem realizando a integração de sistemas mobile aos seus *ERPs* para assim agilizar seus processos. Para realizar a integração de aplicações de plataformas distintas podem ser utilizados Web Services que disponibilizam recursos para que outras aplicações possam ser facilmente integradas. Este trabalho apresenta o desenvolvimento de uma API JAVA com classes e interfaces que mapeiam os conceitos básicos de serviços web baseados em Rest. A estrutura da API facilita o trabalho com diferentes propostas de arquiteturas *Rest*, além disso a API visa facilitar o trabalho de desenvolvimento de uma aplicação *Android* que se comunique com um servidor *Rest*, não sendo necessário que o desenvolvedor preocupe-se com o desenvolvimento de classes de comunicação com o Web Service tampouco classes que realizem o *a serialização e deserialização* de dados. A aplicação mobile consome o *Web Service* através de requisições REST, obtendo o retorno em JSON e serializando para um objeto pré-definido.

Palavras-chave: API, Restful, Web Services, Android.

ABSTRACT

Due to the great power of smartphones and tablets processing and the ease of staying connected to the internet by wi-fi or 3G, mobile devices have become indispensable for many companies. It can use the various features of the device to send and receive data at all times, companies have used the integration of mobile systems to their ERP to thereby streamline its processes. To enable the integration of heterogeneous applications are used Web services that provide resources so that other applications can be easily integrated. This paper presents the development of an API with Java classes and interfaces that map the basics of web-based services Rest, is not necessary that the developer worry with the development of classes that communicate with the Web service either classes that perform the marshall and unmarshall data. The structure of API developed makes working with different proposals for Rest architectures, in addition to API aims to facilitate the development work of an Android application that communicates with a server Rest.

Keywords: API, Restful, Web Services, Android.

1. INTRODUÇÃO

1.1 APRESENTAÇÃO DO TEMA

A computação móvel foi criada com a finalidade de permitir que seus usuários se mantenham conectados a todo o momento, tal como faria em um computador pessoal, porém tendo a flexibilidade de transportar o dispositivo para qualquer lugar. Essa tecnologia se popularizou com a chegada da internet móvel (3G, 4G e Wi-Fi), do aumento da quantidade de memória RAM e das novas arquiteturas de processador que possibilitaram maior desempenho e consequentemente a construção de aplicações mais elaboradas. (MAURÍCIO e NAKAMURA, 2003)

A grande popularização dos sistemas operacionais *Android*, IOS e Windows Phone e o aumento de planos de telefonia acessíveis, despertou no ramo empresarial o interesse em explorar essas novas tecnologias visando disponibilizar seus produtos e serviços nos novos aparelhos celulares. Pois, utilizar esse tipo de tecnologia nos negócios possibilita ter informação sempre à mão aonde quer que esteja, para consultar quando quiser.

Utilizando uma arquitetura orientada a serviços é possível desenvolver pontos de acesso para que aplicações móveis possam se comunicar, consumir recursos e atualizar informações em sistemas legados. No entanto, muitas vezes é necessário realizar a integração entre softwares distintos escritos em linguagens diferentes e sendo executados em plataformas diferentes, que devem ser capazes de se comunicarem e trocarem informações para assim atingir os objetivos desejados.

1.2 OBJETIVOS

Este trabalho objetivou realizar a integração de um aplicativo mobile com um sistema legado, a fim de disponibilizar as informações em uma aplicação através de um aplicativo *Android* utilizando uma arquitetura orientada a serviços.

1.3 JUSTIFICATIVA

De acordo com IDC (International Data Corporation) o mercado de smartphones em todo o mundo cresceu 13,0 % em 2015, com 341,5 milhões de embarques, o *Android* domina o mercado com uma quota de 82,8% em relação aos outros sistemas operacionais para *Smartphones* em 2015.

Tabela 1 - Participação dos Sistemas Operacionais no mercado

Período	Android	iOS	Windows Phone	BlackBerry OS	Outros
2015	82.8%	13.9%	2.6%	0.3%	0.4%
2014	84.8%	11.6%	2.5%	0.5%	0.7%
2013	79.8%	12.9%	3.4%	2.8%	1.2%
2012	69.3%	16.6%	3.1%	4.9%	6.1%

Fonte: IDC, Agosto de 2015.

A plataforma *Android* foi escolhida para fazer essa integração, pois ela possui algumas vantagens em relação a outras plataformas como ser gratuita, não é preciso comprar ferramentas de desenvolvimento, APIS ou algum tipo de software para poder desenvolver uma aplicação. Estas informações ressaltam a importância de se estudar o tema abordado.

Segundo Saudate, a utilização de *Web Services* tradicionais como o SOAP (*Simple Object Access Protocol*) não apresenta um bom desempenho em aplicações móveis, pois estes tipos de Web Services são baseados em trocas de XML, para este cenário isto seria muito custoso, pois existe um consumo de banda elevada. Na maioria das vezes os usuários não estarão conectados a uma rede Wi-Fi, assim dependendo da utilização de uma rede móvel. A arquitetura REST foi escolhida por ser mais sucinta e menos verbosa, além de possuir vantagens para o desenvolvimento.

1.4 PROBLEMA E HIPÓTESE

Nos dias atuais uma tomada de decisão em tempo hábil é um diferencial para uma organização, contudo existem limitações quando se trata do assunto, é preciso que responsável por análises dos dados da organização tenha acesso a um computador ou a documentos impressos.

Para resolver o problema será realizada uma engenharia reversa do modelo de dados convencional, seguido de um acesso direto a base de dados, por um componente de software que expõe as informações através de uma API que utiliza o padrão RESTful sobre o protocolo HTTP e um aplicativo desenvolvido para executar especificamente em sistemas operacionais *Android* que obtém as informações através da API exposta pelo componente. Os dados obtidos serão exibidos na aplicação *Android* através de *dashboards*, gráficos e relatórios.

1.5 METODOLOGIA

Inicialmente foi realizada uma análise junto aos usuários ligados ao departamento de inteligência de mercado para verificar as necessidades do projeto e definir seu escopo. O sistema ERP tem seus dados armazenados em uma base de dados Microsoft SQL Server 2008. Foram estudadas as tecnologias utilizadas para o desenvolvimento de um *Web Service,* para este caso foi escolhido a IDE Eclipse Java EE IDE *for Web Developers* e um aplicativo para ser executado no sistema operacional *Android*. O aplicativo foi desenvolvido na linguagem de programação Java, utilizando a IDE Android Studio, ferramenta que é disponibilizada gratuitamente pela Google. Em seguida foi escolhido o servidor hospedaria a API, o servidor escolhido foi o APACHE TOMCAT 7.0, este disponibilizado pela *Apache Software Fundation* e distribuído sobre a licença de software livre.

A API possui métodos que quando consumidos retornam os recursos solicitados. Para obter as informações da base de dados Microsoft SQL Server é realizado um acesso direto da API. As informações obtidas pela API serão consistidas em uma camada de negócios. Na aplicação *android* os recursos que foram obtidos através do consumo da API serão expostos aos usuários através de gráficos e dashboards. Foram

realizadas pesquisas embasadas em leitura de artigos, endereços eletrônicos e livros que colaboraram para o enriquecimento teórico e prático do tema abordado.

2. REVISÃO BIBLIOGRÁFICA

2.1 SOA (Service Oriented Architecture)

De acordo com GEORGAKOPOULOS e PAPAZOGLOU (2008) a Arquitetura Orientada a Serviços é um paradigma quem propõe a interoperabilidade e o fraco acoplamento entre elementos básicos de software (serviços). Sua definição emprega a utilização de interfaces neutras e a utilização de protocolos de transporte padronizados. Serviços são unidades lógicas de software que encapsulam um método ou até mesmo um grande processo que envolve vários colaboradores. Em outras palavras, um serviço pode ser subentendido por componentes abertos, auto descritíveis que suportam a integração de aplicações distribuídas de forma rápida e com baixo custo.

Serviços devem utilizar um protocolo padrão e aberto para comunicação, por exemplo, o HTTP, disponibilizando uma infraestrutura de computação distribuída que possibilite a utilização de seus serviços tanto dentro como entre empresas (*intra-enterprise* e *cross-enterprise*, respectivamente). Um serviço deve ter uma interface de comunicação que forneça a assinatura do serviço (parâmetros de entrada, saída, erro e tipos de mensagens). Serviços são descritos por fluxogramas. Resumidamente, um serviço é um componente de software que expões recursos para clientes através de interfaces (contratos) bem definidas. Um serviço é um recurso abstrato que possui a capacidade de realizar tarefas que representam uma funcionalidade do ponto de vista de entidades provedoras e entidades requisitoras. (PAPAZOGLOU, MIKE & GEORGAKOPOULOS, DIMITRIS, 2008)

2.1.2 Web Services

Segundo (W3C, 2004), um serviço web é um sistema de software desenvolvido para suportar iterações máquina-máquina interoperáveis sobre uma rede. O serviço web implementa uma interface descrita em um formato que a máquina pode processar, especificamente o WSDL (*Web Service Description Language*), possibilitando a iteração de outros sistemas utilizando o contrato prescrito no documento WSDL utilizando mensagens SOAP (*Simple Object Access Protocol*), frequentemente transportadas usando o HTTP (*HiperText Transfer Protocol*) com serialização XML (*Extensible Markup Language*).

2.1.3 Android

Desenvolvido especialmente para dispositivos móveis como aparelhos celulares e tablets, o *Android* é uma plataforma composta de um sistema operacional, middlewares e um conjunto de aplicativos principais como os Contatos, Navegador de Internet e o Telefone propriamente dito. (MONTEIRO, 2012).

Baseado no kernel do Linux, o *Android* começou a ser desenvolvido em 2003 pela empresa *Android Inc*. Em 2005, a empresa foi comprada pela Google. Um acontecimento importante ocorreu em 2007, com a criação da Open Handset Alliance, que é uma organização de empresas de software, hardware e telecomunicações, no qual a missão é desenvolver uma plataforma para dispositivos móveis que seja completa, aberta e gratuita. O *Android* é *Open Source* (código aberto), Isso permite que qualquer um tenha acesso ao código-fonte e possa contribuir com o projeto *Android*.

2.1.4 REST (Representational State Transfer)

Segundo SAUDATE (2013), o REST é uma arquitetura de *Web Services* que foi desenvolvida em uma tese de doutorado de Roy Filding, co-autor do protocolo HTTP. O REST utiliza os métodos do protocolo HTTP, utilizando das boas práticas dessa tecnologia para garantir a comunicação entre os recursos da aplicação.

Uma das características do Rest é possuir uma única URL (*Universal Resource Locator*) para acesso dos recursos, em Rest todo endereçamento é gerenciado pelo uso de URIs (*Universal Resource Identifier*), toda requisição deve conter a URI do objeto que se está solicitando a informação. Para definir qual método a aplicação deverá executar é necessário que seja enviado no cabeçalho da solicitação o tipo de requisição realizada, que podem ser:

- GET: O método é uma operação apenas de leitura que obtém informações de um determinado recurso, caso informado algum parâmetro de consulta, retornará as informações filtradas.
- PUT: O método é utilizado para atualizar informações de um recurso no servidor.
- POST: O método é utilizado para criar um recurso.
- DELETE: Executa o método para remover um recurso.

O Rest possibilita que informações sejam obtidas em XML ou JSON, sendo que o uso de cada uma deve ser estudado e definido de acordo com a necessidade do ambiente (SAUDATE, 2013).

2.2 HTTP

O protocolo HTTP (*HyperText Transfer Protocol* - Protocolo de Transferência de Hi-pertexto) surgiu em 1996 após um trabalho em conjunto de Roy Fielding e Henrik Frystyk Nielsen, tal trabalho levou os autores a publicarem uma RCF (*Request for Comments*) onde o protocolo é descrito e documentado. O HTTP é um protocolo de camada de aplicação (baseado no modelo OSI). (SAUDATE, 2013)

O protocolo HTTP possui um modelo simples, consiste-se basicamente em um modelo de requisições (*request*) e respostas (*response*). O cliente estabelece uma conexão com o servidor, realiza uma requisição em seguida recebe uma resposta do servidor. No protocolo existem dois tipos de mensagens definidas, uma representa as requisições do lado do cliente e a outra as respostas do lado do servidor, ambas seguem padrões definidos pelo protocolo.

Uma requisição HTTP contém um ou mais cabeçalhos. As requisições HTTP esperam o retorno de documentos HTML, XML ou JSON do servidor. Estes documentos são compactados, não sofrem nenhum tipo de tratamento ao serem enviados do servidor para o cliente, tornando o HTTP viável em termos de consumo de banda. (TORRES, Gabriel. Redes de Computadores, 2014)

2.2.1 Métodos HTTP

- GET
- POST
- PUT
- DELETE
- OPTIONS
- HEAD
- TRACE
- CONNECT

2.2.2 Código de Status

As requisições enviadas para o servidor retornam um código de status. Esses Códigos são divididos em cinco famílias: 1xx, 2xx, 3xx, 4xx e 5xx, sendo:

- 1xx – Informacionais
- 2xx - Códigos de sucesso
- 3xx - Códigos de redirecionamento
- 4xx - Erros causados pelo cliente
- 5xx - Erros originados no servidor

Tabela 2 – Código de Status HTTP por Família

Família	Status-Código	Descrição
1x	100	Continue
	101	Switching Protocols
2x	200	OK
	201	Created
	202	Accepted
	203	Non-Authoritative Information
	204	No Content
	205	Reset Content
	206	Partial Content
3x	300	Multiple Choices
	301	Moved Permanently
	302	Found
	303	See Other
	304	Not Modified
	305	Use Proxy
	307	Temporary Redirect
4x	400	Bad Request
	401	Unauthorized
	402	Payment Required
	403	Forbidden
	404	Not Found
	405	Method Not Allowed
	406	Not Acceptable
	407	Proxy Authentication Required
	408	Request Time-out
	409	Conflict
	410	Gone
	411	Length Required
	412	Precondition Failed
	413	Request Entity Too Large
	414	Request-URI Too Large
	415	Unsupported Media Type
	416	Requested range not satisfiable
	417	Expectation Failed
5x	500	Internal Server Error
	501	Not Implemented
	502	Bad Gateway
	503	Service Unavailable
	504	Gateway Time-out
	505	HTTP Version not supported

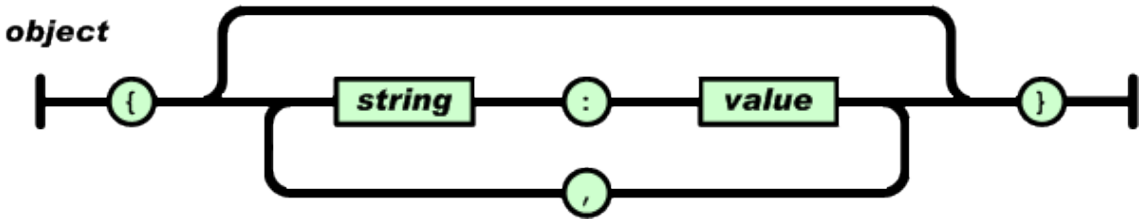
Fonte: RCF (*Request for Comments*), 2016

2.1.5 JSON

O JSON (*JavaScript Object Notation*) é conhecido por seu alto desempenho na troca de dados e foi criado por Douglas Crockford. É utilizado como alternativa ao XML, pois utiliza pouca banda em transferências, ao contrário do XML. (JSON.org, 2015)

O JSON possibilita a troca de informações entre aplicações sem que haja perdas ou lentidões. Sua estrutura é de fácil compreensão e conta com várias ferramentas para interpretação, pode representar quatro tipos de dados primários (strings, números, booleanos e nulos) e dois tipos estruturados (objetos e vetores). O JSON foi desenvolvido com o objetivo de ser simples, portável e textual. (SAUDATE, 2013)

Figura 1 - Representação de um objeto JSON



Fonte: JSON.org, (Setembro de 2016)

2.1.6 Apache Tomcat

Para o desenvolvimento do Web Service foi necessária a utilização do servidor Apache Tomcat v7.0. O Apache Tomcat é um servlet container, que implementa a JSR tal (servlet e JSP 3.0), é fornecido sob licença da Apache Segundo PAMPLONA (2010) o Tomcat é um container para JSP e Servlets muito conhecido e utilizado.

2.1.7 JERSEY

Framework utilizado para abstrair os detalhes de baixo nível da comunicação cliente-servidor, a fim de simplificar o desenvolvimento de serviços Web RESTful e seus clientes em Java. O framework RESTful Web Jersey Services é *open source*, disponibiliza uma estrutura para desenvolver serviços Web RESTful em Java que fornecem suporte para JAX-RS APIs e serve como um JAX-RS (JSR 311 e JSR 339) Implementação de Referência. Além de Jersey ser uma referência de implementação JAX-RS, fornece sua própria API que estende o KIT de ferramentas JAX-RS, com utilitários e ferramentas que simplificam ainda mais a implementação de um serviço Web RESTful. (JERSEY, 2015)

2.1.8 ERP

ERP é um sistema que integra os dados e processos de uma organização. Sua arquitetura é distribuída em módulos (sistemas de finanças, contabilidade, recursos humanos, fabricação, marketing, vendas, compras), é uma plataforma desenvolvida para integrar todos os departamentos da organização alimentados por uma base de dados principal (PADILHA, 2005).

2.1.9 JAVA

Java é uma linguagem de programação interpretada e orientada a objetos. Foi desenvolvida na década de 90 por uma equipe coordenada por James Gosling na organização Sun Microsystems. Java é uma linguagem interpretada que é compilada para *bytecode*, que por sua vez é interpretado por uma JVM (*Java Virtual Machine*). (DEITEL, 2003).

No meio empresarial e corporativo, a cultura do Java vem se expandindo de uma forma gradual e constante, permitindo que diversas empresas desenvolvam aplicações utilizando esta linguagem. (DEITEL, 2003).

3. DESENVOLVIMENTO DO PROJETO

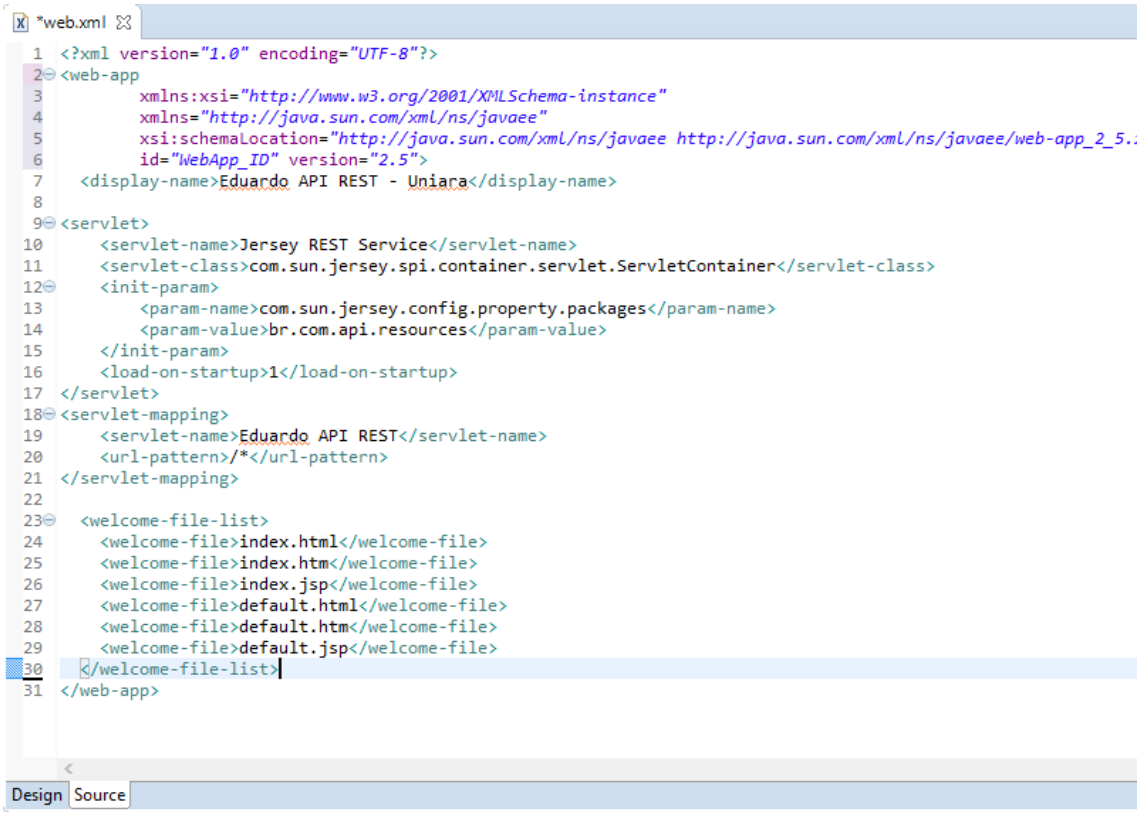
3.1.1 Desenvolvimento do Web Service

Para o desenvolvimento foi necessário criar um *Web Dynamic Project* na IDE Eclipse EE. Foram adicionadas 4 bibliotecas Jersey ao projeto para possibilitar o desenvolvimento dos *Web Services,* são elas:

- asm.jar
- jersey-core
- jersey-server
- jsr311-api

Em seguida foi configurado o *Web Container* apache Tomcat. Após as configurações iniciais realizadas foi necessário configurar o arquivo web.xml, onde foi informado o *servlet* (Tomcat) e o pacote onde os recursos foram criados.

Figura 2 - Configuração do arquivo web.xml.



Fonte: Próprio autor

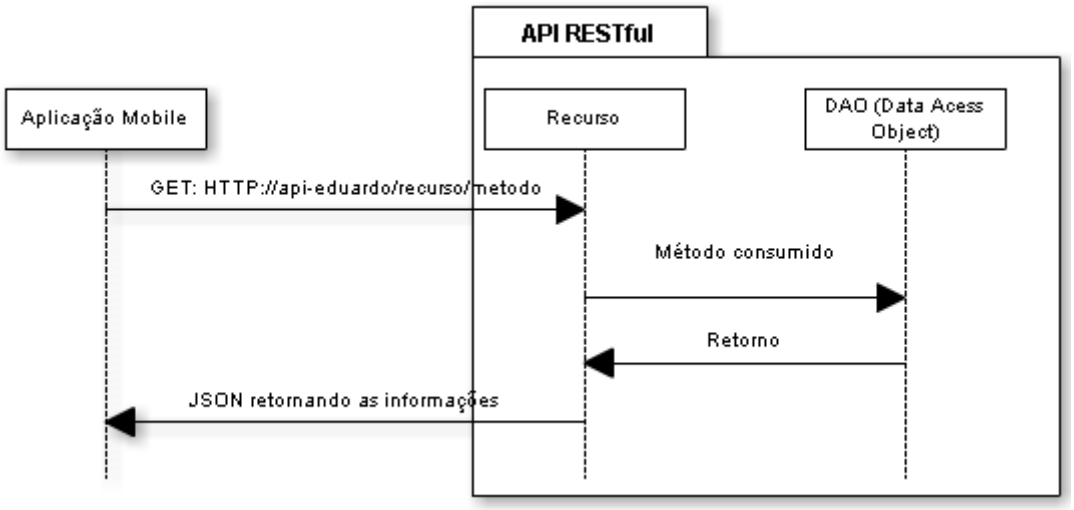
3.1.2 Recursos

Para atender as necessidades deste projeto foi criado o recurso: VendasRES.O recurso de vendas possui métodos que retornam informações de vendas registradas na base de dados.

Os métodos que o recurso VendasRES disponibiliza são:

- ProdutosMaisVendidosPeriodo (String Data_Inicial, String Data_Final, int Numero_Registros)
  Lista um ranking dos produtos mais vendidos dentro do período passado nos parâmetros Data_Inicial e Data_Final
- TotalVendidoPeriodo (String Data_Inicial, String Data_Final)
  Lista todas as vendas dentro do período definido nos parâmetros Data_Inicial e Data_Final.
- TotalVendasCanceladas (String Data_Inicial, String Data_Final)
  Lista o total de vendas que foi cancelada no período passado nos parâmetros Data_Inicial e Data_Final.

Figura 3- Diagrama de sequência da comunicação entre aplicação *Android* e a API Rest.



Fonte: Próprio autor

3.1.3 Aplicação Mobile

A aplicação mobile conta com três gráficos: Produtos mais vendidos no período, Vendas no Período e Quantidade de vendas Canceladas no Período.

3.1.4 Gráficos

Os gráficos serão plotados utilizando a biblioteca GraphView. GraphView é uma biblioteca de código livre que permite a criação de forma rápida e fácil de gráficos além de ser fácil de compreensão. Para utilizar a biblioteca na aplicação foi necessário adicionar sua referência as dependências do Gradle.

Figura 4- Dependência da biblioteca GraphView que foi adicionada ao projeto.

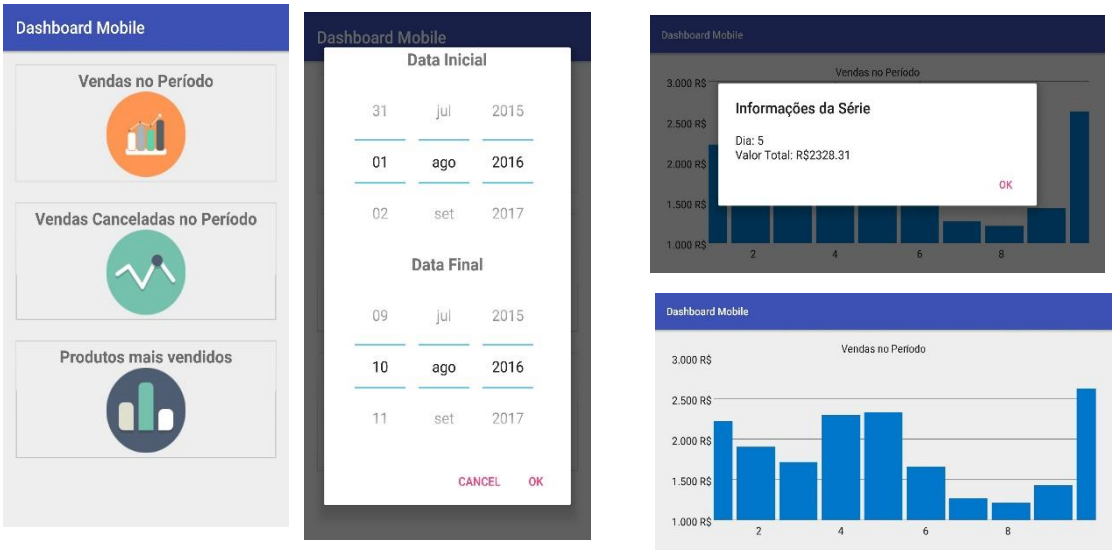
```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile files('libs/GraphView-4.2.1.jar')
    compile 'com.android.support:appcompat-v7:24.2.1'
    compile 'com.google.code.gson:gson:2.2.4'
    compile 'org.springframework.android:spring-android-rest-template:2.0.0.M3'
    compile 'com.android.support:support-v4:24.2.1'
    testCompile 'junit:junit:4.12'

}
```

Fonte: Próprio autor

Os gráficos disponibilizados pela biblioteca são: Line Graphs, Bar Graphs e Point Graphs. No projeto foi utilizado apenas o Gráfico de Barras (Bar Graph).

3.1.5 Activitys

Figura 5 - Telas da aplicação mobile.



Fonte: Próprio autor

3.1.6 Requisições ao Web Service.

Os recursos serão obtidos através de uma chamada assíncrona ao *web service*. A chamada é realizada pela classe HttpRequestTask, está classe estende a classe AsyncTask. A classe HttpRequestTask foi construída de forma genérica permitindo que o retorno JSON do recurso solicitado seja desserializado sem a necessidade de uma nova implementação para cada recurso desejado.

Figura 6 – Classe assíncrona genérica responsável por consumir e desserializar o recurso do Web Service.

```
public class HttpRequestTask<T> extends AsyncTask<String, Void, T> {
    private Class<T> type;
    private ProcessResponse<T> delegate;

    public HttpRequestTask(Class<T> classType, ProcessResponse<T> f)
    {
        type = classType;
        delegate = f;
    }
    @Override
    protected T doInBackground(String... urls) {
        T result = null;
        try {
            //Instância do RestTemplate
            RestTemplate restTemplate = new RestTemplate();
            //Qual será o tipo de conversão
            restTemplate.getMessageConverters().add(new StringHttpMessageConverter());
            // desserializa a resposta.
            result = restTemplate.getForObject(urls[0], type);
            return  result;
        }catch (Exception ex){
            return null;
        }
    }

    protected void onPostExecute(T response)
    {
        delegate.process(response);
    }
}
```

Fonte: Próprio Autor

Para consumirmos um recurso no *Web Service* precisamos definir uma classe de retorno, esta classe deve possuir atributos que correspondam ao retorno em JSON, os atributos devem possuir a anotação @SerializedName("nome_do_atributo") para que a desserialização possa ocorrer. A desserialização é realizada por métodos do Spring Framework for Android.

A chamada deve seguir o padrão new HttpRequestTask<EntidadeRetorno>(EntidadeRetorno.class,this).execute(url);

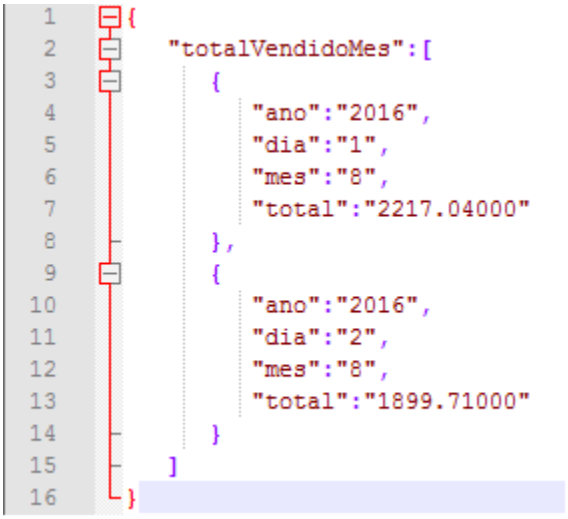
- EntidadeRetono: representa a entidade que será retornada ao consumir o recurso.

- EntidadeRetorno.class: Representa o tipo da classe, recurso utilizado para realizar a desserialização do objeto JSON.
- url: URL contendo o caminho do servidor, porta, api a ser acessada e o método a ser consumido seguido de parâmetros, caso houver.

Para acessar determinado recurso teríamos que fazer a seguinte requisição HTTP http://host:porta/projeto/recurso/metodo/parametros

Por exemplo, para obtermos todas as vendas de um determinado período temos que requisitar a seguinte URL: http://localhost:8080/api-eduardo/venda/listarTodosPeriodo/08-01-2016/08-02-2016. Esta requisição irá retornar o objeto JSON, para este caso, o JSON retornado foi o seguinte:

Figura 7 - JSON retornado ao consumir o método /listarTodosPeriodo



Fonte: Próprio autor.

O objeto JSON acima será desserializado para a classe exibida na imagem abaixo:

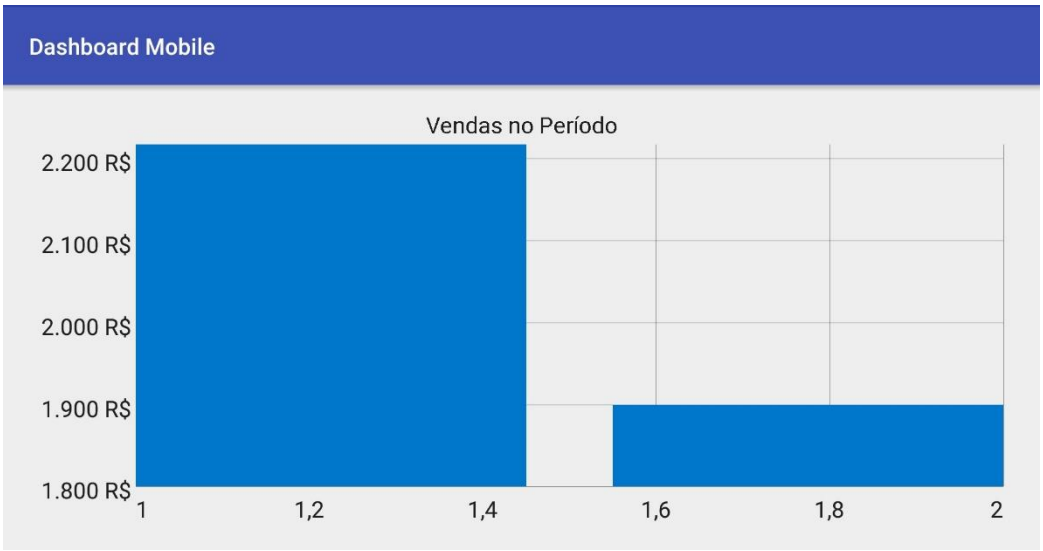
Figura 8 - Classe utilizada na desserialização do objeto JSON retornado

```java
public class Vendas {
    @SerializedName("mes")
    private int Mes;
    @SerializedName("ano")
    private int Ano;
    @SerializedName("total")
    private BigDecimal Total;
    @SerializedName("dia")
    private int Dia;
    ...
    //GET/SETs
    ...
}
```

Fonte: Próprio autor.

Os dados obtidos serão apresentados no gráfico abaixo:

Figura 9 – Gráfico obtido na requisição HTTP http://localhost:8080/api-eduardo/venda/listarTodosPeriodo/08-01-2016/08-02-2016



Fonte: Próprio autor

Ao clicar sobre a segunda barra, será exibido o dia e o valor total que foi vendido no dia.

Figura 10 – Informações detalhadas da Série.

Fonte: Próprio autor.

4. CONCLUSÃO

O objetivo deste trabalho foi desenvolver uma API que realizasse o processo de requisição a um servidor REST para possibilitar a integração de aplicações desenvolvidas em *Android* e Web Services aplicando os princípios REST. O desenvolvimento da API foi feito utilizando a linguagem Java, foram criadas classes e interfaces quem possibilitaram o encapsulamento dos principais métodos HTTP. No desenvolvimento da aplicação *Android* foi possível notar que a plataforma pode ser facilmente utilizada para integrar e interagir com sistemas distintos e realizar a troca de informações de maneira ágil, além de disponibilizar uma documentação bem elaborada, completa e de fácil compreensão. Notou-se também o crescente interesse das empresas em elaborarem Web Services para que seus sistemas fiquem prontos para interagir com outras aplicações.

5. REFERÊNCIAS BIBLIOGRÁFICAS

APACHE TOMCAT. Site Oficial do ApcaheTomcat. Disponível em: <http://tomcat.apache.org/>. Acesso em: 09 de Outubro 2015.

BURKE, B. RESTful Java with JAX-RS, 1Ed. Sebastopol: O'Reilly, 2009.

CAMPOS, J. RESTMB: API RESTful para Android. Disponível em < http://aberto.univem.edu.br/bitstream/handle/11077/988/Monografia%20Final.pdf?sequence=1>. Acessado em: 11 Outubro de 2015.

CHAPPELL, D;JEWELL, T. Java Web Services, 1Ed. Sebastopol: O'Reilly, 2002.

JERSEY. RESTful Web Services in Java Disponível em: < https://jersey.java.net/> acessado em: 15 de Outubro de 2015

JSON. Introducing JSON. Disponível em: <http://www.json.org/> acessado em: 15 de Outubro de 2015

KALIN, M.Java Web Services Up and Running, 1Ed. Sebastopol: O'Reilly, 2009.

GEORGAKOPOULOS, D; PAPAZOGLOU, M. Service-oriented computing: Introduction. Communications of the ACM, THE MIT PRESS, 2008.

MONTEIRO, J. B. Google Android Crie aplicações para celulares e tablets, 1. ed. Casa do Código 2012. 312 p.

PAMPLONA, V.F. Web Services. Construindo, disponibilizando e acessando Web Services via J2SE e J2ME, 2010. Disponível em: <http://javafree.uol.com.br/artigo/871485/Web-Services-Construindo-isponibilizandoe-acessando-Web-Services-via-J2SE-eJ2ME.html>. Acesso em: 20 Outubro de 2015.

SAUDATE, A. REST. Construa API's inteligentes de maneira simples, 1. ed. Casa do Código 2013. 101 p.

Smartphone OS Market Share, 2015 Q2. Disponível em: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Acessado em: 09 de Outubro de 2015

ZANCUL, E; ROZENFELD, H. Sistemas ERP. Disponível em <https://www.ipen.br/biblioteca/cd/conem/2000/CC8808.pdf>. Acessado em: 24 Outubro de 2015.

MAURÍCIO, C; NAKAMURA, E. Computação Móvel: Novas Oportunidades e Novos Desafios, T&C Amazônia, Ano 1, noº2, Junho de 2003.

TORRES, G. Redes de Computadores, 2. ed. Nova Terra 2014.

PADILHA, T.C.C; MARINS, F.A.S. Sistemas ERP: características, custos e tendências. Revista Produção, v. 15, n. 1, p. 102-113, 2005

DEITEL, P; DEITEL, P. Java – Como Programar, 4ª Edição; Editora Bookman.

BELLON, G. Desenvolvendo para *Android* - Introdução. 15 Set 2011. Disponível em: <http://www.mestre*Android*.com.br/desenvolvendo-para-*Android*introducao>. Acesso em 11 Outubro 2015.

IDC. Smartphone OS Market Share, 2016 Q2. Disponível em: < http://www.idc.com/prodserv/smartphone-os-market-share.jsp> acessado em: 15 de Outubro de 2016