UFMT

UNIVERSIDADE FEDERAL DE MATO GROSSO

INSTITUTO DE COMPUTAÇÃO

ESPECIALIZAÇÃO EM ENGENHARIA WEB E GOVERNO ELETRÔNICO

PROPOSTA DE DISPONIBILIZAÇÃO DE DADOS SOBRE AS UNIDADES BÁSICAS DE SAÚDE

HELTON BATISTA MARINHO

CUIABÁ – MT

2016

UFMT

UNIVERSIDADE FEDERAL DE MATO GROSSO

INSTITUTO DE COMPUTAÇÃO

ESPECIALIZAÇÃO EM ENGENHARIA WEB E GOVERNO ELETRÔNICO

PROPOSTA DE DISPONIBILIZAÇÃO DE DADOS SOBRE AS UNIDADES BÁSICAS DE SAÚDE

HELTON BATISTA MARINHO

Orientador: Prof. Dr. Nielsen Simões

Monografia apresentada ao Curso de Especialização em Engenharia Web e Governo Eletrônico do Instituto de Computação da Universidade Federal de Mato Grosso, como requisito para conclusão do Curso de Pós-Graduação *Lato Sensu* em Engenharia Web e Governo Eletrônico.

CUIABÁ – MT

2016

UFMT

UNIVERSIDADE FEDERAL DE MATO GROSSO

INSTITUTO DE COMPUTAÇÃO
ESPECIALIZAÇÃO EM ENGENHARIA WEB E GOVERNO ELETRÔNICO

CERTIFICADO DE APROVAÇÃO

TÍTULO: **PROPOSTA DE DISPONIBILIZAÇÃO DE DADOS SOBRE AS UNIDADES BÁSICAS DE SAÚDE**.

AUTOR: **HELTON BATISTA MARINHO**

Aprovada em 09/12/2016

Prof. Dr. Msc. Nielsen Simões
IC/UFMT (Orientador)

Prof. Dr. Msc. Jivago Medeiros Ribeiro
IC/UFMT

Prof. Dr. Nelcileno de Souza Araújo
IC/UFMT

DEDICATÓRIA

Dedico este trabalho

a todos aqueles que acreditaram em meu potencial

e para todos que me apoiaram direto ou indiretamente.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois sem ele eu não conseguiria chegar a lugar nenhum. Agradeço também todos os colegas e amigos que acompanhou a minha luta para chegar até aqui. Ficam aqui os meus agradecimentos ao professor Nielsen Simões que esteve me orientando em meios aos meus desaparecimentos.

RESUMO

Neste trabalho será apresentado a Lei de Acesso a informação e a maturidade de publicação de dados, mostraremos alguns conceitos de *Web Services* utilizando o protocolo SOAP e REST, apresentaremos um breve conceito sobre o protocolo HTTP e as novidades da linguagem Java 8 com sua expressão Lambda e utilização do Stream, por fim será desenvolvido uma ferramenta, juntamente com seu manual de utilização, que irá fornecer informações que estão disponíveis por meio do arquivo de dados abertos governamentais no sitio http://dados.gov.br/ das Unidades Básicas de Saúde – UBS. Estas informações serão disponibilizadas em diferentes formatos como XML, JSON e CSV. Para isso, serão utilizados os conceitos de computação orientada a serviços, para deixar as informações dinâmicas e possibilitando a integração com diferentes aplicações, seja ela para dispositivos móveis, web ou desktop.

Palavas-chave: Dados Abertos, Web Services, Serviços, Java 8, Lambda, Stream.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

XML	Unified Modeling Language
UBS	Unidades Básicas de Saúde
MER	Modelo Entidade-Relacionamento
SOAP	Simple Object Access Protocol
API	Application Programming Interface
INDA	Infraestrutura Nacional de Dados Abertos
LAI	Lei de Acesso à Informação
JSON	JavaScript Object Notation
CSV	Comma-separated values
HTTP	HyperText Transfer Protoco
CNES	Cadastro Nacional de Estabelecimentos de Saúde
REST	Representational State Transfer
URI	Universal Resource Identifier
URL	Universal Resource Locator
UFMT	Universidade Federal de Mato Grosso
XML	Unified Modeling Language
UBS	Unidades Básicas de Saúde

1

INTRODUÇÃO

Neste capitulo será apresentado à contextualização da lei de acesso a informação, bem como as dificuldades na obtenção das informações produzidas pelo governo. Também serão apresentados os objetivos que este trabalho almeja alcançar, as justificativas, as metodologias e sua breve organização.

1 Apresentação

A partir da lei no 12.527 de 18 de outubro de 2011 (BRASIL, 2011), também conhecida como lei de acesso à informação (LAI), é possível acessar informações disponibilizadas por diversos órgãos públicos. Entretanto, essa informação é disponibilizada de forma não padronizada, o que dificulta a reutilização desses dados pelo cidadão. Para que esses dados possam ser utilizados de forma eficiente, é necessário que haja uma padronização antes de sua disponibilização.

DE MENDONÇA et al. (2015), em seu trabalho, mostrou a dificuldade encontrada em obter os dados sobre Aedes aegypti, o que poderia ser evitado se o governo prontificasse em atender de forma adequada a lei. A reutilização pode ser simplificada de forma a disseminar o conhecimento e incentivar o seu consumo.

Muitas pessoas estariam aptas a desenvolver novos serviços, por exemplo, o acesso aos dados dos Aedes aegypti se estivesse de fácil acesso, sem a necessidade de gerar um documento para conseguir acesso ló, mas por essa disponibilização ser em formato proprietário, ou seja, acessos restritos e em formatos utilizados somente pelo órgão gestor, dificultam a reutilização limitando assim a criatividade e retraindo o cidadão a

não exercer o seu direito, desta forma o órgão gestor guarda para si as informações que deveriam ser da sociedade.

A dificuldade de encontrar facilmente informações referentes a unidades básicas de saúde é a principal motivação para este trabalho. Um cidadão que não conheça bem uma determinada cidade ou região encontra uma grande dificuldade de obter as informações necessárias para, quando necessário, encontrar uma unidade básica de saúde, principalmente sua localização. A disponibilização das informações de forma estruturada permite não somente que um cidadão possa usufruí-la de forma eficiente quanto à integração dos dados com outros sistemas, abrindo uma gama de possibilidades para que novas informações possam ser geradas.

A representação pode ser de várias, como RDF, JSON, XML e o próprio CSV. O RDF é utilizando em projetos Linked Open Data, pois ele é facilmente interpretado pelas máquinas. O JSON é uma representação fácil de ser interpretado pelas linguagens de programação, já o XML é amplamente utilizado nas trocas de dados entre operações distintas, pois ela permite realizar documentação na própria estrutura sem que interfira na interpretação dela. O CSV até pode ser utilizado, pois também é uma representação compacta, que consegue transportar grandes conjuntos de dados, porem requer uma boa documentação para dar significado a cada coluna dos dados.

Este trabalho apresenta uma proposta para padronização de dados públicos das unidades básicas de saúde. Para a disponibilização desses dados, seria excelente que isso acontecesse em forma de serviço web e algumas dessas formas, é a construção de *web services* baseados nas metodologias de implementação SOAP e REST.

1.1 Objetivo Geral

O objetivo principal deste trabalho é desenvolver uma ferramenta que disponibiliza as informações das unidades básicas de saúde de forma a facilitar o acesso aos dados, essas obtidas por meio dos dados abertos governamentais buscando centralizar o acesso, facilitando a disponibilização das informações, automatizando o processo de busca,

promovendo a integração com outros serviços e para atender melhor a lei de acesso à informação.

1.2 Objetivos Específicos

Para atingir o objetivo, devem ser executadas as seguintes etapas:

1.2.1 Pesquisar os conceitos de *web services*;

1.2.2 Definir as tecnologias necessárias para a execução do projeto;

1.2.3 Obter os dados das unidades de saúde para padronização e disponibilização;

1.2.4 Analisar os tipos de consultas e definir as respostas;

1.2.5 Elaborar o manual básico de utilização;

1.2.6 Utilizar os conceitos de *web services* para a construção de uma API.

1.3 Justificativa

As informações do serviço público ainda são publicadas em formatos proprietários ou de modo que impeçam que sejam acessíveis a todas as partes interessadas (DINIZ, Vagner; 2010). As Unidades Básicas de Saúde (UBS) são a porta de entrada preferencial do Sistema Único de Saúde, já que os principais objetivos é a descentralização dos atendimentos nos hospitais e deixar o cidadão mais próximo dos serviços públicos. Portanto, é fundamental que o cidadão consiga encontrar rapidamente uma UBS mais próxima de sua localidade em qualquer lugar que ele se encontre.

A disponibilização dos dados das UBS de forma automatizada e em forma de serviço *web* facilita o uso das informações dos dados abertos governamentais. Dessa forma, amplia-se o atendimento à lei e os princípios dos dados abertos, possibilitando a reutilização dos dados e promovendo a integração com diversos outros sistemas, permitindo inclusive o desenvolvimento de novos sistemas.

1.4 Metodologia

Este trabalho foi elaborado por meio de pesquisas bibliográficas, utilizando revistas eletrônicas, sites na Internet, entre outras fontes confiáveis pertinentes ao tema a ser trabalhado. Realizou-se também experimentos buscando comparar as tecnologias existentes para desenvolver uma melhor solução.

As principais informações utilizadas das UBS estão disponíveis no site http://dados.gov.br. Neste arquivo possuímos os seguintes atributos: latitude, longitude, código do município, código CNES, nome do estabelecimento, endereço, bairro, cidade, telefone, estrutura física da ambiência, adaptação para deficiente físico e Idoso, situação dos equipamentos e situação dos Medicamentos.

Considerando os atributos "a estrutura física da ambiência", "adaptação para deficiente físico e idoso", "situação dos equipamentos" e "situação dos medicamentos", essas informações estão definidas diretamente no arquivo e classificadas como se segue:

- Desempenho mediano ou um pouco abaixo da média
- Desempenho acima da média
- Desempenho muito acima da média

1.5 Organização do Trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica para o tema abordado, enquanto o Capítulo 3 descreve como a ferramenta para padronização dos dados das UBS foi desenvolvida. No Capítulo 4 é apresentada uma pequena utilização da ferramenta. Já o Capítulo 5 elenca as principais contribuições deste trabalho e alguns dos possíveis trabalhos futuros a serem desenvolvidos.

2

FUNDAMENTAÇÃO TEÓRICA

Neste capitulo será elencadas as leis de acesso à informação e os princípios dos dados abertos governamentais, as tecnologias utilizadas para comparação e concepção da API que disponibilizara os dados das UBS. Serão apresentadas algumas tecnologias estudadas para o desenvolvimento da ferramenta. São apresentados também os principais conceitos de API e SOAP, e alguns exemplos de utilização de *web services* SOAP e Rest.

Segundo a LAI, Lei de acesso à informação (BRASIL, 2011):

> *Art. 5º: É dever do Estado garantir o direito de acesso à informação, que será franqueada, mediante procedimentos objetivos e ágeis, de forma transparente, clara e em linguagem de fácil compreensão.*

Portanto, o Estado deve permitir que qualquer pessoa tenha acesso as informações de qualquer órgão ou entidade pública de seu interesse. Estas informações já se encontram disponíveis em diversos meios de comunicação, porém sua utilização ainda é dificultada devido à maneira que estão disponíveis, pois se apresentam de forma despadronizada ou pouco automatizada. Para utilizar estes dados é necessário acessar os endereços eletrônicos de tais plataformas e realizar, de forma manual, o download do arquivo

desejado e então realizar a consultas aos dados de forma não convencional, sem muitas possibilidades.

Segundo ABINADER e LINS (2006), várias tecnologias surgiram nos últimos anos destinadas ao desenvolvimento de aplicações para Internet, porém elas possuem algumas particularidades e são limitadas, impedindo a troca de informações de forma automatizada. No portal Brasileiro de Dados Abertos, existem três leis sobre os dados abertos e oito princípios que foram acordados por um grupo de trinta pessoas, são elas:

- Se o dado não pode ser encontrado e indexado na Web, ele não existe.
- Se não estiver aberto e disponível em formato compreensível por máquina, ele não pode ser reaproveitado.
- Se algum dispositivo legal não permitir sua replicação, ele não é útil.

Estes princípios foram definidos da seguinte forma: eles precisam ser disponibilizados de forma completa, ser dados primários o mais transparente possível, que sejam atuais, acessíveis, que possam ser processados por máquina com acesso não discriminatório, em formatos não proprietários e livres para acessa-los.

Existem também cinco motivos para incentivar a abertura dos dados, os quais são:

- Transparência na gestão pública.
- Contribuição da sociedade com serviços inovadores ao cidadão.
- Aprimoramento na qualidade dos dados governamentais.
- Viabilização de novos negócios.
- Obrigatoriedade por lei.

BERNERS-LEE (2009), encorajando as pessoas, principalmente os proprietários de dados do governo, foi desenvolvido um sistema de classificação por estrelas, onde basta o dado estar publico, sem licença para utilização e independente do formato, já receberá uma estrela. Na Tabela 1 está representando os níveis de cada estrela, que por sua vez, vem sendo adotado como modelo de maturidade de publicação de dados. As estrelas estão organizadas da menor para a maior, sendo que as maiores tende a respeitar as

ordenações menores. Na Figura 1 é apresentada uma escala da maturidade dos dados abertos proposto por BERNERS-LEE (2012).

Tabela 1 - 5 estrelas propostas por Berners-Lee [BERNERS-LEE (2009)]

	Disponível na web (qualquer formato), mas com uma licença aberta, a ser Open Data.
	Disponível como dados estruturados legíveis por máquina (por exemplo, Excel em vez de digitalização de uma imagem de uma tabela).
	Formato não proprietário (por exemplo, CSV ao invés de Excel).
	Usar URIs para identificar recursos e identificar as coisas, de modo que as pessoas podem apontar em seus produtos.
	Vincular seus dados para dados de outras pessoas para promover um contexto.

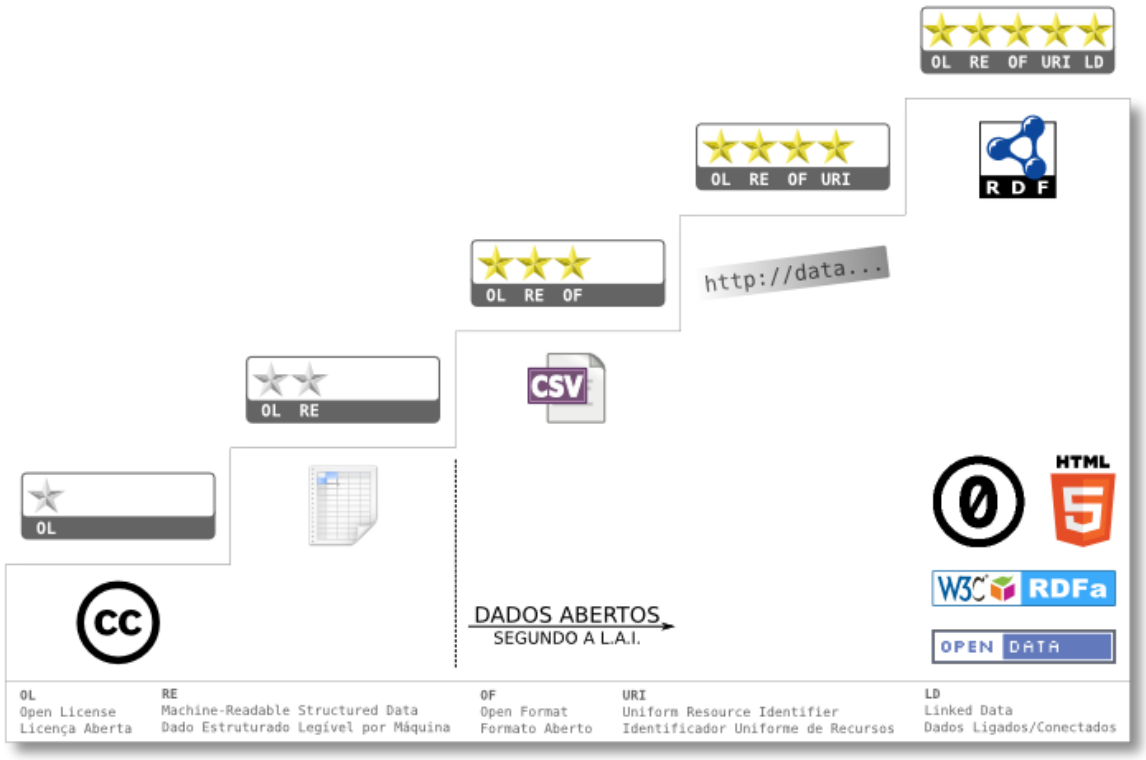

Figura 1 - Esquema de Implementação das 5 estrelas [BERNERS-LEE (2013)]

2.1 API

API (Application Programming Interface) em sua tradução literal (Interface de Programação de Aplicação), é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usa seus serviços. Desenvolver APIs exige bastante responsabilidade do desenvolvedor, tanto na qualidade, quanto na manutenção do seu estado, (Dias 2016).

Muitas empresas adquirem software de vários fornecedores e posteriormente querem que eles se comuniquem, nestes casos, os *Web Services* funciona como o elo entre eles, fazendo funcionar corretamente a comunicação entre eles. Nos últimos tempos vem surgindo vários tipos de tecnologias, como Smart TVs, Smartphones, Tablets, entre

outros, esses surgimentos impacta diretamente no dia a dia das empresas de tecnologia, pois não é nada fácil se adequar, pensando no melhor senário para cada dispositivo.

Com o surgimento de grandes meios tecnológicos e a necessidade de criar novas soluções para essas integrações, as APIs eram tratadas como artigos de luxos, agora passou a ter muito valor aos poderes executivos das grandes empresas como Twillo, *Twitter*, *Google* e *Facebook*, por exemplo, elas dependem fortemente de um sistema computacional aberto, onde haja interação via Web Services permitindo a interação com outros sistemas.

O Google maps possui vários tipos de serviços na web, basicamente cada serviço tem sua API disponível para facilitar sua integração. API de mapas estáticos, API do Street View, API para rotas entre outros. O Facebook por sua vez também possui vários serviços disponíveis, desde uma simples postagem, criação de jogos ou até permitir incorporar em uma pagina da web as funcionalidades como login, mensagens instantâneas, postagem, etc.

2.2 WEB SERVICES

Web services ou Serviço Web é uma arquitetura de comunicação entre software sejam da mesma plataforma ou não, essa comunicação é feito através de mensagens SOAP normalmente utilizando o protocolo HTTP, serializando em XML com outros padrões da WEB (W3C 2004).

Um serviço é caracterizado pelo seu conjunto de funcionalidades que fornece em nome do seu proprietário como empresa ou individuo, nesses casos precisa existir uma troca de mensagens entre o requisitante e um provedor. Essa troca de mensagem precisa ser acordada previamente, assim não terá nenhuma surpresa durante a troca. As trocas de mensagem são representadas pelo serviço WSDL, que por sua vez é a especificação entre o servidor e a interface web com serviço. Ele é responsável por definir os formatos das mensagens, tipos de dados, protocolo de transporte e a serialização a ser utilizado. O núcleo do serviço representa o acordo entre as partes interessadas.

(W3C 2004) mostra vários modelos de arquitetura, por exemplo, Modelo Orientado a Mensagem, Modelo Orientado a Serviço, Modelo Orientado a Recurso e Modelo Orientado a Politica. O Modelo Orientado a Serviço é o mais complexo das arquiteturas, serviços são realizados por meio de troca de mensagens entre os agentes solicitante e provedor, um serviço tem é a representação direta com o mundo real, pois a maioria das implementações são para resolver ou fornecer um funcionalidade para o mundo real, podemos modelar como proprietário de um serviço uma pessoa ou empresa responsável por ela no mundo real. Os modelos das arquiteturas estão representados na Figura 2.

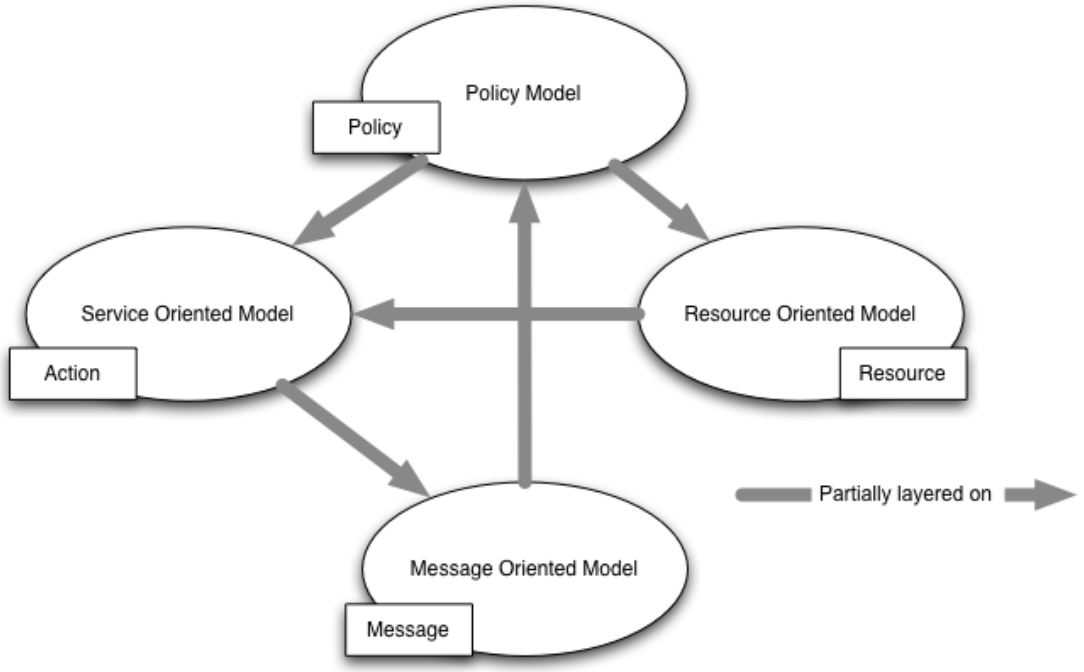

Figura 2 - Modelo da Arquitetura [Fonte: W3C (2004)]

2.2.1 SOAP

SOAP que significa Simple Object Access Protocol, que pode ser dito Protocolo de Simples Acesso a Objetos, é um protocolo de troca de mensagens entre aplicações, porem descentralizada, mas distribuído, tornando o utilizável por qualquer plataforma (W3C 2004).

SAUDATE (2012) afirma que SOAP é tratado como envelope, por conter os elementos cabeçalho e corpo, como no exemplo a seguir mostrado por ele representado na Figura 3.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ser="http://servicos.estoque.knight.com/"/>
   <!-- Aqui pode ter, ou não, um elemento soapenv:Header -->
   <soapenv:Body>
       <ser:listarLivros />
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 3 - Estrutura do XML SOAP [Fonte: SAUDATE (2012)]

Os *web Services* tradicionais são baseados em SOAP, e o trafego das informações são através de arquivos XML, porém podemos perceber que interagir com esse tipo de web Services não é tão prático, segundo SAUDATE (2013). As Figuras 4 e 5 apresentam um exemplo de consulta usando o XML envelopado e sua resposta respectivamente.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <listarClientes xmlns="http://brejaonline.com.br/administracao/1.0/service"
   </soap:Body>
</soap:Envelope>
```

Figura 4 - Consulta SOAP XML [Fonte: SAUDATE (2013)]

```
<soap:Envelope xmlns:domain="http://brejaonline.com.br/administracao/1.0/domain"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
       <listarClientesResponse xmlns="http://brejaonline.com.br/administracao/1.0.
          <domain:clientes>
             <domain:cliente domain:id="1">
                <domain:nome>Alexandre</domain:nome>
                <domain:dataNascimento>2012-12-01</domain:dataNascimento>
             </domain:cliente>
             <domain:cliente domain:id="2">
                <domain:nome>Paulo</domain:nome>
                <domain:dataNascimento>2012-11-01</domain:dataNascimento>
             </domain:cliente>
          </domain:clientes>
       </listarClientesResponse>
    </soap:Body>
</soap:Envelope>
```

Figura 5 - Resposta de uma consulta SOAP XML [Fonte: SAUDATE (2013)]

2.2.2 REST

SAUDATE (2013) define o REST como uma transparência de estado representativo, e é um estilo para desenvolvimentos de web services, a origem veio de uma tese de doutorado de Roy Fielding que também é coautor do protocolo HTTP, então o REST se guiou pelas boas praticas do protocolo HTTP, que são os usos adequados dos seus métodos, usos de URI's, usos de códigos para identificar erros ou falhas, uso adequado de cabeçalhos e interligações com outros recursos. Todos os dados são trafegados pelo protocolo HTTP. Ele ainda nos mostra como seria um web services usando o REST, no exemplo é apresentado como será uma consulta através de URL e o detalhamento de cada parte dela, conforme a Figura 6.

http://localhost:8080/cervejaria/clientes

- **http://** - Indica o protocolo que está sendo utilizado (no caso, HTTP);
- **localhost:8080** - Indica o servidor de rede que está sendo utilizado e a porta (quando a porta não é especificada, assume-se que é a padrão - no caso do protocolo HTTP, 80);
- **cervejaria** - Indica o **contexto** da aplicação, ou seja, a raiz pela qual a aplicação está sendo fornecida para o cliente. Vou me referir a esta, daqui em diante, como **contexto da aplicação** ou apenas **contexto**;
- **clientes** - É o endereço, de fato, do recurso - no caso, a listagem de clientes. Vou me referir a este, daqui em diante, como **endereço do recurso**.

Figura 6 - Detalhamento da utilização do REST [Fonte: SAUDETE (2013)]

Acessando a mesma URL utilizada no exemplo anterior para uma consulta em um serviço utilizando o a tecnologia REST poderemos obter a seguinte resposta conforme mostrado na Figura 7.

```
<clientes>
   <cliente id="1">
      <nome>Alexandre</nome>
      <dataNascimento>2012-12-01</dataNascimento>
   </cliente>
   <cliente id="2">
      <nome>Paulo</nome>
      <dataNascimento>2012-11-01</dataNascimento>
   </cliente>
</clientes>
```

Figura 7 - Resposta de uma consulta REST [Fonte: SAUDATE (2013)]

Embora essas técnicas sendo adequadas para implementar *web services*, SOAP é menos performático do que REST, principalmente devido ao detalhamento das comunicações de SOAP que usa o envelope XML, aumentando o tempo para analisar as mensagens de resposta consequentemente aumentando o tráfego de dados.

Para uma arquitetura ser REST, ela precisa seguir as definições de interface uniforme, que consistem em definir um escopo entre cliente e servidor, assim simplificamos a arquitetura e deixamos em modo desacoplado, possibilitando assim o desenvolvimento de outras aplicações independentes. Ainda é de responsabilidade da Interface padrão a definição do recuso utilizando URI. A URI é o identificador do recurso a ser acessado, podemos acessar esse recurso passando parâmetros com query string ou acessando pelo caminho, falaremos mais sobre esses parâmetros na sessão do protocolo HTTP.

A arquitetura ainda precisa seguir as definições de Independência de estado da aplicação, Cache, Cliente-Servidor, Sistema em camadas e Código sob demanda.

3

CONSTRUÇÃO DA API-UBS

Este capítulo descreve alguns conceitos tecnológicos para construção da ferramenta para disponibilização e seu desenvolvimento, a ferramenta será chamada de API-UBS. Iremos mostrar desde a forma como conseguimos obter os dados até a disponibilização dentro da API-UBS.

3.1 TECNOLOGIA UTILIZADA

Nesta sessão será apresentadas um breve conceito sobre o protocolo HTTP e das novas atualizadas da linguagem Java 8 com seus métodos utilizando expressões lambda juntamente com o Stream.

3.1.1 HTTP

Hypertext Transfer Protocol ou HTTP, que em português significa Protocolo de Transferência de Hipertexto que como o próprio nome diz, é um protocolo de comunicação que transporta textos em blocos, ele é a base para a comunicação na internet.

Este protocolo possui dois grupos essenciais de mensagens, são eles: Mensagens Request, que é basicamente a requisição realizada pelo cliente, e as mensagens response, que são as respostas para as requisições processado pelo servidor conforme a Figura 8.

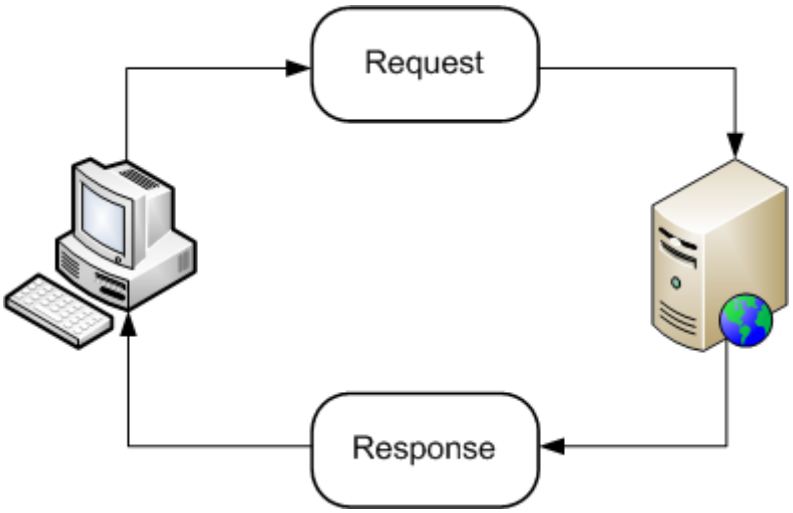

Figura 8 - Exemplo de uma requisição e sua Resposta

O protocolo HTTP possui vários métodos, são eles: GET, POST, HEAD, PUT, DELETE, OPTIONS e TRANCE.

O método GET realiza a solicitação de um recurso para o servidor através de uma URI, por sua vez, os parâmetros devem ser codificadas nela para que o servidor possa o identificar. Para entendermos melhor como funcionam os parâmetros, Gonçalves trata de maneira bem simples e explicativa, para fazer a solicitação da requisição usando o método GET usando apenas uma "/" indica que queremos acessar diretamente na raiz, como no exemplo "http://www.ufmt.br/ " , caso quiser acessar um caminho especifico basta adicionar no final, e ficaria "http://www.ufmt.br/ufmt/site/" . Se for acessar um endereço que possui recursos para serem acessados, podemos acessar utilizando query string. A estrutura pode utilizar alguns caracteres especiais como "?", "%20", "&", "=" entre outros. O "?" nos diz que teremos alguns parâmetros, o "%20" nos navegadores serão trocados por espaço, o "&" significa ter mais de um parâmetro na mesma URL, o sinal de "=" serve para comparar atribuir uma variável ao seu respectivo valor do parâmetro.

As respostas do protocolo tem uma linha de status assim como o corpo da resposta, a

Figura 9 está à representação de uma requisição GET e a resposta com o status de sucesso, mostrado por um navegador ao acessar um endereço.

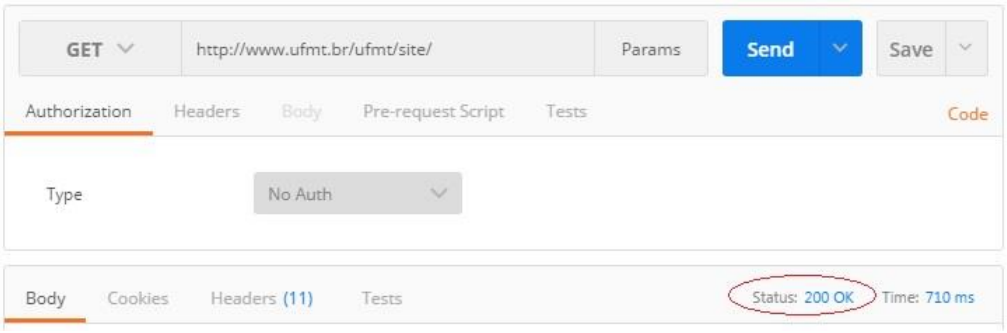

Figura 9 - Status ok de uma requisição GET

O status 200 (ok) nos mostra que a requisição foi de sucesso. Existem também outros números para diferentes situações, como:

- 304 (NOT MODIFIED): Recurso não modificado.
- 401 (UNAUTORIZED): Acesso não autorizado.
- 403 (FORBIDDEN): Falta de autenticação do usuário.
- 404 (NOT FOUND): Recurso não encontrado.

Ainda existem outros códigos, SAUDATE (2013) nos mostra mais alguns que ele considera como importantes em seu livro, 201 Created, 202 Accepted, 204 No Content, 400 Bad Request, 401 Unauthorized, 405 Method Not Allowed, 409 Conflict, 415 Unsupported Media Type e 500 Internal Server Error.

3.1.2 JAVA 8

O Java 8 foi lançado em 2014 SILVEIRA e TURINI (2014), surgindo novas possibilidades com o lambda, Streams e com os métodos por referências. Os desenvolvedores dessa nova versão deixou bem simples a sintaxe para utilização, evitando assim códigos mirabolantes que causa perca na performance da linguagem.

Mesmo com as mudanças novas, todas as outras versões continuam funcionando normalmente com a nova versão, não tirando a compatibilidade.

Para familiarizarmos com a evolução da linguagem, SILVEIRA e TURINI (2014), traz algumas comparações entre as sintaxes utilizada antes e depois do Java 8, incialmente mostrando um trecho de código na maneira antiga, antes do lançamento do Java 8, mostrado na Figura 10.

```
public static void main(String ... args) {
    Usuario user1 = new Usuario("Paulo Silveira", 150);
    Usuario user2 = new Usuario("Rodrigo Turini", 120);
    Usuario user3 = new Usuario("Guilherme Silveira", 190);

    List<Usuario> usuarios = Arrays.asList(user1, user2, user3);

    for(Usuario u : usuarios) {
        System.out.println(u.getNome());

    }
}
```

Figura 10 - Sintaxe anterior ao java 8. [Fonte: SILVEIRA , TURINI (2014)]

Na nova maneira, existem alguns novos métodos, um deles é o forEach que utilizado com a expressão lambda o código fica muita mais simples e sucinto conforme mostrado na Figura 11.

```
usuarios.forEach(u -> System.out.println(u.getNome()));
```

Figura 11 - Sintaxe Java 8. [Fonte: SILVEIRA, TURINI (2014)]

Com a expressão lambda fica claro que os códigos são muito mais concisos, evitando assim varias linhas de código para executar simples ações.

Realizar uma ordenação ou filtragem dos dados de uma lista é uma tarefa comum, segundos as discussões, utilizando a linguagem Java, foi possível identificar um padrão de utilização, que são: Criar listas, realizar transformações, consumir em operações como soma dos valores, médias entre outros, porem a necessidade de criação de variáveis era inevitável, dificultando a sua utilização.

Com essas limitações e dificuldades, a nova versão do Java, traz a implementação do Stream, que tornou muito mais fácil a utilização dessas listas, tornando às mutáveis, evitando muitos conflitos, afirma SILVEIRA e TURINI (2014).

Strem torna as listas de dados muito mais funcionais usando uma interface fluente, separando o Stream das coleções de dados. Criando o Stream conseguimos usar o método chamado Filter, aplicar expressões lambdas e definir qual será o critério do filtro a ser utilizado. Para deixar claro, o Stream não altera o estado da minha lista atual, ele apenas filtra conforme o parâmetro estabelecido assim a cada consulte o Filter irá passar por toda a minha lista novamente. Para ilustrar como funciona a filtragem, segue a Figura 12.

```
public static void main (String... args) {
    Usuario user1 = new Usuario("Paulo Silveira", 150);
    Usuario user2 = new Usuario("Rodrigo Turini", 120);
    Usuario user3 = new Usuario("Guilherme Silveira", 90);

    List<Usuario> usuarios = Arrays.asList(user1, user2, user3);

    Stream<Usuario> stream = usuarios
            .stream()
            .filter(u -> u.getPontos() > 100);

    stream.forEach(System.out::println);
}
```

Figura 12 - Utilização do método filter. [Fonte: SILVEIRA, TURINI (2014)].

O Stream é uma sequencia de elementos que podem ser trabalhos diversas formas com objetivos de executar uma série de operações.

No Java, conseguimos carregar as informações de arquivos CSV facilmente utilizando método readAllLines (ler todas as linhas), da classe Files do próprio JAVA conforme ilustrado na Figura 13.

```
29    public List<Ubs> listarUbs(){
30        List<Ubs> lista = new ArrayList<Ubs>();
31        List<String> linhas = null;
32        try {
33            linhas = Files.readAllLines(Paths.get("C:/ubs.csv"));
34        } catch (IOException e) {
35            e.printStackTrace();
36        }
37    {
```

Figura 13 - Importar arquivos CSV.

3.2 OBTENÇÃO DOS DADOS

Para conseguirmos esses dados, foi necessário acessar a o endereço do portal brasileiro de dados abertos no link http://dados.gov.br/ e realizado uma pesquisa diretamente na página, onde se digitado a palavra UBS, conseguimos encontrar o conjunto de dados requerido. A imagem da Figura 14 nos mostra a tela que foi realizada o download da coleção de dados.

Figura 14 - Download da coleção de dados

O conjunto de dados obtidos possui 37690 registros das Unidades Básicas de Saúde, são informações de todo o Brasil. Essas informações são atualizadas esporadicamente, a ultima atualização foi aproximadamente à 10 meses, conforme a Figura 15.

Figura 15 - Fluxo de Atividades.

Uma prévia dos arquivos está no (Anexo 2) deste trabalho. Para nos ajudar a identifica os atributos do conjunto de dados obtidos foi realizado o download do seu mapa de layout, ou dicionário de dados (Anexo B), como foi o nome estipulado para ele. No dicionário de dados encontrasse todas as descrições de cada atributo utilizado no arquivo. O formato utilizado foi o CSV, pois é o único formato disponível.

3.3 MODELAGEM

Com a abstração do conhecimento adquirido durante as pesquisas sobre o fluxo de atividade de uma API REST, foram realizados dois diagramas, na Figura 16 está o modelo de uma estrutura de um servidor REST, e na Figura 17 um diagrama de sequência e Figura 18 o diagrama de classe, para nos guiar durante o desenvolvimento e mostrar como será o acesso a API-UBS.

Para melhor representar este senário da estrutura de acesso ao servidor REST ficou com da seguinte forma: Acesso ao Recurso e a sua Representação através da URL, e posteriormente chamando os Métodos GETs.

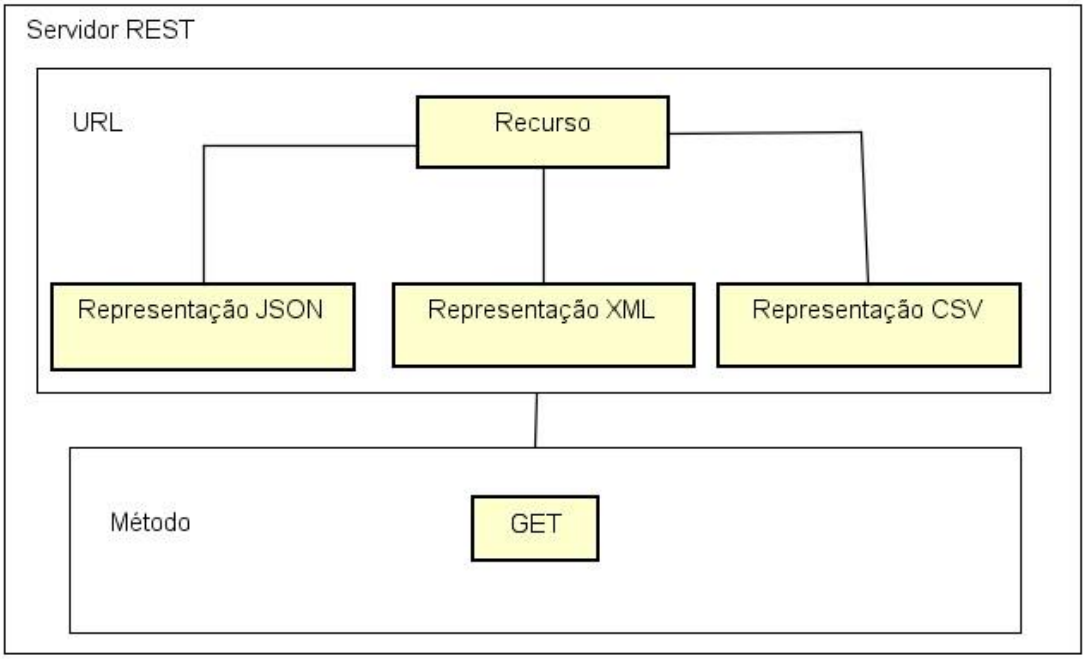

Figura 16 - Estrutura do Servidor Rest.

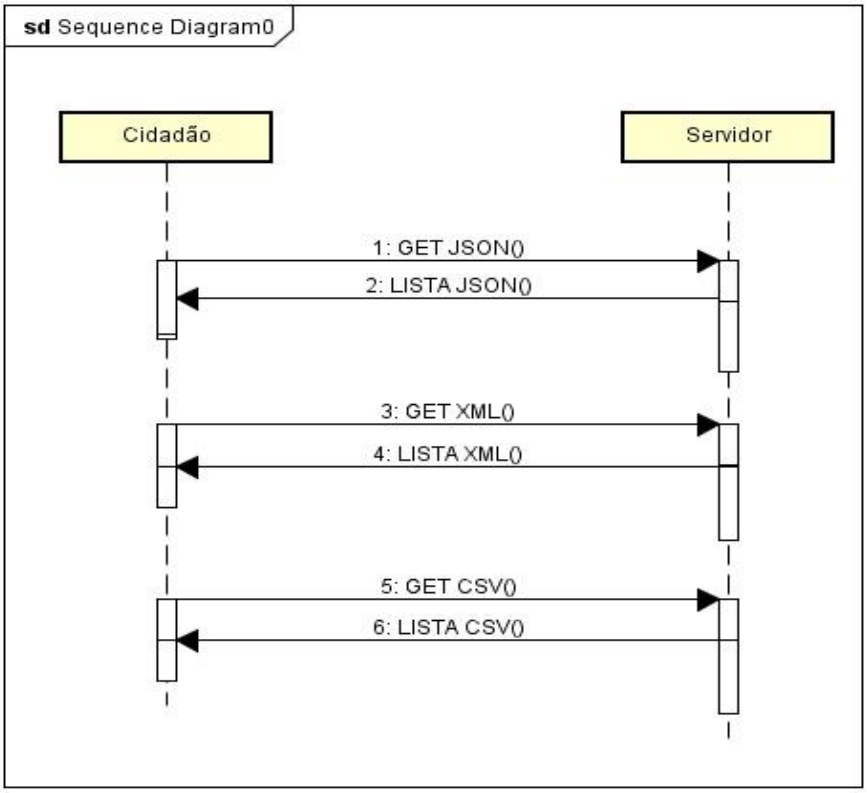

Figura 17 - Diagrama de Sequência.

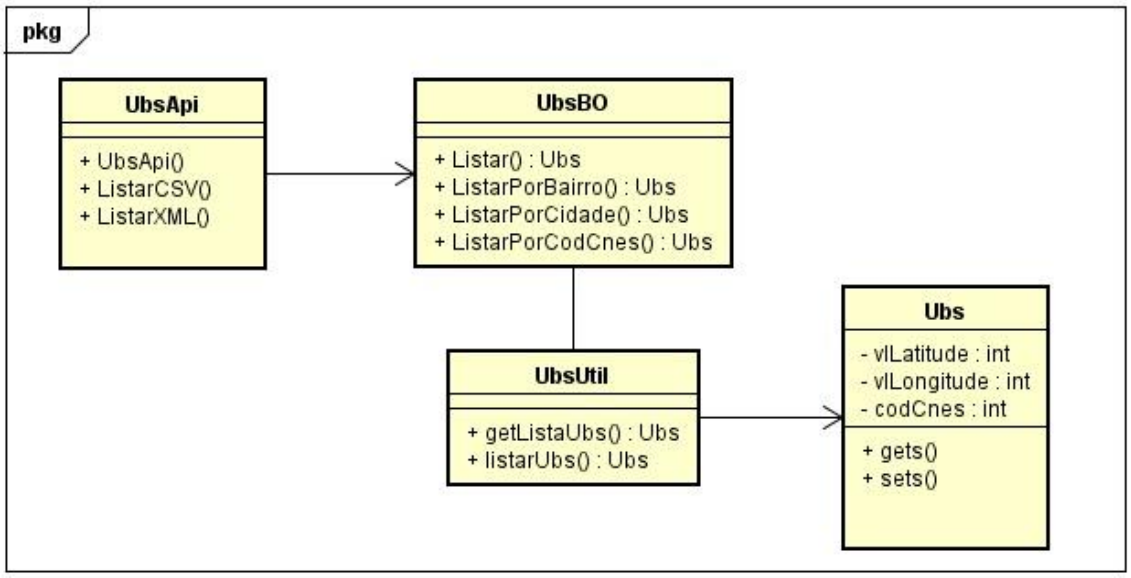

Figura 18 - Diagrama de Classe inicial.

3.4 DESENVOLVIMENTO DA API-UBS

Inicialmente utilizaríamos uma estrutura de banco de dados relacional, com todas as suas tabelas e entidades afins, entretanto o objetivo deste trabalho não é mostrar a estrutura para armazenamento, por esse motivo não iremos elencar a parte do banco de dados. A importação dos dados será a leitura do próprio arquivo disponibilizado em formato CSV.

Para realizarmos o carregamento dos dados, utilizaremos o método mostrado no item 3.1.2 deste trabalho, que é utilizado o método readAllLines. Este método precisa do endereço físico para localizar o arquivo, então disponível em c:/ubs.csv que foi o endereço salvo diretamente no servidor, mostrado na Figura 19.

Ainda na Figura 19, nos mostra o momento em que conseguimos mapear todos os campos através da sua posição da linha. Note que para o primeiro campo a posição é “0” zero, pois é assim que o Java inicia as suas contagens. Com os dados carregados em uma lista de objeto conseguiremos fazer os nossos filtros.

```
29    public List<Ubs> listarUbs(){
30        List<Ubs> lista = new ArrayList<Ubs>();
31        List<String> linhas = null;
32        try {
33            linhas = Files.readAllLines(Paths.get("C:/ubs.csv"));
34        } catch (IOException e) {
35            e.printStackTrace();
36        }
37        if(linhas != null){
38            linhas.remove(0);
39            linhas.forEach(linha -> {
40                Ubs ubs = new Ubs();
41                String[] campos = linha.split(",");
42                ubs.setVlrLatitude(campos[0]);
43                ubs.setVlrLongitude(campos[1]);
44                ubs.setCodMunic(campos[2]);
45                ubs.setCodCnes(campos[3]);
46                ubs.setNomEstab(campos[4]);
47                ubs.setDscEndereco(campos[5]);
48                ubs.setDscBairro(campos[6]);
49                ubs.setDscCidade(campos[7]);
50                ubs.setDscTelefone(campos[8]);
51                ubs.setDscEstrutFisicAmbiencia(campos[9]);
52                ubs.setDscAdapDeficFisicIdosos(campos[10]);
53                ubs.setDscEquipamentos(campos[11]);
54                ubs.setDscMedicamentos(campos[12]);
55
56                lista.add(ubs);
57            });
58        }
59        return lista;
60    }
```

Figura 19 - Código de Importação dos Dados

Com essas informações já é possível criar os filtros da API-UBS. Foi disponibilizado no (Apêndice A), todas as consultas que nossa API irá fazer juntamente com as devidas URLs, sendo assim iremos aplicar os conceitos na pratica, criando agora os filtros de cidade, bairro, código cnes, e pelo raio. A Figura 20 está representada todo o código de filtro.

```
20    public List<Ubs> listaUbs(){
21
22        return ubsUtil.getListaUbs();
23    }
24
25    public String listaUbsCsv(){
26        List<Ubs> retorno = listaUbs();
27
28        return escreverCsv(retorno);
29
30    }
31
32    public List<Ubs> listarPorCidade(String cidade){
33        List<Ubs> retorno = listaUbs().stream().filter(u -> u.getDscCidade().toUpperCase()
34                .contains(cidade.toUpperCase())).collect(Collectors.toList());
35        return retorno;
36    }
37
38    public List<Ubs> listarPorBairro(String bairro){
39        List<Ubs> retorno = listaUbs().stream().filter(u -> u.getDscBairro().toUpperCase()
40                .contains(bairro.toUpperCase())).collect(Collectors.toList());
41        return retorno;
42    }
43
44    public List<Ubs> listarPorCnes(String cnes){
45        List<Ubs> retorno = listaUbs().stream().filter(u -> u.getCodCnes().toUpperCase()
46                .contains(cnes.toUpperCase())).collect(Collectors.toList());
47        return retorno;
48    }
49
50    public List<Ubs> listar(Double latitude, Double longitude, Double raio){
51        List<Ubs> retorno = listaUbs().stream().filter(u -> isDistanciaBetween(latitude, longitude, u.getVlrLatitude(),
52                u.getVlrLongitude(), raio)).collect(Collectors.toList());
53        return retorno;
54    }
```

Figura 20 - Código fonte dos filtros realizados

Para realizar a pesquisa utilizando o raio, no Java já possui uma biblioteca que utiliza a formula de Haversine, que por sua vez considera a distancia mais curta entre um ponto e outro da terra ignorando os efeitos elipsoidais. Logo abaixo na Figura 21, mostra o código fonte do calculo para encontrar em nossa lista os lugares mais próximos conforme os parâmetros passados.

```
126    private boolean isDistanciaBetween(Double latitude1, Double longitude1, String latitude2, String longitude2, Double raio){
127        Double lat2 = new Double(latitude2);
128        Double lon2 = new Double(longitude2);
129        Double dis = haversine(latitude1, longitude1, lat2, lon2);
130        return dis <= raio;
131    }
132
133    private double haversine(double lat1, double lon1, double lat2, double lon2) {
134        final double raioTerra = 6372.8;
135        double dLat = Math.toRadians(lat2 - lat1);
136        double dLon = Math.toRadians(lon2 - lon1);
137        lat1 = Math.toRadians(lat1);
138        lat2 = Math.toRadians(lat2);
139
140        double a = Math.pow(Math.sin(dLat / 2),2) + Math.pow(Math.sin(dLon / 2),2) * Math.cos(lat1) * Math.cos(lat2);
141        double c = 2 * Math.asin(Math.sqrt(a));
142        return raioTerra * c;
143    }
```

Figura 21 - Código fonte do cálculo Haversine.

Com os filtros prontos, iremos mapear todos os endereços utilizando o REST do JAVA, como mostrado no Apêndice A. Para essa nossa API só iremos implementar o método

GET, pois iremos apenas fornecer as informações existente no arquivo CSV disponibilizado no sitio de dados abertos. Para ilustrar como será feito e cada parte do mapeamento, segue a Figura 22 mostrando toda estrutura.

```
96      @GET
97      @Path("/json/cidade/{cidade}")
98      @Produces(MediaType.APPLICATION_JSON + "; charset=UTF-8")
99      @Consumes(MediaType.APPLICATION_JSON)
100     public List<Ubs> listarPorCidadeJson(@Context HttpServletRequest request,
101                                          @PathParam("cidade") String cidade){
102         logger.info("Ip Requisição = "+ getIp(request)+" Path = /json/cidade/"+cidade);
103         return ubsBO.listarPorCidade(cidade);
104     }
```

Figura 22 - Mapeamento das URLs.

O @GET indica que estamos utilizando o método GET do protocolo HTTP, o @Path é o nosso endereço para acessar os recursos, nesse caso que foi apresentado é o recurse de cidade, que obrigatoriamente teremos que seguir a estrutura mostra no manual de utilização. O @produces indica qual tipo de saída ele irá produzir em seu resultado final, no caso mostrado será no formato JSON. O @Consumes serve para base de como será escrito o resultado, garantindo assim um formato uniforme.

O código da Figura 22 nos mostra, o modelo Path, ou seja, necessita digitar a URL completa para que seja possível localizar o recurso desejado. É possível realizar o mesmo filtro por meio de Query Parâmetro. Para que seja ilustrado esse modelo, segue o código para o recuso utilizando a query Sting como parâmetro, Figura 23.

```
124     @GET
125     @Path("/json/cidade")
126     @Produces(MediaType.APPLICATION_JSON + "; charset=UTF-8")
127     @Consumes(MediaType.APPLICATION_JSON)
128     public List<Ubs> listarCidadeJson(@Context HttpServletRequest request, @QueryParam("cidade") String cidade){
129         logger.info("Ip Requisição = "+ getIp(request)+" Query = /json/cidade/"+cidade);
130         return ubsBO.listarPorCidade(cidade);
131     }
```

Figura 23 - Mapeamento da URL da pesquisa por cidade usando query.

Até aqui, foi apresentado o carregamento do arquivo, o filtro por cidade, com suas respectivas URLs de Query e Path Parâmetro. Atualmente a aplicação possui mais de 40 tipos de URLs distintas, entre Query e Path Parâmetro, representados por CSV, JSON e

XML. Todos os filtros estão representados pelo novo diagrama de classe mostrado da Figura 24.

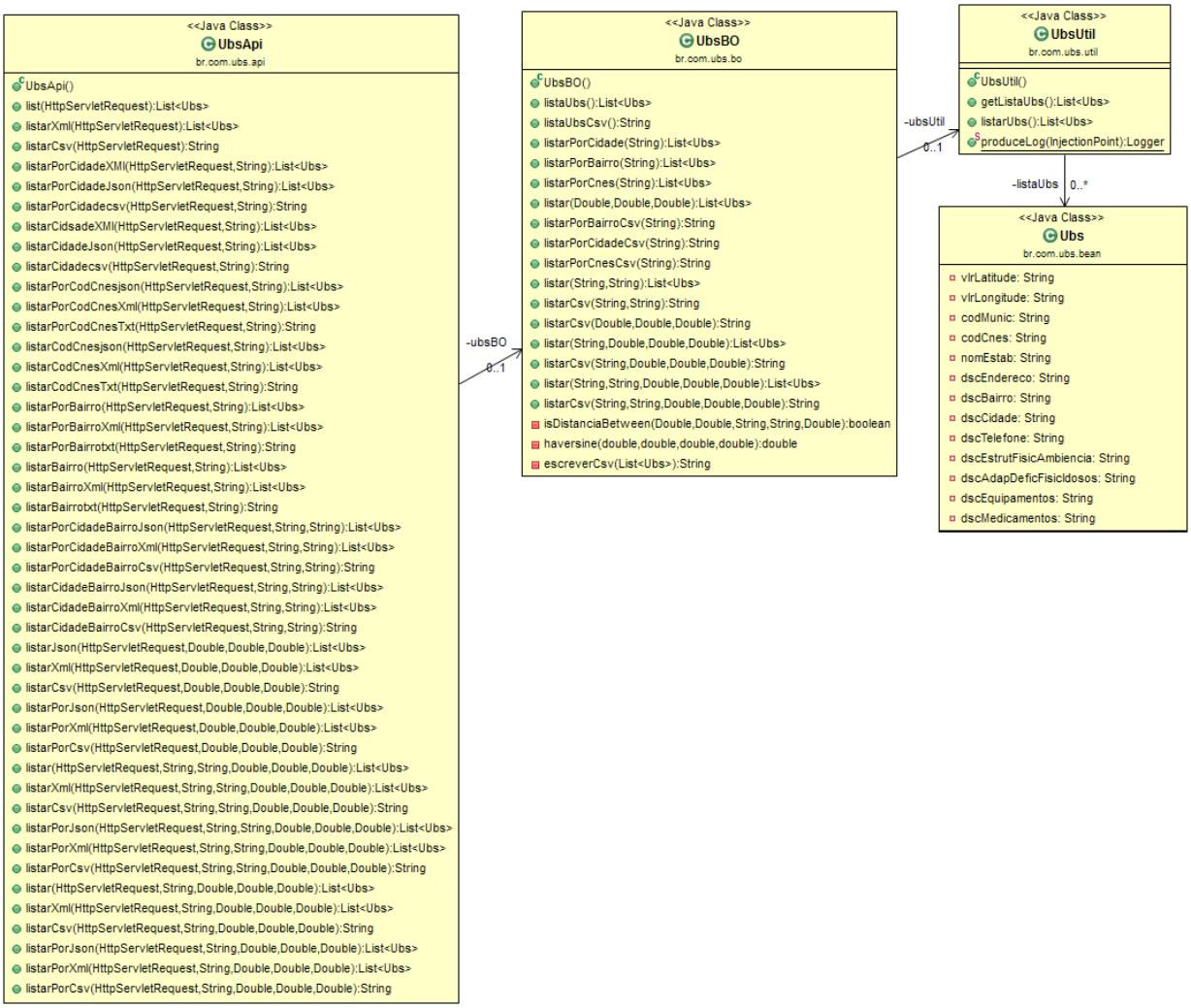

Figura 24 - Novo Diagrama de Classe.

As classes estão com as seguintes responsabilidades:

UBS - Responsável pela representação do objeto com seus respectivos atributos.

UbsUtil - Responsável por realizar a leitura do arquivo CSV mostrado na Figura 19.

UbsBO - Responsável por realizar as filtragens da nossa aplicação, ela é a camada de negócio, conforme apresentado nas Figuras 22 e 23.

UbsApi - Responsável pelo mapeamento das URLs e dos recursos disponibilizados.

4

EXEMPLO DE UTILIZAÇÃO DA API-UBS

Uma reutilização seria um aplicativo que com base na localização atual, mostrar no mapa quais unidades básicas estão na proximidade. Utilizando a localização da UFMT e passa passado um raio de dois quilômetros, obtemos como respostas quatro UBS. Para encontrar a localização correta da UFMT, foi utilizado o mapa do google conforme mostra a Figura 25.

Figura 25 - Localização da IC-UFMT.

De posse da localização da UFMT e das informações do manual de utilização que está no Apêndice A, já é possível montar uma URL de consulta da API-UBS. Para realizarmos os teste e mostrar a URL foi utilizado um aplicativo chamado Postman, ele é um cliente para realizar testes em APIs, capaz de enviar uma requisição e visualizar suas respostas. Para os nossos testes utilizamos a versão gratuita, pois são testes de simples utilização da API-UBS sem nenhuma integração com outros colaboradores ou sistemas. Na Figura 26 nos mostra a URL montada para nossa consulta.

Figura 26- Parâmetro com a localização da UFMT e raio de dois quilômetros.

Conforme orientações do manual de utilização, na URL montada estão o tipo de retorno, que será JSON, e o recurso a ser acessado, nesse caso foi passado */raio*, que tem como parâmetro a latitude, a longitude e o raio em quilômetros. O modelo dessa consulta é do tipo Path, onde passo o endereço real do recurso.

Ao executar essa URL a resposta será um conjunto de UBS, todas que estiverem em uma proximidade de dois quilômetros da localização informada. Ao receber a resposta, nos mostra quatro UBS que está ilustrada na Figura 27.


```
[
  {
    "vlrLatitude": "-15.5992770195003",
    "vlrLongitude": "-56.067481040953",
    "codMunic": "510340",
    "codCnes": "6225926",
    "nomEstab": "USF PEDREGAL",
    "dscEndereco": "RUA TAIAMA",
    "dscBairro": "PEDREGAL",
    "dscCidade": "Cuiabá",
    "dscTelefone": "6536171711",
    "dscEstrutFisicAmbiencia": "Desempenho acima da média",
    "dscAdapDeficFisicIdosos": "Desempenho mediano ou  um pouco abaixo da média",
    "dscEquipamentos": "Desempenho mediano ou  um pouco abaixo da média",
    "dscMedicamentos": "Desempenho mediano ou  um pouco abaixo da média"
  },
  {
    "vlrLatitude": "-15.6111860275264",
    "vlrLongitude": "-56.0819864273055",
    "codMunic": "510340",
    "codCnes": "2471051",
    "nomEstab": "CENTRO DE SAUDE PICO DO AMOR",
    "dscEndereco": "R CAPITAO IPORA",
    "dscBairro": "PICO DO AMOR",
    "dscCidade": "Cuiabá",
    "dscTelefone": "6536171327",
    "dscEstrutFisicAmbiencia": "Desempenho mediano ou  um pouco abaixo da média",
    "dscAdapDeficFisicIdosos": "Desempenho mediano ou  um pouco abaixo da média",
    "dscEquipamentos": "Desempenho mediano ou  um pouco abaixo da média",
    "dscMedicamentos": "Desempenho mediano ou  um pouco abaixo da média"
  },
  {
    "vlrLatitude": "-15.5993843078609",
    "vlrLongitude": "-56.0672020912154",
    "codMunic": "510340",
    "codCnes": "6226132",
    "nomEstab": "USF ADELAIDE ALVES DA SILVA",
    "dscEndereco": "RUA ROSARIO OESTE",
    "dscBairro": "PEDREGAL",
    "dscCidade": "Cuiabá",
    "dscTelefone": "6536171717",
    "dscEstrutFisicAmbiencia": "Desempenho mediano ou  um pouco abaixo da média",
    "dscAdapDeficFisicIdosos": "Desempenho mediano ou  um pouco abaixo da média",
    "dscEquipamentos": "Desempenho mediano ou  um pouco abaixo da média",
    "dscMedicamentos": "Desempenho mediano ou  um pouco abaixo da média"
  },
  {
    "vlrLatitude": "-15.5992877483363",
    "vlrLongitude": "-56.0749483108504",
    "codMunic": "510340",
    "codCnes": "2655497",
    "nomEstab": "CENTRO DE SAUDE JARDIM LEBLON",
    "dscEndereco": "RUA PROJETADA",
    "dscBairro": "JARDIM LEBLON",
    "dscCidade": "Cuiabá",
    "dscTelefone": "6536171336",
    "dscEstrutFisicAmbiencia": "Desempenho mediano ou  um pouco abaixo da média",
    "dscAdapDeficFisicIdosos": "Desempenho mediano ou  um pouco abaixo da média",
    "dscEquipamentos": "Desempenho mediano ou  um pouco abaixo da média",
    "dscMedicamentos": "Desempenho mediano ou  um pouco abaixo da média"
  }
]
```


Figura 27 - Resposta da consulta na API-UBS.

Essa visualização está apenas para mostrar que realmente a consulta está dinâmica e retornando os objetos desejados, possibilitando a integração com qualquer sistema. Para continuarmos nosso exemplo, não entrei em méritos de desenvolver uma pagina que consuma da API-UBS, e posteriormente utiliza o Google Maps, mas manualmente foi montada uma nova URL com os padrões adotados pelo API do Google e utilizando uma navegador qualquer, foi submetido a consulta e o resultado da é apresentado na Figura 28.

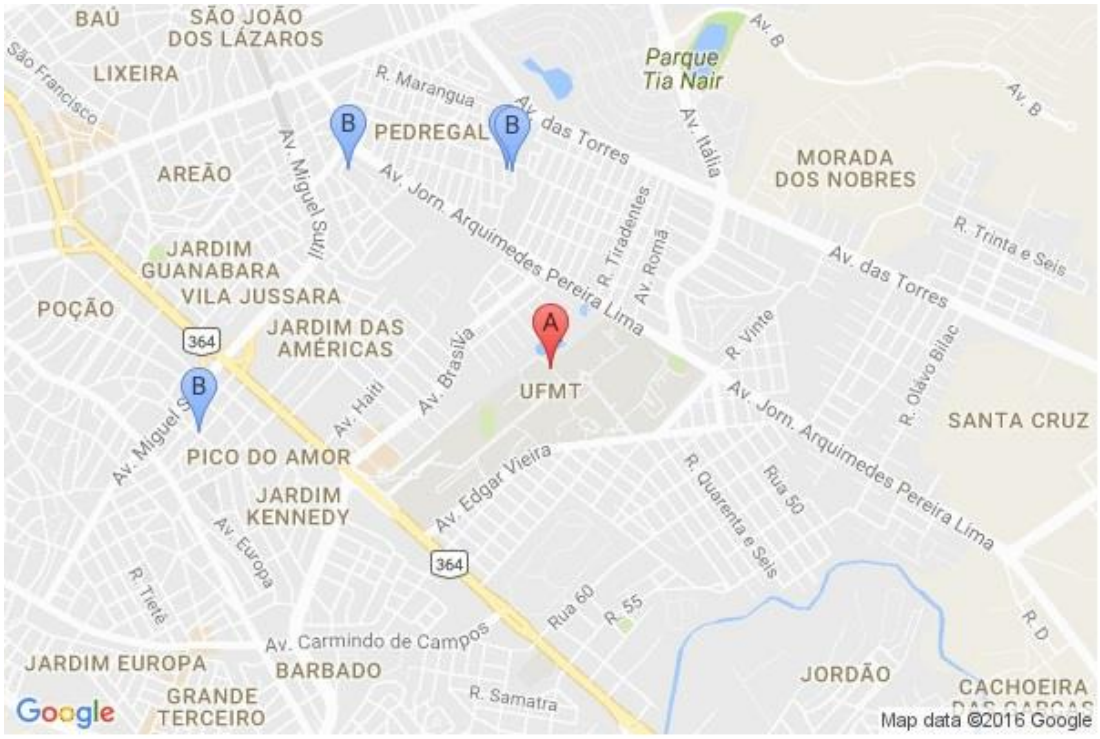

Figura 28 - Imagem ilustrativa consultando a API google passando como parâmetro o resultado da API-UBS.

O ponto em vermelho com a letra “A” foi identificado à localização da UFMT, e os pontos em Azul com a letra “B” as localizações obtidas da API-UBS.

Essa foi apenas uma simples utilização, para mostrar que com os dados dinâmicos abrimos vários espaços para o cidadão transformar, meros dados em dados abertos possibilitando então a reutilização da informação e facilitando o acesso a eles.

5

CONCLUSÕES

Com a construção da API-UBS conseguimos chegar ao nosso objetivo, que é deixar os dados de forma automatizada para realização de consultas e assim então realizar trabalhos mais dinâmicos sem se preocupar com o armazenamento das informações.

Após a disponibilização desses dados por meio da API-UBS, fica claro perceber que as portas ficam abertas para outros desenvolvedores reutilizar os dados disponíveis criando vários outros aplicativos independente de plataforma, dispositivo ou tecnologias.

Nota se que conseguimos prever os princípios da LAI, deixando o dado ser indexado e encontrado facilmente, disponível em formato compreensível por máquina, permitindo a replicação deles, incentivando assim a sociedade a contribuir com os serviços inovadores ao cidadão, facilitando novos negócios e aumentando a qualidade nos dados abertos, atendendo assim a obrigatoriedade.

Ressaltamos também que a API-UBS está atingindo quatro estrelas no esquema proposto por Tim Berners-Lee, onde a disponibilização está identificada pelo acesso direto a suas URIs, cumprindo assim o quarto nível de maturidade de publicação de dados.

Como trabalhos futuros, podemos elencar aqui a construção de novos filtros, utilizando a mistura de informação e extraindo assim novos dados para então atingir o ultimo nível de maturidade de publicação de dados. Podem ser realizadas construções de aplicativos que iram consumir a API-UBS.

REFERÊNCIAS

BERNERS-LEE, Tim; (2009), **Linked Data.** Disponível em: <https://www.w3.org/DesignIssues/LinkedData.html> Acessado em: 14/12/2016.

BERNERS-LEE, Tim. **5 Star Open Data**. Disponível em: <http://5stardata.info/> Acessado em 14/12/2016. (2012)

BRASIL. Lei no. 12.527 de 18 de outubro de 2011. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm> Acessado em: 15/10/2016.

DAL MORO, Tharcis; DORNELES, Carina; REBONATTO, Marcelo Trindade. Web services WS -* versus Web Services REST. **Revista de Iniciação Científica**, v. 11, n. 1, 2009.

DE MENDONÇA, Patricia Graziely Antunes; MACIEL, Cristiano; VITERBO, José. Visualizing Aedes aegypti infestation in urban areas: A case study on open government data mashups. **Information Polity**, v. 20, n. 2, 3, p. 119-134, 2015.

DIAS, Emilio; **Desmistificando REST com Java**, 1ª Edição, 11/02/2016.

DINIZ, Vagner. **Como conseguir dados governamentais abertos**. 2010.

Especificação do HTTP/1.1, disponível em: <http://www.w3.org/Protocols/rfc2616/rfc2616.html > Acessado em: 15/08/2016.

GONÇALVES, Edson; **Desenvolvendo Aplicações Web com JSP, Servlets, JavaServer Faces, Hibernate, EJB 3 Persistence e Ajax**, 1ª Edição, 2007. p.1-6.

Introducing JSON (1999), disponível em:<http://www.json.org> Acessado em: 13/06/2016.

NETO, Abinader E LINS, Rafael Dueire. (2006), **Web Services em Java**, Rio de Janeiro, Brasport.

PBDA: Unidades Básicas de Saúde – UBS. Disponível em: <http://dados.gov.br./dataset/unidades-basicas-de-saude-ubs> Acessado em: 10/06/2016.

Portal Brasileiro de dados Abertos - Disponível em: <http://dados.gov.br> Acessado em: 22/08/2016.

REST, Learn; REST: A Tutorial, disponível em: < http://rest.elkstein.org > Acessado em: 10/06/2016.

SAUDATE, Alechrandre**: Rest - Construa APIs inteligentes de maneira simples**, 2013. 303p. ISBN: 978-85-66250-37-4.

SAUDATE, Alechrandre**: SOA Aplicado, Integrando com web services e além**, 2012. 319p. ISBN: 978-85-66250-15-2.

SILVEIRA, Paulo; TURINI, Rodrigo**: Java 8 Pratico – Lambda, Stream e os novos recursos da linguagem**, 2014. 148p. ISBN: 978-85-66250-46-6.

W3C. Web Services Architecture: W3C Working Group Note 11 February 2004, disponível em: <https://www.w3.org/TR/ws-arch/ > Acessado em: 21/06/2016.

APÊNDICE A – MANUAL DE UTILIZAÇÃO

Serão apresentados os principais recursos disponíveis na ferramenta bem como todos os seus parâmetros para acessar esses recursos e todas as estruturas das URLs a serem utilizadas.

ESTRUTURA DAS URLs

A estrutura para acessar os recursos está representada pela Figura 29.

```
URI:
http://localhost:8080/ubs/rest/json/cidade/cuiabá
                      |           |    |   |   |       |
                      |           |    |   |   |       |
                      |           |    |   |   |       |
                      |           |    |   |   |       | Valor do recurso.
                      |           |    |   |   |
                      |           |    |   |   | Recurso a ser pesquisado
                      |           |    |   |
                      |           |    |   | Representação (Disponivel JSON, XML e CSV)
                      |           |    |
                      |           |    | Contexto da Aplicação.
                      |           |
                      |           |Raiz do Projeto.
                      |
                      | Dominio de Hospedagem
```

Figura 29 - Estrutura da URL utilizando acesso pelo caminho.

RESPOSTAS

Toda consulta retornará os atributos do recurso, que são elas: latitude, longitude, código do município, código CNES, nome do estabelecimento, endereço, bairro, cidade,

telefone, estrutura física da ambiência, adaptação para deficiente físico e Idoso, situação dos equipamentos e situação dos Medicamentos. Se obtiver a resposta da consulta do exemplo da Figura 29, iremos receber uma lista contendo todas as UBS da cidade de Cuiabá, representado por JSON. Segue um exemplo de um retorno em JSON, ilustrado na figura 30.

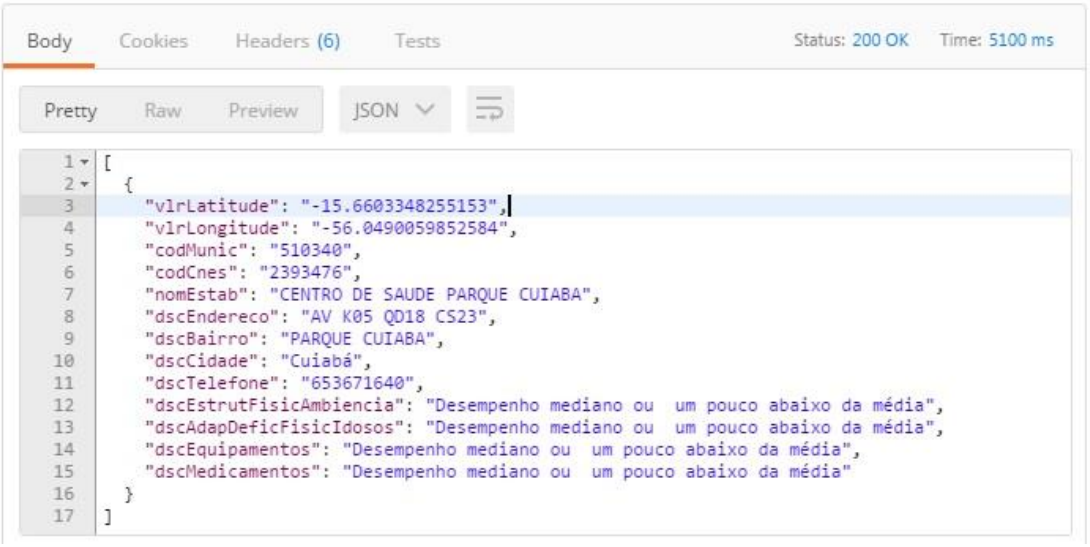

Figura 30 - Representação do Recurso em JSON.

TIPOS DE CONSULTAS

A ferramenta permite realizar diversos tipos de consultas. A seguir, serão apresentados os tipos de consultas que podem ser efetuadas, descrevendo como devem ser realizadas.

CONSULTAR TODOS OS DADOS

Para consultar todos os dados basta acessar as seguintes URLs.

1. http://localhost:8080/ ubs/rest/json
2. http://localhost:8080/ubs/rest/xml
3. http://localhost:8080/ubs/rest/csv

Ao acessar essas URLs irá retornar todos os registros do arquivo em forma de listas. Não é recomendável, pois exigirá muito dos recursos físicos disponíveis tanto para o lado de quem estiver lendo, ou seja, o cliente quanto ao lado do servidor. Para utilização segue na Tabela 2 com os recursos.

Tabela 2 - Consultar todos os registros.

Recuso	Verbo HTTP	Parâmetros	Descrições
ubs/rest/json	GET	(SEM PARÂMETRO)	Método que retornará todas as informações representadas por JSON.
ubs/rest/xml	GET	(SEM PARÂMETRO)	Método que retornará todas as informações representadas por XML.
ubs/rest/csv	GET	(SEM PARÂMETRO)	Método que retornará todas as informações representadas por CSV.

CONSULTAR PELA CIDADE.

Para consultar os registros que contem a cidade passando pelo parâmetro, segue as instruções relatadas na Tabela 3.

Tabela 3 - Consultar todos os registros com parâmetro cidade.

Recuso	Método	Query / Path	Parâmetros	Descrições
ubs/rest/json/cidade/ {cidade}	GET	Path	{cidade}	Método que retornará todas as informações representadas por JSON que contenha o parâmetro passado.
ubs/rest/json/cidade	GET	Query	cidade = {cidade}	Método que retornará todas as informações representadas por JSON que contenha o parâmetro.

Exemplos de Utilização da URL.

Representação dos Dados: XML, JSON, CSV.

1. Tipo de Pesquisa: Path
   1.1. http://localhost:8080/ubs/rest/xml/cidade/Cuiabá
2. Tipo de Pesquisa: Query
   2.1. http://localhost:8080/ubs/rest/json/cidade?cidade=cuiabá

CONSULTAR PELO CODIGO CNES.

Para consultar os registros que contem o código CNES passando pelo parâmetro, segue as instruções relatadas na Tabela 4.

Tabela 4 - Consultar Todos os registros com parâmetro CODCNES.

Recuso	Método	Query / Path	Parâmetros	Descrições
ubs/rest/json/ codcnes /{codcnes}	GET	Path	{codcnes}	Método que retornará todas as informações representadas pro JSON que contenha o parâmetro passado.
ubs/rest/json/codcnes	GET	Query	codcnes={codcnes}	Método que retornará todas as informações representadas pro JSON que contenha o parâmetro passado.

Exemplos de Utilização da URI.

Representação dos Dados: XML, JSON, CSV.

1. Tipo de Pesquisa: Path
   1.1. http://localhost:8080/ubs/rest/json/codcnes/2393476
2. Tipo de Pesquisa: Quer
   2.1 http://localhost:8080/ubs/rest/csv/codcnes?codcnes=2393476

CONSULTAR PELA CIDADE E BAIRRO

Para consultar os registros que contem a cidade e o bairro passados pelo parâmetro, segue as instruções relatadas na Tabela 5.

Tabela 5 - - Consultar Todos os registros com parâmetro cidade e bairro.

Recuso	Método	Query / Path	Parâmetros	Descrições
ubs/rest/json/	GET	Path	{cidade},{bairro}	Método que retornará todas as informações representadas pro JSON que contenha o parâmetro passado.
ubs/rest/json/cidade/bairro	GET	Query	cidade={cidade}, bairro={bairro}	Método que retornará todas as informações representadas pro JSON que contenha o parâmetro passado.

Exemplos de Utilização da URI.

Representação dos Dados: XML, JSON, CSV.

1. Tipo de Pesquisa: Path
   1.1. http://localhost:8080/ubs/rest/json/Várzea%20Grande,centro
2. Tipo de Pesquisa: Query
   2.1. http://localhost:8080/ubs/rest/json/cidade/bairro?cidade=cuiabá&bairro=santa%20rosa

CONSULTAR POR LATITUDE, LONGITUDE E RAIO

Para consultar os registros que contem a latitude, longitude e raio, passados como parâmetro conforme apresentado na Tabela 6.

Tabela 6 - Consultar Todos os registros por latitude, longitude e raio.

Recuso	Método	Query / Path	Parâmetros	Descrições
ubs/rest/json/raio	GET	Path	{latitude}, {longitudo}, {raio}	Método que retornará todas as informações representadas pro JSON que contenha o parâmetro passado.
ubs/rest/json/raio	GET	Query	cidade={cidade}, bairro={bairro}	Método que retornará todas as informações representadas pro JSON que contenha o parâmetro passado.

Exemplos de Utilização da URI.

Representação dos Dados: XML, JSON, CSV.

1. Tipo de Pesquisa: Path
   1.1. http://localhost:8080/ubs/rest/json/raio/-15.6600847,-56.0428662,5
2. Tipo de Pesquisa: Query
   2.1. http://localhost:8080/ubs/rest/json/raio?latitude=-15.6600847&longitude=-56.0428662&raio=2

ANEXO A

ARQUIVO UBS EM CSV

```
vlr_latitude,vlr_longitude,cod_munic,cod_cnes,nom_estab,dsc_endereco,dsc_bairro,dsc_cidade,dsc_telefone,dsc_estrut_fisic_ambiencia,dsc_adap_defic_fisic_idosos,d
-10.9112370014188,-37.0620775222768,280030,3492,US OSWALDO DE SOUZA,TV ADALTO BOTELHO,GETULIO VARGAS,Aracaju,7931791326,Desempenho acima da média,Desempenho mui
-9.48594331741306,-35.8575725555409,270770,6685315,USF ENFERMEIRO PEDRO JACINTO AREA 09,R 15 DE AGOSTO,CENTRO,Rio Largo,Não se aplica,Desempenho mediano ou  um
-23.896,-53.41,411885,6811299,UNIDADE DE ATENCAO PRIMARIA SAUDE DA FAMILIA,RUA GUILHERME BRUXEL,CENTRO,Perobal,4436251462,Desempenho muito acima da média,Desemp
-16.447874307632,-41.0098600387561,313580,6335616,POSTO DE SAUDE DE BOM JESUS DA ALDEIA,RUA TEOFILO OTONI,ALDEIA,Jequitinhonha,3337411423,Desempenho mediano ou
-6.57331109046917,-35.1076054573049,250930,6662226,POSTO ANCORA URUBA,RODOVIA PB N 065,SITIO,Mataraca,Não se aplica,Desempenho acima da média,Desempenho acima d
-7.03715085983256,-37.2887992858876,251080,6713971,UNIDADE DE SAUDE DA FAMILIA ANA RAQUEL,RUA SEVERINO SOARES,JD GUANABARA,Patos,Não se aplica,Desempenho mediar
-5.99885702133161,-40.2936887741077,231330,3322076,PSF ODILON AGUIAR,RUA DOMINGAS GOMES,CENTRO,Tauá,(88)4373850,Desempenho mediano ou  um pouco abaixo da média,
-24.5821130275719,-49.435493946074,412863,6749836,UAPSF PROF HETTY ROSA DE MOURA E COSTA,AVENIDA EZIDIO NEVES,CERRADO,Doutor Ulysses,4136641208,Desempenho muitc
-5.89795231819136,-44.8160004615771,211230,2450801,UNIDADE BASICA DE SAUDE JOSE BIBI,POVOADO SAO JOAQUIM DOS MELOS,ZONA RURAL,Tuntum,9935225071,Desempenho media
-9.55789089202853,-37.3832130432118,270840,6461689,USF NOSSA SENHORA DO BOM PARTO,RUA AFONSO SOARES VIEIRA,CENTRO,São José da Tapera,Não se aplica,Desempenho me
-3.09060215950003,-59.9864888191206,130260,2014815,UBS L 29,RUA FLAVIO COSTA,COROADO I,Manaus,9232492767,Desempenho mediano ou  um pouco abaixo da média,Desempe
-15.6327939033504,-58.1737661361677,510710,2394243,UNIDADE DE SAUDE DA FAMILIA ZEFERINO II,RUA RUI BARBOSA,JARDIM ZEFERINO II,São José dos Quatro Marcos,(65)325
-14.5426476001735,-52.7972888946518,510260,2395487,CENTRO DE SAUDE MUNICIPAL DE CAPINAPOLIS,AV XV DE NOVEMBRO,CENTRO,Campinápolis,(66) 4371631,Desempenho mediar
-16.0691678524013,-47.9820585250841,522185,2383411,UNIDADE BASICA DE SAUDE VALPARAISO,CENTRO COMERCIAL QUADRA 07,VALPARAISO I,Valparaíso de Goiás,6136276636,Des
-27.029885,-48.668039,420320,6816878,UNIDADE SAUDE DA FAMILIA SANTA REGINA,MANAGUA,SANTA REGINA,Camboriú,4733653239,Desempenho muito acima da média,Desempenho n
-29.6088624000541,-51.1745309829697,431080,2230313,POSTO MORADA DO SOL IVOTI,RUA CAXIAS DO SUL,JARDIM BUHLER,Ivoti,35636067,Desempenho mediano ou  um pouco abai
-15.453,-47.614,521760,2438623,UNIDADE BASICA DE SAUDE 06 QA11 LESTE,QA11 MC,SETOR LESTE,Planaltina,(61)6371273,Desempenho mediano ou  um pouco abaixo da média,
-22.0929586887353,-43.9471149444567,315590,2141728,POSTO DE SAUDE DE SAO CRISTOVAO,RUA PE ARLINDO VIEIRA,SAO CRISTOVAO,Rio Preto,3232831542,Desempenho mediano c
-12.431,-38.328,292520,2653486,UNIDADE BASICA DO PSF DA INOCOOP,RUA DALVA MONTE CRUZ,INOCOOP,Pojuca,(71)06452484,Desempenho acima da média,Desempenho mediano ou
-29.1107439994804,-49.6437835693345,421770,2624591,PSF SAO JOSE,RUA ANICETO SILVEIRA,SAO JOSE,Sombrio,4835334525,Desempenho muito acima da média,Desempenho medi
-9.3700182437894,-40.482494831084,261110,2429667,UPS MIGUEL DE LIMA DURANDO,RUA 24,LOTEAMENTO RECIFE,Petrolina,(81)38644168,Desempenho muito acima da média,Dese
-9.34695124626132,-40.3852057456958,261110,2429764,UPS JANUARIO F NUNES,POVOADO DE SERROTE DO URUBU,ZONA RURAL,Petrolina,(87)38611456,Desempenho acima da média,
-22.0368361473077,-42.9901456832873,330540,2294877,POSTO DE SAUDE DE ANTA,RUA ANTONIO ALVES DE CARVALHO,ANTA,Sapucaia,2422710421,Desempenho mediano ou  um pouco
-26.7414307594292,-52.9707098007187,421775,2537893,POSTO DE SAUDE DE SUL BRASIL,RUA DR JOSE LEAL FILHO,CENTRO,Sul Brasil,4933670026,Desempenho mediano ou  um po
-13.6370587348934,-43.4363365173327,290390,5871808,UNIDADE DE SAUDE DA FAMILIA DA BATALHA,ASSENTAMENTO BATALHA,ZONA RURAL,Bom Jesus da Lapa,77 34815039,Desemper
-3.0269479751586,-60.1796722412092,130260,2016117,PSR NOSSA SENHORA DO LIVRAMENTO,COMUNIDADE NOSSA SRA DO LIVRAMENTO,RIO NEGRO,Manaus,9232511120,Desempenho medi
-23.86747598648,-52.8732490539535,412790,2734664,NIS I POSTO 24 HORAS,AV LONDRINA,CENTRO,Tuneiras do Oeste,4436531308,Desempenho mediano ou  um pouco abaixo da
-27.90001630783,-48.9291143417344,421590,2622734,UNIDADE SANITARIA DE SAO BONIFACIO,AV CLEMENTE LEHMKUHL,CENTRO,São Bonifácio,4832520064,Desempenho acima da méc
-14.3435847759243,-60.2327156066877,510550,2395053,POSTO DE SAUDE RICARDO FRANCO,RUA PRINCIPAL,ZONA RURAL,Vila Bela da Santíssima Trindade,(65)32591128,Desemper
-10.5664980411526,-48.9105319976792,171500,2468220,UNIDADE BASICA DE SAUDE NOVA ROSALANDIA,AVENIDA ARAGUAIA,CENTRO,Nova Rosalândia,6335201444,Desempenho acima d
-10.7106292247769,-62.2592854499799,110015,2496909,CENTRO DE SAUDE CARLOS CHAGAS,R D PEDRO II,CENTRO,Ouro Preto do Oeste,6934613310,Desempenho mediano ou  um po
-10.7499611377713,-40.3623318672168,292460,6908357,USF BAIRRO ISMAEL LOPES,RUA ISMAEL,ISMAEL LOPES,Pindobaçu,Não se aplica,Desempenho muito acima da média,Desem
```

Figura 31 - Arquivo UBS em CSV.

ANEXO B

DICIONÁRIO DE DADOS

Nome do campo	Rótulo do campo CSV	Descrição	Características do campo
Latitude	vlr_latitude	Representa a latitude em formato decimal. Considerar valores negativos. O sistema de referência espacial é o EPSG:4326.	**Tipo:** Numérico **Obrig:** N **Formato:** -15.841038 **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** N
Longitude	vlr_longitude	Representa a longitude em formato decimal. Considerar valores negativos. O sistema de referência espacial é o EPSG:4326.	**Tipo:** Numérico **Obrig:** N **Formato:** -15.841038 **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** N
Código do Município	cod_munic	Representa o código IBGE do município.	**Tipo:** Numérico **Obrig:** S **Formato:** Não se aplica **Tamanho:** 7 caracteres O 7º dígito dos códigos IBGE é verificador e nem sempre são fornecidos pelo Ministério da Saúde. Apenas 6 dígitos são obrigatórios **Domínio:** Não se aplica **Filtro:** N
Código CNES	cod_cnes	Representa o código CNES	**Tipo:** Numérico **Obrig:** S **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** S
Nome do Estabelecimento	nom_estab	Representa o nome do estabelecimento.	**Tipo:** Texto **Obrig:** S **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** S
Endereço	dsc_endereco	Representa o endereço da unidade.	**Tipo:** Texto **Obrig:** N **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica

Nome do campo	Rótulo do campo CSV	Descrição	Características do campo
			Filtro: N
Bairro	dsc_bairro	Representa o nome do Bairro.	**Tipo:** Texto **Obrig:** N **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** N
Cidade	dsc_cidade	Representa o nome da cidade	**Tipo:** Texto **Obrig:** N **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** N
Telefone	dsc_telefone	Representa o número do telefone	**Tipo:** Texto **Obrig:** N **Formato:** Não se aplica se aplica **Domínio:** Não se aplica **Filtro:** N
Situação em relação a estrutura física e ambiência	dsc_estrut_fisic_ambiencia	Informa a situação em relação a estrutura física e ambiência	**Tipo:** Texto **Obrig:** N **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** N
Situação em relação a adaptações para deficientes e idosos	dsc_adap_defic_fisic_idosos	Informa a situação em relação a adaptações para deficientes e idosos	**Tipo:** Texto **Obrig:** N **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** N
Situação em relação aos equipamentos	dsc_equipamentos	Informa a situação em relação aos equipamentos	**Tipo:** Texto **Obrig:** N **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** N
Situação em relação aos Mediacamentos	dsc_medicamentos	Informa a Situação em relação aos medicamentos	**Tipo:** Texto **Obrig:** N **Formato:** Não se aplica **Tamanho:** Não se aplica **Domínio:** Não se aplica **Filtro:** N