BusInRio: Explorando Dados Abertos de Transporte Público do Município do Rio de Janeiro

Luan Soares Andrade[1], Sérgio Manuel Serra da Cruz[1,2]

[1] Universidade Federal Rural do Rio de Janeiro - Programa PET-SI/UFRRJ

[2] Programa Pós Graduação em Modelagem Matemática e Computacional/UFRRJ

serra@pet-si.ufrrj.br, luansandrade92@hotmail.com

Resumo. *No contexto de dados abertos ligados foi realizada uma aplicação que visa dispor de forma simples e gratuita aos usuários de transporte público informações em tempo real sobre seus ônibus. No aplicativo BusInRio é possível definir rotas para o destino e conhecer a localização da condução em tempo real. Também é possível conhecer informações sobre o trânsito na cidade de forma a impactar a escolha do usuário na hora de escolher qual melhor condução tomar. Os dados usados para o desenvolvimento do aplicativo são disponibilizados pelo governo do estado.*

1. Introdução

Dados Abertos entraram na pauta do governo brasileiro há pouco mais de quatro anos. Nesse período diversos eventos contribuíram para a crescente demanda dos órgãos governamentais por capacitação no tema. Podemos citar a Lei de Acesso à Informação (LEI 12.527/11[1]), a Parceria para Governo Aberto (OGP do inglês), a criação da Infraestrutura Nacional de Dados Abertos – INDA, além das diversas iniciativas internacionais que incentivam o uso de tecnologias em prol de uma sociedade melhor.

O governo do município do Rio de Janeiro mantém o site (http://data.rio/) onde disponibiliza diversos tipos de dados sobre os mais diversos assuntos como educação, saúde, impostos e transportes em formatos abertos para consulta pela população ou para que desenvolvedores criem suas aplicações. Cada conjunto de dados possui sua estrutura própria e são disponibilizados em vários formatos abertos, a saber: PDF, JSON, CSV, entre outros. Os dados escolhidos para esse trabalho foram os de transporte público, onde encontra-se informações em tempo real de GPS de ônibus, detalhes sobre as linhas do metrô, trem e percurso dos ônibus.

Este artigo utiliza dados abertos sobre ônibus, onde encontram-se informações de GPS em formato JSON e descrições do percurso dos ônibus em listas de arquivos CSV. Nesse sentido utilizamos a abordagem ETL e duas ferramentas NoSQL (MongoDB[2] e Cassandra[3]) para interligação desses dados para que se possa oferecer ao usuário do transporte público em tempo real informações acerca das linhas de ônibus e

[1] http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm

[2] https://www.mongodb.org/

[3] http://cassandra.apache.org/

este possa tomar decisões sobre que modal de transporte irá utilizar em função não só da localização na cidade do Rio de Janeiro como também sobre o estado do trânsito por onde circulará. Especificamente, apresentaremos um protótipo que permita aos usuários verificar graficamente através do Google Maps informações as linhas de ônibus e estado geral trânsito.

Este trabalho está organizado da seguinte forma: a seção 2 apresenta a os fundamentos teóricos, a seção 3 apresenta os matérias e métodos. A seção 4 apresenta a arquitetura *BusInRio* e a seção 5 discute os principais resultados. A seção 6 apresenta trabalhos relacionados e a última seção a conclusão deste trabalho.

2. Fundamentação Teórica

A natureza, a qualidade e a variedade das fontes de dados são os fatores principais que devem ser considerados para a escolha da estratégia mais apropriada para o desenvolvimento do app *BusInRio*. Nesta seção descrevermos os principais fundamentos teóricos utilizados neste trabalho.

2.1 Dados Abertos

A atividade de abrir dados, ou seja, de publicar dados abertos na Web, é realidade apenas para uma pequena parcela do governo brasileiro. Para grande parte dos órgãos público, a falta de pessoas capacitadas é o principal motivo que contribui com essa realidade. Abrir dados não é uma tarefa trivial. Existem várias abordagens, algumas rápidas e outras mais complexas. É nessa perspectiva que nasceu o modelo de maturidade de Berners-Lee para a qualificação das publicações de dados abertos.

A qualificação de dados de Berners-Lee é representada por: 1) Dados estão disponíveis na Web, independente de formato, sob uma licença aberta; 2) Dados na condição anterior mais, dados estruturados e legíveis por máquinas (por exemplo, um arquivo Excel ao invés de uma imagem digitalizada de uma tabela); 3) Dados na condição anterior mais, dados formato não proprietário (arquivo CSV ao invés de um Excel); 4) Dados na condição anterior mais, uso de URIs bem definidas para identificar as coisas, então as pessoas podem referenciá-las. Por fim, 5) Todas as anteriores mais, ligação dos seus dados com dados de outras pessoas para prover contexto.

2.2 ETL

Os dados abertos, assim como qualquer outro tipo de dado, podem se beneficiar da abordagem ETL (Extração-Transformação-Carga). O processo de ETL é um dos fundamentos dos sistemas de Data Warehousing, em que dados provenientes de diferentes fontes são extraídos, limpos, consolidados e, finalmente, disponibilizados através de novos bancos de dados (MENDONÇA, 2013).

2.3 NoSQL

Segundo a literatura, o termo NoSQL foi utilizado pela primeira vez em 1998 quando Carlos Strozzi criou um banco de dados relacional não SQL (MOURA, 2013). Deste então o termo vem sofrendo novas interpretações. Atualmente existem diversas implementação de bancos NoSQL. Neste trabalho utilizamos MongoDB e Cassandra.

O primeiro SGBD é orientado a documentos, utiliza JSON por padrão e apresenta grande velocidade pois guardar os dados na memória RAM, persistindo-o no disco rígido. Por isso, é usamos um segundo nó de réplica para, caso o servidor falhe. O segundo SGBD é orientado a colunas e também apresenta facilidade de escalabilidade, sendo capaz de replicar dados assincronamente através de múltiplas regiões geográficas.

3. Materiais e Métodos

Esta seção apresenta os datasets e a metodologia de desenvolvimento utilizado neste trabalho.

3.1. Metodologia

Neste trabalho utilizamos a modelagem não relacional de dados e a abordagem do ETL para realizar três etapas essências para garantir a qualidade e a integridade dos dados abertos, a saber:

1. Extração de dados: etapa em que os dados brutos são coletados de diversas fontes do rio.data e persistidos em disco, antes que qualquer reestruturação ou manipulação significativa;

2. Limpeza e Consolidação: etapa em que os dados coletados são corrigidos, rejeitados ou transformados de acordo com os requisitos da organização. Tratamento de questões de duplicação de dados ao integrar e consolidar fontes distintas;

3. Carga: etapa em que os dados são agrupados em documentos ou família de colunas e carregados em repositórios apropriados.

3.1. Datasets

Nesse trabalho foram utilizados dois datasets disponibilizados pela Prefeitura do Rio sobre os percursos de cada linha de ônibus e dados em tempo real dos GPS de cada ônibus de cada linha. Os dados sobre os percursos estão disponibilizados em formato CSV (http://data.rio/dataset/pontos-dos-percursos-de-onibus) e os dados dos GPS dos ônibus são disponibilizados em tempo real em JSON http://data.rio/dataset/gps-de-onibus).

Tabela 1. Descrição dos Datasets em LOD

Estrutura do dataset das linhas de ônibus	
Campo	**Descrição**
Descrição	Descrição de início e fim daquela linha
Agencia	Agência que disponibiliza os dados
Sequencia	Nº da sequência em que o ônibus passa num determinado ponto
Latitude	Latitude do ponto de ônibus
Longitude	Longitude do ponto de ônibus
Estrutura do dataset dos GPS dos Ônibus	

Campo	Descrição
datahora	Data e hora da coleta do dado
ordem	Identificação alfanumérica encontrada na lateral do ônibus
linha	Linha do ônibus
latitude	Latitude do ônibus no momento da coleta
longitude	Longitude do ônibus no momento da coleta
velocidade	Velocidade do ônibus no momento da coleta
direção	Medida em graus em relação ao norte, que representa a direção do veículo

4. Arquitetura *BusInRio*

A arquitetura *BusInRio* é distribuída, foi concebida em duas partes e adota o padrão cliente-servidor. O código que executa no servidor foi desenvolvido em PHP e tem como principais funcionalidades conectar-se ao data.rio, utilizar a metodologia ETL para coletar e tratar os dados abertos de transporte público, armazenar em suas respectivas bases NoSQL (Cassandra e MongoDB) e disponibilizar uma API para que a aplicação móvel colete e exiba os dados e mapas após serem tratados.

O cliente é totalmente desenvolvido em JavaScript através do framework Ionic[4] para obtenção de uma aplicação híbrida que funcione em qualquer tipo *smartphone* ou navegador web com as seguintes funções: coletar os dados da API e exibir para o usuário lista de ônibus e representações gráficas através de mapas do Google Maps que facilitem a leitura e visualização de dados do trânsito. A arquitetura da aplicação é representada na Figura 1 e todo seu código está disponível no endereço https://github.com/lsandrade/businrio.

A arquitetura *BusInRio* é escalável e sua porção servidora executa em ambiente do tipo nuvem de computadores do tipo SaaS[5]. O *BusInRio* utiliza o serviço Koding[6] que está disponível no endereço http://oluan.koding.io/transporte/. O serviço oferece máquinas virtuais em servidores da Amazon para que seus clientes possam testar aplicações em ambiente de nuvem. A instância utilizada nesse trabalho foi uma T2, que se trata de um ambiente de baixo custo para uso geral onde pode-se hospedar servidores web, ambientes de desenvolvimento e bancos de dados. *BusInRio* é uma aplicação que explora o princípio da elasticidade das nuvens, à medida que aumentam as requisições dos clientes ou percentagem de CPU e uso de memória variam acima de nível pré-definidos, novas instâncias são dinamicamente iniciadas ou desabilitadas.

[4] http://www.ionicframework.com

[5] https://pt.wikipedia.org/wiki/Software_como_servi%C3%A7o

[6] https://koding.com/

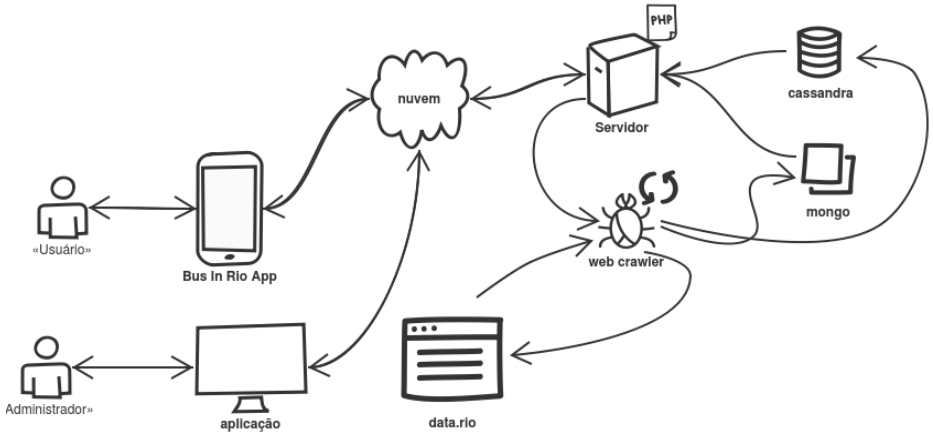

Figura 1. Arquitetura da aplicação distribuída

4.1. Web Crawler

O crawler é um módulo vital da aplicação, ele executa no servidor e consiste em acessar os servidores do data.rio, tratar os dados através da abordagem ETL e retroalimentar as bases de dados NoSQL em tempo real com informações sobre linhas e sobre os ônibus. Para isso ele coleta e trata os dados em formato JSON e armazena-os sob a forma de documentos no MongoDB e os dados do percurso em CSV e armazena-os no Cassandra.

4.2. API BusInRio

Interface de Programação de Aplicativos (API), 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 usar seus serviços. A API BusInRio é composta por uma série de funções acessíveis somente por programação, e que permitem utilizar características do software, ela permite que a aplicação cliente interprete e converta os dados dos transportes para o formato JSON que é mais facilmente interpretado pelas aplicações móveis e navegadores web.

Através da API BusInRio é possível que o aplicativo envie requisições HTML para o servidor web parametrizando, por exemplo, número da linha escolhida pelo usuário e data; então o servidor responde com dados sobre os ônibus em formato JSON que será interpretado pelo aplicativo e disposto na interface para melhor visualização do usuário.

4.3. Bases NoSQL

O Apache Cassandra é um projeto de sistema de banco de dados distribuído altamente escalável de segunda geração, que reúne a arquitetura do DynamoDB, da Amazon Web Services e modelo de dados baseado no BigTable, do Google. Este SGBD e um modelo orientado a colunas e por esse motivo foi escolhido para representar os dados dos percursos dos ônibus. Foi criada uma *Keyspace* chamada transporte e as tabelas "percurso" e "linha" para representar os dados. Em "linha" armazena-se os dados sobre a linha do ônibus (identificação e descrição) e no "percurso" os pontos pelo qual o ônibus passa.

O MongoDB é uma aplicação de código aberto, de alta performance, sem esquemas, orientado a documentos. Foi escrito na linguagem de programação C++. Além de orientado a documentos, é formado por um conjunto de documentos JSON. Muitas aplicações podem, dessa forma, modelar informações de modo muito mais natural, pois os dados podem ser aninhados em hierarquias complexas e continuar a ser indexáveis e fáceis de buscar. Pelo formato JSON dos dados e pela simplicidade (visto que a maioria dos dados seriam utilizados em JSON) foi escolhido para armazenar e manipular os dados de GPS dos ônibus. Foi criado um banco "transporte" no MongoDB e uma coleção "onibus" e, pode-se facilmente inserir um documento que representa as informações coletadas pelo GPS do ônibus

4.3. App *BusInRio*

O Framework Ionic é uma SDK para desenvolvimento de aplicativos híbridos capazes de executar em *smartphones* de qualquer sistema operacional e navegadores web apenas utilizando HTML5, Javascript e CSS. Ele é baseado no framework AngularJS e seu núcleo compõe funcionalidades do Apache Cordova. O aplicativo móvel *BusInRio* foi desenvolvido com auxílio deste framework utilizando os dados NoSQL disponibilizados pela API. A aplicação permite ao usuário escolher a linha de ônibus que desejar e consultar a localização dos ônibus daquela linha em tempo real. A Figura 2 ilustra algumas telas da aplicação.

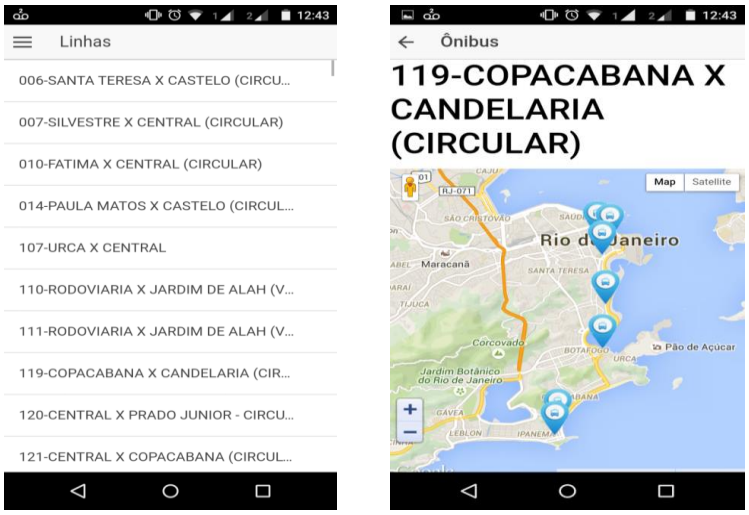

Figura 2. O app *BusInRio* permite que o usuário consulte a lista de linhas disponíveis (esquerda) e a localização e visualização de cada ônibus da linha em tempo real (direita).

5. Resultados e Discussão

Através dos dados abertos disponibilizados pelo data.rio sobre os percursos dos ônibus e GPS em tempo real e com o auxílio de bancos de dados NoSQL MongoDB e Cassandra foi possível estabelecer a ligação entre esses diferentes datasets e fornecer informações úteis ao usuário e de maneira simples.

Através do app *BusInRio* pode-se exibir a localização exata de cada ônibus de uma linha no aplicativo como na Figura 2. Ao analisar os dados de localização e

velocidade de cada ônibus foi possível gerar um relatório do tipo gráfico exibindo um mapa de calor sobre o trânsito na cidade do Rio de Janeiro através do Google Maps.

Neste caso o usuário, através do navegador do seu *smartphone*, pode identificar claramente os pontos onde o tráfego é mais lento como mostra a Figura 3 (esquerda). As regiões onde o trânsito está mais intenso está representado na cor vermelha e onde flui em melhores condições há um gradiente (do vermelho ao verde). Caso a velocidade média de um conjunto de ônibus que percorram um determinado trajeto comum sejam inferiores a 10Km/h o trânsito na região é considerado intenso (vermelho), e superior 20km/h (verde) é considerado bom. Utilizou como valor de referência a velocidade média dos ônibus da cidade de Curitiba-PR (~17Km/h) por ser considerada uma cidade com transporte urbano modelo para o Brasil.

O app *BusInRio* utiliza os dados tratados e armazenados no MongoDB para calcular a velocidade média dos ônibus localizados nas principais regiões da cidade para identificar as condições do trânsito, quanto menor a velocidade média do ônibus, maiores as chances de estrar retido em um congestionamento. Para verificar se os cálculos do *BusInRio*, sobre as condições do trânsito no município do Rio de Janeiro, se aproximavam da realidade efetuamos consultas no Google Maps[7] sobre as condições do transito real calculada por aquele aplicativo e verificamos que houve grande correlação Figura 3 (direita). Na Figura 3 observa-se que o trânsito do dia 13/07/2015 por volta das 11:30h no centro do Rio e nas suas principais vias de acesso (av. Brasil e acessos da Zona Norte) estavam congestionados. Além disso, observa-se congestionamento em partes da Zona Sul da cidade.

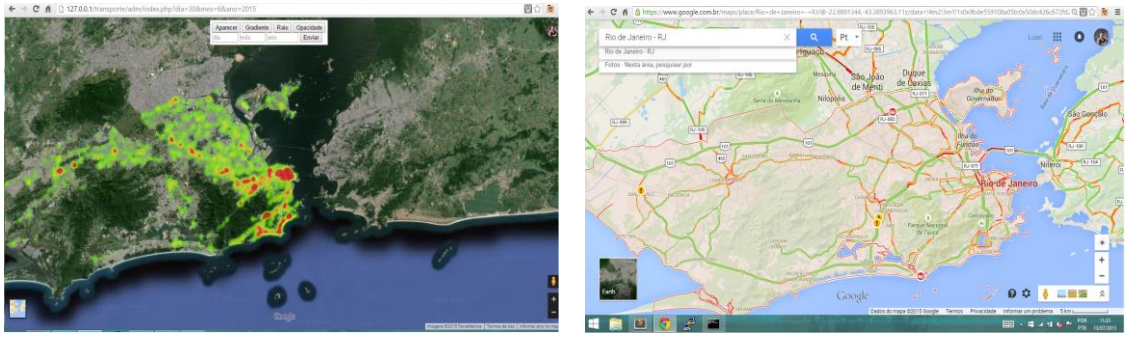

Figura 3. Mapa de calor das condições do trânsito calculadas pelo *BusInRio* (esquerda) em comparação com as condições do trânsito avaliadas pelo Google Maps.

6. Trabalhos Relacionados

As cidades de Chicago (http://www.transitchicago.com/data/), Nova Iorque (https://nycopendata.socrata.com/data?cat=transportation) nos EUA e Londres (https://tfl.gov.uk/info-for/open-data-users/) no Reino Unido publicam através das suas autoridades de trânsito uma variedade de dados abertos sobre transportes públicos e suas rotas para que desenvolvedores de software, pesquisadores, planejadores urbanos, jornalistas, empresas e cidadãos interessados façam uso deles. Por exemplo, na cidade de Nova Iorque existe o app *OnebusAway* apresenta dados de transporte público integrando dados abertos sobre trem, metrô e ônibus, na cidade de Londres o app

[7] https://pt.wikipedia.org/wiki/Google_Maps

CityMapper também está integrado com o metrô. No Rio de Janeiro o app RioBus provê algumas funcionalidades semelhante ao *BusInRio*, no entanto não provê informações sobre o trânsito da cidade. Na cidade de Nova Iorque o grupo de pesquisas da NYU liderado pelos professores Cláudio Silva e Juliana Freire utilizam dados de GPS da frota de taxis da cidade para recomendar não só a oferta deste recurso para o cidadão como também para avaliar as condições do trânsito em função do percurso dos táxis (FERREIRA, 2013).

7. Conclusão

A arquitetura *BusInRio* está totalmente baseada em softwares e dados abertos e em tecnologias NoSQL. A abordagem ETL mostrou-se adequada para o tratamento de dados abertos em ambiente de nuvem. Com relação aos repositórios de dados, verificou-se que o MongoDB se mostrou muito superior na velocidade de resposta das consultas, especialmente considerando-se que tinha um volume muito maior de dados. Como trabalhos futuros é sugerido que a aplicação integre dados de metrô, barcas da CCR e trens da Supervia para oferecer ao usuário uma estimativa aproximada dos percursos e das rotas utilizando outros modais de transportes públicos na cidade do Rio de Janeiro.

8. Referências Bibliográficas

Ferreira, Nivan; Poco, Jorge; Vo, Huy T.; Freire, Juliana and Silva, Claudio T. (2013) "Visual Exploration of Big Spatio-Temporal Urban Data: A Study of New York City Taxi Trips", New York University, Nova Iorque.

Mendonça, Rogers Reiche (2013) "Uma abordagem para coleta e publicação de dados de proveniência no contexto de linked data", Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Moura, Paulo. (2013) "NoSQL: Uma nova necessidade", http://devbrasil.net/profiles/blogs/nosql-uma-nova-necessidade, Acesso em: agosto de 2015.

PORTAL BRASILEIRO DE DADOS ABERTOS. (2013) "Maturidade em dados abertos, entenda as 5 Estrelas", http://dados.gov.br/noticia/maturidade-em-dados-abertos-entenda-as-5-estrelas/, Acesso em: agosto de 2015.