Desafio Data Science

Documento de Arquitetura da Aplicação

Sumário

1.1 Objetivo do Documento

A finalidade deste documento é apresentar a solução que atende a Parte II do desafio de data Science, descrevendo as classes, bibliotecas e softwares necessários para disponibilizar o serviço REST solicitado no desafio.

A segunda parte do desafio consiste em criar um servidor RESTful com endpoints para registrar “edges” e exibir a centralidade do gráfico. O objetivo final é extrair da rede social uma métrica chamada “closeness centrality”.

O código fonte da aplicação esta compactado no arquivo DesafioDataScience.zip, na pasta Part II do GIT HUB.

2 Solução

A solução adotada para o desafio de data Science foi criar um serviço REST utilizando a biblioteca Jersey, que implementa a especificação JSR 311.

REST é um estilo arquitetural baseado em padrões abertos e protocolo HTTP, onde os recursos são acessados através de métodos padrões do HTTP.

A aplicação foi testada em um servidor de aplicação Tomcat 7.0.55.

Foram implementados dois endpoints:

- http://<server>:<port>/DesafioDataScience/edges/node1{node1}/node2/{node2}

Este endpoint possibilita a criação de novos vértices, que são acrescentados no final do arquivo edges.dat. Utiliza uma operação POST.

O método foi implementado através da classe SocialNetworkResource.

Os parâmetros de entrada consistem de 2 strings: node1 e node2, contendo o nome do vértice a ser criado no arquivo edges.dat.

Como retorno, o método devolve uma mensagem indicando o sucesso da operação.

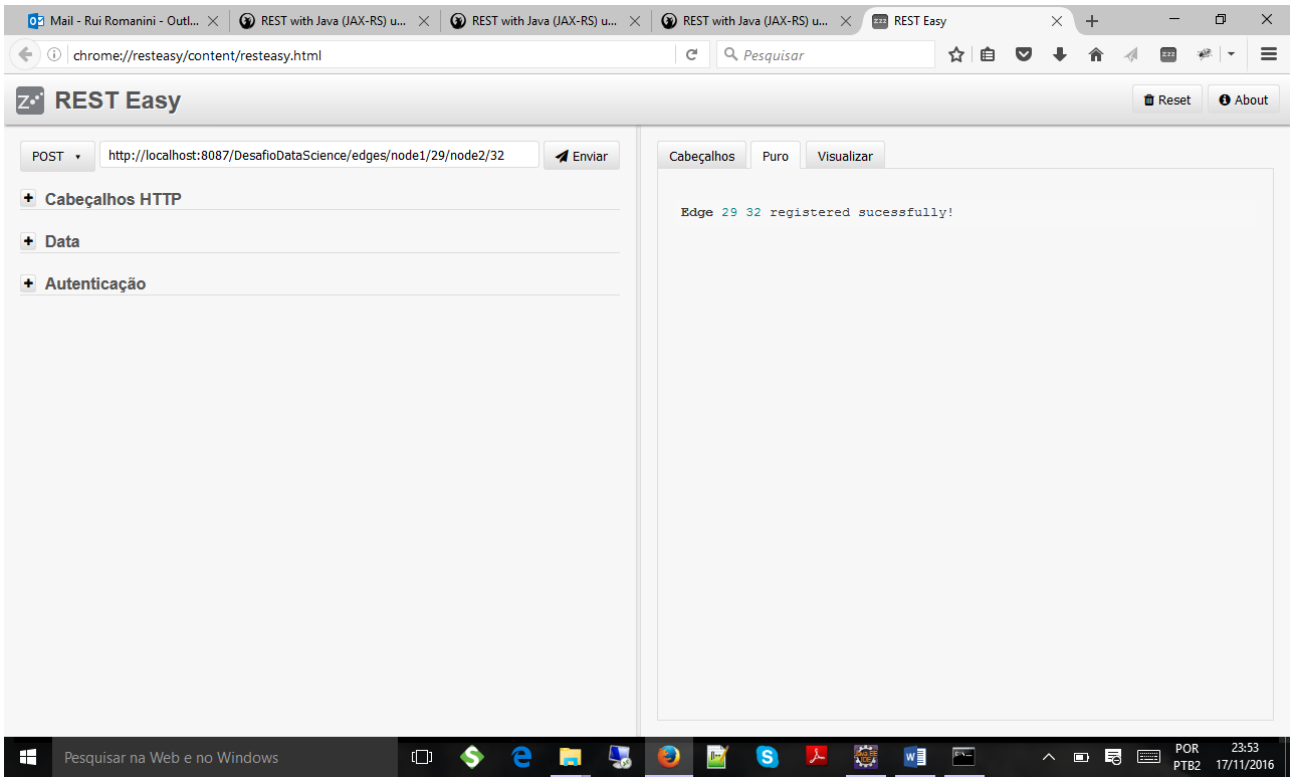

Figura 1 - Teste realizado com o plugin REST Easy no browser Firefox, onde foi incluido um edge com seus 2 vertices: 29 e 32

- http://<server>:<port>/DesafioDataScience/closeness

Este endpoint retorna o score de todos os vértices, ordenados do maior para o menor, tomando como base o arquivo edges.dat . Utiliza uma operação GET.

O método foi implementado através da classe ClosenessCentralityResource.

Não existe parâmetro de entrada. Como saída, o método retorna um json contendo a lista de vértices e seus respectivos scores.

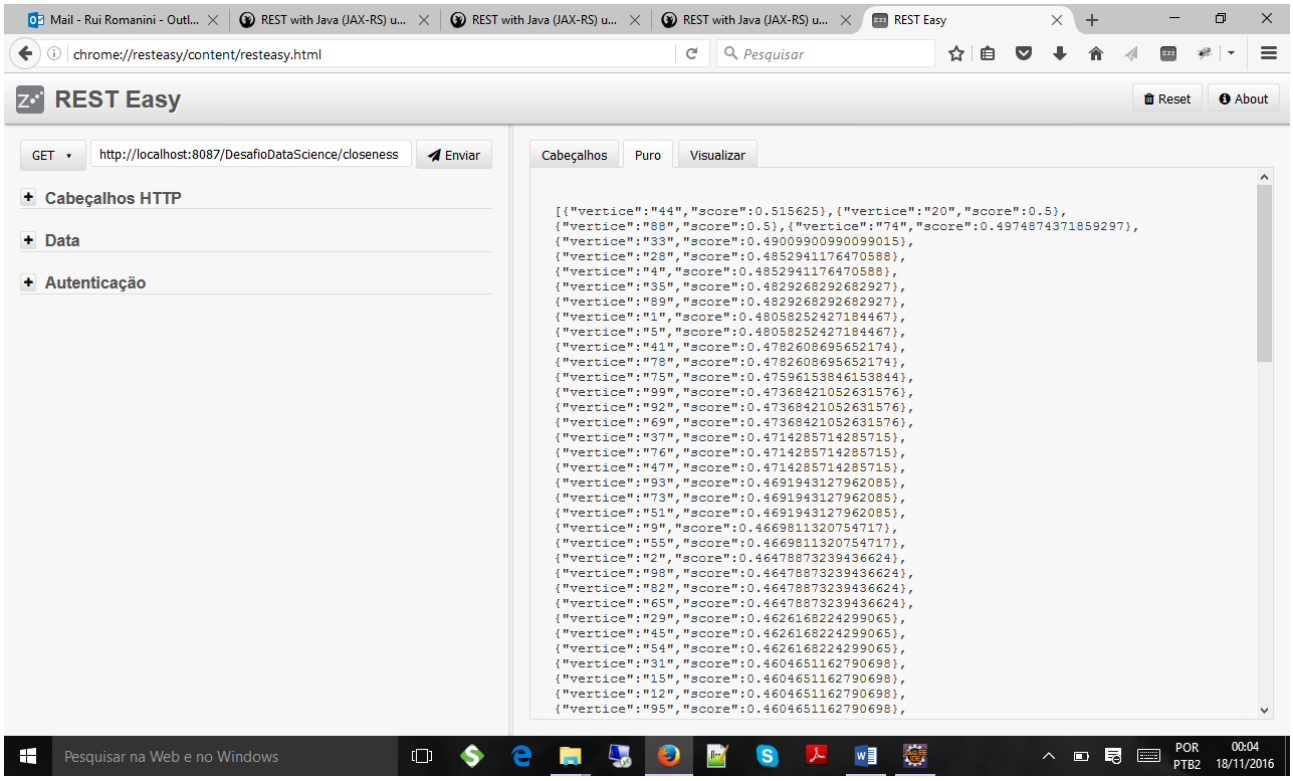

Figura 2 - Teste do endpoint closeness para retorno dos scores do graph representado pelo arquivo edges.dat

Para manuseio da rede e para o cálculo dos scores de “closeness” foi utilizado uma biblioteca JAVA chamada JUNG (http://jung.sourceforge.net/).

A versão do Java utilizada para o desenvolvimento foi a 1.7.0.65

2.1 Arquitetura da Solução

O diagrama a seguir representa as ideias básicas e os candidatos a blocos de construção da solução. Este desenho provê uma visão geral dos principais elementos conceituais e relacionamentos dentro da arquitetura.

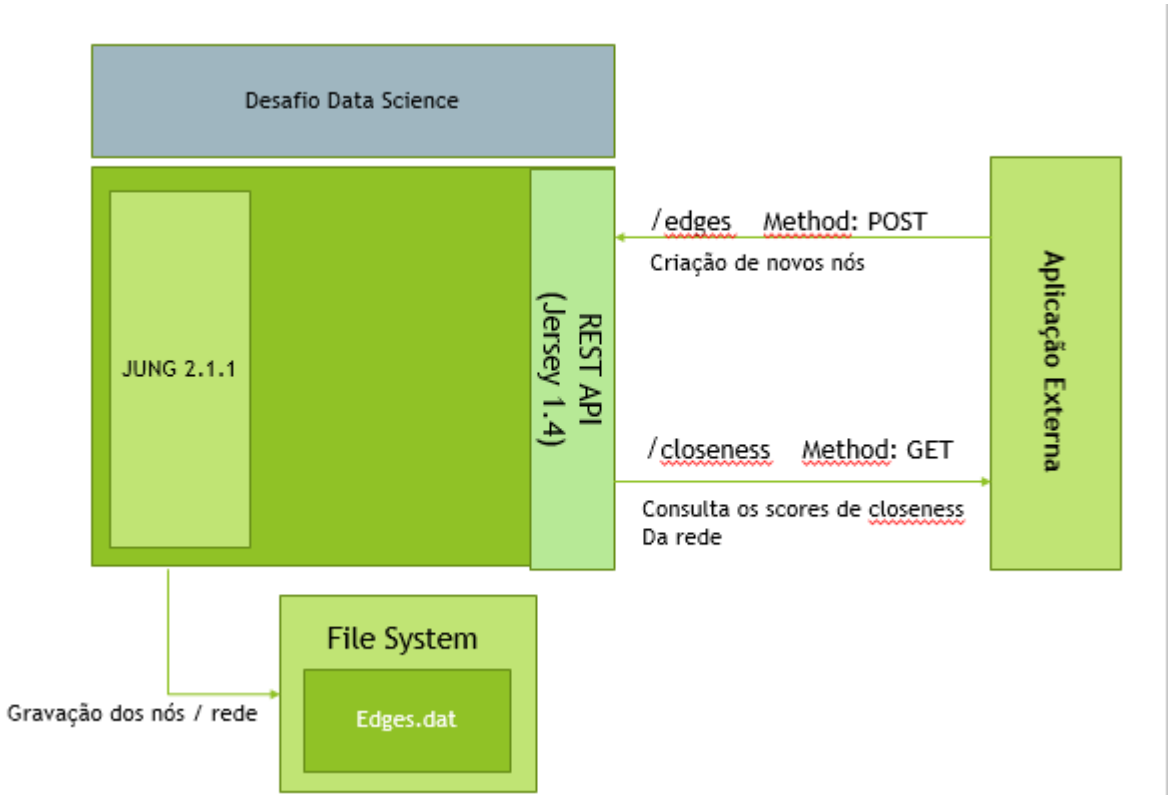

Figura 3 - Diagrama da arquitetura da solução.

3 Softwares e bibliotecas utilizadas na Solução

De forma resumida, é apresentado a seguir as bibliotecas e softwares que compõem a solução.

1. **Apache Tomcat 7.0.55:** utilizado como servidor de aplicação nos testes, por ser leve e simples.

2. **JDK Java 1.7.0_65:** Nenhum recurso avançado do Java foi utilizado

3. **Jersey 1.4:** implementação da especificação REST bastante utilizada. As bibliotecas utilizadas estão disponíveis no arquivo .zip da solução.

4. **JUNG 2.2.1:** Biblioteca Java utilizada para análise de redes sociais.

4 Conclusão

A solução utilizou em sua maioria, bibliotecas comuns de aplicações Java.

A biblioteca JUNG foi utilizada sem muitas dificuldades, já que existiam diversos materiais de consulta à disposição.

O retorno do serviço REST, com os scores da rede analisada batem com a análise realizada na Parte 1 do desafio, utilizando Linguagem R, no que diz respeito a ordenação.