**Instituto Superior de Engenharia do Porto**

****

**Mestrado em Engenharia Informática**

**PSIDI**

**PROGRAMAÇÃO DE SISTEMAS DISTRIBUÍDOS**

**SAD**

**DOCUMENTO DE ARQUITETURA DO SISTEMA**

**GRUPO**

**RESTIFY**

*Daniel Afonso*

*1161660@isep.ipp.pt*

*Leonardo Andrade*

*1160091@isep.ipp.pt*

*Paulo Russo*

*1150285@isep.ipp.pt*

**Docentes:**

*Paulo Gandra de Sousa*

*Luis Miguel Nogueira*

Porto

12 de Janeiro de 2017

**Declaração de originalidade e respeito pelos direitos de autor**

Daniel Afonso, Leonardo Andrade e Paulo Russo, declararam que este trabalho foi por si realizado na íntegra e é original. Confirmam também que o material proveniente de fontes consultadas está devidamente assinalado e foi referenciado na sua totalidade.

[Daniel Afonso]

[Leonardo Andrade]

[Paulo Russo]

**Indice**

[Descrição/Visão geral da API do Sistema 7](#_Toc471573030)

[Descrição da API Datasheet Service 9](#_Toc471573032)

[DESCRIPTION 9](#_Toc471573035)

[Criar user 9](#_Toc471573038)

[Listar todos users 9](#_Toc471573041)

[Obter user 9](#_Toc471573044)

[Modificar user 9](#_Toc471573047)

[Criar dataset 9](#_Toc471573050)

[Listar todos datasets do user 9](#_Toc471573053)

[Obter dataset 9](#_Toc471573056)

[Modificar dataset 9](#_Toc471573059)

[Eliminar dataset 9](#_Toc471573062)

[Criar macro 9](#_Toc471573065)

[Listar macros do user 9](#_Toc471573068)

[Obter macro 9](#_Toc471573071)

[Modificar macro 9](#_Toc471573074)

[Listar estastisticas disponiveis 9](#_Toc471573077)

[Listar tranformações disponiveis 9](#_Toc471573080)

[Obter estatisticas do dataset 9](#_Toc471573083)

[Obter transformações do dataset 9](#_Toc471573086)

[Obter aplicação de macro a um dataset 9](#_Toc471573089)

[Descrição da API Transformation Service 10](#_Toc471573090)

[Introdução 10](#_Toc471573091)

[Objetivos do Trabalho 11](#_Toc471573092)

[Bónus 11](#_Toc471573093)

[Tarefa 11](#_Toc471573094)

[Início 12](#_Toc471573095)

[Definição da arquitetura do sistema 12](#_Toc471573096)

[Definição dos Atores do Sistema 12](#_Toc471573098)

[Definição do Protocolo 12](#_Toc471573099)

[Definição dos Domínios 12](#_Toc471573100)

[Escolha da forma dos dados 12](#_Toc471573101)

[Escolha do tipo de dados 12](#_Toc471573102)

[Desenvolvimento 13](#_Toc471573103)

[Código Datasheet\_srv.js Completo 13](#_Toc471573104)

[Utilização do pacote de passaporte na nossa aplicação 13](#_Toc471573105)

[Armazenamento de Dados 14](#_Toc471573106)

[Criação de Utilizadores da Aplicação 15](#_Toc471573107)

[Conjuntos de dados aleatórios - exemplo inicial 15](#_Toc471573108)

[Código do grupo de funções para calcular Stats ou Transfs e imprimir gráficos numa linha 17](#_Toc471573109)

[Calcular medidas estatísticas de uma linha, coluna, conjunto de dados inteiros 18](#_Toc471573110)

[Armazenamento do Heavy Ops para consulta posterior 18](#_Toc471573111)

[Funções Globais 19](#_Toc471573112)

[Código HeavyOps\_srv.js Completo 40](#_Toc471573113)

[Código para criar um dataset 43](#_Toc471573114)

[Código para criar um user 44](#_Toc471573115)

[Código para criar uma stat 45](#_Toc471573116)

[REST 46](#_Toc471573117)

[Livraria de Código 46](#_Toc471573118)

[Fluxo de Trabalho 46](#_Toc471573119)

[Criar um dataset 46](#_Toc471573120)

[Criar um user 46](#_Toc471573121)

[Atualizando Dados de um Pedido 46](#_Toc471573122)

[Apagar um pedido 47](#_Toc471573123)

[Eventos 47](#_Toc471573124)

[Forma de elementos possíveis 47](#_Toc471573125)

[Lógica do Branching 47](#_Toc471573126)

[Callbacks 47](#_Toc471573127)

[Utilizar callbacks 47](#_Toc471573128)

[Endpoints 47](#_Toc471573129)

[Callback - Segurança do Endpoint 47](#_Toc471573130)

[Autorização de cabeçalho baseado em http 47](#_Toc471573131)

[Autorização 47](#_Toc471573132)

[Controlo de Acesso baseado em Tokens 47](#_Toc471573133)

[Erros de Handling 47](#_Toc471573134)

[Erros de Servidor 47](#_Toc471573135)

[Estado do Sistema 47](#_Toc471573136)

[Teste de integração 47](#_Toc471573137)

[Referências da API 48](#_Toc471573138)

[Anexos 48](#_Toc471573139)

[Diagrama de Arquitetura 48](#_Toc471573140)

[Casos de Uso 49](#_Toc471573142)

[Diagrama de Casos de Uso da Autenticação 49](#_Toc471573143)

[Autenticação 49](#_Toc471573144)

[Registo 49](#_Toc471573145)

[Registo de um utilizador 49](#_Toc471573146)

[49](#_Toc471573147)

[Diagramas de sequência 50](#_Toc471573149)

[Diagrama de sequência – Inserir Dataset 50](#_Toc471573150)

[50](#_Toc471573151)

[Diagrama de sequência – Calcular Estatísticas 51](#_Toc471573153)

[Conclusão 53](#_Toc471573155)

[Bibliografia 54](#_Toc471573156)

**Indice de Figuras**

[Figura 1- Arquitetura do Sistema 5](#_Toc471513718)

[Figura 2- Arquitetura REST 12](#_Toc471573097)

[Figura 3- Diagrama de rquitetura da Aplicação 48](#_Toc471573141)

[Figura 4- Caso de Uso – Autenticação 49](#_Toc471573148)

[Figura 5- Diagrama de sequência – Inserir Dataset 50](#_Toc471573152)

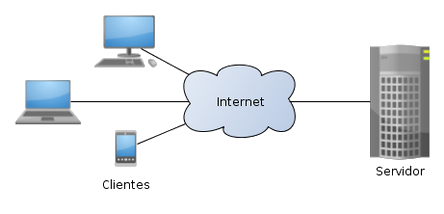
[Figura 6- Diagrama de sequência do Trabalho 52](#_Toc471573154)

**Indice de Tabelas**

[Tabela 1- Tabela de dados 5](#_Toc471506575)

## Descrição/Visão geral da API do Sistema

O presente Sistema utilizará a Arquitetura Cliente-Servidor. Na nossa arquitectura, optamos por desenhar dois serviços, uma interface publica em que disponibiliza todas as operações numa matriz, e outro serviço privado que será utilizado para obter as transformações na matriz.



## Figura 1- Arquitetura do Sistema

## Descrição da API Datasheet Service

|  |  |  |
| --- | --- | --- |
| URL | VERB | DESCRIPTION |
| /Users | POST | Criar user |
| /Users | GET | Listar todos users |
| /Users/{UserID} | GET | Obter user |
| /Users/{UserID} | PUT | Modificar user |
| /Users/{UserID}/Datasets | POST | Criar dataset |
| /Users/{UserID}/Datasets | GET | Listar todos datasets do user |
| /Users/{UserID}/Datasets/{DatasetID} | GET | Obter dataset |
| /Users/{UserID}/Datasets/{DatasetID} | PUT | Modificar dataset |
| /Users/{UserID}/Datasets/{DatasetID} | DELETE | Eliminar dataset |
| /Users/{UserID}/Macros | POST | Criar macro |
| /Users/{UserID}/Macros | GET | Listar macros do user |
| /Users/{UserID}/Macros/{MarcoID} | GET | Obter macro |
| /Users/{UserID}/Macros/{MarcoID} | PUT | Modificar macro |
| /Stats | GET | Listar estastisticas disponiveis |
| /Trnsf | GET | Listar tranformações disponiveis |
| /Users/{UserID}/Datasets/{DatasetID}/Stats | GET | Obter estatisticas do dataset |
| /Users/{UserID}/Datasets/{DatasetID}/Trnsf | GET | Obter transformações do dataset |
| /Users/{UserID}/Datasets/{DatasetID}/{MacroID}/ | GET | Obter aplicação de macro a um dataset |

## Descrição da API Transformation Service

|  |  |  |
| --- | --- | --- |
| URL | VERB | DESCRIPTION |
| /Trnsf/ | POST | Envia dataset com as respectivas operações |

## Introdução

Este Documento da Arquitetura do software descreve uma API REST e os seus recursos. As APIs REST são criadas para developers que desejam integrar a sua aplicação e para administradores que desejam interagir scripts com um servidor.

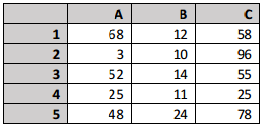
As APIs REST fornecem acesso a recursos (entidades de dados) por meio de caminhos URI. Para usar uma API REST, a aplicação solicitará HTTP e analisará a resposta. O formato de resposta é JSON. Os seus métodos serão os métodos HTTP padrão como GET, PUT, POST e DELETE.

Como a API REST é baseada em padrões abertos, pode-se usar qualquer linguagem de desenvolvimento da Web para acessar a API.

O presente sistema consiste num conjunto de serviços para inserire modificar conjuntos de dados numa tabela (apenas valores numéricos). O sistema deve fornecer alguma forma de autenticação de modo a que cada utilizador possa ter os seus próprios conjuntos de dados.

O utilizador pode trabalhar com vários conjuntos de dados ao mesmo tempo. O cliente deverá poder executar várias análises estatísticas sobre o conjunto de dados (seja na sua totalidade ou apenas numa linha ou coluna).

Tabela 1- Tabela de dados



O sistema deve ter a capacidade de definir um conjunto de transformações a serem aplicadas a um conjunto de dados, guardar transformações e reutilizá-las com outros conjuntos de dados. Esta característica é semelhante para definir uma macro.

## Objetivos do Trabalho

1. Registo de um utilizador
2. Criar um novo conjunto de dados de CSV, XML, JSON
3. Modificar um conjunto de dados existente
4. Retornar um conjunto de dados como CSV, XML, JSON dados
5. Mostrar o número de elementos no conjunto de dados ou linha / coluna selecionada
6. Calcular total da linha
7. Calcular total da coluna
8. Calcular medidas estatísticas de uma linha, coluna, e todo conjunto de dados um(a)

a. média geométrica

b. mediana

c. Modo

d. Média

e. Variação

f. Desvio padrão

1. Execute transformações no conjunto de dados (sem alterar o conjunto de dados original)

a. Transpor o conjunto de dados

b. Escalar

c. Adicionar um escalar

d. Adicionar dois conjuntos de dados

e. Multiplicar dois conjuntos de dados

f. Aumentar o conjunto de dados utilizando uma interpolação linear sobre as linhas ou colunas

10. Retorne uma representação gráfica (imagem de arquivo binário) do conjunto de dados de um

a. gráfico de circular de uma linhas / colunas desejado

b. gráfico de linhas / barras de uma linha / coluna desejado

c. gráfico de linhas / barras de todo o conjunto de dados

11. Definir um macro

12. Executar uma macro num conjunto de dados existente

13. Executar uma macro em um conjunto de dados existente

## Bónus

14. Registar novo trabalhador no registo pool

15. "Batimento cardíaco" de um trabalhador

16. Usar autenticação

## Tarefa

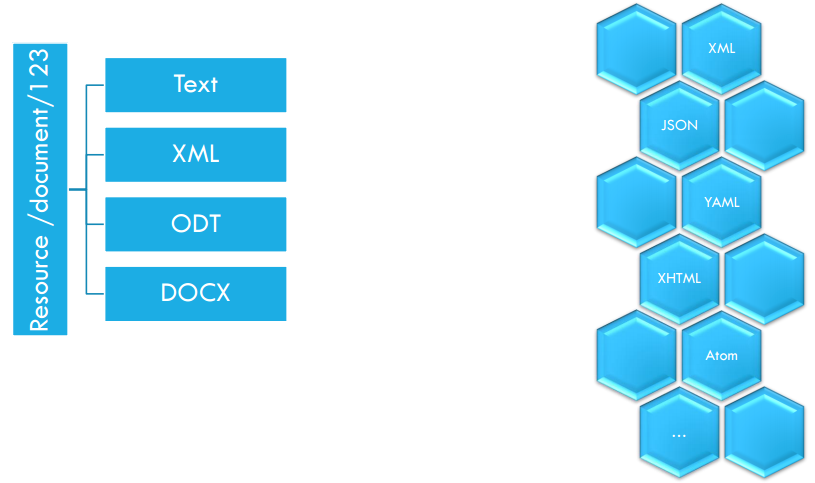
1. Produzir a especificação dos serviços do sistema

2. Desenvolver o protótipo do sistema descrito

# Início

## Definição da arquitetura do sistema

Optou-se por uma arquitetura REST.



## Figura 2- Arquitetura REST

## Definição dos Atores do Sistema

* dataset
* user
* stat
* trnsf

## Definição do Protocolo

Optou-se pela escolha de chamadas assíncronas, para operações de transformação (Callbacks)

## Definição dos Domínios

Definiram-se os domínios URI (Endpoints):

* + /Users/{UserID}/Datasets/{DatasetID}/Stats
  + /Users/{UserID}/Datasets/{DatasetID}/Trnsf

## Escolha da forma dos dados

É requisito do trabalho que os dados sejam passados como XML.

## Escolha do tipo de dados

É requisito do trabalho que os dados estejam no formato Json.

# Desenvolvimento

### Código Datasheet\_srv.js Completo

*/\*\*  
\* Datasheet project  
\* Leonardo Marques de Andrade, Paulo Afonso e Paulo Russo (1160091, 1161660 e 1150285)  
\* PSIDI / MEI / ISEP  
\* (c) 2016  
  
\* Check Readme and Documentation to understand how this servers works  
\* https://bitbucket.org/ODSOFT\_2016\_1160091/restify  
\*\*/  
  
/\*\*\*\*\*\*\*\*\*\*\*\*  
 Global Vars & Constants  
\*\*\*\*\*\*\*\*\*\*\*\*/***var** express = require(**'express'**);  
**var** bodyParser = require(**'body-parser'**);  
*//var methodOverride = require('method-override');***var** request = require(**'request'**);  
  
**var** jsonParser = bodyParser.json();  
**var** json2html = require(**'json-to-html'**)  
**var** matrix = require(**"node-matrix"**)  
**var** get = require(**'simple-object-query'**).**get**;  
**var** where = require(**'simple-object-query'**).where;  
**var** passport = require(**'passport'**);  
  
**var** userController = require(**'./controllers/user'**);  
**var** datasetController = require(**'./controllers/dataset'**);  
**var** authController = require(**'./controllers/auth'**);  
  
**var** connection = require(**'./db/db'**)  
**var** Dataset = require(**'./models/dataset'**);  
  
  
**var** app = express();  
  
app.use(bodyParser.json());  
app.use(bodyParser.urlencoded({**extended**:**true**}));  
*//app.use(methodOverride());*

Utilização do pacote de passaporte na nossa aplicação

*// Use the passport package in our application*app.use(passport.initialize());  
  
**var** callbackApp = express();  
callbackApp.use(bodyParser.json());  
callbackApp.use(bodyParser.urlencoded({**extended**:**true**}));  
  
*// logging : DEBUG  
//app.use(express.logger('dev'));***const** port = process.env.PORT || 3001;  
**const** SERVER\_ROOT = **"http://localhost:"** + port;  
  
**const** serverHeavyOpsPort = process.env.PORT || 3002;  
**const** serverHeavyOps = **"http://localhost:"** + serverHeavyOpsPort;  
  
**const** callbackPort = process.env.PORT || 3005;  
**const** CALLBACK\_ROOT = **"http://localhost:"** + callbackPort;  
  
**var** SequenceID = 1;

## Armazenamento de Dados

*/\*\*\*\*\*\*\*\*\*\*\*\*  
 data store  
\*\*\*\*\*\*\*\*\*\*\*\*/***var** db = mongoose.connection;  
  
db.on(**'error'**, **console**.error);  
db.once(**'open'**, **function**() {  
  
});  
  
**var** datasetSchema = **new** mongoose.Schema({  
 **idDataset**: Number,  
 **numRows**: Number,  
 **numCols**: Number,  
 **values**: [Number]  
});  
**var** Dataset = connection.**model**(**'Dataset'**, datasetSchema);  
  
**var** userSchema = **new** mongoose.Schema({  
 **fullName**: String,  
 **userName**: String,  
 **password**: String  
});  
**var** userSchema = connection.**model**(**'user'**,userSchema);  
  
**var** stat = **new** mongoose.Schema({  
 **stat\_id**: String,  
 **desc\_stat**: String  
})  
**var** statSchema = connection.**model**(**'stat'**, statSchema);  
  
**var** trnsfsSchema = mongoose.Schema({  
 **transf\_id**: String,  
 **desc\_transfs**: String  
});  
**var** trnsfsSchema = connection.**model**(**'trnsfs'**, trnsfSchema);  
  
  
  
**var** connection = mongoose.createConnection(**'mongodb://localhost/datasetdb'**);  
**var** connection = mongoose.createConnection(**'mongodb://localhost/userdb'**);  
**var** connection = mongoose.createConnection(**'mongodb://localhost/statdb'**);  
**var** connection = mongoose.createConnection(**'mongodb://localhost/trnsfsdb'**);  
  
autoIncrement.initialize(connection);  
datasetSchema.plugin(autoIncrement.plugin, { **model**: **'Dataset'**, **field**: **'idDataset'** });  
datasetSchema.plugin(autoIncrement.plugin, { **model**: **'user'**, **field**: **'userid'** });  
datasetSchema.plugin(autoIncrement.plugin, { **model**: **'stat'**, **field**: **'stat\_id'** });  
datasetSchema.plugin(autoIncrement.plugin, { **model**: **'trnsfs'**, **field**: **'trnsf\_id'** });  
  
mongoose.connect(**'mongodb://localhost/datasetdb'**);  
mongoose.connect(**'mongodb://localhost/userdb'**);  
mongoose.connect(**'mongodb://localhost/statdb'**);  
mongoose.connect(**'mongodb://localhost/trnsfsdb'**);  
  
**var** users = {};  
**var** datasets = {};  
**var** datasetTableValues = **""**;  
**var** macros = {};  
**var** stats = {};  
**var** transfs = {};  
**var** charts = {}  
**var** resultsStoreList = [];  
**var** now = **new** Date();

### Criação de Utilizadores da Aplicação

*// INITIAL DATA  
  
//3 users as initial example*users[**'u1'**] = {**username**: **"u1"**, **fullName**:**"Paulo Afonso"**, **Password**:**"node1234"**, **createdOn**: now, **updatedOn**: now};  
users[**'u2'**] = {**username**: **"u2"**, **fullName**:**"Leonardo Andrade"**, **Password**:**"node1234"**, **createdOn**: now, **updatedOn**: now};  
users[**'u3'**] = {**username**: **"u3"**, **fullName**:**"Paulo Russo"**, **Password**:**"node1234"**, **createdOn**: now, **updatedOn**: now};

Conjuntos de dados aleatórios - exemplo inicial  
  
*//3 Random Datasets as initial example***function** *buildRandomDataset*(lines, columns) {  
  
 **var** values = [];  
 **var** dataset\_id = **""**;  
 **var** dataMatrix = matrix({ **rows**: lines, **columns**: columns, **values**: **Math**.random });  
   
 **for**(**var** row = 0; row < dataMatrix.**numRows**; row++) {  
 **for**(**var** col = 0; col < dataMatrix.**numCols**; col++) {  
 values.push(   
 **Math**.round((dataMatrix[row][col]\*100), 3)   
 );  
 }  
 }  
  
 **var** dataset = **new** Dataset({  
 **numRows**: dataMatrix.**numRows**,  
 **numCols**: dataMatrix.**numCols**,  
 **values**: values  
 });  
*/\* var id = mongoose.Types.ObjectId();  
 var dataset = new Dataset({  
 numRows: dataMatrix.numRows,  
 numCols: dataMatrix.numCols,  
 values: values  
 });\*/* dataset.save(  
 **function**(err, dataset) {  
 **if** (err) **return console**.error(err);  
 **console**.log(dataset);  
 dataset\_id = dataset.**idDataset**;  
 }  
 );  
}  
  
**function** *printDatasetHTML*(dataset) {  
  
 **var** values = [];  
 **var** datasetTableValuesFinal = **""**;  
 **for**(**item** = 0; **item** < dataset.**length**; **item**++) {  
  
 **var** dataset\_id = dataset[**item**].**idDataset**;  
 **var** datasetTableValues = **""**;  
 **var** rows = dataset[**item**].**numRows**;  
 **var** cols = dataset[**item**].**numCols**;  
 **var** line = 0;  
 **var** column = 0;  
 **var** arrayPosition = 0;  
   
 **var** tableDatasetError = **"<html><head>"** +  
 **"<style>table { font-family: arial, sans-serif; border-collapse: collapse; width: 100%; } "** +  
 **"td, th { border: 1px solid #dddddd; text-align: center; padding: 8px; } "** +  
 **"tr:nth-child(even) { background-color: #dddddd; } </style> "** +  
 **"</head><body><table>"** +  
 **"<tr>"** +  
 **"<th>Dataset</th>"** +  
 **"</tr>"** +  
 **"<tr>"** +  
 **"<td>Error. There are no Datasets to see yet. Create one first!</td>"** +  
 **"</tr>"** +  
 **"</table></body></html>"  
   
 if** (rows != 0) {  
 **if** (cols != 0) {  
 **const** tableDatasetHead = **"<html><head>"** +  
 **"<style>table { font-family: arial, sans-serif; border-collapse: collapse; width: 100%; } "** +  
 **"td, th { border: 1px solid #dddddd; text-align: center; padding: 8px; } "** +  
 **"tr:nth-child(even) { background-color: #dddddd; } </style> "** +  
 **"</head><body><table style='width:100%' >"**;  
   
 **var** tableDatasetBody =  
 **"<tr>"** +  
 **"<th colspan='"** + (cols + 1 ) + **"'>Dataset ID: "** + dataset\_id + **"</th>"** +  
 **"</tr>"** +  
 **"<tr>"** +  
 **"<td>Row X Col</td>"**;  
 **for**(column = 0; column < cols; column++) {  
   
 tableDatasetBody += **"<td>"** + (column + 1) + **"</td>"**;  
 }  
 tableDatasetBody +=  
 **"</tr>"** +  
 **"<tr>"** ;  
   
 **for**(line = 0; line < rows; line++) {  
 tableDatasetBody += **"<td>"** + (line + 1) + **"</td>"** ;  
 **for**(column = 0; column < cols; column++) {  
 tableDatasetBody += **"<td>"** + dataset[**item**].**values**[arrayPosition] + **"</td>"** ;  
 arrayPosition++  
 }  
 tableDatasetBody +=  
 **"</tr>"** +  
 **"<tr>"** ;  
 }  
   
 **const** tableDatasetTail = **"</tr></table></body></html>"**;  
 datasetTableValues = tableDatasetHead + tableDatasetBody + tableDatasetTail;  
 } **else** {  
 datasetTableValues = tableDatasetError;  
 }  
 } **else** {  
 datasetTableValues = tableDatasetError;  
 }  
 datasetTableValuesFinal = datasetTableValuesFinal.concat(datasetTableValues);  
 }  
 **return** datasetTableValuesFinal;  
}  
*buildRandomDataset*(2, 2);  
*buildRandomDataset*(3, 5);  
*buildRandomDataset*(2, 7);

Código do grupo de funções para calcular Stats ou Transfs e imprimir gráficos numa linha  *//A group of functions to Calculate Stats or Transfs and Prints Charts in a row*macros[**'m1'**] = {**content**: **"s1,t1,c1"**, **createdOn**: now, **updatedOn**: now};  
macros[**'m2'**] = {**content**: **"s1,t1,c1,s2,t2,c2"**, **createdOn**: now, **updatedOn**: now};

### Calcular medidas estatísticas de uma linha, coluna, conjunto de dados inteiros

*//Calculate statistical measures of a row, column, entire data set*stats[**'s1'**] = {**stat\_id**: **"s1"**, **desc\_stat**:**"Geometric mean"** };  
stats[**'s2'**] = {**stat\_id**: **"s2"**, **desc\_stat**:**"Median"** };  
stats[**'s3'**] = {**stat\_id**: **"s3"**, **desc\_stat**:**"Mode"** };  
stats[**'s4'**] = {**stat\_id**: **"s4"**, **desc\_stat**:**"Midrange"** };  
stats[**'s5'**] = {**stat\_id**: **"s5"**, **desc\_stat**:**"Variance"** };  
stats[**'s6'**] = {**stat\_id**: **"s6"**, **desc\_stat**:**"Standard deviation"**};  
  
Transformações executadas no conjunto de dados (sem alterar o conjunto de dados original)

*//Perform transformations on the data set (without changing the original data set)*transfs[**'t1'**] = {**transf\_id**: **"t1"**, **desc\_transfs**:**"Transpose the dataset"** };  
transfs[**'t2'**] = {**transf\_id**: **"t2"**, **desc\_transfs**:**"Scale"** };  
transfs[**'t3'**] = {**transf\_id**: **"t3"**, **desc\_transfs**:**"Add a scalar"** };  
transfs[**'t4'**] = {**transf\_id**: **"t4"**, **desc\_transfs**:**"Add two data sets"** };  
transfs[**'t5'**] = {**transf\_id**: **"t5"**, **desc\_transfs**:**"Multiply two data sets"** };  
transfs[**'t6'**] = {**transf\_id**: **"t6"**, **desc\_transfs**:**"Augment the data set using linear interpolation on the rows or columns"**,};  
  
Devolução de uma representação de gráfico (arquivo binário de imagem) do conjunto de dados

*//Return a chart representation (image binary file) of the dataset*charts[**'c1'**] = {**chart\_id**: **"c1"**, **desc\_chart**:**"Pie chart of a desired row / column"**};  
charts[**'c2'**] = {**chart\_id**: **"c2"**, **desc\_chart**:**"Line / bar chart of a desired row / column"**};  
charts[**'c3'**] = {**chart\_id**: **"c3"**, **desc\_chart**:**"Line / bar chart of the entire data set"**};

Armazenamento do Heavy Ops para consulta posterior  
*//Store Heavy Ops for later consulting*resultsStoreList [1] = {**ResultNumber**: **"No results from Heavy Ops to see yet"**}

### Funções Globais

*/\*\*\*\*\*\*\*\*\*\*\*\*  
global functions  
\*\*\*\*\*\*\*\*\*\*\*\*/***function** *buildMessageCreation*(newID, text, user){  
 **const** now = **new** Date();  
 **return** {  
 **id** : newID,   
 **text** : text,  
 **sender** : user,   
 **createdOn** : now,  
 };  
}  
  
**function** *buildMessageUpdate*(newID, text, user){  
 **const** now = **new** Date();  
 **return** {  
 **id** : newID,   
 **text** : text,  
 **sender** : user,   
 **updatedOn**: now,  
 };  
}  
  
Função que incrementa Callbacks automaticamente

*/\*  
 \* Function for auto-increment Callbacks ID  
 \*/***function** *getSequence*(seqtype) {  
 **if** (seqtype = **"stID"**)  
 **return** SequenceID++;   
}

*// Create our Express router***var** router = express.Router();  
*// Register all our routes with /api*app.use(**'/'**, router);  
  
  
router.route(**'/Users'**)  
 .post(userController.postUsers)  
 .**get**(userController.getUsers);  
  
*//  
//URL: /Users  
//  
//GET return all users, returns 200  
//POST create new entry, returns 201 or 400  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//  
  
/\*app.route("/Users")  
 .get(function(req, res) {  
 //****TODO = Develop here what happens*** *console.log("»»» Accepted GET to this resource. Develop here what happens");  
   
 res.json(users);  
 })  
 .post(function(req, res) {  
 //for debug  
 //console.log(req.body.username + req.body.password);  
 if (req.body.username && req.body.password) {  
   
 //****TODO = Develop here what happens*** *console.log("»»» Accepted POST to this resource. Develop here what happens");  
   
   
 // send 201 response  
 res.statusCode = 201;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "The username: " + req.body.username + " was successfully created! " +  
 "</h1></body></html>");  
 console.log("»»» Username: " + req.username + " was successfully created!");  
  
 }  
 else {  
 res.statusCode = 400;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "Bad request. Check the definition documentation. " +  
 "</h1></body></html>");  
 console.log("»»» Bad request. Check the definition documentation.");   
 }  
 })  
 .put(function(req, res) {  
 res.statusCode = 405;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "Method not allowed in this resource. Check the definition documentation " +  
 "</h1></body></html>");  
 })  
 .delete(function(req, res) {  
 res.statusCode = 405;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "Method not allowed in this resource. Check the definition documentation " +  
 "</h1></body></html>");  
 });\*/  
  
/\*\*  
 \* URL: /Users/:userID  
 \* GET return specific user 200 or 404  
 \* POST not allowed, returns 405  
 \* PUT overwrite data for existent user, returns 200, 400 or 404   
 \* DELETE delete an user, returns 200 or 404  
\*\*/*app.param(**'userID'**, **function**(req, res, next, userID){  
req.**username** = userID;  
**return** next()  
})  
app.route(**"/Users/:userID"**)   
 .**get**(**function**(req, res) {  
 *//for debug  
 //console.log(req.body.fullName + req.username + req.body.password);* **if** (req.**username**) {  
 **console**.log(**"»»» Accepted GET to this resource. Develop here what happens"**);  
 res.**json**(users[req.**username**])  
 } **else** {  
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"User "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» User "** + req.**username** + **" not found!"**);  
 }  
   
 })  
 .put(**function**(req, res) {  
 *//for debug  
 //console.log(req.body.fullName + req.username + req.body.password);* **if** (req.**username** && req.**body**.**fullName** && req.**body**.**password** ||   
 req.**username** && req.**body**.**password** ||  
 req.**username** && req.**body**.**fullName**) {  
   
 *//****TODO = Develop here what happens* console**.log(**"»»» Accepted PUT to this resource. Develop here what happens"**);  
   
   
 *// send 200 response* res.**statusCode** = 200;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"User "** + req.**username** + **" successfully updated! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» User "** + req.**username** + **" successfully updated!"**);  
 }  
 **else** {  
 **if** (req.**username** === **undefined**) {  
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"User "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» User "** + req.**username** + **" not found!"**);   
 } **else** {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);  
 }  
 }  
 })  
 .post(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 **var** entry = users[req.**username**];  
 **if** (entry === **undefined**) {  
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"User "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» User "** + req.**username** + **" not found!"**);  
 }  
 **else** {  
 **delete** users[req.**username**];  
 res.**statusCode** = 204 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"User "** + req.**username** + **" successfully deleted! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» User "** + req.**username** + **" successfully deleted!"**);  
 }  
 });  
  
  
*///DATASETS  
  
//  
//URL: /Users/:userID/Datasets  
//  
//GET return all users, returns 200  
//POST create new entry, returns 201 or 400  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405*app.route(**"/Users/:userID/Datasets"**)  
 .post(datasetController.postDataset)  
 .**get**(datasetController.getDataset);  
  
 */\*.get(function(req, res) {  
 console.log("»»» Accepted GET to .../Datasets/ resource");  
 //var result = "There are no Datasets to this user";  
 var result = "";  
 var allDatasets = "";  
 for(i=0; i < Object.keys(datasets).length; i++){  
 allDatasets = "d" + (i+1) + ".datasetValues" ;  
 result += get(datasets, allDatasets) ;  
 }  
 res.statusCode = 200;  
 res.setHeader("Content-Type", "application/html");  
 Dataset.find(function (err, datasets) {  
 if (err) return console.error(err);  
 //console.log(datasets);  
 res.end(printDatasetHTML(datasets));  
 });   
 console.log("»»» Returned GET for all existents Datasets");  
 })  
 .post(function(req, res) {  
 //for debug  
 //console.log(req.username + req.body.dataset\_id);  
 //inserir function para iterar os valores informados para criar o dataset  
 //inserir function para validar se a quantidade linhas x colunas fazem match com os valores informados  
   
 if (Number (req.body.rows) && Number (req.body.cols) && req.body.values ) {  
 console.log("entrei no full");  
 //var id = mongoose.Types.ObjectId();  
 var dataset = new Dataset({  
 rows: req.body.rows,  
 cols: req.body.cols,  
 values: req.body.values  
 });  
  
 dataset.saveDataset(function(err, dataset) {  
 if (err) return console.error(err);  
 console.dir(dataset);  
 });  
 // send 201 response  
 res.statusCode = 201;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "The specific Dataset: " + dataset\_id + " was successfully created for username: " + req.username +  
 "</h1></body></html>");  
 console.log("»»» The specific Dataset: " + dataset\_id + " was successfully created for username: " + req.username);  
 }   
 else {  
 if (Number (req.body.rows) && Number (req.body.cols) ) {  
   
 buildRandomDataset(req.body.rows, req.body.cols);  
   
 // send 201 response  
 res.statusCode = 201;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "A Random Dataset for username: " + req.username + " was successfully created. Your DatasetID = " + dataset\_id +   
 "</h1></body></html>");  
 console.log("»»» A Random Dataset for username: " + req.username + " was successfully created. Your DatasetID = " + dataset\_id);  
 }   
 else {  
   
 res.statusCode = 400;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "Bad request. Check the definition documentation. " +  
 "</h1></body></html>");  
 console.log("»»» Bad request. Check the definition documentation.");  
 }  
 }  
 })  
 .put(function(req, res) {  
 res.statusCode = 405;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "Method not allowed in this resource. Check the definition documentation " +  
 "</h1></body></html>");  
 })  
 .delete(function(req, res) {  
 res.statusCode = 405;  
 res.setHeader("Content-Type", "application/html");  
 res.end("<html><body><h1> " +  
 "Method not allowed in this resource. Check the definition documentation " +  
 "</h1></body></html>");  
 });\*/  
  
//  
//URL: /Users/:userID/Datasets/:datasetID  
//  
//GET return specific user 200 or 404  
//POST not allowed, returns 405  
//PUT overwrite data for existent user, returns 200, 400 or 404   
//DELETE delete an user, returns 200 or 404*app.param(**'datasetID'**, **function**(req, res, next, datasetID){  
 req.**dataset\_id** = datasetID;  
 **return** next()  
 })  
  
app.route(**"/Users/:userID/Datasets/:datasetID"**)   
 .**get**(**function**(req, res) {  
  
 **if** (req.**dataset\_id**) {  
 **console**.log(**"»»» Accepted GET to /Datasets/ID? resource. "**);  
 res.**statusCode** = 200;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 *//res.end( datasets[req.dataset\_id].datasetValues );* Dataset.find({ **idDataset**: req.**dataset\_id** },**function** (err, dataset) {  
 **if** (err) **return console**.error(err);  
 **console**.log(dataset);  
 res.end(*printDatasetHTML*(dataset));  
 })  
  
 } **else** {  
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Datase: or User "** + req.**dataset\_id** + **" or "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Dataset or User "** + req.**dataset\_id** + **" or "** + req.**username** + **" not found! "**);  
 }  
   
 })  
 .put(**function**(req, res) {  
 *//for debug  
 //console.log(req.body.fullName + req.username + req.body.password);* **if** (req.**username** && req.**dataset\_id**) {  
   
 *//****TODO = Develop here what happens* console**.log(**"»»» Accepted PUT to this resource. Develop here what happens"**);  
   
 *// send 200 response* res.**statusCode** = 200;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Dataset: "** + req.**dataset\_id** + **" successfully updated! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Dataset: "** + req.**dataset\_id** + **" successfully updated!"**);  
 }  
 **else** {  
 **if** (req.**username** === **undefined** || req.**dataset\_id** === **undefined** ) {  
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Dataset or User "** + req.**dataset\_id** + **" or "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Dataset or User "** + req.**dataset\_id** + **" or "** + req.**username** + **" not found! "**);   
 } **else** {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);  
 }  
 }  
 })  
 .post(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 **if** (req.**username** === **undefined** || req.**dataset\_id** === **undefined** ) {  
   
 **console**.log(**"»»» Accepted DELETE to this resource. Develop here what happens"**);  
   
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Dataset or User "** + req.**dataset\_id** + **" or "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Dataset or User "** + req.**dataset\_id** + **" or "** + req.**username** + **" not found! "**);  
 }  
 **else** {  
 **if** (req.**username** && req.**dataset\_id**) {  
   
 **delete** datasets[req.**dataset\_id**];  
   
 **console**.log(**"»»» Accepted DELETE to this resource. Develop here what happens"**);  
   
 res.**statusCode** = 200 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Dataset: "** + req.**dataset\_id** + **" successfully deleted for username: "** + req.**username** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Nada a ver "** + req.**dataset\_id** + **" successfully deleted for username: "** + req.**username**);  
 } **else** {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);  
 }  
 }  
 });  
  
*///MACROS  
  
//  
//handling individual items in the collection  
//  
//URL: /Users/:userID/Macros  
//  
//GET return all users, returns 200  
//POST create new entry, returns 201 or 400  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//*app.route(**"/Users/:userID/Macros"**)   
 .**get**(**function**(req, res) {  
 *//for debug  
 //console.log(req.username);* **console**.log(**"»»» Accepted GET to this resource. Develop here what happens"**);  
 res.**json**(macros);  
   
 })  
 .post(**function**(req, res) {  
 *//for debug  
 //console.log(req.username + req.body.macro\_id);  
 //inserir function para iterar os valores informados para criar a macro  
 //inserir function para validar se a quantidade linhas x colunas fazem match com os valores informados* **if** (req.**username** && req.**body**.**macro\_id**) {  
 **if** (req.**body**.**stat\_id** || req.**body**.**transf\_id** || req.**body**.**chart\_id**) {  
   
 *//****TODO = Develop here what happens* console**.log(**"»»» Accepted POST to this resource. Develop here what happens"**);  
   
 *// send 201 response* res.**statusCode** = 201;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"The Macro: "** + req.**body**.**macro\_id** + **" was successfully created for username: "** + req.**username** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Macro: "** + req.**body**.**macro\_id** + **" was successfully created for username: "** + req.**username**);  
 }  
 }  
 **else** {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);   
 }  
 })  
 .put(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 });  
  
*//  
//handling individual items in the collection  
//  
//URL: /Users/:userID/Macros/:macroID  
//  
//GET return specific user 200 or 404  
//POST not allowed, returns 405  
//PUT overwrite data for existent user, returns 200, 400 or 404   
//DELETE delete an user, returns 200 or 404  
//*app.param(**'macroID'**, **function**(req, res, next, macroID){  
 req.**macro\_id** = macroID;  
 **return** next()  
 })  
  
app.route(**"/Users/:userID/Macros/:macroID"**)   
 .**get**(**function**(req, res) {  
 *//for debug  
 //console.log(req.username + req.macro\_id);* **if** (req.**macro\_id**) {  
 **console**.log(**"»»» Accepted GET to this resource. Develop here what happens"**);  
 res.**json**(macros[req.**macro\_id**]);  
 } **else** {  
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Macro or User "** + req.**macro\_id** + **" or "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Macro or User "** + req.**macro\_id** + **" or "** + req.**username** + **" not found! "**);  
 }  
   
 })  
 .put(**function**(req, res) {  
 *//for debug  
 //console.log(req.username + req.macro\_id);* **if** (req.**username** && req.**macro\_id**) {  
 **if** (req.**body**.**stat\_id** || req.**body**.**transf\_id** || req.**body**.**chart\_id**) {  
   
 *//****TODO = Develop here what happens* console**.log(**"»»» Accepted PUT to this resource. Develop here what happens"**);  
   
 *// send 200 response* res.**statusCode** = 200;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Macro: "** + req.**macro\_id** + **" successfully updated! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Macro: "** + req.**macro\_id** + **" successfully updated!"**);  
 }  
 }  
 **else** {  
 **if** (req.**username** === **undefined** || req.**macro\_id** === **undefined** ) {  
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Macro: or User "** + req.**macro\_id** + **" or "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Macro: or User "** + req.**macro\_id** + **" or "** + req.**username** + **" not found! "**);   
 } **else** {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);  
 }  
 }  
 })  
 .post(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 **if** (req.**username** === **undefined** || req.**macro\_id** === **undefined** ) {  
   
 **console**.log(**"»»» Accepted DELETE to this resource. Develop here what happens"**);  
   
 res.**statusCode** = 404 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Macro or User "** + req.**macro\_id** + **" or "** + req.**username** + **" not found! "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Macro or User "** + req.**macro\_id** + **" or "** + req.**username** + **" not found! "**);  
 }  
 **else** {  
 **if** (req.**username** && req.**macro\_id**) {  
   
 **delete** macros[req.**macro\_id**];  
   
 **console**.log(**"»»» Accepted DELETE to this resource. Develop here what happens"**);  
   
 res.**statusCode** = 200 ;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Macro: "** + req.**macro\_id** + **" successfully deleted for username: "** + req.**username** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Macro: "** + req.**macro\_id** + **" successfully deleted for username: "** + req.**username**);  
 } **else** {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);  
 }  
 }  
 });  
  
*//  
//handling individual items in the collection  
//  
//URL: /Stats  
//  
//GET return specific user 200 or 400  
//POST not allowed, returns 405  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//*app.route(**"/Stats"**)   
 .**get**(**function**(req, res) {  
 *//for debug  
 //console.log(req.username);* **console**.log(**"»»» Accepted GET to this resource. Develop here what happens"**);  
 res.**json**(stats);  
 })  
 .put(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .post(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
  
*//  
//handling individual items in the collection  
//  
//URL: /Transfs  
//  
//GET return specific user 200 or 400  
//POST not allowed, returns 405  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//*app.route(**"/Transfs"**)   
 .**get**(**function**(req, res) {  
 *//for debug  
 //console.log(req.username);* **console**.log(**"»»» Accepted GET to this resource. Develop here what happens"**);  
 res.**json**(transfs);  
 })  
 .put(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .post(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
   
*//  
//handling individual items in the collection  
//  
//URL: /Charts  
//  
//GET return specific user 200 or 400  
//POST not allowed, returns 405  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//*app.route(**"/Charts"**)   
 .**get**(**function**(req, res) {  
 *//for debug  
 //console.log(req.username);* **console**.log(**"»»» Accepted GET to this resource. Develop here what happens"**);  
 res.**json**(charts);  
 })  
 .put(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .post(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.**statusCode** = 405;   
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
  
*//  
//URL: /Results  
//  
//GET return specific user 200  
//POST not allowed, returns 405  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//*app.route(**"/Results"**)   
 .**get**(**function**(req, res) {  
   
 **var** stringList = **""**;  
 **console**.log(**"»»» Accepted GET to this resource. Develop here what happens "**);  
   
 **for**(**var** i = 1; i < resultsStoreList.**length**;i++) {  
 stringList += **"<p>"**+ json2html(resultsStoreList[i]) + **"</p>"**;  
 }  
 **console**.log(stringList);

res.**statusCode** = 200;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> Results stored untill now </h1>"** +  
 stringList +  
 **"</body></html>"**);   
 })  
 .put(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .post(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
  
*///HEAVY OPS  
  
//  
//URL: /callback/:myRefID  
//  
//internal usage  
//*callbackApp.route(**"/callback/:myRefID"**)   
 .post(**function**(req, res) {  
 *// reply back* res.**status**(204).send(**"No Content"**);  
 *// process the response to our callback request  
 // handle callbacks that are not sent by our server "security". postman can't invoke this endpoint directly.  
 //persists the result in the resultsStoreList[].* **console**.log( **"The result of callback number "** + req.params.myRefID + **" is "** + req.**body**.myRefValue );  
   
 **var** resultJson = {};  
 resultJson.**key** = req.params.myRefID;  
 resultJson.**value** = req.**body**.myRefValue;  
   
 resultsStoreList [req.params.myRefID] = resultJson;  
 **console**.log(**"»»» Received a callback request with: "** + req.**body**.**result** + **" for cliRef = "** + req.params.myRefID + **" Develop here what happens!!!"**);  
 });  
  
  
*//  
//URL: /Users/:userID/Datasets/:datasetID/:statID  
//  
//GET not allowed, returns 405  
//POST return specific user 202 or 400  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//  
  
/\*app.param('statID', function(req, res, next, statID){  
 req.stat\_id = statID;  
 return next()  
 })\*/*app.route(**"/Users/:userID/Datasets/:datasetID/Stats"**)  
 .post(**function**(req, res) {  
  
 **console**.log(**"»»» Accepted POST request to calculate statID: "** + req.query.StatID + **"for DatasetID: "** + req.**dataset\_id** + **" and UserID: "** + req.**username** + **" Develop here what happens"**);  
 **if** (req.**username** && req.**dataset\_id** && req.query.StatID ) {  
 **callbackID** = *getSequence*(**"stID"**);  
 **var** urlCallback = CALLBACK\_ROOT + **"/Users/"** + req.**username** + **"/Datasets/"** + req.**dataset\_id** + **"/Stats/"**+req.query.StatID+**"/Results"  
  
 var** datasetV = **""**;  
 Dataset.find({ **idDataset**: req.**dataset\_id** },**function** (err, dataset) {  
 **if** (err) **return console**.error(err);  
 **console**.log(dataset);  
 datasetV = dataset[0];  
 *//setTimeout(function() {* request({  
 **uri** : serverHeavyOps + **"/HeavyOps/"** + req.**username** + **"/"** + req.**dataset\_id** + **"/"** + req.query.StatID,  
 **method**: **"POST"**,  
 **json** : {**text**:**"test of callback post"**, **sender**:**"Datasheet\_srv.js"**, **callbackURL**: urlCallback, **myRef**:**callbackID** , **dataset**:datasetV},  
 },  
 **function**(err, res, body){  
  
 **if** (!err && 202 === res.**statusCode**) {  
 **console**.log(**"»»» Posted a Heavy Operation request and got "** + res.**statusCode** );  
 **console**.log(**"»»» Success!... Gets your callback results within 30 seconds in "** + urlCallback );  
 res.**statusCode** = 202;  
 } **else** {  
 **console**.log(**"»»» Internal error in HeavyOps server. Please contact system administrator. Status Code = "** + res.**statusCode**);  
 }  
 });  
 *//}, 2000);* })  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"<p>Success!... Your request operation number is "** + **callbackID** + **"</p>"** +  
 **"<p>This is a heavy operation so gets your callback result within 30 seconds in <a href='"** + urlCallback + **"'"** + **">Results</a></p>"** +  
 **"<p>Or come back to Home Page to request more operations <a href='http://localhost:3001/index.html'>Home Page</a></p>"** +  
 **"</h1></body></html>"**);  
 } **else** {  
 **if** (req.**username** === **undefined** || req.**dataset\_id** === **undefined** || req.**stat\_id** === **undefined**) {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);  
 }   
 }  
 })  
 .**get**(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .put(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
  
  
  
callbackApp.route(**"/Users/:userID/Datasets/:datasetID/Stats/:statID/Results/:callbackID"**)  
 .**get**(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .post(**function**(req, res) {  
 *// reply back* res.**status**(204).send(**"No Content"**);  
 *// process the response to our callback request  
 // handle callbacks that are not sent by our server "security". postman can't invoke this endpoint directly.  
 //persists the result in the resultsStoreList[].* **console**.log( **"The result of dataset"**+ req.params.statID +**" callback number "** + req.params.callbackID + **" is "** + req.**url** );  
  
 **var** resultJson = {};  
 resultJson.**key** = req.**url**;  
 resultJson.**value** = req.**body**.myRefValue;  
  
 resultsStoreList [req.params.myRefID] = resultJson;  
 **console**.log(**"»»» Received a callback request with: "** + req.**body**.**result** + **" for cliRef = "** + req.**url** + **" Develop here what happens!!!"**);  
 });  
 *//  
//URL: /Users/:userID/Datasets/:datasetID/:transfID  
//  
//GET not allowed, returns 405  
//POST return specific user 202 or 400  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//*app.param(**'transfID'**, **function**(req, res, next, transfID){  
 req.**transf\_id** = transfID;  
 **return** next()  
 })  
   
app.route(**"/Users/:userID/Datasets/:datasetID/:transfID"**)   
 .post(**function**(req, res) {  
 **console**.log(**"»»» Accepted POST request to calculate transfID: "** + req.**transf\_id** + **" for DatasetID: "** + req.**dataset\_id** + **" and UserID: "** + req.**username** + **" Develop here what happens"**);  
 **if** (req.**username** && req.**dataset\_id** && req.**transf\_id** ) {  
 **callbackID** = *getSequence*(**"stID"**);  
   
 *//setTimeout(function() {* request({  
 **uri** : serverHeavyOps + **"/HeavyOps/"** + req.**username** + **"/"** + req.**dataset\_id** + **"/"** + req.**transf\_id**,  
 **method**: **"POST"**,  
 **json** : {**text**:**"test of callback post"**, **sender**:**"Datasheet\_srv.js"**, **callbackURL**: CALLBACK\_ROOT + **"/callback/"**, **myRef**:**callbackID**},  
 },   
 **function**(err, res, body){  
   
 **if** (!err && 202 === res.**statusCode**) {  
 **console**.log(**"»»» Posted a Heavy Operation request and got "** + res.**statusCode** );  
 **console**.log(**"»»» Success!... Gets your callback results within 30 seconds in http://localhost:3001/Results/"** + **callbackID** );  
 res.**statusCode** = 202;  
 } **else** {  
 **console**.log(**"»»» Internal error in HeavyOps server. Please contact system administrator. Status Code = "** + res.**statusCode**);  
 }  
 });  
 *//}, 2000);* res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"<p>Success!... Your request operation number is "** + **callbackID** + **"</p>"** +  
 **"<p>This is a heavy operation so gets your callback result within 30 seconds in <a href='"** + **"http://localhost:3001/Results/"** + **"'"** + **">Results</a></p>"** +  
 **"<p>Or come back to Home Page to request more operations <a href='http://localhost:3001/index.html'>Home Page</a></p>"** +  
 **"</h1></body></html>"**);  
 } **else** {  
 **if** (req.**username** === **undefined** || req.**dataset\_id** === **undefined** || req.**transf\_id** === **undefined**) {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);  
 }   
 }  
 })  
 .**get**(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .put(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
  
*//  
//URL: /Users/:userID/Datasets/:datasetID/:macroID  
//  
//GET not allowed, returns 405  
//POST return specific user 202 or 400  
//PUT not allowed, returns 405  
//DELETE not allowed, returns 405  
//*app.route(**"/Users/:userID/Datasets/:datasetID/:macroID"**)   
 .post(**function**(req, res) {  
 **console**.log(**"»»» Accepted POST request to calculate transf\_id: "** + req.**macro\_id** + **" for DatasetID: "** + req.**dataset\_id** + **" and UserID: "** + req.**username** + **" Develop here what happens"**);  
 **if** (req.**username** && req.**dataset\_id** && req.**macro\_id** ) {  
 **callbackID** = *getSequence*(**"stID"**);  
   
 *//setTimeout(function() {* request({  
 **uri** : serverHeavyOps + **"/HeavyOps/"** + req.**username** + **"/"** + req.**dataset\_id** + **"/"** + req.**macro\_id**,  
 **method**: **"POST"**,  
 **json** : {**text**:**"test of callback post"**, **sender**:**"Datasheet\_srv.js"**, **callbackURL**: CALLBACK\_ROOT + **"/callback/"**, **myRef**:**callbackID**},  
 },   
 **function**(err, res, body){  
   
 **if** (!err && 202 === res.**statusCode**) {  
 **console**.log(**"»»» Posted a Heavy Operation request and got "** + res.**statusCode** );  
 **console**.log(**"»»» Success!... Gets your callback results within 30 seconds in http://localhost:3001/Results/"** + **callbackID** );  
 res.**statusCode** = 202;  
 } **else** {  
 **console**.log(**"»»» Internal error in HeavyOps server. Please contact system administrator. Status Code = "** + res.**statusCode**);  
 }  
 });  
 *//}, 2000);* res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"<p>Success!... Your request operation number is "** + **callbackID** + **"</p>"** +  
 **"<p>This is a heavy operation so gets your callback result within 30 seconds in <a href='"** + **"http://localhost:3001/Results/"** + **"'"** + **">Results</a></p>"** +  
 **"<p>Or come back to Home Page to request more operations <a href='http://localhost:3001/index.html'>Home Page</a></p>"** +  
 **"</h1></body></html>"**);  
 } **else** {  
 **if** (req.**username** === **undefined** || req.**dataset\_id** === **undefined** || req.**macro\_id** === **undefined**) {  
 res.**statusCode** = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 **console**.log(**"»»» Bad request. Check the definition documentation."**);  
 }   
 }  
 })  
 .**get**(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .put(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.**statusCode** = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
  
*/\*  
 \* RUNNING  
 \*/  
   
   
// STARTING ...*app.listen(port, **function**() {  
 **console**.log(**"Listening requests on "** + port);  
});  
  
*//STARTING callback*callbackApp.listen(callbackPort, **function**() {  
 **console**.log(**"Listening callbacks on "** + callbackPort);  
});

### Código HeavyOps\_srv.js Completo

*/\*\*  
\* Datasheet project  
\* Leonardo Marques de Andrade, Daniel Afonso e Paulo Russo (1160091, 1161660 e 1150285)  
\* PSIDI / MEI / ISEP  
\* (c) 2016  
  
\* Check Readme and Documentation to understand how this servers works  
\* https://bitbucket.org/ODSOFT\_2016\_1160091/restify  
\*\*/  
  
/\*\*\*\*\*\*\*\*\*\*\*\*  
 Global Vars & Constants  
\*\*\*\*\*\*\*\*\*\*\*\*/***var** express = require(**'express'**);  
**var** bodyParser = require(**'body-parser'**);  
*//var methodOverride = require('method-override');***var** request = require(**'request'**);  
  
  
**var** app = express();  
  
app.use(bodyParser.json());  
app.use(bodyParser.urlencoded({extended:**true**}));  
*//app.use(methodOverride());  
  
// logging : DEBUG  
//app.use(express.logger('dev'));***const** port = process.env.PORT || 3002;  
**const** SERVER\_ROOT = **"http://localhost:"** + port;  
  
*/\*\*\*\*\*\*\*\*\*\*\*\*  
global functions  
\*\*\*\*\*\*\*\*\*\*\*\*/***function** calculation(callbackURL, myRefValue, oper\_id, dataset\_id, username) {  
 *// POST message  
 //Calculate here de stat. You can/should do internal requests to Datasheet\_srv to enhance the calculation* **var** myRefURLcallback = callbackURL +**"/"**+ myRefValue;  
 **var** result = **"1 + 1 = 2"**;  
   
 **var** resultJson = {**'key'** : **'value'**};  
 resultJson.key = myRefValue;  
 resultJson.value = result;  
   
 request({  
 uri : myRefURLcallback,  
 method: **"POST"**,  
 json : {myRefValue: result},   
 },  
 **function**(err, res, body){   
 **if** (!err) {  
 console.log(**"»»» Posted callback successfully in the URL: "** + myRefURLcallback + **" and the result: "** + result + **" and got StatusCode: "** + res.statusCode);  
 **if** (204 != res.statusCode ) {  
 console.log(**"»»» Internal error in Datasheet server. Please contact system administrator "**);  
 }   
 }  
 **else** {  
 console.log(**"»»» Unknown Error. Maybe Datasheet server is unavailable. Please contact system administrator \n"** + err);  
 }  
 });  
   
}  
  
*/\*\*  
 \* URL: /:userID/:datasetID/:operID  
 \* GET return specific user 200 or 404  
 \* POST not allowed, returns 405  
 \* PUT overwrite data for existent user, returns 200, 400 or 404   
 \* DELETE delete an user, returns 200 or 404  
\*\*/*app.param(**'userID'**, **function**(req, res, next, userID){  
 req.username = userID;  
 **return** next()  
 })  
  
app.param(**'datasetID'**, **function**(req, res, next, datasetID){  
 req.dataset\_id = datasetID;  
 **return** next()  
 })  
  
app.param(**'operID'**, **function**(req, res, next, operID){  
 req.oper\_id = operID;  
 **return** next()  
 })  
  
app.route(**"/HeavyOps/:userID/:datasetID/:operID"**)   
 .post(**function**(req, res) {  
   
 console.log(**"»»» Accepted POST request to calculate operID: "** + req.oper\_id + **" for DatasetID: "** + req.dataset\_id + **" and UserID: "** + req.username + **" Develop here what happens"**);  
   
 *//validate mandatory body fields in this IF* **if** (req.body.callbackURL && req.body.myRef) {  
   
 *// queue the request - handle it when possible - remove it after tests* calculation(req.body.callbackURL, req.body.myRef, req.oper\_id, req.dataset, req.username);  
  
 *// send 202 Accepted* res.statusCode = 202;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +   
 **"Accepted POST request to calculate operID: "** + req.oper\_id + **" for DatasetID: "** + req.dataset\_id + **"and UserID: "** + req.unsername +   
 **"<h2><br>Your CallbackID is: "** + req.body.myRef + **"</br></h2>"** +  
 **"</h1></body></html>"**);  
   
 console.log(**"»»» When the calculation finish I will POST callback to Ref: "**+ req.body.myRef);   
 } **else** {  
 res.statusCode = 400;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Bad request. Check the definition documentation. "** +  
 **"</h1></body></html>"**);  
 console.log(**"»»» Bad request. Check the definition documentation."**);  
 }  
   
 })  
 .get(**function**(req, res) {  
 res.statusCode = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .put(**function**(req, res) {  
 res.statusCode = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
 .delete(**function**(req, res) {  
 res.statusCode = 405;  
 res.setHeader(**"Content-Type"**, **"application/html"**);  
 res.end(**"<html><body><h1> "** +  
 **"Method not allowed in this resource. Check the definition documentation "** +  
 **"</h1></body></html>"**);  
 })  
   
   
*// STARTING ...*app.listen(port, **function**() {  
 console.log(**"Listening requests on "** + port);  
});

### Código para criar um dataset

dataset.save(  
 **function**(err, dataset) {  
 **if** (err) **return console**.error(err);  
 **console**.log(dataset);  
 dataset\_id = dataset.**idDataset**;  
 }  
 );  
}  
  
**function** *printDatasetHTML*(dataset) {  
  
 **var** values = [];  
 **var** datasetTableValuesFinal = **""**;  
 **for**(**item** = 0; **item** < dataset.**length**; **item**++) {  
  
 **var** dataset\_id = dataset[**item**].**idDataset**;  
 **var** datasetTableValues = **""**;  
 **var** rows = dataset[**item**].**numRows**;  
 **var** cols = dataset[**item**].**numCols**;  
 **var** line = 0;  
 **var** column = 0;  
 **var** arrayPosition = 0;  
   
 **var** tableDatasetError = **"<html><head>"** +  
 **"<style>table { font-family: arial, sans-serif; border-collapse: collapse; width: 100%; } "** +  
 **"td, th { border: 1px solid #dddddd; text-align: center; padding: 8px; } "** +  
 **"tr:nth-child(even) { background-color: #dddddd; } </style> "** +  
 **"</head><body><table>"** +  
 **"<tr>"** +  
 **"<th>Dataset</th>"** +  
 **"</tr>"** +  
 **"<tr>"** +  
 **"<td>Error. There are no Datasets to see yet. Create one first!</td>"** +  
 **"</tr>"** +  
 **"</table></body></html>"  
   
 if** (rows != 0) {  
 **if** (cols != 0) {  
 **const** tableDatasetHead = **"<html><head>"** +  
 **"<style>table { font-family: arial, sans-serif; border-collapse: collapse; width: 100%; } "** +  
 **"td, th { border: 1px solid #dddddd; text-align: center; padding: 8px; } "** +  
 **"tr:nth-child(even) { background-color: #dddddd; } </style> "** +  
 **"</head><body><table style='width:100%' >"**;  
   
 **var** tableDatasetBody =  
 **"<tr>"** +  
 **"<th colspan='"** + (cols + 1 ) + **"'>Dataset ID: "** + dataset\_id + **"</th>"** +  
 **"</tr>"** +  
 **"<tr>"** +  
 **"<td>Row X Col</td>"**;  
 **for**(column = 0; column < cols; column++) {  
   
 tableDatasetBody += **"<td>"** + (column + 1) + **"</td>"**;  
 }  
 tableDatasetBody +=  
 **"</tr>"** +  
 **"<tr>"** ;  
   
 **for**(line = 0; line < rows; line++) {  
 tableDatasetBody += **"<td>"** + (line + 1) + **"</td>"** ;  
 **for**(column = 0; column < cols; column++) {  
 tableDatasetBody += **"<td>"** + dataset[**item**].**values**[arrayPosition] + **"</td>"** ;  
 arrayPosition++  
 }  
 tableDatasetBody +=  
 **"</tr>"** +  
 **"<tr>"** ;  
 }  
   
 **const** tableDatasetTail = **"</tr></table></body></html>"**;  
 datasetTableValues = tableDatasetHead + tableDatasetBody + tableDatasetTail;  
 } **else** {  
 datasetTableValues = tableDatasetError;  
 }  
 } **else** {  
 datasetTableValues = tableDatasetError;  
 }  
 datasetTableValuesFinal = datasetTableValuesFinal.concat(datasetTableValues);  
 }  
 **return** datasetTableValuesFinal;  
}  
*buildRandomDataset*(2, 2);  
*buildRandomDataset*(3, 5);  
*buildRandomDataset*(2, 7);

### Código para criar um user

*// INITIAL DATA  
  
//3 users as initial example*users[**'u1'**] = {**username**: **"u1"**, **fullName**:**"Paulo Afonso"**, **Password**:**"node1234"**, **createdOn**: now, **updatedOn**: now};  
users[**'u2'**] = {**username**: **"u2"**, **fullName**:**"Leonardo Andrade"**, **Password**:**"node1234"**, **createdOn**: now, **updatedOn**: now};  
users[**'u3'**] = {**username**: **"u3"**, **fullName**:**"Paulo Russo"**, **Password**:**"node1234"**, **createdOn**: now, **updatedOn**: now};

### Código para criar uma stat

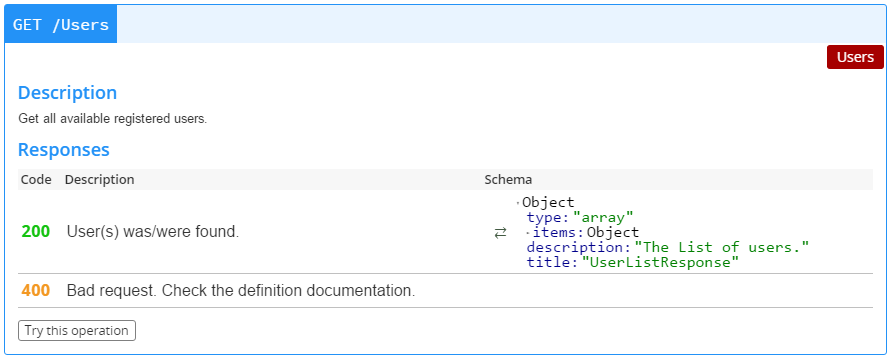
*//A group of functions to Calculate Stats or Transfs and Prints Charts in a row*macros[**'m1'**] = {**content**: **"s1,t1,c1"**, **createdOn**: now, **updatedOn**: now};  
macros[**'m2'**] = {**content**: **"s1,t1,c1,s2,t2,c2"**, **createdOn**: now, **updatedOn**: now};  
  
*//Calculate statistical measures of a row, column, entire data set*stats[**'s1'**] = {**stat\_id**: **"s1"**, **desc\_stat**:**"Geometric mean"** };  
stats[**'s2'**] = {**stat\_id**: **"s2"**, **desc\_stat**:**"Median"** };  
stats[**'s3'**] = {**stat\_id**: **"s3"**, **desc\_stat**:**"Mode"** };  
stats[**'s4'**] = {**stat\_id**: **"s4"**, **desc\_stat**:**"Midrange"** };  
stats[**'s5'**] = {**stat\_id**: **"s5"**, **desc\_stat**:**"Variance"** };  
stats[**'s6'**] = {**stat\_id**: **"s6"**, **desc\_stat**:**"Standard deviation"**};

Criar uma trnsf

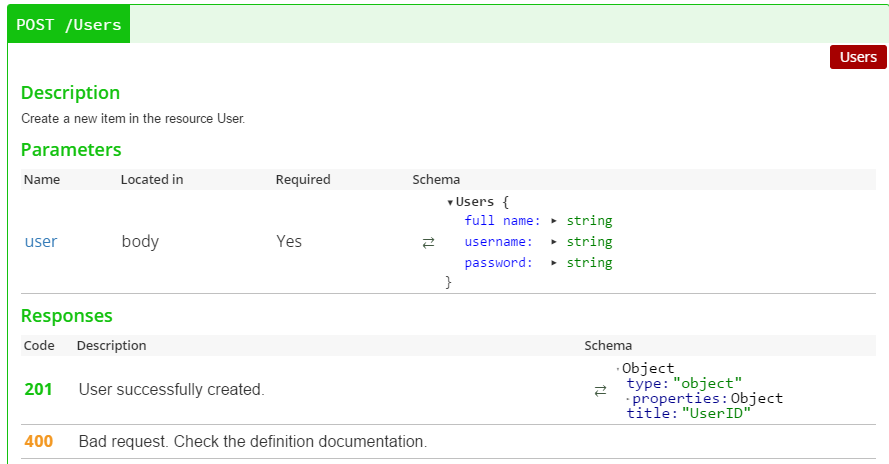
*//Perform transformations on the data set (without changing the original data set)*transfs[**'t1'**] = {**transf\_id**: **"t1"**, **desc\_transfs**:**"Transpose the dataset"** };  
transfs[**'t2'**] = {**transf\_id**: **"t2"**, **desc\_transfs**:**"Scale"** };  
transfs[**'t3'**] = {**transf\_id**: **"t3"**, **desc\_transfs**:**"Add a scalar"** };  
transfs[**'t4'**] = {**transf\_id**: **"t4"**, **desc\_transfs**:**"Add two data sets"** };  
transfs[**'t5'**] = {**transf\_id**: **"t5"**, **desc\_transfs**:**"Multiply two data sets"** };  
transfs[**'t6'**] = {**transf\_id**: **"t6"**, **desc\_transfs**:**"Augment the data set using linear interpolation on the rows or columns"**,};  
  
*//Return a chart representation (image binary file) of the dataset*charts[**'c1'**] = {**chart\_id**: **"c1"**, **desc\_chart**:**"Pie chart of a desired row / column"**};  
charts[**'c2'**] = {**chart\_id**: **"c2"**, **desc\_chart**:**"Line / bar chart of a desired row / column"**};  
charts[**'c3'**] = {**chart\_id**: **"c3"**, **desc\_chart**:**"Line / bar chart of the entire data set"**};

## Recursos do Swagger

* **USERS**
* **Método GET**

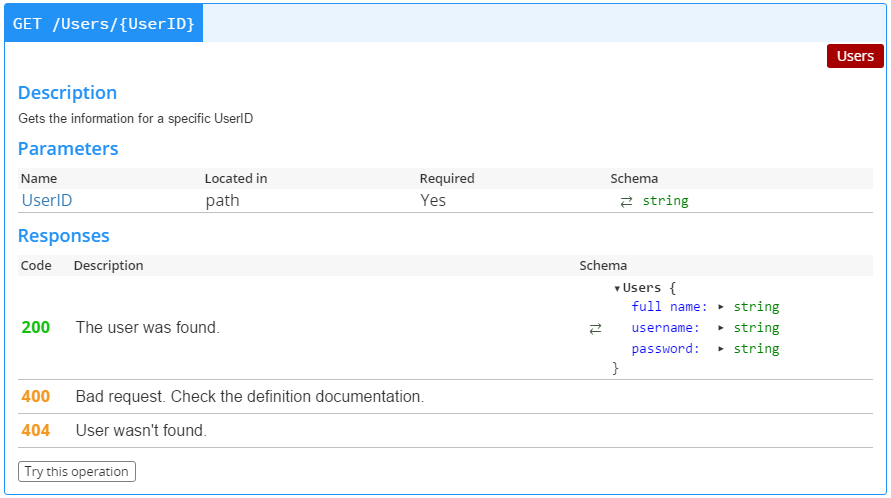


* **Método POST**

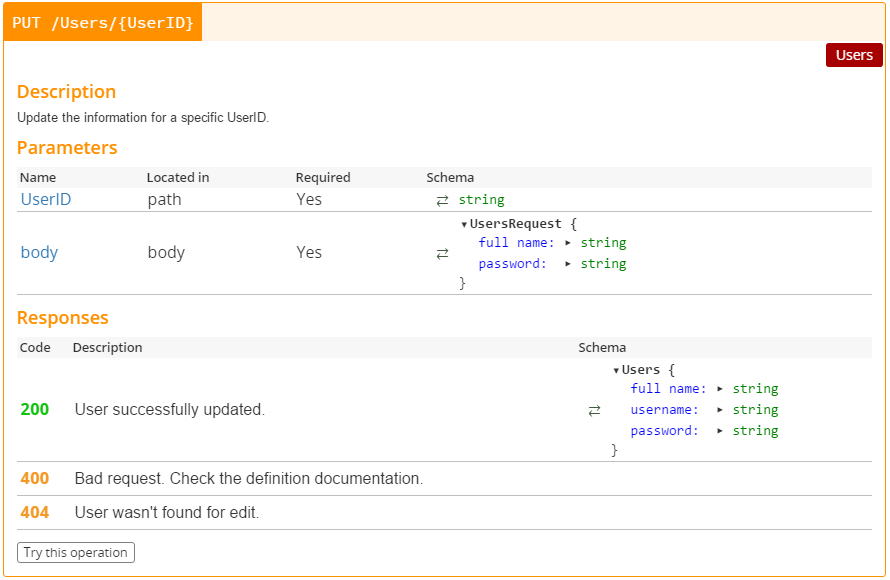


### Figura 2- Printscreen do Swagger POST /Users

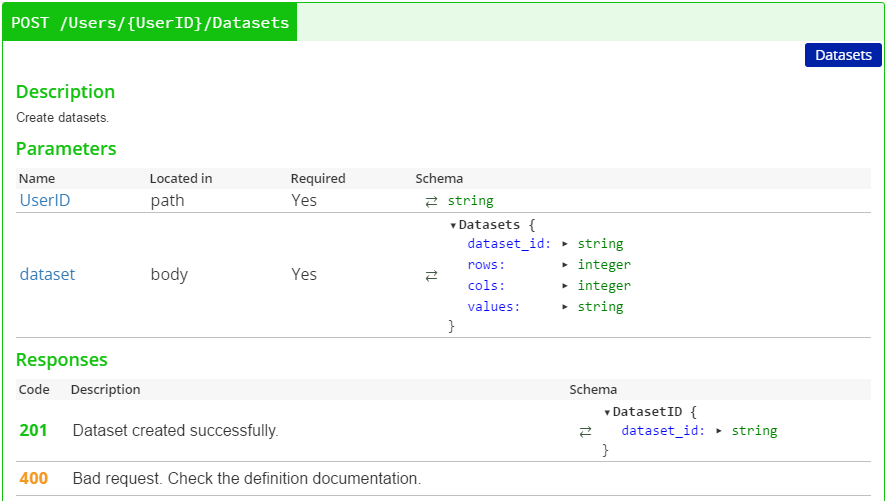
* **/Users/{UserID}**
* **Método GET**



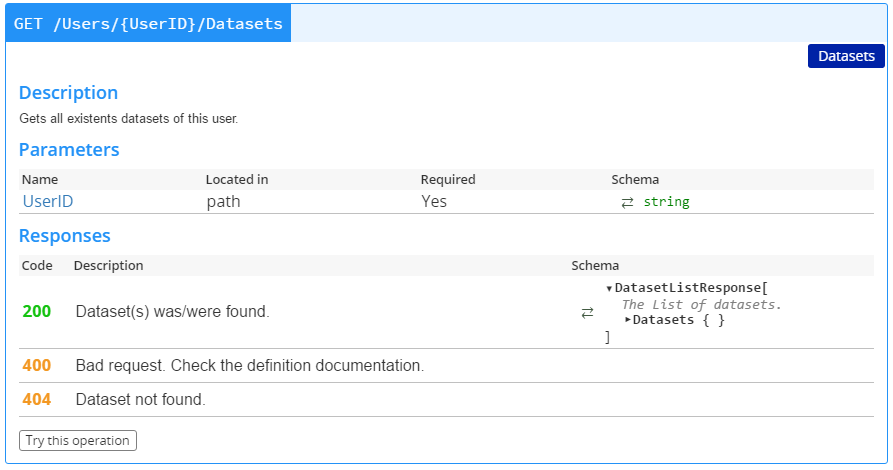
* **Método PUT**



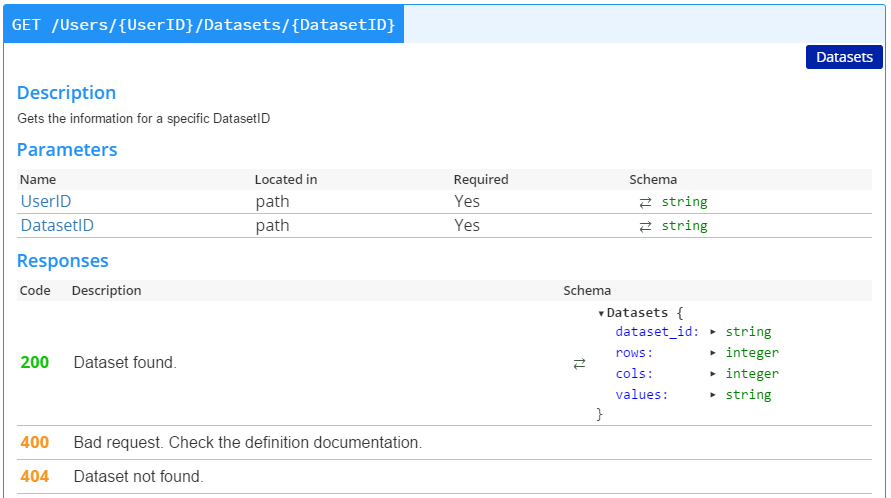
* **/Users/{UserID}/Datasets**
* **Método POST**



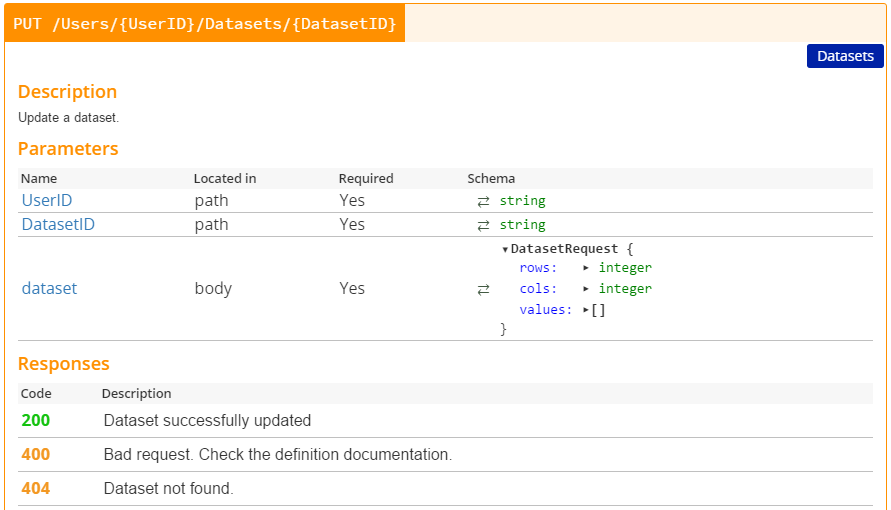
* **Método GET**



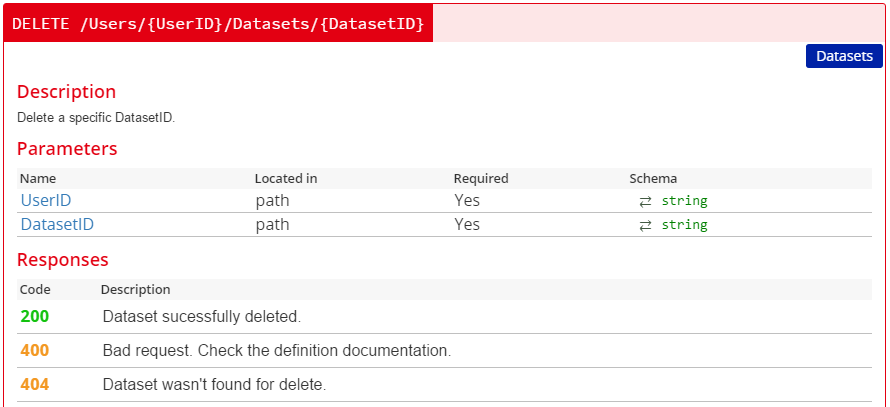
* **/Users/{UserID}/Datasets/{DatasetID}**
* **Método GET**



* **Método PUT**



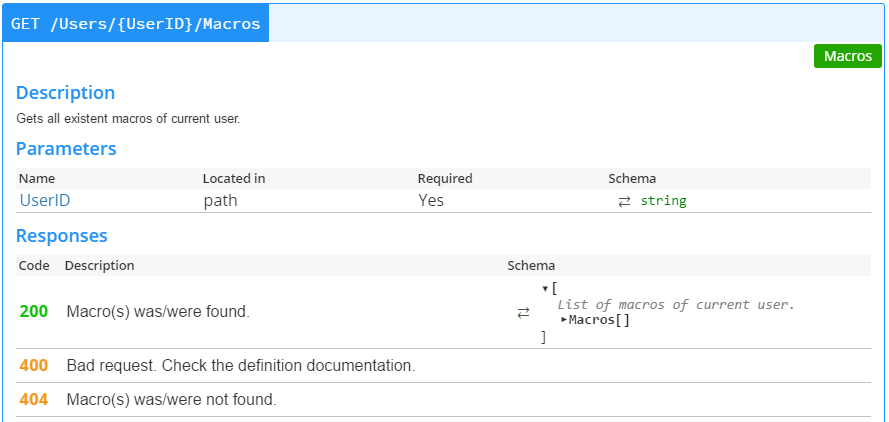
* **Método DELETE**



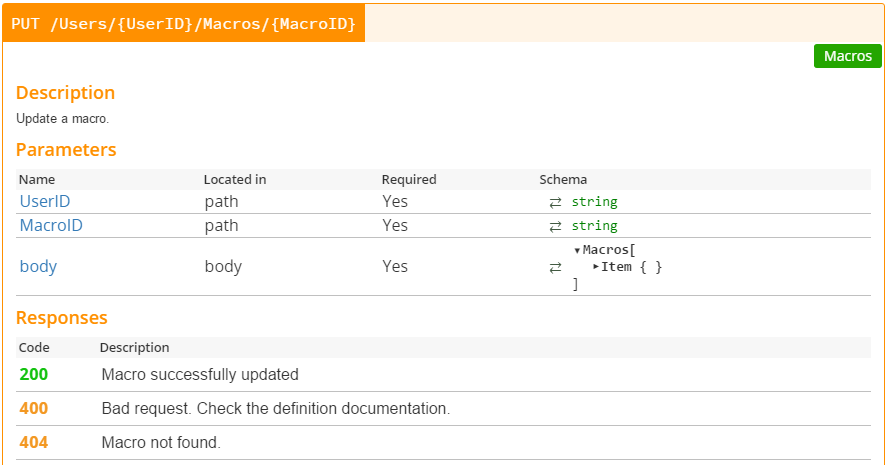
* **/Users/{UserID}/Macros**
* **Método POST**



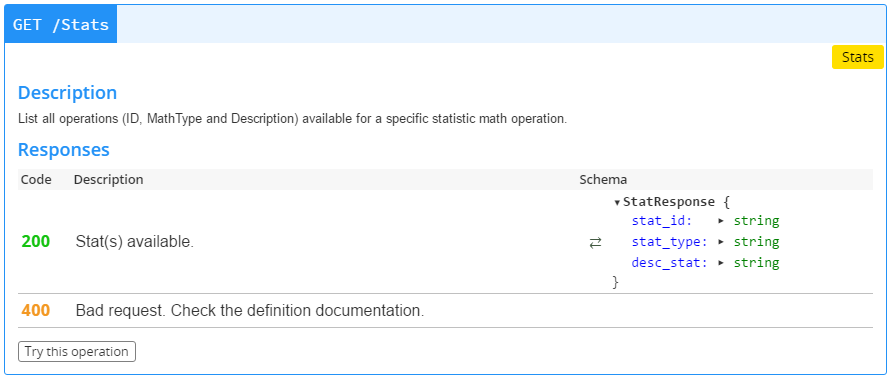
* **Método GET**



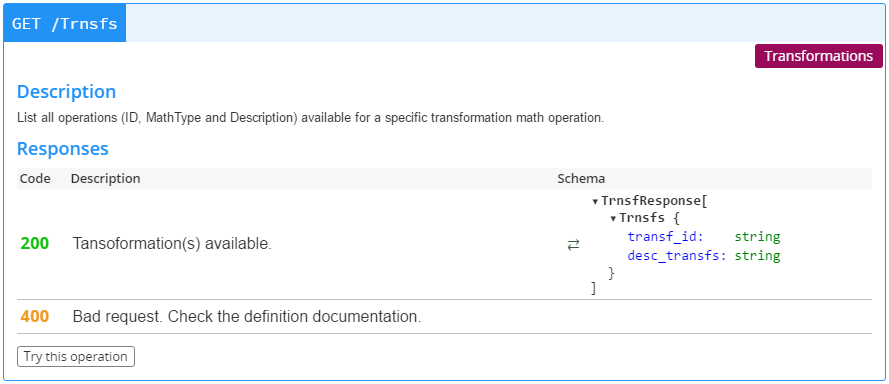
* **/Users/{UserID}/Macros/{MacroID}**
* **Método PUT**



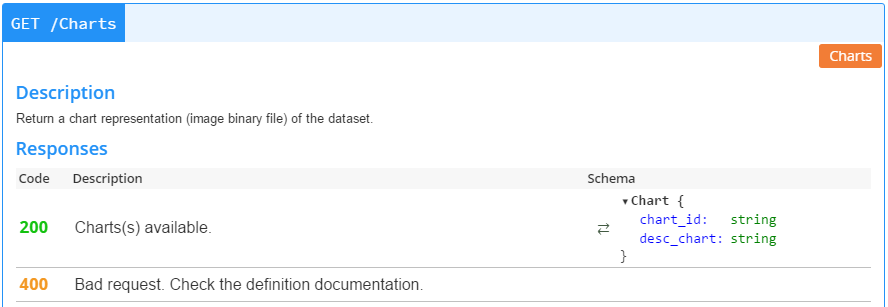
* **/Stats**
* **Método GET**



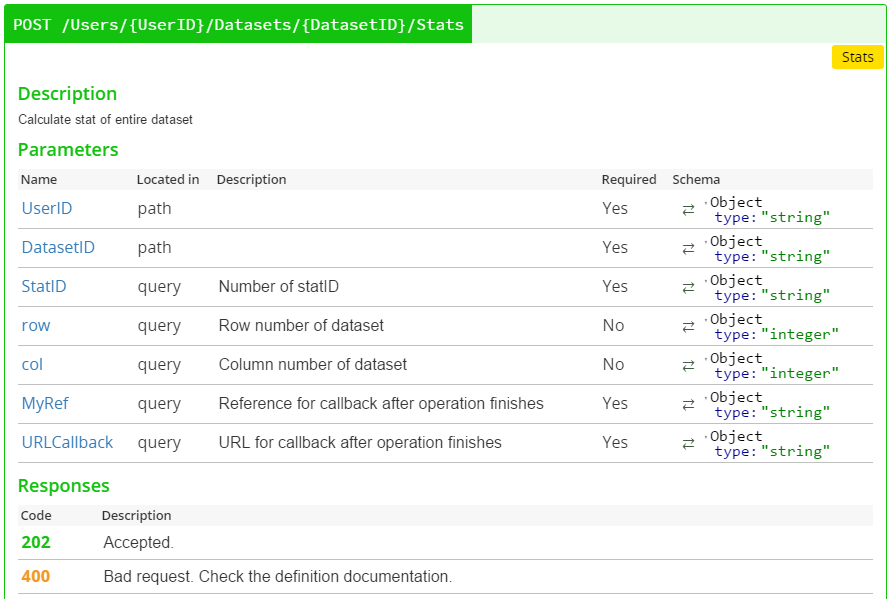
* **/trnsfs**
* **Método GET**



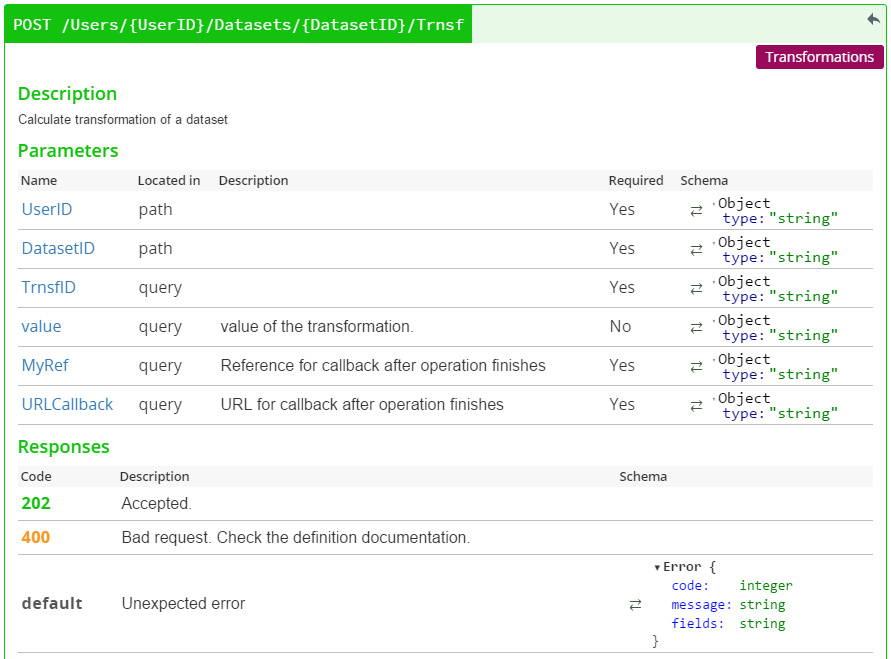
* **/Charts**
* **Método GET**



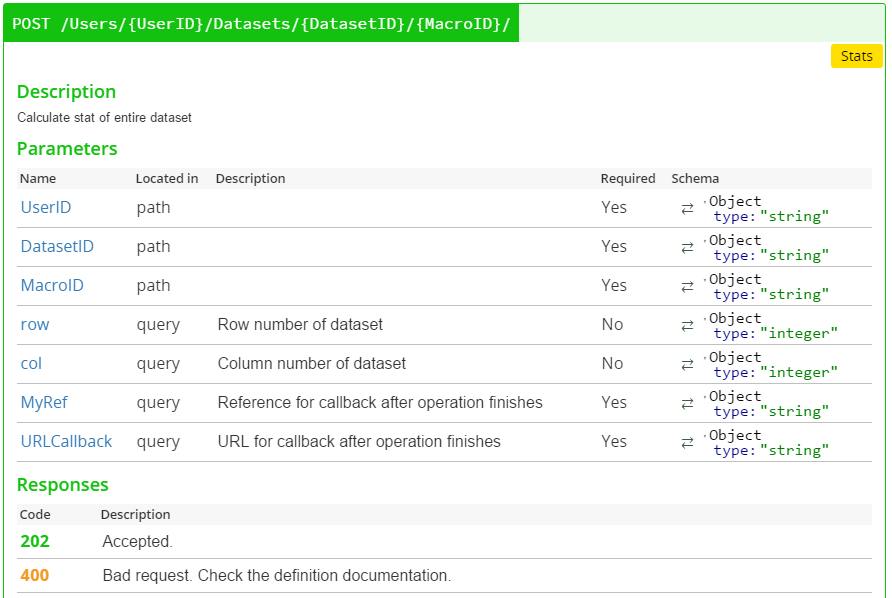
* **/Users/{UserID}/Datasets/{DatasetID}/Stats**
* **Método POST**



* **/Users/{UserID}/Datasets/{DatasetID}/Trnsf**
* **Método POST**



* **/Users/{UserID}/Datasets/{DatasetID}/{MacroID}/**
* **Método POST**



### REST

### Livraria de Código

### Fluxo de Trabalho

### Criar um dataset

### Criar um user

### Atualizando Dados de um Pedido

### Apagar um pedido

## Eventos

## Forma de elementos possíveis

### Lógica do Branching

## Callbacks

### Utilizar callbacks

### Endpoints

### Callback - Segurança do Endpoint

### Autorização de cabeçalho baseado em http

## Autorização

### Controlo de Acesso baseado em Tokens

### Erros de Handling

### Erros de Servidor

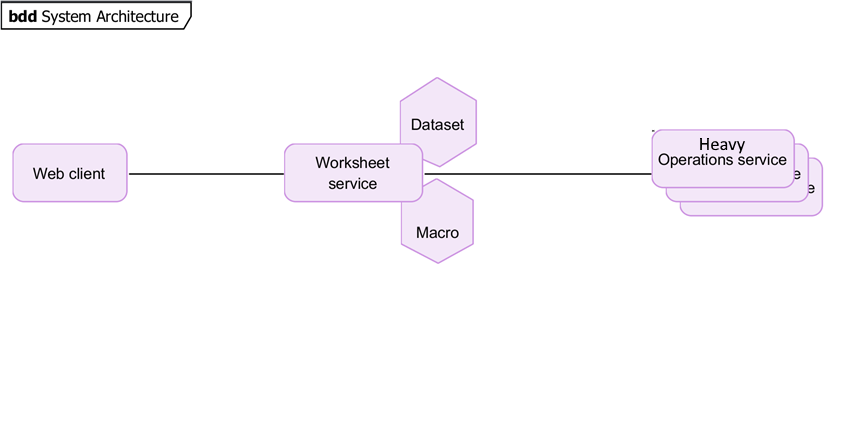
### Estado do Sistema

### Teste de integração

### Referências da API

## Anexos

### Diagrama de Arquitetura



## Figura 3- Diagrama de rquitetura da Aplicação

### Casos de Uso

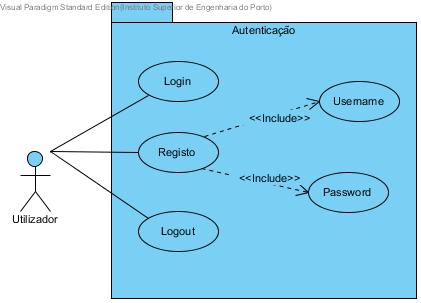
### Diagrama de Casos de Uso da Autenticação

### Autenticação

### Registo

### Registo de um utilizador

### 



### Figura 4- Caso de Uso – Autenticação

# Diagramas de sequência

### Diagrama de sequência – Inserir Dataset

### 

### Figura 5- Diagrama de sequência – Inserir Dataset

### Diagrama de sequência – Calcular Estatísticas

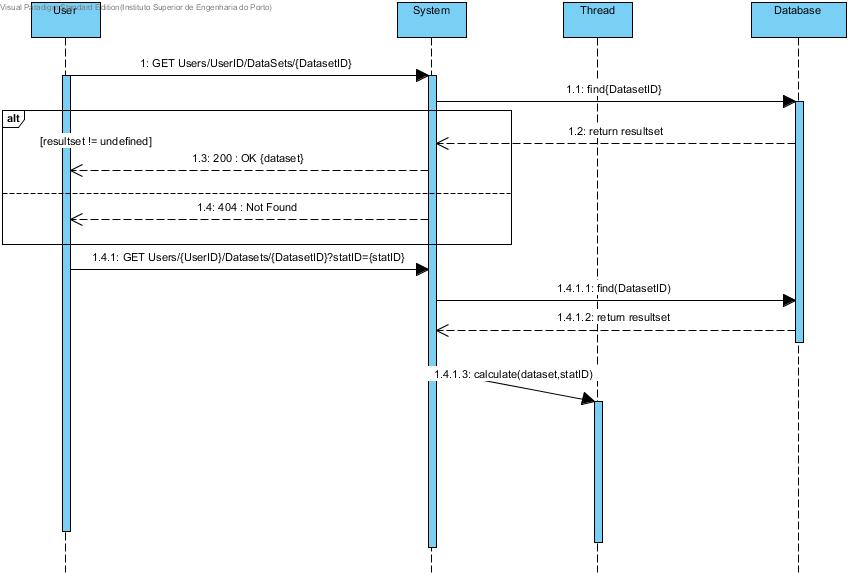
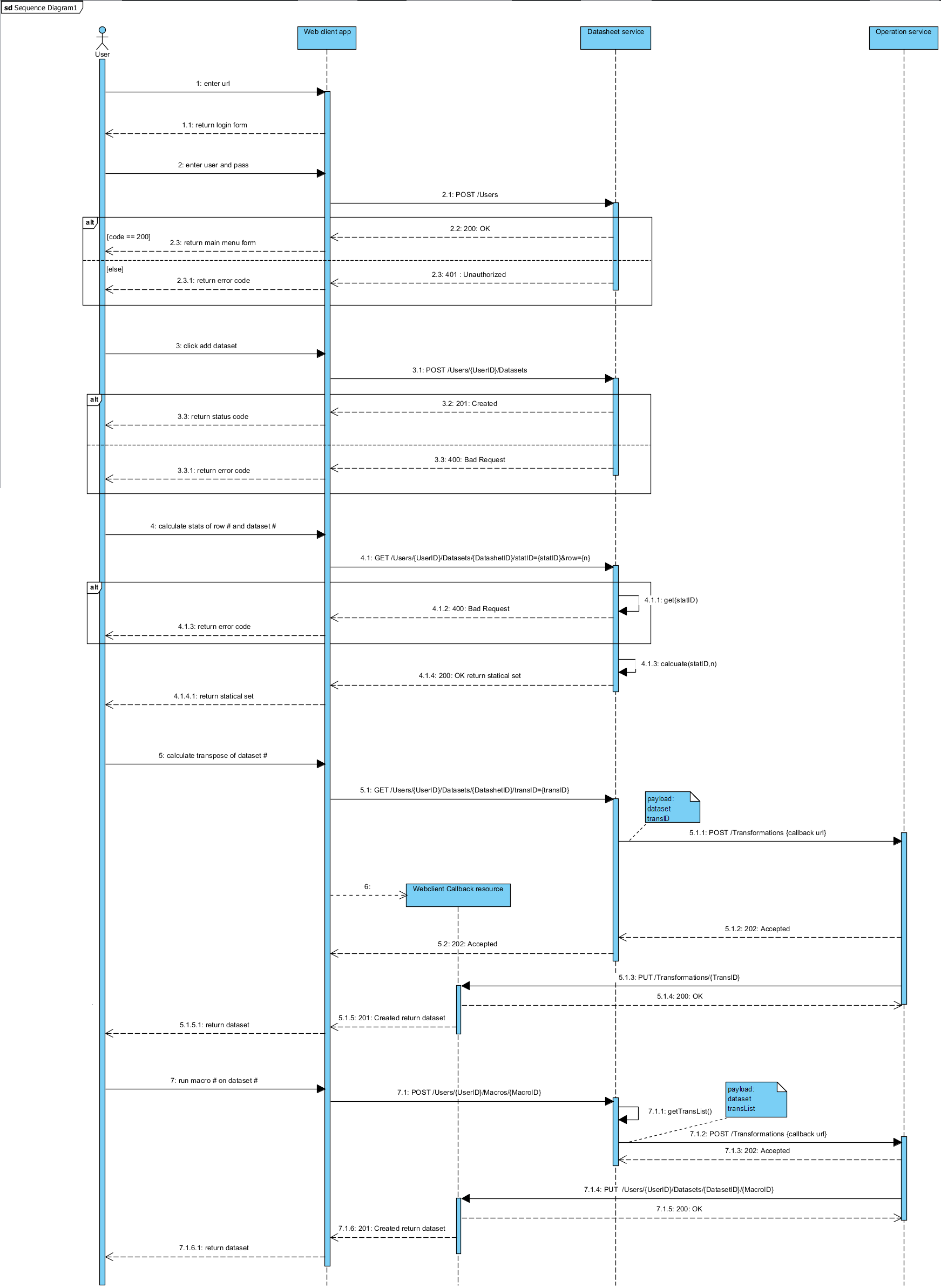


Figura 6- Diagrama de sequência – Calcular Estatísticas



## Figura 5- Diagrama de sequência do Trabalho

## Conclusão

Pensa-se ter-se atingido o principal objetivo do presente projeto que consistia em conseguir criar uma aplicação que conseguisse resolver problemas simples de análises estatísticas.

Apenas não se satisfizeram na totalidade os problemas das macros e dos gráficos devido a problemas na implementação e escassez de tempo.

Desenvolveu-se o protótipo do sistema e posteriormente criou-se uma especificação dos serviços do sistema descrito.

Desenvolveu-se o protótipo do sistema e posteriormente criou-se uma especificação dos serviços do sistema descrito.

Após uma má abordagem dos elementos do grupo ao terem optado por apresentar os resultados em HTML conseguiu-se refazer o código para que os dados aparecessem em Json de modo que após um esforço adicional do grupo de trabalho é da opinião ampla do grupo que finalmente conseguiu concretizar os objetivos a que se propês na sua maioria.

É da opinião amplamente geral do grupo de trabalho Restify que o trabalho foi concretizado com sucesso.

## Bibliografia

|  |  |
| --- | --- |
| [1] | “developers.covermymeds.com,” covermymeds, 2016. [Online]. Available: https://developers.covermymeds.com. [Acedido em 01 Janeiro 2016]. |
| [2] | Junos Space RESTful Developer Reference for Security Director, “Juniper Networks,” Junipe r Space , 2016. [Online]. Available: http://www.juniper.net/techpubs/en\_US/junos-space14.1/information-products/topic-collections/junos-space-security-designer/security-director-api.pdf. [Acedido em 2 Janeiro 2016]. |
| [3] | L. R. a. M. A. (O’Reilly), Restful Web APIs, S. Ruby, Ed., Tokyo: Leonard Richardson e Mike Amundsen (O’Reilly), 2013, pp. 1-37,37-49,55-94,171-202,333-352. |