Faculdade de Informática e Administração Paulista – FIAP

RODRIGO PEREIRA DE SOUZA **RA:38613**

GUILHERME RODRIGUES BARRETO DE ANDRADE **RA:38614**

**Twitter Small Analytics**

São Paulo

2017

**LISTA DE ABREVIATURAS E SIGLAS**

ALOI *Amsterdam Library of Object Images* (Base de imagens com distorções fotométricas)

C/C++ Linguagens de programação. C é uma linguagem estruturada enquanto C++ é orientada a objetos.

CIE CIELAB Sistema de cores perceptualmente uniforme desenvolvido pela CIE. Também conhecido como L\*a\*b\*.

dE Distância Euclidiana.

Dev-C++ Ambiente integrado de desenvolvimento de aplicações escritas em C/C++.

FPGA *Field Programmable Gate Array.*

JPEG *Joint Photographic Experts Group* (Formato de codificação de imagens).

Matlab *MATrix LABoratory* (software de alto desempenho para cálculo numérico e operações com matrizes).

OpenCV *Open Computer Vision* (Biblioteca com rotinas para processamento de imagens e visão computacional, desenvolvida pela Intel).

SURF *Speeded Up Robust Features* (Método para detecção de pontos chaves em imagens).

**ÍNDICE**

**LISTA DE ABREVIATURAS E SIGLAS**

[1 COMPONENTES, BIBLIOTECAS E FRAMEWORKS 7](#__RefHeading___Toc287867303)

[2 EXPLICAÇÃO DE USO DE PACOTES, CLASSES E MÉTODOS](#__RefHeading__144_1369635603) 10

3 [DIAGRAMA DE CLASSE](#__RefHeading__148_1369635603) 12

4 [DIAGRAMA DE SEQUÊNCIA](#__RefHeading__150_1369635603) 13

5 [CAPTURA DE TELAS](#__RefHeading__260_1369635603) 14

6 [GITHUB 1](#__RefHeading__154_1369635603)5

# 1 COMPONENTES, BIBLIOTECAS E FRAMEWORKS

A principal biblioteca utilizada foi **twitter4j** para consultar a api do twitter, é uma biblioteca não oficial entretanto opensource e cheia de recursos para facilitar a integração entre aplicação e api. Tem integração com o Maven ou também pode ser fácilmente encontrada para donwload no site oficial. Dentro dessa api foram útilizados diversos recursos como:

***ConfigurationBuilder***: Usada para construir uma configuração twitter4j com as configurações desejáveis;

***TwitterFactory***: Cria uma instáncia segura que pode ser re-utilizada smultaneamente;

***Query***: Utilizada para fazer pesquisa;

***QueryResult***: Utilizada para receber a resposta de uma pesquisa feita na api;

***Status***: Utilizada para representar o status do usuário;

***TwitterException***: utilizada para tratamento de erros da biblioteca;

Foi utilizada também outras bibliotecas nativas do Java IO, UTIL, TIME e SWING:

***InputStream***:

***IOException***:

***FileInputStream***:

***FileNotFoundException***:

***Properties***:

***List***:

***ArrayList:***

***Collections:***

***Comparator***:

***LocalDate***:

***JoptionPane:***

***JscrollPane:***

***Jtable:***

***DefaultTableModel:***

# 2 EXPLICAÇÃO DE USO DE PACOTES, CLASSES E MÉTODOS

A classe responsábel pela execuxão da aplicassão é a App que fica dentro do projeto entretanto fora dos pacotes citados que por sua vez estão separados devido a seus objetivos especificos. Essa classe tem apenas o método main que é um void e chama o método searchTweets que é o base de todo o projeto.

O sistema foi organizado em 4 pacotes:

**connection**: Pacote responsável por guardar as classes que fazem conexão. Dentro dele está localizada apenas uma classe Connection que é responsavél por primeiramente pegar a autentificação e depois fazer toda a configuração de conexão setando a autentificação necessária. Nessa classe há apenas um método:

* configureConf: Método chamado para fazer a conexão com o twiter, quem o chama espera receber um twitter autentificado e pronto para ser usado.

**core**: Onde fica toda regra de negócio da aplicalção. Nesse pacote existe apenas apenas a classe BaseTwitterSmallAnalytics. Os métodos dessa classe são dividos em dois:

* searchTweets: Este, ao receber a conexão do método configureConf utiliza de seus meios para fazer a pesquisa dos twittes, organizar o result na entidade e passa a entidade, o result e a conexção para que o método consiga finalizar a tarefa.
* PostTweet: Após receber entidade, conexão e result ele mostra o resultado em tabela e faz a postagem do resultado na timeline do autentificador.

**entity**: Pacote responsavel por guardar as entidades. Nele existe apenas a classe StatusJSONImpl que guarda os atributos nickname, nome, data, reTweets e favoritos e seus getters e setters. Essa classe é essencial para que se possa guardar cada usuário pesquisado no searchTweets e com a informação se possa fazer o levantamento necessário para a atividade.

**util**: Pacote extra que foi criado para ser um facilitador, onde fica algumas classes de helper, que nos ajudam a realizar alguma tarefa especifica. Nesse pacote existe apenas a classe Order que é responsável por ordenar a lista de nomes e de datas que é passada pra ela. Dentro dessas tarefas foram criadas os seguintes metodos:

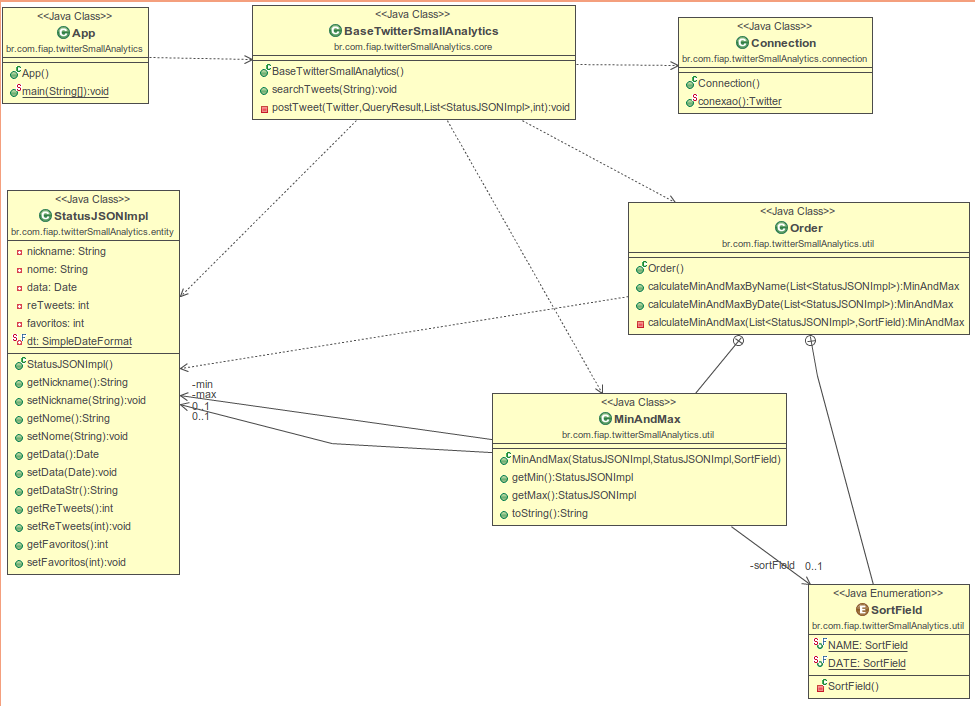
* calculateMinAndMaxByName:
* calculateMinAndMaxByDate:
* calculateMinAndMax:

Na classe Order existe ainda um SortField que é um ENUM utilizado para a tomada de decisão se o método calculateMinAndMax irá ordenar por NOME ou por DATA e ao retornar o resultado e da um NEW utilizando a class MinAndMax que por sua conta retorna apenas o MIN e o MAX de cada objeto, utilizando do enum SortField também para a tomada de decisão do que irá printar no método toString dela.

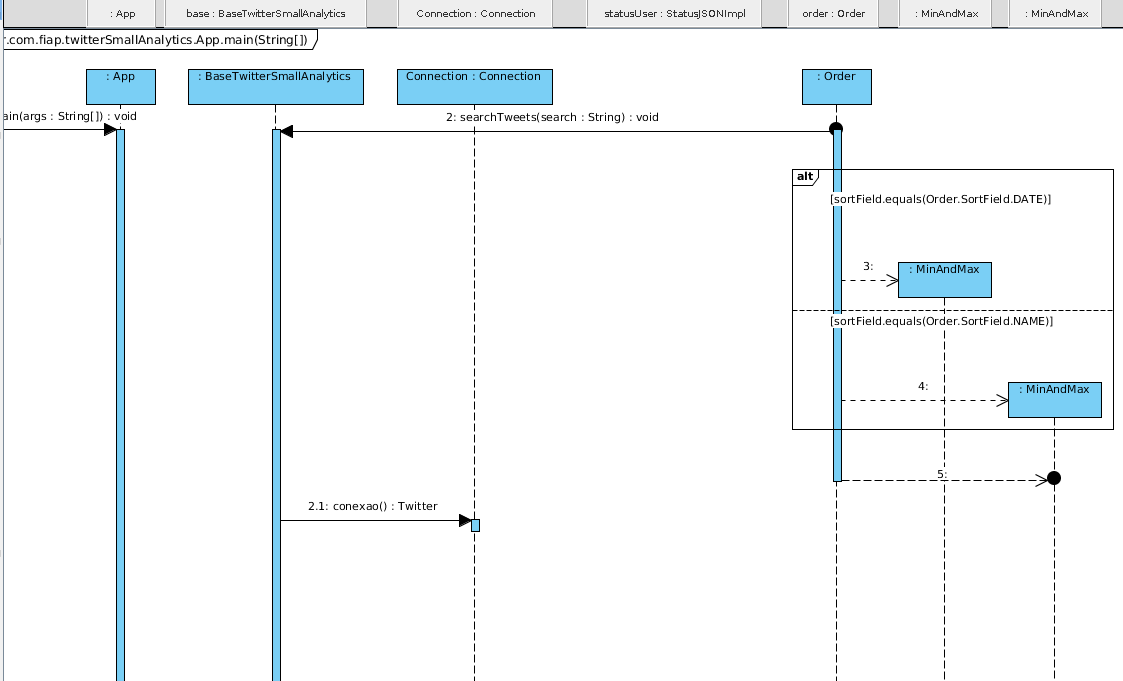
Também dentro do projeto tem uma pasta lib com as bibliotecas necessárias e outra pasta resource com um arquivo twitter4j.properties contendo a autentificação necessária para conseguir se comunicar com o Tweeter.

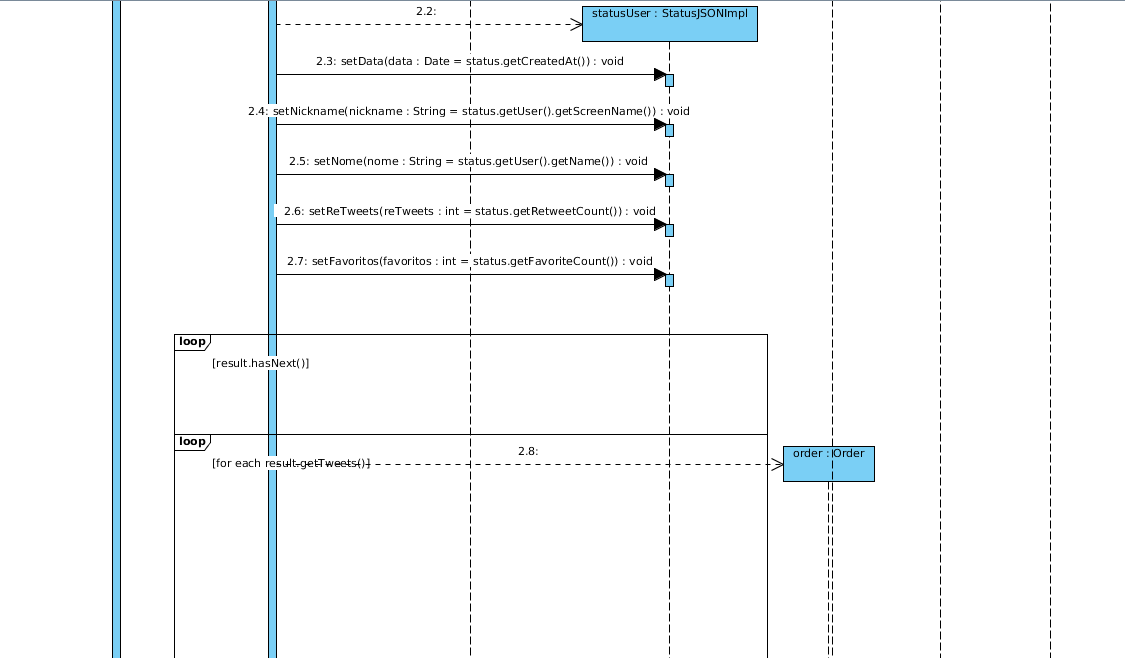
Para obter essa autentificação é necessário ter uma conta de desenvolvedor no Tweeter.

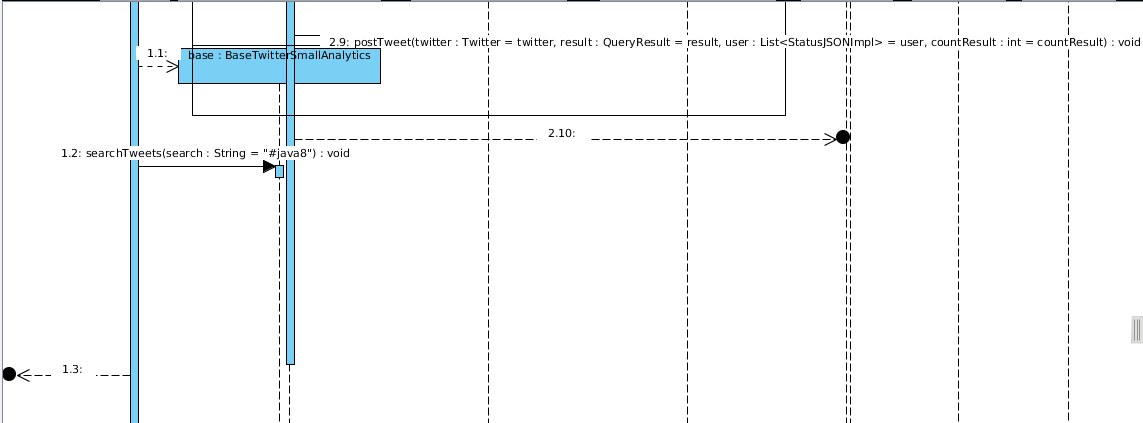
# 3 DIAGRAMA DE CLASSE



# 4 DIAGRAMA DE SEQUÊNCIA







# 5 CAPTURA DE TELAS

# 6 GITHUB

https://github.com/DiigoO/AtividadeFinal