Faculdade de Informática e Administração Paulista – FIAP

RODRIGO PEREIRA DE SOUZA **RA:38613**
GUILHERME RODRIGUES BARRETO DE ANDRADE **RA:38643**

TWITTER SMALL ANALYTICS

SÃO PAULO
2017

ÍNDICE

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:

IOException: Exceção que pode ser lançada quando ocorre algum erro ***FileInputStream e InputStream***: Utilizados para carregar os dados de configuração da Properties

FileNotFoundException: Exceção para o caso de o arquivo de Properties não existir

Properties: utilizado para armazenar os dados de acesso do Twitter

List e ArrayList: Utilizados para armazenamento e iterações em uma Lista de objetos.

Collections e Comparator: Utilizados para fazer a Ordenação dos objetos

LocalDate: Utilizada para armazenar e tratar datas

JOptionPane, JScrollPane, JTable e DefaultTableModel: Utilizado para interagir com o Usuário

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. No `main` optamos por deixar quem está executando o programa escolher sua hashtag utilizando uma tela de entrada.

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:

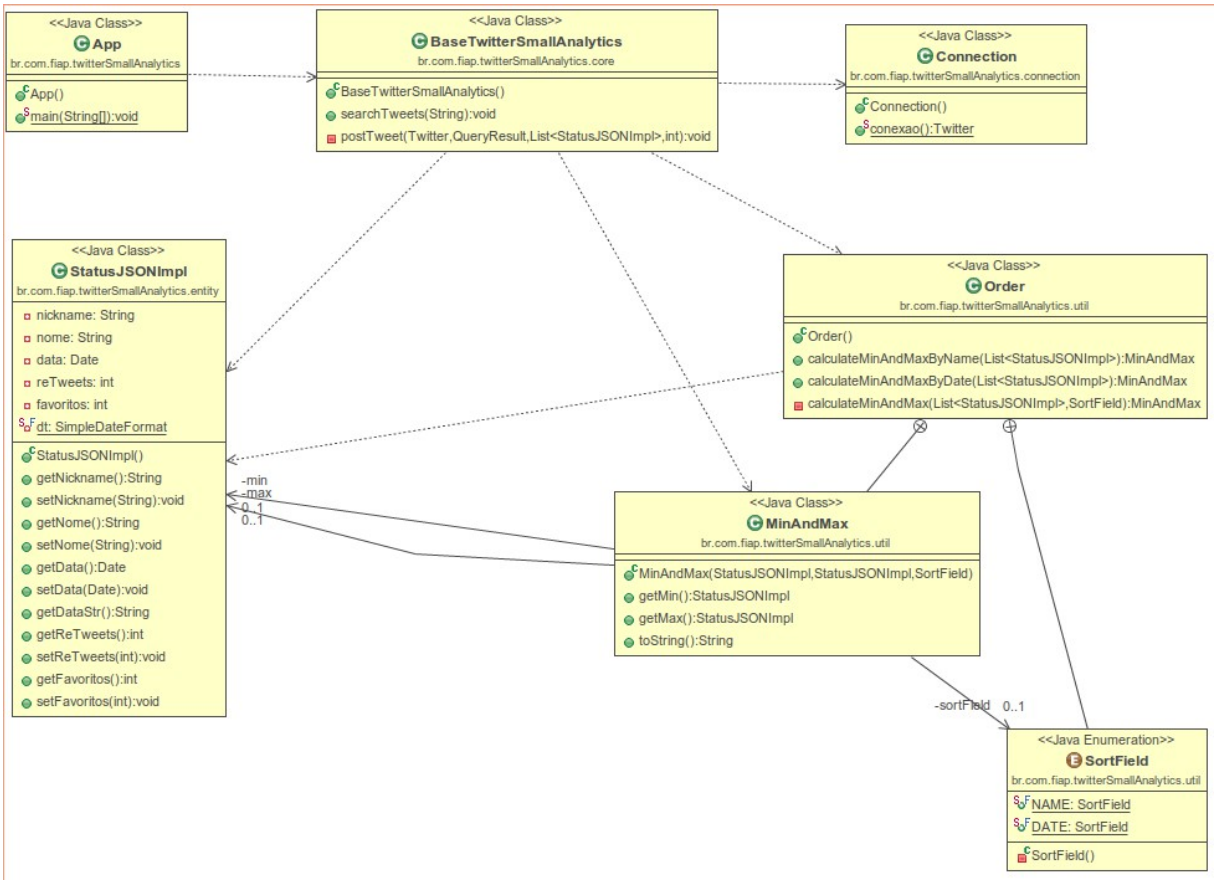
- `calculateMinAndMaxByName:`
- `calculateMinAndMaxByDate:`
- `calculateMinAndMax:`

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.

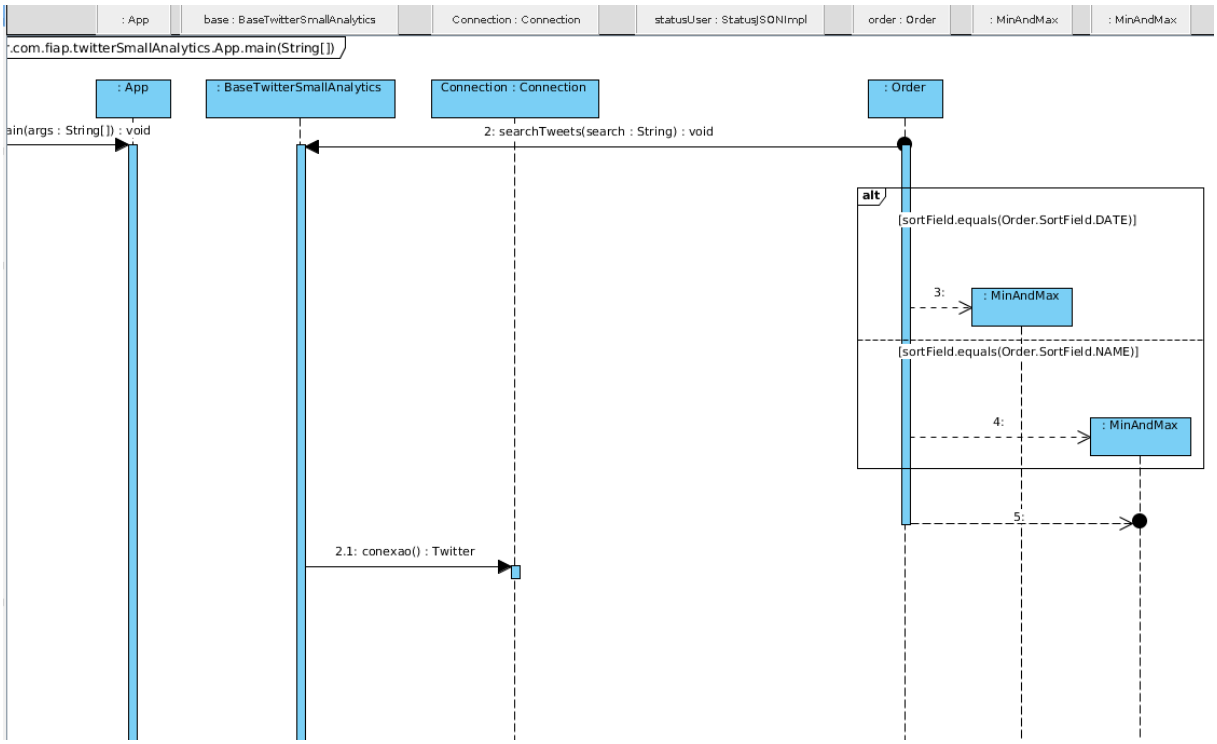
Também dentro do projeto tem uma pasta lib com as bibliotecas necessárias e outra pasta resource com um arquivo twitter4j.properties contendo a autentificação necessária para conseguir se comunicar com o Tweeter.

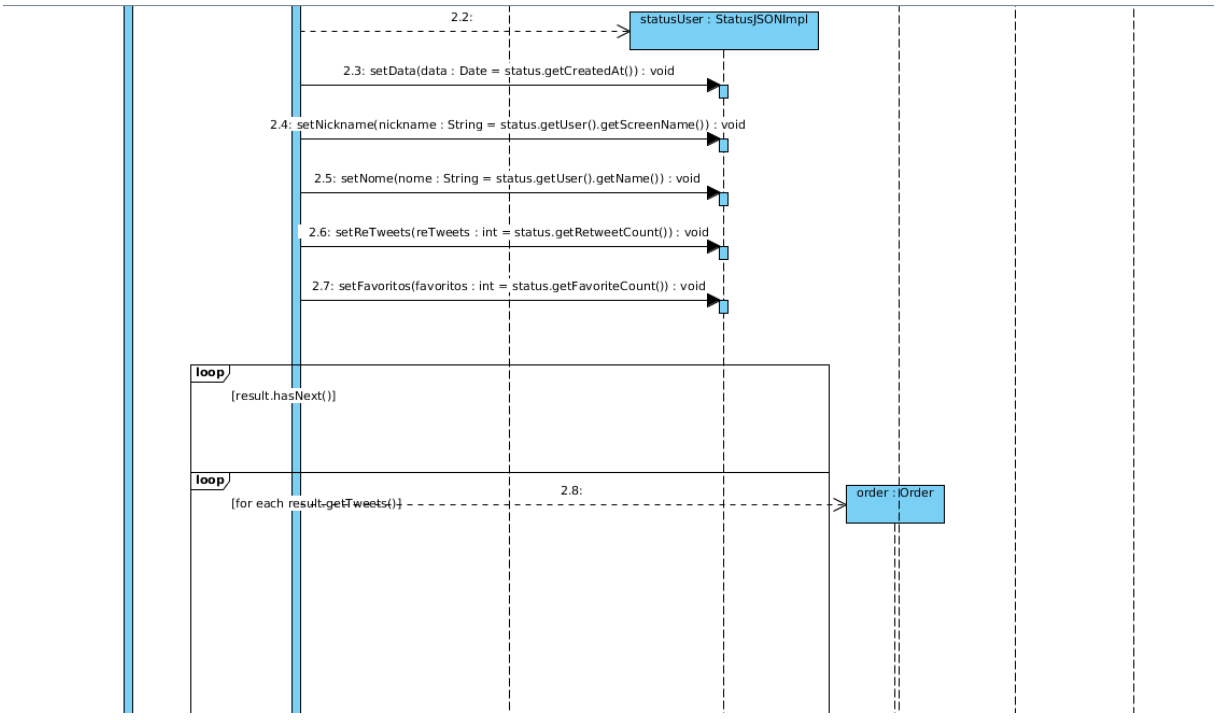
Para obter essa autentificação é necessário ter uma conta de desenvolvedor no Tweeter.

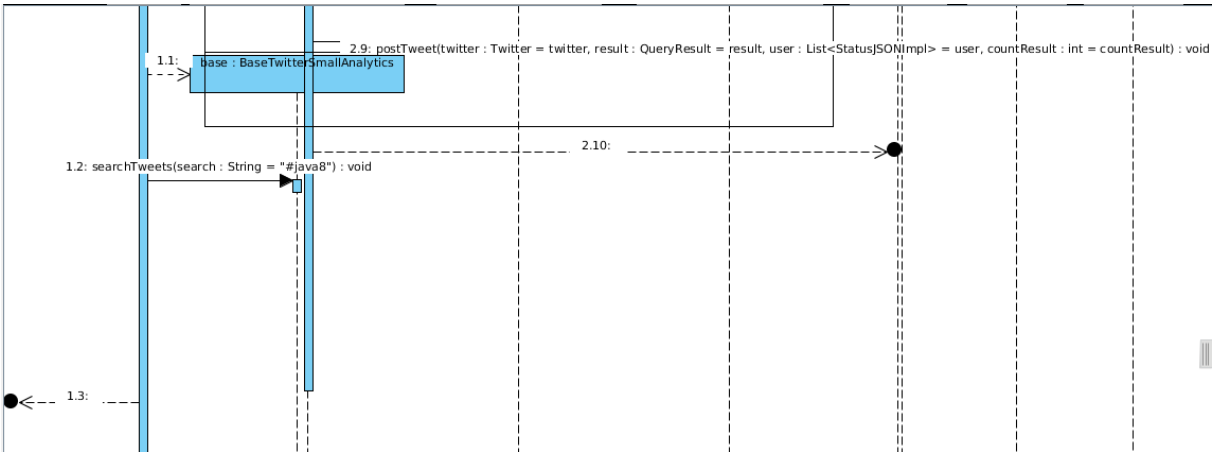
3 DIAGRAMA DE CLASSE



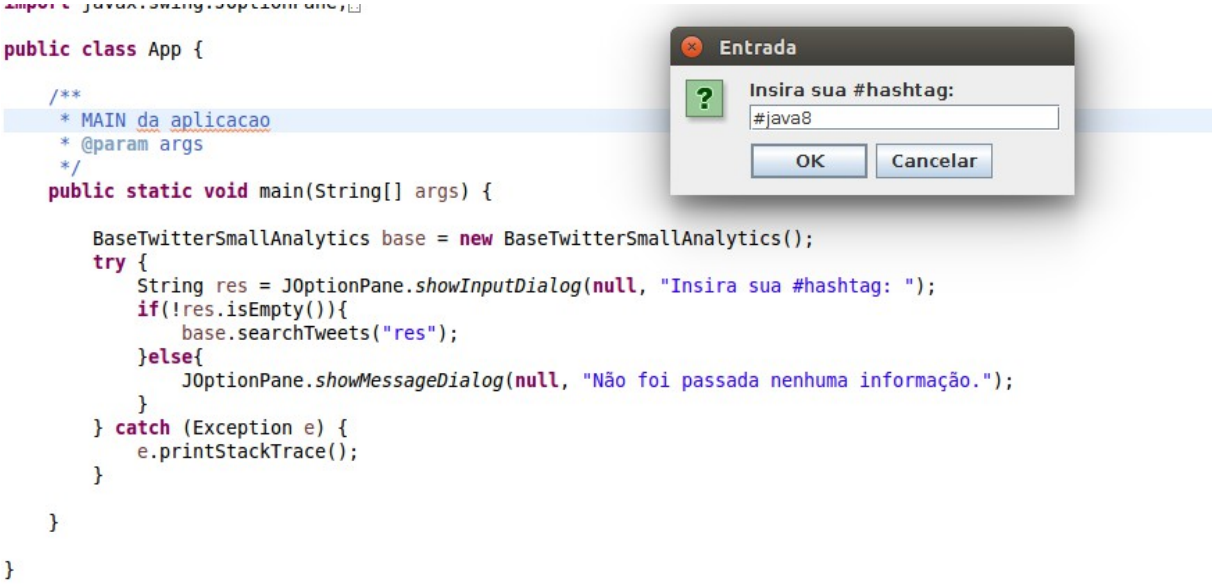
4 DIAGRAMA DE SEQUÊNCIA





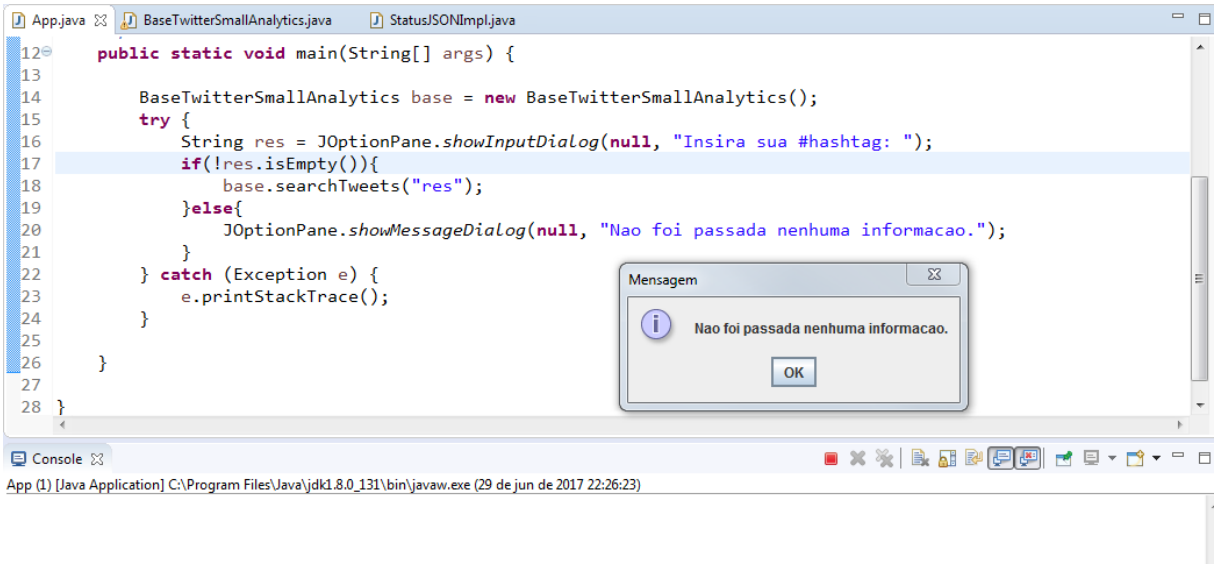

2.9: postTweet(twitter : Twitter = twitter, result : QueryResult = result, user : List<StatusJSONImpl> = user, countResult : int = countResult) : void
1.1:
base : BaseTwitterSmallAnalytics
2.10:
1.2: searchTweets(search : String = "#java8") : void
1.3:

5 CAPTURA DE TELAS

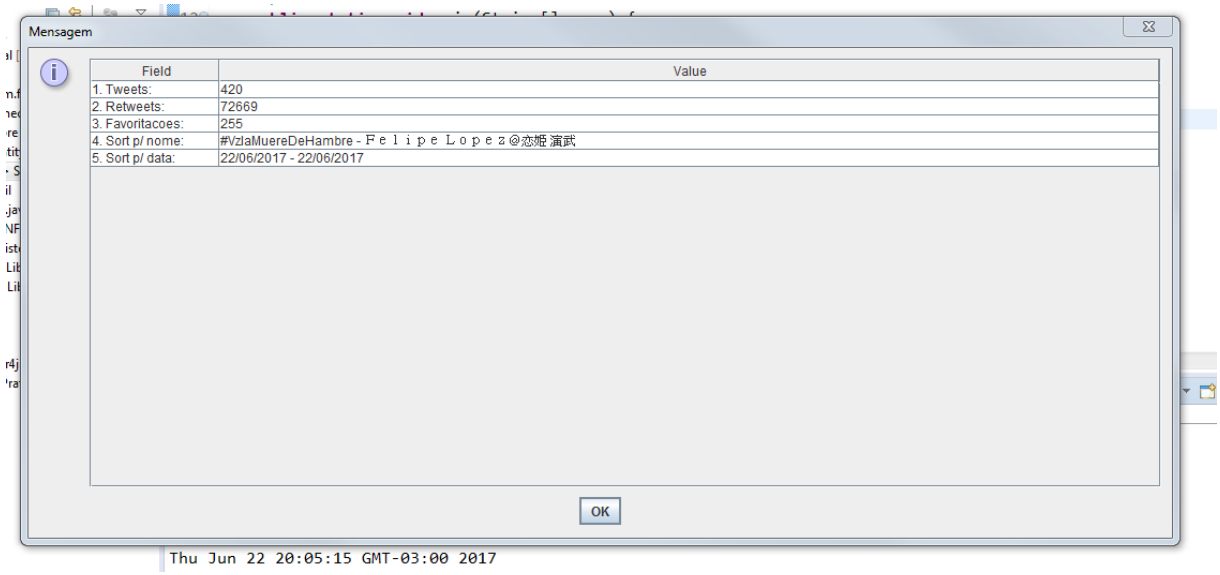



```
public class App {

    /**
     * MAIN da aplicacao
     * @param args
     */
    public static void main(String[] args) {

        BaseTwitterSmallAnalytics base = new BaseTwitterSmallAnalytics();
        try {
            String res = JOptionPane.showInputDialog(null, "Insira sua #hashtag: ");
            if(!res.isEmpty()){
                base.searchTweets("res");
            }else{
                JOptionPane.showMessageDialog(null, "Não foi passada nenhuma informação.");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

}
```

A primeira tela pede que seja inserida a hashtag para que o processo de procura comece.

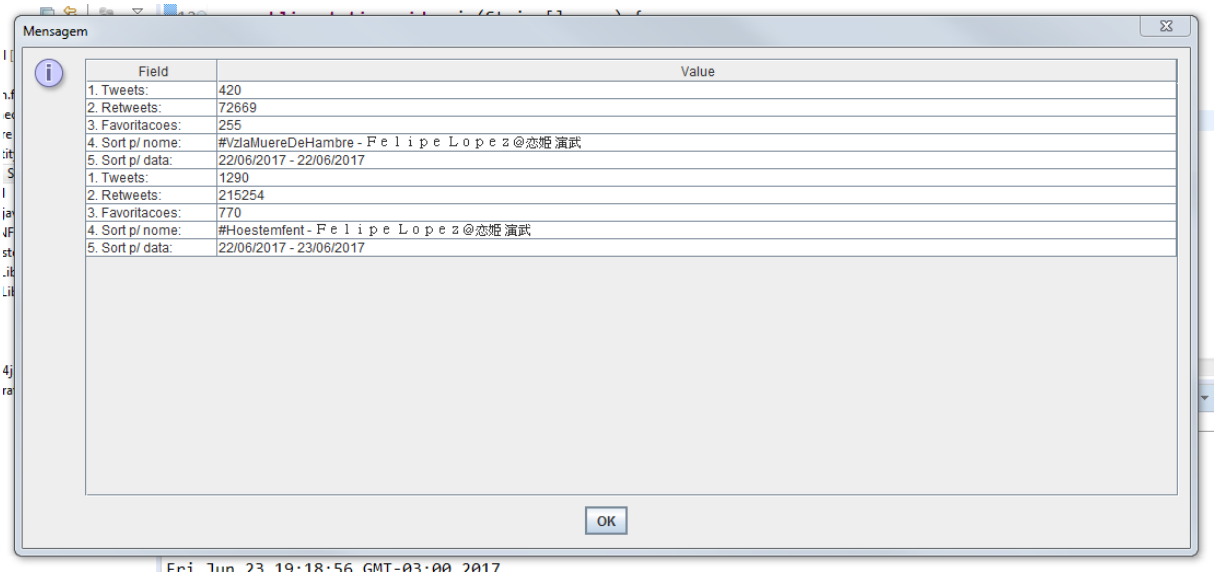


Caso não seja inserida nenhuma palavra um dialogo de menssagem é aberto dizendo que não é possível realizar a pesquisa.

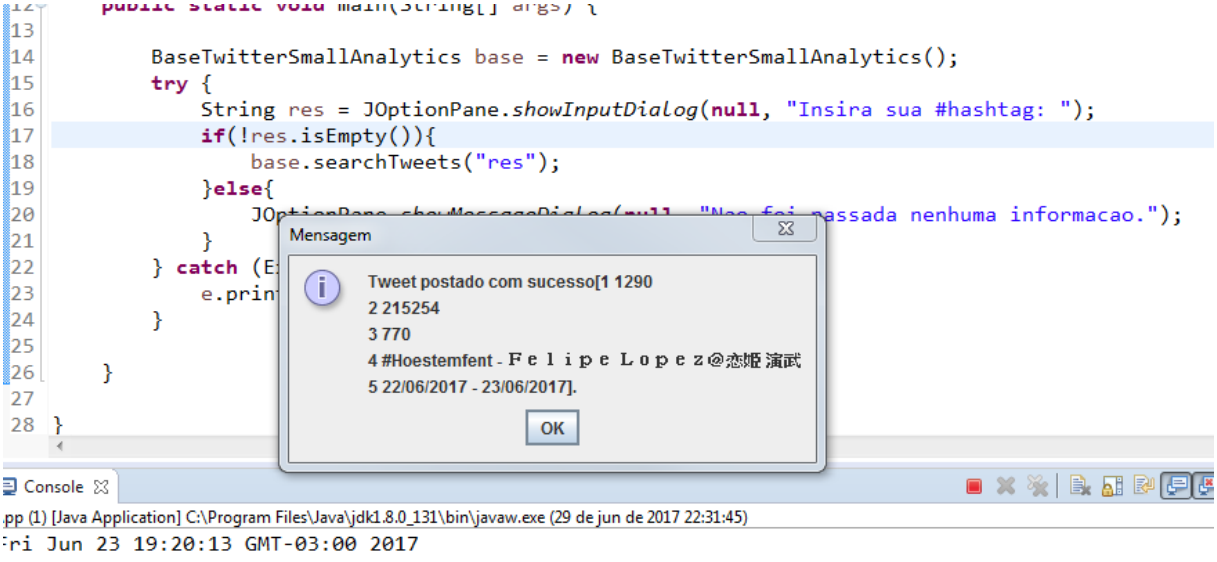


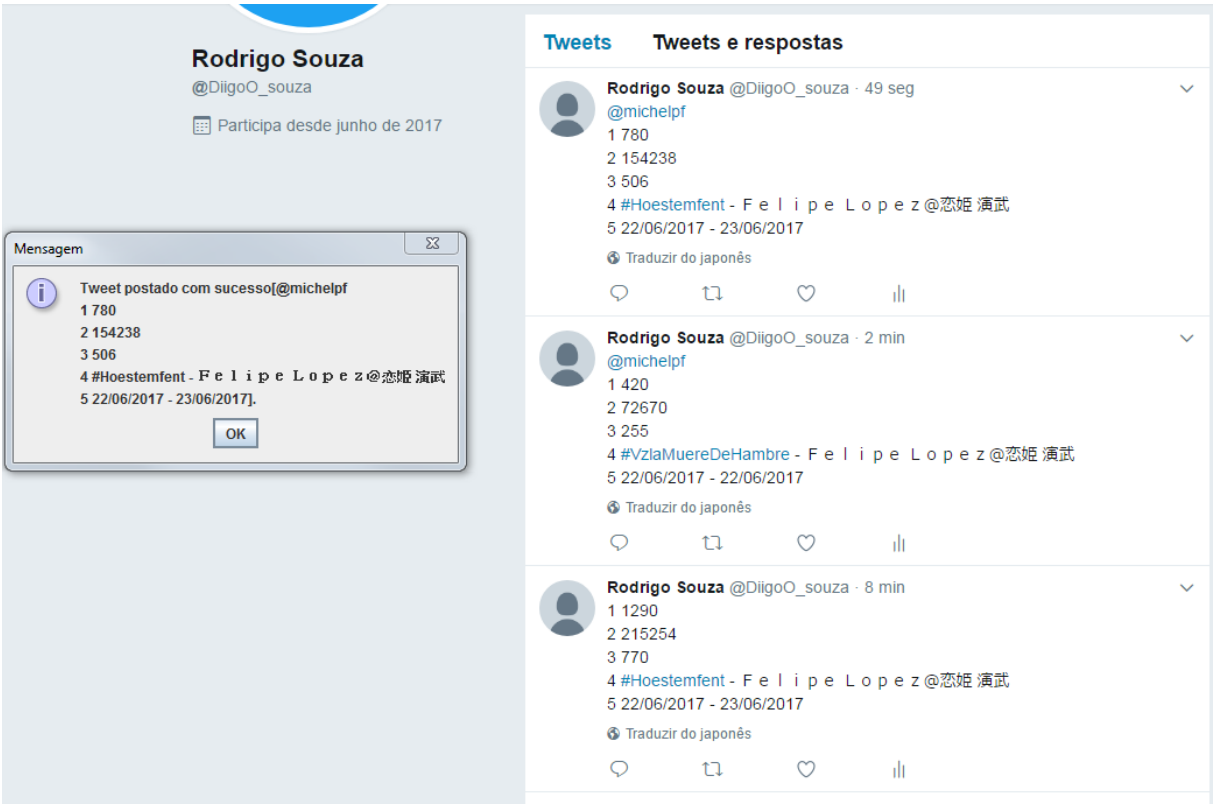
Quando a pesquisa do primeiro dia termina os resultados são mostrados em tela.

Optamos por deixar o úsuário escolher se vai querer postar a pesquisa. Caso a escolha seja não, o programa continua rodando todos os próximos dias acumulando na table todos os próximos resultados.



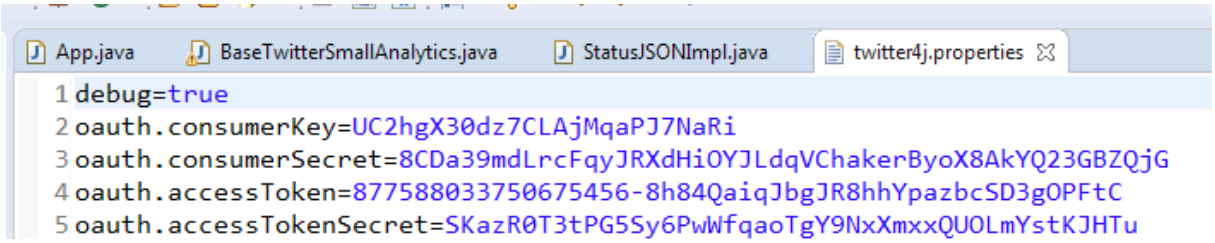
Caso a resposta seja sim, uma caixa de diálogo é aberta com os resultados e a menssagem de sucesso! Ao dar ok a aplicação retoma com os próximos dias.





E finalmente o resultado esperado é postado como na tela acima.

É necessário configurar o twitte4j.properties com a autentificação necessária que deve ser criada pelo Twitter.



6 GITHUB

https://github.com/DiigoO/AtividadeFinal