UNIVERSIDADE FEDERAL DO MARANHÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO (PPGCC)
PROGRAMA DE PÓS-GRADUAÇÃO DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO (DCCMAPI)
Disciplina: **Engenharia de *Software***
Aluno: **Wellington Luis Mineiro França**
Prof: **Dr. Davi Viana**
Data: **20/07/2020**

PROJETO DA DISCIPLINA ENGENHARIA DE SOFTWARE – DEFINIÇÃO DE REQUISITOS E ARQUITETURA – TÉCNICAS ANÁLISE DOCUMENTAL e PROTOTIPAGEM

1. INTRODUÇÃO

Elicitação de requisitos funcionais e não funcionais, além da definição e modelagem da arquitetura do *software* - produto do projeto de tese de doutorado – capaz de recuperar dados de geolocalização (latitude, longitude) de usuários da rede social *Twitter* no município de São Luís/MA, a partir de um termo encontrado em seus *twits*, como atividade da disciplina Engenharia de *Software*.

2. REGISTRO DE DOCUMENTOS

2.1 Documentos analisados

- Documentação *online* da *Twitter* API – Interface de programação que permite amplo acesso aos dados públicos do *Twitter* que os próprios usuários escolheram compartilhar com o mundo [1].

3. ANÁLISE DOS DOCUMENTOS

3.1 Subsistema: Conta de usuário no *Twitter*:

- Verificar se o usuário que deseja obter os dados na rede social já possui uma conta cadastrada no *Twitter*;
  Se o usuário já possuir conta cadastrada:
  - Efetuar *logon* na conta;

  Senão:
  - Cadastrar uma conta e efetuar *logon*;

3.2 Subsistema: Obter dados de acesso aos *twits*

- Registrar o aplicativo no *Twitter* em *https://developer.twitter.com/apps*;
- Obter chaves de acesso:
  - Obter a chave CONSUMER KEY de um servidor do *Twitter* através da página *https://dev.twitter.com/*;
  - Obter a chave CONSUMER SECRET de um servidor do *Twitter* através da página *https://dev.twitter.com/;*

- Obter *tokens* provisórios de usuário:
  - Obter o *token* provisório ACCESS TOKEN de um servidor do *Twitter* através da página *https://dev.twitter.com/* ou via implementação de código no aplicativo usando o protocolo de autenticação OAUTH1 ou OAUTH2;

  - Obter o *token* provisório ACCESS TOKEN SECRET de um servidor do *Twitter* através da página *https://dev.twitter.com/* ou via implementação de código no aplicativo usando o protocolo de autenticação OAUTH1 ou OAUTH2;

- Solicitar *tokens* de usuário final aos servidores *Twitter* passando CONSUMER KEY, CONSUMER SECRET, ACCESS TOKEN(provisório) e ACCESS TOKEN SECRET(provisório) como parâmetros;

- Obter *tokens* de usuário final ACCESS TOKEN e ACCESS TOKEN SECRET a partir da URL retornada pelos servidores do *Twitter*;

3.3 Subsistema: Extrair dados brutos do *Twitter*

- O sistema deve solicitar autenticação aos servidores *Twitter* passando as chaves e *tokens* de usuário final como parâmetros;
- Se autenticação for validada:
  - Efetuar consulta ao *Twitter* com base em uma ou mais palavras-chave como filtro, obtendo como resposta as coordenadas de localização geográfica (latitude, longitude), a data e hora da postagem;
- Cadastrar os dados resultantes da consulta: DATA, HORA, LATITUDE, LONGITUDE, dos últimos 03 meses, em um arquivo texto.

3.4 Subsistema: Processar dados brutos extraídos do *Twitter*

- Usando um modelo matemático, o sistema deve agrupar os dados em aglomerados com base nas coordenadas obtidas para cada hora do dia;
- O sistema deve rotular cada aglomerado: A1, A2,...,An
- Com base nos aglomerados, o sistema deve calcular trajetórias, partindo de um ponto P1 de um aglomerado A1 até um ponto P2 de um aglomerado A2, segundo um grafo orientado em que o peso de cada aresta será a densidade populacional de cada aglomerado (quantidade de pessoas) adjacente ao ponto P analisado, e o caminho será determinado pelo aglomerado com maior densidade populacional;
- O sistema deve solicitar um ponto P1 de origem para cálculo das trajetórias;
- O sistema deve solicitar um ponto P2 de destino para cálculo das trajetórias;
- O sistema deve calcular trajetórias de P1 a P2 para cada hora do dia nos últimos 03 meses;
- O sistema deve sugerir uma trajetória para cada hora do dia, considerando os aglomerados de maior freqüência adjacentes a um ponto P em todas as trajetórias calculadas para o horário específico.

3.5 Subsistema: Sugerir trajetória

- Para cada hora do dia:
  - Exibir a trajetória sugerida em um mapa da cidade de São Luís, representando-a visualmente como uma linha contínua sobre aglomerados específicos;

4. REQUISITOS

4.1 Não Funcionais

1. **NF01: Verificar conta cadastrada no *Twitter***
   Se o usuário já possuir conta cadastrada:
   - Efetuar *logon* na conta;

   Se o usuário não possuir conta:
   - Cadastrar uma conta e efetuar *logon*;

2. **NF02: Registrar aplicativo na página de desenvolvedor do *Twitter***
   Confirmar registro do aplicativo

3. **NF03: Obter chaves de acesso na página de desenvolvedor do *Twitter*:**
   - Obter a chave CONSUMER KEY de um servidor do *Twitter* através da página *https://dev.twitter.com/*;
   - Obter a chave CONSUMER SECRET de um servidor do *Twitter* através da página *https://dev.twitter.com/;*

4. **NF04: O aplicativo deve ser executado em um ecossistema de *Big Data*, a fim de processar um grande volume de dados extraídos de diversas redes sociais.**

4.2 Funcionais

5. **RF01: Obter *tokens* provisórios de usuário:**
   - Obter o *token* provisório ACCESS TOKEN de um servidor do *Twitter* através do protocolo de autenticação OAUTH1 ou OAUTH2;

   - Obter o *token* provisório ACCESS TOKEN SECRET de um servidor do *Twitter* através do protocolo de autenticação OAUTH1 ou OAUTH2;

6. **RF02: Solicitar *tokens* de usuário final aos servidores *Twitter* passando CONSUMER KEY, CONSUMER SECRET, ACCESS TOKEN(provisório) e ACCESS TOKEN SECRET(provisório) como parâmetros;**

7. **RF03: Obter *tokens* de usuário final ACCESS TOKEN e ACCESS TOKEN SECRET a partir da URL retornada pelos servidores do *Twitter*;**

8. **RF04: O sistema deve solicitar autenticação aos servidores *Twitter* passando as chaves e *tokens* de usuário final como parâmetros;**
   Se autenticação for validada:
   - Efetuar consulta ao *Twitter* com base em uma ou mais palavras-chave como filtro, obtendo como resposta as coordenadas de localização geográfica (latitude, longitude), a data e hora da postagem;

9. **RF05: Cadastrar os dados resultantes da consulta: DATA, HORA, LATITUDE, LONGITUDE, dos últimos 03 meses, em um arquivo texto.**

10. **RF06: Com base em um modelo matemático, o sistema deve agrupar os dados em aglomerados com base nas coordenadas obtidas para cada hora do dia;**

11. **RF07: O sistema deve rotular cada aglomerado: A_1, A_2,...,A_n;**

12. **RF08: Com base nos aglomerados, o sistema deve calcular trajetórias, partindo de um ponto P_1 de um aglomerado A_1 até um ponto P_2 de um aglomerado A_2, segundo um grafo orientado em que o peso de cada aresta será a densidade populacional de cada aglomerado (quantidade de pessoas) adjacente ao ponto P analisado, e o caminho será determinado pelo aglomerado com maior densidade populacional;**

13. **RF09: O sistema deve solicitar um ponto P_1 de origem para cálculo das trajetórias;**

14. **RF10: O sistema deve solicitar um ponto P_2 de destino para cálculo das trajetórias;**

15. **RF11: O sistema deve calcular trajetórias de P_1 a P_2 para cada hora do dia nos últimos 03 meses;**

16. **RF12: O sistema deve sugerir uma trajetória para cada hora do dia, considerando os aglomerados de maior freqüência adjacentes a um ponto P em todas as trajetórias calculadas para o horário específico.**

17. **RF13: Para cada hora do dia:**
    - Exibir a trajetória sugerida em um mapa da cidade de São Luís, representando-a visualmente como uma linha contínua sobre aglomerados específicos;

5. SIGLAS E ACRÔNIMOS

Referências

[1] Sobre as APIs do Twitter. Disponível em <*https://help.twitter.com/pt/rules-and-policies/twitter-api*>. Acesso em 19 jul 2020.

6. ARQUITETURA A SER UTILIZADA

- **MVC**

7. MODELAGEM – DIAGRAMA DE CLASSES

Modell

Coordenada
- data: date - hora: time - latitude: float - longitude: float
+cadastrar(data, hora, latitude, longitude): void

Twitter
- key, key_secret, token, token_secret: string - chave_consulta: string
+consultar_key(key, key_secret): string +consultar_token(key, key_secret, token, token_secret): string +filtrar(chave_consulta): json

Aglomerado
- aglomerado: array
+agrupar(latitude, longitude): void +rotular(aglomerado): void +calcular_trajetorias(p1, p2): void +trajetoria_otima(): void

Controller

Calcular trajetórias
+consultar_token( ): void +cadastrar ( ): void +agrupar ( ): void +rotular(): void +carregartelaSolicitarpontos( ): void +Solicitarpontos( ): void +calcular_trajetorias( ): void

Sugerir trajetória
+ trajetoria_ otima( ): void +carregartelaTrajetoriasugerida( ): void

View

Solicitar pontos origem-destino
- p1: coordinate - p2: coordinate - calcular: Button

Apresentar trajetória sugerida
- mapa: map - linha: line