TwitterClientPrac7.pdf

GeXx_

Redes y Sistemas Distribuidos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

```java
package es.uma.rysd.app;

import javax.net.ssl.HttpsURLConnection;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.Base64;

import com.google.gson.Gson;

import es.uma.rysd.entities.TokenResponse;
import es.uma.rysd.entities.FollowerResponse;
import es.uma.rysd.entities.SearchResponse;
import es.uma.rysd.entities.Tweet;
import es.uma.rysd.entities.User;

public class TwitterClient {
   private String bearer_token = null;

   // TODO: Completar los atributos con sus datos
   private final String consumer_key = "18mBRe5AJsQ8zbwBAeTAPP3yR";
   private final String consumer_secret = "UojpjbdDQOEgbkuOUxB2aSdAG13QftlCf5S9DmpvxPDtrgUbsa";
   private final String app_name = "RSDAppDelAlumnoEugenio";

   private final String url_auth = "https://api.twitter.com/oauth2/token";
   private final String url_search = "https://api.twitter.com/1.1/search/tweets.json?result_type=popular&q=";
   private final String url_user = "https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=";
   private final String url_followers = "https://api.twitter.com/1.1/followers/list.json?screen_name=";
   private final String url_friends = "https://api.twitter.com/1.1/friends/list.json?screen_name=";

   private String getKey(){
      String key = consumer_key + ":" + consumer_secret;
      return new String(Base64.getEncoder().encode(key.getBytes()));
   }

   public TwitterClient() throws IOException{
      super();
      // TODO: Crear una conexión a la url indicada en url_auth
      URL servicio = new URL(url_auth);
      HttpsURLConnection connection = (HttpsURLConnection) servicio.openConnection();

      // TODO: Añadir las cabeceras User-Agent, Content-Type, Authorization y Accept:
      //   En User-Agent indique el nombre de la aplicación app_name
      //   En Content-Type: "application/x-www-form-urlencoded;charset=UTF-8"
      //   En Authorization: Basic seguido de las claves codificadas (el método getKey se encarga de esto).
      //    En Accept: application/json

      connection.setRequestProperty("User-Agent", app_name);
      connection.setRequestProperty("Content-type", "application/x-www-form-urlencoded;charset=UTF-8");
      connection.setRequestProperty ("Authorization", "Basic "+getKey());
      connection.setRequestProperty ("Accept", "application/json");

      // TODO: Indicar el método POST y que el mensaje SÍ lleva datos
      connection.setRequestMethod("POST");
      connection.setDoOutput(true);

      // TODO: Obtener el flujo de salida (OutputStream) y escriba (write) el mensaje adecuado ("grant_type=client_credentials").
      //    Debe cerrar el flujo tras escribir
      OutputStream os = (OutputStream) connection.getOutputStream();
      String mensaje = "grant_type=client_credentials";
```

```
        os.write(mensaje.getBytes());
        os.close();

        // TODO: Obtener el código de respuesta y comprobar que es correcto
        if(connection.getResponseCode() > 299 || connection.getResponseCode() < 200) {
         System.out.println("Código de respuesta es incorrecto("+connection.getResponseCode()+")");
        }
        // TODO: Obtener el flujo de entrada (InputStream) y deserializar su contenido en un objeto de tipo BearerToken usando
Gson
        Gson parser = new Gson();
        InputStream in = (InputStream) connection.getInputStream(); // TODO: indicar el de la conexion
        TokenResponse b = parser.fromJson(new InputStreamReader(in), TokenResponse.class);

        // TODO: Almacenar el token en el atributo bearer_token
        bearer_token= b.access_token;
    }

    public boolean hasToken(){
        return this.bearer_token != null;
    }

    public Tweet[] search(String text) throws IOException{
        // TODO: Crear una conexión a la url indicada en  url_search junto al text recibido y codificado de forma adecuado
       URL servicio = new URL(url_search+URLEncoder.encode(text, "utf-8"));
        HttpsURLConnection connection = (HttpsURLConnection) servicio.openConnection();

        // TODO: Indicar el método GET y que el mensaje NO lleva datos
        connection.setRequestMethod("GET");
        connection.setDoOutput(false);

        // TODO: Añadir las cabeceras User-Agent con el nombre de su aplicación,
//    Accept con el valor "application/json" y
     //   Authorization con "Bearer " seguida del bearer token
        connection.setRequestProperty("User-Agent", app_name);
        connection.setRequestProperty ("Accept", "application/json");
        connection.setRequestProperty ("Authorization", "Bearer "+bearer_token);

        // TODO: Obtener el código de respuesta y comprobar que es correcto
        if(connection.getResponseCode() > 299 || connection.getResponseCode() < 200) {
         System.out.println("Código de respuesta es incorrecto("+connection.getResponseCode()+")");
        }
        // TODO: Obtener el flujo de entrada (InputStream) y deserializar su contenido en un objeto de tipo SearchResponse
usando Gson
        Gson parser = new Gson();
        InputStream in = (InputStream) connection.getInputStream(); // TODO: indicar el de la conexión
        SearchResponse b = parser.fromJson(new InputStreamReader(in), SearchResponse.class);

        // TODO: Devuelva el atributo statuses del objeto deserializado
        Tweet[] t = b.statuses;
        return t;
    }

    public Tweet[] getTweetsUser(String username) throws IOException{
        Tweet[] t = null;
        // TODO: Crear una conexión a la url indicada en url_user junto al username recibido y codificado de forma adecuado
        URL servicio = new URL(url_user+URLEncoder.encode(username, "utf-8"));
        HttpsURLConnection connection = (HttpsURLConnection) servicio.openConnection();

        // TODO: Indicar el método GET y que el mensaje NO lleva datos
        connection.setRequestMethod("GET");
        connection.setDoOutput(false);

        // TODO: Añadir las cabeceras User-Agent con el nombre de su aplicación,
//    Accept con el valor "application/json" y
     //   Authorization con "Bearer " seguida del bearer token
        connection.setRequestProperty("User-Agent", app_name);
```

```
        connection.setRequestProperty ("Accept", "application/json");
        connection.setRequestProperty ("Authorization", "Bearer "+bearer_token);

        // TODO: Obtener el código de respuesta y comprobar que es correcto
        if(connection.getResponseCode() > 299 || connection.getResponseCode() < 200) {
          System.out.println("Código de respuesta es incorrecto("+connection.getResponseCode()+")");
        }

        // TODO: Obtener el flujo de entrada (InputStream) y deserializar su contenido en un objeto de tipo Tweet[] usando Gson
        Gson parser = new Gson();
        InputStream in = (InputStream) connection.getInputStream();
        t = parser.fromJson(new InputStreamReader(in), Tweet[].class);

        // TODO: Devuelva el objeto deserializado
        return t;
    }

    public User[] getFollowers(String username) throws IOException{
        User[] u = null;
        // TODO: Crear una conexión a la url indicada en url_followers junto al username recibido y codificado de forma adecuado
        URL servicio = new URL(url_followers+URLEncoder.encode(username, "utf-8"));
        HttpsURLConnection connection = (HttpsURLConnection) servicio.openConnection();

        // TODO: Indicar el método GET y que el mensaje NO lleva datos
        connection.setRequestMethod("GET");
        connection.setDoOutput(false);

        // TODO: Añadir las cabeceras User-Agent con el nombre de su aplicación,
    //    Accept con el valor "application/json" y
      //   Authorization con "Bearer " seguida del bearer token
        connection.setRequestProperty("User-Agent", app_name);
        connection.setRequestProperty ("Accept", "application/json");
        connection.setRequestProperty ("Authorization", "Bearer "+bearer_token);

        // TODO: Obtener el código de respuesta y comprobar que es correcto
        if(connection.getResponseCode() > 299 || connection.getResponseCode() < 200) {
          System.out.println("Código de respuesta es incorrecto("+connection.getResponseCode()+")");
        }

        // TODO: Obtener el flujo de entrada (InputStream) y deserializar su contenido en un objeto de tipo FollowerResponse
usando Gson
        Gson parser = new Gson();
        InputStream in = (InputStream) connection.getInputStream();
        FollowerResponse b = parser.fromJson(new InputStreamReader(in), FollowerResponse.class);

        // TODO: Devuelva el atributo users del obejto deserializado
        u= b.users;
        return u;
    }
}
```