ISET SFAX

AU 2019/2020 SEMESTRE I

Correction Devoir Surveillé

Classe : *SEM31*	***Matière :*** *Développement Mobile Avancé*	***Nb pages :*** *3*
Documents Non Autorisés	***Enseignant :*** *Souissi Hafedh*	***Durée :*** *1 heure*

Barème : 20 = 7 + 13

Exercice1(7points = 3+ 2 + 2)

```
private void remplir() {
    SQLiteTache b = new SQLiteTache(this, "taches.db", null, 1);
    SQLiteDatabase db = b.getWritableDatabase();
    Cursor c = db.rawQuery("Select * from tache where avn < 100;",null);
    adpT.clear();
    while (c.moveToNext()) {
        int id = c.getInt(0);
        String nom = c.getString(1);
        int avn = c.getInt(2);
        Tache t = new Tache(id, nom, avn);
        adpT.add(t);
    }
    c.close();
    db.close();
}

private void ajouter() {
    SQLiteTache b;
    b = new SQLiteTache(this, "taches.db", null, 1);
    SQLiteDatabase db;
    db = b.getWritableDatabase();
    ContentValues v = new ContentValues();
    v.put("nom", edNom.getText().toString());
    v.put("avn", 0);
    db.insert("tache", null, v);
    db.close();
    setResult(RESULT_OK);
    finish();
}



private void valider() {
    SQLiteTache b;
    b = new SQLiteTache(this, "taches.db", null, 1);
    SQLiteDatabase db;
    db = b.getWritableDatabase();
    ContentValues v = new ContentValues();
    v.put("avn", seekAvn.getProgress());
    db.update("tache",v,"id=?",new String[]{edId.getText().toString()});
    db.close();
    setResult(RESULT_OK);
    finish();
}
```

Exercice2 (13points = 4 + 9)

```
  private void ajouter() {
RequestQueue queue = Volley.newRequestQueue(this);
String url = " http://192.168.10.15 :80/LocalisationPoste/AjoutGouv.php";
StringRequest sr = new StringRequest(Request.Method.POST, url,
     new Response.Listener<String>() {
       @Override
       public void onResponse(String response) {
       try{
         JSONObject json = new JSONObject(response);
            String reponse = json.getString("ETAT");
            if (reponse.equals("SUCCES"))
               finish();
            else {
           Toast t = Toast.makeToast(Ajout.this,"Problème dans
                             Ajout!",Toast.LENGTH_LONG);
                 t.show() ;
            }
       } catch (JSONException error) {
             Toast t = Toast.makeText(Ajout.this, "Problème d'analyse JSON: " +
                     error.getMessage(), Toast.LENGTH_LONG);
                t.show();
       }
       }
     }, new Response.ErrorListener() {
          @Override
          public void onErrorResponse(VolleyError e) {
              Toast t = Toast.makeText(Ajout.this, "Problème d'appel HTTP: " +
                      e.getMessage(), Toast.LENGTH_LONG);
          t.show();
          }
        }) {
  @Override
  public Map<String, String> getParams() throws AuthFailureError {
    HashMap<String, String> headers = new HashMap<String, String>();
    headers.put("nom", edNom.getText().toString());
    headers.put("lat", edLat.getText().toString());
    headers.put("long", edLong.getText().toString());
    return headers;
  }
  };
  queue.add(sr);
}
```

```
private void rechercher() {

RequestQueue queue = Volley.newRequestQueue(this);
String url = " http://192.168.10.15 :80/LocalisationPoste/Recherche.php";
StringRequest sr = new StringRequest(Request.Method.POST, url,
     new Response.Listener<String>() {
       @Override
       public void onResponse(String response) {
try{
         JSONObject json = new JSONObject(response);
           JSONArray aPr = json.getJSONArray("bureaux");
           for (int i = 0; i < aPH.length(); i++) {
             JSONObject o = aPr.getJSONObject(i);
             int id = Integer.parseInt(o.getString("id"));
             String nom = o.getString("nom");
             String cp = o.getString("cp");
             double lat = Double.parseDouble(o.getString("lat"));
             double lon = Double.parseDouble(o.getString("long"));
             //afficher un marqueur
             MarkerOptions m = new MarkerOptions().position(new LatLng(lat, lon));
             m.title(nom);
             m.snippet(cp);
             mMap.addMarker(m);
          }
} catch (JSONException error) {
                Toast t = Toast.makeText(Recherche.this, "Problème d'analyse JSON: " +
error.getMessage(), Toast.LENGTH_LONG);
                t.show();
              }
       }
     }, new Response.ErrorListener() {
          @Override
          public void onErrorResponse(VolleyError e) {
              Toast t = Toast.makeText(Recherche.this, "Problème d'appel HTTP: " +
e.getMessage(), Toast.LENGTH_LONG);
          t.show();
          }
        }) {
  @Override
  public Map<String, String> getParams() throws AuthFailureError {
    HashMap<String, String> headers = new HashMap<String, String>();
    headers.put("gouv", (String) spGouv.getSelectedItem());
    headers.put("bureau", edNom.getText().toString());
    return headers;
  }
  };
  queue.add(sr);

   }
```