

**Devoir Surveillé****Classe : SEM31****Matière : Développement Mobile Avancé****Nb pages : 6****Documents Non Autorisés****Enseignant : Souissi Hafedh****Durée : 1 heure****Barème : 20 = 7 + 13****N.B. Ne donner que le code des méthodes demandées.****Exercice1 (7 points = 3 + 2 + 2)**

« **PlanificationTache** » est une application Android qui permet à un son utilisateur de gérer un ensemble de tâches avec une base de données SQLite "taches.db". Une tâche est définie par un id (id), un nom (nom) et un avancement (avn) de 0 à 100. L'utilisateur peut :

- consulter les tâches qui ne sont pas achevées (avancement < 100),
- ajouter une nouvelle tâche,
- modifier l'avancement d'une tâche.

« **PlanificationTache** » contient trois activités : « MainActivity », « Ajout » et « Modification ». De plus elle contient deux autres classes : « SQLiteTache » et « Tache ». Le code de ces classes et les interfaces sont donnés dans « Annexe1 ».

- 1- Donner le code de la méthode **remplir()** de la classe « **MainActivity** » qui permet de remplir le ListView par les tâches non achevées en utilisant la requête :

« **Select \* from tache where avn < 100;** ».

- 2- Donner le code de la méthode **ajouter()** de la classe « **Ajout** » qui permet d'ajouter la tâche saisie, de mettre le résultat à RESULT\_OK et de fermer l'activité.
- 3- Donner le code de la méthode **valider()** de la classe « **Modification** » qui permet de modifier l'avancement de la tâche, de mettre le résultat à RESULT\_OK et de fermer l'activité.

## Exercice2 (13points = 4 + 9)

« **LocalisationPoste** » est une application Android qui permet à son utilisateur d'ajouter des gouvernorats et des bureaux de poste à une base de données distante et d'effectuer des recherches de bureaux de poste par le gouvernorat et le nom du bureau ; le résultat d'une recherche est affiché sous forme de marqueurs sur une carte (Google Map). Un gouvernorat est défini par un id (id), un nom (nom), une latitude (lat) et une longitude (long). Un bureau de poste est défini par un id (id), un nom (nom), un code postal (cp), une latitude (lat), une longitude (long) et le nom d son gouvernorat (nom\_gov).

« **LocalisationPoste** » contient quatre activités : « MainActivity », « AjoutGouv », « AjoutBureau » et « Recherche ». Le code de ces classes et leurs interfaces sont donnés dans « Annexe2 ».

« **LocalisationPoste** » appelle une application Web hébergées dans l'adresse « `http://192.168.10.15:80/LocalisationPoste/` » et elle contient quatre pages « AjoutGouv.php », « AjoutBureau.php », « ListeGouv.php » et « Recherche.php » qui utilisent la méthode « POST ».

« AjoutGouv.php » prend **trois paramètres** qui sont le nom (nom), la latitude (lat) et la longitude (long) du gouvernorat, elle ajoute le gouvernorat à la table « Gouv » de la base MySQL et retourne `{" ETAT" : "SUCCES"}` en cas de succès ou `{" ETAT" : "ECHEC"}` en cas d'échec.

« ListeGouv.php » ne prend aucun paramètre et retourne un objet JSON qui contient tous les noms des gouvernorats.

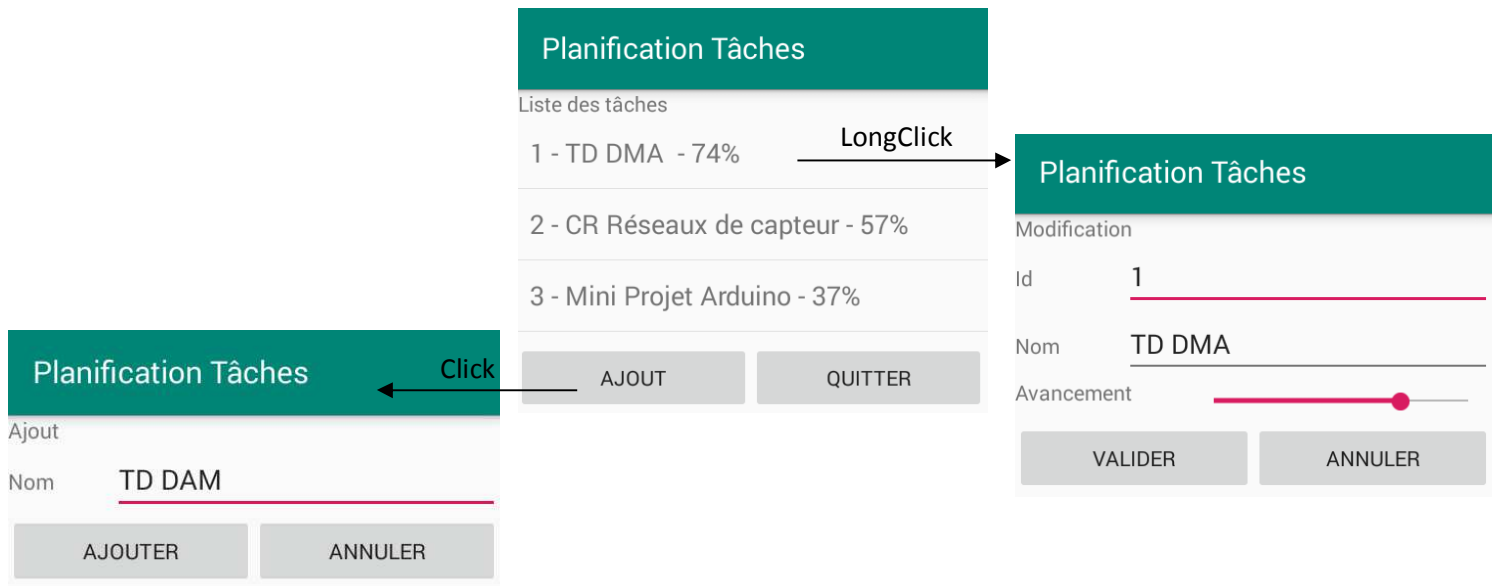
« Recherche.php » prend **deux paramètres** qui sont le gouvernorat (gouv) et le nom du bureau (bureau), elle effectue une recherche dans la table « bureau » de la base MySQL et retourne un objet JSON de la forme suivante :

```
{"bureaux": [
  {"id": "1", "nom": "ElBousten-Sfax", "cp": "3099",
    "lat": "34.7505372", "long" : "10.7674128" },
  {"id": "2", "nom": "Sidi Mansour-Sfax", "cp": "3094",
    "lat": "34.7613744", "long" : "10.6630581" }  ]}
```

- 1- Donner le code de la méthode **ajouter()** de la classe « **AjoutGouv** » qui permet d'ajouter le gouvernorat saisie, si la réponse est `{" ETAT" : "SUCCES"}` elle ferme l'activité, sinon elle affiche un message d'erreur.
- 2- Donner le code de la méthode **rechercher()** de la classe « **Recherche** » qui permet de :
  - a. rechercher les bureaux qui correspondent au gouvernorat et au nom du bureau,
  - b. analyser la réponse JSON,

- c. afficher un marqueur sur la carte pour chaque bureau trouvé. La latitude et la longitude du marqueur sont la latitude et la longitude du bureau. Le titre du marqueur est le nom du bureau. Le snippet du marqueur est le code postal du bureau.

## Annexe1



### SQLite

#### Insertion et modification

```
SQLiteBib b = new SQLiteBib(this, "base.db", null, 1);
SQLiteDatabase db;
db = b.getWritableDatabase();
ContentValues v = new ContentValues();
v.put("nb", Integer.parseInt("10"));
db.insert("livre", null, v);
db.update("livre", v, "id=?", new String[]{"1"});
db.close();
```

#### Selection

```
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
SQLiteDatabase db = b.getWritableDatabase();
Cursor c = db.rawQuery("Select * from livre where nbPage>50;", null);
adpLivre.clear();
while (c.moveToNext()) {
    int id = c.getInt(0); ...
    adpLivre.add(...);
}
```

```
package com.plan.tache;
public class Tache {
    private int id;
    private String nom;
    private int avn;
    public Tache(int id, String nom, int avn) {
        this.id = id;
        this.nom = nom;
        this.avn = avn;
    }
    public int getId() { return id; }
    public String getNom() { return nom; }
    public int getAvn() { return avn; }
    @Override
    public String toString() {
        return id + " - " + nom + " - " + avn + "%";
    }
}
```

```

package com.plan.tache;
public class SQLiteTache extends SQLiteOpenHelper {
    public SQLiteTache(Context context, String name, SQLiteDatabase.CursorFactory factory,
        int version) {
        super(context, name, factory, version);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String sql="create table tache (id INTEGER PRIMARY KEY AUTOINCREMENT,nom text NOT
        NULL,avn INTEGER );";
        db.execSQL(sql);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}

package com.plan.tache;
public class MainActivity extends AppCompatActivity{
    private ListView lstT;
    private ArrayAdapter<Tache> adpT;
    ...
    private void init() {
        ...
        adpT=new ArrayAdapter<Tache>(this,android.R.layout.simple_list_item_1);
        lstT.setAdapter(adpT);
        ajouterEcouteurs();
        remplir();
    }
    private void remplir() {
        //Permet de remplir le ListView par les tâches non achevées (avn < 100)
    }
}

package com.plan.tache;
public class Ajout extends AppCompatActivity {
    private EditText edNom;
    private Button btnAjouter;
    private Button btnAnnuler;
    ...
    protected void ajouter() {
        // Permet d'ajouter la tâche saisie, de mettre le résultat à RESULT_OK et de fermer
        // l'activité
    }
}

package com.plan.tache;
public class Modification extends AppCompatActivity {
    private EditText edId;
    private EditText edNom;
    private SeekBar seekAvn;
    private Button btnModifier;
    private Button btnAnnuler;
    ...
    protected void valider() {
        // Permet de modifier l'avancement de la tâche de mettre le résultat à RESULT_OK et
        // de fermer l'activité.
    }
}

```

# Annexe2

**Localisation Poste**

Ajout Gouvernorat

Nom **Sfax**

Latitude **34.7613744**

Longitude **10.6630581**

AJOUTER ANNULER

**Localisation Poste**

AJOUT GOUVERNORAT

AJOUT BUREAU

RECHERCHE BUREAU

**Localisation Poste**

Ajout Bureau

Nom **El Bousten**

Code Postal **3099**

Latitude **35.02038**

Longitude **10.52042**

Gouvernorat **Sfax**

AJOUTER ANNULER


**Localisation Poste**

Recherche Bureau

Gouvernorat **Sfax**

Nom Bureau **t**

RECHERCHER



## Volley

```
RequestQueue queue = Volley.newRequestQueue(this);
String url = "http://100.20.12.2:80/Page.php";
StringRequest sr = new StringRequest(Request.Method.POST,
    url, new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            //Traitement de la réponse
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError e) {
            //Traitement de l'erreur
        }
    }) {
    @Override
    public Map<String, String> getParams() throws AuthFailureError {
        HashMap<String, String> headers = new HashMap<String, String>();
        headers.put("param", "val");
        return headers;
    }
};
queue.add(sr);
```

## JSONObject

```
JSONObject json = new JSONObject("{\"c1\":\"v1\",\"c2\":\"12\",\"c3\":\"12.4\"");
String s = json.getString("c1");
int s = json.getInt("c2");
float s = Float.parseFloat(json.getString("c3"));
```

## JSONArray

```
try {
    JSONArray a = json.getJSONArray("liste");
    for (inti = 0; i<aLivre.length(); i++) {
        JSONObject o = a.getJSONObject(i);
    }
} catch (JSONException e) {
    e.printStackTrace();
}
```

## Placer un mrqueur sur une carte

```
static final float ISET_SFAX_LAT =34.756932f;
static final float ISET_SFAX_LONG =10.772176f;
MarkerOptions marker1 = new MarkerOptions().position(new LatLng(ISET_SFAX_LAT,
    ISET_SFAX_LONG));
    marker1.title("Iset Sfax");
    marker1.snippet("Sfax");
    mMap.addMarker(marker1);
```

```

package com.loc.poste;
//imports
public class MainActivity extends AppCompatActivity {
    ...
}
package com.loc.poste;
//imports
public class AjoutGouv extends AppCompatActivity {
    private EditText edNom;
    private EditText edLat;
    private EditText edLong;
    private Button btnAjouter;
    private Button btnAnnuler;
    ...

    protected void ajouter() {
        // permet d'ajouter le gouvernement saisie,
        // si la réponse est {" ETAT" : "SUCCES"} elle ferme l'activité,
        // sinon elle affiche un message d'erreur.
    }
}
package com.loc.poste;
public class AjoutBureau extends AppCompatActivity {
    ...
}
package com.loc.poste;
public class Recherche extends FragmentActivity implements OnMapReadyCallback {
    private Spinner spGouv;          // Contient les noms des gouvernorats
    private EditText edNom;
    private Button btnRechercher;
    private ArrayAdapter<String> adpGouv;
    ...
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        init();
    }
    private void init() {
        ...
        ajouterEcouteur();
    }

    protected void rechercher() {
        // Permet de :
        // - rechercher les bureaux qui correspondent au gouvernement et au nom du bureau,
        // - analyser la réponse JSON,
        // - afficher un marqueur sur la carte pour chaque bureau trouvé.
        // La latitude et la longitude du marqueur sont la latitude
        // et la longitude du bureau.
        // Le titre du marqueur est le nom du bureau.
        // Le snippet du marqueur est le code postal du bureau.
    }
}
}

```