Université d'El-Oued

Faculté des Sciences Exactes
Département d'informatique
3ème année LMD

14/05/2018
Durée :1h30
Applications mobiles

Examen

1 Problème

Notre université désire **développer une application mobile de suivi et localisation (tracking) des bus du transport universitaire**, où chaque **bus peut servir une ou plusieurs lignes**. Elle utilise aussi **les technologies Web** pour **accéder au système back-end de l'université**.

Exercice 1 *(Compréhension)(7 pts)*

1. Quel est le type de cette application ? **justifier** ?
2. Dans quelle plateforme mobile fonctionne-elle ? **justifier** ?
3. Quel est le type de client utilisé ? **justifier** ?
4. Etablir le diagramme de cas d'utilisation(use case)global ?
5. Spécifier l'ensemble des **besoins non fonctionnels** liés à l'application ?
6. Expliquer comment peut-on localiser les bus sous Android ?

Exercice 2 *(UI, Ressource, XML, Activity)(7 pts)*

L'application dispose d'une fonctionalité de consultation de lignes gérée par une interface graphique.

1. Donner les caractéristiques de ce type d'interfaces ?
2. Expliquer les mesures à prendre pour sa conception ?
3. Un utilisateur veut **interagir avec l'application** en utilsant **la voix**. Expliquer comment ?
4. Etant donnée **10 lignes identifiées** par un **code comme suit : XXXI** (XXX : désigne **les premières lettres du nom** d'un **lieu dans la ville d'El oued**, I : designe **numéro de la ligne**). En utilsant un **compsant graphique Spinner** ?
   (a) Ecrire le code d'activité permattant de décrire cette interface ?
   (b) Dessiner l'interface obtenue ?

Exercice 3 *(stockage de données, Persistance)(6 pts)*

Les informations d'une ligne sont stockées sous forme de table **LignTrUni(Num-lgn,Dest ,Num-bus ,Horarire ,Longitude ,Latitude)**

1. Quelle est la technique de stockage convenable ? **justifier** ?
2. Écrire le code JAVA qui déclare la classe **Lign-database** permettant de stocker les informations des lignes de bus.
3. Déclarer la classe qui représente **le modèle de données** ?
4. Écrire le corps de la méthode **TrackBus()** permettant de suivi la trajectoire d'un bus donné.

5. Faire un appel de cette méthode dans le code du service ?

Good Luck

Université d'El-Oued

Faculté des Sciences Exactes
Département d'Informatique
3eme année Licence
Module : **Applications mobiles**

Date : 14/05/2018
Durée : 1h30

Corrigé type de Contrôle

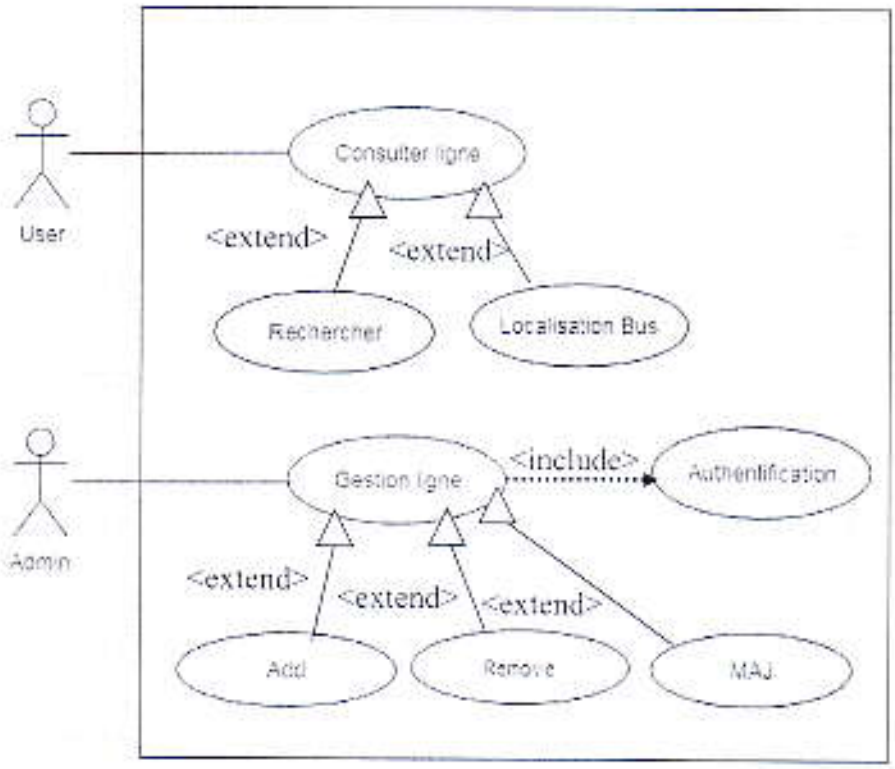
Exercice 1 : (7 points)

1) le type de cette application est **une application mobile hybride**. Parce qu'elle:
   - utilise des technologies Web pour fonctionner ce qui nécessite un environnement de développement Web dédié aux applications mobiles comme **Ionic**.
   - nécessite l'accès aux fonctionnalités du téléphone (GPS et service de localisation) pour localiser les **bus du transport universitaire**.

2) Pas de plateforme mobile spécifie. Cette application peut être déployée sur multiples plateformes mobiles (cross Platform). Ce type d'application mobile n'exige pas une plateforme mobile spécifique pour fonctionner. de plus avec **Ionic** on peut développer une application mobile multi-plateforme c.-à-d. fonctionne sous Web avec la possibilité d'accéder aux fonctionnalités du téléphone.

3) Le type de client est un **client lourd** puisque il nécessite un stockage temporaire des informations de localisation des bus nécessaires pour les suivre. Dans un environnement hostile (déconnexion fréquent de réseau) Ces informations seront ensuite envoyées au serveur de l'université pour le stockage permanant.

4)



le chargé du module

5) besoins non fonctionnels liés à cette application sont entre autres :

- Pour la localisation, elle récupère les informations nécessaires à partir de GPS ou Internet.
- Pour fonctionner elle nécessite une connexion à l'Internet. Ceci est assuré par le mobile data via l'accès à un réseau mobile (2G, 3G ou 4G).
- Intégration des systèmes existants puisque elle est considérée comme une extension mobile de système au niveau de l'université.

6) Pour localiser les bus sous Android on a besoin de :

- Une interface matérielle GPS (réseau satellitaire) ou Internet pour obtenir les informations de la localisation.
- Un service de localisation d'Android
  - Déclaré au niveau de fichier de configuration de l'application « AndroidManifest.xml »
  - Avoir les permissions nécessaires pour utiliser les fournisseurs de position.
  - L'implémentation de la classe décrivant ce service, celle-ci doit hériter la classe **android.app.Service.**
    - Redéfinition de la méthode onCreat()
  - Le démarrage de ce service à partir de l'activité via **startService()**.

Exercice 2 (UI, Ressource, XML, Activity) : (7 points)

1) Les caractéristiques d'une interface **graphique (GUI)** sont :

- Communication basée sur objets dessinés à l'écran
- Basée sur **souris, stylet, écran tactile, clavier** comme entrées.
- Assurée l'aspect ergonomique **(l'élégance et l'esthétique)**

2) les mesures à prendre pour sa conception :

- aspect ergonomique **de l'interface**.
- L'assurance **de l'expérience utilisateur mobile « User Experience (UX) mobile »**
- l'élégance et l'esthétique
- Simplicité et facilité d'usage

3) **un utilisateur peut communiquer vocalement** avec un terminal mobile (application) **à travers une interface vocale (VUI) via des moyens d'entrés** comme : **micro, kit mains libre.** Cette interface peut être intégrée dans une interface graphique pour mieux gérer l'IHM (sortie).

4) a- le code d'activité permettant de décrire l'interface de **consultation de lignes de bus** en utilisant le composant graphique **Spinner**

```
Public class SpinnerActivity extends Activity{

   String myData[] = {"tek1","sal2","oue3","bay4","rem5","mel6", "tri7","sou8","kou9", "has10"};

  Spinner myspinner;
  TextView selectionText;

@Override
Protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_spinner);

    myspinner= (Spinner)findViewById(R.id.spinner);
```

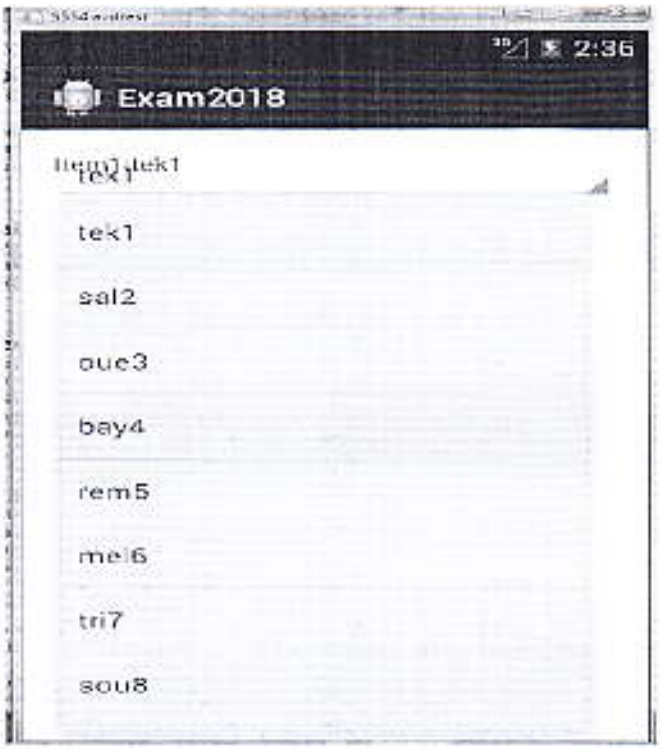

le chargé du module

```
    selectionText = (TextView)findViewById(R.id.selectionText);

// Create an ArrayAdapter using the string array and a default spinner layout
    ArrayAdapter<String> adapter = new ArrayAdapter<String>
(this,android.R.layout.simple_spinner_item,myData);
  // Specify the layout to use when the list of choices appears
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

     // Apply the adapter to the spinner
     mySpinner.setAdapter(adapter);
   myspinner.setOnItemSelectedListener(evenhandl);
    }
private OnItemSelectedListener evenhandl= new OnItemSelectedListener() {
@Override
    public void onItemSelected(AdapterView<?> parent, View view,
            int pos, long id) {
        selectionText.setText("Item" + (pos + 1)+ ":" + ((TextView) view).getText());
    }
@Override
    public void onNothingSelected(AdapterView<?> parent) {
        // Another interface callback
    }
}; }
```

b- l'interface obtenue ;



Exercice 3 (stockage de données, pertinence) : (7 points)

1) les informations d'une ligne de bus sont stockées **sous forme d'une base de données SQLite. Cette technique est convenable pour raisons de :**
   - les données stockées sont volumineuses et structurées.
   - L'application est une extension d'un système préexiste dont la coupure de connexion (réseau mobile) perturbe son fonctionnement (nécessité de stockage de données).

le chargé du module

2) classe permettant de stockés les informations des lignes de transport.

```
public class Lign-database extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION = 1;
    // Database Name
    private static final String DATABASE_NAME = "LignTrUnidb.db";
    // ligne table name
    private static final String TABLE_LIGNS = "LignTrUni";
    // ligns Table Columns names
    private static final String Col_ID = "Num-lgn";
    private static final String Col_DEST = "Dest";
    private static final String Col_NBUS = "Num-bus";
    private static final String Col_Long = "Longitude";
    private static final String Col_Lat = "Latitude";

    public Lign-database(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_LIGNS_TABLE = "CREATE TABLE " + TABLE_LIGNS + "("
                + Col_ID + "INTEGER PRIMARY KEY," + Col_DEST + " TEXT,"
                + Col_NBUS + "INTEGER," + Col_Long + " REAL," + Col_Lat + " REAL" ")";
        db.execSQL(CREATE_LIGNS_TABLE);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_LIGNS);
        // Create tables again
        onCreate(db);
    }
}
```

3) la classe qui représente le modèle de données

```
public class Lign {
    int _numlgn;
    int _numbus;
    String _dest;
    double _long;
    double _lati;
    ……
    // constructor
    public Lign(int numlgn, int numbus, double long, double lati,…){
          this._numlgn = numlgn;
          this._numbus = numbus;
          this._long = long;
          this._lati = lati;
          ……
    }
    public int getNBUS(){
    return this._numbus;
    }
    public void setNBUS(int numbus){
          this._numbus = numbus;
    }
    ……
    public double getLONG(){
          return this._prenom;
    }
```


le chargé du module

```
    public void setLONG(double long){
          this._long = long;
    }
public double getLATI(){
          return this._lati;
    }
    public int setLATI(double lati){
          this._lati = lati;
    }
}
```

4) la méthode **TrackBus() permettant de suivi la trajectoire d'un bus donné.**

```
void TrackBus(numbus, longitude, latitude) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(Col_Long, longitude);
    values.put(Col_Lat, latitude);
    // Inserting Row
    db.update(TABLE_LIGNS, values, Col_NBUS+ " = " + numbus, null);
    db.close(); // Closing database connection
}
```

5) appel de cette méthode dans le code du service

```
public class ServiceLoc extends Service {
        private LocationManager locationMgr = null;
        private LocationListener onLocationChange = new LocationListener()
{
@Override
        public void onLocationChanged(Location location)
        {
        Double latitude = location.getLatitude();
        Double longitude = location.getLongitude();
        TrackBus(numbus, longitude, latitude) ;
} };
}
```


le chargé du module