

TP Programmation d'une application android "LeBonCoin"

IMT Nord Europe

1 Introduction

Le secteur du bâtiment génère annuellement 46 millions de tonnes de déchets en France, mais ce potentiel de valorisation est largement sous-exploité. Ainsi bon nombres de déchets issus de la déconstruction des bâtiments pourraient en fait être vus comme des ressources utilisables à nouveau sur des chantiers. L'objectif de ce TP est de valoriser ces déchets comme des ressources à nouveau utilisables en développant une application qui mettra en relation des offrants (ceux qui disposent de matériaux à valoriser sur un site de déconstruction) et des demandeurs (à la recherche d'un type de matériau). L'application ressemblera à une sorte de "Bon coin des matériaux de seconde vie".

Dans la description qui suit, une base de l'application vous est proposé et sera à réaliser. Vous devrez ensuite choisir au minimum deux fonctionnalités à rajouter parmi un ensemble de fonctionnalités proposées.

2 Modalité d'évaluation

Vous serez évalué lors d'une soutenance qui aura lieu lors de la dernière séance. Vous devrez pour cette échéance préparer un .zip contenant les sources de votre projet, un .apk permettant d'installer l'application ainsi qu'une **démo fonctionnelle** qui sera présentée durant une dizaine de minutes. Il ne vous est pas demandé de préparer une présentation type Powerpoint pour cet exercice, seule la démo de l'application sera évaluée.

3 Base de l'application (1 séance)

L'application consiste à lister un certain nombre d'annonces et pour chacune d'entre elles des informations, comme leur titre, l'adresse et une image, par exemple. Voici les différentes fonctionnalités que nous allons tout d'abord programmer :

- Pouvoir naviguer entre les différentes activités.
- Modéliser une annonce, c.-à-d., représenter les informations de chaque annonce.
- Lister des annonces.
- Afficher une annonce.
- Ajouter une annonce à cette liste

Vous pouvez déjà créer les différents fichiers **layout** (**activity_main.xml**, **activity_listview_ad.xml** et **activity_add_ad.xml**) dans le dossier *layout* et les différentes activités (**MainActivity**, **AdListViewActivity** et **AdAddActivity**) dans votre dossier *java/com/example/votre_package/* :

3.1 Naviguer entre les activités

Au sein de votre activité principale (**MainActivity**), utiliser les deux boutons pour pouvoir passer dans les deux autres activités.



FIGURE 1 – Main activity



FIGURE 2 – Liste d'annonces

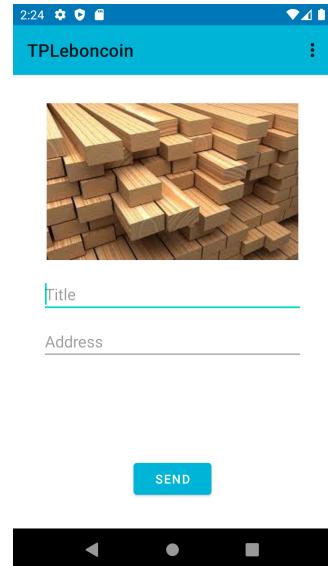


FIGURE 3 – Page d'ajout

FIGURE 4 – Résultat final à la fin de cette section

 Aidez-vous de la fonction *findViewById* pour obtenir les boutons en java. Appelez la fonction *startActivity* pour passer à une autre activité lorsque l'utilisateur clique sur un bouton à l'aide d'un *click listener*.

Vous devez être capable de passer de la **MainActivity** à l'activité **AdListViewActivity** et l'activité **AdAddActivity** à l'aide des 2 boutons puis de revenir vers l'activité principale avec le bouton retour du téléphone.



Attention, il faut ajouter vos activités dans votre Manifest (**AndroidManifest.xml**), sinon vous risquez d'avoir des erreurs.

3.2 Modéliser une annonce

Définissez une classe Java **AdModel** représentant un modèle d'une annonce, cette classe contient les différents attributs (informations) qui nous intéresse : titre, adresse et image.

```
public class AdModel {
    private String title;
    private String address;
    private int image;

    // Constructor
    public AdModel(String title, String address, int image) {
        this.title = title;
        this.address = address;
        this.image = image;
    }
}
```

```

// Getter and Setter
public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

// ...
}

```



On définit une image (**Drawable** en android) comme un **Integer**, car nous les manipulons avec leur adresse.

3.3 Lister les annonces

L'activité **AdListViewActivity** va nous permettre de définir un ensemble d'annonces et de l'afficher à l'aide d'une **list view**. Avant cela, il nous faut définir un modèle de design. Créez un nouveau fichier ressource (**item_listview_ad.xml**) dans le dossier **layout** puis placez-y une **image view** et deux **text view**. Ils ont pour rôles de contenir les informations d'une annonce, informations que l'on donnera par la suite.

Créez un certain nombre d'annonces (**AdModel**) dans votre activité **AdListViewActivity** puis mettez-les dans une liste. Afin de pouvoir afficher votre liste d'annonce, il est nécessaire de créer un **Adapter** (**AdAdapter**) qui hérite de **BaseAdapter**. C'est cet **Adapter** qui va transformer votre instance Java d'annonce en un élément de la liste déroulante, en utilisant le design que vous venez de réaliser comme modèle.

```

public class AdAdapter extends BaseAdapter {
    private final Context context;
    private final ArrayList<AdModel> adModelArrayList;
    private final LayoutInflater inflater;

    // Constructor
    public AdAdapter(Context context, ArrayList<AdModel> adModelArrayList) {
        this.context = context;
        this.adModelArrayList = adModelArrayList;
        inflater = (LayoutInflater.from(context));
    }

    @Override
    public int getCount() { return ... ; } // Return ad number

    @Override
    public Object getItem(int i) { return ... ; } // Return ad number i

    @Override
    public long getItemId(int i) { return ... ; } // Return ad id i

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        // Get ad number i
    }
}

```

```

AdModel ad = ... ;

view = inflater.inflate(R.layout.votre_design_layout, null);

// Get the image view and both text views
ImageView imageIV = ... ;
TextView titleTV = ... ;
TextView addressTV = ... ;

imageIV.setImageResource(...);
titleTV.setText(...);
addressTV.setText(...);
return view;
}
}

```

En instanciant votre adaptateur et en utilisant la fonction *setAdapter* sur votre list view, récupérer précédemment dans votre activité **AdListViewActivity**, vous devriez obtenir le résultat de la figure 2.

3.4 Voir une annonce

Vous devez être capable de cliquer sur une annonce de votre liste et d'afficher une page avec les informations de celle-ci. Créer une nouvelle activité **AdViewActivity** qui va afficher ces informations. Ensuite, dans l'activité **AdListViewActivity** ajouté un **OnItemClickListener** à votre **list view** et faite passer les informations de l'annonce cliqué à votre nouvelle activité.

 Récuperez la position de l'item cliqué : *onItemClick(AdapterView adapterView, View view, int position, long l)*. Avec la méthode *getItem* de votre adaptateur, vous pouvez récupérer les informations de l'annonce à l'aide de la position de celle-ci dans la liste.

3.5 Créez une annonce

Pour ajouter une annonce à votre liste, nous allons utiliser l'activité **AdAddActivity**. Si ce n'est pas déjà fait, commencez par faire le layout de l'activité, vous pouvez vous inspirer de la figure 7.

Dans l'activité, utilisez un *click listener* sur le bouton de validation. C'est lorsque l'utilisateur clique sur ce bouton que vous récupérez les informations des champs remplis pour les envoyer vers une autre activité. Pour le moment, il n'est pas prévu de stocker les informations, envoyez les informations à l'activité **AdListViewActivity**. Il vous faut modifier légèrement celle-ci pour qu'elle prenne en considération les informations envoyées. Vous devriez voir votre nouvelle annonce parmi celle qui était déjà présentes. Vu que les informations ne sont pas stockées, c'est normal si votre annonce s'efface lorsque vous réalisez un changement d'activité.

4 Fonctionnalité à rajouter (2 séances)

Vous pouvez désormais choisir différentes fonctionnalités à réaliser. Pour rappel, vous devez compléter votre application avec au moins une fonctionnalité.

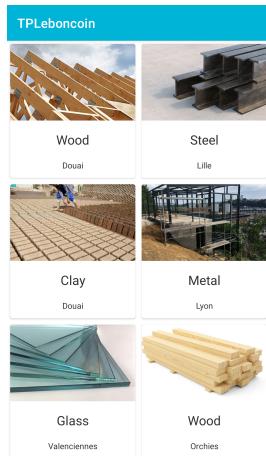


FIGURE 5 – Card view

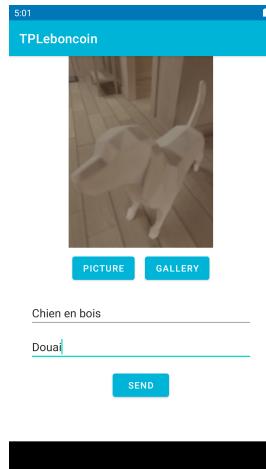


FIGURE 6 – Caméra

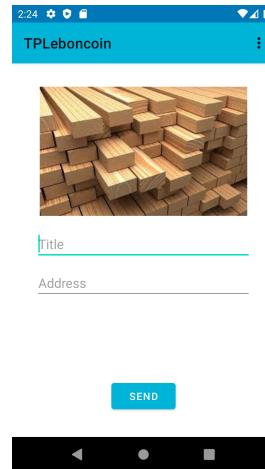


FIGURE 7 – Page d'ajout

FIGURE 8 – Différentes fonctionnalités supplémentaires

À partir de maintenant vous allez devoir modifier ce qui à été fait, je vous encourage à sauvegarder la base de votre application ou à copier-coller les différentes activités pour les modifier.

4.1 Vue en carte

Il est possible de faire des vues un peu plus sympa qu'une simple liste déroulante, par exemple les **card view** permettent d'avoir un design plus soigné (voir figure 5).

Pour pouvoir réaliser une vue en 2 ou plusieurs colonnes, il est nécessaire de transformer la **list view** en **recycler view**, mais cela va entraîner des modifications supplémentaires pour l'adaptateur.

```
public class RecyclerViewAdAdapter extends  
    RecyclerView.Adapter<RecyclerViewAdAdapter.RecyclerViewHolder>{  
    private List<AdModel> data;  
  
    public RecyclerViewAdAdapter(List<AdModel> data) {this.data = data;}  
  
    @NonNull  
    @Override  
    public RecyclerViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
        LayoutInflator inflater = LayoutInflator.from(parent.getContext());  
        View view = inflater.inflate(R.layout.item_card_material, parent, false);  
        return new RecyclerViewHolder(view);  
    }  
  
    @Override
```

```

public void onBindViewHolder(@NonNull RecyclerViewHolder holder, int position) {
    // This method is called for each of the visible rows displayed in our RecyclerView. It is
    // usually here that we update their appearance.
    AdModel ad = data.get(...);

    holder.titleTextView.setText(...);
    holder.addressTextView.setText(...);
    holder.imageView.setImageResource(...);
}

@Override
public int getItemCount() {return data.size();}

public static class RecyclerViewHolder extends RecyclerView.ViewHolder {
    public final ImageView imageView;
    public final TextView titleTextView;
    public final TextView addressTextView;

    public RecyclerViewHolder(@NonNull View itemView) {
        super(itemView);
        imageView = itemView.findViewById(...);
        titleTextView = itemView.findViewById(...);
        addressTextView = itemView.findViewById(...);
    }
}

```

4.2 Ajout d'image par appareil photo

Il s'agit de pouvoir récupérer une image à l'aide de la caméra ou d'une image dans la galerie du téléphone et de l'afficher dans votre activité, comme sur l'image 6. Nous allons utiliser des **ActivityResultLauncher** qui vont nous permettre de lancer une activité (caméra ou galerie) et de réaliser des choses en fonction du résultat et des données renvoyés (ajouté l'image dans votre activité).



Vous pourrez trouver la base du code à remplir sur MyLearningSpace.

Faites attention, vous devez ajouter les permissions dans votre **AndroidManifest.xml**. Un plus serait de rajouter la demande de permission, je vous laisse chercher comment faire sur internet !

4.3 Base de données

Jusqu'à maintenant, les données des annonces étaient écrites dans le code directement, ce que l'on ne veut surtout pas ! Nous allons donc transférer ces données dans une base de données SQLite. Dans un même temps, nous allons gérer les images via des URLs (String) et non plus comme des adresses (Integer) dans Android Studio. Pour ce faire, nous allons créer deux classes Java :

- **DBHelper**. Permet de gérer la création et les requêtes de la base de données. Donne toutes les informations concernant la base de données (nom, noms des tables, noms des colonnes, version, ...).
- **DBManager**. Va faire le lien entre vos modèles et la base données, exemple : fonction qui créer une annonce en base de données.

C'est le **DBManager** qui vous instancierait dans vos activités pour récupérer/modifier/supprimer les annonces.



Vous pourrez trouver le code des classes **DBHelper** et **DBManager** sur MyLearningSpace.

Après avoir implémenté ces classes, il vous faut modifier vos activités qui affichent les annonces. Vous aurez donc dans votre activité **AdListView** :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    DBManager dbManager = DBManager.getDBManager(this);
    dbManager.open();
    Cursor cursor = dbManager.fetch();

    CursorAdapter adapter = new DbAdAdapter(this, cursor, R.layout. ...);
    adapter.notifyDataSetChanged();

    listView.setAdapter(adapter);
    ...
}
```

Vous remarquerez qu'il vous faut un adaptateur particulier lorsque l'on utilise une base de données (**CursorAdapter**) :

```
public class DbAdAdapter extends CursorAdapter {

    private final int item_layout;

    public DbAdAdapter(Context context, Cursor c, int layout) {
        super(context, c);
        item_layout = layout;
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup viewGroup) {
        return LayoutInflater.from(context).inflate(item_layout, viewGroup, false);
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        TextView idTextView = (TextView) view.findViewById(...);
        TextView titleTextView = (TextView) view.findViewById(...);
        TextView addressTextView = (TextView) view.findViewById(...);
        ImageView imageView = (ImageView) view.findViewById(...);

        String id = cursor.getString(cursor.getColumnIndexOrThrow(DBHelper._ID));
        String title = cursor.getString(cursor.getColumnIndexOrThrow(DBHelper.TITLE));
        String address = cursor.getString(cursor.getColumnIndexOrThrow(DBHelper.ADDRESS));
        String image = cursor.getString(cursor.getColumnIndexOrThrow(DBHelper.IMAGE));

        idTextView.setText(id);
        titleTextView.setText(title);
```

```
addressTextView.setText(address);
Glide.with(view).load(image).into(imageView); \\ Glide is a library to insert an image into
an imageview with a url.
}
}
```

Pour plus d'explications ou de détails, vous pouvez trouver beaucoup de tutoriel sur internet concernant les bases de données SQLite pour Android [ici](#) ou sur d'autres manières de faire, ici avec [Firebase](#).

5 Fonctionnalités en plus possibles (pour les plus rapides)

- **Bouton d'appel.** Ajouter un bouton avec le numéro de téléphone de la personne et compose le numéro lorsque vous appuyez dessus (renvoie vers votre application d'appel).
- **Bouton pour mail.** Même principe, ajouter un bouton permettant d'envoyer un mail à la personne.
- **Action Bar.** Ajout d'une barre en haut de votre application pour afficher un titre (nom de l'application, accueil, ajout d'annonce, etc).
- **Options.** Ajout d'option (en haut à droite de l'application) pour régler différentes choses, vous pouvez faire changer les couleurs de votre application juste pour l'exemple.
- **Google Map.** Vous pouvez rajouter une carte via google map sur l'affichage de vos annonces.
- **Navigation drawer.** Ajout d'un menu sur le côté permettant de naviguer entre les activités/fragments, on évite ainsi les boutons dans l'activité principale.