

Rapport de projet de développement d'une application mobile : LotoQuinote

David Boisedu et Alexandre Boyer

15 mai 2020

Résumé

L'application que nous avons développée répond à la problématique suivante :
Comment faciliter le suivi de tirage des numéros d'un loto-quiné grâce à une application mobile ?

Sommaire

1	Introduction	2
2	Description générale de l'application LotoQuinote	2
3	Architecture du code	4
3.1	La partie "Contrôleur" de l'application	5
3.1.1	MainActivity	5
3.1.2	DrawTrackingActivity	6
3.1.3	DrawListActivity	8
3.1.4	ModifyPopUpActivity	9
3.2	La partie "Modèle" de l'application	10
3.2.1	La classe "Number"	10
3.2.2	La classe "Tirage"	10
3.2.3	La classe "TirageAdapter"	10
4	Quelques points délicats/intéressants	11
4.1	Choix de la méthode d'édition et de suppression des tirages	11
5	Conclusion	11
6	Références	12

1 Introduction

Cela peut être parfois fastidieux de noter sur une feuille tous les numéros sortis lors d'un tirage de loto-quiné. C'est pour cette raison précise que nous avons décidé de créer une application qui facilitera cette tâche. En plus de permettre de suivre le tirage d'un loto-quiné, l'application LotoQuinote permet aussi d'enregistrer chaque tirage pour pouvoir les consulter à n'importe quel moment. De plus, grâce à des fonctions de tri, il est possible de voir très rapidement les numéros qui ont été tirés lors d'un tirage.

2 Description générale de l'application LotoQuinote

L'application LotoQuinote comporte trois vues :

— La vue d'accueil :

Cette vue permet la création d'un nouveau suivi de tirage et affiche tous les suivis de tirage créés sous forme d'une liste pouvant être défilée.

LotoQuinote

Suivi du tirage n° 1

Suivi du tirage n° 2

Suivi du tirage n° 3

Suivi du tirage n° 4

Suivi du tirage n° 5

Suivi du tirage n° 6

Suivi du tirage n° 7

Suivi du tirage n° 8

Suivi du tirage n° 9

Suivi du tirage n° 10

Suivi du tirage n° 11

Suivi du tirage n° 12

Suivi du tirage n° 13

CRÉER UN NOUVEAU TIRAGE

Suivi du tirage n° 1

Date de modification : 14-05-20 5:6

ANNULER

VALIDER

SUPPRIMER

La vue d'accueil possède aussi une fonctionnalité d'édition lors d'un appui-long sur un suivi de tirage. Cette fonctionnalité permet de :

- Modifier le titre du suivi de tirage
- Afficher la date de dernière modification
- Supprimer le suivi de tirage

— La vue de suivi de tirage :

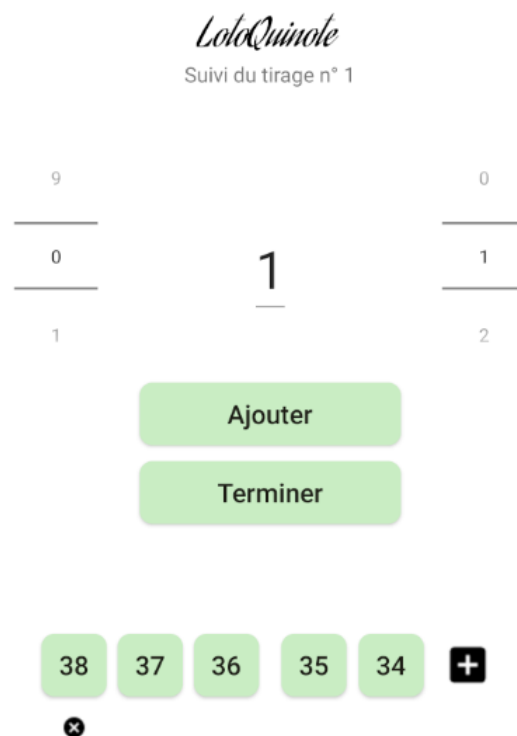
La vue de suivi de tirage permet de choisir le numéro qui a été tiré lors du tirage du loto-quine. Le numéro est choisit soit par le biais du NumberPicker ou soit en l'écrivant soi-même grâce à l'EditText.

Lors du clique sur le bouton "Ajouter", le numéro choisit est inséré à la liste des numéros tirés. Les cinq derniers numéros tirés de cette liste sont visibles dans les cases situées en bas de cette vue.

Le bouton "Terminer" permet de sauvegarder les numéros ajoutés à la liste et de revenir à la vue d'accueil.

La croix située en-dessous de la 1ère case à gauche permet de supprimer le dernier numéro ajouté à la liste.

Le bouton "+" à droite de la dernière case permet de basculer sur la vue de la liste complète des numéros tirés.



Remarque : A chaque ajout d'un numéro à la liste, un toast nous informe que celui-ci a bien été ajouté. Si le numéro que l'on veut ajouter est déjà présent dans la liste un toast apparaît pour signaler le fait que ce numéro a déjà été tiré et qu'il ne sera donc pas ajouté.

— La vue de la liste des numéros tirés :

LotoQuinote

38	37	36	35	34
33	32	31	30	29
28	27	26	25	24
23	22	20	18	17
16	15	14	13	12
21	11	19	46	51

Cette vue permet de voir l'ensemble des numéros tirés lors du tirage du loto-quiné. Par défaut, les numéros sont triés par ordre de tirage : le dernier numéro tiré est le premier de la liste.

D'autres fonctions de tri sont disponibles par le biais d'une petite liste déroulante. On peut ainsi trier les numéros par :

- Ordre de tirage
- Ordre croissant
- Ordre décroissant

Trier par : **Ordre de tirage** ▼

Retour

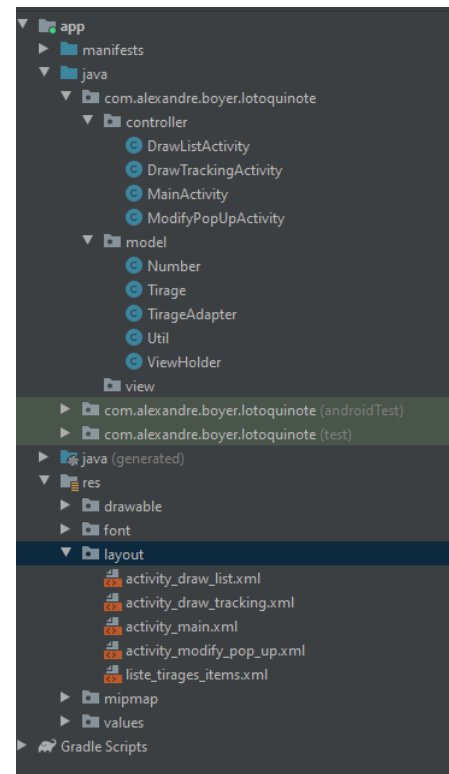
3 Architecture du code

Le code de notre application suit l'architecture MVC (Modèle, Vue, Contrôleur).

Ainsi, la partie "Contrôleur" contient les activités principales de notre application.

La partie "Modèle" contient les classes que nous utilisons pour définir les objets nécessaires à notre application.

Et la partie "Vue" représenté par le dossier "layout" contient les fichiers .xml qui permettent l'interaction entre l'application et l'utilisateur.



3.1 La partie "Contrôleur" de l'application

3.1.1 MainActivity

Le code de MainActivity est construit en suivant cette architecture :

Tout d'abord, la liste qui va contenir les tirages est créée par le biais de la classe objet "TirageAdapter" :

```
// Création de la liste des tirages
mTirageAdapter = new TirageAdapter( context: this, (ArrayList<Tirage>) draws);
mListView.setAdapter(mTirageAdapter);
```

Ensuite, une écoute sur le bouton "Créer un nouveau tirage" est réalisé pour permettre la création d'un tirage lors de l'appui sur celui-ci. Un toast est aussi créé lors de l'appui pour signaler la création d'un nouveau tirage :

```
// Création du tirage
Date today = new Date();
Tirage mDraw = new Tirage( title: "Suivi du tirage n° " + (draws.size() + 1), today);
draws.add(mDraw);
Toast toast = Toast.makeText(getApplicationContext(), text: "" + mDraw.getTitle() +
    " crée", Toast.LENGTH_SHORT);
toast.setGravity(Gravity.CENTER_VERTICAL, xOffset: 0, yOffset: 550);
toast.show();

mTirageAdapter = new TirageAdapter(mContext, (ArrayList<Tirage>) draws);
mListView.setAdapter(mTirageAdapter);
saveData();
```

Un délai a été mis en place sur ce bouton pour qu'on puisse l'actionner qu'une fois toutes les deux secondes, cela pour éviter de spammer la création de tirages :

```
// Timer de 2s après appui sur le bouton qui permet d'ajouter un tirage
Timer buttonTimer = new Timer();
buttonTimer.schedule(() -> {
    runOnUiThread(() -> {
        mNewDrawButton.setEnabled(true);
    });
}, delay: 2000);
```

De la même manière, une écoute sur les items de la liste est réalisée pour pouvoir changer d'activité lors de l'appui. Ainsi, si l'utilisateur appui sur un tirage de la liste, il sera dirigé vers la vue de suivi de tirage (DrawTrackingActivity).

Il en est de même pour la gestion de l'appui long sur un tirage : lorsque l'utilisateur réalise un appui long sur un tirage de la liste, une fenêtre d'édition apparaît comme expliqué plus haut.

Enfin, la dernière partie du code de MainActivity concerne la sauvegarde des données qui se fait dans notre cas grâce aux SharedPreferences [1].

3.1.2 DrawTrackingActivity

Le code de DrawTrackingActivity est construit en suivant cette architecture :

Au début de cette activité, nous récupérons le titre du tirage depuis MainActivity pour l'afficher sur la vue de DrawTrackingActivity grâce la classe Intent [2] :

```
Intent intent = getIntent();
if(intent != null)
{
    //mDrawTitle.setText(intent.getStringExtra("mDrawObject"));
    Tirage originalDraw = (Tirage) intent.getSerializableExtra( name: "mDrawObject");
    assert originalDraw != null;
    //Ici, on ne fait pas de mDraw = draw, car l'objet ne sera pas dupliqué
    //On utilisera la méthode copy() de la classe Tirage qui permet de faire cela
    mDraw.copy(originalDraw);
    mDrawTitle.setText(mDraw.getTitle());
}
```

Dans la suite du code, nous gérons les interactions avec les différents éléments présents sur cette activité. Par exemple, nous avons codé le fonctionnement des NumberPickers pour qu'ils répondent aux besoins d'une application qui permet le suivi du tirage d'un loto-quiné et donc par conséquent des numéros qui doivent être compris entre 0 et 90 :

```
// Gestion de la sélection des nombres dans les NumberPicker
mNumberPicker1.setOnValueChangedListener((picker, oldVal, newVal) → {
    mNumber.setNumber((newVal*10) + mNumber.getNumber()%10);
    int numberSet = mNumber.getNumber();
    if(numberSet <= 90)
    {
        mNumber.setNumber(numberSet);
        mNumberDrewText.setText(String.valueOf(mNumber.getNumber()));
    }
    else
    {
        // Un numéro du lotoquiné ne peut pas être supérieur à 90,
        // on gère donc ce cas en "bloquant"
        // les NumberPicker pour qu'ils donnent tous le temps la valeur 90
        // quand la valeur choisit par l'utilisateur est supérieure à 90
        {
            mNumber.setNumber(90);
            mNumberPicker1.setValue(9);
            mNumberPicker2.setValue(0);
            mNumberDrewText.setText(String.valueOf(mNumber.getNumber()));
        }
    }
});
```

La gestion de l'ajout et de la suppression des numéros à la liste de tirage constitue la dernière partie du code de DrawTrackingActivity. Ainsi, dans cette partie du code, grâce à un listener sur le bouton "Ajouter", on ajoute le numéro choisit à la liste en fonction de s'il a déjà été tiré ou non. De plus, on gère aussi l'affichage des cinq derniers numéros tirés dans les cases situées en bas de cette activité. Cette gestion de l'affichage se fait à l'aide d'un switch qui se base sur la taille de la liste.

```
mDraw.addNumber(mNumber);
switch(mDraw.getDraw().size()) // permet d'afficher les 5 derniers nombres
    // tirés dans les 5 cases situées en bas de cette activité
    // le dernier nombre tiré est celui le plus à gauche
{
    case 0:
        break;
    case 1:
        mPos1List.setText(mDraw.getNumberAt( i: mDraw.getDraw().size()-1).toString());
        break;
    case 2:
        mPos1List.setText(mDraw.getNumberAt( i: mDraw.getDraw().size()-1).toString());
        mPos2List.setText(mDraw.getNumberAt( i: mDraw.getDraw().size()-2).toString());
        break;
    case 3:
        mPos1List.setText(mDraw.getNumberAt( i: mDraw.getDraw().size()-1).toString());
        mPos2List.setText(mDraw.getNumberAt( i: mDraw.getDraw().size()-2).toString());
        mPos3List.setText(mDraw.getNumberAt( i: mDraw.getDraw().size()-3).toString());
        break;
```

Dans ce switch, la position des numéros dans les cases change en fonction du nombre de numéros pour que le dernier numéro tiré soit tous le temps celui le plus à gauche.

La gestion de la suppression du dernier numéro tiré par le biais de l'ImageButton représentant une petite croix se fait de la même manière que lors de l'ajout d'un numéro, sauf qu'ici, au lieu d'ajouter un numéro de la liste, on l'enlève de cette liste :

```
// Quand on clique sur la petite croix en dessous du dernier numéro tiré, on supprime ce numéro
mDeleteNumberPos1.setOnClickListener((v) → {
    int deletedNumber = mDraw.getNumberAt( i: mDraw.getDraw().size()-1).getNumber();

    mDraw.deleteNumber();
```

Enfin, l'appui sur le bouton "+" permet de lancer la troisième et dernière activité de notre application, DrawListActivity. Se faisant, on récupère aussi tous les numéros ajoutés à la liste grâce à ces deux lignes de code :

```
Intent drawListActivity = new Intent(DrawTrackingActivity.this, DrawListActivity.class);
drawListActivity.putExtra("drawList", mDraw);
```

Ainsi, ces deux lignes permettent de récupérer les données de l'activité courante pour les transférer à l'activité suivante.

3.1.3 DrawListActivity

Le code de DrawListActivity est construit en suivant cette architecture :

Tout comme pour la DrawTrackingActivity, nous allons récupérer le tirage par le biais de la classe Intent :

```
// On récupère le tirage
Intent intent = getIntent();
if(intent != null)
{
    Tirage originalDraw = (Tirage) intent.getSerializableExtra( name: "drawList");
    assert originalDraw != null;
    mDraw.copy(originalDraw);
}
```

Suite à cela, nous allons mettre en place un Spinner[3], qui va permettre à l'utilisateur de choisir la manière dont les numéros sont triés pour les afficher :

```
// Gestion du spinner qui permet de choisir la méthode de tri de la liste des numéros tirés
ArrayAdapter<CharSequence> spinnerAdapter = ArrayAdapter.createFromResource( context: this,
    R.array.sort_array, android.R.layout.simple_spinner_dropdown_item);
spinnerAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
mSpinner.setAdapter(spinnerAdapter);
```

Enfin, le comportement de notre Spinner va être géré par la structure de contrôle switch. Chacun des "cases" sera lié à un élément de notre Spinner, et va permettre à l'utilisateur de choisir le mode de tri des numéros, et ainsi changer l'affichage. Les affichages sont gérés par la création d'un ArrayAdapter[4] sur notre élément graphique GridView[5] :

```
switch(position)
{
    case 0: // Tri par ordre de tirage
        ArrayAdapter<Number> gridViewArrayAdapter = new ArrayAdapter<~>
            (getApplicationContext(), android.R.layout.simple_list_item_1,
                mDraw.drawedSort());
        gv.setAdapter(gridViewArrayAdapter);
        break;
    case 1: // Tri par ordre croissant (on utilise la fonction
        // ascendingSort définie dans l'objet Tirage)
        System.out.println("Ordre croissant");
        gridViewArrayAdapter = new ArrayAdapter<Number>
            (getApplicationContext(), android.R.layout.simple_list_item_1,
                mDraw.ascendingSort());
        gv.setAdapter(gridViewArrayAdapter);
        break;
    case 2: // Tri par ordre décroissant (on utilise la fonction
        // descendingSort définie dans l'objet Tirage)
        System.out.println("Ordre décroissant");
        gridViewArrayAdapter = new ArrayAdapter<Number>
            (getApplicationContext(), android.R.layout.simple_list_item_1,
                mDraw.descendingSort());
        gv.setAdapter(gridViewArrayAdapter);
        break;
    default:
        break;
}
```


3.1.4 ModifyPopUpActivity

Cette activité permet la gestion de l'interface d'édition qui apparaît lors de l'appui long sur un tirage de la vue d'accueil. Le code de cette activité se décompose comme cela :

On récupère encore une fois les données du tirage grâce à un Intent :

```
Intent drawToModify = getIntent();  
final Tirage currentDraw = (Tirage) drawToModify.getSerializableExtra( name: "currentDraw");  
final int positionDraw = drawToModify.getIntExtra( name: "position", defaultValue: -1);
```

On met à jour les éléments graphiques de la vue avec les données récupérées par l'Intent :

```
mDrawNameEdtTxt.setText(currentDraw.getTitle());  
mDrawDateTxt.setText(currentDraw.getDate());
```

On gère ensuite les interactions avec les différents boutons présents sur cette activité :

```
mCancelBtn.setOnClickListener((v) → {  
    currentDraw.setTitle(oldDrawTitle);  
    modifiedDraw.putExtra( name: "canceledDraw", currentDraw);  
    setResult(MODIFY_POPUP, modifiedDraw);  
    finish();  
});  
  
mDeleteBtn.setOnClickListener((v) → {  
    modifiedDraw.putExtra( name: "deletedDraw", value: 1);  
    modifiedDraw.putExtra( name: "position", positionDraw);  
    setResult(MODIFY_POPUP, modifiedDraw);  
    finish();  
});  
  
mValidateBtn.setOnClickListener((v) → {  
    currentDraw.setDate(new Date());  
    modifiedDraw.putExtra( name: "modifiedDraw", currentDraw);  
    modifiedDraw.putExtra( name: "position", positionDraw);  
    setResult(MODIFY_POPUP, modifiedDraw);  
    finish();  
});
```

3.2 La partie "Modèle" de l'application

L'application LotoQuinote se base sur 3 principaux modèles :

3.2.1 La classe "Number"

Cette classe permet de définir l'objet Number qui est l'un des objets indispensables de notre application. Celle-ci implémente les interfaces Serializable[6] et Comparable[7] pour pouvoir respectivement utiliser la classe Intent et comparer les objets Number entre eux.

La classe Number dispose de Getter/Setter simples, d'une méthode toString() qui convertit un objet Number en chaîne de caractères et d'une méthode compareTo() liée à l'interface Comparable.

3.2.2 La classe "Tirage"

Cette classe définit l'objet Tirage. Cet objet est au coeur de notre application puisque c'est sur lui que repose le principe de suivi de tirage. Lui aussi implémente l'interface Serializable qui permet l'utilisation de l'Intent.

Cette classe possède trois attributs :

- Un titre
- Une date
- Un ArrayList d'objets Number

Elle possède des getter/setter pour chacun de ces attributs ainsi que des méthodes supplémentaires utiles dans les différentes activités de l'application, comme notamment les méthodes de tri :

```
public ArrayList<Number> ascendingSort(){
    //On créer la ArrayList à retourner
    ArrayList<Number> sorted_draw = new ArrayList<>();
    //On copie la ArrayList draw dans la nouvelle ArrayList
    for(Number nb : draw){
        sorted_draw.add(nb);
    }

    //On trie la ArrayList à retourner
    Collections.sort(sorted_draw);
    //On retourne ensuite le tirage trié*/
    return sorted_draw;
}

public ArrayList<Number> descendingSort(){
    //On créer la ArrayList à retourner
    ArrayList<Number> sorted_draw = new ArrayList<>();
    //On copie la ArrayList draw dans la nouvelle ArrayList
    for(Number nb : draw){
        sorted_draw.add(nb);
    }
}
```

3.2.3 La classe "TirageAdapter"

Cette classe définit l'objet TirageAdapter. Elle prend en paramètre un ArrayList d'objets Tirage et possède différentes méthodes qui permettent de gérer la liste des tirages de l'application LotoQuinote.

4 Quelques points délicats/intéressants

4.1 Choix de la méthode d'édition et de suppression des tirages

Au tout début du projet, concernant l'activité "MainActivity", nous avions pour objectif de gérer la modification/-suppression des différents tirages par le biais de CheckBox. En effet, elles auraient permis de choisir le ou les tirages à modifier/supprimer, puis par le biais d'icônes, de réaliser l'action souhaitée.

Malheureusement, cela nous a pris beaucoup de temps sur notre projet ainsi que beaucoup de bugs à la chaîne. Nous avons donc pris le choix d'abandonner cette idée et de partir sur la création d'une nouvelle activité permettant la modification/suppression d'un tirage, se manifestant sous la forme d'une Pop-Up.

5 Conclusion

La réalisation de ce projet s'est plutôt bien déroulé. Nous avons pu réaliser ce que nous souhaitions, c'est-à-dire, une application permettant la sauvegarde de tirages d'un loto-quine. Malgré les difficultés rencontrées, nous avons également réussi à respecter les consignes imposées, cela grâce à une bonne entente et une bonne organisation du travail.

Nous pensons que ce projet peut encore s'étendre, notamment avec l'implémentation de paramètres de personnalisation de l'interface, la gestion des joueurs (sauvegarde des noms, nombres de victoires, etc ...), et des gains.



6 Références

- [1] Tutos android, *SharedPreferences* : <http://tutos-android-france.com/sharedpreferences-2/>
- [2] Android Developers, Intent Class : <https://developer.android.com/reference/android/content/Intent>
- [3] Android Developers, *Spinners* : <https://developer.android.com/guide/topics/ui/controls/spinner>
- [4] Medium, *Array Adpaters* : <https://medium.com/mindorks/custom-array-adapters-made-easy-b6c4930560dd>
- [5] Android Developers, *GridView* : <https://developer.android.com/reference/android/widget/GridView>
- [6] Android Developers, *Serializable* : <https://developer.android.com/reference/java/io/Serializable>
- [7] Android Developers, *Comparable* : <https://developer.android.com/reference/java/lang/Comparable>