

TP AMA v4

TP0 : Prise en main de Android Studio, création d'un projet

TP1 : Création d'une interface graphique simple pour saisir les coordonnées d'une personne

TP2 : Gestion des événements, accès aux ressources string

TP3 : Création d'une nouvelle activité pour contenir une liste. Mise en place du RecyclerView

TP4 : Gestion des Checkboxes et utilisation de ScrollView

TP5 : Navigation entre activités

TP6 : Notifications, permission et timer : notifier d'un anniversaire ce jour

TP7 : Persistance des données dans un fichier, prise de photo

TP8 : Persistance des données dans une base SQLite3

TP9 : Envoi de SMS

TP10 : Accès à un WS pour gérer les données

TP0 : Prise en main de Android Studio, création d'un projet

TP0

TP1 : Création d'une interface graphique simple pour saisir les coordonnées d'une personne

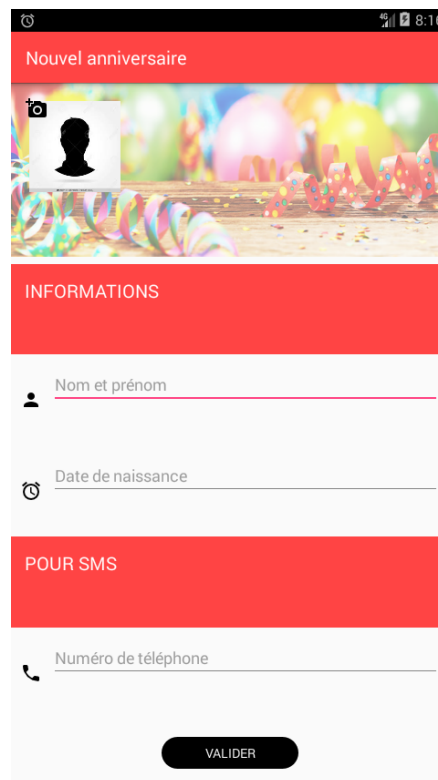
Objectif

Dans cette étape, vous créez une interface graphique simple pour saisir :

- le prénom et le nom de la personne
- le numéro de téléphone
- la date de naissance

Le haut de l'écran sera occupé par une image (sera la photo de la personne)

Résultat attendu



Consignes

Créer un projet avec le SDK minimum 19 (compatible avec Wiko Lenny)

Dans le Designer, choisir API19 et 5" FWVGA

Créer une activity avec un modèle EmptyActivity. Nommer cette activité CreerPersonneActivity.

Vérifier dans le manifest la présence de la déclaration de l'activité.

Vérifier la méthode onCreate de l'activité : elle doit contenir un appel à setContentView avec le nom du fichier layout.

Modifier le fichier de layout. Changer le conteneur ConstraintLayout en LinearLayout. Attention, le nom du package ne doit pas être conservé.

Ajouter les vues :

- ImageView permet d'afficher une image
- TextView est un texte fixe
- EditText est un champ éditable
- Button est un bouton

L'image en haut de fenêtre telle qu'elle est prévue est obtenue par :



Il faut donc créer les 3 drawable prévus. Utilisez l'image que vous voulez pour la bannière et la silhouette. L'icône est fournie par Android Studio.

Le « poids » layout_weight permet de répartir l'espace entre composants. Par exemple si 2 composants enfants ont un poids de 1, alors ils prennent chacun la moitié du parent. Si le premier a un poids de 2 et le second de 1, alors le premier prend les 2/3 de l'espace du parent, etc

Pour obtenir sur la même ligne une icône et un champ texte, il faut introduire un LinearLayout horizontal.

En plus des attributs obligatoires width et height et éventuellement text ou src, utilisez les attributs suivants :

padding : introduit un espace dans le composant, autour de son contenu

layout_margin : introduit un espace autour du composant

→ ces attributs ont des variantes pour top/bottom/start/end

textAllCaps : les caractères sont en majuscules

textColor : couleur du texte

→ des couleurs prédéfinies existent, exemple @android:color/white

textSize : taille de la police, en unité sp

inputType : pour un TextField, propose un clavier adapté (phone, date, etc)

hint : pour un TextField, propose un texte guidant l'utilisateur

background : positionne un fond, couleur ou drawable

layout_gravity : positionne le composant dans son parent LinearLayout

Pour affadir une image (transparence, canal alpha), utiliser l'attribut alpha dans ImageView.

Pour obtenir un fond rond au bouton, créer un fichier drawable qui contient ce genre de description :

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">
  <solid android:color="@android:color/black"/>
  <corners android:radius="25dp"></corners>
</shape>
```

Puis positionner ce « drawable » comme background du bouton.

Tester sur l'émulateur mais aussi sur le téléphone Wiko. Pour utiliser un téléphone et exécuter l'application directement dessus, il faut :

- le connecter au pc
- sur le téléphone, tapoter 7 fois sur le numéro de build (dans les settings)
- revenir un cran en arrière, tapoter sur les options développeur (nouvellement apparues)
- autoriser le débogage USB

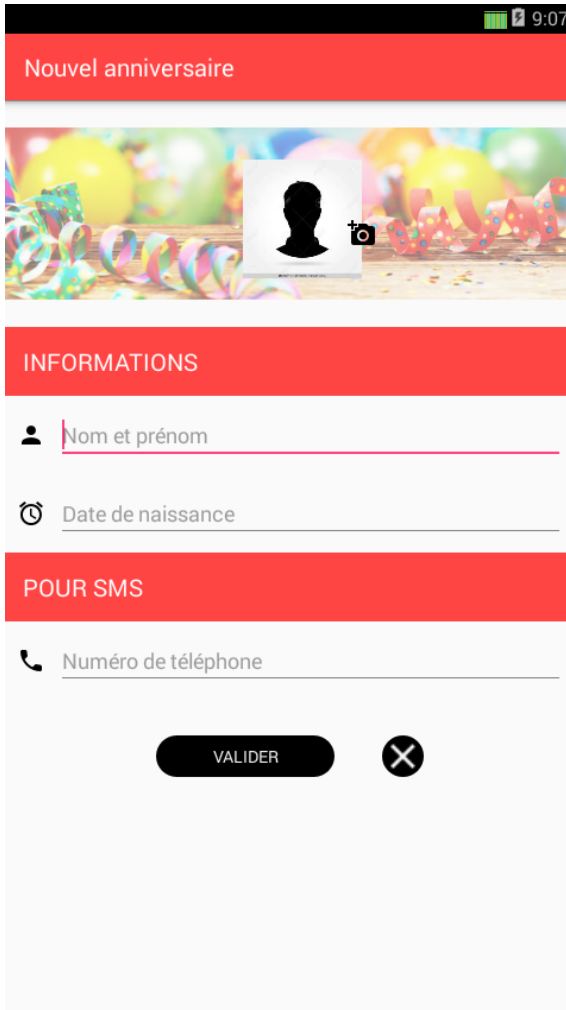
TP2 : strings en ressource avec format, internationalisation

Objectif

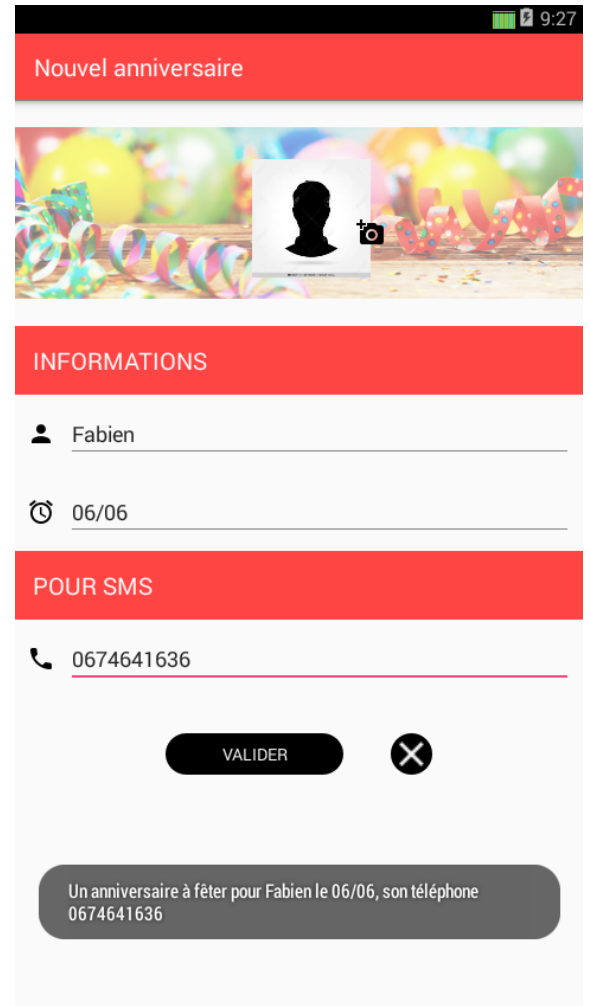
Dans ce TP, vous ajoutez une gestion de l'action sur le bouton de commande *de validation et d'annulation*.

Lorsque l'utilisateur actionne le bouton de validation, on affichera un toast rappelant la saisie. Lorsque l'utilisateur actionne le bouton d'annulation, on remettra les champs à vide.

Résultat attendu



The screenshot shows the 'Nouvel anniversaire' app interface. At the top is a red header with the title 'Nouvel anniversaire'. Below it is a decorative banner with balloons and streamers, featuring a placeholder for a profile picture. The main form has three sections: 'INFORMATIONS' with fields for 'Nom et prénom' and 'Date de naissance', and 'POUR SMS' with a field for 'Numéro de téléphone'. At the bottom are two buttons: 'VALIDER' and a circular button with a cross icon.



The screenshot shows the 'Nouvel anniversaire' app interface after data entry. The 'INFORMATIONS' section now contains 'Fabien' for the name and '06/06' for the date. The 'POUR SMS' section contains '0674641636' for the phone number. A toast message at the bottom reads: 'Un anniversaire à fêter pour Fabien le 06/06, son téléphone 0674641636'. The 'VALIDER' button and the cross icon button are still present.

Consignes

Ajouter un bouton rond comportant une croix. Pour cela, créer un bouton avec un background pointant vers un fichier drawable que vous créez de la façon suivante :

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/bouton_rond"/>
    <item android:drawable="@drawable/ic_clear"/>
</layer-list>
```

Le drawable ic_clear est seulement créé avec les formes prédéfinies, en couleur blanche.

Créer dans MainActivity une méthode valider () et une méthode remiseAZero() que vous appelez en branchant des listener sur les boutons resp. valider et annuler.

```
Button btnAnnuler = findViewById(R.id.main_activity_annuler);
btnAnnuler.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        remiseAZero();
    }
});
```

Dans la méthode valider(), faites apparaître un Toast avec le code suivant :

```
String message = getString(R.string.chaine_message, nom, date, telephone);
Toast.makeText(context: MainActivity.this, message, Toast.LENGTH_LONG).show();
```

La chaîne de caractère de format contient «%s » pour chaque paramètre attendu.

Les chaînes de caractères devraient être toutes en ressource.

Créer un autre fichier strings.xml, avec une « locale » différente, et y copier coller les chaînes de caractères, puis les traduire.

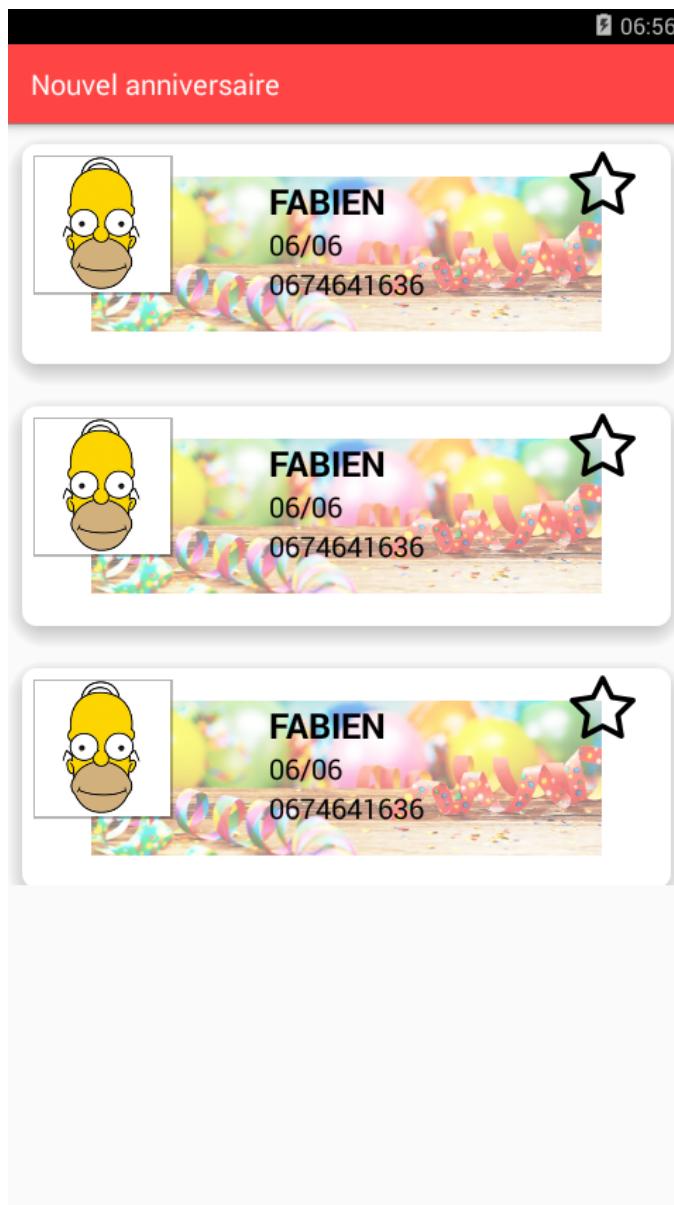
Changer la langue du téléphone et constater.

TP3 : manipulation d'une liste

Objectif

Dans ce TP, vous ajoutez une nouvelle Activity constituée d'une liste (scrollable) d'anniversaires. Chaque élément de la liste sera constitué des éléments tels que saisis précédemment, avec en plus une case à cocher pour sélectionner les alarmes.

Résultat attendu

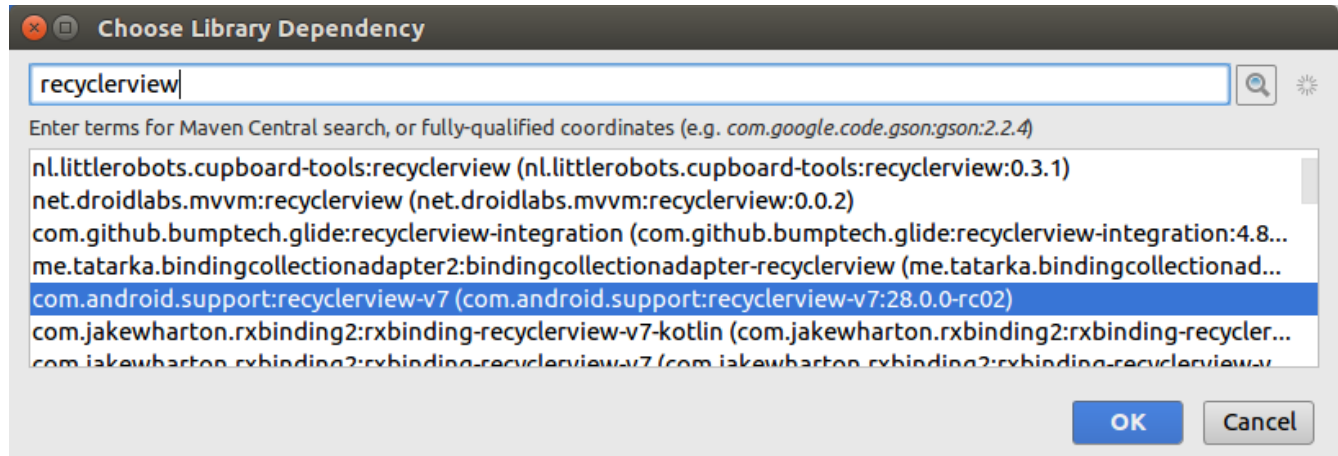


Consignes

Créez une activité nommée ListAnniversairesActivity. Conservez le ConstraintLayout comme layout principal.

Placer un RecyclerView dans le fichier layout xml correspondant. Pour cela, il faut ajouter la bibliothèque correspondante dans le fichier Gradle du module.

Sur le module, ouvrez dans le menu contextuel « Open Module Settings ». Puis choisissez l'onglet « dependencies ». Puis « + », puis « Library dependencies ». Tapez « recyclerview » et sélectionnez :



Validez toutes les boîtes, les scripts Gradle vont être synchronisés. Puis vérifiez dans le script Gradle du module que la bibliothèque est bien référencée dans la zone des dépendances.

Paramétrez en taille et positionnez le RecyclerView comme vous le souhaitez.

Placez en dessous du RecyclerView un ScrollView contenant un TextView. Ce champ texte sera rempli plus tard.

Créer un autre fichier layout xml pour la description des items.

Ce fichier de layout doit pouvoir afficher le nom, la date d'anniversaire, le téléphone et la photo. Pour cela, utilisez un CardView. Cette classe est accessible en utilisant la bibliothèque de même nom. Faites la même manipulation que précédemment, mais en cherchant « cardview ».

Pour obtenir l'item ci-dessus, il a été utilisé le layout suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    card_view:cardCornerRadius="10dp"
    card_view:cardElevation="10dp">

    <FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="150dp">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_margin="20dp"
            android:src="@drawable/banniere"
            android:alpha="0.5"/>

        <ImageView
            android:id="@+id/list_activity_item_photo"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:paddingRight="3dp"
            android:paddingBottom="3dp"
            android:paddingLeft="1dp"
            android:paddingTop="1dp"
            android:layout_margin="5dp"
            android:src="@drawable/homer"
            android:background="@drawable/bordure"
            />

        <LinearLayout...>
        <CheckBox
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="end"
            android:layout_marginEnd="20dp"
            android:button="@drawable/selected_checkbox"/>
    </FrameLayout>
</android.support.v7.widget.CardView>
```

Le drawable banniere est une image.

Le drawable bordure est défini comme ceci :

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:bottom="1dp" android:right="1dp">
        <shape android:shape="rectangle" >
            <solid android:color="#43000000" />
        </shape>
    </item>
    <item android:bottom="2dp" android:right="2dp">
        <shape android:shape="rectangle" >
            <solid android:color="#10000000" />
        </shape>
    </item>
</layer-list>
```

Et le drawable selected_checkbox est défini comme ceci :

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_checked="false" android:drawable="@drawable/empty_star" />
    <item android:state_checked="true" android:drawable="@drawable/filled_star" />
</selector>
```

Les drawable empty_star et filled_star sont des images png.

Vous avez tout loisir de modifier le layout et le rendu.

Puis dans le LinearLayout ... :

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:orientation="vertical"
>

    <TextView
        android:id="@+id/list_activity_item_nom"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAllCaps="true"
        android:textSize="25sp"
        android:textColor="@android:color/black"
        android:textStyle="bold"/>

    <TextView
        android:id="@+id/list_activity_item_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAllCaps="true"
        android:textSize="20sp"
        android:textColor="@android:color/black"/>

    <TextView
        android:id="@+id/list_activity_item_telephone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAllCaps="true"
        android:textSize="20sp"
        android:textColor="@android:color/black"/>

</LinearLayout>
```

Créer une classe Anniversaire, comportant les attributs nom, date, téléphone et photo.

Créer l'adaptateur qui permet à partir d'une ArrayList<Anniversaire> de créer les items de la liste.

Il doit normalement hériter de RecyclerView.Adapter<T>.

Créez une classe héritant de RecyclerView.ViewHolder, interne à la précédente. Cette classe comporte typiquement : les attributs pointant vers les View (TextView pour nom, date et téléphone, ImageView pour la photo), un constructeur pour affecter ces attributs.

Dans l'adaptateur, créer les 3 méthodes imposées par le compilateur et ajouter un constructeur pour passer en paramètre l'activité (nécessaire pour accéder au LayoutInflater) et les données. La classe

d'adaptateur doit conserver l'inflater et les données. Puis les méthodes onCreateViewHolder et onBindViewHolder doivent respectivement créer la View avec le inflater et positionner les données sur les champs du ViewHolder.

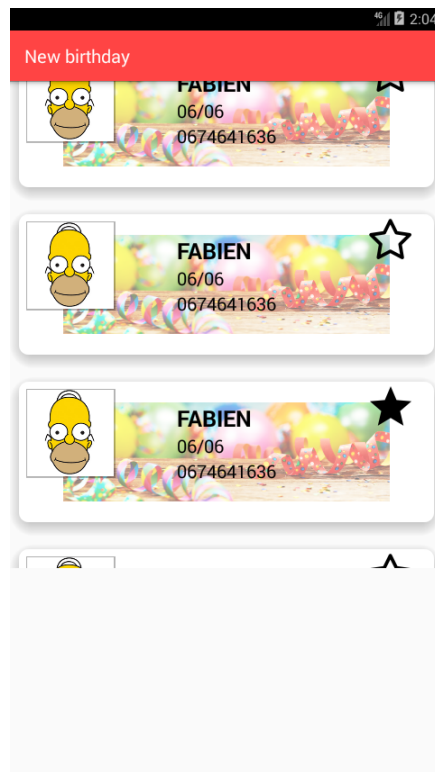
REMARQUE : tapotez sur une checkbox, puis scrollez. Vous devez remarquer que les cases cochées se répètent. Un bug ?

TP4 : Gestion des Checkbox

Objectif

Dans ce TP, vous ajoutez la possibilité de récupérer les actions sur les cases à cocher de la liste des personnes.

Résultat attendu



Consignes

1 Pour récupérer les clics sur les CheckBox

Modifier le layout de l'item pour donner un id à la CheckBox.

Modifier la classe Anniversaires Adapter pour ajouter un listener sur la checkbox :

```

vh.checkBox.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // traitement à déclencher sur le clic
    }
});

```

2 Ajouter un tableau pour maintenir la connaissance sur l'état des CheckBox

```

private final ArrayList<Anniversaire> data,
private boolean[] checkBoxState;

public AnniversairesAdapter(Activity cxt, ArrayList<Anniversaire> data, boolean[] checkBoxState) {
    this.data = data;
    this.inflater = cxt.getLayoutInflater();
    this.checkBoxState = checkBoxState;
}

```

Le tableau checkBoxState sera (par exemple) passé en paramètre de construction de l'adaptateur. Il est modifié lors du clic sur la checkbox correspondante.

Rappel Java : les classes internes n'accèdent pas aux variables locales, sauf si elles sont final.

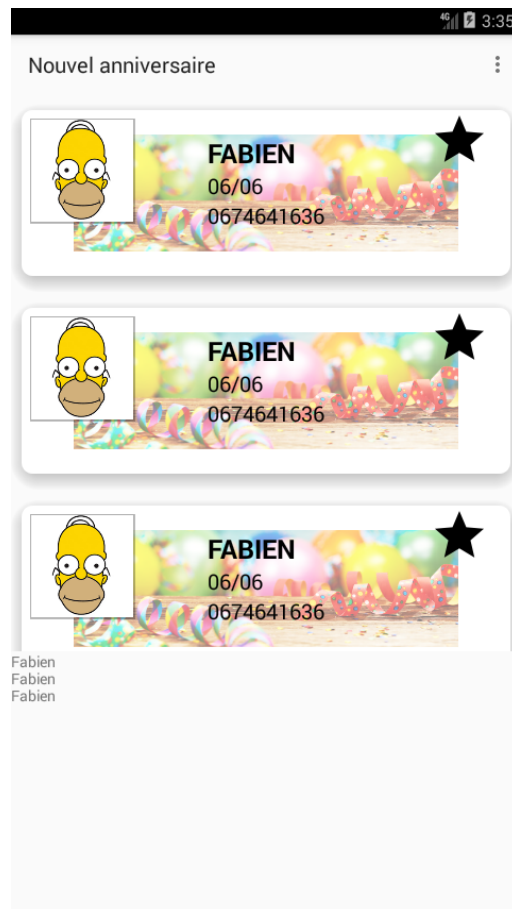
3 Lors du BindView, positionner avec setChecked la valeur trouvée dans le tableau à la position i.

TP5 : Navigation entre les activités

Objectif

Dans ce TP, vous ajoutez la possibilité d'enrichir la liste des personnes. Autrement dit créer une personne, puis l'ajouter dans la liste des personnes à rappeler. Il faut mixer les 2 activités. Il va être intéressant aussi d'utiliser un menu.

Résultat attendu



Consignes

- 1 Créer un menu pour remplacer le bouton « Souhaiter les anniversaires ».
- Créer un répertoire « menu » dans le « res ».
- Créer un fichier de layout dans le répertoire « menu », et y placer les items :

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_souhaiter" android:title="Souhaiter"/>
    <item android:id="@+id/menu_nouveau" android:title="Nouveau"/>
</menu>
```

Créer la méthode de chargement du menu dans l'activité :

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.principal, menu);
    return true;
}
```

Vérifier le thème utilisé par votre application. La barre d'action peut ne pas apparaître.

Nous allons donc utiliser une Toolbar :

1) choisir un thème SANS actionbar

```
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@android:color/holo_red_light</item>
```

2) créer sa propre toolbar dans le layout

```
<android.support.v7.widget.Toolbar
    android:id="@+id/list_activity_menu"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent">
</android.support.v7.widget.Toolbar>
```

La toolbar est placée en haut du layout, modifiez en conséquence les autres éléments.

3) paramétrer la toolbar dans le OnCreate de l'activité


```

        Toolbar menu = findViewById(R.id.list_activity_menu);
        setSupportActionBar(menu);
    }

```

Créer une méthode pour traiter le clic dans le menu :

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if(item.getItemId() == R.id.menu_nouveau) {
        //ouvrir l'activité pour saisir un nouvel anniversaire
    }
    else if(item.getItemId() == R.id.menu_souhaiter) {
        // alimenter un textview placé en bas de l'écran
    }
    return super.onOptionsItemSelected(item);
}

```

Le premier item doit ouvrir l'activité qui permet de créer un nouvel anniversaire. L'activité fille doit alors retourner un anniversaire, qui sera récupéré dans l'activité mère dans la méthode :

```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent payload) {
    if(requestCode == CODE_CREATION_ANNIV) {
        if(resultCode == RESULT_OK) {
            Anniversaire anniv = payload.getParcelableExtra( name: "anniv");
            data.add(anniv);
        }
    }
}

```

Le second item alimente le textview (placé dans un scrollview) décrit en bas de l'écran de liste des anniversaires :

```

<ScrollView
    android:id="@+id/list_activity_scroll"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@id/list_activity_guideline">
    <TextView
        android:id="@+id/list_activity_selectionnes"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    />
</ScrollView>

```

Pour que l'activité de création d'un anniversaire soit bien considérée comme une activité fille, ne pas oublier de spécifier cette parenté dans le manifest.

Etant donné que vous avez utilisé un thème sans ActionBar, et une Toolbar, il faut écrire ceci dans le onCreate de l'activité fille :

```

Toolbar menu =findViewById(R.id.list_activity_menu);
setSupportActionBar(menu);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

```

Cela reproduit le comportement qui existe par défaut : une flèche en haut à gauche, vers la gauche, pour revenir à l'activité parent.

Bonus : en cliquant sur l'icône appareil photo, l'utilisateur déclenche la prise de photo, qui est ensuite récupérée dans l'image superposée.

Pour déclencher la prise de photo :

```

Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
if(intent.resolveActivity(getPackageManager()) != null) {
    startActivityForResult(intent, CODE_IMAGE);
}

```

Pour positionner l'image récupérer sur l'ImageView, implémenter la méthode :

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if(requestCode == CODE_IMAGE) {
        if(resultCode == RESULT_OK) {
            Bundle extras = data.getExtras();
            Bitmap bm = (Bitmap) extras.get(key: "data");
            ((ImageView)findViewById(R.id.main_activity_photo)).setImageBitmap(bm);
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

TP6 : avec notifications façon Android 8.

Objectif

Dans ce TP, vous ajoutez une notification qui permettra à l'utilisateur de savoir qu'il a des anniversaires à rappeler. Lors du boot du système, un timer sera lancé, qui fera les notifications.

Résultat attendu

Consignes

1 Il faut modifier le fichier AndroidManifest.xml pour déclarer :

- Un BroadcastReceiver qui implémente la méthode onReceive qui est déclenchée lorsque l'alarme se déclenche
- Un BroadcastReceiver qui réagit lors du boot du système.
- Ne pas oublier de déclarer la permission pour cette application d'être appelée au boot.

Permission :

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

Déclaration du receiver alarme :

```
<receiver android:name=".AlarmReceiver"/>
```

Déclaration du receiver de boot :

```
<receiver android:name=".DeviceBootReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

Il faut créer les classes AlarmReceiver et DeviceBootReceiver.

Pour la classe DeviceBootReceiver, implémenter la méthode onReceive avec un toast pour avertir l'utilisateur que l'alarme va être enclenchée, puis :

```
Intent chrono = new Intent(context, AlarmReceiver.class);
PendingIntent pi = PendingIntent.getBroadcast(context, CODE_ALARM, chrono, 0);
AlarmManager am = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
am.setRepeating(AlarmManager.RTC_WAKEUP, System.currentTimeMillis(), 1000*10, pi);
```

Pour la classe AlarmReceiver, implémenter onReceive pour notifier l'utilisateur :

- Récupérer le service de notification
-

```
NotificationManager mNotif =  
    (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
```

- Créer Intent et PendingIntent
-

```
Intent ala = new Intent(context, MainActivity.class);  
ala.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP | Intent.FLAG_ACTIVITY_CLEAR_TOP);  
PendingIntent pi = PendingIntent.getActivity(context, requestCode: 0, ala, PendingIntent.FLAG_UPDATE_CURRENT);
```

- Lancer la notification

```
String ident = "utile pour API 27";  
  
if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {  
    CharSequence channelName = "Ce code marche pour API >= 27";  
    int importance = NotificationManager.IMPORTANCE_LOW;  
    NotificationChannel notificationChannel = new NotificationChannel(ident, channelName, importance);  
    mNotif.createNotificationChannel(notificationChannel);  
}  
  
NotificationCompat.Builder bld = new NotificationCompat.Builder(context, ident)  
    .setSmallIcon(R.drawable.ic_notif)  
    .setContentTitle("Attention !")  
    .setContentText("Il faut fêter des anniversaires")  
    .setAutoCancel(true)  
    .setContentIntent(pi);  
  
Notification note = bld.build();  
mNotif.notify(id: 1, note);
```

Dans ce code, la partie spécifique aux SDK supérieurs ou égaux à Android 8 (VERSION_CODES.O) correspond à la création d'un canal de notifications. L'utilisateur peut bloquer certains canaux, sans tout bloquer, depuis cette version.

Tester dans l'émulateur ou avec le mobile connecté. Pour redémarrer l'émulateur, dans le terminal, taper : adb reboot.

Remarques sur adb :

pour obtenir les matériels connectés :	adb devices
pour sélectionner le reboot d'un matériel :	adb -s <nom-du-device> reboot

TP7 : Persistence des données dans un fichier

Objectif

Dans ce TP, vous enregistrez les contacts dans un fichier. La liste représentera le contenu du fichier. Dans la barre d'outil, il faudra créer un menu pour accéder aux préférences, pour envoyer un message à la personne.

Résultat attendu

Consignes

Enregistrement dans un fichier anniv.txt

Créer une classe HistoryFile, avec un constructeur qui attend un Context et qui garde cette référence.

La classe contient aussi 3 méthodes :

```
public boolean isFileExist() {...}  
  
List<String> getAnniversaires() throws Exception {...}  
  
public void saveAnniversaires(String personne) throws IOException {...}
```

La méthode isFileExists teste la présence du fichier avec :

```
mAnnivFile = new File(mContext.getFilesDir(), "anniversaires.txt");
```

La méthode getAnniversaires() charge les contacts du fichier :

```
List<String> getAnniversaires() throws Exception {  
    List<String> retval = new ArrayList<>();  
    try (FileReader fr = new FileReader(mAnnivFile);  
         BufferedReader br = new BufferedReader(fr)) {  
        String ligne;  
        while((ligne = br.readLine()) != null) {  
            retval.add(ligne);  
        }  
    }  
    return retval;  
}
```

La méthode saveAnniversaires ajoute un contact au fichier :

```
if(!isFileExist())  
    mAnnivFile.createNewFile();  
  
try (BufferedWriter bw = new BufferedWriter(new FileWriter(mAnnivFile,true))) {  
    bw.write(personne);  
}
```

Modifier l'activité principale pour ajouter une ligne au retour de l'activité de création, et d'utiliser le fichier pour charger les anniversaires.

TP8 : Persistence des données dans une base SQLite3

Objectif

Dans ce TP, vous enregistrez dans la base de données l'ensemble des appels par sms effectués depuis Android. Au lancement de l'appli, la liste inférieure affichera tous les appels effectués. Cette liste sera effacée lors de l'action du bouton.

Résultat attendu

Consignes

1 Créer une class AnnivsStructureDB qui hérite de SQLiteOpenHelper. La table aura une structure déduite :

```
public static final String TABLE_ANNIVS = "table_anniversaires";
public static final String COL_ID = "_id";
public static final int NUM_COL_ID = 0;

public static final String COL_NOM = "nom";
public static final int NUM_COL_NOM = 1;

public static final String COL_PRENOM = "prenom";
public static final int NUM_COL_PRENOM = 2;

public static final String COL_TELEPHONE = "telephone";
public static final int NUM_COL_TELEPHONE = 4;

public static final String COL_DATE = "date";
public static final int NUM_COL_DATE = 3;
```

Implémenter le constructeur, onCreate et onUpgrade.

2 Ajouter un id à la classe Anniversaire

3 Créer une classe AnniversaireCrud

Cette classe doit implémenter la création et la récupération des anniversaires:

```
public AnniversaireCrud(Context cxt) { mAnniversaires = new AnnivsStructureDB(cxt,NOM_BDD,null,1);}

public void openForWrite() { bdd = mAnniversaires.getWritableDatabase();}
public void openForRead() { bdd = mAnniversaires.getReadableDatabase();}

public void close() { mAnniversaires.close();}

public long insertAnniversaires(Anniversaire anniv) {...}

public ArrayList<Anniversaire> getAllAnniversaires() {...}
```

La base de données est nommée anniversaires.

4 Faire l'enregistrement et la lecture dans la base de données

Tester dans l'émulateur ou avec le mobile connecté.

TP9 : Envoi de SMS

Objectif

Dans ce TP, vous envoyez des SMS aux personnes sélectionnées.

Résultat attendu

Consignes

- 1 Créer une classe qui se charge d'envoyer les sms. Il faudra aussi se mettre en écoute d'un BroadcastReceiver, pour vérifier que le sms est bien arrivé.
Nommer cette classe SmsSending, déclarer un attribut mContext, un autre mReceiver de type BroadcastReceiver.
Créer le constructeur qui garde le contexte.
Créer une méthode nommée sendSMS, attendant un String pour le téléphone et un String pour le message.
Déclarer un String nommé DELIVERED, contenant « SMS_DELIVERED ».
Déclarer et initialiser un PendingIntent, utilisé pour l'envoi du sms.
Enregistrez un BroadcastReceiver :

```
context.registerReceiver(new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
  
        switch(getResultCode()) {  
            case Activity.RESULT_OK :  
                Toast.makeText(context, "Message envoyé", Toast.LENGTH_LONG).show();  
                break;  
        }  
    }  
}, new IntentFilter(DELIVERED));
```

Pour trouver le smsmanager , utiliser SmsManager.getDefault()

Pour envoyer le sms , utiliser sms .sendTextMessage(tel,null, msg, deliveredPI)

- 2 Créer une classe pour lancer les sms en asynchrone

Cette classe hérite de AsyncTask<Void,Void,Void>.

Elle possède la collection gérée par la classe principale et le tableau des checkbox. Elle garde aussi une référence vers l'Activity.

La méthode doInBackground doit être implémentée Void doInBackground(Void ... params) de la façon suivante par exemple :

Sur l'activité principale, on appelle `activity . updateResultsTextFromSms(sendSms())`. Ces méthodes doivent être développées.

3 La méthode `sendSms()` retourne la liste de messages contenant les envois de sms réalisés. Dans cette méthode, il faut instancier `SmsSending`, puis appeler pour chaque contact la méthode `sendSMS`. On ne prend que les checkbox sélectionnées et on retourne une collection de messages (`ArrayList<String>`).

4 La méthode `updateResultsFromSms` de l'activité principale est implémentée de façon à mettre à jour l'interface graphique mais provient d'un appel asynchrone... elle doit donc être implémentée en utilisant :

```
runOnUiThread(new Runnable() {  
    @Override  
    public void run() {  
        if (messageenvoi.size() == 0) {
```

Dans la méthode `run()`, il faut changer la `TextView` des noms rappelés à partir de la collection passée en paramètres, ou un message pour signaler qu'aucun contact n'a été sélectionné.

5 La méthode `sendSMS` doit être implémentée en instanciant la classe dérivant de `AsyncTask` et en appelant `execute()`

Tester dans l'émulateur ou avec le mobile connecté.

6 Gestion des permissions

Ne pas oublier de déclarer les permissions.

Pour se conformer aux possibilités de Android6, il est recommandé de demander au dernier moment la permission.

Utiliser le code suivant pour modifier `MainActivity`.

```

private static final int MY_PERMISSIONS_REQUEST_SEND_SMS =0 ;

public void tenterSMS() {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.SEND_SMS)
        != PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.SEND_SMS)) {
            //inutile ici
        } else {
            ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.SEND_SMS},
                MY_PERMISSIONS_REQUEST_SEND_SMS);
        }
    } else {
        envoyerSMS();
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_SEND_SMS: {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                envoyerSMS();
            } else {
                Toast.makeText(this, "Echec de l'envoi du SMS, non autorisé", Toast.LENGTH_LONG).show();
                return;
            }
        }
    }
}
}

```

TP10 : Accès à un WS pour gérer les données

Objectif

Les données seront envoyées sur un serveur via un Webservice.

Résultat attendu

Consignes

- 1) Mettre en place le serveur.
- 2) Ajouter la bibliothèque Volley dans les dépendances

Volley fournit les outils pour accéder à internet :

```
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

- 3) Créer une classe pour gérer les appels

```
public class DataProvider {

    private static String TAG = "WSDATA";

    private Context mContext;
    private RequestQueue mRequestQueue;
    private String mUrlPost;
    private String mUrlGet;
```

Préparer un constructeur qui permet de conserver les infos de manière asynchrone, ce constructeur sera appelé lors du onCreate.

Créer une interface qui sera implémentée par MainActivity et appelée par le DataProvider pour signaler le retour serveur ou bien l'erreur.

Ajouter une méthode pour récupérer les données

```
JSONArrayRequest request = new JSONArrayRequest(Request.Method.GET, mUrlGet, null,
    new Response.Listener<JSONArray>() {
        @Override
        public void onResponse(JSONArray tab) {
            HashMap<String, String> map = new HashMap<>();
            try {
                for (int i=0; i<tab.length(); i++) {
                    JSONObject response = tab.getJSONObject(i);
                    map.put("nom", response.get("nom").toString());
                    map.put("prenom", response.get("prenom").toString());
                    map.put("date", response.get("date").toString());
                    map.put("telephone", response.get("telephone").toString());
                    ((DataResponseListener) mContext).response(map);
                }
            } catch (JSONException e) {
```

Ajouter une méthode pour ajouter une donnée, passant en paramètre l'url et la map stockée dans le fichier.

```
JSONObjectRequest request = new JSONObjectRequest(Request.Method.POST, mUrlPost, new JSONObject(donnees),
    new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            ((DataResponseListener) mContext).response("ok");
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            ((DataResponseListener) mContext).response(error.toString());
        }
    }
));
```

Penser à déclarer la permission.

Vérifier le fonctionnement de l'appel par les traces fournies par nodejs.

A ajouter dans les slides :

- diagramme d'états de l'activité, avec description de
 - quand est appelée une méthode : onStart pour un retour à la vie