

DÉVELOPPEMENT D'APPLICATIONS MOBILES



Alex Morel - Code Lutin

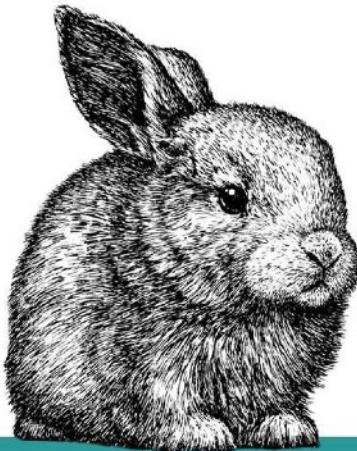
 @alex_morel_



 **Code
Lutin**

DISCLAIMER

Feigning knowledge of a word you've heard a few times



Expert

Pretending to Know
About Stuff

O RLY?

@ThePracticalDev

APPS

Le but du module n'est pas :

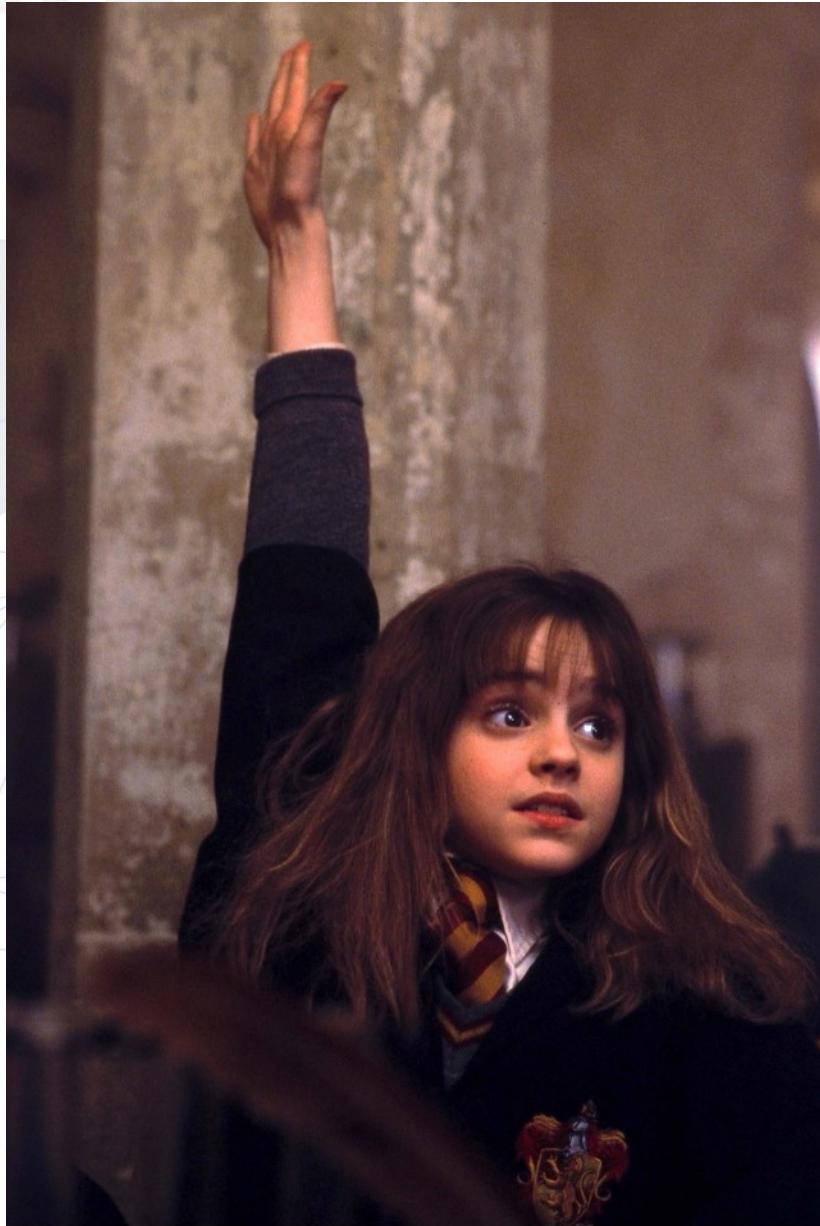
- d'apprendre les frameworks les plus récents (Kotlin/AAC)
- de vous affirmer « Expert Android » sur LinkedIn

Le but du module c'est :

- comprendre les problématiques propres au dev mobile
- maîtriser les concepts permettant de résoudre et des frameworks simples (en native et en hybride)
- pouvoir postuler à un poste de développeur Android Junior
- répondre à vos questions (techniques ou sur le monde du travail)

NAZ
AIRE

applications mobiles



Arrêtez-moi quand
vous voulez !

Programme du module :

Tout est sur madoc (j'y mets mes slides)

Le cours c'est aussi approfondir chez vous :

- par mail : lagardealex@gmail.com
- sur discord : <https://discord.gg/4K9gWTt>

Ce module a pour but de permettre à l'étudiant d'appréhender les problématiques propres au développement mobile (fonctionnement hors-ligne, diversité des terminaux, notifications push, géolocalisation...) ainsi que de découvrir les bases de la programmation Android native et hybride.

Le module est divisé en 4 grandes parties

1. **Programmation Native Android (Java)** : comprendre la structure d'une applications native Android, les mécanismes de build (Gradle) et couvrir les besoins fonctionnels de base (listes, cartes, interactions avec des APIs Rest, stockage en base de données via un ORM).
2. **Progressive Web Apps (Vue.js)** : comprendre les bases de Vue.js, et comment implémenter une Progressive Web App (PWA) simple (conception offline et responsive)
3. **Programmation Hybride (Capacitor)** : génération d'applications Android à partir de PWA, utilisation de plugins Capacitor (géolocalisation, accès à l'appareil photo...)
4. **En fil rouge** : bonnes pratiques de debuggage, utilisation de Git, design patterns utiles en programmation mobile

Le TP de fin d'année demandera à l'étudiant de mobiliser ses capacités de conception, d'architecture et de développement pour mettre en oeuvre, à partir d'un cahier des charges, une application mobile répondant au besoin, en utilisant les technologies qui lui semblent adaptées.

Un examen sur table testera la bonne compréhension de l'étudiant des notions présentées dans ce module.

Programme du cours Android natif :

1. Bases Android
2. HTTP, Asynchronicité, Évènements, Google Maps
3. Base de données, Notifications
4. Bonnes pratiques, débugage, architecture

Je peux inclure les sujets de votre choix (prévenez-moi tôt)

**MOB
ILE
APPS**

Programme du jour :

1. Structure d'un projet Android (Rappels)
2. Build d'un projet Android (Gradle)
3. Git - niveau 1 (Local work)
4. Listes & Adapteurs
5. Gestion des resources (Assets & Drawables)

**MOB
ILE
APPS**

1 - Structure d'un projet Android

MOB
ILE
APPS



1 - Structure d'un projet Android

Choose your project

Create New Project

Phone and Tablet Wear OS TV Android Auto Android Things

Add No Activity Basic Activity Empty Activity Bottom Navigation Activity

Empty Activity
Creates a new empty activity

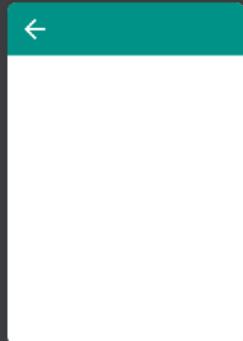
Cancel Previous Next Finish

The screenshot shows the 'Create New Project' wizard in the Android Studio interface. The title bar says 'Create New Project'. Below it, the heading 'Choose your project' is displayed. A navigation bar at the top includes tabs for 'Phone and Tablet' (which is selected), 'Wear OS', 'TV', 'Android Auto', and 'Android Things'. The main area contains four card-like templates: 'Add No Activity' (a simple white screen), 'Basic Activity' (a white screen with a yellow '+' button at the bottom right), 'Empty Activity' (a white screen with a dark blue footer bar containing a back arrow and three dots), and 'Bottom Navigation Activity' (a white screen with a dark blue footer bar containing a back arrow, three dots, and a yellow '+' button). Below these cards, the 'Empty Activity' card is highlighted with a bold title 'Empty Activity' and the subtitle 'Creates a new empty activity'. At the bottom of the screen are buttons for 'Cancel', 'Previous', 'Next', and 'Finish'.



1 - Structure d'un projet Android

Configure your project

Empty Activity
Creates a new empty activity

Name: PlaceSearcher

Package name: org.miage.placesearcher

Save location: /Users/alexmorel/Documents/Git/org.miage.placesearcher 

Language: Java

Minimum API level: API 14: Android 4.0 (IceCreamSandwich) 

 Your app will run on approximately **100%** of devices.
[Help me choose](#)

This project will support instant apps

Use androidx.* artifacts

HTTP Proxy

Host name:

cache.ensinfo.sciences.univ-nantes.prive

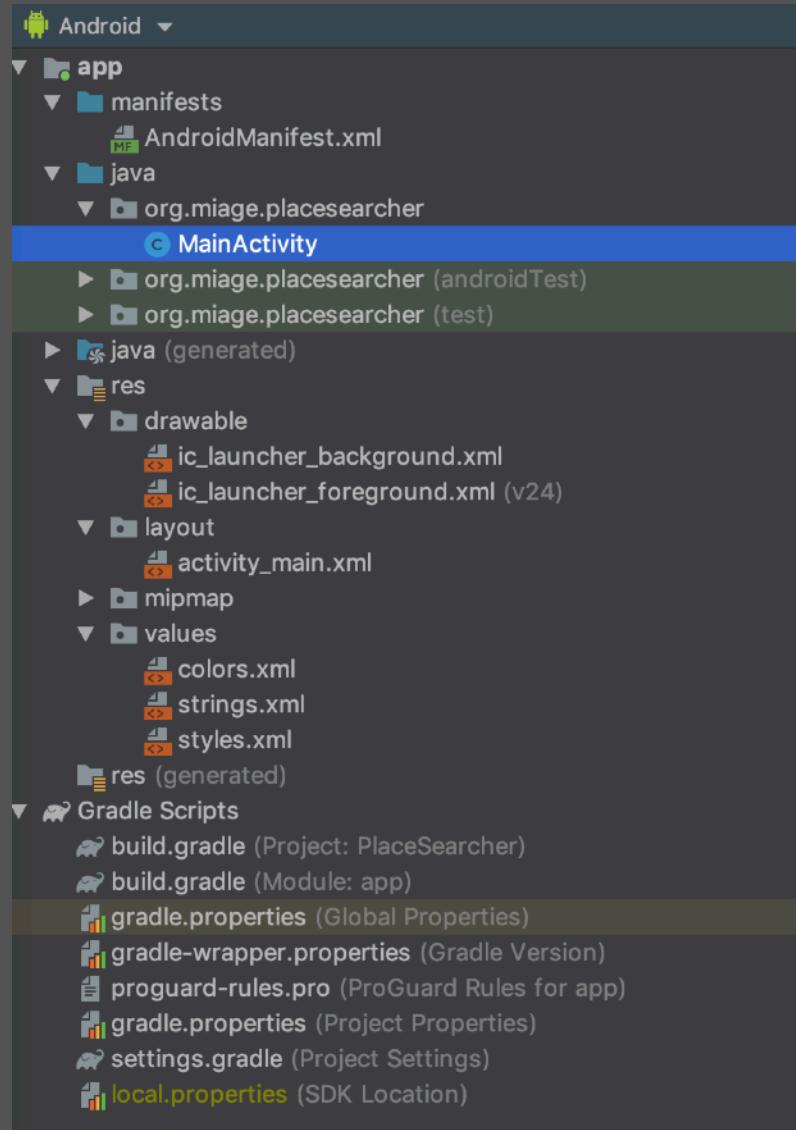
Port number:

3128





1 - Structure d'un projet Android





1 - Structure d'un projet Android

Le Manifest Android : le contrat de votre App (et une source infinie de bugs)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.miage.place searcher">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PlaceSearcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

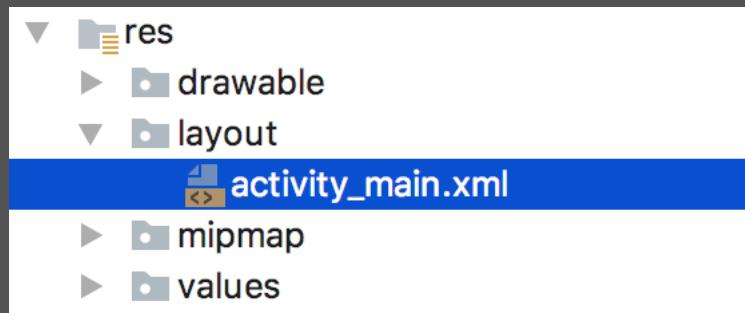
- Liste les permissions nécessaires à l'application
- Déclare l'application (icône, nom...)
- Liste les activités
- Liste les services

Pari n°1 : quelqu'un va oublier de déclarer une activité ou une permission dans le Manifeste



1 - Structure d'un projet Android

Les Ressources : tout ce que votre app embarque au download



- Assets (sons, vidéos, pdf...)
- Drawables (multi-résolution)
- Layouts (vues XML)
- Values : constantes (couleurs, strings...)
- Supporte le multilingue (même le danois)

Tip : faire très attention à ce que vous embarquez, le poids explose très vite



1 - Structure d'un projet Android

Les Activités : les contrôleurs de votre Application

```
package org.miage.placesearcher;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Lien entre :

- Vue (setContentView)
- Interactions utilisateurs (clics, scrolls...)
- Logique applicative



1 - Structure d'un projet Android



Cycle de vie d'une Activité

- A connaitre pour brancher les comportements aux bons moments
 - Impossible de toucher à l'UI si l'application n'est pas dans l'état « running » (asynchrone)
 - Surcharger proprement (appeler la super méthode systématiquement)

Pari n°2 : quelqu'un va oublier d'appeler la super-méthode



1 - Structure d'un projet Android

Les vues : fichiers XML (proche HTML)

Android Studio shortcut : control + clic

The screenshot shows the Android Studio interface. On the left, the XML editor displays the code for `activity_main.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="org.miage.placesearcher.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

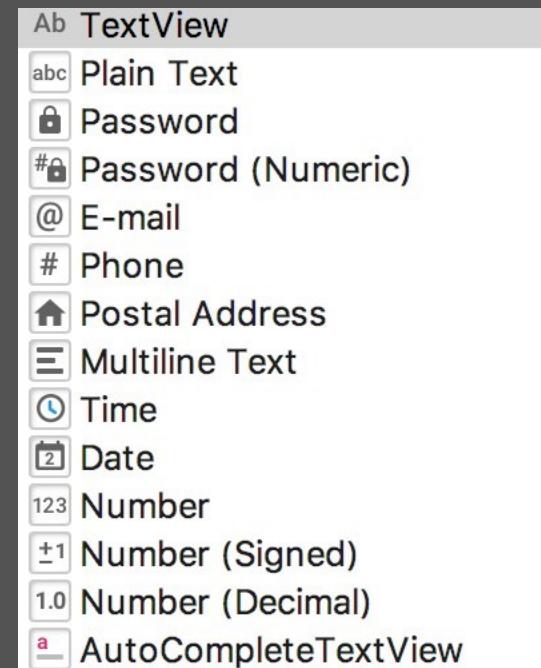
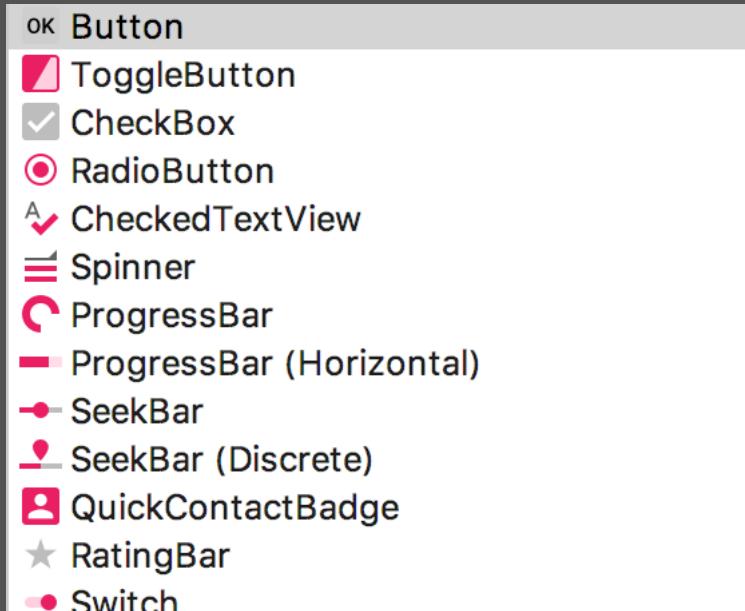
</android.support.constraint.ConstraintLayout>
```

On the right, the preview window shows a Nexus 4 device with the title "PlaceSearcher" and the text "Hello World!" displayed on the screen.



1 - Structure d'un projet Android

Les vues : des widgets natives



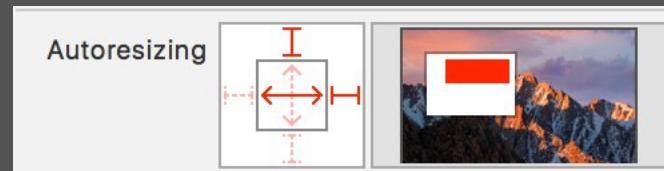
- Possibilité d'installer d'autres vues
(carousel, signature...)
- Possibilité de définir ses propres vues



1 - Structure d'un projet Android

Les vues : responsive

iOS : peu de résolutions différentes



Android : infinité de possibilité





1 - Structure d'un projet Android

Les vues : responsive

2 Grandes Familles de Layout

- Layout structurés :
 - **LinearLayout** (horizontal/vertical, chaque enfant déclare son poids)
 - **GridLayout** (grille)
 - **TableLayout** (tableaux)
- Layouts relatifs :
 - **RelativeLayout** (éléments placés relativement à leur parent/siblings)
 - **ConstraintLayout** (s'approche d'iOS) : à privilégier vs Relative Layout



1 - Structure d'un projet Android

Exercice n° 1

- Lancer le projet d'exemple sur vos téléphones
- Ajouter une RatingBar à votre ConstraintLayout
- La RatingBar doit être située en bas et centrée horizontalement peu importe la résolution
- Récupérer la ratingBar dans votre activité avec findViewById()
- Incrémenter la valeur de la ratingBar à chaque fois que l'activité passe en premier plan

Android Studio shortcuts

- Contrôle + espace
- Alt + Entrée

<https://developer.android.com/training/constraint-layout/index.html>



1 - Structure d'un projet Android

Passer votre téléphone en developer mode

- Paramètres > Etat du téléphone (ou A propos de l'appareil)
- Cliquer 5 fois sur le numero de build
- Puis dans le dévelopeur mode : autoriser le debuggage USB



1 - Structure d'un projet Android

Exercice n°2

- Ajouter un OnClickListener sur la TextView HelloWorld pour décrémenter la RatingBar à chaque clic

Android Studio shortcuts

- Shift Shift Shift

2 - Build Android (gradle)

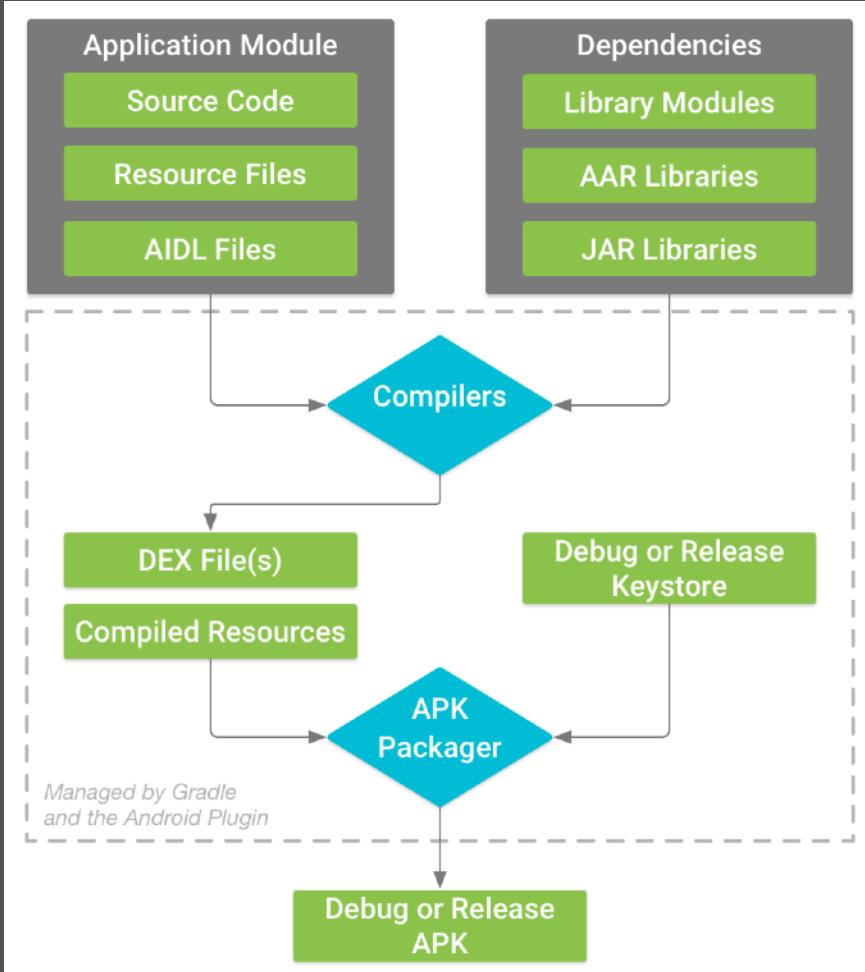


MOB
ILE
APPS

Gradle



2 - Build Android (gradle)



- Déclarer les URL des repos sources
- Déclarer les dépendances du projet
- Déclarer les contraintes de build
 - version Android
 - version de l'app
 - ...
- Définir des règles custom (release...)
- Gradle fait le reste et génère l'APK/AAB



2 - Build Android (gradle)

2 (ou plus) build.gradle (Shift Shift Shift !)

- Le build.gradle «projet» (top-level : sources, configuration générale)
- Un build.gradle pour chaque module (app : dépendances, contraintes)
- En plus du module app, possiblement plusieurs autres (librairies)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "org.miage.placesearcher"
        minSdkVersion 14
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.0.0-beta1'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:0.5'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:2.2.2'
}
```



2 - Build Android (gradle)

Exercice n°3

- Installer le framework ButterKnife sur notre projet

<https://github.com/JakeWharton/butterknife>





Apparté : ButterKnife

- findViewById() et setOnClickListener() sont verbeuses, sauvages (Cast) et pénibles
- ButterKnife est un « sucre syntaxique » basé sur des annotations
- Exécute toutes les annotations à l'appel de ButterKnife.bind()

```
private RatingBar mRatingBar;
private TextView mTextView;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Get the rating bar from layout file
    this.mRatingBar = (RatingBar) this.findViewById(R.id.ratingBar);

    // Get the textView from layout file
    this.mTextView = (TextView) this.findViewById(R.id.textView);

    // Define onClickListener on the text view : decrease rating at each click
    mTextView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            MainActivity.this.mRatingBar.setRating(MainActivity.this.mRatingBar.getRating() - 1);
        }
    });
}
```

```
@BindView(R.id.ratingBar) RatingBar mRatingBar;
```

```
@OnClick(R.id.textView)
public void clickedOnTextField() {
```

Pari n°3 : quelqu'un va oublier d'appeler ButterKnife.bind()



Apparté : ButterKnife

Exercice n°4

- Refactorer la MainActivity pour utiliser ButterKnife (plus de findViewById ni OnClickListener)

<https://github.com/JakeWharton/butterknife>

3 - Git - niveau 1





3 - Git - niveau 1

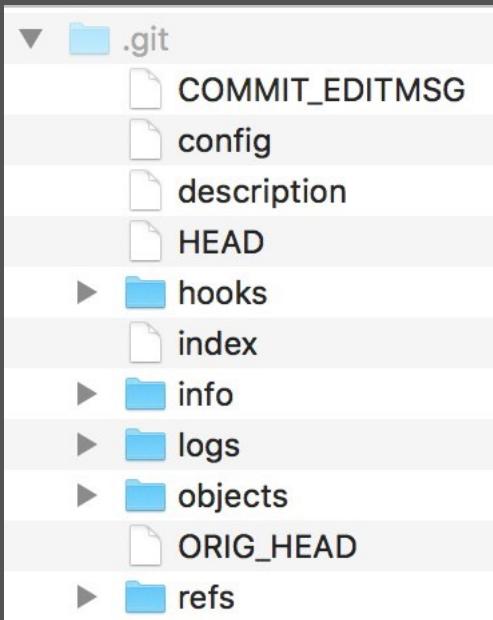
NE TRAVAILLEZ **JAMAIS** SANS REPO GIT

COMMITEZ **PETIT** POUR LES CONFLITS

COMMITEZ **TOUT** LE TEMPS



3 - Git - niveau 1



- Repository Git
 - VCS distribué (pas comme SVN)
 - dossier .git à la racine
 - et c'est tout (pas comme SVN)
- Un arbre de commits
 - Un message
 - Un parent
 - Un patch (ensemble de modifications)
 - Une somme de contrôle SHA-1 (calculée)
- Passage de branche en branche rapide



Torvalds: Méprisable et désagréable ? - Le conflit qui créa GIT

cocadmin • 35 k vues • il y a 2 mois

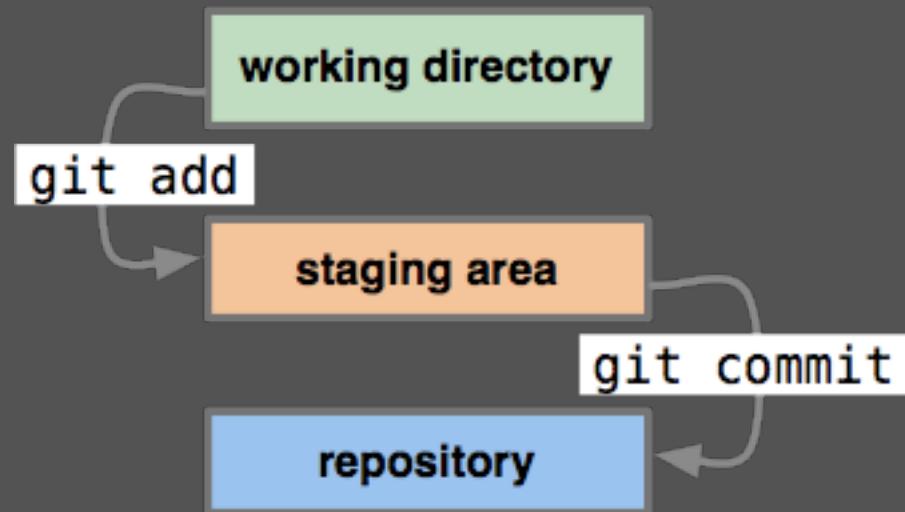
L'histoire du 2eme gros projet de Linus Torvalds : GIT Newsletter: <http://cocadmin.com> Formations: <https://cours.cocadmin.com> ...

21:17



3 - Git - niveau 1

Travailler en local : working, staging, repo





3 - Git - niveau 1

```
@OnClick(R.id.textView)
public void clickedOnTextField() {
    // Decrease rating (i.e. subtract one to value) at each click on TextField
    this.mRatingBar.setRating(this.mRatingBar.getRating() - 1);

    // This is an awesome comment to add on git
}
```

```
alexmorel@~/Documents/Git/org.miage.placesearcher (master $) $ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   app/src/main/java/org/miage/placesearcher/MainActivity.java

no changes added to commit (use "git add" and/or "git commit -a")
alexmorel@~/Documents/Git/org.miage.placesearcher (master *) $
```

```
alexmorel@~/Documents/Git/org.miage.placesearcher (master *) $ git diff
diff --git a/app/src/main/java/org/miage/placesearcher/MainActivity.java b/app/src/main/java/org/miage/placesearcher/MainActivity.java
index 2077014..d56cb18 100644
--- a/app/src/main/java/org/miage/placesearcher/MainActivity.java
+++ b/app/src/main/java/org/miage/placesearcher/MainActivity.java
@@ -32,7 +32,10 @@ public class MainActivity extends AppCompatActivity {
    @OnClick(R.id.textView)                                // This is an awesome comment to add on git
    public void clickedOnTextField() {                      }
        // Decrease rating at each click on TextField
        // Decrease rating (i.e. subtract one to value) at each click on TextField
        this.mRatingBar.setRating(this.mRatingBar.getRating() - 1);

    // This is an awesome comment to add on git
}
```

4: Run TODO 6: Logcat 7: Android Profiler 9: Version Control Terminal 0: Messages



3 - Git - niveau 1

Staged files

Unstaged files

...age/placesearcher/MainActivity.java •••

app/src/main/java/org/miage/placesearcher/MainActivity.java

Hunk 1 : Lines 32-41

```
32 32
33 33     @OnClick(R.id.textView)
34 34     public void clickedOnTextField() {
35 -         // Decrease rating at each click on TextField
35 +         // Decrease rating (i.e. subtract one to value) at each click on TextField
36 36         this.mRatingBar.setRating(this.mRatingBar.getRating() - 1);
37 +
38 +
39 +         // This is an awesome comment to add on git
37 40     }
38 41 }
```



3 - Git - niveau 1

```
[alexmorel@~/Documents/Git/org.miage.placesearcher (master *) $ git add *.java
```

```
[alexmorel@~/Documents/Git/org.miage.placesearcher (master +$) $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   app/src/main/java/org/miage/placesearcher/MainActivity.java
```

```
alexmorel@~/Documents/Git/org.miage.placesearcher (master +$) $
```

The screenshot shows a GitHub desktop application interface. On the left, there's a sidebar with checkboxes for 'Staged files' (checked) and 'Unstaged files' (unchecked). The main area displays a diff view for the file `app/src/main/java/org/miage/placesearcher/MainActivity.java`. The diff shows a single hunk of code with line numbers 32 to 41. The code contains a method `clickedOnTextField` with a comment block. The diff highlights changes in red and green, indicating deletions and additions respectively. A new line of code has been added at the bottom of the hunk.

Hunk 1 : Lines 32-41	
32	32
33	33 @OnClick(R.id.textView)
34	34 public void clickedOnTextField() {
35	35 // Decrease rating at each click on TextField
36	35 // Decrease rating (i.e. subtract one to value) at each click on TextField
37	36 this.mRatingBar.setRating(this.mRatingBar.getRating() - 1);
38	37 }
39	38 // This is an awesome comment to add on git
40	39 }
41	40 }



3 - Git - niveau 1

```
alexmorel@~/Documents/Git/org.miage.placesearcher (master +$) $ git commit -m "[Android] Very important changes to comments that will bring an end to all wars"
```

```
alexmorel@~/Documents/Git/org.miage.placesearcher (master $) $ git status
On branch master
nothing to commit, working tree clean
```

alexmorel@~/Documents/G

Graph	Description	Commit
	master [Android] Very important changes to comments that will bring an end to all wars [Android] Exercice 4 : refactor MainActivity by using ButterKnife [Android] [Build] Exercice 3 : add ButterKnife to build.gradle [Android] Exercice 2 - Define OnClickListener on TextView [Android] Exercice 1 - Add rating bar to default activity and increment its value each time activity is shown [Android] [Build] Initial default project import [Android] [Build] Configure gitignore file - from gitignore.io	e0f3cfb c170159 ee00c08 bb7fd20 f926277 679f973 3edb5b3

LES 3 CHOSES À GARDER EN TÊTE

1. Je fais mes modifs
2. Je git add ce que je veux embarquer dans le commit
3. Je comitte

NOU
VELLES
INTER
FACES



3 - Git - niveau 1

Exercice n°5

- Vous placer dans le dossier du projet Android
- Supprimer le fichier `.gitignore` s'il a été créé automatiquement
- Initialiser un repository git avec la commande `git init`
- Ajouter tout le contenu du dossier (expression régulière `.`)
- Etudier la liste des fichiers prêts à être comités, comprenez-vous le problème ?



3 - Git - niveau 1

.gitignore.io

Create useful .gitignore files for your project

✗ Android

✗ AndroidStudio

Create



3 - Git - niveau 1

```
$ git reset  
$ git add .gitignore  
$ git commit -m "[Android] [Build] Configure .gitignore"
```



3 - Git - niveau 1

Exercice n°5 (le vrai cette fois)

- Ajouter tout le contenu du dossier (expression régulière .)
- Etudier la liste des fichiers prêts à être commités, est-ce acceptable ?
- Commiter
- Ajouter un commentaire, ajouter, vérifier la diff, commiter
- Ajouter un commentaire, ajouter, modifier le commentaire, vérifier la diff

LES 3 CHOSES À GARDER EN TÊTE

1. Je fais mes modifs, j'add, je committe
2. Je pense à toujours commencer par initialiser mon gitignore
3. Je vérifie toujours le contenu de mes commits (auto-revue)
`git init, add, status, commit, reset`

NOU
VELLES
INTER
FACES

4 - Listes et adaptateurs



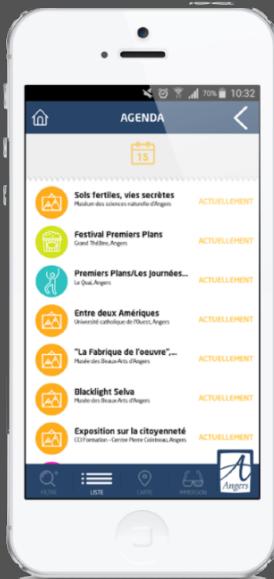


4 - Listes et adaptateurs

View dans le layout (xml)

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/listView"  
    android:layout_width="0dp"  
    android:layout_height="0dp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

RecyclerView : permet d'afficher une liste d'objets Java sous forme graphique



Chaque éléments de la liste va être rendu sous forme d'Item

Par rapport aux ListView, GridView...

- Force à implémenter les bonnes pratiques (e.g. Pattern ViewHolder)
- Plus facile à animer
- Bien plus customizables (organisation des items entre eux notamment)



4 - Listes et adaptateurs

View dans le layout (xml)

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/listView"  
    android:layout_width="0dp"  
    android:layout_height="0dp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

Associée à un Adapter (pour lier chaque élément de la liste à un item)

Associée à un LayoutManager (pour positionner les items)

Associée à un ItemAnimator (e.g. animations d'ajouts/suppressions)



4 - Listes et adapteurs

Associée à un Adapter (pour lier chaque élément de la liste à un item)

Rôle de l'adapter: afficher les éléments de la liste sous forme d'Item

Méthode « naïve » : je créé un Item par élément visible de ma liste

- Pour une liste de 10k éléments
- 10 éléments visibles au départ -> 10 items créés (OK)
- Je scrolle jusqu'en bas, on a créé 10k items (KO)





4 - Listes et adaptateurs

Associée à un Adapter (pour lier chaque élément de la liste à un item)

Rôle de l'adapter: afficher les éléments de la liste sous forme d'Item

Méthode « finaude » : le pattern **ViewHolder**

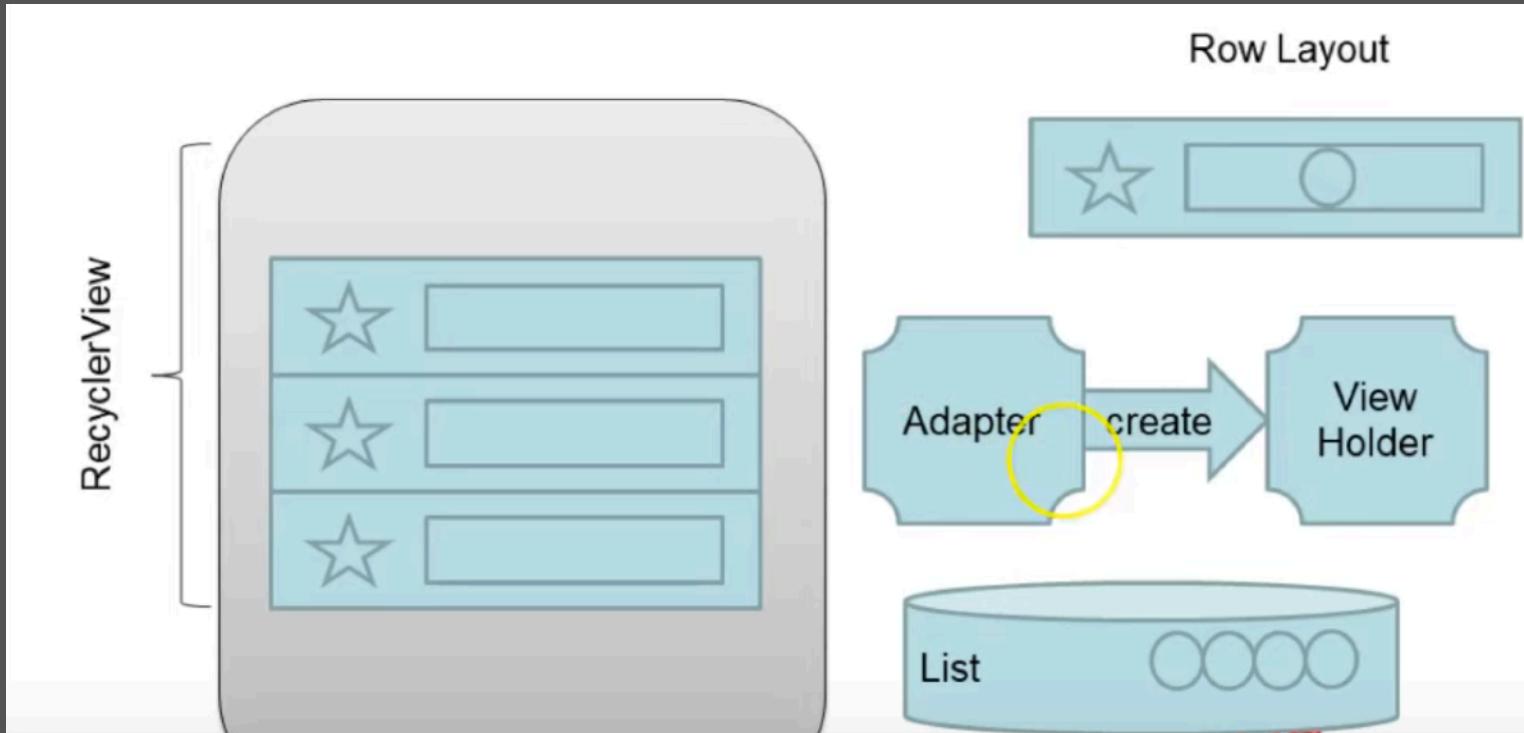
Quand je scroll dans ma liste, au lieu de créer de nouveaux items

Je réutilise les items existants, je change juste les valeurs (des images, des textViews...)





4 - Listes et adaptateurs



<https://youtu.be/uh6lKnfp5hY>



4 - Listes et adaptateurs

Associée à un Adapter (pour lier chaque élément de la liste à un item)

Méthode « finaude » : le pattern **ViewHolder**

Un ViewHolder doit stocker tous les champs qui peuvent « bouger » d'un item à l'autre

```
// String Adapter : convert string list into Items
public class StringAdapter extends RecyclerView.Adapter<StringAdapter.StringViewHolder>

    // Pattern ViewHolder
    class StringViewHolder extends RecyclerView.ViewHolder
    {
        TextView text1;

        public StringViewHolder(View itemView) {
            super(itemView);
            text1 = itemView.findViewById(android.R.id.text1);
        }
    }
}
```



4 - Listes et adaptateurs

Associée à un Adapter (pour lier chaque élément de la liste à un item)

Méthode « finaude » : le pattern **ViewHolder**

```
// String Adapter : convert string list into Items
public class StringAdapter extends RecyclerView.Adapter<StringAdapter.StringViewHolder>{

    private LayoutInflater inflater;
    private Context context;
    private List<String> mStrings;

    public StringAdapter(Context context, List<String> strings) {
        inflater = LayoutInflater.from(context);
        this.context = context;
        this.mStrings = strings;
    }

    @Override
    public StringViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = inflater.inflate(android.R.layout.simple_list_item_1, parent, false);
        StringViewHolder holder = new StringViewHolder(view);
        return holder;
    }

    @Override
    public void onBindViewHolder(StringViewHolder holder, int position) {
        // Adapt the ViewHolder state to the new element
        holder.text1.setText(mStrings.get(position));
    }
}
```



4 - Listes et adaptateurs

View dans le layout (xml)

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/listView"  
    android:layout_width="0dp"  
    android:layout_height="0dp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

Associée à un Adapter (pour lier chaque élément de la liste à un item)

Associée à un LayoutManager (pour positionner les items)

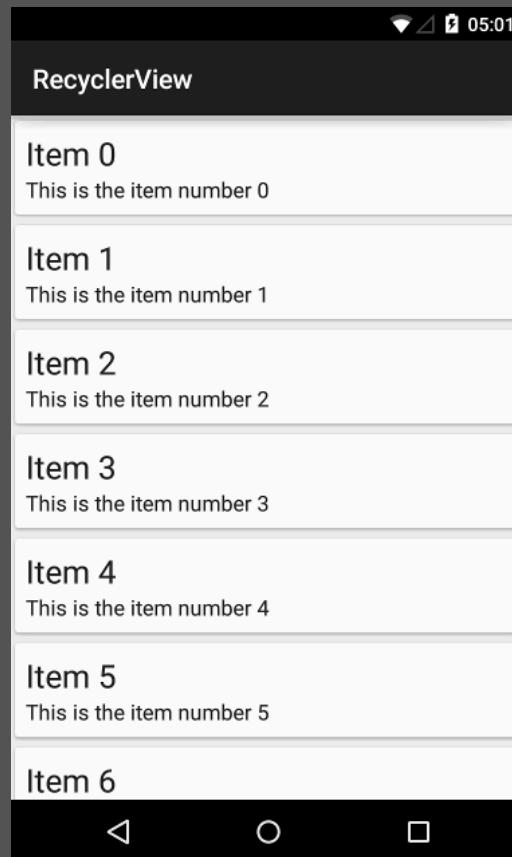
Associée à un ItemAnimator (e.g. animations d'ajouts/suppressions)



4 - Listes et adaptateurs

Associée à un LayoutManager (pour positionner les items)

```
LinearLayoutManager layoutManager = new LinearLayoutManager(context: this);  
layoutManager.setOrientation(LinearLayoutManager.VERTICAL);
```

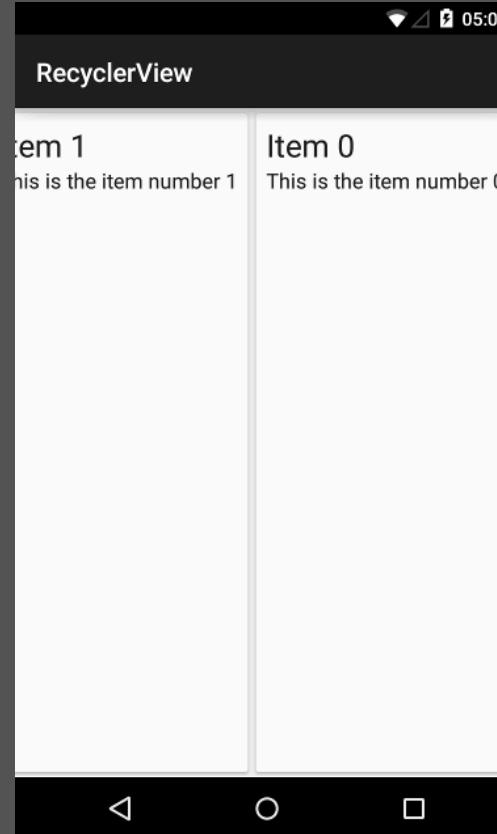




4 - Listes et adaptateurs

Associée à un LayoutManager (pour positionner les items)

```
LinearLayoutManager layoutManager = new LinearLayoutManager( context: this );
layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
```

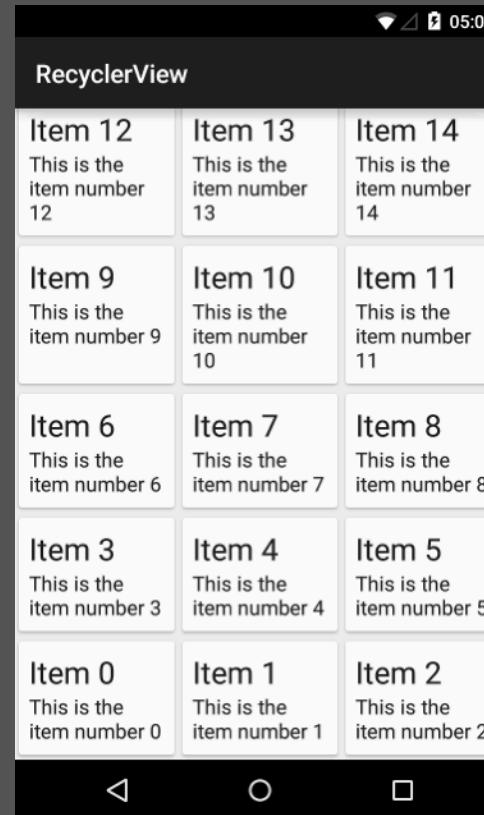




4 - Listes et adaptateurs

Associée à un LayoutManager (pour positionner les items)

```
GridLayoutManager layoutManager = new GridLayoutManager(context: this, spanCount: 2);
```

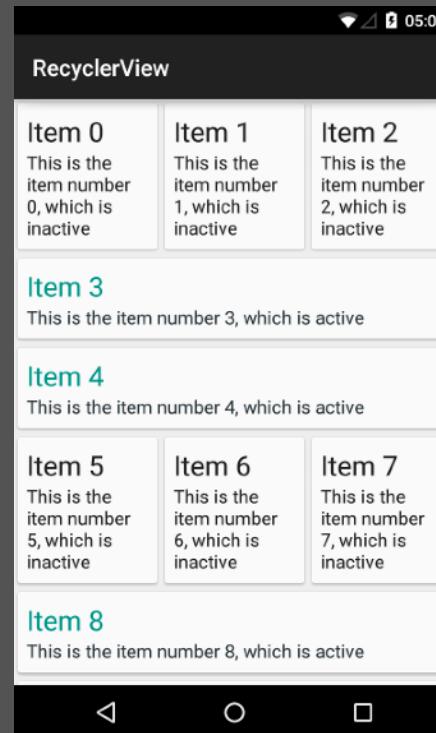




4 - Listes et adaptateurs

Associée à un LayoutManager (pour positionner les items)

```
StaggeredGridLayoutManager layoutManager = new StaggeredGridLayoutManager(spanCount: 3,  
    StaggeredGridLayoutManager.VERTICAL);
```





4 - Listes et adaptateurs

Etape 1 : créer le modèle

```
public class Person {  
    private String firstName;  
    private String lastName;
```

Android Studio shortcuts
• Alt + Insérer (Generate)

Etape 2 : créer le layout de notre Item

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal"  
>  
  
<TextView  
    android:id="@+id/person_adapter_firstname"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"/>  
  
<TextView  
    android:id="@+id/person_adapter_lastname"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"/>  
</LinearLayout>
```



4 - Listes et adaptateurs

Etape 3.1 : créer le ViewHolder dans l'adaptateur

```
public class PersonAdapter extends RecyclerView.Adapter<PersonAdapter.PersonViewHolder> {

    // Pattern ViewHolder
    class PersonViewHolder extends RecyclerView.ViewHolder
    {
        @BindView(R.id.person_adapter_firstname)
        TextView mFirstNameTextView;

        @BindView(R.id.person_adapter_lastname)
        TextView mLastNameTextView;

        public PersonViewHolder(View itemView) {
            super(itemView);
            ButterKnife.bind(@target this, itemView);
        }
    }
}
```



4 - Listes et adaptateurs

Etape 3.2 : créer l'adapter

```
public class PersonAdapter extends RecyclerView.Adapter<PersonAdapter.PersonViewHolder> {

    private LayoutInflator inflater;
    private Context context;
    private List<Person> mPersons;

    public PersonAdapter(Context context, List<Person> persons) {
        inflater = LayoutInflator.from(context);
        this.context = context;
        this.mPersons = persons;
    }

    @Override
    public PersonAdapter.PersonViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = inflater.inflate(R.layout.person_item, parent, false);
        PersonAdapter.PersonViewHolder holder = new PersonAdapter.PersonViewHolder(view);
        return holder;
    }

    @Override
    public void onBindViewHolder(PersonAdapter.PersonViewHolder holder, int position) {
        // Adapt the ViewHolder state to the new element
        holder.mFirstNameTextView.setText(mPersons.get(position).getFirstName());
        holder.mLastNameTextView.setText(mPersons.get(position).getLastName());
    }

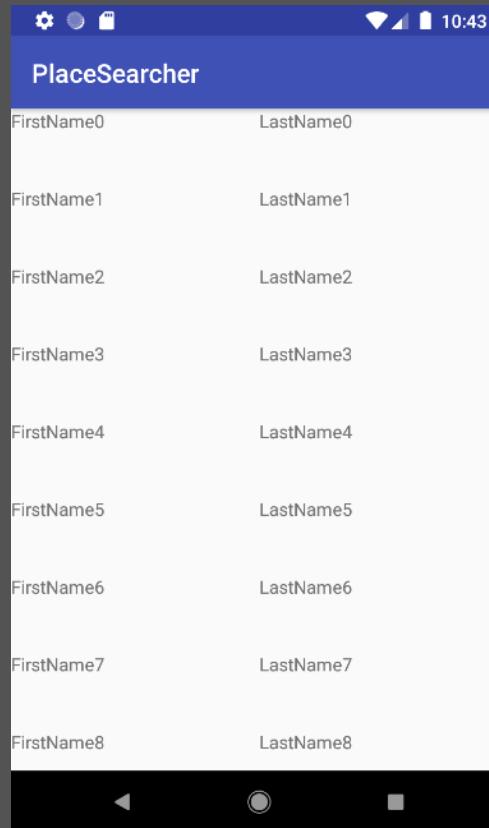
    @Override
    public int getItemCount() {
        return mPersons.size();
    }
}
```

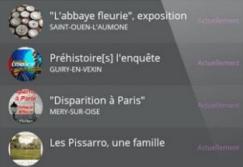


4 - Listes et adaptateurs

Etape 4 : brancher RecyclerView et Adapter dans l'activité

```
// Instantiate a PersonAdapter
PersonAdapter adapter = new PersonAdapter( context: this, listItems);
mRecyclerView.setAdapter(adapter);
mRecyclerView.setLayoutManager(new LinearLayoutManager( context: this));
```





4 - Listes et adaptateurs

Exercice n°7

- Avec le modèle suivant

```
public class Place {  
  
    private double latitude;  
    private double longitude;|  
    private String street;  
    private String zipCode;  
    private String city;
```

- Créer une Liste d'Adresses et l'afficher

Street0	44000	Nantes
Street1	44000	Nantes
Street2	44000	Nantes
Street3	44000	Nantes
Street4	44000	Nantes
Street5	44000	Nantes
Street6	44000	Nantes
Street7	44000	Nantes
Street8	44000	Nantes
Street9	44000	Nantes
Street10	44000	Nantes
Street11	44000	Nantes
Street12	44000	Nantes

5 - Gestion des ressources (assets, drawables)





5 - Gestion des ressources (assets, drawables)

Drawables : tout ce qui peut se dessiner (images ou formes géométriques)



Support multi-résolution possible

- Meilleures performances
- Optimisation mémoire

```
<ImageView  
    android:layout_width="50dp"  
    android:layout_height="50dp"  
    android:src="@drawable/home_icon"/>
```

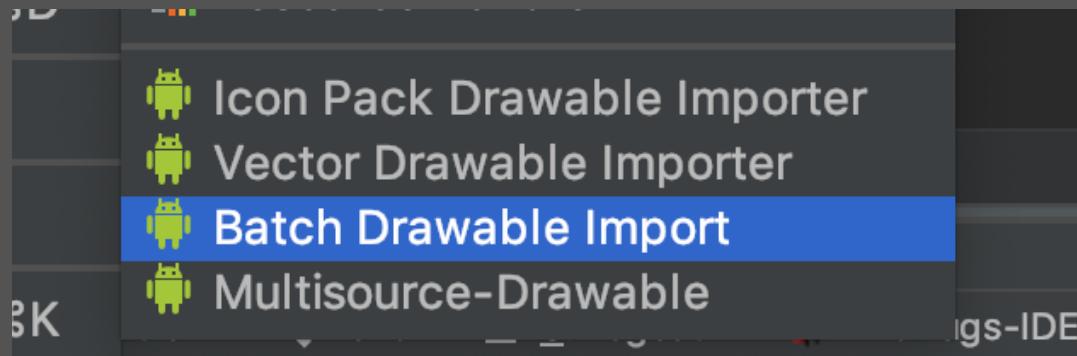
```
mPlaceIcon.setImageResource(R.drawable.home_icon);
```



5 - Gestion des ressources (assets, drawables)

Super Utile !!!!!

<https://www.javahelps.com/2015/02/android-drawable-importer.html>





5 - Gestion des ressources (assets, drawables)

Drawables : tout ce qui peut se dessiner (images ou formes géométriques)

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">

    <solid
        android:color="#666666"/>

    <size
        android:width="120dp"
        android:height="120dp"/>
</shape>
```



5 - Gestion des ressources (assets, drawables)

Assets & raw : le reste (sons, vidéos, PDF...)



```
AssetFileDescriptor afd = getAssets().openFd("AudioFile.mp3");
player = new MediaPlayer();
player.setDataSource(afd.getFileDescriptor(), afd.getStartOffset(), afd.getLength());
player.prepare();
player.start();
```

```
VideoView view = (VideoView) findViewById(R.id.videoView);
String path = "android.resource://" + getPackageName() + "/" + R.raw.video_file;
view.setVideoURI(Uri.parse(path));
view.start();
```



5 - Gestion des ressources (assets, drawables)

Exercice n°8

- Ajouter un drawable pour les maisons et un drawable pour les rues
- Modifier le layout des items pour afficher une image à gauche de l'adresse
- Modifier le code pour afficher une image différente si l'adresse contient « 1 »
- Jouer un son quand on clique sur une image de la liste

	Street0 44000	Nantes
	Street1 44000	Nantes
	Street2 44000	Nantes
	Street3 44000	Nantes
	Street4	

Pari n°4 : quelqu'un va s'amuser avec les sons

THAT'S ALL FOLKS !

NEXT TIME

- Intents
- HTTP
- Asynchronicité
- Evènements

D'ICI LÀ

- Entraînez-vous (RecyclerViews)
- Harcelez-moi (lagardealex@gmail.com)
- <https://discord.gg/4K9gWTt>

