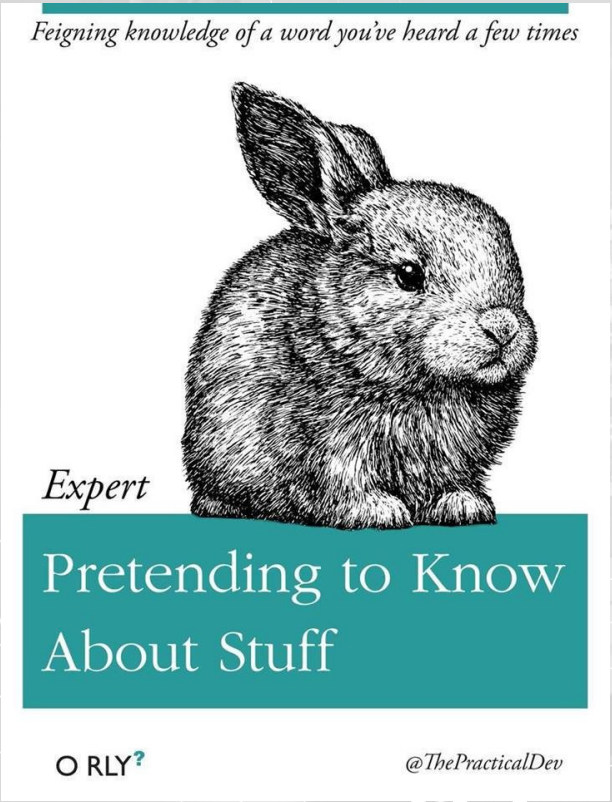
DÉVELOPPEMENT D'APPLICATIONS MOBILES

Alex Morel - iRéalité

@alex_morel_

DISCLAIMER



Le but du cours n'est pas :

- d'apprendre des frameworks à la mode
- de développer en hybride
- de vous affirmer « Expert Android » sur votre CV

Le but du cours c'est :

- comprendre les problématiques propres au dev mobile
- maitriser les concepts permettant de les résoudre et des frameworks simples
- pouvoir postuler à un poste de développeur Android Junior

Arrêtez-moi quand vous voulez !

iREALiTE

Programme du cours :

1. Bases Android
2. HTTP, Asynchronicité, Évènements, Google Maps
3. Base de données, Notifications
4. Bonnes pratiques, débugage, architecture
5. Libre (iOS, Kotlin, Google Face, Framework de tests…)

MOB
ILE
APPS

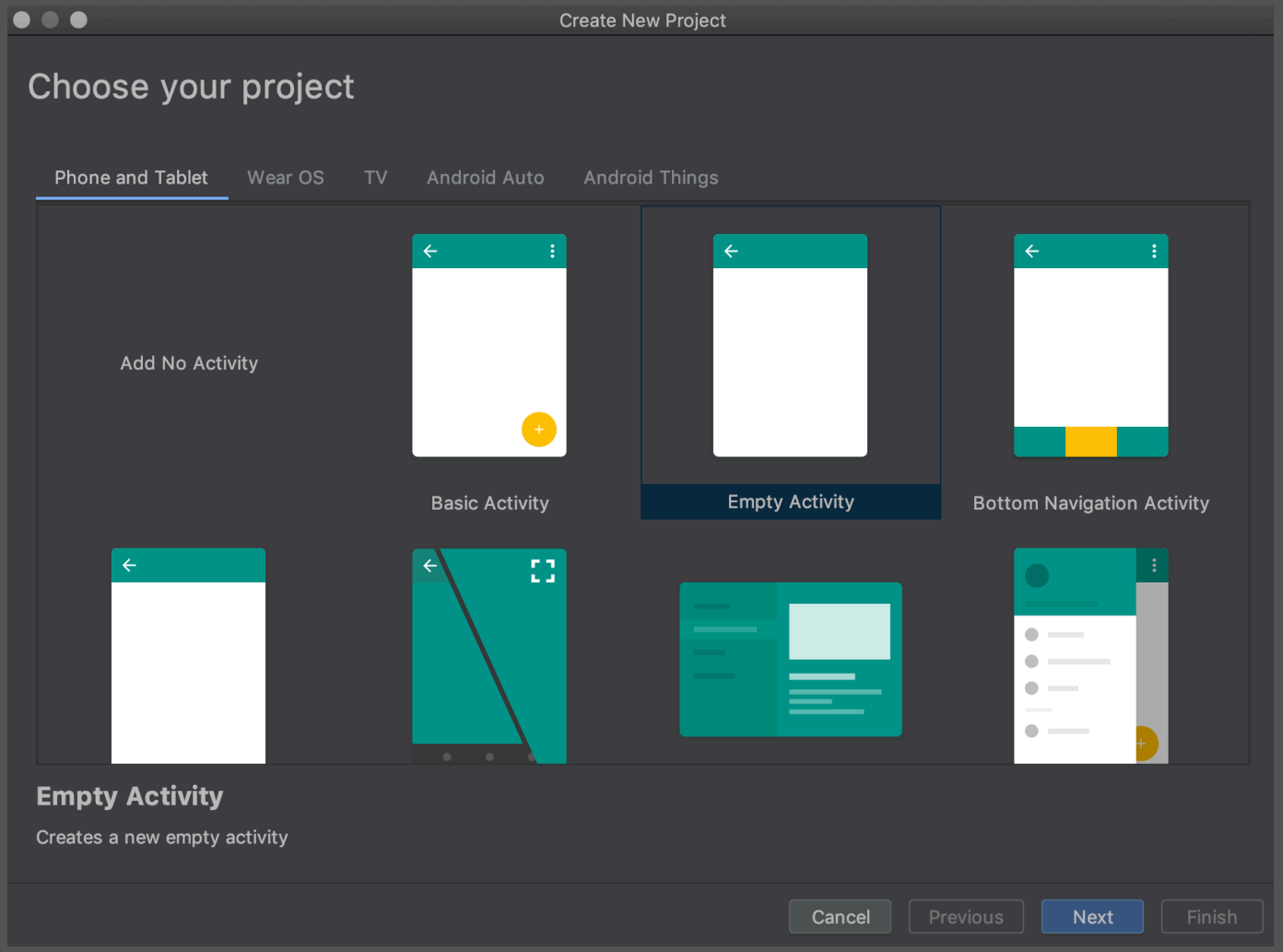
Programme du jour :

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)

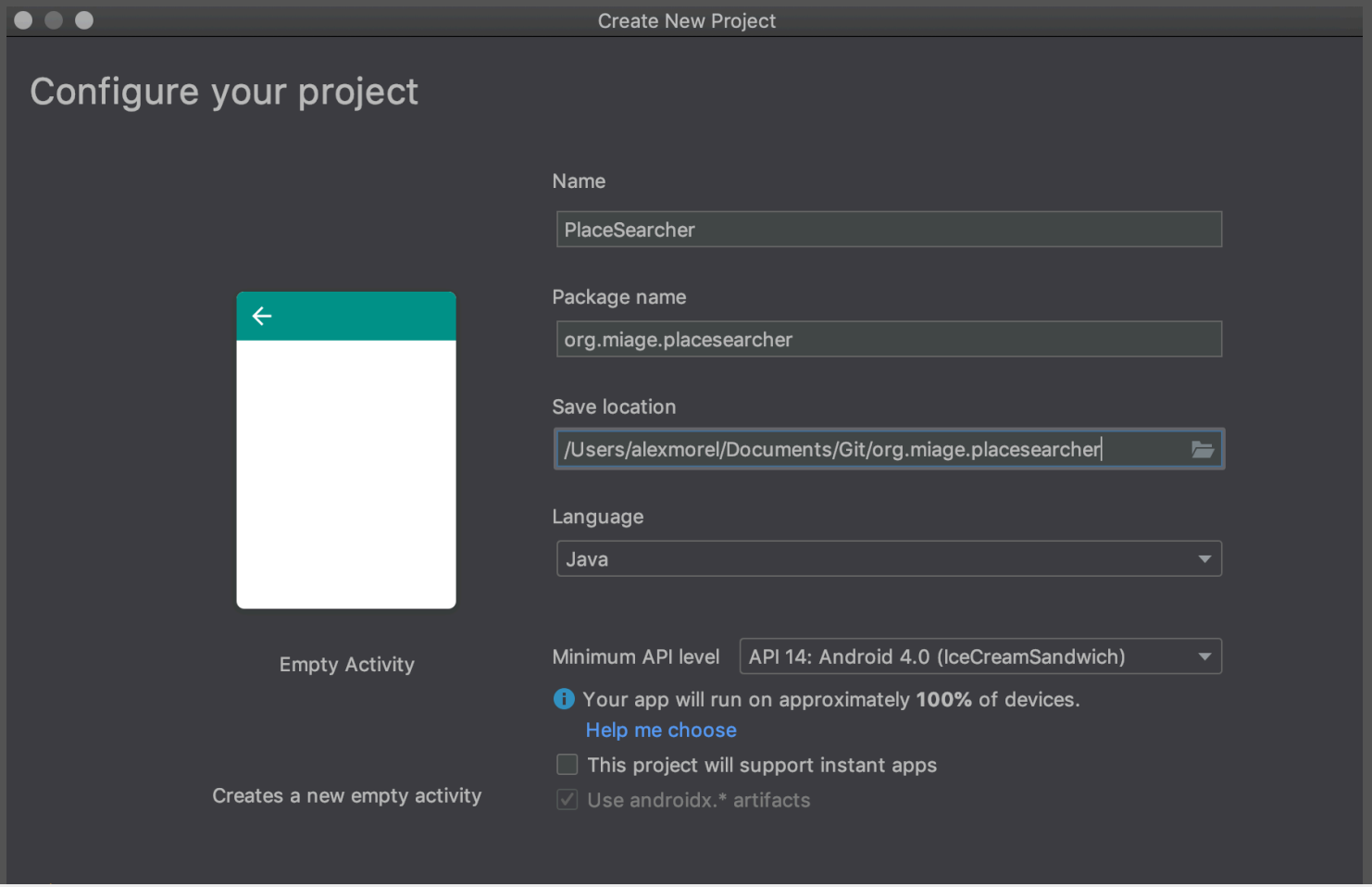
1 - Structure d'un projet Android

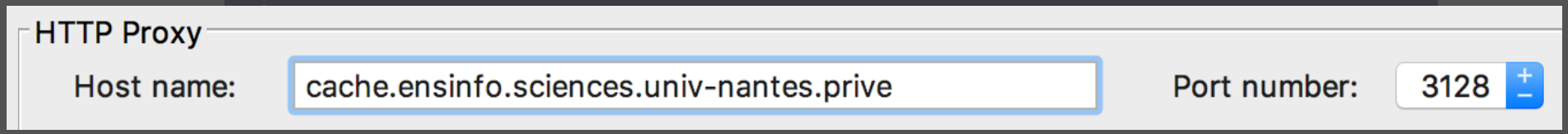
MOB
ILE
APPS

1 - Structure d'un projet Android

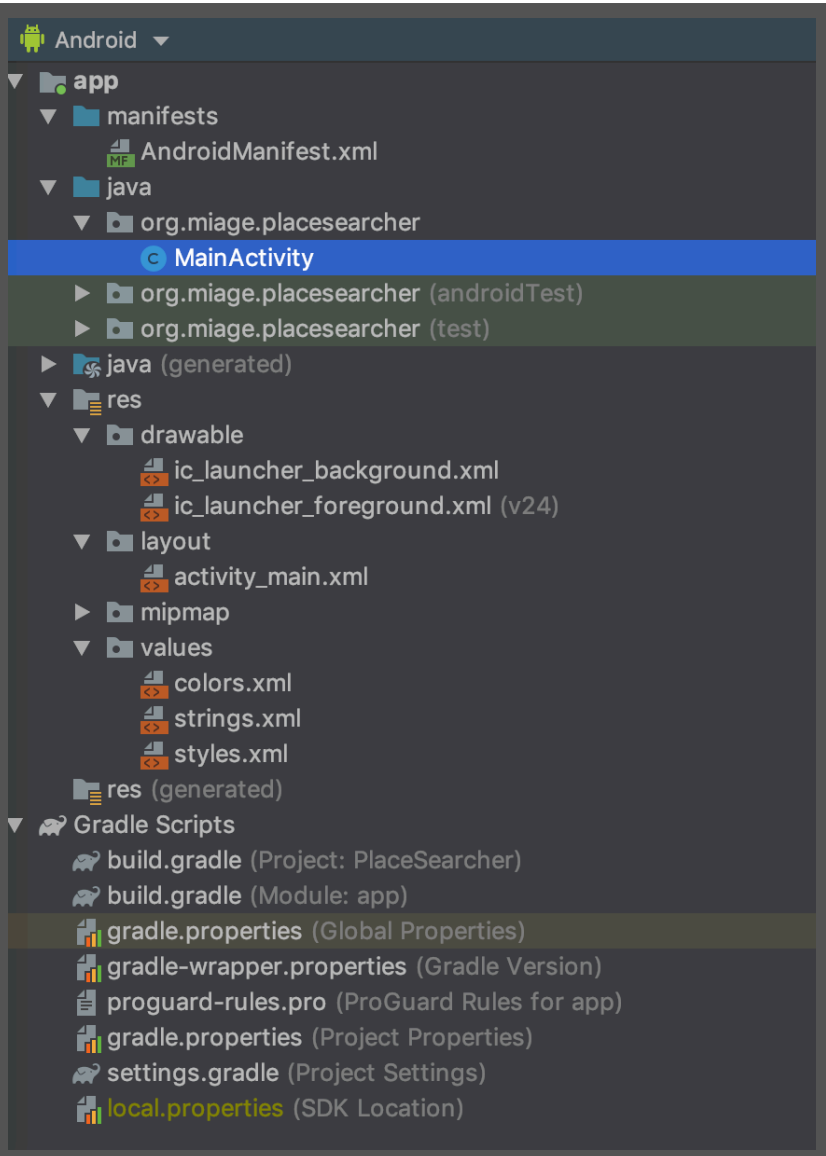


1 - Structure d'un projet Android





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)

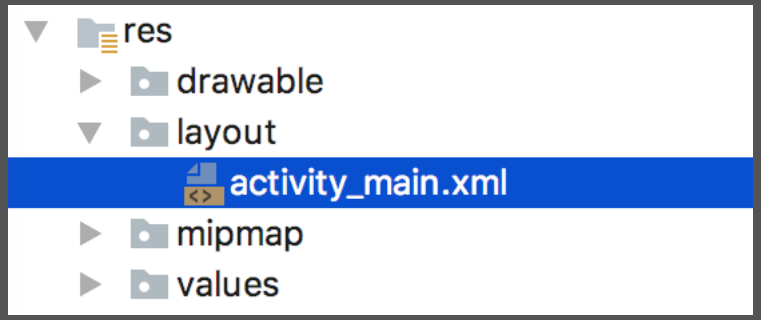
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.miage.placesearcher">

    <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

```java
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

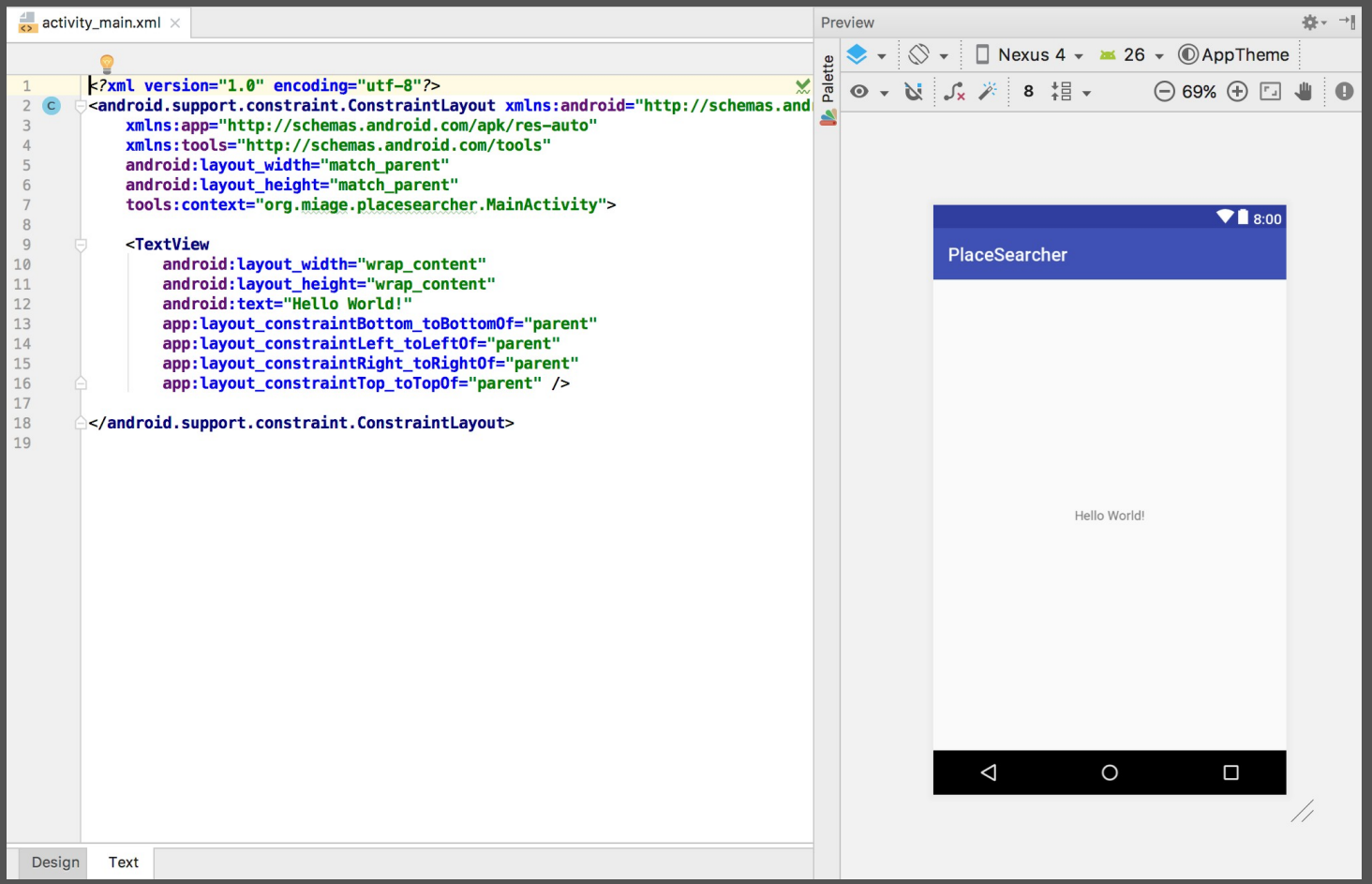
- A connaitre pour brancher les comportements aux bons moments
- Impossible de toucher à l'UI si l'application n'est pas dans l'état « running » (asynchrone)
- Surcharger proprement (appeler la super méthode systématiquement)

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

1 - Structure d'un projet Android

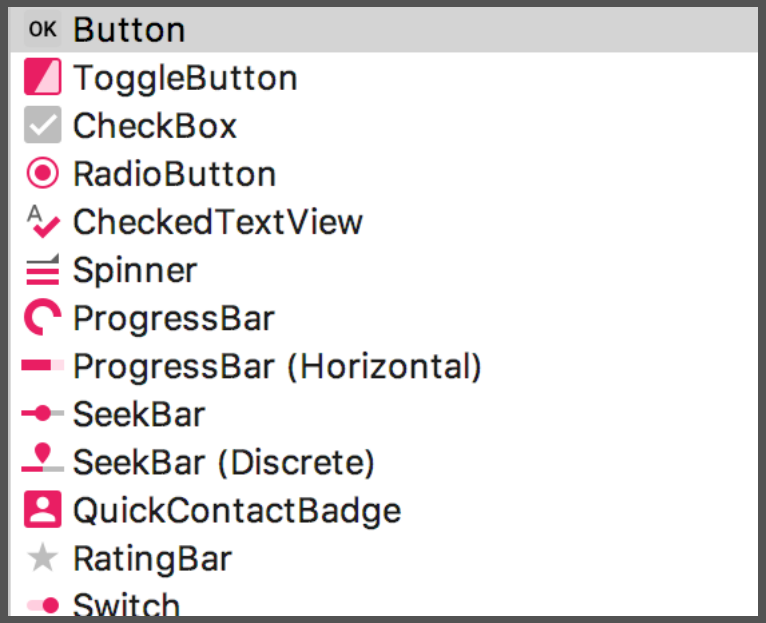
Les vues : fichiers XML (proche HTML)

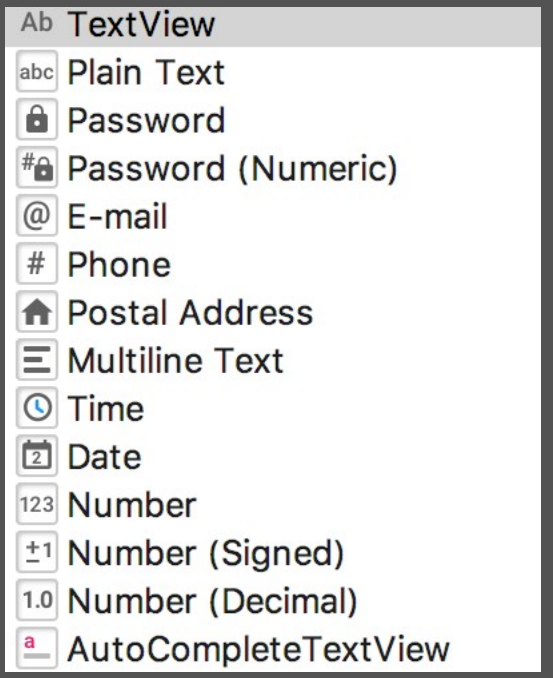
Android Studio shortcut : control + clic



1 - Structure d'un projet Android

Les vues : des widgets natives



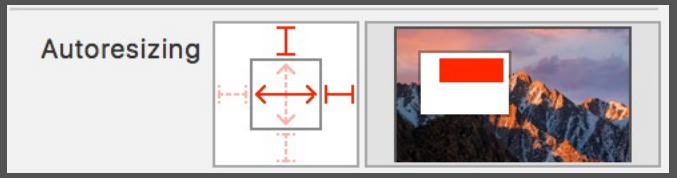


- Possibilité d'installer d'autres vues (carousel, signature…)
- Possibilité de définir ses propres vues

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/slibings)
  - **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
- Controle + 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 développeur 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

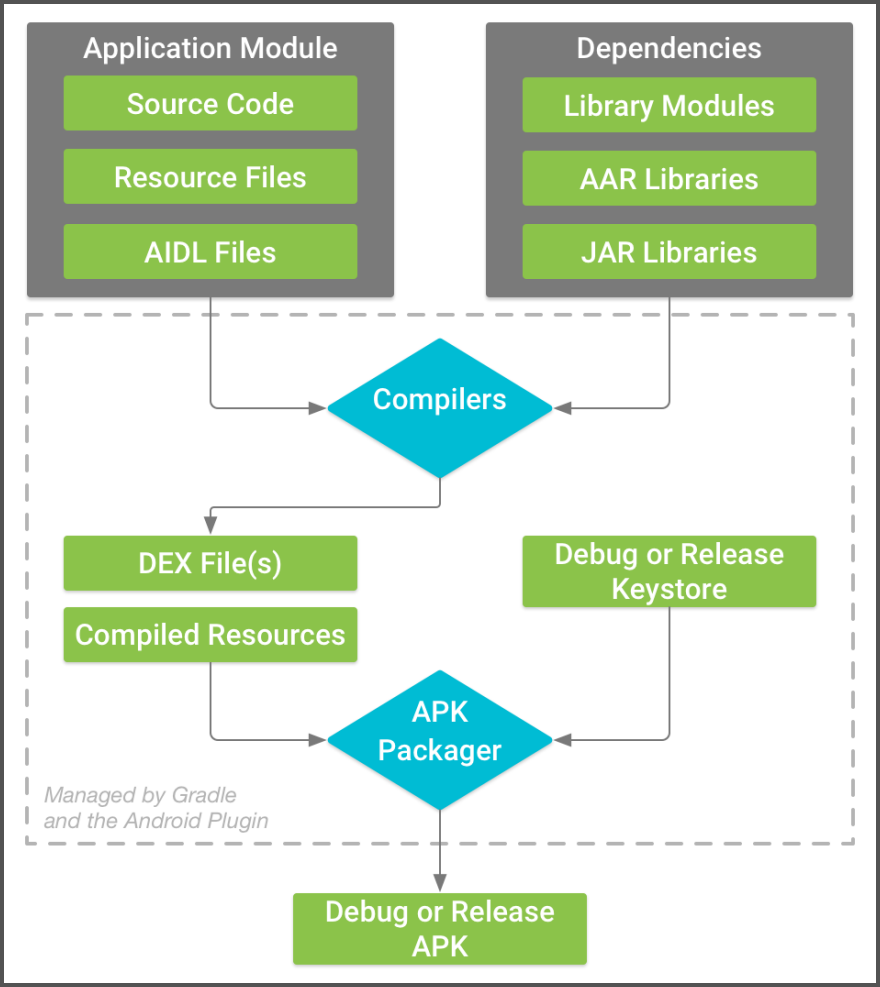
Android Studio shortcuts
- Shift Shift Shift

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

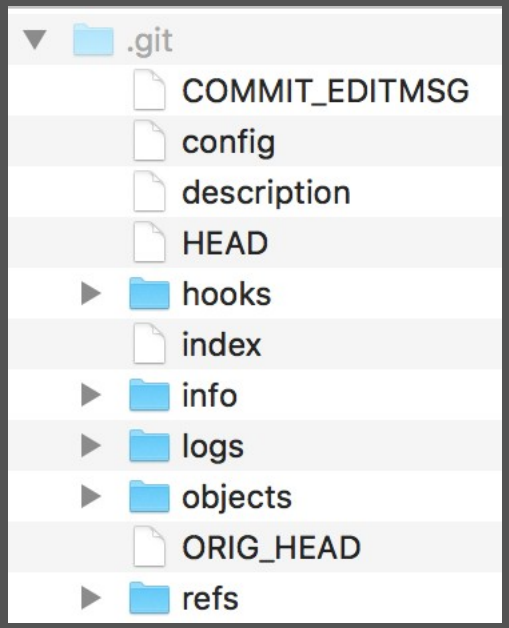
3 - Git - niveau 1

NE TRAVAILLEZ JAMAIS SANS REPO GIT

COMMITEZ PETIT POUR LES CONFLITS

COMMITEZ TOUT LE TEMPS

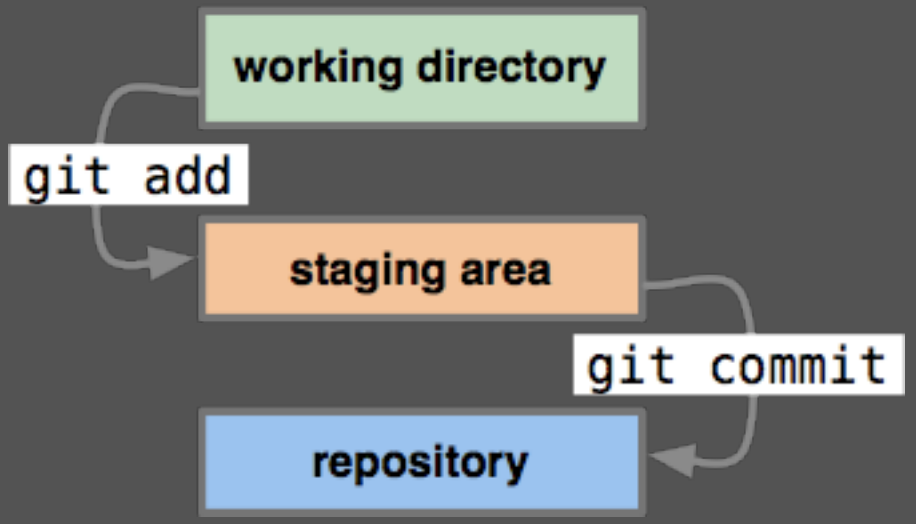
3 - Git - niveau 1



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

3 - Git - niveau 1

Travailler en local : working, staging, repo



3 - Git - niveau 1

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


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

3 - Git - niveau 1

Staged files

Unstaged files

...iage/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. substract 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 +$) $
```

Staged files

...iage/placesearcher/MainActivity.java

Unstaged files

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. substract 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

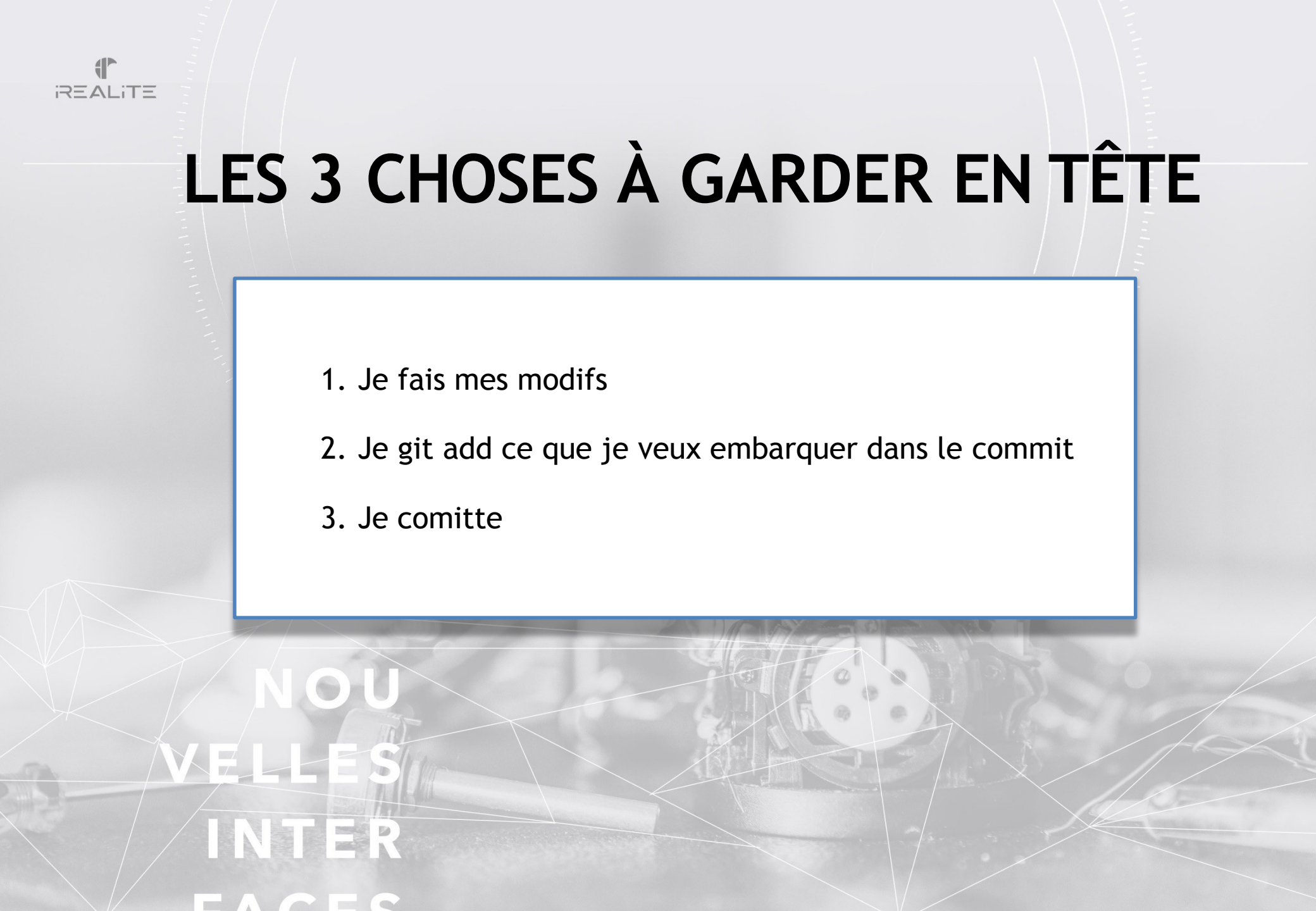
```
alexmorel@~/Documents/Git/org.miage.placesearcher (master +$) $ git commit -m "[Android] Very important changes to comments that will bring an end to all wars"
```

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

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



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

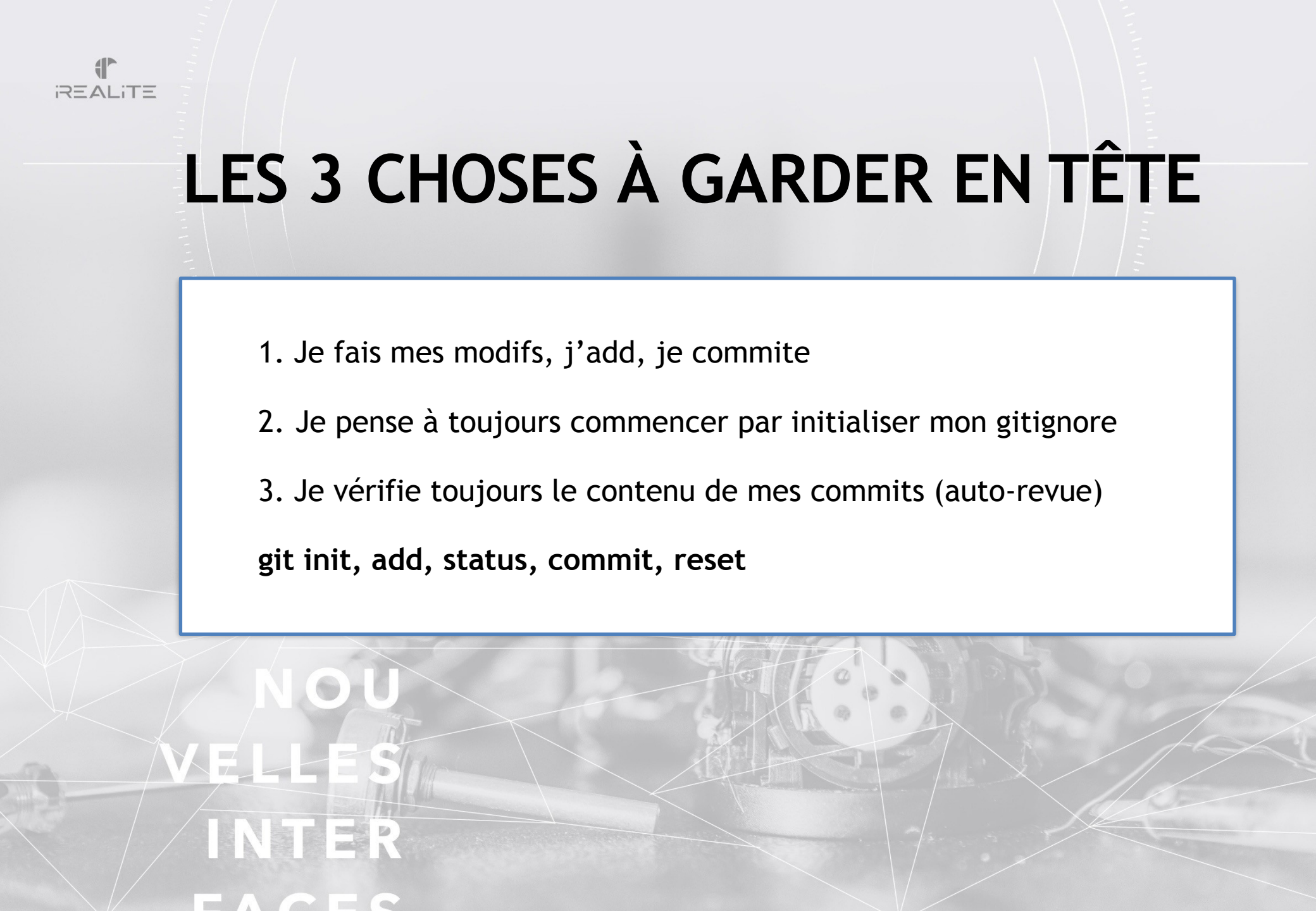
Exercice n°5 (le vrai cette fois)

- 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 commite
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



4 - Listes et adapteurs

APPS

4 - Listes et adapteurs

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 adapteurs

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 élément 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 adapteurs

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

Rôle de l'adapter: afficher les élément 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 adapteurs

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 adapteurs

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

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

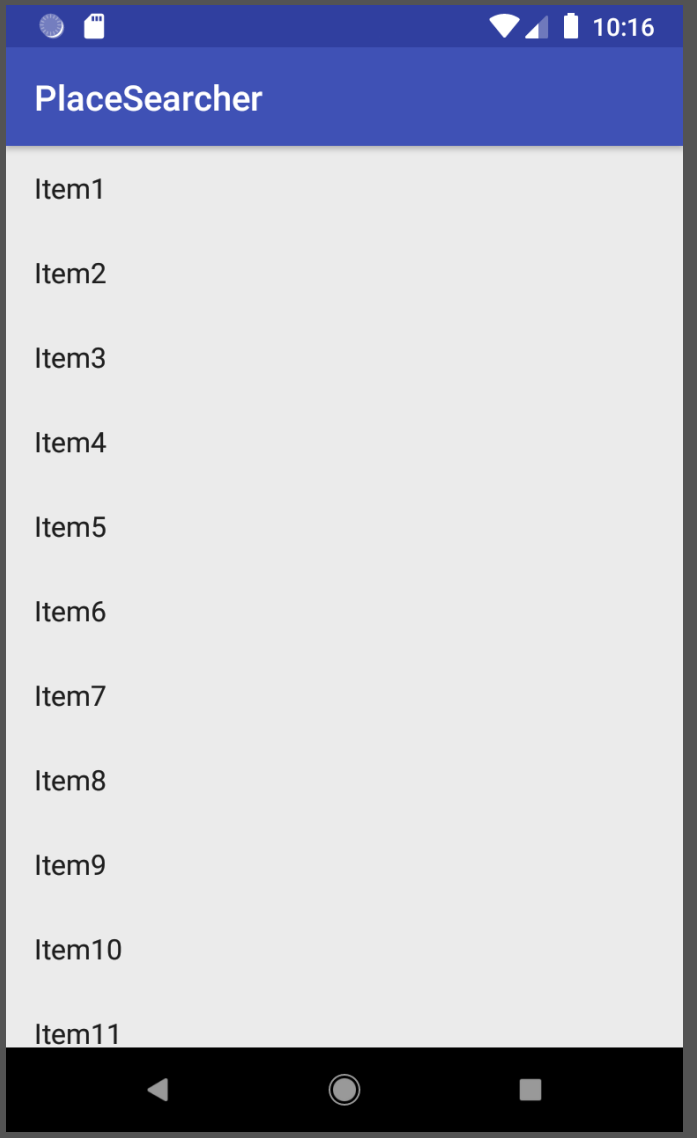
```
// String Adapter : convert string list into Items
public class StringAdapter extends RecyclerView.Adapter<StringAdapter.StringViewHolder>{

    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,  attachToRoot: 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 adapteurs

Exercice n°6

- Vider le layout précédent
- Ajouter une RecyclerView au layout
- Afficher une liste de 50 éléments « Item1 », « Item2 »… « Item50 »
- Pour chaque item vous pouvez utiliser
- android.R.layout.simple_list_item_1 ou créer votre propre vue

https://guides.codepath.com/android/using-the-recyclerview



4 - Listes et adapteurs

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)

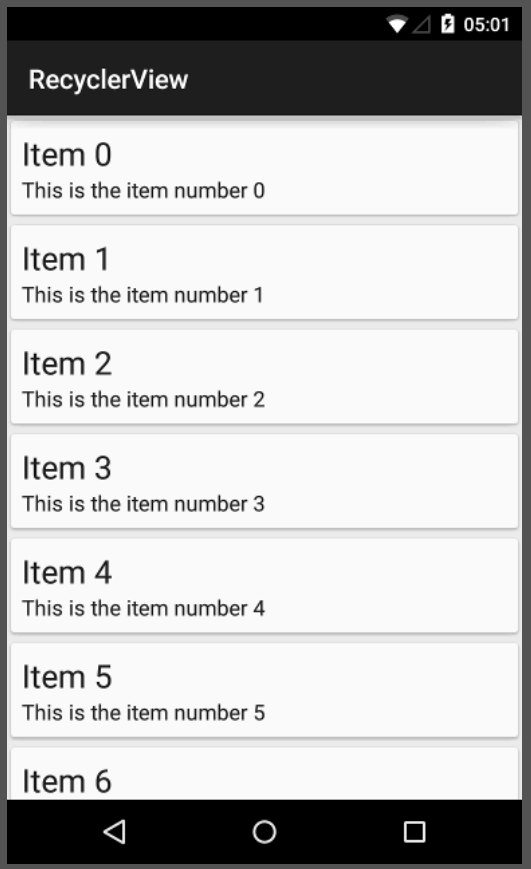
Associée à un LayoutManager (pour positionner les items)

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

4 - Listes et adapteurs

Associée à un LayoutManager (pour positionner les items)

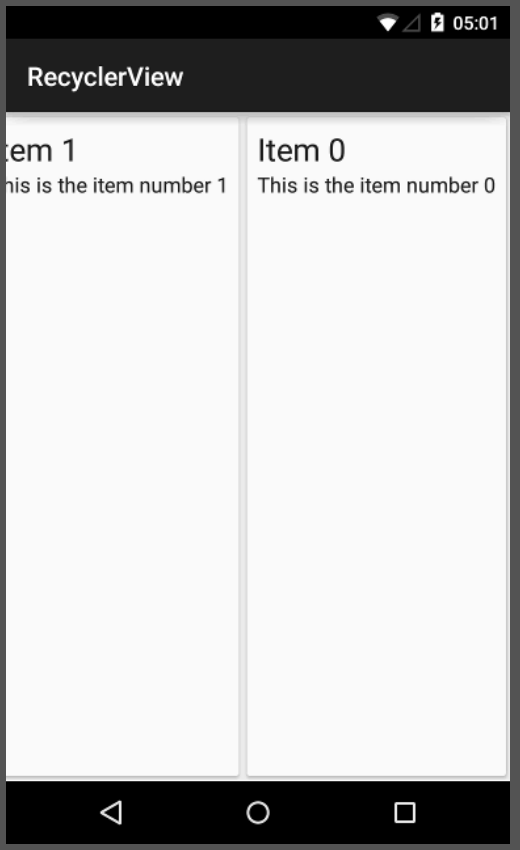
```
LinearLayoutManager layoutManager = new LinearLayoutManager( context: this);
layoutManager.setOrientation(LinearLayoutManager.VERTICAL);
```



4 - Listes et adapteurs

Associée à un LayoutManager (pour positionner les items)

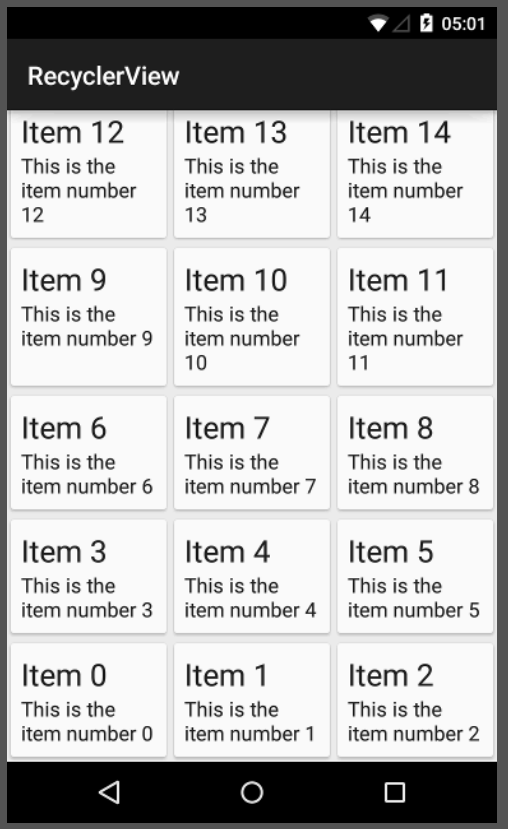
```
LinearLayoutManager layoutManager = new LinearLayoutManager( context: this);
layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
```



4 - Listes et adapteurs

Associée à un LayoutManager (pour positionner les items)

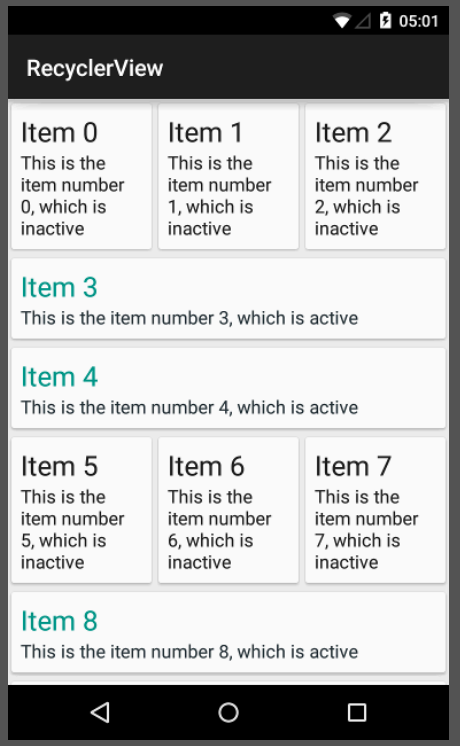
```
GridLayoutManager layoutManager = new GridLayoutManager( context: this, spanCount: 2);
```



4 - Listes et adapteurs

Associée à un LayoutManager (pour positionner les items)

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



4 - Listes et adapteurs

Etape 1 : créer le modèle

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

Android Studio shortcuts
- Alt + Insérer (Generate)

Etape 2 : créer le layout de notre Item

```xml
<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 adapteurs

Etape 3.1 : créer le ViewHolder dans l'adapteur

```
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 adapteurs

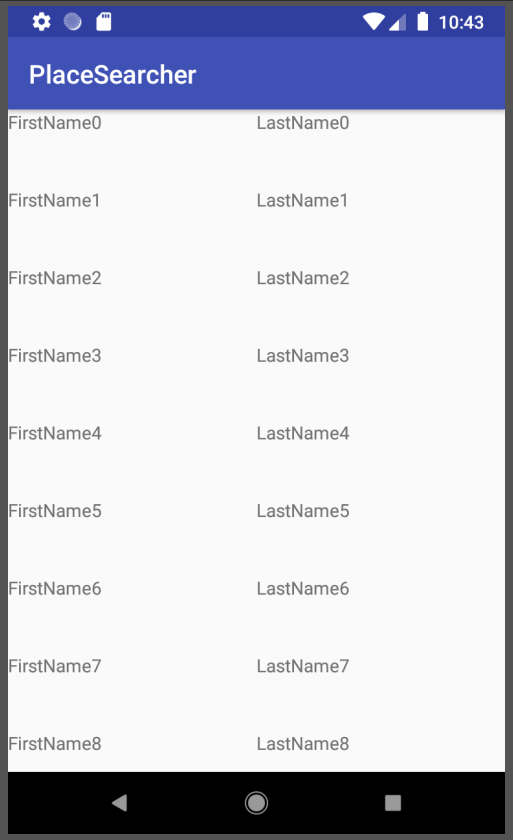
Etape 3.2 : créer l'adapter

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

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

    public PersonAdapter(Context context, List<Person> persons) {
        inflater = LayoutInflater.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,  attachToRoot: 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 adapteurs

Etape 4 : brancher RecyclerView et Adapter dans l'activité

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



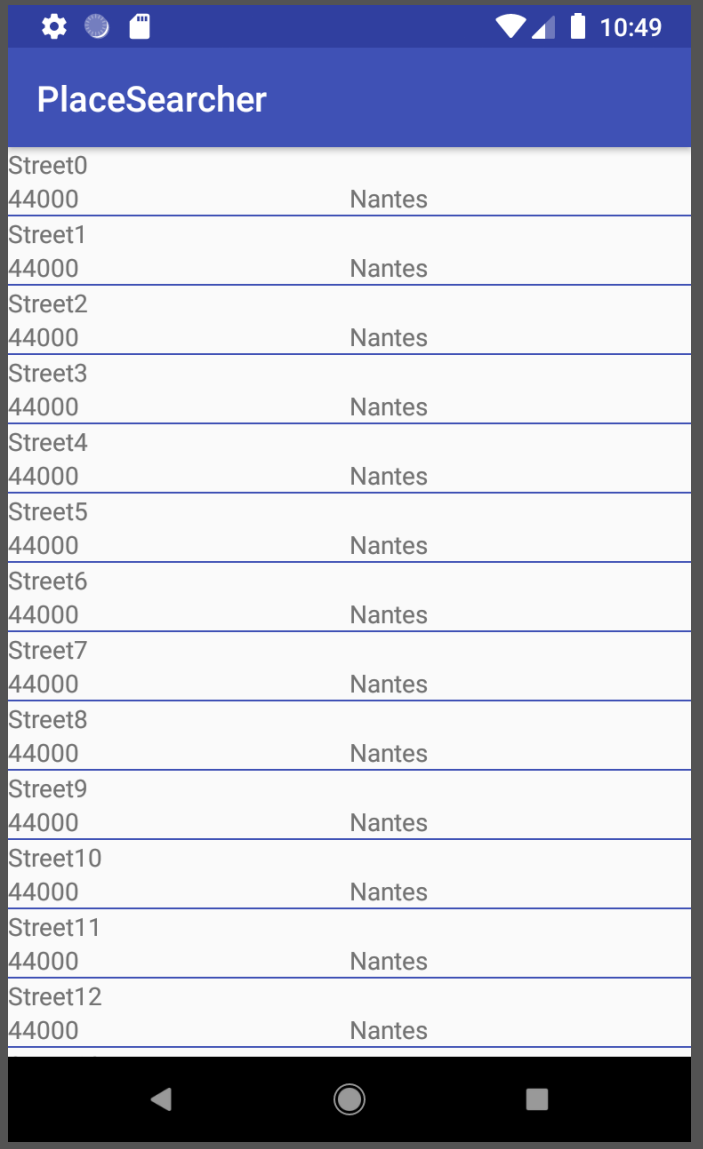
4 - Listes et adapteurs

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

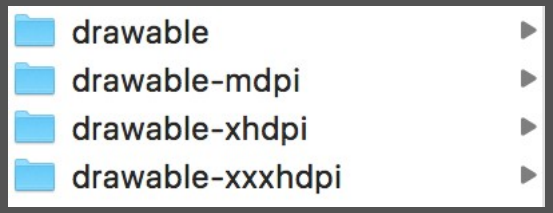


5 - Gestion des ressources (assets, drawables)

MOBILE APPS

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

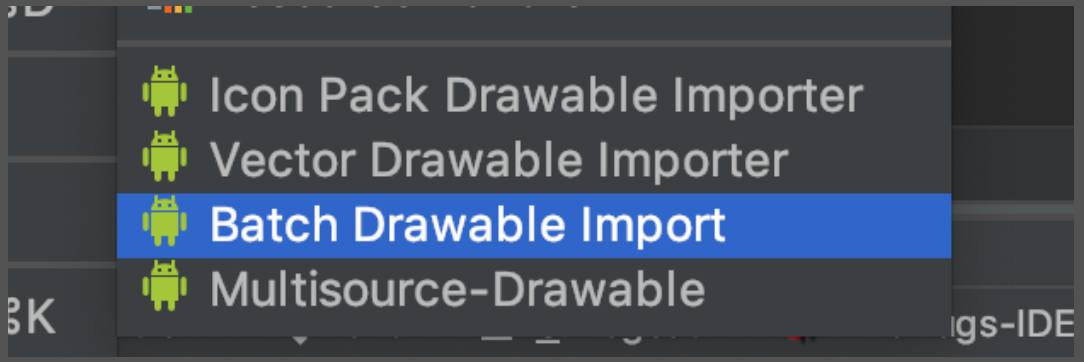
```
<ImageView
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:src="@drawable/home_icon"/>
```

```
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

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

THAT'S ALL FOLKS !

NEXT TIME

- Intents
- HTTP
- Asynchronicité
- Evènements

D'ICI LÀ

- Entrainez-vous (RecyclerViews)
- Harcelez-moi (alex.morel@irealite.com)