

# DÉVELOPPEMENT D'APPLICATIONS MOBILES



Alex Morel - Code Lutin



@alex\_morel\_



## Programme du cours :

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

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

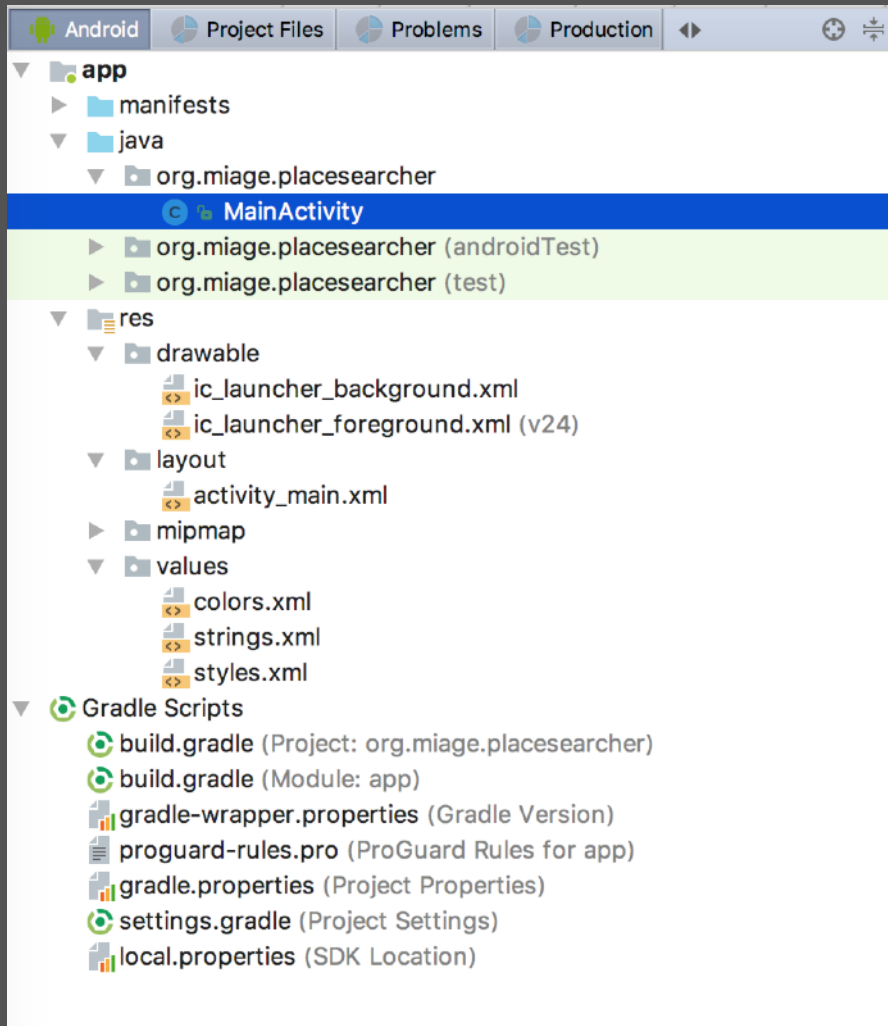
MOB  
ILE  
APPS

PREVIOUSLY ON

*Développement d'applications mobiles*  
*Cours 1*



# Structure d'un projet Android



- Le Manifest : contrat de votre App
- Les activités : les contrôleurs de votre App
- Les layouts : vues décrites en XML
- Les ressources : drawables & assets





# Structure d'un projet Android



## Cycle de vie d'une Activité

- `onCreate()` c'est pas comme `onResume()`



## Structure d'un projet Android



**Build Android (gradle)** : lister les dépendances dans le build.gradle



**ButterKnife** : sucre syntaxique : @BindView, @OnClick



**Git - niveau 1** : travail local (git init, add, commit, status, reset)



**Listes et adaptateurs**



## 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 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());
    }

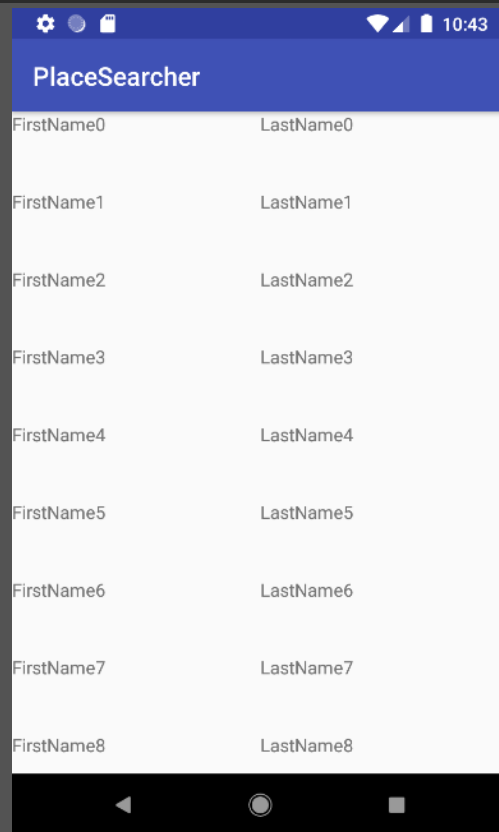
    @Override
    public int getItemCount() {
        return mPersons.size();
    }
}
```



## 4 - Listes et adapteurs

### Etape 4 : brancher RecyclerView et Adapter dans l'activité

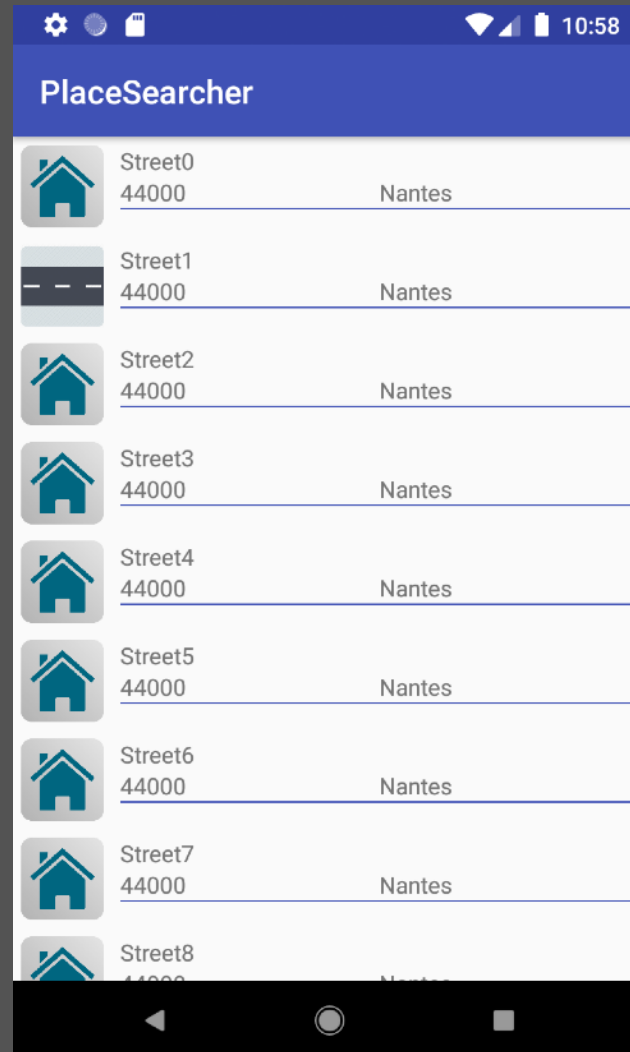
```
// Instanciate a PersonAdapter
PersonAdapter adapter = new PersonAdapter( context: this, listItems);
mRecyclerView.setAdapter(adapter);
mRecyclerView.setLayoutManager(new LinearLayoutManager( context: this));
```







## 4 - Listes et adaptateurs



## 5 - Gestion des ressources (assets, drawables)



MOB  
ILE  
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"/>
```

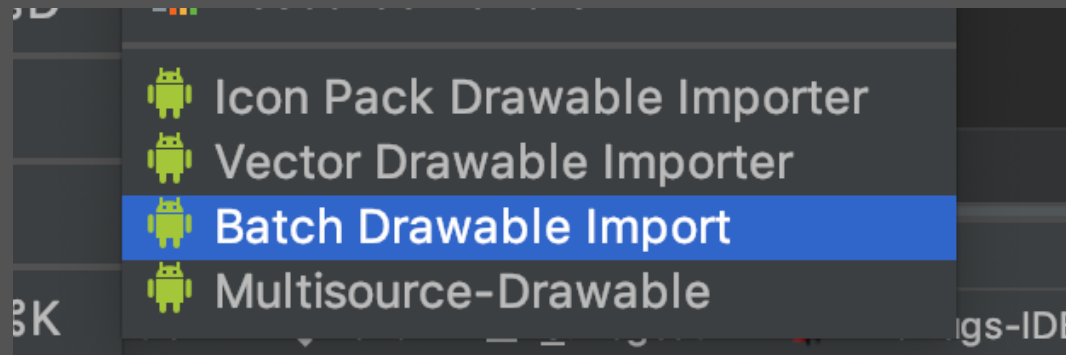
```
mPlaceIcon.setImageResource(R.drawable.home_icon);
```



## 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*

|   |                  |        |
|---|------------------|--------|
|    | Street0<br>44000 | Nantes |
|   | Street1<br>44000 | Nantes |
|  | Street2<br>44000 | Nantes |
|  | Street3<br>44000 | Nantes |
|  | Street4          |        |

## Nos objectifs du jour :

1. Les Intents : vers d'autres activités et au-delà
2. Git - niveau 2
3. Requêtes HTTP : les bases
4. Asynchronicité & parallélisme
5. Évènements
6. Retrofit (si vous êtes encore vivants)

MOB  
ILE  
APPS

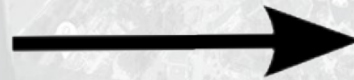
## 1 - Les Intents : vers d'autres activités et au-delà

MOB  
ILE  
APPS





# 1 - Les Intents : vers d'autres activités et au-delà



Intent

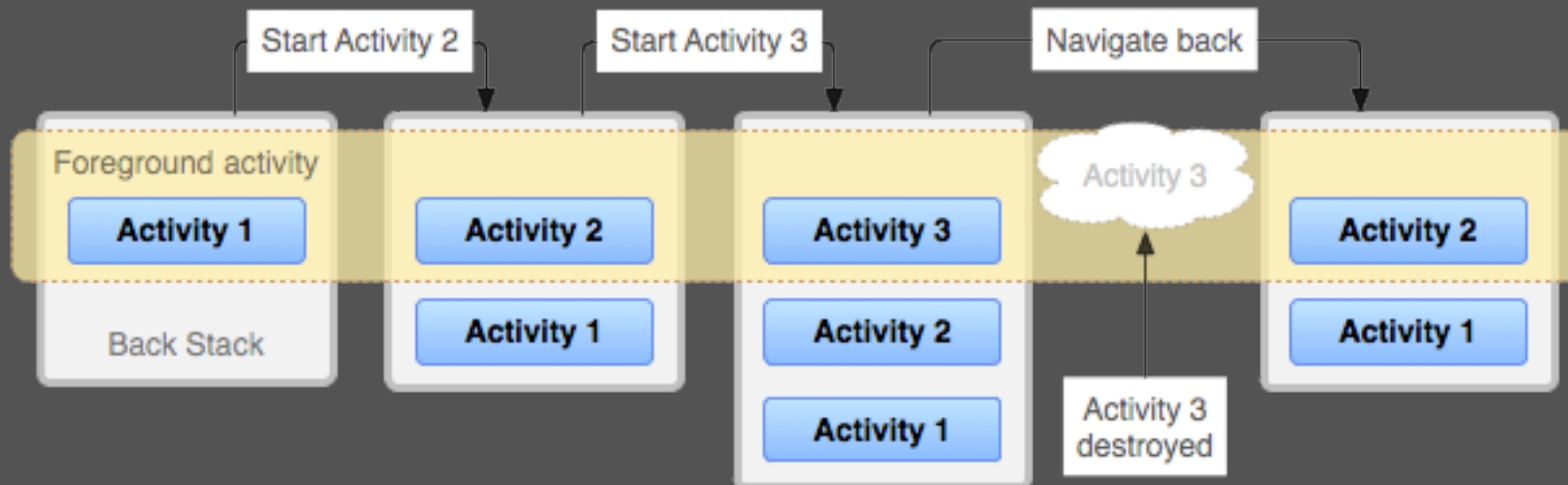


```
Intent seePlaceDetailsIntent = new Intent( packageContext: this, PlaceDetailActivity.class);  
startActivity(seePlaceDetailsIntent);
```

APPS



# 1 - Les Intents : vers d'autres activités et au-delà

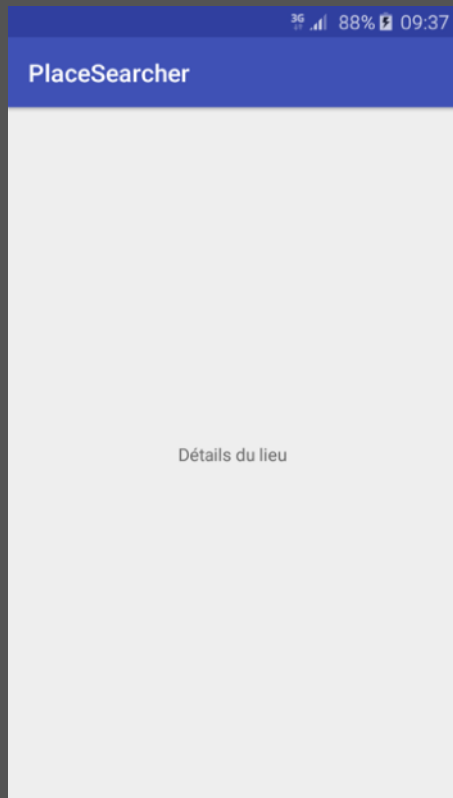


```
PlaceDetailActivity.this.finish();
```



# 1 - Les Intents : vers d'autres activités et au-delà

## Exercice n°9



- Créer une nouvelle activité PlaceDetailsActivity
- Lorsqu'on clique sur une icône de la liste, lancer cette activité
- Appuyer sur le bouton « retour » du téléphone, que se passe-t-il ?
- Au clic sur la TextView de la PlaceDetailsActivity :
  - Revenir à l'écran précédent (MainActivity)
  - 2 façons de procéder (avec ou sans Intent), quelle différence ?

[irealite.com/miage](http://irealite.com/miage)





# 1 - Les Intents : vers d'autres activités et au-delà

```
public class PlaceDetailActivity extends AppCompatActivity {  
  
    @BindView(R.id.acti  
    TextView mPlaceStre  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```

- ✎ Add activity to manifest ►
- ✎ Create Test ►
- ✎ Create subclass ►
- ✎ Make package-private ►

Android Studio shortcuts : alt + Entrée, tout le temps, partout !



# 1 - Les Intents : vers d'autres activités et au-delà

Ajouter des paramètres à l'Intent

```
seePlaceDetailIntent.putExtra( name: "myIntentExtra1", value: "SomeValue");
```

Récupérer les paramètres dans l'activité lancée

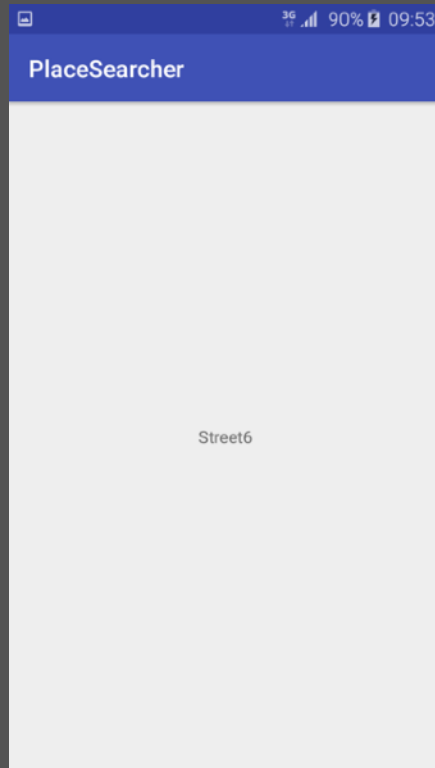
```
String myIntentExtra1Value = getIntent().getStringExtra( name: "MyIntentExtra1");
```



## 1 - Les Intents : vers d'autres activités et au-delà

### Exercice n° 10

- Faire en sorte que la PlaceDetailsActivity affiche la « street » du lieu sur lequel on a cliqué





# 1 - Les Intents : vers d'autres activités et au-delà

Lancer une application externe avec Intent.ACTION\_\*

```
// Open browser using an Intent  
Uri url = Uri.parse("http://irealite.com");  
Intent launchBrowser = new Intent(Intent.ACTION_VIEW, url);  
startActivity(launchBrowser);
```

```
// Open share picker using an Intent  
Intent sendIntent = new Intent();  
sendIntent.setAction(Intent.ACTION_SEND);  
sendIntent.putExtra(Intent.EXTRA_TEXT, value: "SOME TEXT TO SHARE");  
sendIntent.setType("text/plain");  
startActivity(sendIntent);
```



# 1 - Les Intents : vers d'autres activités et au-delà

## Exercice n° 10



- Créer des boutons permettant de :
  - chercher la street sur google (navigateur)  
<https://www.google.com/search?q=Ma%20Super%20Recherche>
  - partager la street (via SMS, Mail, Facebook...)



# 1 - Les Intents : vers d'autres activités et au-delà

On a souvent besoin de récupérer le résultat d'un Intent

```
// Open gallery picker using an Intent
Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
photoPickerIntent.setType("image/*");
startActivityForResult(photoPickerIntent, SELECT_PHOTO);
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent imageReturnedIntent) {
    super.onActivityResult(requestCode, resultCode, imageReturnedIntent);

    // If we get a result from the SELECT_PHOTO query
    switch(requestCode) {
        case SELECT_PHOTO:
            if(resultCode == RESULT_OK){
                // Get the selected image as bitmap
                Uri selectedImage = imageReturnedIntent.getData();
```





# 1 - Les Intents : vers d'autres activités et au-delà

## Exercice n°11 (à faire chez vous si vous le souhaitez)



- Créer un bouton permettant de :
  - sélectionner une image dans votre galerie
  - afficher l'image choisie dans la vue

## 2 - Git - niveau 2





## 2 - Git - niveau 2

NE TRAVAILLEZ **JAMAIS** SANS REPO GIT

COMMITEZ **PETIT** POUR LES CONFLITS

COMMITEZ **TOUT** LE TEMPS

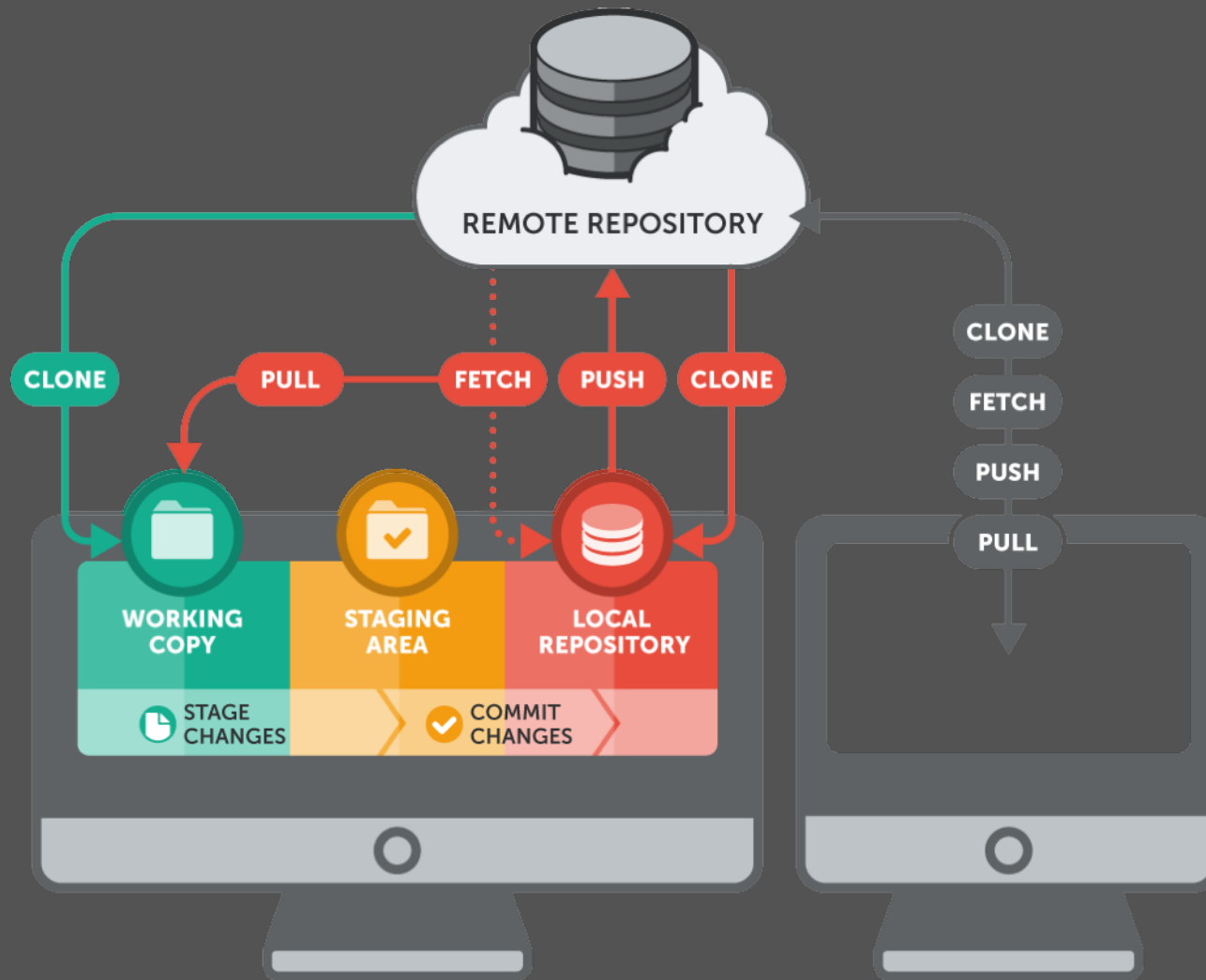


## 2 - Git - niveau 2





## 2 - Git - niveau 2





## 2 - Git - niveau 2

Un repo git peut connaître d'autre repo git (« Remotes »)

|         |               |   |
|---------|---------------|---|
| 192d2ef | <b>master</b> | [Android] Exercice 14 : refactor PlaceService using Retrofit  |
| ecb943d |               | [Android] Exercice 13 : use Otto Event Bus & create SearchResultEvent   |
| 1dc7908 |               | [Android] Exercice 12 - Step 3 : parse result as JSON   |
| 54667b2 |               | [Android] Exercice 12 - Step 2 : extract logic in PlaceSearchService  |
| 6809061 |               | [Android] Exercice 12 - Step1 : use OkHTTP to make HTTP request inside an AsyncTask                           |
| b16dfa3 |               | [Android] Exercice 11 : add button allowing to pick a picture and display it in the Place Details Activity    |
| 69285d3 |               | [Android] Exercice 10 : add Intents to share & search a place   |
| dffbe48 |               | [Android] Exercice 9 : add Place details Activity   |
| 040b694 |               | [Android] Exercice 8 - Step 2 : play mp3 sound on item clicks   |
| 24ac240 |               | [Android] Exercice 8 - Step 1 : add drawables in Places items   |
| d00b351 |               | [Android] Exercice n°7 : create Place ListView  |
| 8489593 |               | Revert "[Android] Course - create Person adapter"   |
| b6d4ba1 |               | [Android] Course - create Person adapter  |
| d8d1981 |               | [Android] Exercice 6 : ArrayAdapter with 50 items   |
| c170159 |               | [Android] Exercice 4 : refactor MainActivity by using ButterKnife   |
| ee00c08 |               | [Android] [Build] Exercice 3 : add ButterKnife to build.gradle  |
| bb7fd20 |               | [Android] Exercice 2 - Define OnClickListener on TextView   |
| f926277 |               | [Android] Exercice 1 - Add rating bar to default activity and increment its value each time activity is shown |
| 679f973 |               | [Android] [Build] Initial default project import  |
| 3edb5b3 |               | [Android] [Build] Configure gitignore file - from gitignore.io  |

Mon Repo Git Local





## 2 - Git - niveau 2

Un repo git peut connaître d'autre repo git (« Remotes »)

```
193d3ef [master] [Android] Exercise 14 : refactor PlaceService using Retrofit
ecb943d [Android] Exercise 13 : use Otto Event Bus & create SearchResultEvent
1dc7908 [Android] Exercise 12 - Step 3 : parse result as JSON
54667b2 [Android] Exercise 12 - Step 2 : extract logic in PlaceSearchService
6809061 [Android] Exercise 12 - Step1 : use OkHttp to make HTTP request inside an AsyncTask
b16dfa3 [Android] Exercise 11 : add button allowing to pick a picture and display it in the Place Details Activity
69285d3 [Android] Exercise 10 : add intents to share & search a place
dffe4e [Android] Exercise 9 : add Place details Activity
040b694 [Android] Exercise 8 - Step 2 : play mp3 sound on item clicks
24ac240 [Android] Exercise 8 - Step 1 : add drawables in Places items
d00b351 [Android] Exercise n°7 : create Place ListView
8489593 Revert "[Android] Course - create Person adapter"
b6d4ba1 [Android] Course - create Person adapter
d8d1981 [Android] Exercise 6 : ArrayAdapter with 50 items
c170159 [Android] Exercise 4 : refactor MainActivity by using ButterKnife
ee00c08 [Android] [Build] Exercise 3 : add ButterKnife to build.gradle
bb7fd20 [Android] Exercise 2 - Define OnClickListener on TextView
f926277 [Android] Exercise 1 - Add rating bar to default activity and increment its value each time activity is shown
679f973 [Android] [Build] Initial default project import
3edb5b3 [Android] [Build] Configure gitignore file - from gitignore.io
```

Mon Repo Git Local

```
040b694 [team/master] [Android] Exercise 8 - Step 2 : play mp3 sound on item clicks
24ac240 [Android] Exercise 8 - Step 1 : add drawables in Places items
d00b351 [Android] Exercise n°7 : create Place ListView
8489593 Revert "[Android] Course - create Person adapter"
b6d4ba1 [Android] Course - create Person adapter
d8d1981 [Android] Exercise 6 : ArrayAdapter with 50 items
c170159 [Android] Exercise 4 : refactor MainActivity by using ButterKnife
ee00c08 [Android] [Build] Exercise 3 : add ButterKnife to build.gradle
bb7fd20 [Android] Exercise 2 - Define OnClickListener on TextView
f926277 [Android] Exercise 1 - Add rating bar to default activity and increment its value each time activity is shown
679f973 [Android] [Build] Initial default project import
3edb5b3 [Android] [Build] Configure gitignore file - from gitignore.io
```

Un autre Repo Git (Bitbucket)



## 2 - Git - niveau 2

```
git remote add <NOM DU REMOTE> <URL DU REMOTE>
```

```
git remote add team https://alexmorel@bitbucket.org/alexmorel/org.miage.placesearcher
```

```
193d3ef [master] [Android] Exercice 14 : refactor PlaceService using Retrofit
ecb943d [Android] Exercice 13 : use Otto Event Bus & create SearchResultEvent
14c7908 [Android] Exercice 12 - Step 3 : parse result as JSON
54667b2 [Android] Exercice 12 - Step 2 : extract logic in PlaceSearchService
6809061 [Android] Exercice 12 - Step1 : use OkHttpClient to make HTTP request inside an AsyncTask
b16dfa3 [Android] Exercice 11 : add button allowing to pick a picture and display it in the Place Details Activity
69285d3 [Android] Exercice 10 : add intents to share & search a place
dffe48 [Android] Exercice 9 : add Place details Activity
040b694 [Android] Exercice 8 - Step 2 : play mp3 sound on item clicks
24ac240 [Android] Exercice 8 - Step 1 : add drawables in Places items
d00b351 [Android] Exercice n°7 : create Place ListView
8489593 Revert "[Android] Course - create Person adapter"
b6d4ba1 [Android] Course - create Person adapter
d8d1981 [Android] Exercice 6 : ArrayAdapter with 50 items
c170159 [Android] Exercice 4 : refactor MainActivity by using ButterKnife
ee00c08 [Android] [Build] Exercice 3 : add ButterKnife to build.gradle
bb7f020 [Android] Exercice 2 - Define OnClickListener on TextView
f926277 [Android] Exercice 1 - Add rating bar to default activity and increment its value each time activity is shown
679f973 [Android] [Build] Initial default project import
3ed5b3 [Android] [Build] Configure gitignore file - from gitignore.io
```

team

```
040b694 team/master [Android] Exercice 8 - Step 2 : play mp3 sound on item clicks
24ac240 [Android] Exercice 8 - Step 1 : add drawables in Places items
d00b351 [Android] Exercice n°7 : create Place ListView
8489593 Revert "[Android] Course - create Person adapter"
b6d4ba1 [Android] Course - create Person adapter
d8d1981 [Android] Exercice 6 : ArrayAdapter with 50 items
c170159 [Android] Exercice 4 : refactor MainActivity by using ButterKnife
ee00c08 [Android] [Build] Exercice 3 : add ButterKnife to build.gradle
bb7f020 [Android] Exercice 2 - Define OnClickListener on TextView
f926277 [Android] Exercice 1 - Add rating bar to default activity and increment its value each time activity is shown
679f973 [Android] [Build] Initial default project import
3ed5b3 [Android] [Build] Configure gitignore file - from gitignore.io
```

Mon Repo Git Local

Un autre Repo Git (Bitbucket)





## 2 - Git - niveau 2

```
git push <NOM DU REMOTE> <NOM DE LA BRANCHE>
```

```
git push team master
```

```
193d3ef [master] [Android] Exercice 14 : refactor PlaceService using Retrofit
ecb943d [Android] Exercice 13 : use Otto Event Bus & create SearchResultEvent
1dc7808 [Android] Exercice 12 - Step 3 : parse result as JSON
54667b2 [Android] Exercice 12 - Step 2 : extract logic in PlaceSearchService
6809061 [Android] Exercice 12 - Step1 : use OkHttpClient to make HTTP request inside an AsyncTask
b16dfa3 [Android] Exercice 11 : add button allowing to pick a picture and display it in the Place Details Activity
69285d3 [Android] Exercice 10 : add intents to share & search a place
dffe4e [Android] Exercice 9 : add Place details Activity
040b694 [Android] Exercice 8 - Step 2 : play mp3 sound on item clicks
24ac240 [Android] Exercice 8 - Step 1 : add drawables in Places items
d00b351 [Android] Exercice n°7 : create Place ListView
8489593 Revert "[Android] Course - create Person adapter"
b6d4ba1 [Android] Course - create Person adapter
d8d1981 [Android] Exercice 6 : ArrayAdapter with 50 items
c170159 [Android] Exercice 4 : refactor MainActivity by using ButterKnife
ee00c08 [Android] [Build] Exercice 3 : add ButterKnife to build.gradle
bb7f20 [Android] Exercice 2 - Define OnClickListener on TextView
f926277 [Android] Exercice 1 - Add rating bar to default activity and increment its value each time activity is shown
679f973 [Android] [Build] Initial default project import
3ed5b3 [Android] [Build] Configure gitignore file - from gitignore.io
```

push

```
040b694 team/master [Android] Exercice 8 - Step 2 : play mp3 sound on item clicks
24ac240 [Android] Exercice 8 - Step 1 : add drawables in Places items
d00b351 [Android] Exercice n°7 : create Place ListView
8489593 Revert "[Android] Course - create Person adapter"
b6d4ba1 [Android] Course - create Person adapter
d8d1981 [Android] Exercice 6 : ArrayAdapter with 50 items
c170159 [Android] Exercice 4 : refactor MainActivity by using ButterKnife
ee00c08 [Android] [Build] Exercice 3 : add ButterKnife to build.gradle
bb7f20 [Android] Exercice 2 - Define OnClickListener on TextView
f926277 [Android] Exercice 1 - Add rating bar to default activity and increment its value each time activity is shown
679f973 [Android] [Build] Initial default project import
3ed5b3 [Android] [Build] Configure gitignore file - from gitignore.io
```

Mon Repo Git Local



Un autre Repo Git (Bitbucket)



## 2 - Git - niveau 2

`git push <NOM DU REMOTE> <NOM DE LA BRANCHE>`

`git push team master`

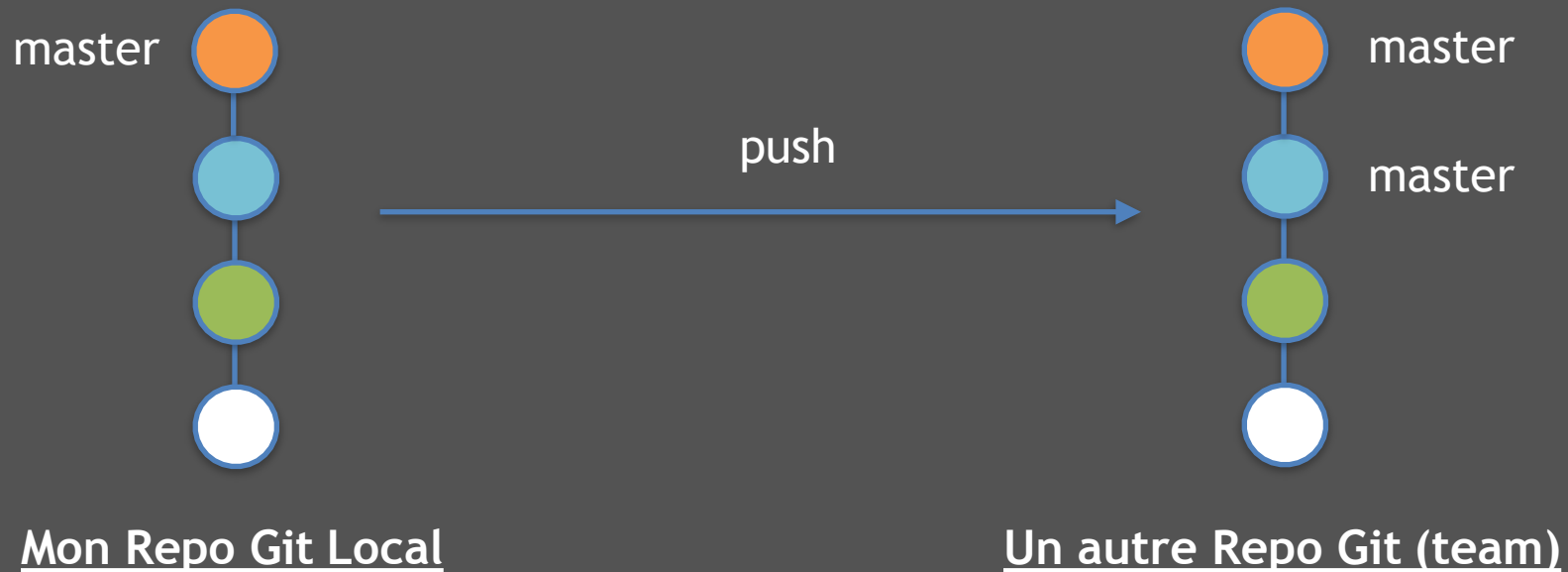
|         |  |   |
|---------|--|---|
| 192d2ef |  master  team/master | [Android] Exercice 14 : refactor PlaceService using Retrofit  |
| ecb943d |  | [Android] Exercice 13 : use Otto Event Bus & create SearchResultEvent   |
| 1dc7908 |  | [Android] Exercice 12 - Step 3 : parse result as JSON   |
| 54667b2 |  | [Android] Exercice 12 - Step 2 : extract logic in PlaceSearchService  |
| 6809061 |  | [Android] Exercice 12 - Step1 : use OkHTTP to make HTTP request inside an AsyncTask                           |
| b16dfa3 |  | [Android] Exercice 11 : add button allowing to pick a picture and display it in the Place Details Activity    |
| 69285d3 |  | [Android] Exercice 10 : add Intents to share & search a place   |
| dffbe48 |  | [Android] Exercice 9 : add Place details Activity   |
| 040b694 |  | [Android] Exercice 8 - Step 2 : play mp3 sound on item clicks   |
| 24ac240 |  | [Android] Exercice 8 - Step 1 : add drawables in Places items   |
| d00b351 |  | [Android] Exercice n°7 : create Place ListView  |
| 8489593 |  | Revert "[Android] Course - create Person adapter"   |
| b6d4ba1 |  | [Android] Course - create Person adapter  |
| d8d1981 |  | [Android] Exercice 6 : ArrayAdapter with 50 items   |
| c170159 |  | [Android] Exercice 4 : refactor MainActivity by using ButterKnife   |
| ee00c08 |  | [Android] [Build] Exercice 3 : add ButterKnife to build.gradle  |
| bb7fd20 |  | [Android] Exercice 2 - Define OnClickListener on TextView   |
| f926277 |  | [Android] Exercice 1 - Add rating bar to default activity and increment its value each time activity is shown |
| 679f973 |  | [Android] [Build] Initial default project import  |
| 3edb5b3 |  | [Android] [Build] Configure gitignore file - from gitignore.io  |



## 2 - Git - niveau 2

`git push team master`

- Cherche l'ancêtre commun (ici bleu)
- Les changements locaux doivent être au-dessus de team
- Pousse les commits locaux manquants sur team





## 2 - Git - niveau 2

`git push team master`

- Cherche l'ancêtre commun (ici bleu)
- Les changements locaux doivent être au-dessus de team



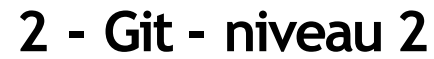


## 2 - Git - niveau 2

`git fetch team`

- Mets à jour le pointeur local team/master
- Ne change pas votre historique local (master)





- Applique tous les comits de master en partant de team/master
- Peut générer des conflits

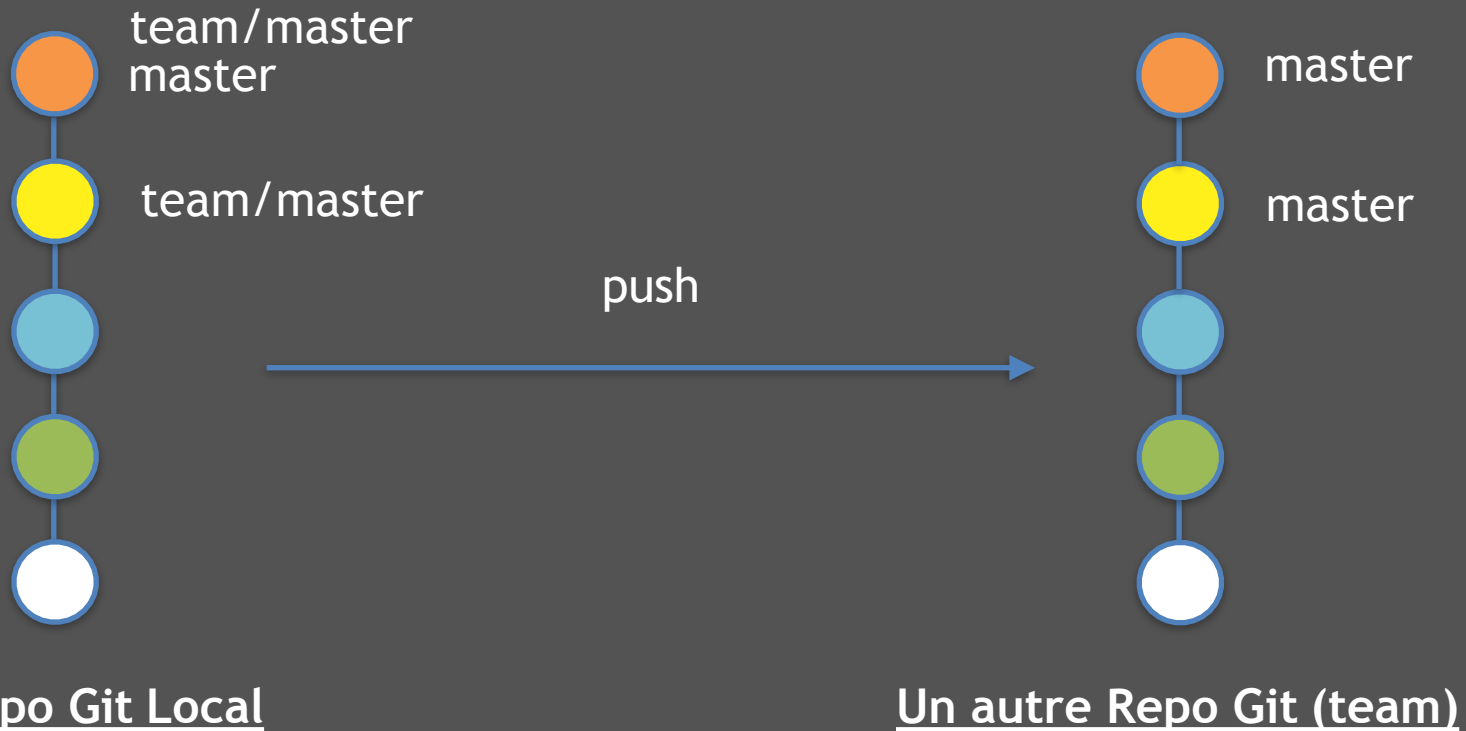




## 2 - Git - niveau 2

`git push team master`

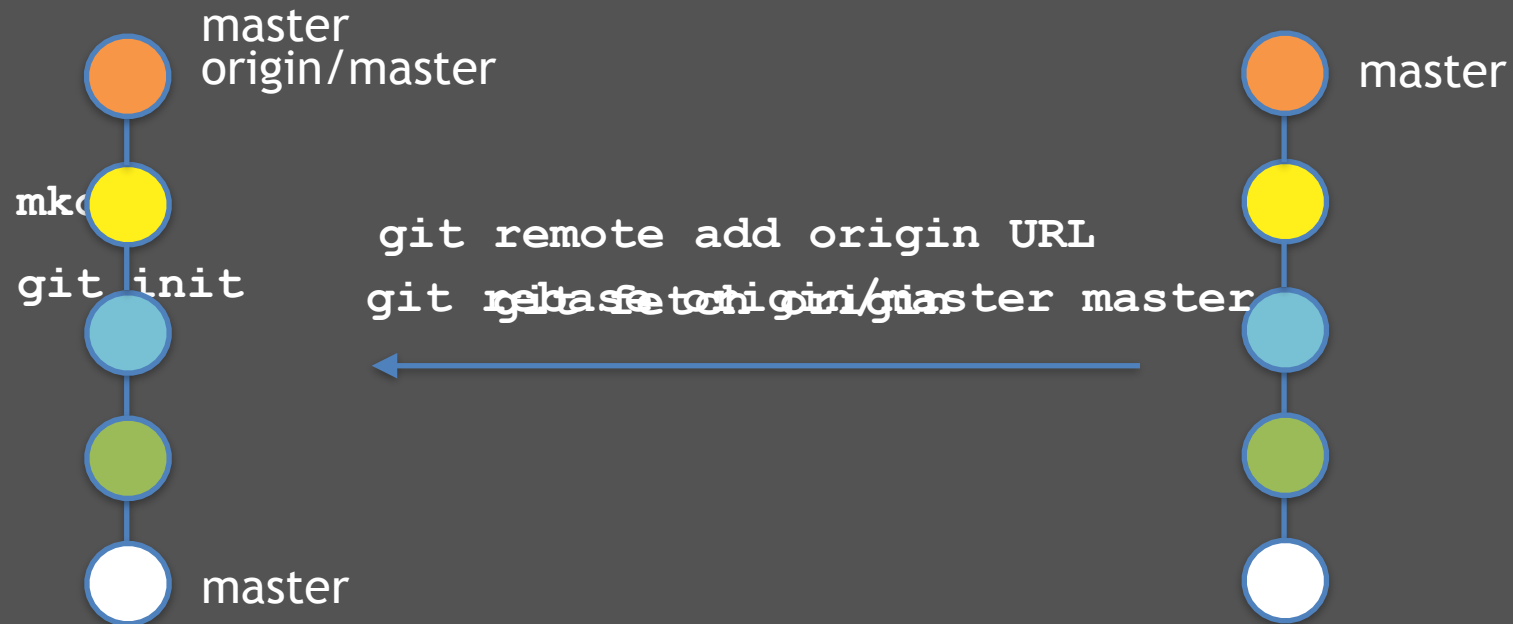
- Cherche l'ancêtre commun (ici **jaune**)
- **Les changements locaux doivent être au-dessus de team**





## 2 - Git - niveau 2

`git clone URL`



Mon Repo Git Local

Un autre Repo Git (origin)





## 2 - Git - niveau 2

### Exercice GIT

- Créer un repository vide sur Bitbucket ou Github
- Pusher vos changements actuels
- À partir de maintenant, pensez à pusher sur origin après chaque exercice

# LES 3 CHOSES À GARDER EN TÊTE

1. Je fais mes modifs sur mon repo local
  2. Fetch va chercher les changements, rebase les intègre
  3. Je pushe régulièrement au cas où mon ordi prend feu
- git add remote, fetch, rebase, push, clone**

NOU  
VELLES  
INTER  
FACES

### 3 - Requêtes HTTP : les bases



MOB  
ILE  
APPS



## 3 - Requêtes HTTP : les bases

Vos Apps vont avoir besoin d'utiliser des services distants (WebServices)

- Flux RSS : fichiers XML
- SOAP : compliqué mais fonctionnel
- Le plus répandu : Serveurs REST



## 3 - Requêtes HTTP : les bases

Serveurs REST : le REUH et le STEUH

- REUH = Resources (tout est basé sur des ressources HTTP accédées via des requêtes)
- STEUH = Stateless (les appels ne dépendent pas des appels précédents)



## 3 - Requêtes HTTP : les bases

Serveurs REST : concrètement c'est quoi

- Un serveur (php, Node.js, python)
- Qui fournit des APIs HTTP
  - GET sur <http://monserver.com/fiches> retourne toutes les fiches
  - GET sur <http://monserver.com/fiches/2> retourne la fiche d'id 2
  - POST sur <http://monserver.com/fiches/2> permet d'éditer la fiche 2
  - DELETE sur <http://monserver.com/fiches/2> permettre de supprimer la fiche 2
- Boite noire : OSEF de la logique serveur (traitement BD, caching, algorithmes)



## 3 - Requêtes HTTP : les bases

Exemple de serveur REST :

- GET <https://api-adresse.data.gouv.fr/search/?q=<MA RECHERCHE>>

```
JSON
├── attribution : "BAN"
├── features
│   └── 0
│       ├── type : "Feature"
│       ├── geometry
│       └── properties
│           ├── city : "Paris"
│           ├── type : "street"
│           ├── context : "75, Paris, Île-de-France"
│           ├── id : "75115_XXXX_a00a13"
│           ├── name : "Place du Commerce"
│           ├── importance : 0.3926
│           ├── postcode : "75015"
│           ├── label : "Place du Commerce 75015 Paris"
│           ├── score : 0.8538727272727272
│           ├── x : 648103.9
│           ├── citycode : "75115"
│           └── y : 6860818
```



## 3 - Requêtes HTTP : les bases

OkHttp : framework Android pour effectuer des requêtes HTTP simplement



```
OkHttpClient okHttpClient = new OkHttpClient();
Request request = new Request.Builder()
    .url("https://api-adresse.data.gouv.fr/search/?q=Place%20du%20commerce")
    .build();

try {
    Response response = okHttpClient.newCall(request).execute();
} catch (IOException e) {
    // Silent catch, no places will be displayed
}
```





@

## Exercice n° 12

- Installer OkHttp <https://github.com/square/okhttp>
- Dans le onResume() de la MainActivity, lancer une requête GET vers <https://api-adresse.data.gouv.fr/search/?q=Place%20du%20commerce>
- L'application Crashe... Chercher l'erreur sur internet

## 4 - Asynchronicité et parallélisme

MOB  
ILE  
APPS





## 4 - Asynchronicité et parallélisme

L'accès au Thread UI Android est bien gardé !

- Une seule personne à la fois (système de lock : sémaphore)
- Tous les autres Threads qui en ont besoin sont en attente





## 4 - Asynchronicité et parallélisme

Conséquence : **le Thread UI, c'est pour l'UI !**

- Réseau interdit (NetworkOnMainThreadException)
- Éviter à tout prix les calculs non-liés à l'UI (requêtes SQL...)





## 4 - Asynchronicité et parallélisme

Conséquence :

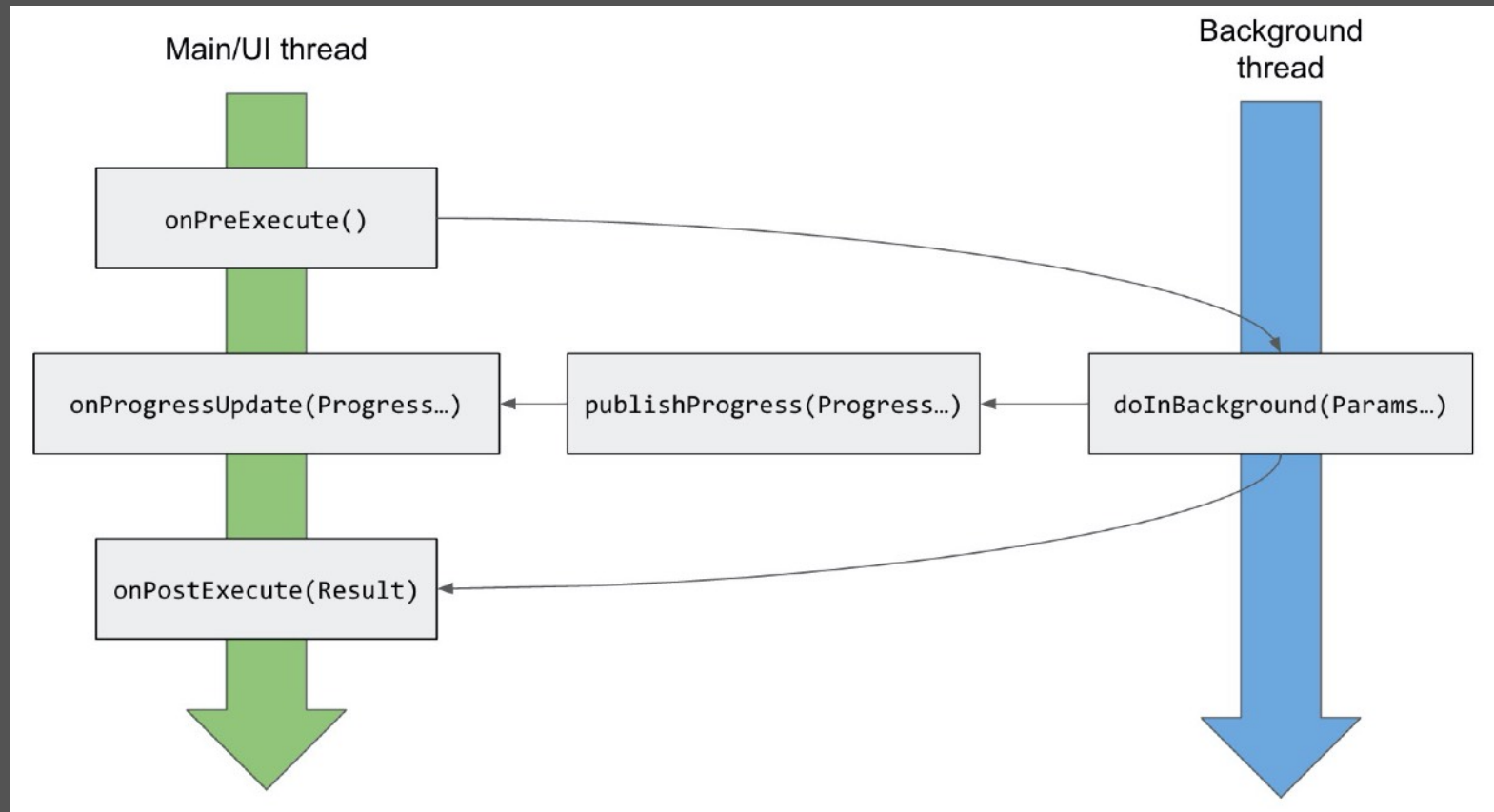
- On va devoir effectuer certains traitements hors Thread UI
- Et donc créer des nouveaux threads
- Comment faire ça proprement (nettoyage notamment) ?





## 4 - Asynchronicité et parallélisme

Les AsyncTask : Here, There, And Back Again !







## 4 - Asynchronicité et parallélisme

```
AsyncTask<String, Void, Response> asyncTask = new AsyncTask<String, Void, Response>() {

    @Override
    protected Response doInBackground(String... params) {
        // Here we are in a new background thread
        try {
            final OkHttpClient okHttpClient = new OkHttpClient();
            final Request request = new Request.Builder()
                .url(params[0])
                .build();
            return okHttpClient.newCall(request).execute();
        } catch (IOException e) {
            // Silent catch, no places will be displayed
        }
        return null;
    }

    @Override
    protected void onPostExecute(Response response) {
        super.onPostExecute(response);

        // Here we are in caller Thread
    }
};

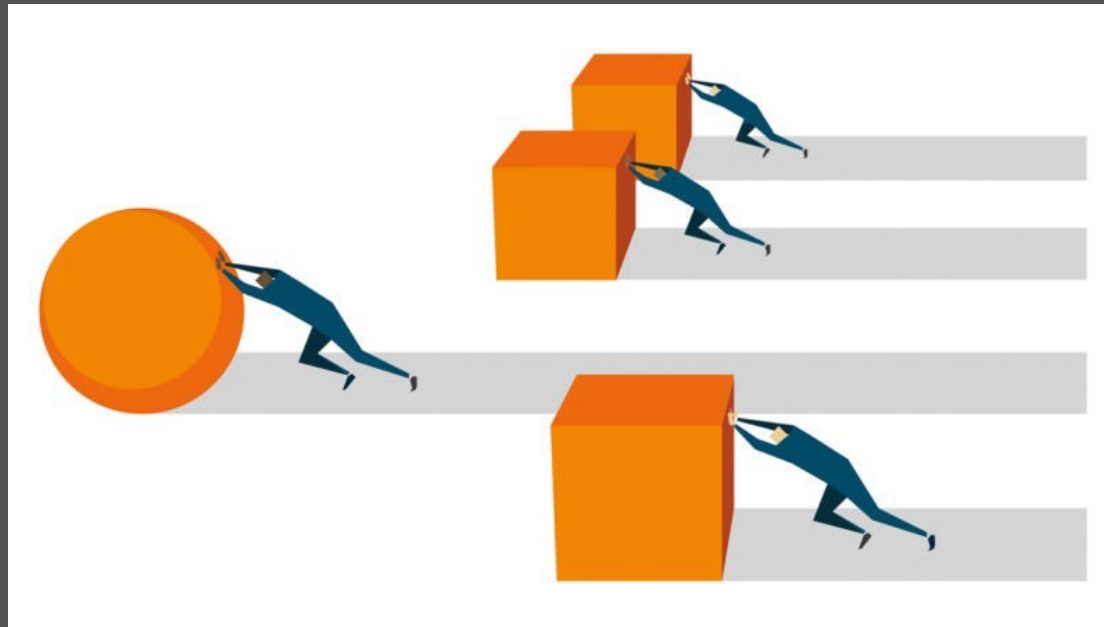
asyncTask.execute("http://google.fr");
```



## 4 - Asynchronicité et parallélisme

Pour résumer : si vous souhaitez faire une application performante

- Tout ce qui peut être fait en background doit être fait en background
- Les AsyncTasks permettent de facilement exécuter du code en background
- Ne pas oublier l'utilisateur : Loaders et feedback en attendant







## 4 - Asynchronicité et parallélisme

### Exercice n° 12 (le vrai cette fois)

- Dans un AsyncTask, lancez une requête GET sur <https://api-adresse.data.gouv.fr/search/?q=Place%20du%20commerce>
- Afficher le résultat dans le log Android avec la méthode Log.d()
- Extraire la logique dans une classe dédiée (PlaceSearchService)
- Parser le body du résultat en JSON (avec l'api JSONObject)
- Loguer l'attribut « label » de chaque lieu trouvé

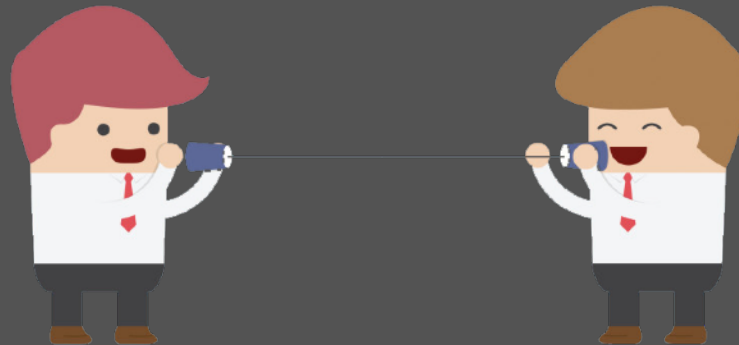
```
01-05 15:11:25.800 9494-9531/org.miage.placesearcher D/RECEIVED PLACE: Place du Commerce 75015 Paris
01-05 15:11:25.800 9494-9531/org.miage.placesearcher D/RECEIVED PLACE: Place du Commerce 14000 Caen
01-05 15:11:25.800 9494-9531/org.miage.placesearcher D/RECEIVED PLACE: Place du Commerce 44000 Nantes
01-05 15:11:25.800 9494-9531/org.miage.placesearcher D/RECEIVED PLACE: Place du Commerce 44600 Saint-Nazaire
01-05 15:11:25.800 9494-9531/org.miage.placesearcher D/RECEIVED PLACE: Place du Commerce 70000 Vesoul
```



## 4 - Asynchronicité et parallélisme

Ca fonctionne mais...

- Comment prévenir notre activité qu'on a le résultat de la requête ?



## 5 - Évènements





## 5 - Évènements

Quel Design Pattern permet d'avoir des objets qui écoutent d'autres objets ?

Design Pattern Observable



## 5 - Évènements

### Design Pattern Observable



- Storm1 connaît Hux
- Storm2 connaît Hux
- Hux connaît Storm1
- Hux connaît Storm2



Notifies

Listens

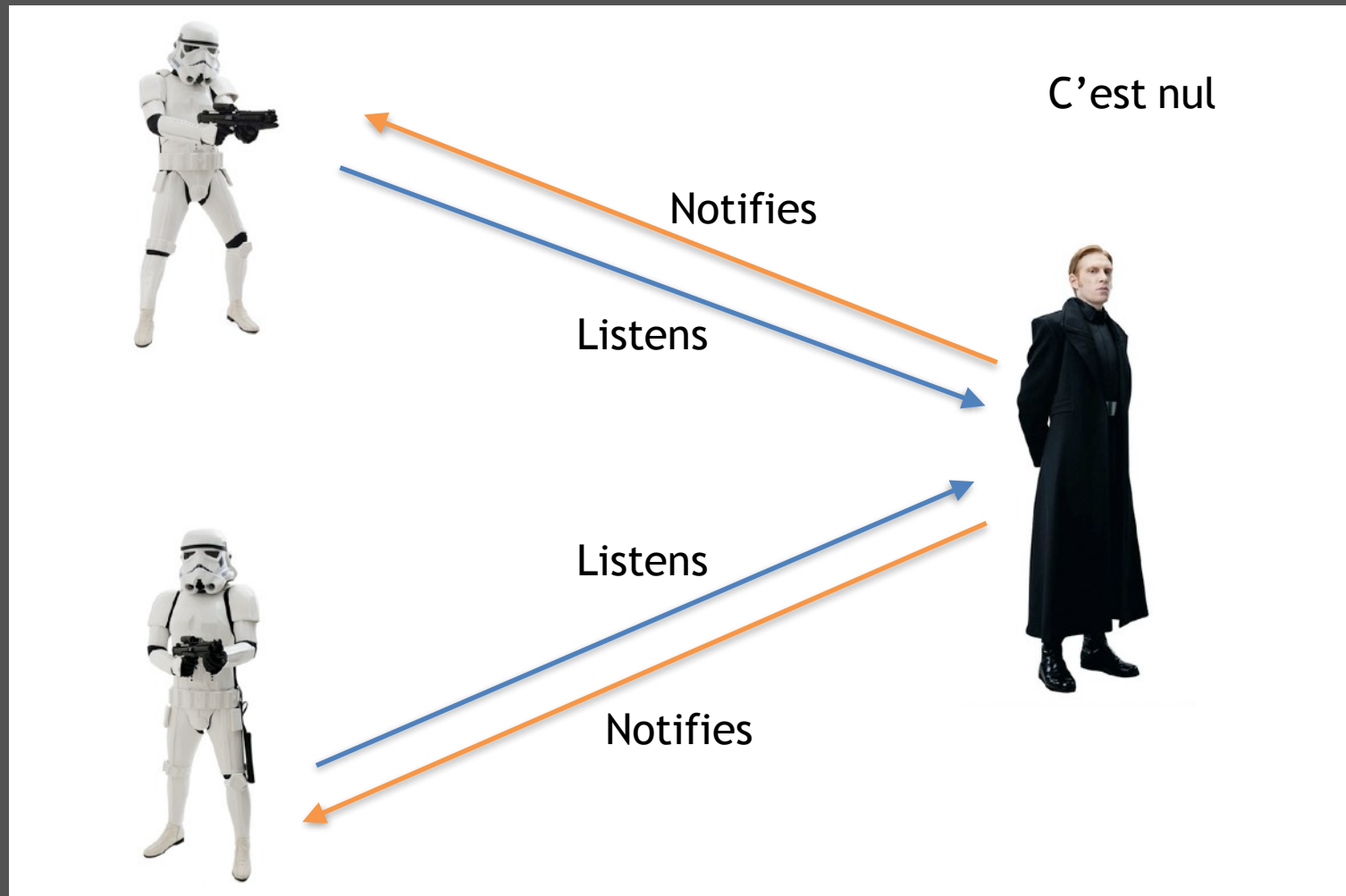
Listens

Notifies



## 5 - Évènements

### Design Pattern Observable





## 5 - Évènements

Design Pattern Observable : pourquoi c'est nul ?

- Les listeners connaissent l'Observable
- L'Observable connaît tous ses listeners

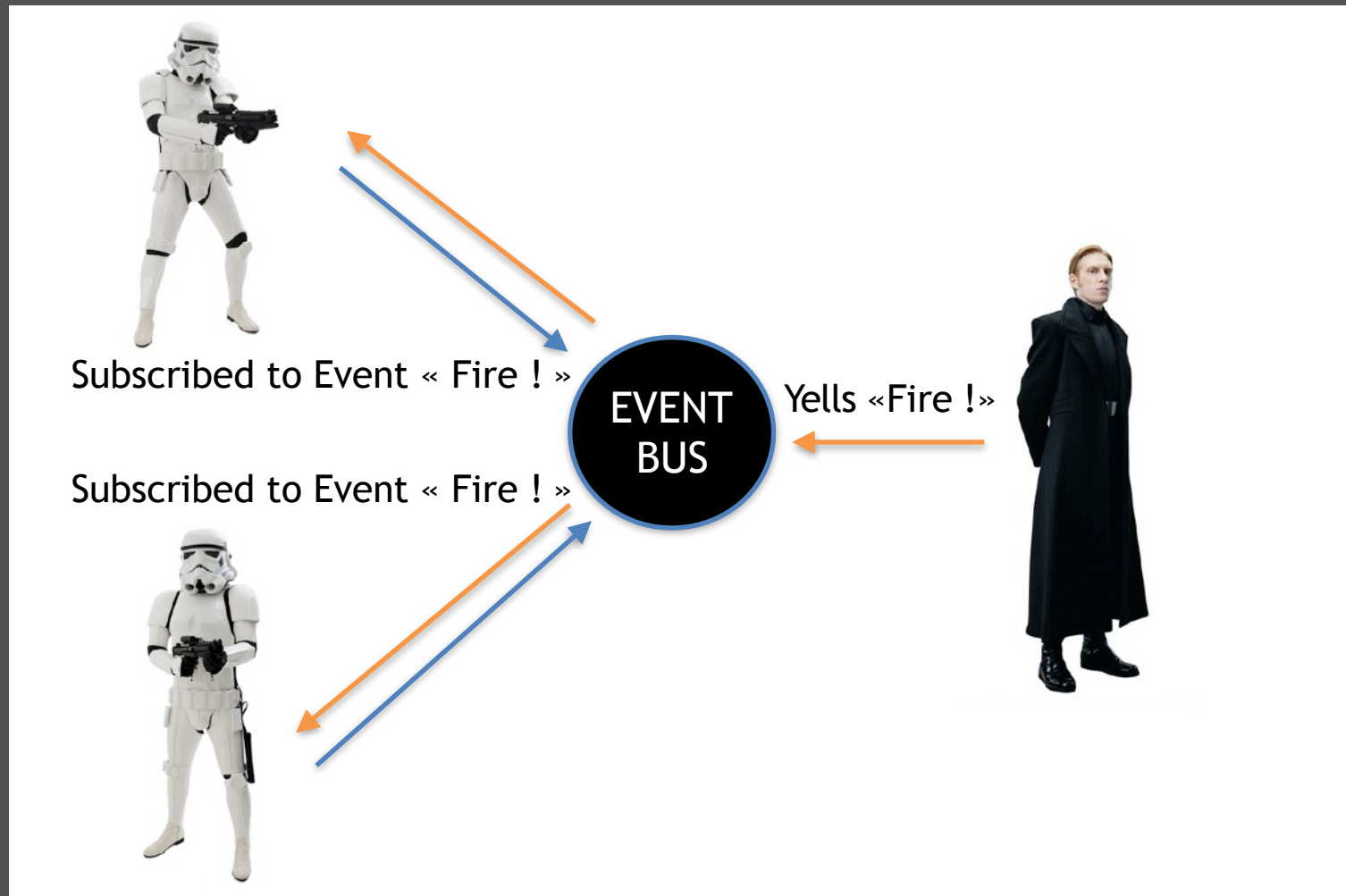
Dans notre contexte asynchrone

- Le listener = L'activité
- Au moment où on veut notifier l'Activité
  - elle est peut être déjà morte
  - donc problème Thread UI
- Pose également des problèmes mémoire (Garbage Collector)



## 5 - Évènements

### Design Pattern EventBus







## 5 - Évènements

### Design Pattern EventBus

- Les listeners écoutent un évènement (et plus un objet)
- L'émetteur crie un évènement sans avoir à savoir qui écoute
- Tous les listeners intéressés sont notifiés

Dans notre contexte asynchrone

- Plus de couplage fort entre les objets
- Plus de problème de cycle de vie



## 5 - Évènements



### EventBus sur Android : Otto

- Créer des évènements sous forme de classe
  - donc potentiellement envoyer des données
- Envoyer des évènements
- Recevoir des évènements (annotations)
- S'inscrire / Se désinscrire du bus



## 5 - Évènements

Etape 1 : créer notre Bus d'évènement

```
public class EventBusManager {  
    public static Bus BUS = new Bus(ThreadEnforcer.ANY);  
}
```

Etape 2 : créer un évènement

```
public class SearchResultEvent {  
    private List<Place> places;  
  
    public SearchResultEvent(List<Place> places) {  
        this.places = places;  
    }  
  
    public List<Place> getPlaces() {  
        return places;  
    }  
}
```



## 5 - Évènements

### Etape 2 : créer un évènement

```
public class SearchResultEvent {  
    private List<Place> places;  
  
    public SearchResultEvent(List<Place> places) {  
        this.places = places;  
    }  
  
    public List<Place> getPlaces() {  
        return places;  
    }  
}
```

### Etape 3 : Poster cet évènement

```
EventManager.BUS.post(new SearchResultEvent(myPlaces));
```



## 5 - Évènements

### Etape 4 : Inscrire/Désinscrire l'activité

- Une idée du bon endroit pour s'inscrire ?
  - Sur le onResume()
- Une idée du bon endroit pour se désinscrire ?
  - Sur le onPause()

```
@Override
protected void onResume() {
    // Do NOT forget to call super.onResume()
    super.onResume();

    EventBusManager.BUS.register( object: this);
}
```

```
@Override
protected void onPause() {
    EventBusManager.BUS.unregister( object: this);

    super.onPause();
}
```



## 5 - Évènements

### Etape 5 : écouter et réagir à des évènements

- @Subscribe devant n'importe quelle méthode
- Le paramètre doit être un évènement
- La méthode sera appelée à chaque fois que quelqu'un poste cet évènement

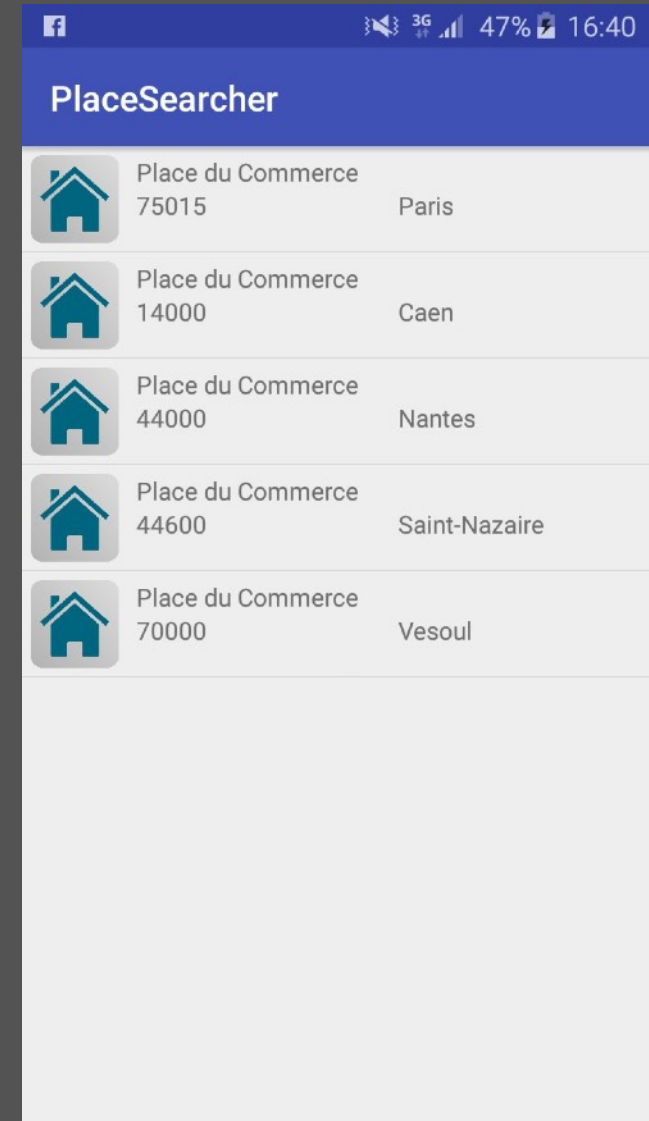
```
@Subscribe
public void searchResult(SearchResultEvent event) {
    for(Place place : event.getPlaces()) {
        // Do something with the place
    }
}
```



## 5 - Évènements

### Exercice n° 13

- Installer le framework Otto <http://square.github.io/otto/>
- Créer votre BUS partagé
- Modifier le PlaceSearchService pour :
  - Poster un « SearchResultEvent »
  - Cet évènement doit stocker une liste de Places
- Modifier la Main Activity pour
  - Se mettre à écouter l'EventBus au bon moment
  - Arrêter d'écouter l'EventBus au bon moment
  - Être prévenu de l'envoi de « SearchResultEvent »
  - À la réception d'un SearchResultEvent :
    - Mettre à jour l'adapter de la liste
    - Méthodes de l'adapter : clear() et addAll()





## 5 - Évènements

Ca fonctionne mais...

- OkHTTP c'est pratique mais encore beaucoup de chose à gérer à la main
- Traitement de tous les cas d'erreurs (errorCode, body null, réponse au mauvais format...)
- Parser le JSON à la main c'est pénible, error-prone, peu maintenable
- Écrire des AsyncTask pour chaque requête va être fastidieux



# THAT'S ALL FOLKS !

## NEXT TIME

- Retrofit
- Google Maps
- Bases de données

## D'ICI LÀ

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

