DÉVELOPPEMENT D'APPLICATIONS MOBILES

Alex Morel - iRéalité

@alex_morel_

iREALiTE

Programme du cours :

1. Bases Android
2. Intents, HTTP, Asynchronicité, Évènements
3. Retrofit, Google Maps
4. Base de données mobiles, Notifications
5. TP Encadré

Développement d'applicationsmobiles
Cours 3

Retrofit : make REST servers great again

Git - niveau 3 : forks et pull requests

Mode debug : breakpoints conditionnels / d'exceptions

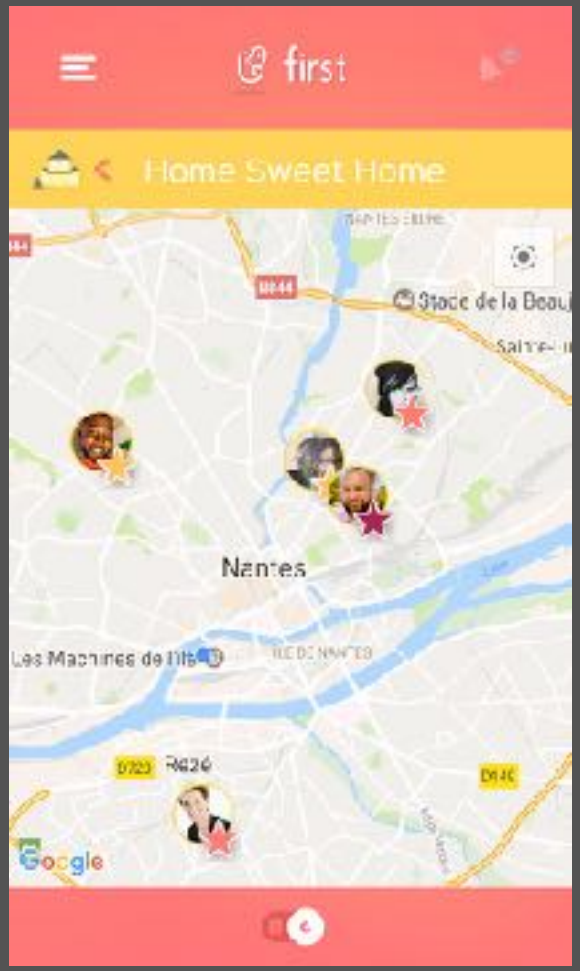
Google Maps : branchement

Nos objectifs du jour :

1. Google maps : marqueurs
2. Git - niveau 4 : régler les conflits & cool stuff
3. Base de données mobiles
4. Notifications avec Firebase Cloud Messaging

1 - Google Map sur Android

1 - Google Map sur Android

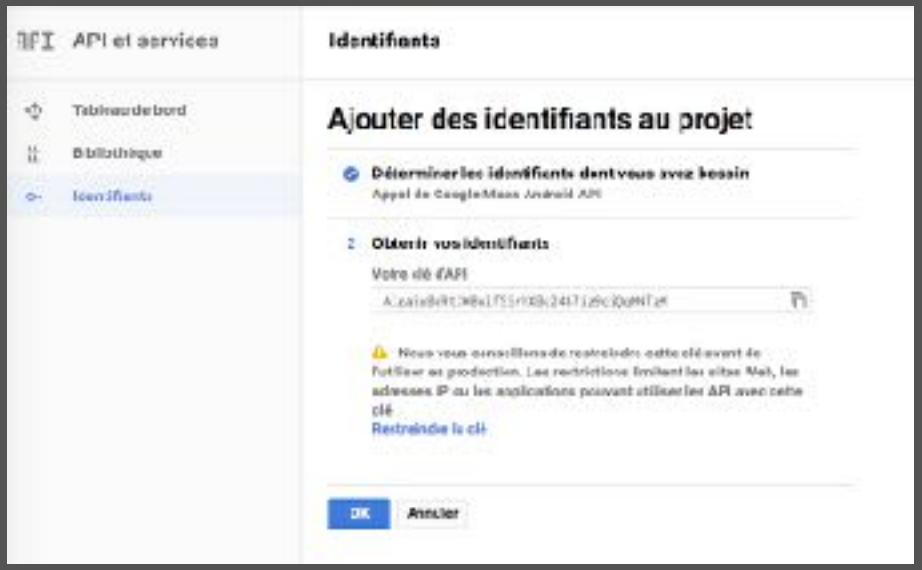


- Framework multi-plateforme pour afficher des cartes

Sur Android :

- Personnalisable à 100%
  - UI disponible (zoom, boutons)
  - Type de carte (vue satellite, mixte…)
  - Contrôles tactiles (pinch rotation, zoom…)
  - Position de l'utilisateur
- Facile d'accès pour des besoins simples
- Peu très vite demander de bien connaitre Android
  - Passage à l'échelle (clusters, asynchronicité)
  - Customisation (drawables & bitmaps…)

1 - Google Map sur Android



<u>Étape 1 : obtenir une clé</u>
créer un compte et une clé API sur la Google Developer Console

1 - Google Map sur Android

Étape 2 : préparer le build

Modifier le build.gradle

```
// Add google maps
compile 'com.google.android.gms:play-services-maps:11.8.0'
```

inclure la clé dans le Manifeste
pour l'exercice : AIzaSyBcRt3W8uifSjrhXBu2467iz9ciQqMNTzM

```
<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">


    <!-- Google Maps API Key -->
    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyBcRt3W8uifSjrhXBu2467iz9ciQqMNTzM"/>
```

1 - Google Map sur Android

Étape 3 : ajouter des cartes à vos vues

Ajouter un MapFragment à votre layout (on pourra reparler des Fragments plus tard)

```
    <fragment
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment" />
</LinearLayout>
```

1 - Google Map sur Android

Étape 4 : obtenir la map dans l'activité

Récupérer le fragment dans votre activité et obtenir la Map

```
// Get map fragment
SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.map);
mapFragment.getMapAsync( onMapReadyCallback: this);
```

Votre Activity doit implémenter OnMapReadyCallback

```
MapActivity extends … implements OnMapReadyCallback {
```

Et la méthode correspondante

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mActiveGoogleMap = googleMap;
```

1 - Google Map sur Android

Exercice 15

- Ajoutez un bouton pour switcher de list à map dans la MainActivity

•Créer une MapActivity contenant un MapFragment et un champ texte (vous pouvez copier-coller puis modifier activity_main.xml)

- Faites en sorte qu'on puisse passer de la MapActivity à la MainActivity
  - sans perdre la valeur du champ texte de recherche
  - faire de même de la MainActivity à la MapActivity

- Faire une pull request pour proposer votre exercice

- Optionnel : modifier les paramètres de la carte
  - Obtenir la carte avec getMapAsync() et onMapReady()
  - Changer le type de carte : satellite
  - Activer les boutons permettant de modifier le niveau de zoom
  - Consulter la doc Google pour voir les paramètres disponibles

1 - Google Map sur Android

Vous avez la main sur les marqueurs :

- position
- titre et snippet (affichés dans l'infoWindow)
- apparence (icône voir même vues complexes)
- onClick : sur le marqueur et/ou l'infoWindow

```
MarkerOptions markerOptions = new MarkerOptions()
        .position(new LatLng( v: 49.5,  v1: -1.5))
        .title("Place du commerce")
        .snippet("Une bien fameuse place");
Marker marker = mActiveGoogleMap.addMarker(markerOptions);
```

```
mActiveGoogleMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {
    @Override
    public void onInfoWindowClick(Marker marker) {
        PlaceAddress associatedPlace = mMarkersToPlaces.get(marker.getId());
        if (associatedPlace != null) {
            // Do something with the place
        }
    }
});
```

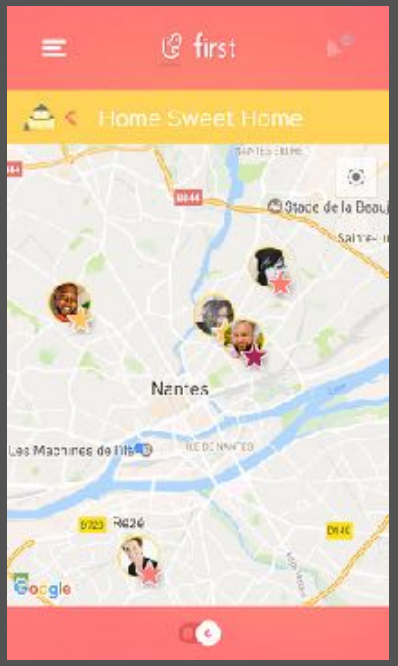
1 - Google Map sur Android

- Mouvements de caméra : devenez le Darren Aronofsky des cartes !

```
mActiveGoogleMap.animateCamera(CameraUpdateFactory.newLatLngBounds(myBounds, width, height, padding));
```

Vous avez la main sur tout le reste :

- Listeners de région visible (pour comportements complexes e.g. live updates)
- Création de clusters (pour la montée en charge et l'user experience)
- Affichages complexes (itinéraires KML, formes GeoJSON…)



1 - Google Map sur Android

Exercice 16

- Modifier le modèle : stocker la latitude/longitude des points retournés
- Faire en sorte que la MapActivity affiche les Place avec des marqueurs
- Au clic sur l'info window d'un marqueur, lancer la PlaceDetailActivity
- Faire une pull request pour proposer votre exercice
- Optionnel : centrer la caméra sur les résultats (cf LatLngBounds)
- Optionnel : fournir vos propres icônes de marqueurs
- Optionnel :
  - Vous devez constater que la recherche entraine des ralentissements
  - Comprendre pourquoi et m'appeler
  - Si votre explication est validée, mettre en place un correctif

•A faire chez vous (optionnel) : créer des clusters pour regrouper les marqueurs https://developers.google.com/maps/documentation/android-api/utility/marker-clustering

1 - Google Map sur Android

Pour éviter d'effectuer trop souvent la même tâche, le Scheduler est un excellent outil

```
public void searchPlacesFromAddress(final String search) {
    // Cancel last scheduled network call (if any)
    if (mLastScheduleTask != null && !mLastScheduleTask.isDone()) {
        mLastScheduleTask.cancel( b: true);
    }

    // Schedule a network call in REFRESH_DELAY ms
    mLastScheduleTask = mScheduler.schedule(new Runnable() {
```

2 - Git - niveau 4

MOB
ILE
APPS

2 - Git - niveau 4

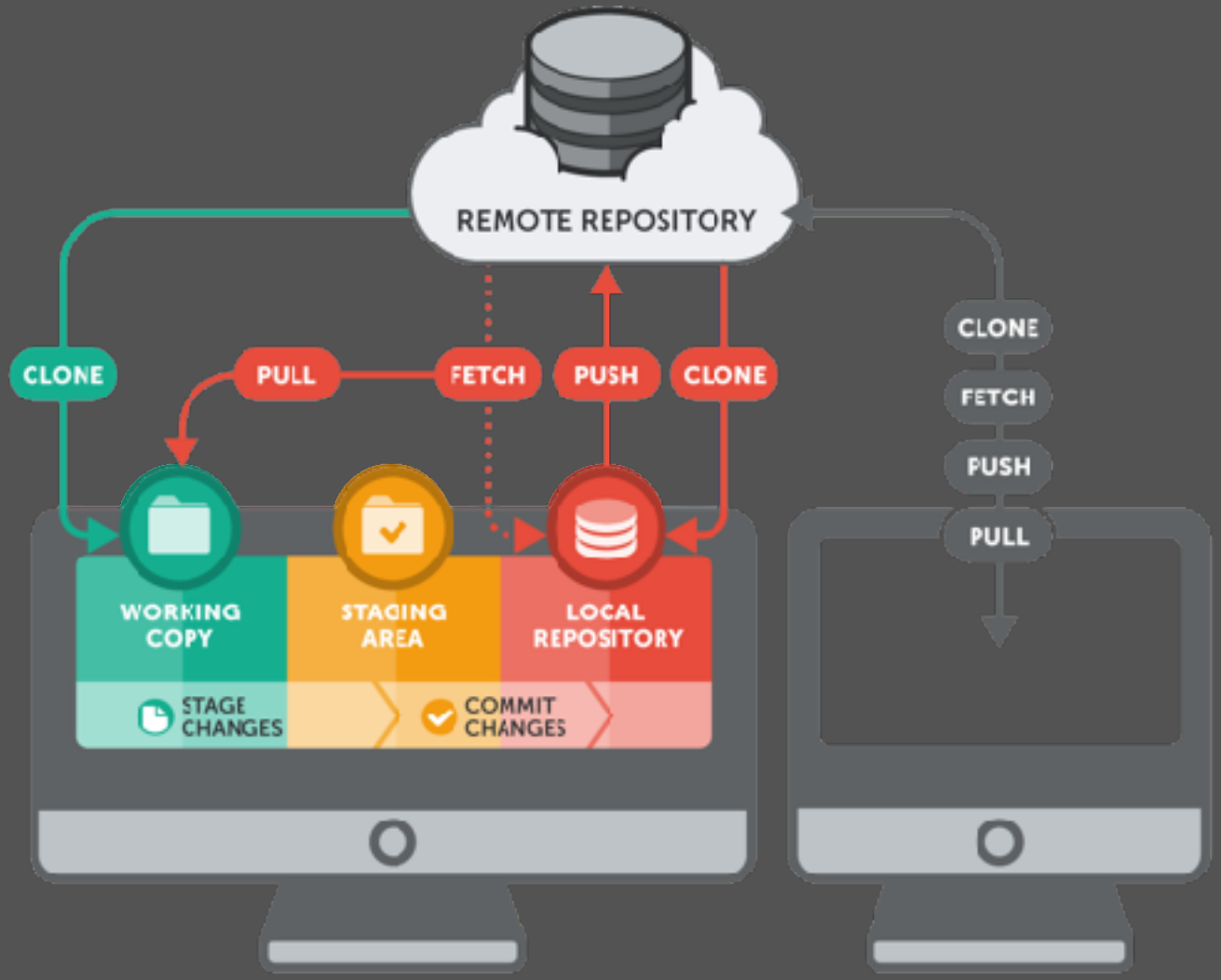
NE TRAVAILLEZ JAMAIS SANS REPO GIT

COMMITEZ PETIT POUR LES CONFLITS

COMMITEZ TOUT LE TEMPS

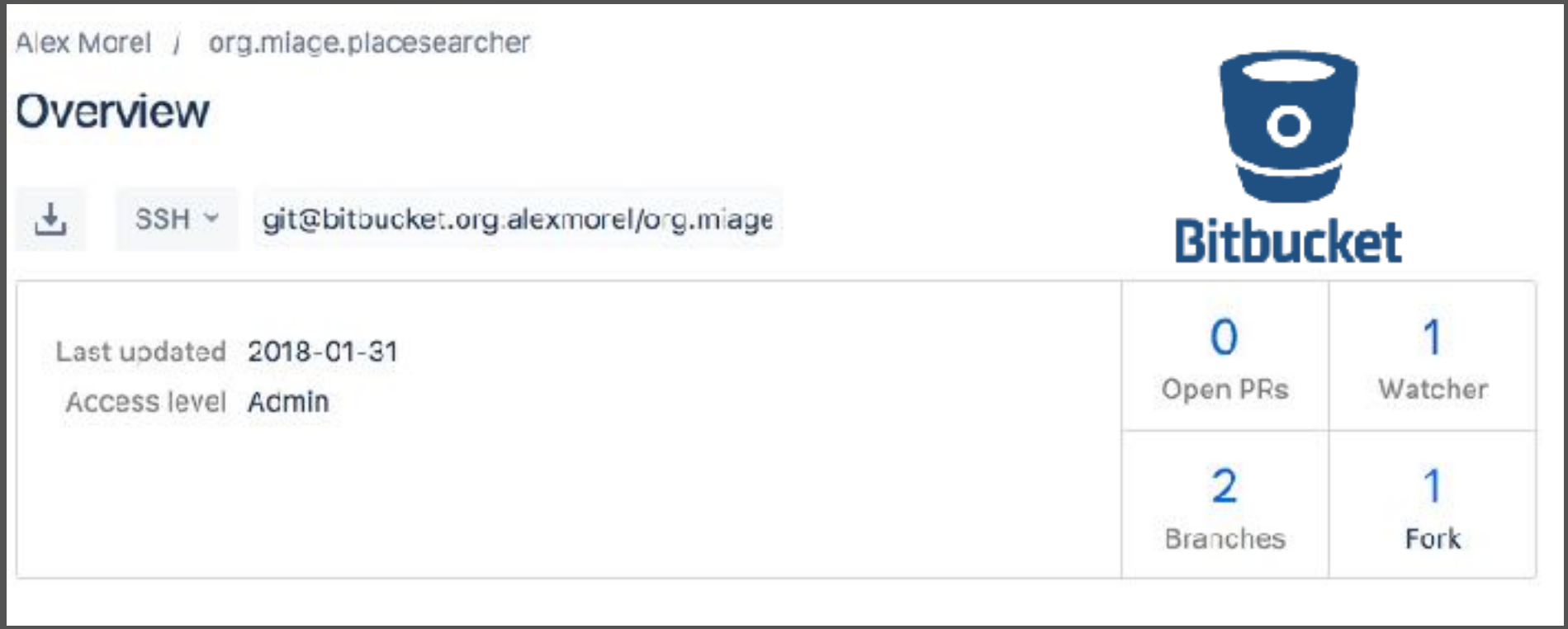
2 - Git - niveau 4

2 - Git - niveau 4



2 - Git - niveau 4

Bitbucket : serveur git pour repositories publics & privés



2 - Git - niveau 4

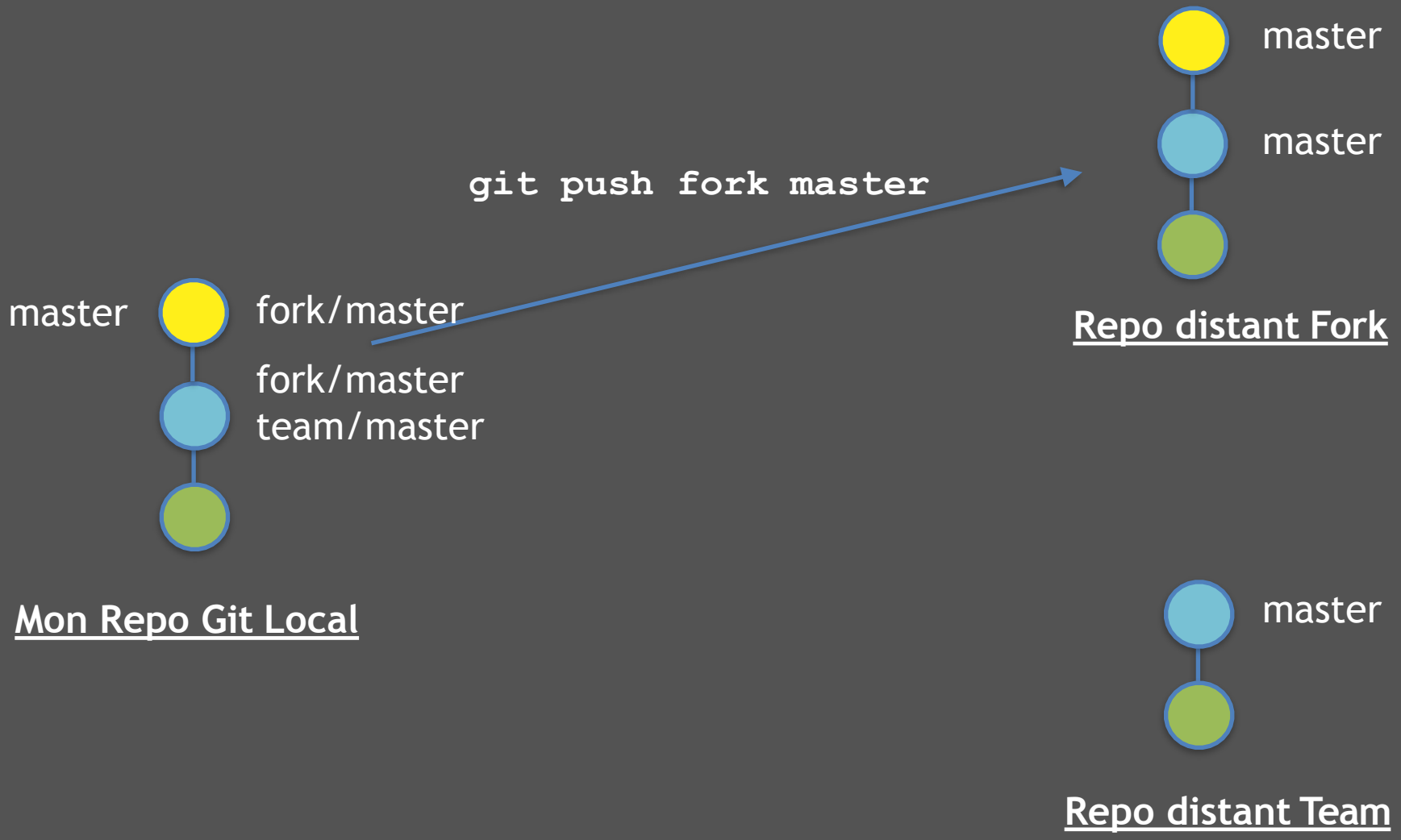
Repo distant Fork

master

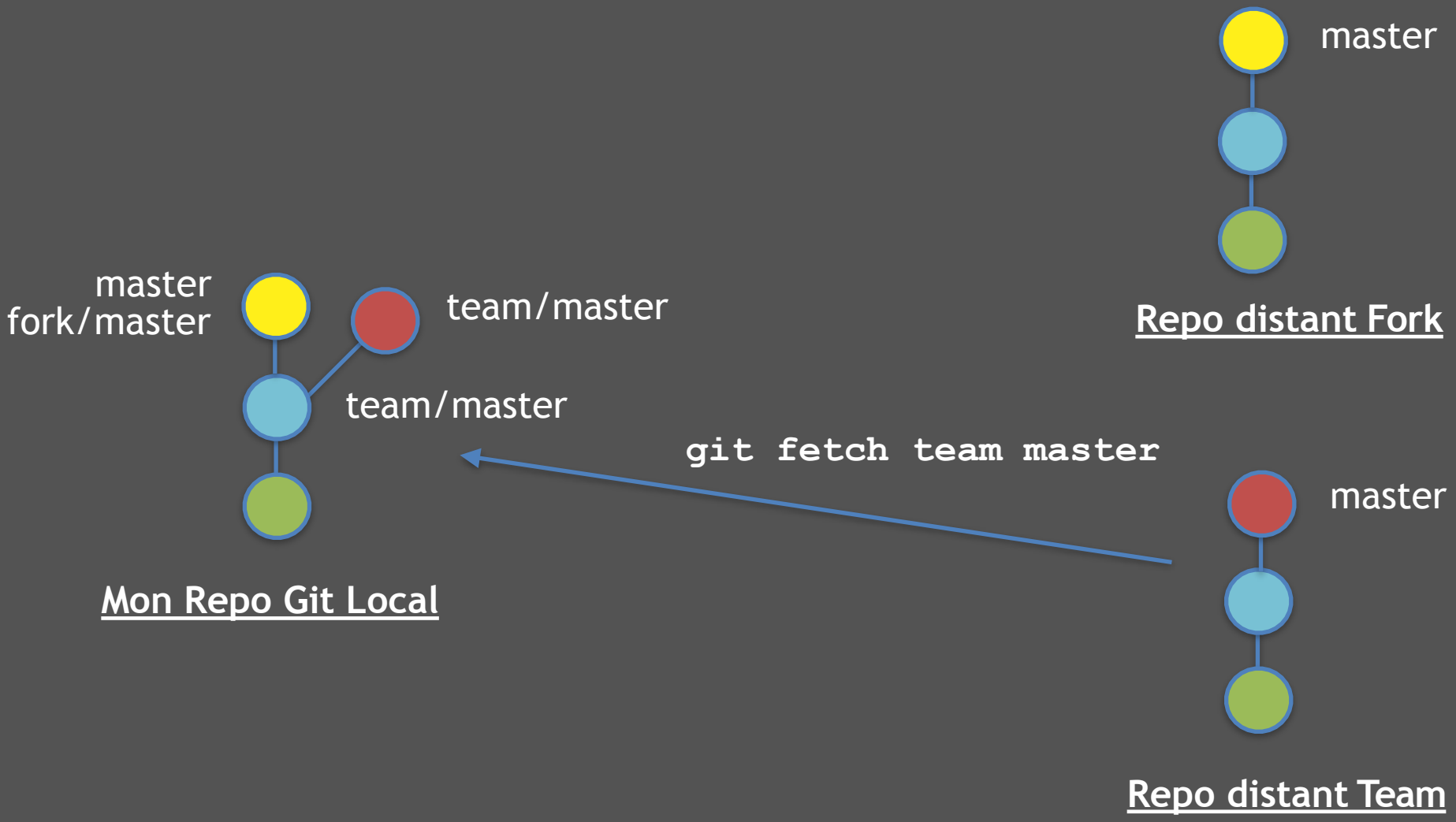
Mon Repo Git Local

Repo distant Team

2 - Git - niveau 4



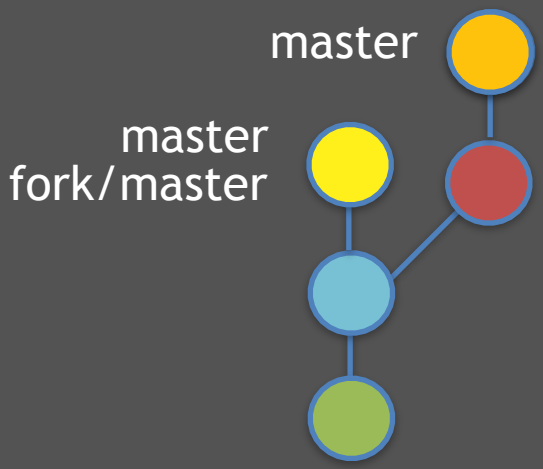
2 - Git - niveau 4



2 - Git - niveau 4

```
git rebase team/master master
```

Pas le même parent
-> pas le même sha1
-> pas le même commit



Mon Repo Git Local

Repo distant Fork

Repo distant Team

2 - Git - niveau 4

```
git push -f fork master
```

master
fork/master
fork/master
team/master

Mon Repo Git Local

master
master

Repo distant Fork

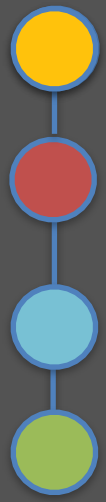
master

Repo distant Team

2 - Git - niveau 4

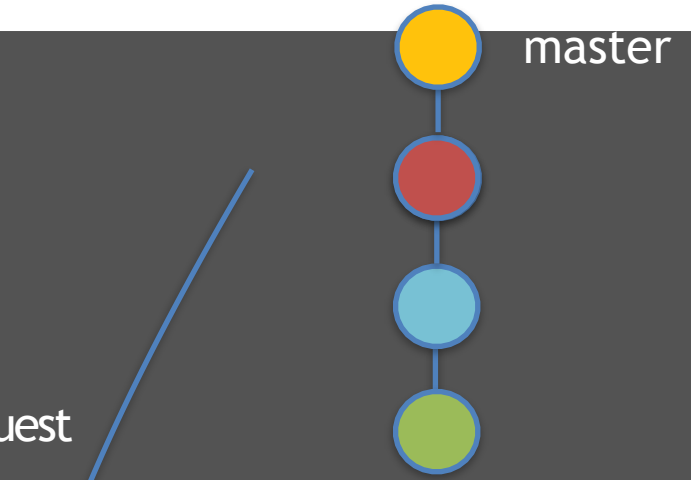
team/master
master
fork/master

team/master

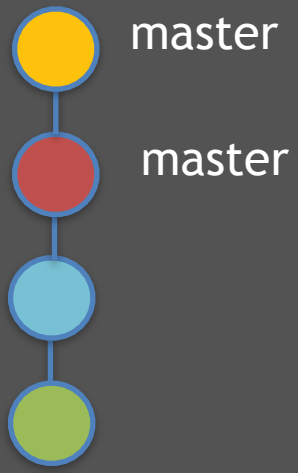


Mon Repo Git Local

Fait une pull request



Repo distant Fork



Repo distant Team

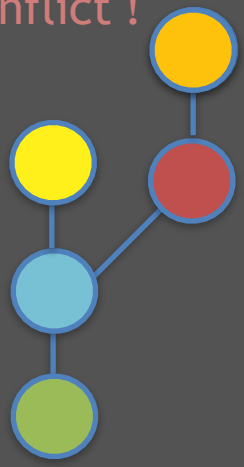
2 - Git - niveau 4

```
git rebase team/master master
```

Repo distant Fork

conflict !

Les commits rouge et jaune
modifie le même code



Mon Repo Git Local

Repo distant Team

2 - Git - niveau 4

- Alexis a fait une pull request pour toujours chercher en lowercase
- J'ai un commit qui retire les espaces de la recherche
- Je fetch, je rebase -> conflit

```
alexmorel@~/Documents/Git/org.miage.placesearcher (correction $) $ git rebase perso/correction correction
First, rewinding head to replay your work on top of it...
Applying: Make search without space
Using index info to reconstruct a base tree...
M       app/src/main/java/org/miage/placesearcher/PlaceSearchService.java
Falling back to patching base and 3-way merge...
Auto-merging app/src/main/java/org/miage/placesearcher/PlaceSearchService.java
CONFLICT (content): Merge conflict in app/src/main/java/org/miage/placesearcher/PlaceSearchService.java
error: Failed to merge in the changes.
Patch failed at 0001 Make search without space
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
```

2 - Git - niveau 4

- Alexis a fait une pull request pour toujours chercher en lowercase
- J'ai un commit qui retire les espaces de la recherche
- Je fetch, je rebase -> conflit

```
alexmorel@~/Documents/Git/org.miage.placesearcher (correction *+$|REBASE 1/1) $ git status
rebase in progress; onto 69361d4
You are currently rebasing branch 'correction' on '69361d4'.
  (fix conflicts and then run "git rebase --continue")
  (use "git rebase --skip" to skip this patch)
  (use "git rebase --abort" to check out the original branch)

Unmerged paths:
  (use "git reset HEAD <file>..." to unstage)
  (use "git add <file>..." to mark resolution)

        both modified:   app/src/main/java/org/miage/placesearcher/PlaceSearchService.java

no changes added to commit (use "git add" and/or "git commit -a")
alexmorel@~/Documents/Git/org.miage.placesearcher (correction *+$|REBASE 1/1) $
```

2 - Git - niveau 4

- Alexis a fait une pull request pour toujours chercher en lowercase
- J'ai un commit qui retire les espaces de la recherche
- Je fetch, je rebase -> conflit

```
                searchPlacesFromDB(search);

                // Step 2 : Call to the REST service
+ +<<<<<<<HEAD
  +             mPlaceSearchRESTService.searchForPlaces(search.toLowerCase()).enqueue(new Callback<PlaceSearchResult>() {
+ +=======
+               mPlaceSearchRESTService.searchForPlaces(search.replace(" ", "")).enqueue(new Callback<PlaceSearchResult>() {
+ +>>>>>>> Make search without space
                    @Override
                    public void onResponse(Call<PlaceSearchResult> call, Response<PlaceSearchResult> response) {
                        // Post an event so that listening activities can update their UI
```

2 - Git - niveau 4

- Alexis a fait une pull request pour toujours chercher en lowercase
- J’ai un commit qui retire les espaces de la recherche
- Je fetch, je rebase -> conflit
- Je corrige le conflit en réécrivant ma modification à partir de son code

```
// Step 2 : Call to the REST service
mPlaceSearchRESTService.searchForPlaces(search.toLowerCase().replace( target: " ",  replacement:
    @Override
```

2 - Git - niveau 4

- Alexis a fait une pull request pour toujours chercher en lowercase
- J'ai un commit qui retire les espaces de la recherche
- Je fetch, je rebase -> conflit
- Je corrige le conflit en réécrivant ma modification à partir de son code
- Je git add les fichiers corrigés

```
git add app/src/main/java/org/miage/placesearcher/PlaceSearchService.java
```

```
[alexmorel@~/Documents/Git/org.miage.placesearcher (correction +$|REBASE 1/1) $ git status
rebase in progress; onto 69361d4
You are currently rebasing branch 'correction' on '69361d4'.
  (all conflicts fixed: run "git rebase --continue")

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

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

alexmorel@~/Documents/Git/org.miage.placesearcher (correction +$|REBASE 1/1) $
```

2 - Git - niveau 4

- Alexis a fait une pull request pour toujours chercher en lowercase
- J'ai un commit qui retire les espaces de la recherche
- Je fetch, je rebase -> conflit
- Je corrige le conflit en réécrivant ma modification à partir de son code
- Je git add les fichiers corrigés
- Je dit à git de continuer le rebase : git rebase —continue
- Mon conflit est réglé, je peux pusher puis faire une pull request

```
[alexmorel@~/Documents/Git/org.miage.placesearcher (correction +$|REBASE 1/1) $ git rebase --continue
Applying: Make search without space
```

```
* e5c9310 (HEAD -> correction) Make search without space
* 69361d4 (perso/correction) Make search in lowercase
```

2 - Git - niveau 4

- Je met à votre disposition mes git alias (sur le wiki du repo Team)
- A copier/coller dans le .gitconfig de votre utilisateur

```
[alexmorel@~/Documents/Git/org.miage.placesearcher (correction *+$) $ git st
 M app/build.gradle
 M app/src/main/AndroidManifest.xml
M  app/src/main/java/org/miage/placesearcher/MainActivity.java
M  app/src/main/java/org/miage/placesearcher/MapActivity.java
M  app/src/main/java/org/miage/placesearcher/PlaceSearchService.java
M  app/src/main/java/org/miage/placesearcher/model/PlaceAddress.java
M  app/src/main/java/org/miage/placesearcher/model/PlaceCoordinates.java
M  app/src/main/java/org/miage/placesearcher/model/PlaceProperties.java
M  app/src/main/java/org/miage/placesearcher/ui/PlaceAdapter.java
 M build.gradle
 M gradle.properties
```

2 - Git - niveau 4

- Je met à votre disposition mes git alias (sur le wiki du repo Team)
- A copier/coller dans le .gitconfig de votre utilisateur

```
[alexmorel@~/Documents/Git/org.miage.placesearcher (correction) $ git lol
* b1ad848 (HEAD -> correction, team/correction, perso/correction) [Android] Exercice 17 - Step 3 : PlaceSearchService
* ee12bbc [Android] Exercice 17 - Step 2 : add ORM annotations
* f1a1040 [Android] Exercice 17 - Step 1: add ActiveAndroid to build and configure DB
* 2324ad4 [Android] Exercice 16 - Step 6 : performance improvement : the right way
* 4ba54fa [Android] Exercice 16 - Step 5 : performance improvement : naive way
* 9a45293 [Android] Exercice 16 - Step 4 : automatic camera position
* 749f9b0 [Android] Exercice 16 - Step 3 : add info window click listener
* 7be7fce [Android] Exercice 16 - Step 2 : add markers to map
* 4cabd96 [Android] Exercice 16 - Step 1: add latitude & longitude to Retrofit Model
* 34cf0be [Android] Exercice 15 - Step 3 (optionnel) : Map configuration
* 9b56f16 [Android] Exercice 15 - Step 2 : pass search from List to Map Activities
* f934122 [Android] Exercice 15 - Step 1 : basic Google Maps Wiring
* 580b421 (team/master, perso/master, master) [Android] Exercice 14 : refactor PlaceService using Retrofit
* 1aa4c8c [Android] Exercice 14 : add textfield allowing to enter search
* 805a89a [Android] Exercice 13 : refactor PlaceSearchService by extracting dedicated AsyncTask
* 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
```

LES 3 CHOSES À GARDER EN TÊTE

1. Je fais mes modifs sur mon repo local
2. Je pushe sur MON fork (et j'ai le droit de -f)
3. Je fetch/rebase team et je règle les conflits

git rebase --continue

3 - Bases de données mobile

3 - Bases de données mobile

Vrai partout mais encore plus sur mobile : vous n'aurez **jamais** de conditions idéales

- Vieux modèles de téléphones
- Réseaux lents (H+)
- Mode offline (Jura FTW)

3 - Bases de données mobile

Sur Android : chaque application dispose d'une base de données SQLite

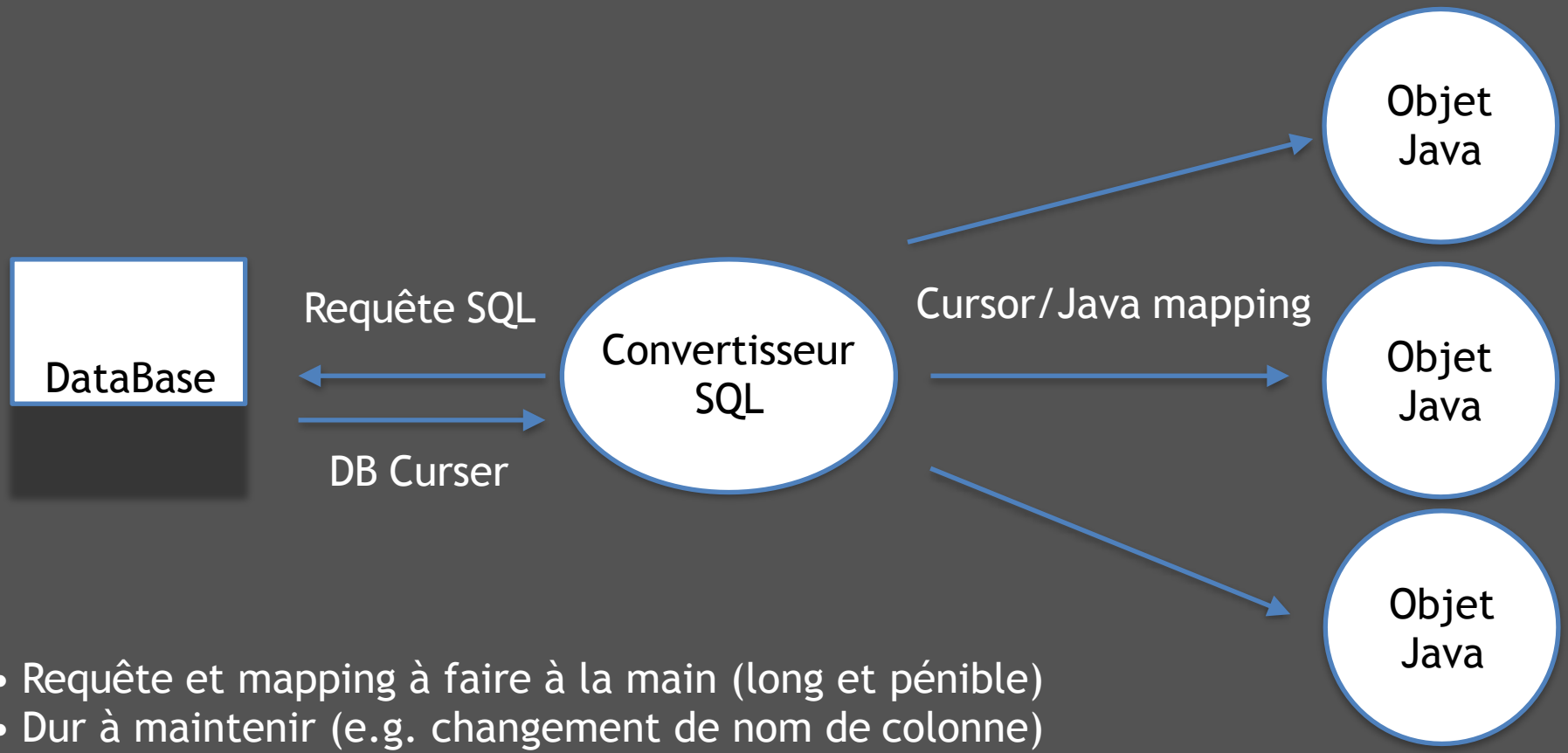
2 grandes approches pour gérer la DB :

- Manipuler directement la base de données (requêtes et tables à la mano)
  - Avantages : si on est bon en SQL on peut avoir de bonnes performances
  - Désavantages : sensible aux changements, beaucoup de chose à gérer manuellement

- Utilisation d'ORM (Object Relational Mapping)
  - Avantages : facile à écrire et à maintenir
  - Désavantages : peu poser des problèmes de performances en cas de grosses jointures

3 - Bases de données mobile

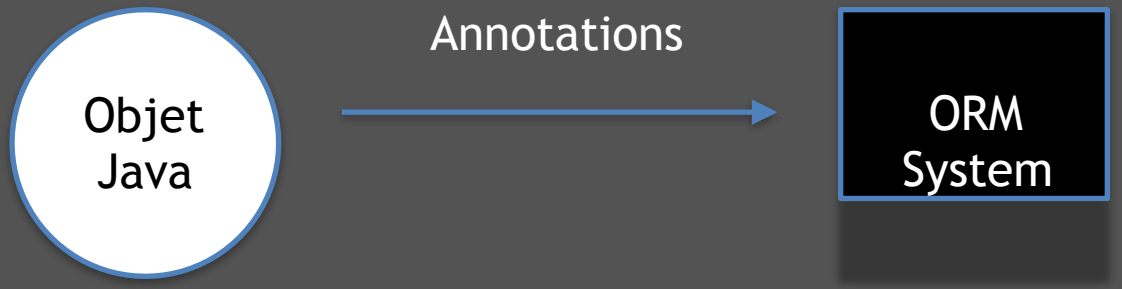
- Utilisation d'ORM (Object Relational Mapping)



- Requête et mapping à faire à la main (long et pénible)
- Dur à maintenir (e.g. changement de nom de colonne)

3 - Bases de données mobile

- Utilisation d'ORM (Object Relational Mapping)



- Tables créés à partir des annotations Java
- Requêtes et Mapping fait automatiquement
- Si je change un nom de champ Java ça modifie la base de données
- Scripts de migration automatique

3 - Bases de données mobile

- Utilisation d'ORM (Object Relational Mapping)

ActiveAndroid : un des ORM disponibles sur Android (pas le mieux, mais le plus simple)

- Description du schéma SQL via des annotations (oui, j'aime les annotations)
- Requêtes ultra-facilitées et résultat directement en objet Java
- Insertion/Update en une méthode .save(), tout les conflits sont gérés automatiquement
- Fonctionnalités avancées si besoin (transactions ACID, migration automatique…)

```
@Table(name = "Items")
public class Item extends Model {
        @Column(name = "Name")
        public String name;

        @Column(name = "Category")
        public Category category;
}
```

3 - Bases de données mobile

Etape 1 : configurer le build

- Ajouter une nouvelle source au build.gradle top-level

```
allprojects {
    repositories {
        google()
        jcenter()
        mavenCentral()
        maven { url "https://oss.sonatype.org/content/repositories/snapshots/" }
    }
```

- Ajouter la dépendance dans le build.gradle de l'application

```
// Add Active Android
implementation 'com.michaelpardo:activeandroid:3+'
```

- Préciser nom et version de la base de données (utile pour de futures migrations)

```
<!-- Active Android Database definition -->
<!-- DB Name -->
<meta-data android:name="AA_DB_NAME" android:value="PlaceSearcher" />
<!-- DB Version -->
<meta-data android:name="AA_DB_VERSION" android:value="1" />
```

3 - Bases de données mobile

Etape 1 : configurer le build

- Votre application doit maintenant hériter de com.activeandroid.app.Application

```
<application
    android:name="com.activeandroid.app.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">
```

- Au démarrage et si la base de données n'est pas initialisée
- Va chercher toutes les classes ORM (possible de donner la liste en dur)
- Créer les tables et index correspondants

3 - Bases de données mobile

Etape 2 : définir le modèle ORM

Via les annotations @Table et @Column

```
@Table(name = "PlaceProperties")
public class PlaceProperties extends Model {

    @Expose
    @Column(name = "name")
    public String name;

    @Expose
    @Column(name = "postcode")
    public String postcode;
```

- @Table : nom de la table
- @Column : nom de la colonne
- Penser à étendre « Model »
- Possible d'utiliser une classe existante…

```
<meta-data
    android:name="AA_MODELS"
    android:value="org.miage.placesearcher.model.PlaceAddress,
                   org.miage.placesearcher.model.PlaceCoordinates,
                   org.miage.placesearcher.model.PlaceProperties" />
```

Chaque @Table doit être listé dans le AA_MODELS du Manifeste

3 - Bases de données mobile

Etape 2 : définir le modèle ORM

@Column :

- détermine automatiquement le type de champ à créer
- possibilité d'ajouter de la logique (INSERT/UPDATE, index…)

```
@Table(name = "PlaceCoordinates")
public class PlaceCoordinates extends Model {

    @Column(name = "label", index = true, unique = true, onUniqueConflict = Column.ConflictAction.REPLACE)
    public String label;
```

3 - Bases de données mobile

Etape 3 : sauver des objets

Appeler save() : s'occuper de faire un INSERT ou UPDATE en fonction de l'id

Attendez…
Si Retrofit permet de parser automatique une réponse JSON en objet Java…
… et qu'ActiveAndroid permet de sauver une classe Java en DB via .save()…
… Alors…

```
mPlaceSearchRESTService.searchForPlaces(search).enqueue(new Callback<PlaceSearchResult>() {
    @Override
    public void onResponse(Call<PlaceSearchResult> call, Response<PlaceSearchResult> response) {
        // Post an event so that listening activities can update their UI
        if (response.body() != null && response.body().features != null) {
            // Save all results in Database
            for (PlaceAddress place : response.body().features) {
                place.save();
            }
```

3 - Bases de données mobile

Etape 4 : interroger la base de données

Méthodes toute prêtes, conversion Java automatique

```
// Get places matching the search from DB
List<PlaceAddress> matchingPlacesFromDB = new Select().
        from(PlaceAddress.class)
        .where("label LIKE '%" + search + "%'")
        .execute();
```

```
properties = new Select().from(PlaceProperties.class).where("label='" + label + "'").executeSingle();
```

3 - Bases de données mobile

Etape 4 : interroger la base de données

Toutes les features de SQL restent possibles

```
// Get places matching the search from DB
List<PlaceAddress> matchingPlacesFromDB = new Select().
        from(PlaceAddress.class)
        .where("label LIKE '%" + search + "%'")
        .orderBy("label")
        .limit(50)
        .off
        m offset(int offset)                    From
```

3 - Bases de données mobile

Exercice 17

- Mettre en place les Annotations ORM nécessaires

- Modifier le Place SearchService pour :
  - sauver les PlaceAdress, PlaceProperties & PlaceCoordinates obtenues via REST
  - attention aux champs sans @Expose (label, latitude, longitude)

- Créer une méthode dans le PlaceSearchService pour :
  - Envoyer un event à partir de la base de données et non du résultat serveur
  - Attention ! Retourner seulement les places dont le label matche la recherche

- Modifier le PlaceSearchService pour :
  - D'abord chercher les places dans la BD local
  - Puis lancer une requête et retourner le résultat

- Une fois quelques recherches effectuées, lancer l'application en mode Avion

3 - Bases de données mobile

Conclusions & retours d'expérience

- Comme pour n'importe quelle DB, vos performances sont liées à la config (indexes…)
- ActiveAndroid & Retrofit sont faits l'un pour l'autre…
- … si le modèle serveur est bien pensé !

3 - Bases de données mobile

Conclusions & retours d'expérience

- … si le modèle serveur est bien pensé !

  - 90% des développeurs négligent de soigner le modèle

  - D'après mon expérience:
    - Un modèle mal pensé et votre code sera 10 fois plus complexe
    - Un modèle mal pensé et votre code ne passera jamais à l'échelle
    - Des routes serveurs mal pensées… same story
    - Passez du temps sur le modèle et les routes, et 50% du job est fait
    - Passez du temps sur les composants et l'architecture, et 80% du job est fait
    - Vous êtes avant tout des architectes logiciels, pas seulement des maçons
    - Et un architecte, ça dessine des plans sur papier avant de construire

3 - Bases de données mobile

Conclusions & retours d'expérience

- … si le modèle serveur est bien pensé !

  - Aucun d'entre vous ne va vraiment retenir cette mise en garde
  - Essayez d'analyser comment vous travaillez : si vous vous jetez sur le clavier, c'est mal
  - Je décris ici le défaut de 100% des jeunes développeurs que je rencontre…
  - …Et un bon 50% des développeurs expérimentés (moi aussi parfois)

4 - Notifications avec Firebase Cloud Messaging

4 - Notifications avec Firebase Cloud Messaging

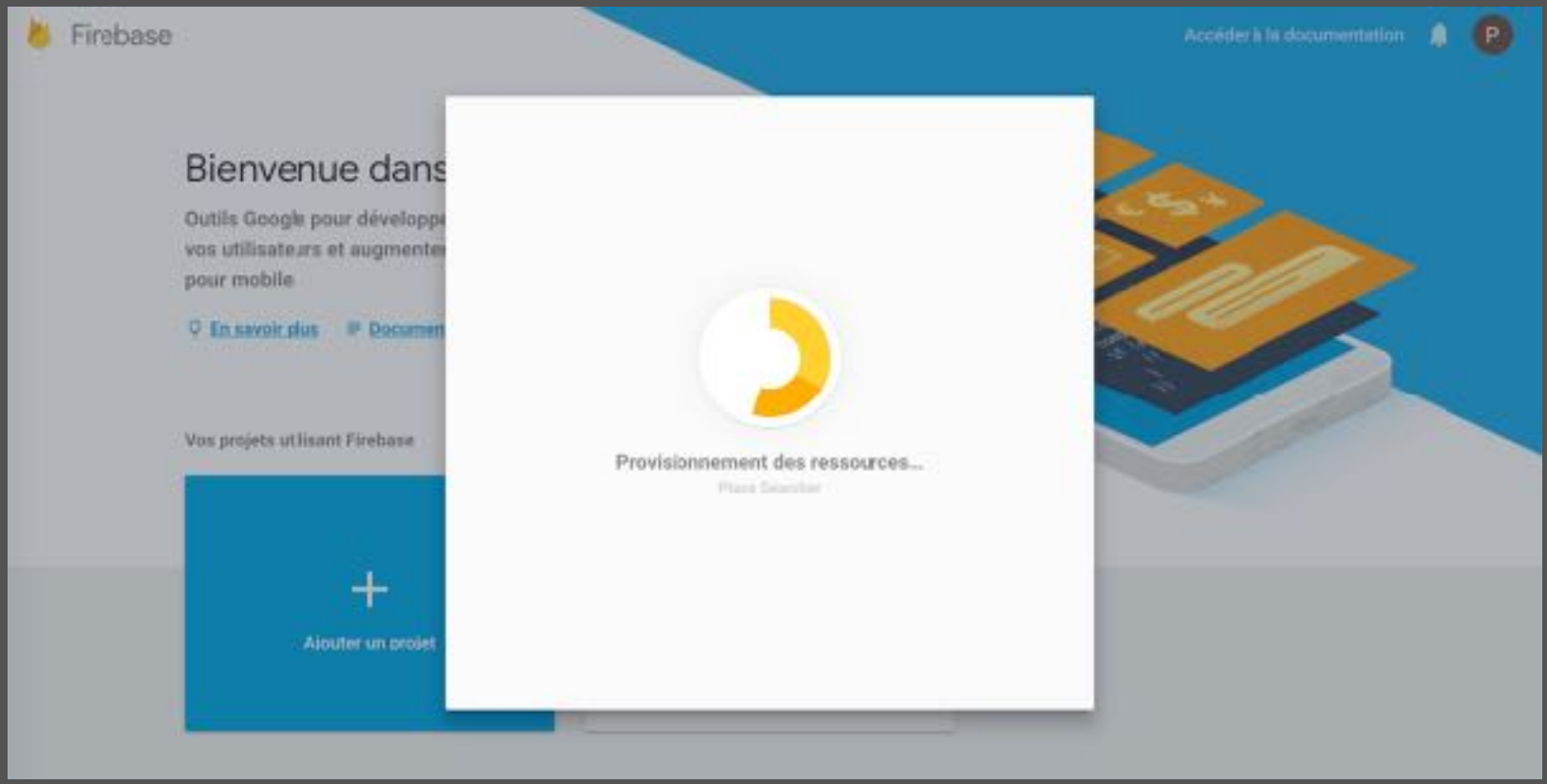
Firebase est un service Google regroupant de nombreux services (login, analytics…)
Pour les notifications, on utilise FCM (Firebase Cloud Messaging).

L'envoi de notifications est assuré par le framework :

- L'application mobile récupère un token de notification (unique pour le téléphone+l'app)
- Elle envoie le token au serveur de l'application qui l'associe à un utilisateur
- Le serveur de l'application utilise un web service pour envoyer une notification
- L'application reçoit la notification, elle peut agir dessus ou la montrer telle quelle

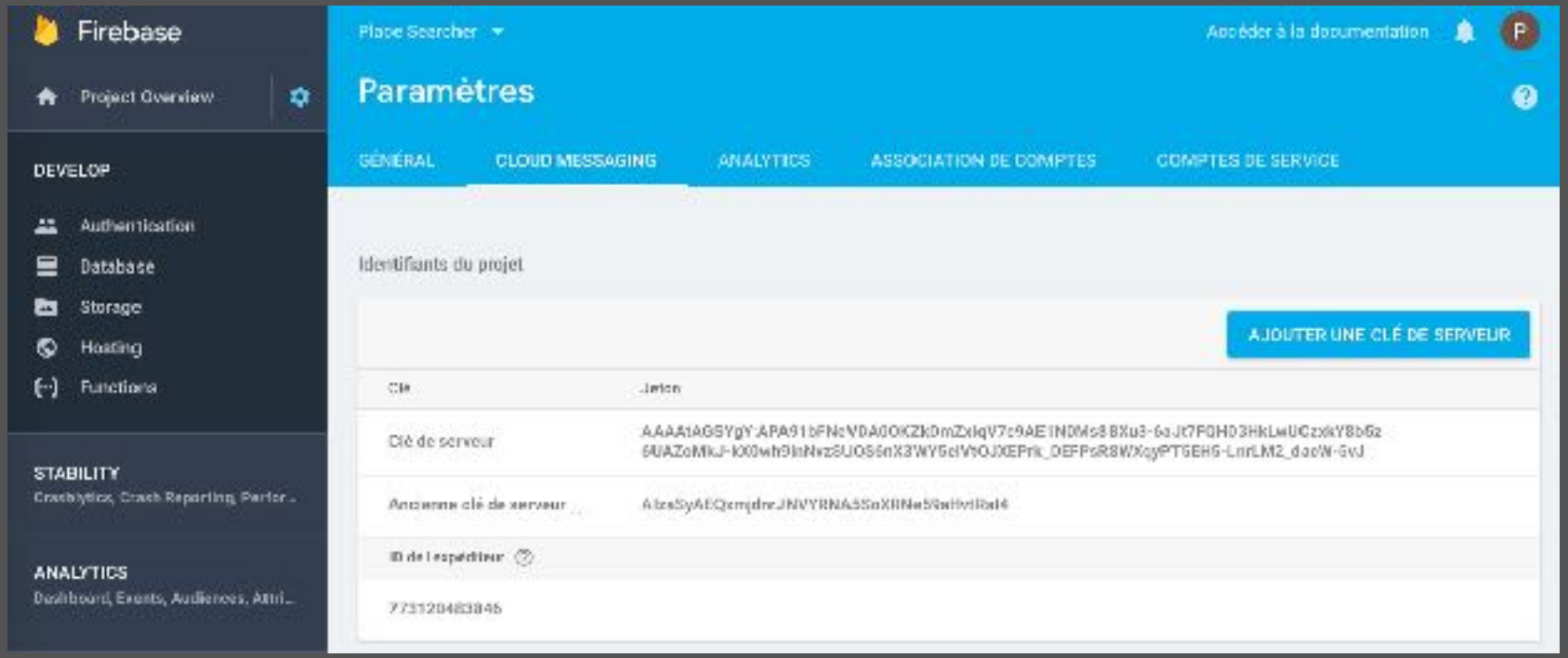
4 - Notifications avec Firebase Cloud Messaging

Etape 1 : créer une application firebase et la configurer



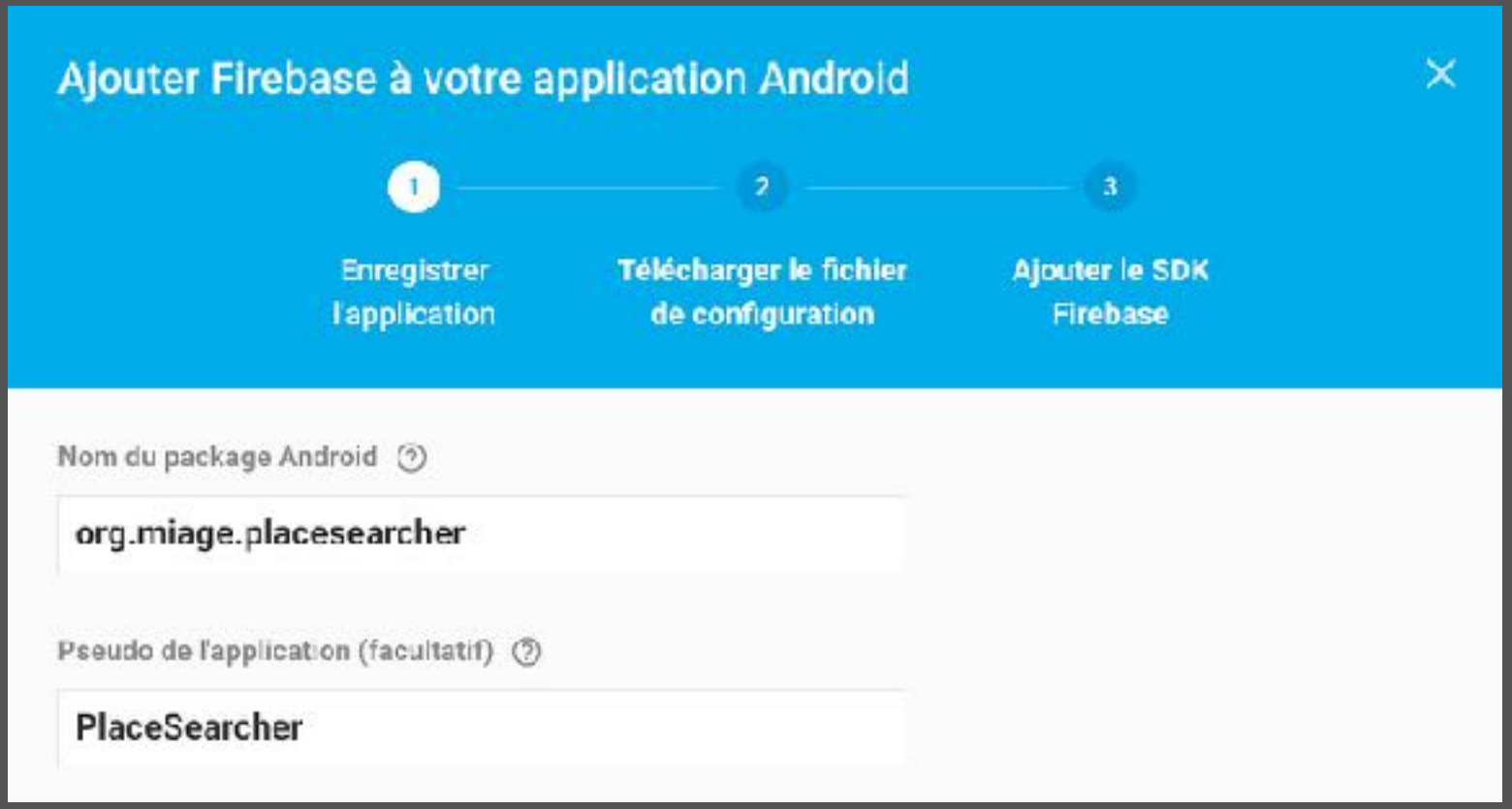
4 - Notifications avec Firebase Cloud Messaging

Etape 1 : créer une application firebase et la configurer



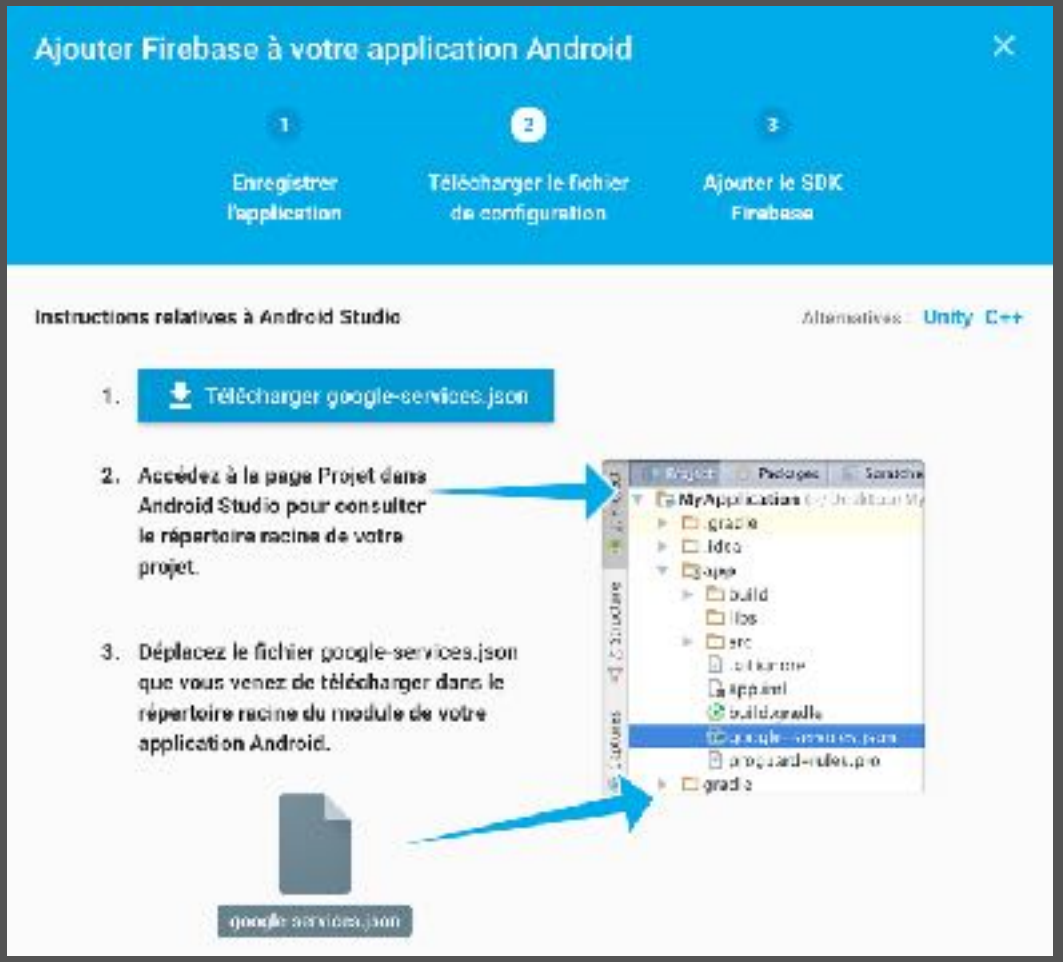
4 - Notifications avec Firebase Cloud Messaging

Etape 1 : créer une application firebase et la configurer



4 - Notifications avec Firebase Cloud Messaging

Etape 2 : modifier le build de l'application



4 - Notifications avec Firebase Cloud Messaging

Etape 2 : modifier le build de l'application

4 - Notifications avec Firebase Cloud Messaging

Etape 3 : récupérer le token firebase

```
<service
    android:name=".service.PlaceSearcherFirebaseNotificationService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>
```

4 - Notifications avec Firebase Cloud Messaging

Etape 3 : récupérer le token firebase

```
public class PlaceSearcherFirebaseNotificationService extends FirebaseMessagingService {

    private static final String NOTIF_CHANNEL_ID = "PlaceSearcher Notifications";


    @Override
    public void onNewToken(@NonNull String s) {
        super.onNewToken(s);
        // Step 1: Get  token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();
        Log.d( tag: "[FireBase Token]",  msg: "Refreshed token: " + refreshedToken);

        // Step 2: send token to server
        // Here we won't do that but in a real project we should
    }

    @Override
```

cB4RSW9VoTU:APA91bFcgXKYjuj7HIBqYae5dGZ_G2tMbVS7cQZiBpLtcEBnK7i0fxU
6ZKsUnGXuragZyC7HKMRRsMd6MzOWhEvTBcjcSBNw5Ebv45HL-
cwT0EkU8_aUlnbXnTRpwsdbyA5jvPDHjNpf

4 - Notifications avec Firebase Cloud Messaging

Etape 4 : réagir aux notifications reçues

```
// Step 1 : Parse notification
String body = message.getNotification().getBody();
```

4 - Notifications avec Firebase Cloud Messaging

Etape 4 : réagir aux notifications reçues

```
// Step 2 : Build push notification
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder( context: this, NOTIF_CHANNEL_ID);
mBuilder.setStyle(new NotificationCompat.BigTextStyle(mBuilder)
        .bigText(body)
        .setBigContentTitle("Place Searcher Notification"))
        .setContentTitle("Place Searcher Notification")
        .setContentText(body)
        .setSmallIcon(R.drawable.home_icon)
        .setLargeIcon(BitmapFactory.decodeResource(getResources(),
                R.drawable.street_icon))
        .setDefaults(android.app.Notification.DEFAULT_VIBRATE);
```

4 - Notifications avec Firebase Cloud Messaging

Etape 4 : réagir aux notifications reçues

```
// Step 3 : Define clic behavior
PendingIntent contentIntent = PendingIntent.getActivity( context: this,  requestCode: 26,
        new Intent( packageContext: this, MainActivity.class),  flags: 0);
mBuilder.setContentIntent(contentIntent);
```

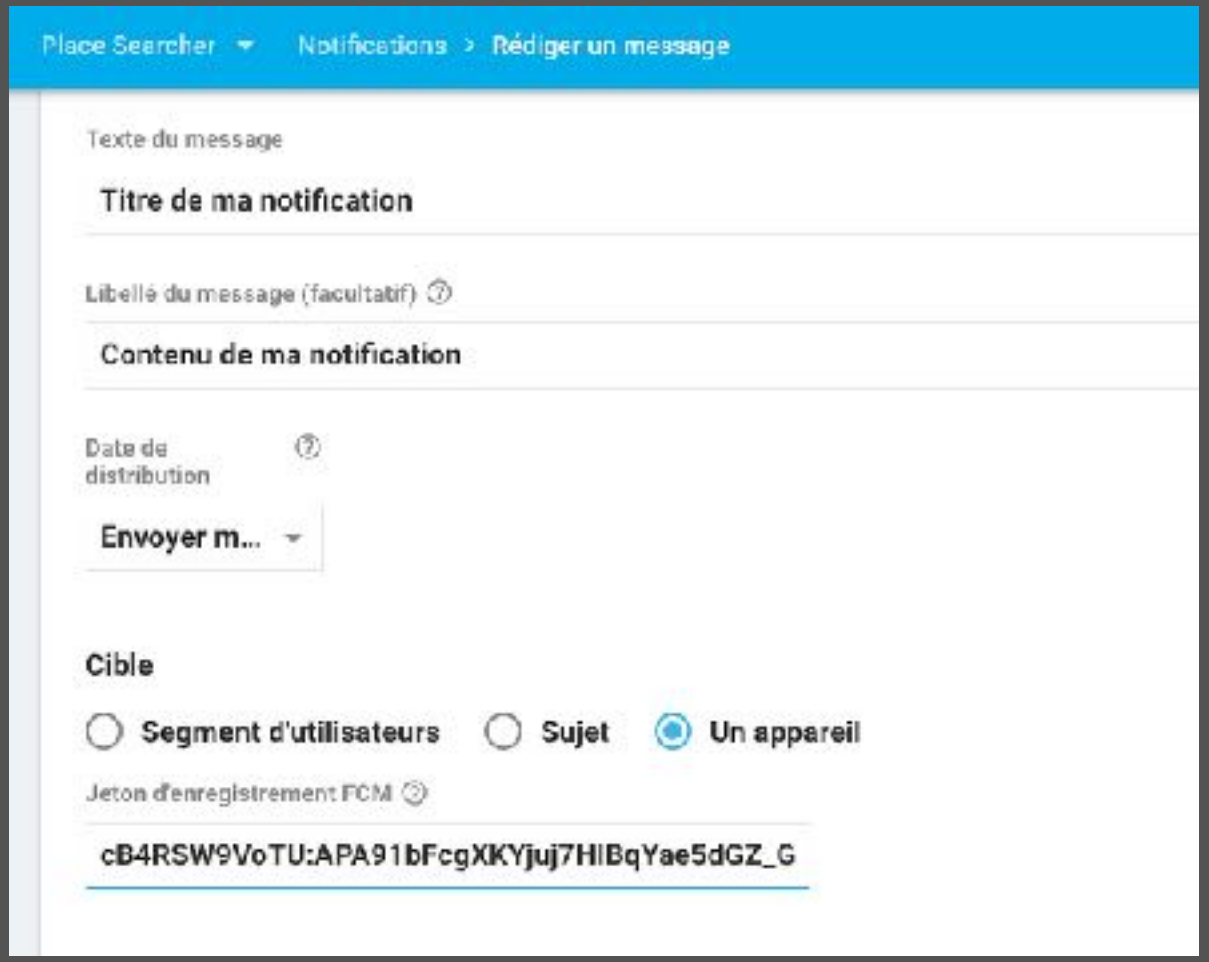
4 - Notifications avec Firebase Cloud Messaging

Etape 4 : réagir aux notifications reçues

```
// Step 4 : send notification
NotificationManager mNotificationManager = ((NotificationManager)getSystemService(NOTIFICATION_SERVICE));
mNotificationManager.notify( id: 26, mBuilder.build());
```

4 - Notifications avec Firebase Cloud Messaging

Etape 5 : tester l'envoi de notifications



4 - Notifications avec Firebase Cloud Messaging

Exercice 18 :

- Installer firebase sur votre projet
- Récupérer et logger votre token Firebase
- Préparer votre service de réception de notification
- Vous connecter sur https://console.firebase.google.com/project/place-searcher-78b5b/notification/compose
  - Login : placesearchermiage@gmail.com
  - Password : miageAreTheBest
- Envoyer une notification à partir de votre registration ID

4 - Notifications avec Firebase Cloud Messaging

- Framework très riche
  - possibilité d'envoyer des données attachées
  - apparence des notifications entièrement personnalisable
  - très facile à intégrer côté serveur (Node.js notamment)
- Firebase embarque gratuitement Google Analytics
- A vous de vous fouiller, les tutoriaux sont très bien faits

THAT'S ALL FOLKS !

NEXT TIME

- TP

D'ICI LÀ

- Avancez sur le TP tant que je suis disponible
- Harcelez-moi (alex.morel@irealite.com)

iREALiTE