UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Sauvegarder sur un cloud

Android : projets individuels

Université Paris 8 - M2 HANDI - Charpentier Samantha

I. Le cloud : définition

Définition : “ *Un modèle qui permet un accès omniprésent à un réseau partagé et à un ensemble de ressources informatiques configurables [...] ”*

Source : culture-informatique.net

Source : pixabay

II. Le cloud : exemples

Dropbox, Google Photos (Ou Drive), Microsoft OneDrive, Mega…

Textes, photos, vidéos, musiques, agenda, contacts… Pas de données sensibles

III. Le cloud : Sync adapter

Android's sync adapter framework Automatise les transferts de données. Plusieurs fonctionnalités disponibles :

- Plug in basé architecture (Implémenter des produits basés applications ; packagées et téléchargeables) Autorise le transfert de données au système sous la forme de composants appelables, exemple : private static SyncAdapter sSyncAdapter = null ;

```
synchronized (sSyncAdapterLock){...}
```

- Détecte la connexion internet automatiquement : onPerformSync() quand une synchronisation a lieu

Doc : https://developer.android.com/training/sync-adapters/

III. Le cloud : Sync adapter

Account-authenticator : “ The **sync adapter** framework assumes that your **sync adapter** transfers data between device storage associated with an **account** and server storage that requires **login** access. For this reason, the framework expects you to provide a component called an **authenticator** as part of your **sync adapter**. [...] Even if your app doesn't use accounts, you still need to provide an authenticator component. “

Auth-token pour contacter le serveur et celui-ci est fourni par l’authenticator.
AbstractThreadedSyncAdapter ---> AccountManager ?

III. Le cloud : Sync adapter

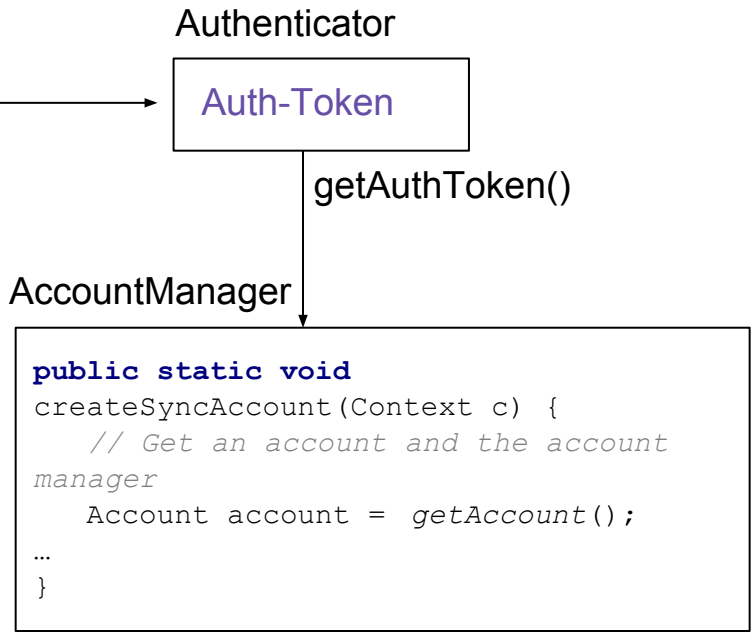
AuthenticatorService

```
public void onCreate() {
    // Create a new authenticator object
    mAuthenticator = new Authenticator(this);
}
```

Est appelé par SyncAdapter : Transfert les données entre le client et le serveur

ContentResolver accès au ContentProvider

```
public SyncAdapter(Context context, boolean
autoInitialize, boolean allowParallelSyncs) {
        super(context, autoInitialize,
allowParallelSyncs);    mContentResolver =
context.getContentResolver();
}
```



III. Le cloud : Sync adapter

SyncAdapter (classe) : Encapsule les données pour les envoyer

Sync adapter XML metadata file : Le framework lit les informations pour trouver comment charger et prévoir le transfert de données.

SyncAdapter.xml : … <sync-adapter

```
android:contentAuthority="com.example.android.provider"
```

// pckg + nomClasse

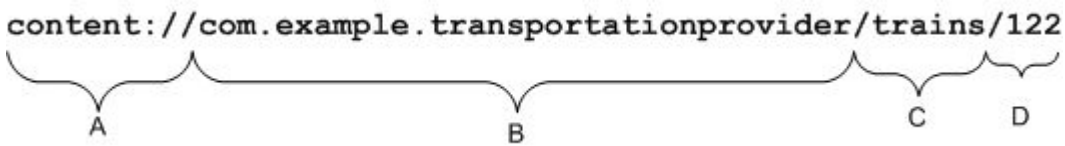
```
android:accountType="com.android.example.datasync"
```

/>

III. Le cloud : Sync adapter : ContentProvider

URI : Uniform Resource Identifier : “ is a compact sequence of characters that identifies an abstract or physical resource. “

content://MyAuthority/pathToData/data



source : https://mathias-seguy.developpez.com/tutoriels/android/content-provider/

A : ContentProvider, B : Package du Provider (authority), C : Type de données, D : ID spécifique

III. Le cloud : Sync adapter : ContentProvider

```
    public static final String AUTHORITY = "myAuthority"
    public static final String PATH_TO_DATA = "myPath"; // Peut être vide

    public class Constants extends BaseColumns {
        public static final URI CONTENT_URI = URI.parse("content://" +
MyDataProvider.AUTHORITY + "/" + MyDataProvider.PATH_TO_DATA);
        …
        }
    }
```

III. Le cloud : Sync adapter / ContentProvider

```
public boolean onCreate() { Instanciation de la base de données (myDatabaseHelper) }

public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sort) { Requête SELECT sur la table }

public Uri insert(Uri _uri, ContentValues _initialValues) { Insère la nouvelle ligne }

public int delete(Uri uri, String where, String[] whereArgs) { Si c'est une collection on
supprime tout, si c'est un objet particulier on le supprime  // switch(URIMATCHER)  }

public int update(Uri uri, ContentValues values, String where, String[] whereArgs) { Collection
ou item // switch(URIMATCHER)  }

static { uriMatcher = new UriMatcher(UriMatcher.NO_MATCH); uriMatcher.addURI(AUTHORITY, "users",
COLLECTION); uriMatcher.addURI(AUTHORITY, "users/#", ITEM); }
```

III. Le cloud : Sync adapter / User

Classe User (ou tout autre objet à manipuler)

```
public static final String KEY_COL_ID = "_id";

public static final String KEY_COL_NAME = "name";

public static final String KEY_COL_WEIGHT = "weight";

public static final int ID_COLUMN = 1;

public static final int NAME_COLUMN = 2;

public static final int WEIGHT_COLUMN = 3;

public static final Uri CONTENT_URI = Uri.parse("content://" +
s.test.providers.StubProvider. AUTHORITY + "/users");
```

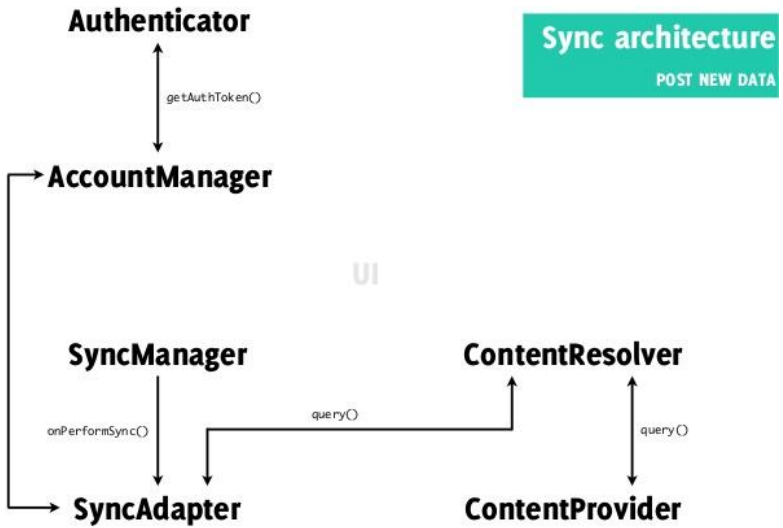
III. Le cloud : Sync adapter / Résumé

ContentProvider construit les requêtes

ContentResolver sélectionne le ContentProvider souhaité

SyncAdapter envoie les données, SyncManager autorise syncAdapter à appeler onPerformSync()

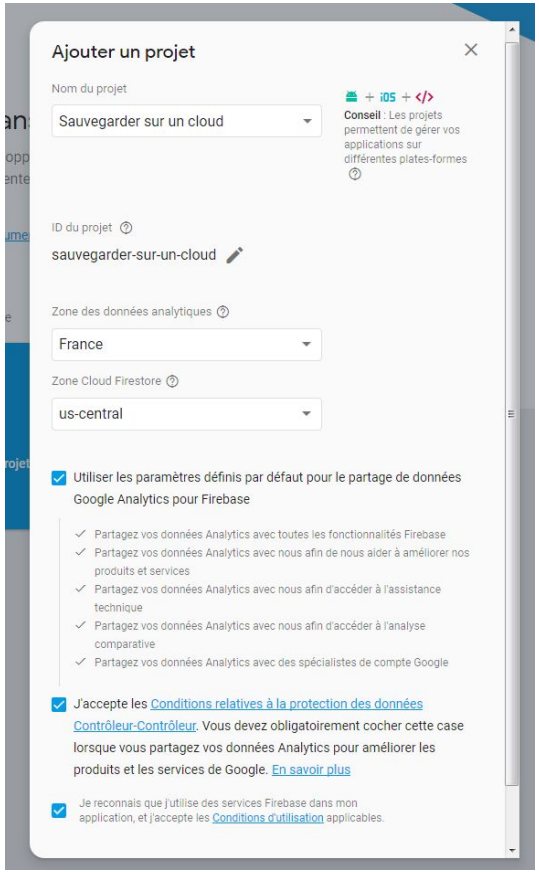
AccountManager : fournit un accès à un registre centralisé des comptes de l'utilisateur.



Source : https://www.slideshare.net/chalup/sync-on-android

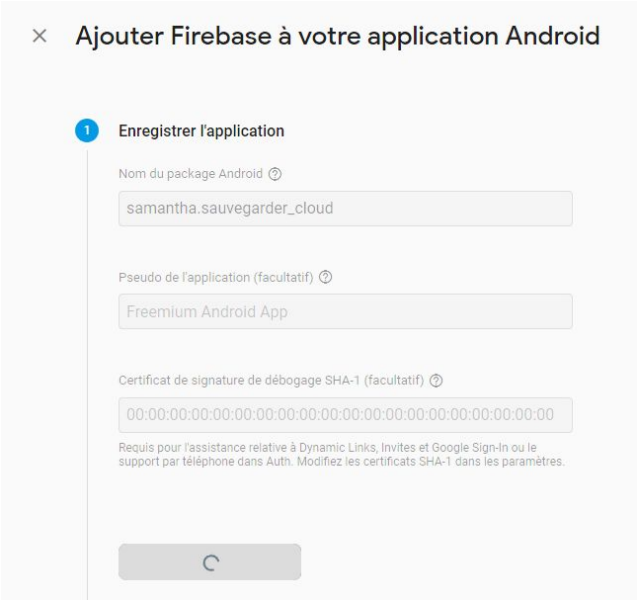
III. Firebase

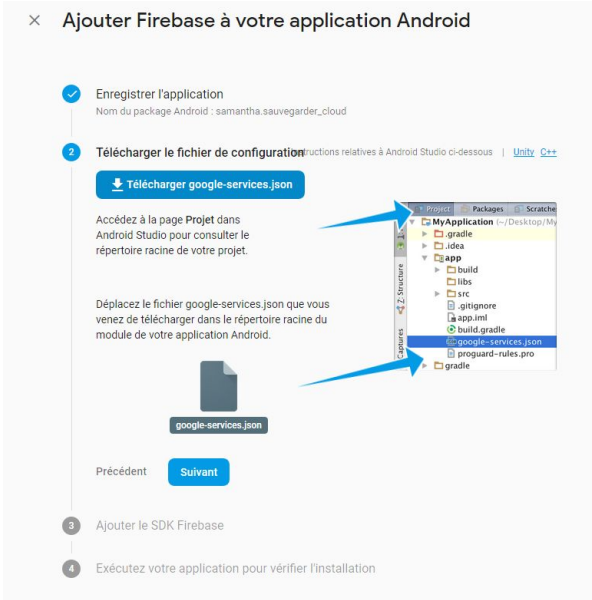
1) Ajouter un projet : https://console.firebase.google.com/?hl=fr&pli=1

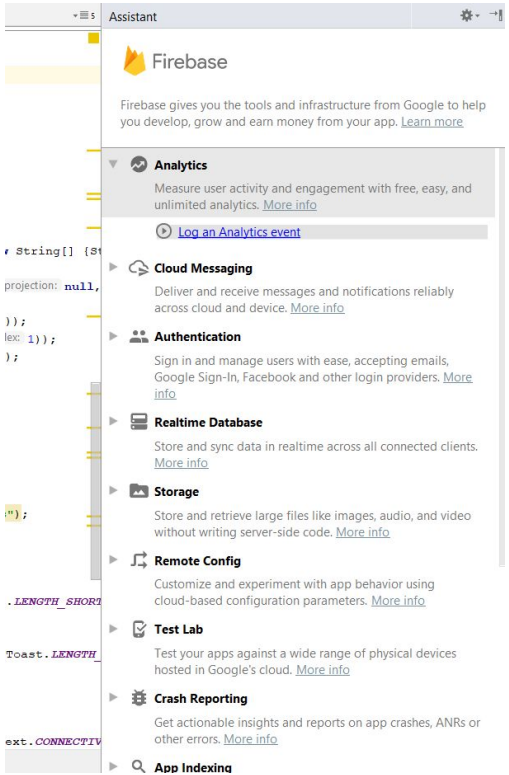


III. Firebase

Ajout manuel

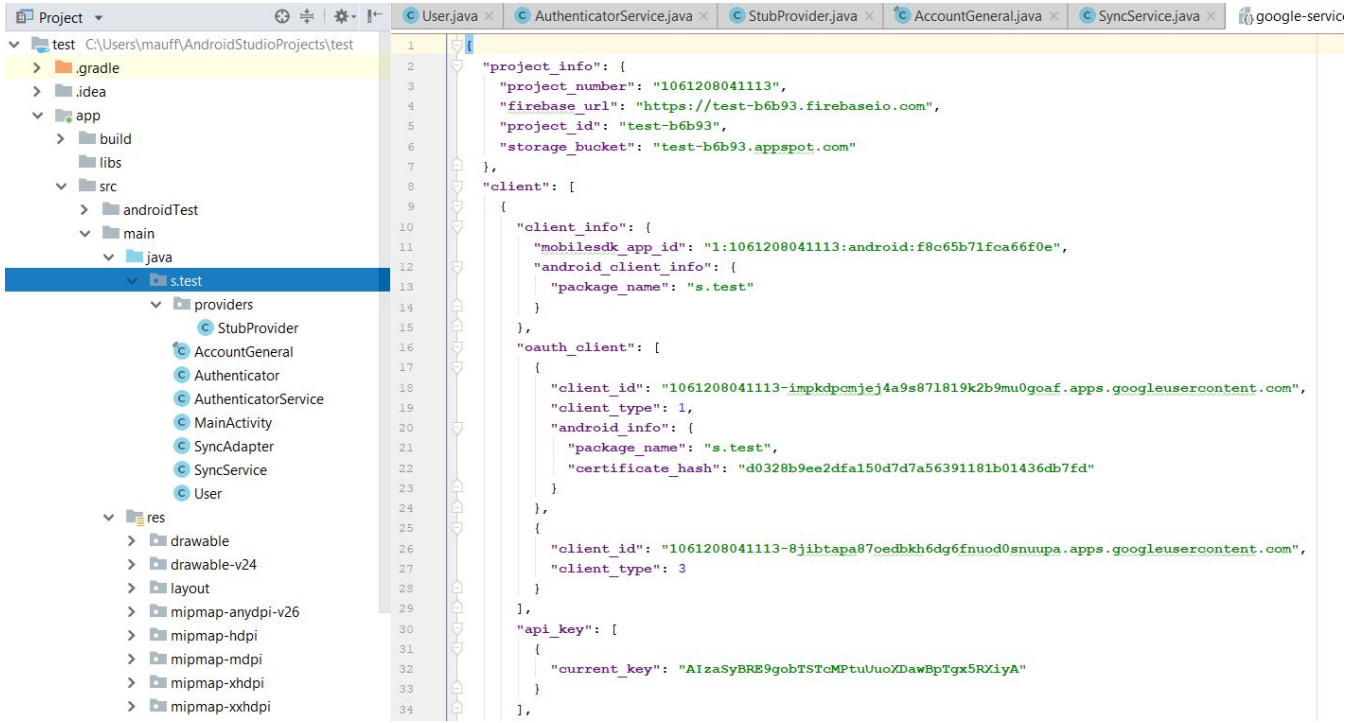






III. Firebase

Fichier Json



III. Firebase

```
if (isInternetConnected(getApplicationContext())){

   FirebaseDatabase database = FirebaseDatabase. getInstance();

   DatabaseReference myRef = database.getReference( "weight");

   myRef.setValue(result);

   DateFormat dateFormat =  new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

   Date date =  new Date();

   String strDate = dateFormat.format(date).toString();

   DatabaseReference champ_date = database.getReference( "date");

   champ_date.setValue(strDate);

}
```

III. Firebase