

Sauvegarder sur un cloud

Android : projets individuels

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



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();  
}
```

Authenticator

Auth-Token

getAuthToken()

AccountManager

```
public static void  
createSyncAccount(Context c) {  
    // Get an account and the account  
    manager  
    Account account = getAccount();  
    ...  
}
```

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" // pkg + nomClasse
```

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

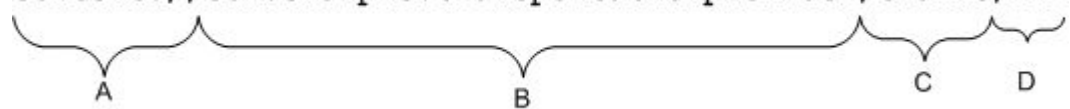
```
/>
```

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`

`content://com.example.transportationprovider/trains/122`



The diagram shows the URI `content://com.example.transportationprovider/trains/122` with four brackets underneath it. Bracket A is under `content://`, bracket B is under `com.example.transportationprovider`, bracket C is under `trains`, and bracket D is under `122`.

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");
```

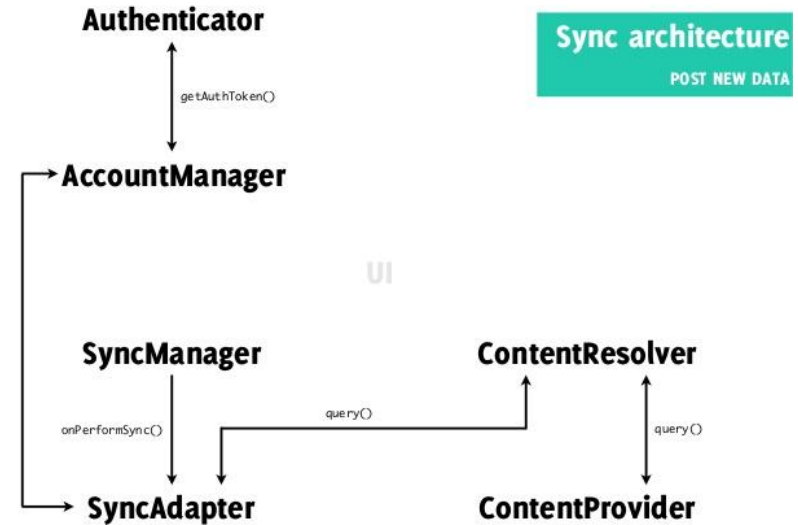
III. Le cloud : Sync adapter / Résumé

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




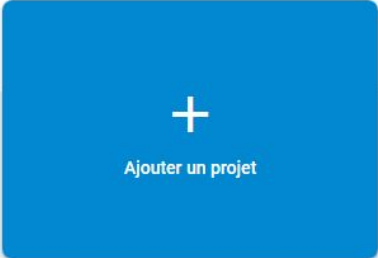
1) Ajouter un projet : <https://console.firebase.google.com/?hl=fr&pli=1>

Bienvenue dans Firebase

Outils Google pour développer des applications, échanger avec vos utilisateurs et augmenter vos revenus avec des annonces pour mobile.

[En savoir plus](#) [Documentation](#) [Assistance](#)

Vos projets utilisant Firebase



Ajouter un projet

Nom du projet + iOS + </>
Conseil : Les projets permettent de gérer vos applications sur différentes plates-formes

Sauvegarder sur un cloud

ID du projet
sauvegarder-sur-un-cloud

Zone des données analytiques
France

Zone Cloud Firestore
us-central

Utiliser les paramètres définis par défaut pour le partage de données Google Analytics pour Firebase

- Partagez vos données Analytics avec toutes les fonctionnalités Firebase
- Partagez vos données Analytics avec nous afin de nous aider à améliorer nos produits et services
- Partagez vos données Analytics avec nous afin d'accéder à l'assistance technique
- Partagez vos données Analytics avec nous afin d'accéder à l'analyse comparative
- Partagez vos données Analytics avec des spécialistes de compte Google

J'accepte les [Conditions relatives à la protection des données Contrôleur-Contrôleur](#). Vous devez obligatoirement cocher cette case lorsque vous partagez vos données Analytics pour améliorer les produits et les services de Google. [En savoir plus](#)

Je reconnais que j'utilise des services Firebase dans mon application, et j'accepte les [Conditions d'utilisation](#) applicables.

III. Firebase



Ajout manuel

Ajouter Firebase à votre application Android

1 Enregistrer l'application

Nom du package Android

Pseudo de l'application (facultatif)

Certificat de signature de débogage SHA-1 (facultatif)

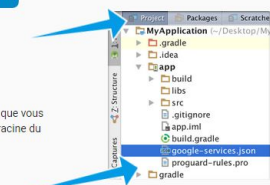
Requis pour l'assistance relative à Dynamic Links, Invites et Google Sign-In ou le support par téléphone dans Auth. Modifiez les certificats SHA-1 dans les paramètres.

Ajouter Firebase à votre application Android

2 Télécharger le fichier de configuration

Accédez à la page **Projet** dans Android Studio pour consulter le répertoire racine de votre projet.

Déplacez le fichier **google-services.json** que vous venez de télécharger dans le répertoire racine du module de votre application Android.



3 Ajouter le SDK Firebase

4 Exécutez votre application pour vérifier l'installation

Assistant

Firebase

Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. [Learn more](#)

Analytics

Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)

Cloud Messaging

Deliver and receive messages and notifications reliably across cloud and device. [More info](#)

Authentication

Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. [More info](#)

Realtime Database

Store and sync data in realtime across all connected clients. [More info](#)

Storage

Store and retrieve large files like images, audio, and video without writing server-side code. [More info](#)

Remote Config

Customize and experiment with app behavior using cloud-based configuration parameters. [More info](#)

Test Lab

Test your apps against a wide range of physical devices hosted in Google's cloud. [More info](#)

Crash Reporting

Get actionable insights and reports on app crashes, ANRs or other errors. [More info](#)

App Indexing

III. Firebase



Fichier Json

```
1
2 "project_info": {
3   "project_number": "1061208041113",
4   "firebase_url": "https://test-b6b93.firebaseio.com",
5   "project_id": "test-b6b93",
6   "storage_bucket": "test-b6b93.appspot.com"
7 },
8 "client": [
9   {
10    "client_info": {
11      "mobilesdk_app_id": "1:1061208041113:android:f8c65b71fca66f0e",
12      "android_client_info": {
13        "package_name": "s.test"
14      }
15    },
16    "oauth_client": [
17      {
18        "client_id": "1061208041113-impkdpcmje4a9s871819k2b9mu0goaf.apps.googleusercontent.com",
19        "client_type": 1,
20        "android_info": {
21          "package_name": "s.test",
22          "certificate_hash": "d0328b9ee2dfa150d7d7a56391181b01436db7fd"
23        }
24      },
25      {
26        "client_id": "1061208041113-8jibtapa87oedbk6dg6fnuod0snuupa.apps.googleusercontent.com",
27        "client_type": 3
28      }
29    ],
30    "api_key": [
31      {
32        "current_key": "AIzaSyBRE9gobTSTcMPtuUuoXDawBpTgx5RXiyA"
33      }
34    ]
35  }
36 ],
```

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

A screenshot of the Firebase console interface. The left sidebar is dark grey and contains the following sections: 'Développer' (with sub-items: Authentication, Database, Storage, Hosting, Functions, ML Kit), 'Qualité' (with sub-items: Crashlytics, Performance, Test L...), 'Données analytiques' (with sub-items: Dashboard, Events, Conversions,...), and 'Enrichir'. The main content area has a blue header with 'test' and a dropdown menu. Below the header, the word 'Database' is displayed in large white text, followed by a 'Realtime Database' dropdown menu. A navigation bar below the header contains the tabs 'Données', 'Règles', 'Sauvegardes', and 'Utilisation'. The main content area shows a URL 'https://test-b6b93.firebaseio.com/' and a JSON object for 'test-b6b93' with the following data: 'date: "2019/01/15 16:16:59"', 'name: "Laura"', and 'weight: "68kg"'.

test

Database

Realtime Database

Données Règles Sauvegardes Utilisation

<https://test-b6b93.firebaseio.com/>

```
test-b6b93
├── date: "2019/01/15 16:16:59"
├── name: "Laura"
└── weight: "68kg"
```