# Melodienet\_android\_client

**ANALYSE ORGANIQUE**

Table des matières

[Melodienet\_android\_client 1](#_Toc482375280)

[1.1 Schéma de fonctionnement général 2](#_Toc482375281)

[1.2 Gestion des langues 3](#_Toc482375282)

[1.3 Fonctionnement par activite 4](#_Toc482375283)

[1.2.1 ActiviteLancement 4](#_Toc482375284)

## Schéma de fonctionnement général

En bleu : ce qui est terminé et fonctionnel

En orange : ce qu’il reste à faire

Get Running Mode()

Get Line()+ Get Station()

Line + station

Get Line()

Get (« Date »)Production()

Line + Production Date

WebService

DataBase

Parameters

Temporaire

Check password

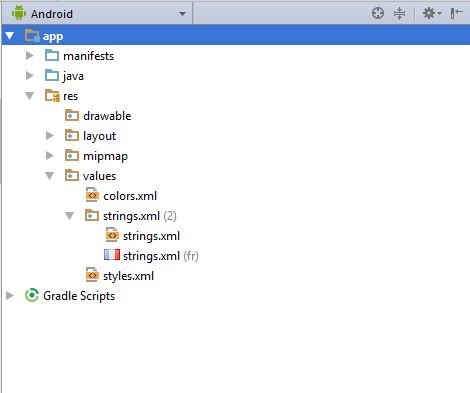
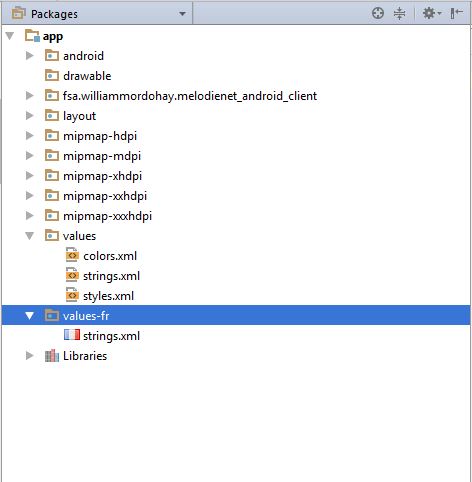
Send the phone langage

Check password

## Gestion des langues

Rajouter une langue :

* Création d’un « Android ressources directory » dans values avec la valeur « values-fr » pour le français. Dans le cas d’une autre langue, il suffit de remplacer le code fr par le code de la langue voulue.
* Dedans, on crée un fichier « strings.xml » qui contiendra toutes les chaines de caractères utilisées par l’application pour la langue en question



L’application est entièrement fonctionnelle pour l’anglais et le français. L’anglais est la langue définie par défaut, ce qui signifie que tout téléphone qui est dans une autre langue que le français démarrera automatiquement la version anglaise de l’application.

Chaque fois qu’une nouvelle chaine de caractère visible par l’utilisateur est créée, il faut penser à la définir dans les string.xml de chaque langue et utiliser le même identifiant quand on y fait référence de manière à permettre à l’application d’afficher la bonne traduction.

## Objets utilises

### Structure de données des informations fournies par le service Web

#### **ObjetConnexion**(package securite) :

Constructeur ? OUI

**Attributs :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom d’attribut** | **Type** | **Muni d’un setter ?** | **Muni d’un getter ?** |
| nomUtilisateur | String | OUI | OUI |
| accord | Booléen | OUI | OUI |

#### **CryptageMdp**(package securite) :

Constructeur ? NON, méthode globale.

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| *Static* hashMdp | String : mdp | String | Applique un hashage de type « md5 » au mot de passe passé en paramètre (voir algorithme) |

*Algorithme :*

*-On crée un string mdpMd5 initialisé à nul*

*-Si le mot de passe passer en paramètre est nul on retourne un hashage nul*

*-On crée un objet de type « MessageDigest » en précisant que l’algorithme utilisé sera de type MD5.*

*-on charge le mot de passe dans cet objet*

*-On crée un String à partir d’un BigInteger contenant le hashage md5 du mot de passe via l’appel à la méthode « digest() »*

update(mdp)

crée

MessageDigest

CryptageMdp

Digest()

ToString()

#### **ConstructeurUrl**(package connexionserviceweb) :

Constructeur ? OUI

**Attributs :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom d’attribut** | **Type** | **Muni d’un setter ?** | **Muni d’un getter ?** |
| baseUrl | String | NON | NON |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| obtenirLogin | String : utilisateur  String : motDePasse | String | Retourne L’URL permettant d’obtenir le résultat de l’authentification à partir de l’URL de base et des deux chaines passées en paramètre. |
| obtenirListeCelulles | String : numLigne | String | Retourne l’URL permettant d’obtenir la liste des cellules en fonction de l’URL de base et du numéro de ligne passé en paramètre. |
| obtenirListeLignes | Vide | String | Retourne l’URL permettant d’obtenir la liste des lignes en fonction de l’URL de base |
| obtenirModesMarche | String : numLigne  String : numCellules | String | Retourne l’URL permettant d’obtenir les modes de marche en fonction de l’URL de base, du numéro de ligne et du numéro de cellule. |
| obtenirProdHeure | String : numLigne | String | Retourne l’URL permettant d’obtenir la production horaire en fonction de l’URL de base et du numéro de ligne. |
| obtenirProdJour | String : numLigne | String | Retourne l’URL permettant d’obtenir la production journalière en fonction de l’URL de base et du numéro de ligne. |
| obtenirProdSemaine | String : numLigne | String | Retourne l’URL permettant d’obtenir la production hebdomadaire en fonction de l’URL de base et du numéro de ligne. |
| envoyerLang | String lang | String | Retourne l’URL permettant d’envoyer la langue au service web en fonction de l’URL de base et du paramètre. |

## Asynctask

#### **lectureDonnéesWeb**(package connexionserviceweb) :

**doInBackground :**

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom d’attribut** | **Type** | **Valeur initiale** |
| connexion | HttpURLConnection | Nulle |
| lecteur | BufferedReader | Nulle |

**Algorithme :**

-On crée un URL avec le string passé en paramètre.

-On ouvre la connection

-On se connecte

-On crée un InputStream sur la connexion

-On lit sur cet inputStream et on met les informations lues sur un StringBuffer.

Crée un URL à partir du String + openConnection

URL en format String

Connect() + getInputStream

*InputStream*

Crée

Rempli un StringBuffer

Crée

## Description par activite

### 1.2.1 ActiviteLancement

**Fichier xml associé :**  activite\_lancement.xml

**Variables :**

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| ivFsa | ImageView | Contient l’image View de nom « logoView » |
| ivMelodie | ImageView | Contient l’image View de nom « melodieView » |

**Méthodes :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Paramètres** | **Valeurs retournées** | **Description** |
| genereAnim | Animation, ImageView | vide | Applique une animation à une image |
| quitteAnim | Animation, ImageView | vide | Quitte l’activité « ActiviteLancement » dès la fin de l’animation sur l’image |