13/04 :

-Tests de non-régression(onglet Performance -> defaut procédé

-Réunion avec le tuteur pour définir les besoins et objectifs de l’application

-Réalisation de la maquette de l’application

14/04

-Etude des préférences pour rentrer des paramètres dans l’application et les conserver.

<https://openclassrooms.com/courses/creez-des-applications-pour-android/le-stockage-de-donnees-2>

-pb : pour ajouter le xml : « preference.xml », on ne peut plus utiliser addPreferencesFrom Ressource qui est devenu obsolète (depricated). On doit donc passer par l’utilisation d’un fragment.

https://www.youtube.com/watch?v=1RDt0Lic3Bc

<https://www.youtube.com/watch?v=oXlQzs76QIQ>

18/04 :

Le fragment est de type PreferenceFragment (Il en hérite). Le but est de charger l’xml des paramètres dans ce fragment puis ensuite, dans une activité, de charger ce fragment dans un objet de type FragmentTransaction. Cela nous pemettra ensuite d’utiliser cette activité avec l’xml paramètre associé. L’un des avantages d’utiliser un fragment est entre autres de sauvegarder les éventuels modifications dans les paramètres.

Create CellObject class

18/04 Add an AlertDialog when the user quit the application and add the resume what i did

Adapt the text for both langages in URL parameters and set the default value of URL with the MelodieNet's adress

Rename classes and begin the ListView

19/04 : Make the CellAdapter, populate the ListView in MachineActivity.java

Add all the colour corresponding to color code in the cellAdapter.

Beaucoup galéré : pb -> je voulais parcourir la ListView dans l’activité pour mettre à chaque fois le fond TextView du code couleur avec la couleur correspondant au code. C’était impossible, la solution a été de le faire directement dans CellAdapter au moment du remplissage des vues.

* Doing the same with ProductionActivity

Rafraichir : <http://www.softwarepassion.com/android-series-custom-listview-items-and-adapters/>

20/04 :

Fini les tests de non-régression

Ajout du refresh dans la liste en éxecutant un thread en arrière-plan

21/04 : On ajoute un écran de chargement de l’application au lancement permettant de mettre en avant l’entreprise et le produit « MelodieNet »

On met le refresh sur l’activité production aussi

On fait toutes les traductions oubliées

Travail sur le style général de l’application : couleurs des boutons, couleurs de fond

Ajout de la classe Gson à build.gradle.

Création d’une variable statique pour récupérer l’URL dans paramètres et la rendre accessible au service Web

Ecriture de la classe WebService qui enverra des requetes au service Web

Travail sur le style, la couleur du fond.

Login, register :https://www.youtube.com/watch?v=QxffHgiJ64M&list=PLe60o7ed8E-TztoF2K3y4VdDgT6APZ0ka

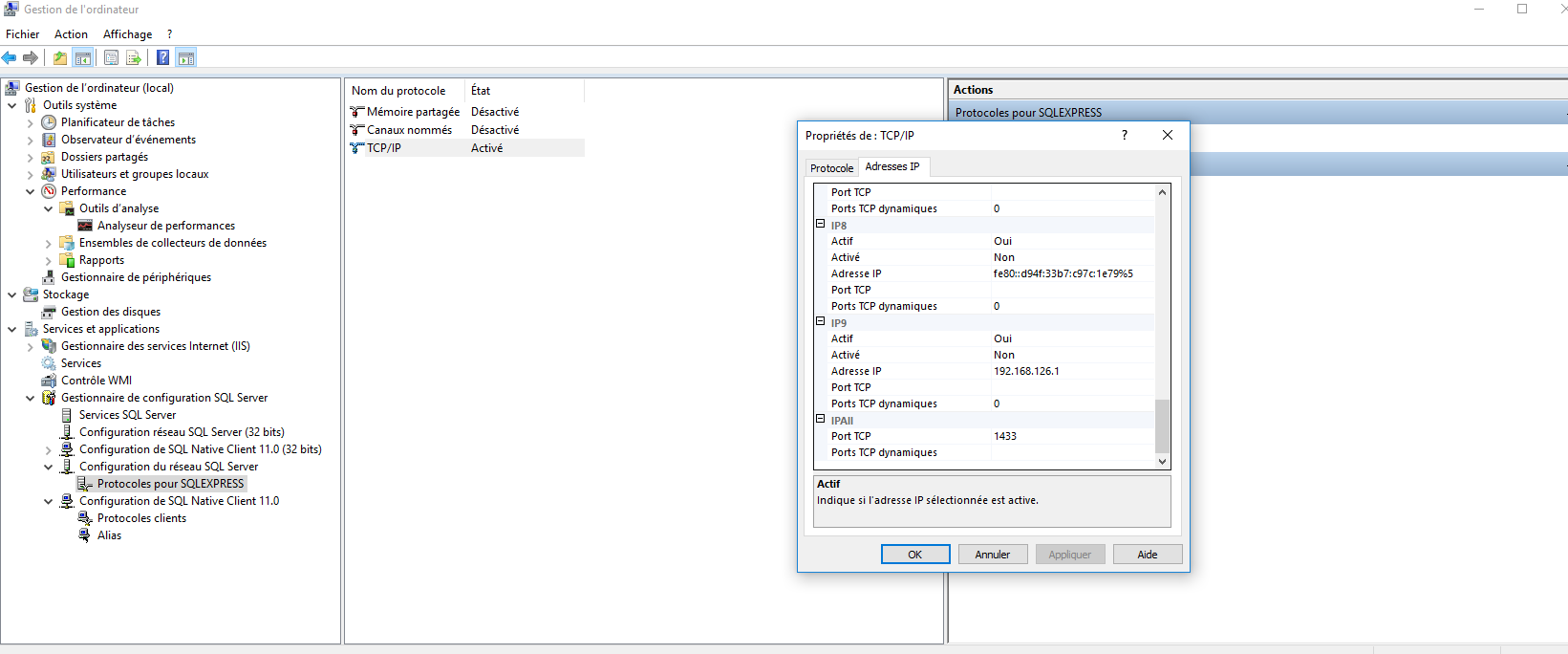
Pour la page de login, réalisation d’une partie

24/04 : Récupération du logo doerfer sans fond.

Pb : remplir l’attribut Hint ne marchait pas en xml pour les edit text de type password. En le faisant en Java via la méthode setHint cela a résolu le problème.

On écrit deux script php login et register et on les upload sur le file manager de 000 webhost après avoir au préalable créer notre bdd.

27/04 : Refactoring the two activity into a single View Activity \*

* Faire un projet login qui utilise un fichier jar de type JDBC pour communiquer avec la base de données concernant les identifiants et mots de passe.
* 
* Il fallait aller dans « Gestion de l’ordinateur » puis aller dans la configuration du réseau « SQL server » puis protocole pour SQLEXPRESS, TCP/IP et mettre à null la valeur de ports TCP dynamique et mettre 1433 dans la valeur port TCP (=statique). En effet cette configuration m’a permis de me connecter à la BDD ce qui plantait avant.

02/05 : travail sur le hashage du mot de passe par le protocole MD5 lors de la connexion. Le problème est que le mot de passe entré par l’utilisateur que je hash via la méthode hashPassword de la classe Hash ne le crypte pas exactement comme il l’est dans la BDD.. On a « c2fca0234de447157701d0c2d64094 » au lieu de « c2fca02304de447157701d00c2d64094 » pour administrateur+. J’ai donc vérifié via <http://md5encryption.com/> lequel était le bon. C’était celui de la base de données. Le soucis provenait de la :

|  |
| --- |
| StringBuffer sb = new StringBuffer(); |
|  |

|  |
| --- |
| for(byte b1 : byteTable){ |
|  |

|  |
| --- |
| sb.append(Integer.toHexString(b1 & 0xff).toString()); |
|  |

}

Il fallait remplacer ça par ça :

*//Converts message digest value in base 16 (hex)*md5 = **new** BigInteger(1, digest.digest()).toString(16);

Le fait de remplacer le StringBuffer par un BigInteger a été déterminant.

* On inclut le module dans le projet principal
* On s’occupe de la déconnexion

Problème : j’utilisais *informUser = R.string.Password\_failure;* pour récupérer une chaine dans les ressources au sein du code java, ce qui déclenchait une erreur car *R.string.Password\_failure est considéré comme un entier par java. J’ai donc plutôt utilisé*

informUser =getResources().getString(R.string.***Password\_failure***);

ce qui fait bien ce que je désirais, c’est-à-dire mettre le string « Password\_failure » dans informUser.

03/05 : <http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-call-restful-webservice-in-android-part-3/>

library : <http://loopj.com/android-async-http/>

Prise en main de la librairie « Android Asynchronous Http Client » grâce à la documentation fournie à l’adresse suivante (<http://loopj.com/android-async-http/>). Récupération de données depuis le Service Web. Remplissage d’une liste à partir de ces données Json.

09/05 : comparaison voir tableau link :https://www.youtube.com/watch?v=\_7r\_vdwmW0o#t=25.883208

Pas possible de se connecter (.connect) au WebService car il faut l’exécuter en tache de fond. Android le bloque par défaut pour ne pas qu’il bloque le thread UI

Asynctask

<http://stackoverflow.com/questions/19624193/how-to-handle-return-value-from-asynctask>

<http://stackoverflow.com/questions/12575068/how-to-get-the-result-of-onpostexecute-to-main-activity-because-asynctask-is-a>

|  |  |  |
| --- | --- | --- |
| 363 down vote accepted  +50 | Google's Android Documentation Says that :  An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.  AsyncTask's generic types :  The three types used by an asynchronous task are the following:  Params, the type of the parameters sent to the task upon execution.  Progress, the type of the progress units published during the background computation.  Result, the type of the result of the background computation.  Not all types are always used by an asynchronous task. To mark a type as unused, simply use the type Void:  private class MyTask extends AsyncTask<Void, Void, Void> { ... }  You Can further refer : <http://developer.android.com/reference/android/os/AsyncTask.html>  Or You Can clear whats the role of AsyncTask by refering [Sankar-Ganesh's Blog](http://sankarganesh-info-exchange.blogspot.com/p/need-and-vital-role-of-asynctas-in.html) Well The structure of a typical AsyncTask class goes like : private class MyTask extends AsyncTask<X, Y, Z>  protected void onPreExecute(){  }  This method is executed before starting the new Thread. There is no input/output values, so just initialize variables or whatever you think you need to do.  protected Z doInBackground(X...x){  }  The most important method in the AsyncTask class. You have to place here all the stuff you want to do in the background, in a different thread from the main one. Here we have as an input value an array of objects from the type “X” (Do you see in the header? We have “...extends AsyncTask” These are the TYPES of the input parameters) and returns an object from the type “Z”.  protected void onProgressUpdate(Y y){  }  This method is called using the method publishProgress(y) and it is usually used when you want to show any progress or information in the main screen, like a progress bar showing the progress of the operation you are doing in the background.  protected void onPostExecute(Z z){  }  This method is called after the operation in the background is done. As an input parameter you will receive the output parameter of the doInBackground method.  **What about the X, Y and Z types?**  As you can deduce from the above structure:  X – The type of the input variables value you want to set to the background process. This can be an array of objects.  Y – The type of the objects you are going to enter in the onProgressUpdate method.  Z – The type of the result from the operations you have done in the background process.  How do we call this task from an outside class? Just with the following two lines:  MyTask myTask = new MyTask();  myTask.execute(x);  Where x is the input parameter of the type X.  Once we have our task running, we can find out its status from “outside”. Using the “getStatus()” method.  myTask.getStatus();  and we can receive the following status:  **RUNNING** - Indicates that the task is running.  **PENDING** - Indicates that the task has not been executed yet.  **FINISHED** - Indicates that onPostExecute(Z) has finished.  **Hints about using AsyncTask**   1. Do not call the methods onPreExecute, doInBackground and onPostExecute manually. This is automatically done by the system. 2. You cannot call an AsyncTask inside another AsyncTask or Thread. The call of the method execute must be done in the UI Thread. 3. The method onPostExecute is executed in the UI Thread (here you can call another AsyncTask!). 4. The input parameters of the task can be an Object array, this way you can put whatever objects and types you want.  |  | | --- | | [share](http://stackoverflow.com/a/6053673) | |