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

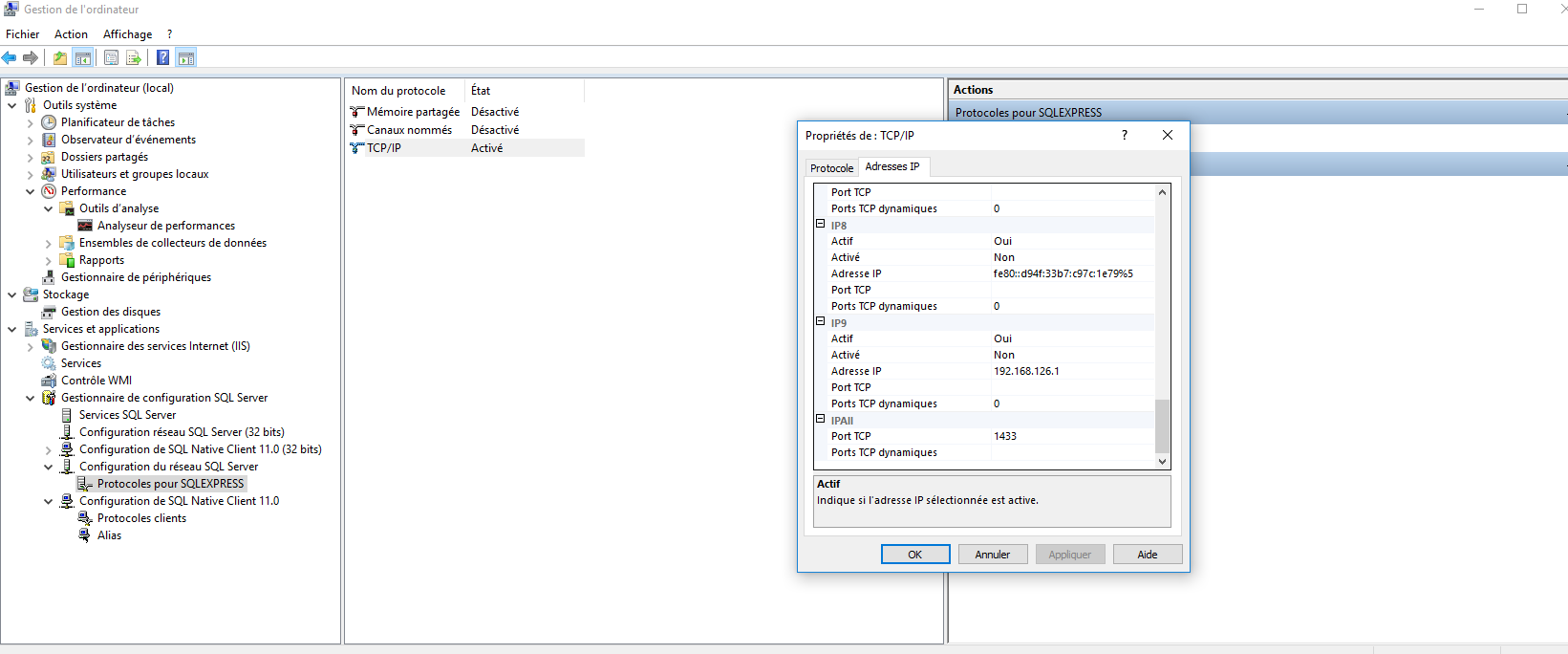
Pour la page de login, réalisation d’une partie

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.