Développement Mobile

Android : Client d'une BDD SqlServer

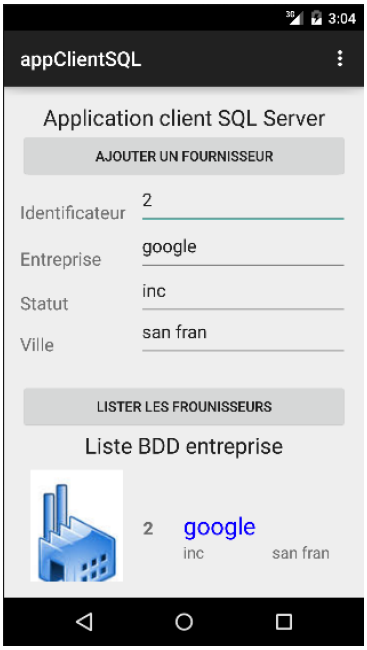
DURAND Valentin
TD 2 - TP 4
DUT Informatique
IUT Caen
2014 - 2015

IUT CAEN

Université de Caen
Basse-Normandie

Afin de développer cette application il était nécessaire dans un premier temps de créer le panneau de préférence permettant de fixer ou modifier les constante de connexion à la base de données.
Pour ce faire, le xml de l'activité préférence était déjà fourni, il ne reste plus qu'à créer la classe *SetPreferencesFragmentActivity* qui s'occupe de charge le fragment *PreferencesFragments* par dessus l'activité courante. Dans le *onCreate* du *PreferencesFragments* on charge le xml fourni, dans lequel on a spécifié au paravant les valeurs par défaut de connexion à la base.
Pour finir, il faut spécifier dans l'activité principale d'ouvrir l'activité préférence lors du click sur le bouton "settings". On ajouter dans un premier temps un sous bouton "Préférence" grâce à la méthode *onCreateOptionsMenu*, puis dans la méthode *onOptionsItemSelected* on spécifie l'ouverture de la vue de préférence par l'intermédiaire d'un intent.
Il est ensuite important de récupérer les informations de préférence dans l'activité principale afin de pouvoir les utiliser pour se connecter à la base. On créé donc une méthode *updateAttributsFromPreferences* qui attribut les valeurs récupérées dans le fragment de préférence via *getString* aux variables propre à la classe *MainActivity*.

Insertion dans la base



Activité principale de l'application android

Maintenant que les variables de connexion sont définies on peut instancier dans le *onCreate* la classe *ClientSQLmetier* avec ces variable dans le constructeur afin d'être connecté à la base et de pouvoir interagir avec celle-ci.
Pour insérer un fournisseur dans la base, on récupère les *EditText* grâce à leur *Id* spécifié dans le *XML* auxquels on applique les méthode *getText* et *toString* afin d'avoir leur contenu sous forme de string. Dans le cas de l'identifiant d'un fournisseur qui doit être nécessairement un entier on applique la méthode *parseInt*. La méthode *clickBtnAddFourn* se chargeant de l'insertion est lié au bouton soit simplement en ajoutant un *android:onClick* dans le *XML* correspondant à ce bouton, ou en instanciant un *OnClickListener* sur ce bouton dans l'activité principale.
Pour faire l'insertion en elle même, on créé un thread afin de pourvoir se connecter à internet (sans oublier de le spécifier dans le manifest) dans lequel on appelle la méthode *addNewFournisseur* sur l'instance de *ClientSQLmetier* avec en paramètres les variables récupérées des *EditText*. On utilise des *AlertDialog* pour avertir de la réussite (1) ou de l'échec (-1) de l'insertion, ainsi que lorsque les champs sont vide (méthode *matches*).

Sélection dans la base

Pour récupérer les données présentes dans la base on utilise la méthode *getTableFournisseurs* similairement à l'insertion (thread, try/catch, instance de *ClientSQLmetier*). Cette méthode nous retourne un objet du type *ResultSet* à partir duquel on peut extraire les informations de la base via la méthode *getString* avec en paramètre le nom de la colonne désirée, ou son index. Par une boucle while, à chaque row de la base on fait correspondre un objet de type Fournisseur que l'on ajoute à l'*ArrayList* stockant tous les fournisseurs à afficher.

Classe Fournisseur

Les objets de type *Fournisseur* sont des instances de la classe *Fournisseur*, qui a été créé afin de contenir un identifiant, un nom, un statut et une ville pour chaque fournisseur. Sont constructeur prend donc en paramètre chacune de ces variables. Cette classe implémente l'interface *Serializable* pour assurer la persistance de la vue.

La ListView

Pour la structure de *ListView*, on utilise le *XML* fourni, la classe *ArrayFournisseurAdapter* permet ensuite de remplir celle-ci avec l'*ArrayList* de fournisseurs. On instancie donc *ArrayFournisseurAdapter* dans le *onCreate* de l'activité principale avec en paramètre l'id *TextView* et

l'ArrayList de fournisseur. On ajoute cet adaptateur à la *ListView* via la méthode setAdapter. Enfin on appel la méthode *notifyDataSetChanged* à la fin de la méthode *LoadFournisseurList* afin de recharger la *ListView* à chaque click.

Suppression dans la base

Pour gérer un click long sur la *ListView* on instancie un *OnItemLongClickListener* que l'on ajoute à cette dernière par la méthode *setOnItemLongClickListener* qui appelle *onItemLongClick*. Cette méthode prend en paramètre la position de l'élément cliqué, on peut donc ainsi récupérer le fournisseur présent en cette position dans l'*ArrayList* et avoir son identifiant que l'on passe en paramètre de la méthode *removeOneFournisseur* qui se charge de la suppression.
Cette dernière utilise un thread comme pour l'ajout et la sélection, afin de se connecter à internet pour appeler la méthode deleteFournisseur sur l'instance de *ClientSQLmetier* avec en paramètre l'identifiant du fournisseur à supprimer. Comme pour l'ajout, des *AlertDialog* se charge d'avertir l'utilisateur de la réussite ou d'éventuelles erreurs. On utilise aussi une *AlertDialog* avec un *setPositiveButton* et *setNegativeButton* pour confirmer la suppression.

Rotation

Comme indiqué précédemment on a serialisé la classe *Fournisseur* en implémentant l'interface *Serializable*. Dans le *onCreate* de l'activité principale, si l'instance récupérée *savedInstanceState* n'est pas null on récupère *l'ArrayList* de fournisseur que l'on aura sauvegardé au préalable via la méthode *onSaveInstanceState*, autrement on instancie une nouvelle *ArrayList* de fournisseur que l'on remplira lors du click sur le bouton 'Lister les fournisseurs".