

TP3



Afficher le RecyclerView si le compte existe.

Création d'un nouveau compte s'il n'existe pas. Sinon on avertit l'utilisateur

L'objectif de ce Tp est la manipulation d'une simple base de données par Room Database pour gérer les login et les mots de passe via l'interface ci-dessus.

La structure et les consignes présentées ci-dessous vous guident à réaliser ce TP.



Environnement :

Ajouter dans votre gradle et le synchroniser:

```
//ROOM  
implementation 'android.arch.persistence.room:runtime:1.0.0'  
annotationProcessor "android.arch.persistence.room:compiler:1.0.0"
```

Etape 1 :

Développement des applications mobiles (Android)

- Pour la gestion de la zone de texte cliquable associé à la création d'un nouveau compte (avec changement de couleur) :
 - o Ajoutez dans le fichier layout principal, le TextView pour l'affichage de « Créer un nouveau compte » puis y ajoutez

```
android:textColor="@color/couleurs"  
android:clickable="true"  
android:selectAllOnFocus="true"
```



<http://developer.android.com/reference/android/widget/TextView.html>

- o Ajoutez le dossier color (dans res) contenant le fichier couleurs.xml qui contient les items suivantes :

```
<selector xmlns:android="http://schemas.android.com/apk/res/android" >  
  <item android:state_selected="true" android:color="#87E990" />  
  <item android:state_focused="true" android:color="#FF8000" />  
  <item android:state_pressed="true" android:color="#0000FF" />  
  <item android:color="#000000" />  
</selector>
```



<http://developer.android.com/guide/topics/resources/color-list-resource.html>

Etape 2 :

- Construction de la class User (Table users) avec annotations

```
@Entity(tableName = "users")  
public class User {  
  
  @PrimaryKey  
  @NonNull  
  public String login;  
  
  @ColumnInfo(name = "user_pass")  
  public String pass;  
}
```

Complétez la classe avec les méthodes `getID()`, `setID(int)`, `getLogin()`, `setLogin(string)`, `getPass()`, `setPass(string)` que vous pouvez générer automatiquement .Ex :

```
public String getLogin() {  
    return login;  
}  
  
public void setLogin(String login) {  
    this.login = login;  
}
```

Etape 3 :

Ajouter l'interface user Dao qui permet de manipuler les données de la classe « Table » User

```
@Dao  
public interface Usr_dao {  
  
    @Insert (onConflict=OnConflictStrategy.IGNORE)  
    public void adduser(User user);  
  
    @Query("select * from users where login = :login")  
    public List<User> getUser(String login);  
  
    @Query("select * from users")  
    public List<User> getUsers();  
  
    @Delete  
    public void deletuser(User user);  
  
    @Update  
    public void updatuser(User user);  
}
```



Etape 4 :

- Ajoutez la classe MyDatabase qui étend RoomDatabase et qui permet de **lier** votre classe User (table) User et votre user DAO à la base de données:

```
@Database(entities = {User.class }, version = 1)

public abstract class MyDatabase extends RoomDatabase {

    public abstract Usr_dao mydao();

}
```

Etape 5 :

Dans la classe principale ajoutez les deux variables :

```
static User user= new User();
static MyDatabase mydatabase;
```

Ajoutez dans le onCreate de l'activité:

```
connecter= (Button) this.findViewById(R.id.button);
login = (EditText) this.findViewById(R.id.login);
pass = (EditText) this.findViewById(R.id.pass);
Text = (TextView) this.findViewById(R.id.nouv);
mydatabase =
Room.databaseBuilder(getApplicationContext(),MyDatabase.class,
"user_bd").allowMainThreadQueries().build() ;
```

- o Ajouter pour connecter et text des listeners et Dans le onClick(View v) appelez votre méthode Myclick.

- o Dans Myclick :

```
private void myClick(View v) {
    user.login = login.getText().toString();
    user.pass = pass.getText().toString();
    if ( !(login.getText().toString().matches("")) ) {
        List<User> usr =
MainActivity.mydatabase.mydao().getUser(login.getText().toString());

        switch (v.getId()) {
            case R.id.nouv:
                if (usr.size() != 0) {
```

