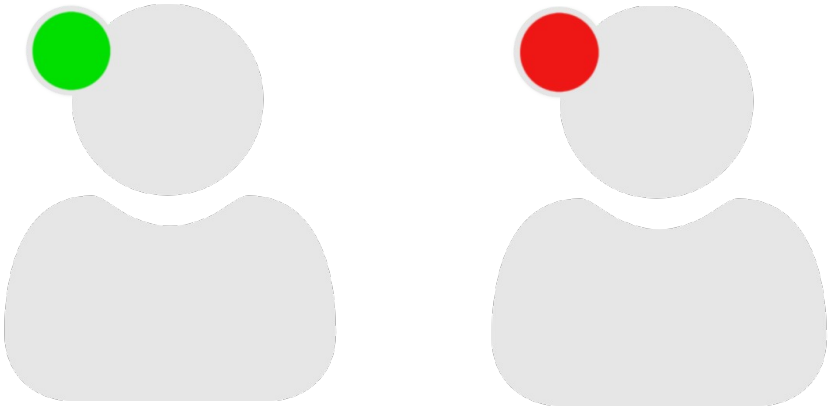
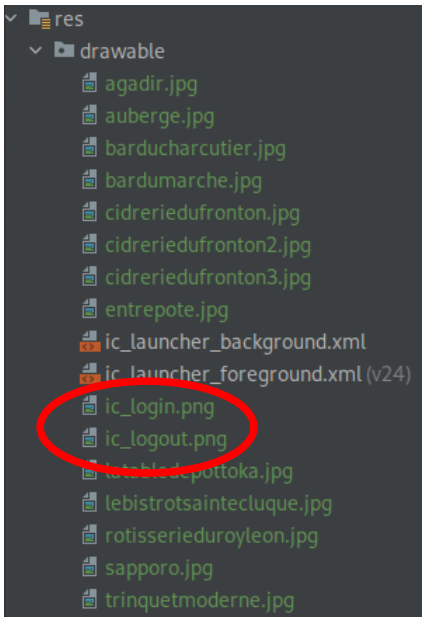
Cette fonctionnalité représente la première amélioration entreprise et développée entièrement seul. Elle consiste en la modification de la barre d'action afin d'obtenir un rendu plus esthétique et de proposer à l'utilisateur de pouvoir se connecter/déconnecter et accéder à son profil directement en interagissant avec celle-ci.

1 – Création de l'icône de connexion

Dans un premier temps, je souhaitais pouvoir afficher une icône de connexion/déconnexion cliquable, qui évoluerait dynamiquement afin d'informer l'utilisateur de son statut de connexion. Pour ce faire, il m'a fallu designer deux icônes avec Adobe Design.



J'ai ensuite ajouté ces images dans le dossier drawable de mon projet.



2 – Création de la vue

J'ai ajouté un fichier « menu_main.xml » dans un dossier « menu » dans « res ». Ce fichier fait office de vue pour la barre d'action.

MainActivity.java | menu_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- res/menu/menu_main.xml -->
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_login"
        android:icon="@drawable/ic_login"
        android:title="Connexion"
        app:showAsAction="ifRoom"/>
    <item
        android:id="@+id/action_logout"
        android:icon="@drawable/ic_logout"
        android:title="Déconnexion"
        app:showAsAction="ifRoom"/>
</menu>
```

3 – Ajout des méthodes dans les contrôleurs

Afin d'assurer le bon fonctionnement de cette nouvelle barre d'action, il m'a fallu développer 4 méthodes et les insérer dans chacun des contrôleurs.

1. onCreateOptionsMenu

Cette méthode est appelée une seule fois pour initialiser le menu d'options lorsque l'activité est créée. Elle sert à « gonfler » (inflate) le menu à partir des ressources XML, ce qui ajoute les éléments définis dans le fichier XML ('menu_main.xml') à la barre d'action de l'activité si elle est présente.

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Influe le menu; ajoute les items à la barre d'action si elle est présente.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

2. onPrepareOptionsMenu

Cette méthode est appelée chaque fois que le menu est sur le point d'être affiché. Elle permet de modifier les éléments du menu dynamiquement juste avant qu'ils ne soient affichés. C'est utile pour activer/désactiver des éléments ou modifier leur visibilité basée sur certains états de l'application.

```java
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    // Prépare les items du menu avant qu'ils soient affichés.
    super.onPrepareOptionsMenu(menu);
    boolean isLoggedIn = checkLoginStatus();  // Vérifier l'état de connexion
    menu.findItem(R.id.action_login).setVisible(!isLoggedIn);
    menu.findItem(R.id.action_logout).setVisible(isLoggedIn);
    return true;
}
```

3. checkLoginStatus

Cette méthode aide à déterminer si l'utilisateur est connecté en consultant les « SharedPreferences », un système de stockage de paires clé-valeur léger utilisé pour sauvegarder des préférences ou des états d'application.

```
private boolean checkLoginStatus() {
    // Vérifie le statut de connexion à partir des SharedPreferences
    SharedPreferences prefs = getSharedPreferences( name: "UserPrefs", MODE_PRIVATE);
    return prefs.getBoolean( s: "isLoggedIn",  b: false);
}
```

4. onOptionsItemSelected

Cette méthode est appelée chaque fois qu'un élément du menu est sélectionné. Elle permet de gérer les actions à réaliser lorsque les options du menu sont utilisées.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Gestion des actions de la barre d'action.
    int id = item.getItemId();
    if (id == R.id.action_login) {
        // Lancer l'activité de connexion
        Intent intent = new Intent( packageContext: this, LoginActivity.class);
        startActivity(intent);
        return true;
    } else if (id == R.id.action_logout) {
        // Naviguer vers le profil de l'utilisateur
        Intent intent = new Intent( packageContext: this, ProfilActivity.class);
        startActivity(intent);
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Ces méthodes ensemble permettent de contrôler et de gérer la barre d'action et le menu d'options de l'application, offrant ainsi une navigation et des fonctionnalités accessibles depuis n'importe quelle partie de l'application.

4 – Modification du nom de l'application

Pour un meilleur rendu, j'ai configuré la barre d'action avec un titre personnalisé. Pour cela, rien de plus simple :

```
// Configuration de la barre d'action avec un titre
if (getSupportActionBar() != null) {
    getSupportActionBar().setDisplayShowTitleEnabled(true);
    getSupportActionBar().setTitle("R3STO.FR");
}
```

Il suffit d'ajouter cette partie de code au sein des contrôleurs.

4 – Jeu d'essai

Pour ce jeu d'essai il convient de vérifier 3 éléments :

- affichage de l'icône indiquant que l'utilisateur n'est pas connecté ;
- affichage de l'icône indiquant que l'utilisateur est connecté ;
- affichage du titre de l'application.

Utilisateur non connecté

Utilisateur connecté

Titre de l'application