Action Bar

1. Introduction:

L'Action Bar ou App Bar est l'un des éléments les plus importants d'une activité, il offre une vue simple tout en haut avec la possibilité d'ajouter des actions contextuelles.
Elle permet à l'utilisateur de se localiser dans l'appli, facilite la navigation entre les différents vues et donne un accès immédiat aux actions les plus importantes comme la recherché…

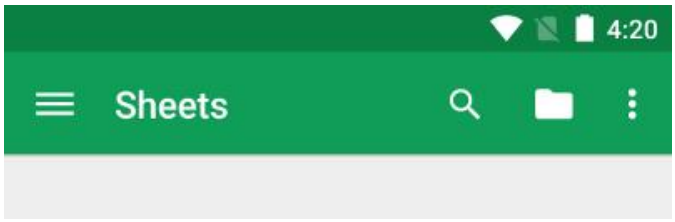

Figure 1 : exemple d'Action Bar.

Pour intégrer une Action Bar à une activité, vous avez deux choix possibles :

- **Utiliser un thème d'activité qui propose une Action Bar** : vous pouvez d'ailleurs remarquer que lors de la création d'une nouvelle `EmptyActivity`, Android studio vous crée une activité avec un thème qui intègre une Action Bar, mais limité en terme de fonctionnalités et de personnalisation. Pour ce TP nous allons optez pour la deuxième solution.
- **Utiliser un objet de type Toolbar** : la solution conseillé par Google qu'on va voir en détails tout au long de ce TP.

2. Une simple Action bar:

Pour commencer, je vous propose de réaliser une Action Bar basique, qui va afficher le titre de l'activité.

Il faut tout d'abord créer une activité qui doit impérativement hériter de la classe `AppCompatActivit`.

```
public class MainActivity extends AppCompatActivity {

  // ...
}
```

Après sa création, la nouvelle activité dispose déjà d'une Action Bar par défaut, car cette dernière utilise un thème. Pour éviter ça, on rajoute dans le `manifest` la ligne suivante, pour éliminer l'Action Bar générée par le thème.

```
<application
android:theme="@style/Theme.AppCompat.Light.NoActionBar"
>
```

Maintenant qu'on a enlevé l'Action Bar du thème, on va créer la notre. Sur le `layout` de notre activité, on rajoute le code suivant.

```
<android.support.v7.widget.Toolbar
android:id="@+id/my_toolbar"
android:layout_width="match_parent"
android:layout_height="?attr/actionBarSize"
android:background="?attr/colorPrimary"
android:elevation="4dp"
android:theme="@style/ThemeOverlay.AppCompat.ActionBar"
app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>
```

Il ne reste plus qu'à initialiser l'objet de type `Toolbar` dans la méthode `onCreate()` de notre activité.

```
Toolbar myToolbar = (Toolbar) findViewById(R.id.my_toolbar);
setSupportActionBar(myToolbar);
```

Il faut importer android.support.v7.widget.Toolbar pour la classe Toolbar

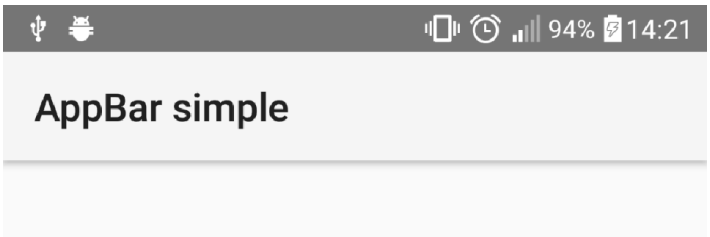

Figure 2 : Action Bar simple: résultat

3. Avec des boutons, c'est mieux:

Maintenant qu'on dispose d'une Action Bar simple, on peut aller un peu plus loin on ajoutant des boutons.
Pour rajouter des boutons sur notre App Bar, on commence par créer un nouveau répertoire `menu` dans notre répertoire de ressources `res`
En suite on crée un fichier `monmenu.xml` qui va contenir les boutons de notre menu et on le place dans le nouveau répertoire `menu`.
Il en résulte : `res/menu/monmenu.xml`
Le code ci-dessous de `monmenu.xml` définit trois boutons, 2 avec des icônes qui vont s'afficher normalement, et le troisième sans icône qui va s'afficher sous forme d'un sous menu que l'on verra dans la capture d'écran pour cette partie.

```
<menu xmlns:android=http://schemas.android.com/apk/res/android
xmlns:app="http://schemas.android.com/apk/res-auto">
<item
android:id="@+id/action1"
android:icon="@android:drawable/btn_star_big_on"
android:title="@string/action_button"
app:showAsAction="ifRoom"/>
<item
android:id="@+id/action2"
android:icon="@android:drawable/btn_star_big_off"
android:title="@string/action_button"
app:showAsAction="ifRoom"/>
<item android:id="@+id/action3"
android:title="@string/action_settings"
app:showAsAction="never"/>
</menu>
```

Après la création de notre menu, il faut l'intégrer dans notre activité, pour cela on définit la méthode `onCreateOptionsMenu()` dans notre `MainActivity`.

```
public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.monmenu, menu);
      return true;
}
```

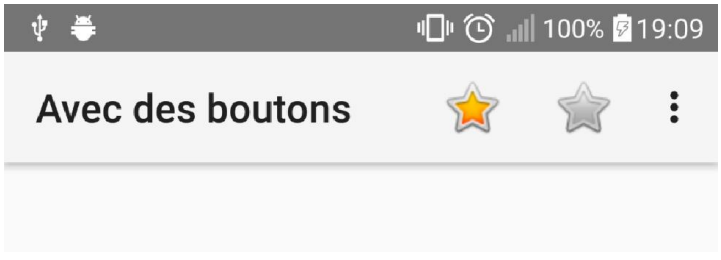

Figure 3 : Action Bar avec boutons: résultat

4. Des boutons avec des actions, c'est encore mieux !

Il est temps de passer aux choses sérieuses, il faut bien que ces boutons fassent quelque chose.
Pour ça, il faut redéfinir la fonction `onOptionsItemSelected()` pour qu'elle prend en charge les interactions avec nos boutons. Dans notre `MainActivity` on redéfinit la fonction comme suit.
Pour cet exemple j'ai choisit d'afficher un simple `toast`, mais ce n'est qu'un exemple, et les possibilités restent infinis.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
      Context context = getApplicationContext();
      CharSequence text = "YOU'R A STAR !";
      int duration = Toast.LENGTH_SHORT;
      Toast toast = Toast.makeText(context, text, duration);
      switch (item.getItemId()) {
            case R.id.action1:
            toast.show();
            return true;
            case R.id.action2:
            toast.show();
            return true;
            default:
            return super.onOptionsItemSelected(item);
      }
}
```

5. Action View

Une Action View est une action qui fournit des fonctionnalités plus riches dans l'Action Bar comme la fameuse barre de recherche qui apparait dans l'Action Bar et qui permet de faire une recherche sans avoir à changer d'activité.
Un widget est disponible sur Android pour cette barre de recherche, comme on peut créer nos propre Action View avec différentes fonctionnalités selon nos besoins, il faudra un peu de courage bien sûr.
Pour ajouter une `SearchView` à notre Action Bar, on rajoute un item dans notre fichier `monmenu.xml`

```
<item android:id="@+id/action_search"
android:title="@string/action_search"
android:icon="@android:drawable/ic_menu_search"
app:showAsAction="ifRoom|collapseActionView"
app:actionViewClass="android.support.v7.widget.SearchView"/>
```

Pour pouvoir interagir avec ce *widget*, il faut créer un objet `SearchView` qui, grâce a des méthodes comme `getQuery()` qui permet de récupérer le texte entré par l'utilisateur et grâce aux différents event *listener* que l'on peut ajouter, va permettre d'effectuer des recherches à partir de lAction Bar. Je donne un exemple d'initiation de l'objet `SearchView` sans entrer dans les détails de la fonction de recherche que l'on ne va pas aborder dans ce TP.

```
MenuItem searchItem = menu.findItem(R.id.action_search);
SearchView searchView =(SearchView) MenuItemCompat.getActionView(searchItem);
```

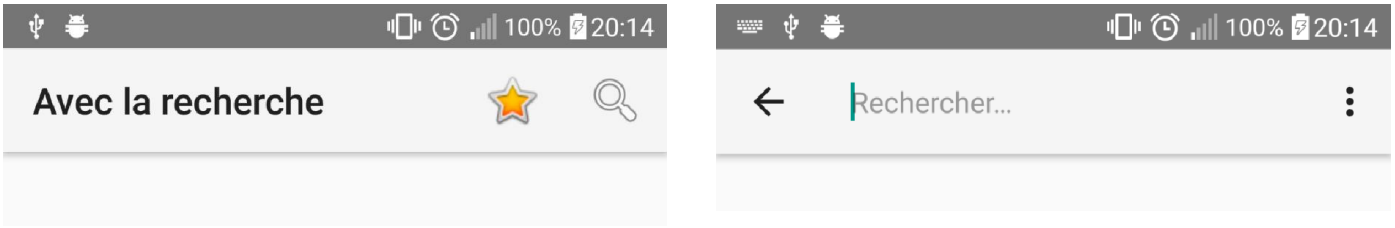

Figure 4 : searchView : résultat