TP°3 : Effectuer des traitements longs

Exercice 1 : Exécuter une tâche longue dans Handler (1)

Corriger l'Activity Exercice1Activity pour qu'elle exécute la fonction workToDo() dans un background Thread :

- Définir en attribut un Handler pour que le background Thread communique avec l'UI Thread
- Au clic sur le bouton Charger, créer et lancer un Thread
- Dans le Thread, avant d'appeler la fonction workToDo(), utiliser le Handler pour cacher le bouton et afficher la progress bar à l'aide de la méthode post()
- Après avoir appelé la fonction, utiliser le Handler pour réafficher le bouton et cacher la progress bar à l'aide de la méthode post()
- Vérifier dans les logs que la tâche a bien été effectuée

Exercice 2 : Exécuter une tâche longue dans Handler (2)

Corriger l'Activity Exercice2Activity pour qu'elle exécute la fonction workToDo() dans un background Thread :

- Définir en attribut un Handler pour que le background Thread communique avec l'UI Thread
- Implémenter la méthode handlerMessage du Handler pour :
  - cacher le bouton et afficher la progress bar à la réception du message START
  - afficher le bouton et cacher la progress bar à la récepction du message DONE
- Au clic sur le bouton Charger, créer et lancer un Thread
- Dans le Thread, avant d'appeler la fonction workToDo(), utiliser le Handler pour envoyer le message START à l'aide de la méthode sendEmptyMessage()
- Après avoir appelé la fonction, utiliser le Handler pour envoyer le message DONE à l'aide de la méthode sendEmptyMessage()
- Vérifier dans les logs que la tâche a bien été effectuée

Exercice 3 : Exécuter une tâche longue dans une AsyncTask

Réaliser la même chose que l'exercice 1 et 2 mais à l'aide d'un AsyncTask :

- Créer une classe étendant la classe AsyncTask
- Dans la méthode onPreExecute, cacher le bouton et afficher la progress bar
- Dans la méthode doInBackground, exécuter la fonction workToDo()
- Dans la méthode onPostExecute, afficher le bouton et cacher la progress bar
- Au clic sur le bouton, exécuter l'AsyncTask

Exercice 4 : Exécuter une tâche longue dans une AsyncTask avec paramètre et résultat

Corriger l'Activity Exercice4Activity pour qu'elle exécute la fonction *workToDo* (param) dans une AsyncTask :

- Dans la méthode onPreExecute, cacher le bouton et afficher la progress bar
- Dans la méthode doInBackground, récupérer la valeur passée en paramètre lors de l'exécution de l'AsynsTask, exécuter la fonction *workToDo*(param) avec ce paramètre et retourner le résultat
- Dans la méthode onPostExecute, afficher le résultat passé en paramètre dans la TextView R.id.text, afficher le bouton et cacher la progress bar
- Au clic sur le bouton, exécuter l'AsyncTask en passant en paramètre la valeur de l'EditText R.id.edit_text

Exercice 5 : Exécuter une tâche longue dans une AsyncTask avec suivi d'avancement

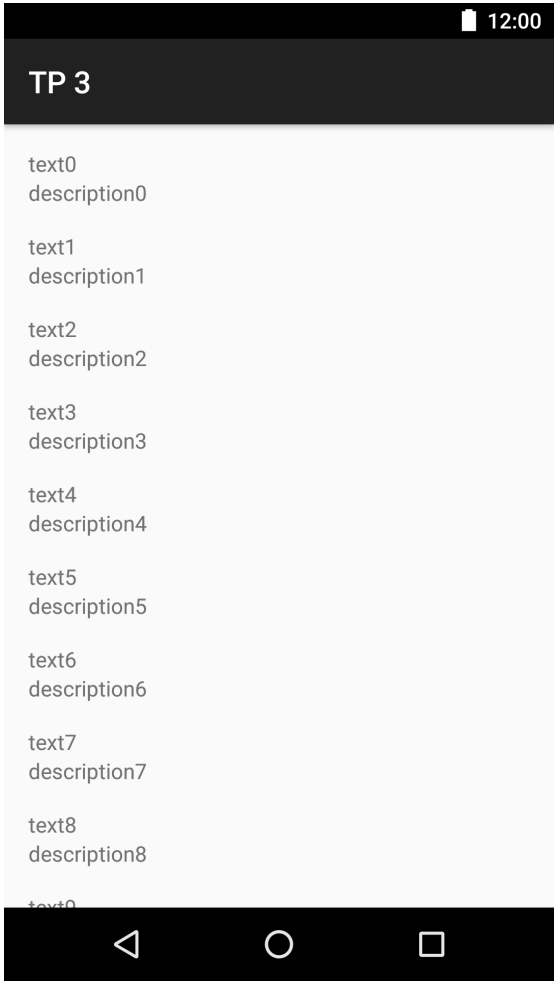
Corriger l'Activity Exercice4Activity pour qu'elle exécute la fonction *smallWorkToDo*() dans une AsyncTask :

- Dans la méthode onPreExecute, cacher le bouton et afficher la progress bar
- Dans la méthode onProgressUpdate, afficher l'avancement actuel de la tâche dans la TextView R.id.text
- Dans la méthode doInBackground, exécuter la fonction *smallWorkToDo*() et publier l'avancement actuel en appelant publishProgress dans la boucle
- Dans la méthode onPostExecute, afficher le bouton et cacher la progress bar
- Au clic sur le bouton, exécuter l'AsyncTask

TP°3 : Utilisation des RecyclerViews

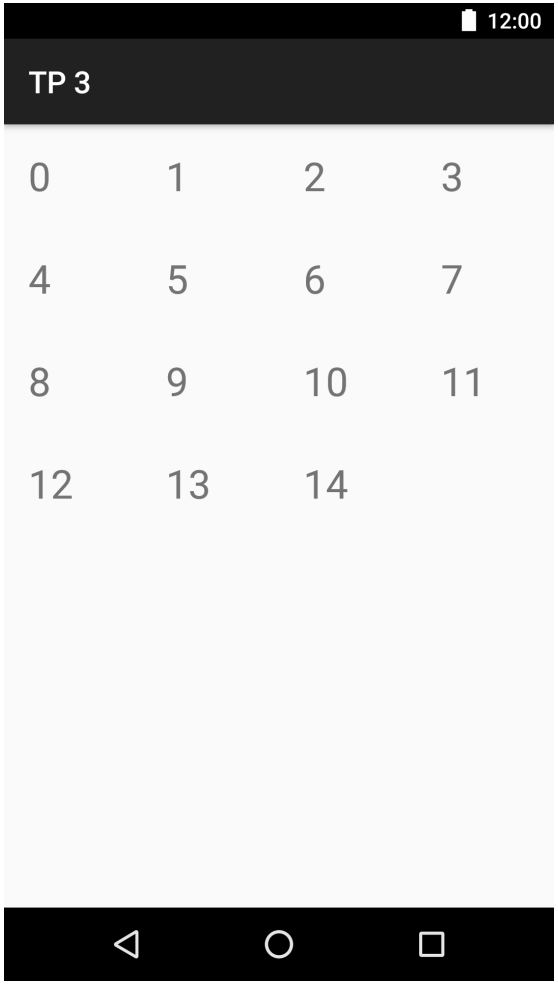
Exercice 6 : Affichage de données simples sous forme de liste

- Dans l'Activity Exercice6Activity, afficher les données retournées par la fonction DataGenerator.getDataForSimpleList() dans une RecyclerView sous forme de liste.
- Utiliser le layout R.layout.exercice_06_item_list pour l'affichage des items de la liste



Exercice 7 : Affichage de données simples sous forme de grille

- Dans l'Activity Exercice7Activity, afficher les données retournées par la fonction DataGenerator.getDataForSimpleGrid() dans une RecyclerView sous forme de grille.
- Utiliser le layout R.layout.exercice_07_item_grid pour l'affichage des items

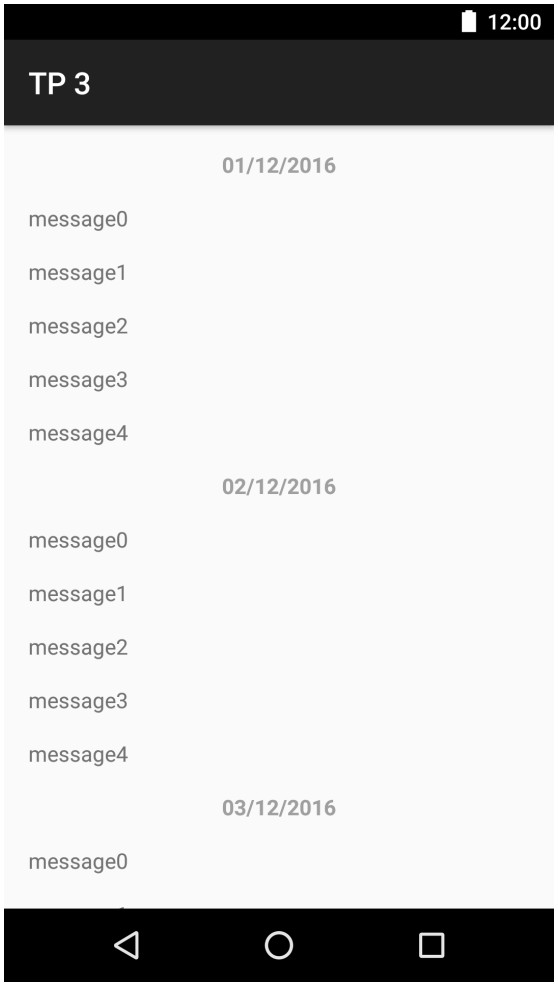


Exercice 8 : Affichage de deux types de données différentes sous forme de liste

- Dans l'Activity Exercice8Activity, afficher les données retournées par la fonction DataGenerator.getDataForTwoTypesList () dans une RecyclerView sous forme de liste.
- Utiliser le layout R.layout.exercice_08_1_item_list pour l'affichage du type Date
- Utiliser le layout R.layout.exercice_08_2_item_list pour l'affichage du type String

Info

Utiliser la fonction Utils.*formatDate()* pour convertir une Date sous forme de texte.



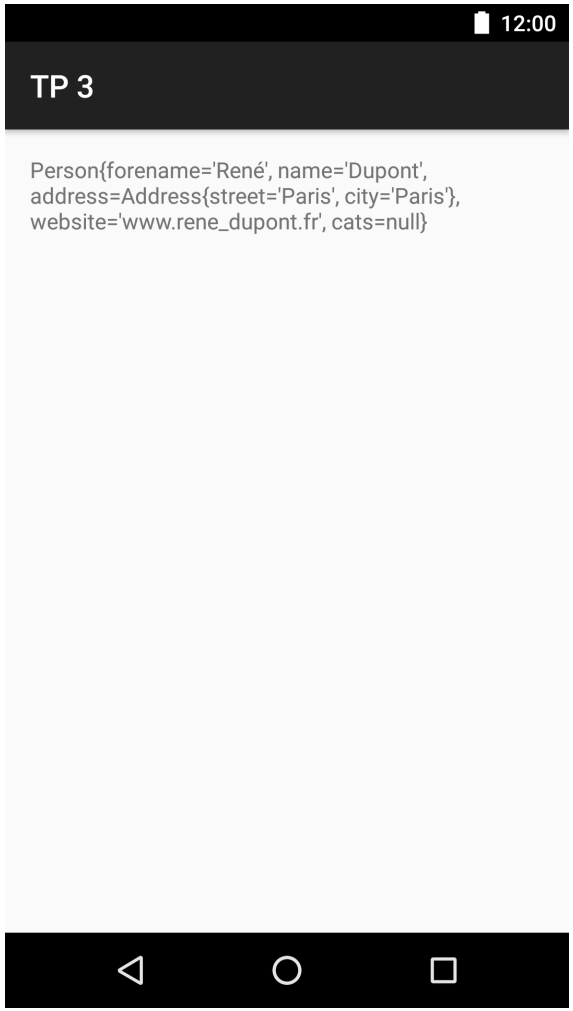
Exercice 9 : Implémenter un listener click dans une liste simple

- Dans l'Activity Exercice9Activity, afficher les données retournées par la fonction DataGenerator.getDataForSimpleList() dans une RecyclerView sous forme de liste.
- Au clic sur un item de la liste, afficher le contenu de l'item à l'aide de la méthode Utils.showMessage() et ToString() de la classe TextDescription

TP°3 : Appel réseau

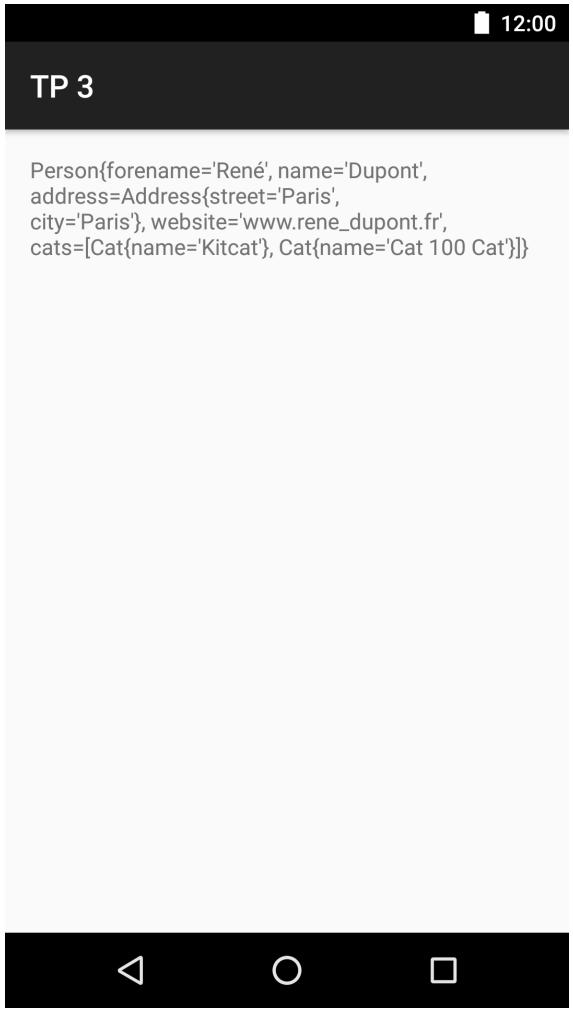
Exercice 10 : Parser un texte JSON simple

- Dans l'Activity Exercice10Activity, parser le JSON retourné par la fonction DataGenerator.getSingleJsonData() en l'objet Person.
- Afficher l'objet Person obtenu dans la TextView R.id.text à l'aide de la méthode toString()



Exercice 11 : Parser un texte JSON contenant une liste

- Dans l'Activity Exercice11Activity, parser le JSON retourné par la fonction DataGenerator.getListJsonData() en l'objet Person.
- Afficher l'objet Person obtenu dans la TextView R.id.text à l'aide de la méthode toString()



Exercice 12 : Effectuer un appel réseau asynchrone

- Dans l'Activity Exercice12Activity, effectuer un appel réseau en GET sur l'url https://www.google.com
  - Créer une Request
  - Utiliser le client OkHttp retourné par Webservice.*getOkHttpClient*() pour effectuer un appel asynchrone
- Vérifier dans les logs que Google a bien répondu :

```
V/Webservice: --> GET https://www.google.com/ http/1.1
V/Webservice: --> END GET
D/OpenGLRenderer: endAllStagingAnimators on 0xb4ce3c80 (ListView) with handle 0xa063f5d0
V/Webservice: <-- 200  https://www.google.fr/?gfe_rd=cr&ei=Iaw_WLP5Ca338Aeh2ZiAAg (966ms)
V/Webservice: date: Thu, 01 Dec 2016 04:50:41 GMT
V/Webservice: expires: -1
V/Webservice: cache-control: private, max-age=0
V/Webservice: content-type: text/html; charset=ISO-8859-1
V/Webservice: p3p: CP="This is not a P3P policy! See https://www.google.com/support/accounts/answer/151657?hl=en for more info."
V/Webservice: server: gws
V/Webservice: x-xss-protection: 1; mode=block
V/Webservice: x-frame-options: SAMEORIGIN
V/Webservice: set-cookie: NID=91=wc46FIXQcih9ByLh94k_4xjMVn6WpSQ_NjBZ-R9EtUfPIjozL6nKCYrBhi_SuoiE1ac0wT4qss9-GbheDCZtxfJQPZeCUwGm
V/Webservice: alt-svc: quic=":443"; ma=2592000; v="36,35,34"
V/Webservice: OkHttp-Sent-Millis: 1480567839628
V/Webservice: OkHttp-Received-Millis: 1480567839751
V/Webservice: <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="fr"><head><meta content="text/html; ch
              </style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;overflow-y:scroll}#gog{padding:3px 8px 0}t
V/Webservice: <-- END HTTP (10468-byte body)
```

TP°3 : Mise en pratique des trois parties

Exercice 13 : Effectuer un appel réseau et mettre à jour l'interface graphique

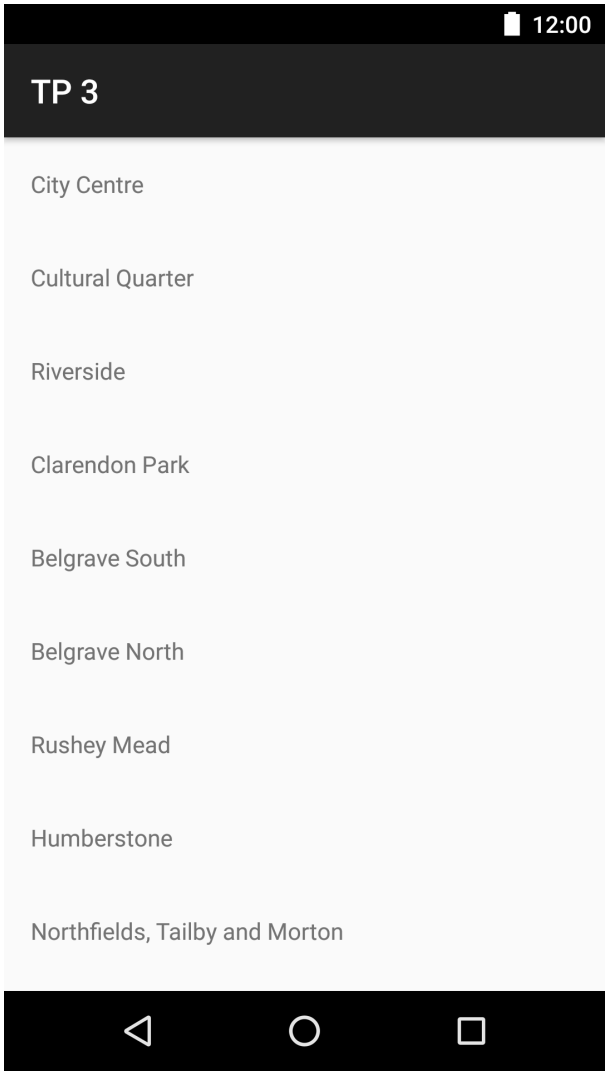
- Dans l'Activity Exercice13_1Activity, effectuer un appel réseau en GET sur l'url **https://data.police.uk/api/leicestershire/neighbourhoods** pour récupérer une liste de tous les quartiers d'Angleterre
- Définir un handler pour gérer les retours de l'appel
- Avant d'effectuer l'appel, afficher la progress bar et cacher le bouton
- Si l'appel échoue, utiliser le handler pour cacher la progress bar et afficher le bouton Réessayer
- Au clic sur le bouton Réessayer, afficher la progress bar, cacher le bouton et réeffectuer un appel
- Si l'appel fonctionne, utiliser le handler pour cacher la progress bar, le bouton et pour afficher le résultat brut dans la TextView R.id.text

Info

Tester l'echec d'un appel en passant le téléphone en mode avion

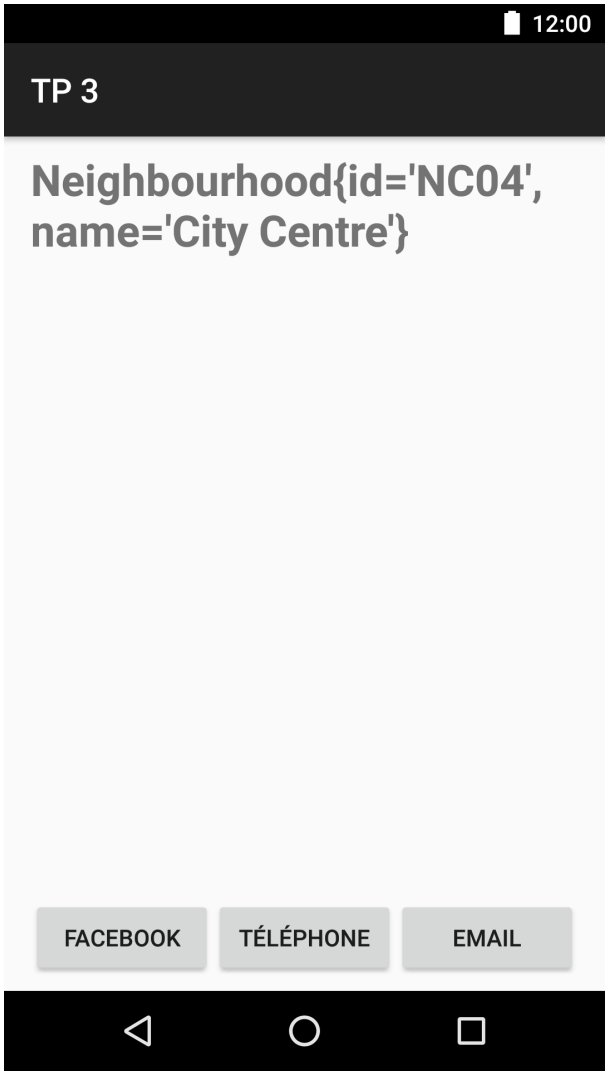
Exercice 14 : Afficher une liste de quartiers

- Modifier l'Activity Exercice13Activity pour que le résultat de l'appel à **https://data.police.uk/api/leicestershire/neighbourhoods** soit affiché dans une liste :
  - Remplacer la TextView par une RecyclerView dans le layout
  - Parser le résultat obtenu par l'appel Webservice en liste d'objets Neighbourhood
  - Utiliser la RecyclerView pour afficher cette liste



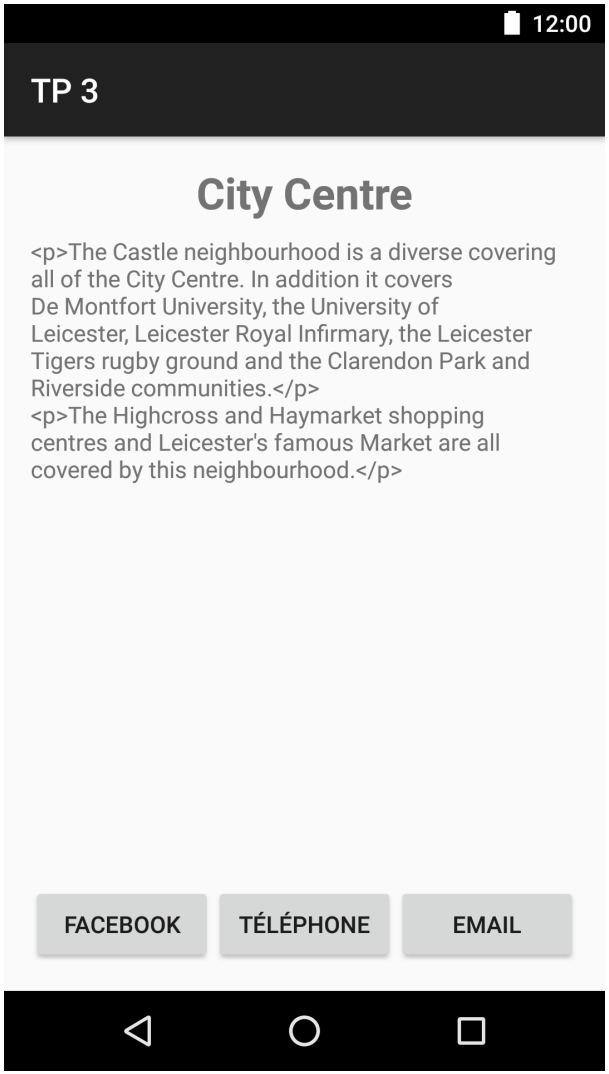
Exercice 15 : Afficher une liste de quartiers

- Implémenter un listener click sur les items de la liste
- Au clic que un item, lancer l'Activity Exercice13_2Activity en passant le Neighbourhood de l'item en paramètre
- Afficher le Neighbouthood dans la TextView R.id.name



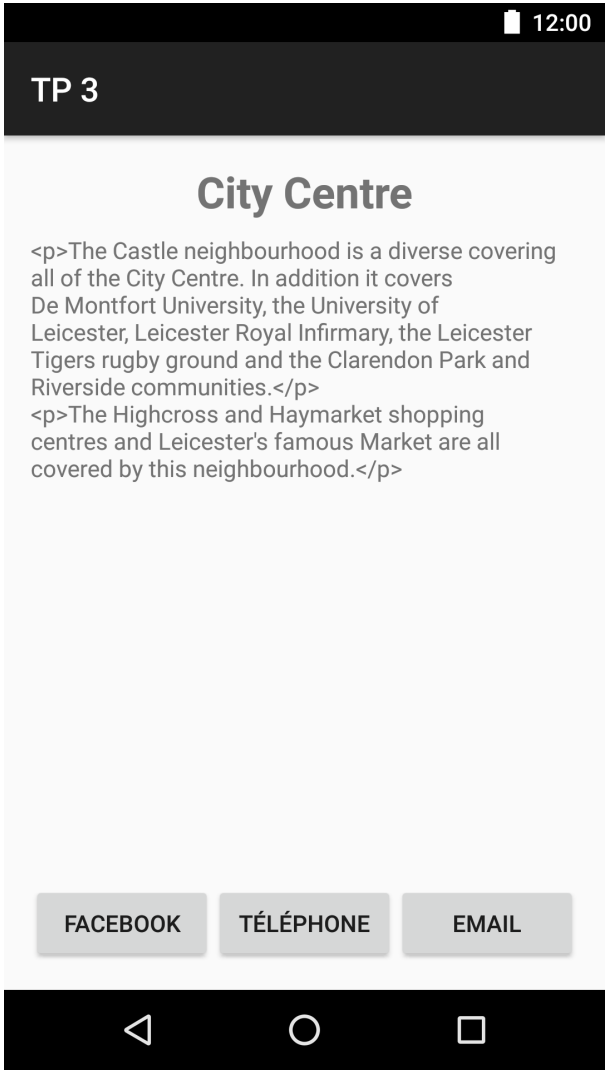
Exercice 16 : Afficher le détail d'un quartier

- Sur l'Activity Exercice13_2Activity, récupérer le Neighbourhood passé en paramètre et concaténer l'id à l'url https://data.police.uk/api/leicestershire/
- En suivant le même procédé que l'exercice 13, effectuer un appel réseau GET sur l'url https://data.police.uk/api/leicestershire/ **+ id**
- Si l'appel échoue, afficher le bouton et cacher la progress bar
- Si l'appel fonctionne
  - Afficher le ViewGroup R.id.content
  - Parser le retour en objet NeighbourhoodDetail en récupérant uniquement les champs nécessaires
  - Mettre à jour le ViewGroup avec les informations de l'objet



Exercice 17 : Ajout des implicit intents

- Au clic sur le bouton Facebook, ouvrir l'url facebook renseigner dans l'objet NeighbourhoodDetail
- Au clic sur le bouton Téléphone, ouvrir l'application téléphone
- Au clic sur le bouton Email, ouvrir l'application Email



Info

Pour ouvrir une page web :

```
new Intent(Intent.ACTION_VIEW, Uri.parse(url);
```

Pour ouvrir une application mail :

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, new String[]{ mail });
intent.setType("message/rfc822");
```