1 Modalités

Le projet est à réaliser en binôme (ou, éventuellement, en monôme). Vous avez le temps jusqu'à 7 novembre pour constituer les groupes et s'inscrire sur moodle (les deux membres de chaque binôme doivent s'inscrire).
Les soutenances auront lieu pendant la semaine du 17 décembre, la date précise sera donnée ultérieurement. Pendant la soutenance chaque membre du binôme doit montrer la maîtrise de la totalité du code.
Vous devez déposez la totalité du projet sur moodle comme un seul fichier archive soit `zip` soit `tar.gz` (pas de fichier `rar`) au plus tard la veille du jour des soutenances 20h00.

2 Lecteur mplrss de fils RSS

Le projet consiste à écrire un lecteur de fils RSS, qu'on nommera par la suite `mplrss` (mon propre lecteur rss).
Les fils RSS sont de simples fichiers xml dont la structure est décrite par exemples sur ces pages :

- `http://www.xmlfacile.com/guide_xml/flux_rss_1.php5`,
- `https://validator.w3.org/feed/docs/rss2.html`, ou
- `https://www.w3schools.com/xml/xml\_rss.asp`.

Même si c'est une technologie ancienne il existe toujours beaucoup de fils rss qu'on trouve facilement avec un moteur de recherche.
L'utilisateur du lecteur `mplrss` donnera l'adresse http (ou https), soit en l'écrivant dans un `EditText` soit en choisissant dans la base de données de votre application. L'application doit charger le fichier xml indiqué, l'analyser et afficher les informations contenues dans le fichier.
Puisque le chargement de fichier par internet est une opération qui peut prendre un certain temps, pour ne pas bloquer le thread principal de votre application, le chargement doit s'effectuer à l'aide de `DownloadManager`. Vous installerez un `BroadcastReceiver` qui sera averti quand le chargement est fini. L'utilisateur doit avoir la possibilité d'interrompre le chargement à tout moment. Pendant le chargement l'utilisateur doit être informé que le chargement est en cours, par exemple par un message approprié accompagné d'un `ProgessBar`.

2.1 Parseurs XML

Une fois le fichier rss xml chargé il convient d'informer l'utilisateur de la fin du chargement. Pour analyser le document xml obtenu il existe (au moins) deux parseurs java pour : `DocumentBuilder` (un parseur DOM) et `SAXParser` (les deux parseurs se trouvent dans le package `javax.xml.parsers`, notez que ce ne sont pas des classes d'Android mais bien les classes de java 7).
Pour les deux parseurs vous trouverez les tutoriels sur le parsing de fichiers xml sur le site d'Oracle :

- `https://docs.oracle.com/javase/tutorial/jaxp/dom/index.html`
- `https://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html`

`DocumentBuilder` est un parseur DOM, c'est-à-dire il construit un arbre DOM (Document Object Model) du document xml (un objet de la classe `org.w3c.dom.Document`). Ce parseur convient bien pour de petits fichiers xml, pour de gros fichiers le temps de construction de l'arbre DOM peut être prohibitif.
L'utilisation de `DocumentBuider` est très simple.
Dans le code ci-dessous `path` est un String qui contient l'uri donnant l'emplacement local, sur votre appareil Android, de fichier rss xml.

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document document = db.parse(uri_path);
```

Une fois vous avez l'objet `Document` vous pouvez utiliser les méthodes de la classe `Document` pour retrouver le contenu de différentes balises xml.
Dans l'exemple ci-dessus `uri_path` est un String qui contient uri du document xml. A priori cela peut être uri un d'un document quelconque sur le web mais **nous exigeons** que vous devez d'abord charger le document localement sur votre appareil. Donc `uri_path` que vous utiliserez doit être un uri de l'emplacement local de ce document (votre `uri_path` commencera par `file://` et non pas par `http://` ou `https://`).

Le parseur SAX est plus rapide que le parseur DOM, il ne construit pas d'arbre de fichier xml, mais l'utilisation du parseur SAX est un peu plus délicate.

2.2 Les informations à extraire de fichier rss

Ce qui est pertinent pour ce projet ce sont les balises `item` du document rss xml. Chaque `item` correspond à un item d'information à afficher. Nous nous intéressons à trois éléments (balises) de chaque item :

1. `title` - le titre,
2. `link` - l'URL de l'item,
3. `description` - la description de l'item.

Initialement juste la liste de titres sera affichée. En sélectionnant le titre l'utilisateur pourra voir la description. Et finalement, grâce à l'URL dans `link`, à partir de la description il pourra voir s'il le souhaite le document l'information complète dans un `WebView`. (Pour le fil rss de journaux voir l'information complète peut s'avérer impossible, le site web est souvent payant, il faudra essayer avec d'autres sites.)

2.3 Bases de données

Votre projet devra gérer une base de données. La base de données doit stocker les adresses http/https de fils rss qui vous intéressent. Grâce à cette base l'utilisateur pourra facilement choisir le fil à charger sans qu'il soit obligé de taper chaque fois l'adresse http.

La base de données permettra aussi de stocker les fils rss où plutôt stocker une partie de contenu de ces fichiers : le contenu de balises `title`, `link`, `description` de chaque item, peut-être aussi la date de la dernière modification.
L'exemple typique d'utilisation de la BD : l'utilisateur pourra charger les fils rss chez lui où la connexion internet est bonne pour ensuite les lire grâce à la BDD pendant le trajet dans le métro.
Il faut prévoir le mécanisme de nettoyage de la base de données pour ne pas garder des anciennes fils. Un mécanisme de nettoyage automatique utilisant par exemple les dates serait plus utile que la méthode où l'utilisateur devra péniblement choisir un à un les fils rss à supprimer. Mais la suppression manuelle a aussi sa place. D'autre côté il peut être utile de garder à l'infini certains items particulièrement intéressants pour l'utilisateur.
La base de données peut servir aussi pour faire les recherche dans les textes de fils rss, par exemple l'utilisateur peut souhaiter chercher dans la base de données tous les items dont la description ou le titre contiennent un String particulier.
La base de données devra être implémentée avec un **ContentProvider** et les requêtes SELECT (query) seront exécutées dans un thread séparé grâce au `Loader`.

2.4 Nettoyage de fichier

C'est un petit point mais important. Quand l'utilisateur charge un fichier rss xml ce fichier est stocké sur son appareil. Mais quand le fichier n'est plus utile pour le lecteur `mplrss` il faut le supprimer et cette action doit être automatique, sans intervention de l'utilisateur (on ne va pas demander à l'utilisateur de regarder le répertoire de fichiers sur son appareil pour supprimer ces fichiers à la main !).

2.5 Améliorations facultatives

L'utilisation de Fragments n'est pas exigée mais l'application implémentée avec les Fragments et une seule activité est plus facile à adapter sur les appareils différents. L'affichage des items (tels que titres des news) dans des RecyclerView donnerait plus de souplesse et de stabilité à vos listes. Donc ce seraient des plus pour votre projet.