Cégep Ste-Foy – Hiver 2019
Programmation Mobile II – 420-V60-SF

# Énoncé du Travail Pratique 1

14 janvier 2019

Préparé par
Benjamin Lemelin

# 1 Résumé

En guise de révision, créez une application permettant de consulter la météo. La météo doit être obtenue à partir du serveur en ligne fourni.

# 2 Conditions de réalisation

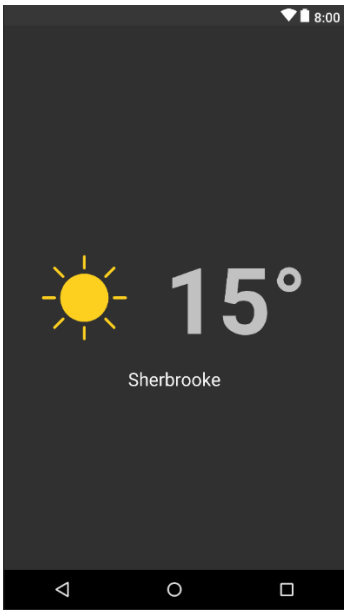
| Valeur de la note finale | Type | Durée | Nombre de remises |
|---|---|---|---|
| 20 % | En équipe de 2 | 4 jours | 1 |

# 3 Spécifications

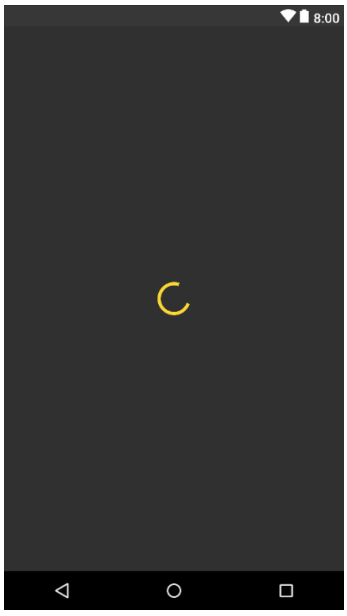
## 3.1 Projet Android Studio

Importez le projet fourni avec cet énoncé. Placez-le dans un dépôt Git (GitHub, GitLab ou BitBucket). Donnez en accès à votre professeur. Tout le code doit être en Kotlin.

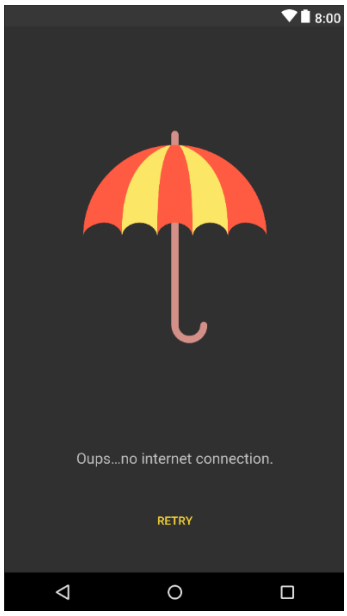
## 3.2 Interface utilisateur




*Météo*




*Chargement*




*Erreurs*

Il ne doit y avoir qu'une seule activité : affichez/cachez les éléments en fonction du contexte. Modifiez images et textes selon les réponses du serveur (type de température, ville, erreur de connectivité, erreur de serveur, …). Le thème de l'application (couleurs, images) peut être modifié à votre guise.

### 3.3 Service Web (Météo)

L'application doit utiliser le faux service météorologique à cette adresse.

https://m2t1.csfpwmjv.tk/api/v1/weather

L'appel à ce service produit une réponse tel que :

```
{
  "type": "SUNNY",
  "temperatureInCelsius": 25,
  "city": "Longueuil"
}
```

Où :

| Champ | Type | Nombre de remises |
|---|---|---|
| `type` | Enum | Soit `SUNNY`, `PARTLY_SUNNY`, `CLOUDY`, `RAIN` ou `SNOW`. |
| `temperatureInCelsius` | Integer | Valeur de `-30` à `30`. |
| `city` | String | Nom d'une ville. |

Pour plus de détails, consultez la documentation : https://m2t1.csfpwmjv.tk/api/doc.

### 3.4 Fonctionnalités

| Description |
|---|
| Au lancement, l'application doit obtenir la météo actuelle à partir du serveur. |
| En cas d'erreur, l'usager doit pouvoir réessayer via le bouton prévu à cet effet. |
| En cas d'erreur, l'usager doit être mis au courant du type d'erreur (connectivité ou serveur). |
| Le nom de la ville doit être affiché. |
| La température actuelle (en °C) doit être affichée. |
| Une image représentant le type de température (nuageux, ensoleillé, …) doit être affichée. |
| L'interface de l'application doit supporter l'anglais et le français. |

### 3.5 Autres spécifications

| Description |
|---|
| L'application doit afficher une barre de progression indéterminée durant tout appel réseau. |
| L'interface de l'application doit supporter l'orientation portrait et paysage. |

## 4 Modalités de remise

Remettre sur LÉA un fichier texte incluant :

1. L'adresse vers le dépôt Git
2. Les matricules de chaque membre de l'équipe

Une seule équivaut à une pénalité de 15 %. Au-delà de ce délai, le travail la note « 0 » est attribuée.

# 5 Évaluation

| Éléments |
| --- |
| Fonctionnalités (incluant, mais sans s'y limiter) :<br>• Affiche la météo en provenance du serveur.<br>• Affiche la température en °C.<br>• Affiche le type de température via une image.<br>• Affiche le nom de la ville.<br>• En cas d'erreur, affiche le type d'erreur.<br>• En cas d'erreur, il est possible de réessayer. |
| Interface utilisateur (incluant, mais sans s'y limiter) :<br>• Fichiers XML d'interface propres.<br>• Visuel de l'interface utilisateur propre et stable. Interface disponible en anglais et en français. |
| Service Web (incluant, mais sans s'y limiter) :<br>• Obtention des données de l'application à partir du serveur fourni.<br>• Gestion des erreurs de réseau et d'obtention des données.<br>• Désérialisation des données effectuée selon les standards. |
| Qualité générale de l'application mobile (incluant, mais sans s'y limiter) :<br>• Traitements lourds toujours effectués en tâche de fond.<br>• Rétroaction visuelle au sujet des traitements en tâche de fond.<br>• Changement d'orientation et d'application supporté, sans perte de données. |
| Qualité générale du code (incluant, mais sans s'y limiter) :<br>• Logique bien pensée, juste, et « non patché ».<br>• Découpage adéquat du code en classes, méthodes et packages.<br>• Architecture adéquate au type d'application développée.<br>• Séparation raisonnable des responsabilités entre les classes.<br>• Respects des standards du langage de programmation.<br>• Nommage clair et sans ambiguïté des éléments.<br>• Utilisation de constantes, lorsque nécessaires.<br>• Commentaires compensant uniquement le manque d'expressivité du code.<br>• Respect des bonnes pratiques de programmation générales.<br>• Code propre, sans résidus et facilement lisible.<br>• Aucune erreur ni avertissement à la compilation. |
| Qualité générale du travail (incluant, mais sans s'y limiter) :<br>• Configuration Gradle fonctionnelle.<br>• Respect des consignes de remise.<br>• Aucun fichier inutile remis avec le projet. |

La qualité de la langue française fait partie de l'évaluation. Chaque faute de français retire 0,5 % à la note finale jusqu'à concurrence de 20 %.