CÉGEP STE-FOY – HIVER 2019
PROGRAMMATION MOBILE II – 420-V60-SF

Énoncé du Travail Pratique 2

18 janvier 2019

Préparé par
Benjamin Lemelin

1 Résumé

Créez une application permettant de répondre à des questions de type « Would you rather … ». Les questions doivent être obtenues à partir du serveur en ligne fourni.

2 Conditions de réalisation

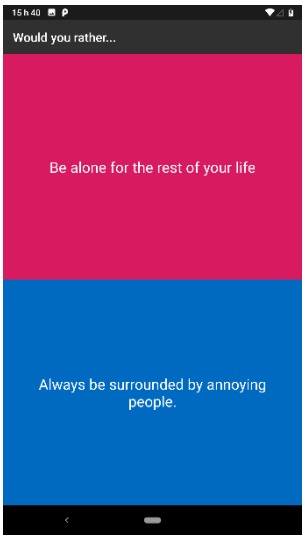
Valeur de la note finale	Type	Durée	Nombre de remises
20 %	En équipe	2 semaines	1

3 Spécifications

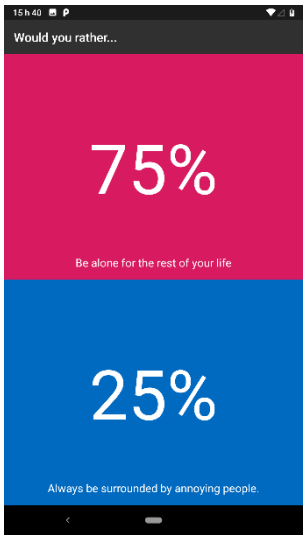
3.1 Projet Android Studio

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.

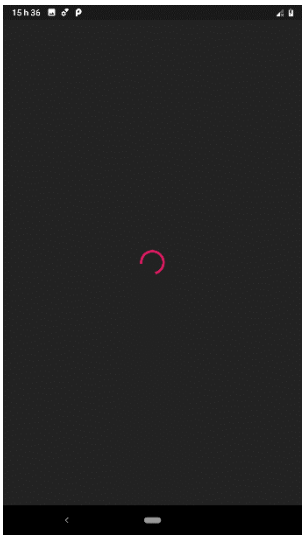
3.2 Interface utilisateur



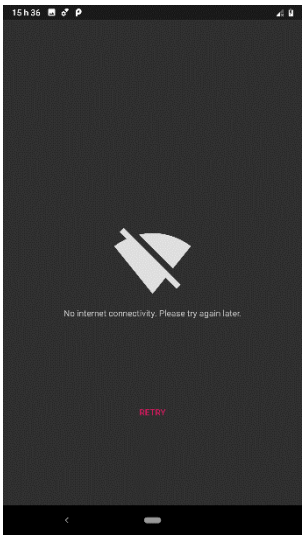
Question



Statistiques



Chargement



Erreurs

L'interface est fournie. Il ne doit y avoir qu'une seule activité : affichez/cachez les éléments en fonction du contexte. Modifiez le texte selon les réponses du serveur (texte de la question, choix de réponse, répartition des choix, erreur de connectivité, erreur de serveur, …). Le thème de l'application (couleurs, images) peut être modifié à votre guise.

Un utilisateur effectue un choix en cliquant sur la section rouge ou bleue. Une fois son choix effectué, la répartition des réponses est montrée. L'utilisateur peut cliquer n'importe où pour passer à la question suivante.

3.3 Service Web (Questions)

L'application doit utiliser les services disponibles à cette adresse.

https://m2t2.csfpwmjv.tk/api/question

Plusieurs services sont offerts, tel qu'obtenir ou créer une question. Par exemple, l'appel à https://m2t2.csfpwmjv.tk/api/question/random produit une réponse tel que :

```
{
  "id": "912900ee-dc0e-475d-aee0-4f8559201d01",
  "text": "Would you rather...",
  "choice1": "Be alone for the rest of your life",
  "choice2": "Always be surrounded by annoying people.",
  "nbChoice1": 3,
  "nbChoice2": 1
}
```

Où :

Champ	Type	Nombre de remises
`id`	UUID	Identifiant unique de la question. Dans votre code, il est plus facilement manipulable sous forme de String.
`text`	String	Texte de la question. Très similaire d'une question à une autre, mais peut varier légèrement en fonction du contexte.
`choice1`	String	Premier choix.
`choice2`	String	Second choix.
`nbChoice1`	Entier	Nombre de fois où le choix 1 a été sélectionné.
`nbChoice2`	Entier	Nombre de fois où le choix 2 a été sélectionné.

Vous aurez besoin d'utiliser d'autres services pour ce travail pratique, tel que confirmer un choix de réponse. Consultez la documentation pour les détails : https://m2t2.csfpwmjv.tk/api/doc. N'hésitez pas à créer des questions pour les besoins de ce travail, même s'il y en a déjà.

3.4 Fonctionnalités

Description
Au lancement, l'application doit obtenir une question aléatoire.
En cas d'erreur, l'usager doit pouvoir réessayer via le bouton prévu à cet effet. Dans tous les cas, ce bouton tente d'obtenir une nouvelle question, et non pas de resoumettre une réponse.
En cas d'erreur, l'usager doit être mis au courant du type d'erreur (connectivité ou serveur).
La question doit être affichée dans la barre de titre de l'application.
Le premier choix doit être affiché en haut, dans la section rouge.
Le second choix doit être affiché en bas, dans la section bleue.
Si l'utilisateur sélectionne le choix 1, ce dernier doit être soumis au serveur.
Si l'utilisateur sélectionne le choix 2, ce dernier doit être soumis au serveur.
Lorsqu'un choix est soumis au serveur, la répartition des choix doit être montrée.
Le pourcentage d'usagers ayant fait le premier choix doit être affiché en haut, dans la section rouge.

Le pourcentage d'usagers ayant fait le second choix doit être affiché en bas, dans la section bleue.
L'usager peut passer à la question suivante en cliquant n'importe où dans l'interface (sauf la barre de titre). Une nouvelle question aléatoire est alors obtenue.
L'interface de l'application doit supporter l'anglais et le français, même si les questions sont toutes en anglais.

3.5 Spécifications techniques

Description
L'usage de « Android Annotations » est obligatoire pour : • La sélection du layout d'une activité. • La gestion des clics dans l'interface. • La création de tâches de fond et l'appels de fonctions sur le thread principal.
L'usage de « Retrofit » est obligatoire pour : • L'obtention de questions à partir du serveur. • L'envoi de réponses au serveur.
La (dé)sérialisation Json peut se faire avec Jackson tel que montrée en classe. Cependant, toute librairie de (dé)sérialisation Json est aussi acceptée, tel que Gson.
L'usage de « Parceler » est obligatoire pour la création d'objets parcelables. Parmi ces objets, votre « ViewModel » et votre « Model » devraient être parcelables.
L'usage d'une architecture « MVVM » est obligatoire. Aussi, en lien avec cette architecture, l'usage du « DataBinding » Android est obligatoire pour relier les données de l'application à l'interface.

3.6 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) : • Affiche des questions aléatoires en provenance du serveur. • Choisir une réponse l'envoie au serveur et montre la répartition des choix. • La répartition des choix est affichée sous la forme d'un pourcentage. • Cliquer n'importe où durant l'affichage des statistiques permet de passer à la question aléatoire suivante. • En cas d'erreur, affiche le type d'erreur. • En cas d'erreur, il est possible de réessayer et ainsi de répondre à nouvelle question aléatoire.
Interface utilisateur (incluant, mais sans s'y limiter) : • Fichiers XML d'interface propres. • Visuel de l'interface utilisateur propre et stable. Interface disponible en anglais et en français.
Service Web (incluant, mais sans s'y limiter) : • Obtention des données de l'application à partir du serveur fourni. • Gestion des erreurs de réseau et d'obtention des données. • Désérialisation des données effectuée selon les standards.
Métaprogrammation (incluant, mais sans s'y limiter) : • Usage correct d'Android Annotation dans tous les cas concernés. • Usage correct de Retrofit dans tous les cas concernés. • Usage correct de Parceler dans tous les cas concernés. • Usage correct du DataBinding dans tous les cas concernés.
Style architectural MVVM (incluant, mais sans s'y limiter) : • L'interface est complètement gérée par l'architecture MVVM et le DataBinding. • Bindings dans les fichiers XML se font sur un objet de type ViewModel. • Les ViewModels de l'application font le pont entre le modèle et l'outil de DataBinding. • Les ViewModels sont autonomes (ils notifient par eux même l'outil de DataBinding des changements). • Les ViewModels formatent les données pour l'interface au besoin.
Qualité générale de l'application mobile (incluant, mais sans s'y limiter) : • Traitements lourds toujours effectués en tâche de fond. • Rétroaction visuelle au sujet des traitements en tâche de fond. • Changement d'orientation et d'application supporté, sans perte de données.
Qualité générale du code (incluant, mais sans s'y limiter) : • Logique bien pensée, juste, et « non patché ». • Découpage adéquat du code en classes, méthodes et packages. • Architecture adéquate au type d'application développée. • Séparation raisonnable des responsabilités entre les classes. • Respects des standards du langage de programmation. • Nommage clair et sans ambiguïté des éléments. • Utilisation de constantes, lorsque nécessaires. • Commentaires compensant uniquement le manque d'expressivité du code. • Respect des bonnes pratiques de programmation générales. • Code propre, sans résidus et facilement lisible. • Aucune erreur ni avertissement à la compilation.

Qualité générale du travail (incluant, mais sans s'y limiter) :

- Configuration Gradle fonctionnelle.
- Respect des consignes de remise.
- 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 %.