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

Énoncé du Travail Pratique 2

16 janvier 2020

Préparé par
Benjamin Lemelin

1 Résumé

Créez une application permettant de répondre à des questions de type « Would you rather … ». Les questions sont 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

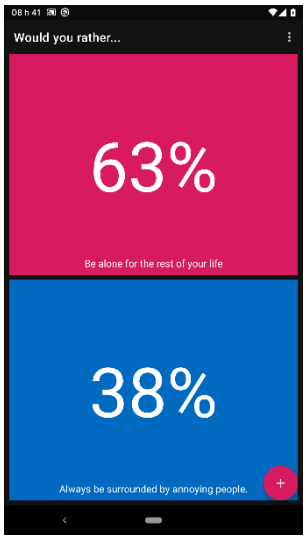
3.1 Projet Android Studio

Importez le projet fourni et placez-le dans un dépôt Git. Donnez en accès à votre professeur.

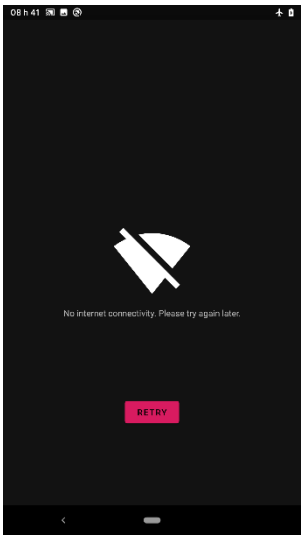
3.2 Interface utilisateur

Les interfaces sont fournies. Le thème de l'application (couleurs, images) peut être modifié à votre guise.

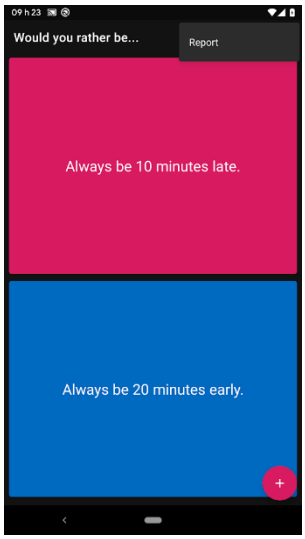
Activité principale



Répartitions



Erreurs

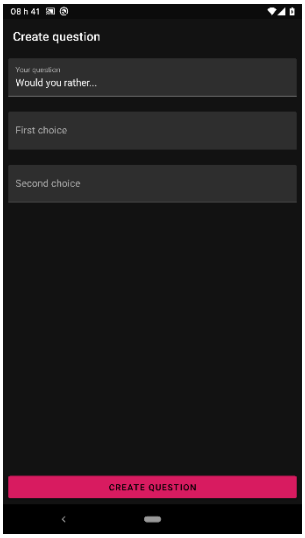


Signalement

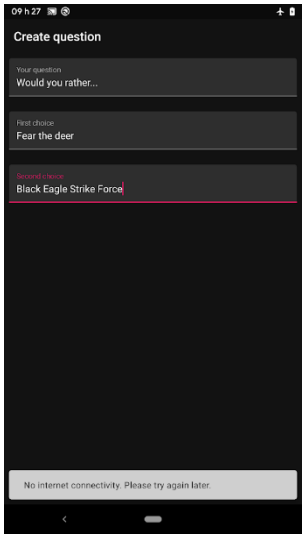
Dans l'activité principale, l'utilisateur effectue un choix en cliquant sur le bouton rouge ou bleu. Son choix effectué, la répartition des réponses est alors montrée. L'utilisateur clique ensuite sur n'importe quelle couleur pour passer à la question suivante. Les questions montrées sont aléatoires.

Il est possible de signaler une question comme inappropriée via un menu dans le coin supérieur droit. Le menu doit faire partie de la barre de titre en haut de l'écran.

Le « FloatingActionButton » en bas de l'écran permet d'accéder à l'activité de création de questions.



Création de questions



Erreur

Réponse initiale

Dans l'activité de création de questions, l'utilisateur peut saisir une question. Le champ « *Your Question* » doit être prérempli, mais l'utilisateur peut le modifier à sa guise.

Tous les champs sont obligatoires. En cas d'erreur de n'importe quelle nature (validation, serveur, connectivité), affichez un « Snackbar » en bas de l'écran.

L'utilisateur doit pouvoir répondre à la question qu'il vient de créer lorsqu'il retourne à l'activité principale. Vous devez donc utiliser « startActivityForResult ».

3.3 Service Web (Questions)

Avec cet énoncé est fourni le fichier « .jar » du serveur de l'application. Vous pouvez l'exécuter via le fichier « .bat » fourni. Une fois démarré, le serveur est accessible sur le port « 8080 ». Consultez la documentation pour les détails : http://localhost:8080/api/doc.

Un fichier JSON vous est fourni avec quelques questions pour démarrer. Importez-le en utilisant l'api fourni. Notez aussi qu'il existe deux versions l'api : choisissez celui qui vous conviendra le mieux.

3.4 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. • Injecter l'outil de « DataBinding » dans une activité. • Créer et injecter une « Bean » dans une activité. • Créer une « Bean » qui est un « Singleton ». • Injecter une « View » dans une activité. • Utiliser une autre activité avec « startActivityForResult ».

L'usage de « Retrofit » est obligatoire pour : • L'obtention de questions à partir du serveur. • L'envoi de réponses au serveur. • Ainsi que tout autre appel réseau nécessaire à l'application.
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.

3.5 Autres spécifications

Description
Le code doit être en Kotlin.
L'interface de l'application doit supporter l'orientation portrait et paysage.
L'interface de l'application doit supporter l'anglais et le français.
L'application doit afficher une barre de progression indéterminée durant tout travail long.
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.
La question doit être affichée dans la barre de titre de l'application.
Lorsqu'une question est signalée, une nouvelle question doit être affichée.

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. • Il est possible de signaler une question. • 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. • Permets de créer une nouvelle question. • Tous les champs du formulaire de création de questions sont obligatoires. • Il est possible de répondre à une question après l'avoir créé.
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 ambigüité 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 %.