IFT3150: Projet supervisé

Développeur Android MONA

Nom du Superviseur: Guy Lapalme
Nom: Philippe Auclair
Matricule: p0991331

J’ai fait quelques projets à mon compte, plusieurs que j’utilise personnellement, et d’autres dans un cadre académique. Mais je n’ai jamais fait un travail avec un grand groupe. Plusieurs projets n’ont jamais été finis, ou du moins, ils ne répondraient pas aux standards s’ils devraient être publiés. C'est pourquoi avant même de m’inscrire officiellement avec l’université pour le projet, je me suis joint à l’équipe MONA. Mon but était d’obtenir de l’expérience dans le travail d’équipe et dans la production d’un projet qui sera utilisé par le public. Je crois aussi qu’il serait bon que je m’habitue à travailler avec des échéanciers fixes et à pouvoir m’adapter aux demandes d’autres personnes.

Je crois qu’il est important de mentionner que mon entrée dans le projet s’est faite dans un contexte particulier. La pandémie a fait en sorte que j’ai fait l’entièreté du projet chez moi. Auparavant, l’équipe MONA organisait des visites de groupes dans des lieux publics. Ils en profitaient pour faire tester l’application et obtenir l’avis d’utilisateurs. N’ayant pas eu accès à ce genre de données, j’ai dû me fier aux simulations de téléphone intelligent sur Android Studio pour les tests, et à mon instinct et à celui de Lena pour les décisions pratiques liées aux fonctionnalités et au désign de l’application. Je réalise donc qu’il peut y avoir des incongruités entre mon expérience sur le projet et celle que l’on aurait normalement.

.

.1- Présentation du projet

MONA est une application mobile gratuite qui invite à la découverte de l'art et des lieux culturels au Québec. Elle permet à l'utilisateur de "collectionner" des oeuvres et des lieux. En parcourant des listes, ou en regardant la carte, ils peuvent découvrir de nouvelles œuvres qu'ils devront physiquement rejoindre pour la prendre en photo, émettre un commentaire, et lui donner une note. Tous ces éléments sont ensuite enregistrés dans la collection, et peuvent être accédés à tout moment.

Il s’agit d’un projet créé initialement par Lena Krause, étudiante en maîtrise en histoire de l’art, qui a été mon intermédiaire tout au long de mon projet. J’ai aussi travaillé avec Mathieu Perron, responsable de la programmation coté serveur.

2- Objectifs

Je me suis joint à l'équipe en tant que développeur Android. Ma tâche est de gérer les éléments visuels et du code pour l'application sur les téléphones Android. Lorsque j'ai rejoint l'équipe, l'application était en phase "Alpha", ou du moins en début de la "Beta". Le squelette du programme était présent, l'utilisateur pouvait prendre des photos et elles étaient enregistrées dans le système. Par contre, il y avait plusieurs problèmes évidents:

- L'application ne pouvait pas être utilisée sans une connexion internet(ce qui est problématique pour une application qui est basée sur le déplacement)
- Il y a des problèmes dans la navigation sur la carte lorsque l'on recherche une œuvre
- Les données n'étaient pas enregistrées dans le serveur.

- Il y a un problème de performance, qui cause un temps de chargement long et pénible à chaque fois que l'application est lancée.
- D'un point de vue du code, il y avait plusieurs incongruités. Une grande partie du code a été écrit durant des Hackotons (des compétitions informatiques) qui imposent une contrainte de temps. Il y donc des raccourcis qui ont été pris, ce qui se révèle surtout par de la duplication de code.
- Il y a aussi des mentions de Badges, qui doivent, en théorie, récompenser l'utilisateur quand il accomplit des objectifs particuliers. Cette fonctionnalité n'a jamais été implémentée.

Tous ces éléments font en sorte que l'application apparaît incomplète, même pour l'utilisateur moyen.

Mon objectif, dans le cadre de ce projet, était d'amener l'application à un stade proche d'une sortie officielle, comme c'est déjà le cas sur la version IOS(Iphone). De façon pratique, cela signifie que du point de vue de l'utilisateur, toutes les fonctionnalités mentionnées sur le site sont implémentées, la recherche d'oeuvre est simple, il n'y a pas de problème au niveau de l'utilisation, et nous pouvons recevoir les informations du côté serveur. Il fallait aussi changer le code pour le rendre plus intuitif et plus facile à maintenir sur le moyen/long terme, lorsque je passerai le relai à une autre personne.

Pour arriver à ces fins, j’ai voulu compléter ses objectifs précis, que j’ai divisés en deux grandes phases au courant de la session.
Phase 1:

- Éliminer la redondance dans le code
- Ordonner les listes des Œuvres et de lieux en listes
- Enregistrer les données des utilisateurs dans notre serveur

- Créer un mode hors-ligne

Phase 2:

- Créer un tutoriel de première utilisation
- Implémenter les Badges
- Faire des optimisations sur la carte et les listes.

Dans le cadre de ce rapport, je vais présenter ces objectifs en détail. Je présenterai les démarches entreprises pour les accomplir, ainsi que les leçons personnelles que j'ai tirées du processus.

3-Descriptions changement

3.1 - Éliminer la redondance dans le code

Techniquement, la première tâche qui m'a été attribuée, lorsque je me suis joint à l'équipe, était l'implémentation des liens entre l'application et le serveur. Mais lorsque j'ai commencé à étudier le code, j'ai observé qu'il y avait une plusieurs répétitions dans le code, dû au fait que le programme traitait deux types d'objets, les *Oeuvres* et les *Lieux*. Plusieurs fichiers, que ce soit des éléments visuels comme des layouts, du code comme des fragments, ou éléments de la database, étaient presque entièrement dédoublés, pour répondre à des besoins qui étaient sensiblement les mêmes. Après une discussion avec l'équipe, il a été décidé que de régler ce problème deviendrait ma priorité.

Comme nous pouvons le constater, tous les attributs des *Lieux* sont contenus dans les *Oeuvres*. Ceci m'a permis, après des ajustements, de créer un élément une classe unique ayant tous les attributs des *Oeuvres*, qui peuvent contenir les lieux. Pour les différentier, j'ai créé un attribut supplémentaire nommé *type*, qui est un string qui est soit "artwork" ou "place".

La problématique principale vient du fait que je suis dépendant du formatage des données du côté du serveur. De celui-ci, j'obtiens deux listes différentes, une pour chaque type. Je dois donc faire quelques manipulations sur les JSONs pour pouvoir traiter les données, mais cela n'occasionne pas de changement au niveau de la performance. Toutefois, j'ai dû rajouter un autre attribut, que j'ai nommé *idServeur*, car comme les manipulations peuvent occasionner des changements dans l'ordre des listes, il faut que je puisse faire la différence entre sa position dans l'application et dans le serveur.

Ceci permet de créer des fonctions communes, et lorsque le type influence le résultat, nous pouvons faire un test simple sur le type (ex : if(*type* == *artwork*). Pour ce qui est du visuel, alors qu'avant nous avions de *layout* différents pour chacun des types, il s'agit maintenant simplement d'une manipulation sur la couleur. Par exemple nous avions un deux *layouts*, l'un avec un cercle jaune, l'autre un cercle mauve. Maintenant la couleur

du cercle passe de l'un à l'autre de façon conditionnelle. Ceci m'a permis d'éliminer 14 fichiers redondants.

Je crois que c'est une très bonne chose d'avoir commencé par cette partie. De par sa nature, cela m'a forcé à traverser presque toute la structure du projet, ce qui m'a permis d'apprendre son fonctionnement en profondeur. D'une autre part, je considère que mon travail était bien fait, dans la mesure qu'il n'y a pas de différence visible depuis l'application entre cette nouvelle version et l'ancienne. Dans ce cas, je me considère chanceux que Lena ait une bonne base en informatique pour apprécier cette partie du travail, mais je ne sais pas comment cela pourrait se transmettre à un client sur le marché du travail qui ne peut pas voir de différence entre les deux versions.

3.2 - Ordonner les Oeuvres et les Lieux en listes / Optimisation des listes[1]

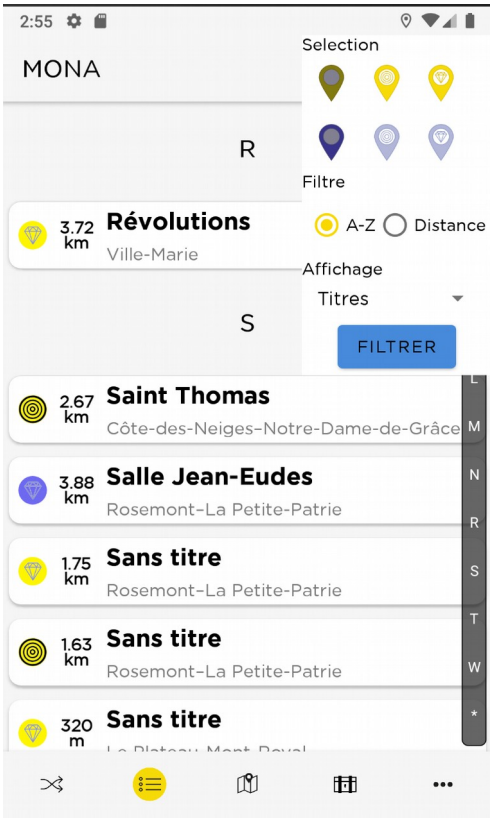
Après avoir fait une esquisse du code pour la connexion avec les serveurs, j'ai réalisé que je devrais travailler directement avec le responsable des serveurs. Mais comme il était dans une période occupée, j'ai dû attendre quelque temps avant de pouvoir commencer. Donc en attendant, je me suis mis sur ce qui devait être originellement la troisième partie du travail, la gestion des listes.

Originellement, l'application avait une seule liste d'items sous la forme d'un *RecyclerView*, qui est une classe qui permet d'afficher plusieurs objets l'un à la suite d'un autre. En accédant au menu au-dessus de la page, il était possible de les ordonner de différentes façons (id, alphabétique, quartier …), ce qui changeait l'ordre de la liste courante. Notre but était de se rapprocher de la version IOS, qui offre l'option de naviguer à travers des sous-menus pour accéder à des listes précises.

[1] PS : Je mets ici les deux parties puisqu'elles sont complémentaires

Si j’ai tiré une leçon de cette expérience, c’est qu’il est très important d’avoir un plan clair des objectifs finaux, et de faire des recherches approfondies sur les techniques utilisées par les professionnels dans le domaine. En effet, lorsque j’ai commencé cette section, il m’a semblé logique d’utiliser ce qui s’appelle des *ExpandableList*, qui sont des listes déroulantes qui apparaissent lorsque l’on appuie sur un bouton. Toutefois, il n’est pas prévu, à la base, de pouvoir créer des listes dans des listes(donc des menus déroulants dans d’autres menus déroulants). J’ai donc dû modifier des librairies que j’ai trouvées en ligne pour pouvoir me rapprocher de cette vision. Cette sensation d’aller à contre-courant des normes de Android aurait dû être le premier signe révélateur que quelque chose n’allait pas. De plus, il y avait le problème que toutes les sous-listes devaient être générées lors de l’initiation de la page, ce qui pouvait causer de problème de performance.

Après avoir fait une implémentation sommaire des listes, j’ai l’ai présentée à Lena. Ceci a mené à une discussion sur les objectifs à long terme, ce que l’on recherchait pour l’utilisateur. Nous en sommes donc venus à la conclusion qu’il serait préférable, au lieu de sous-listes, de créer un système de filtres, qui permettrait à l’utilisateur, avec quelques clics, d’avoir une liste personnalisée.
Ironiquement, cette implémentation est plus proche de ce que l’on avait originellement. J’ai donc rejeté le travail que j’avais fait précédemment, et j’ai recommencé depuis la source. Après quelques recherches, j’ai réalisé que cette façon de faire est la façon recommandée de gérer des listes en Android, une seule liste ou l’on change le contenu.



Lors de la seconde phase, je suis donc revenu implémenter les filtres. Il est maintenant possible de filtrer les listes

selon le statut de l'item (collectionné, ciblé), de changer l'ordonnancement(par distance, alphabétique) ainsi que l'information qui apparaît sur chaque bannière(artistes, catégories ...). J'en ai aussi profité pour implémenter des icônes pour pouvoir déterminer visuellement si une œuvre à été collectionnée ou ciblée, ainsi qu'un *FastScroll*, qui permet de naviguer rapidement sur la liste avec une barre à la gauche. Cette barre est peuplée dynamiquement, ce qui permet de changer son contenu dépendant de l'information. Par exemple, lorsque l'on trie par distance, ce sont des nombres. Il a d'ailleurs fallu aussi que je change la formule pour la distance (la version originelle retournait des chiffres étranges). J'ai opté pour une méthode mathématique basée sur le calcul de longitude et de latitude au lieu d'aller chercher une librairie pour le faire.

Si j'ai appris quelque chose de cette section, c'est qu'il est important d'avoir une idée claire des objectifs avant de commencer l'implémentation, et qu'un «gut feeling» sur les techniques à utiliser ne vaut pas une recherche approfondie. Cela m'aurait sauvé beaucoup de temps, mais je suis content d'en avoir tiré une leçon.

Enregistrer les données des utilisateurs dans notre serveur

Après tout cela, je me suis finalement mis sur mon objectif initial. Nous voulions que les images de l'utilisateur puissent être enregistrées dans nos serveurs, ce qui est l'un des aspects les plus importants de l'application. Nous devons collecter l'*id* de l'item (le *idServeur* mentionné plus tôt), la note, le commentaire et le fichier image.
Heureusement, je ne partais pas de rien, je pouvais me fier à la version IOS, qui permettait déjà de le faire.

L'implémentation elle-même ne fut pas très difficile, mais j'ai eu beaucoup de problème à la faire fonctionner. Après beaucoup d'essais et erreurs, j'ai réalisé que la librairie que

nous utilisions pour les connexions avec les serveurs n'était pas à jour, et après quelques changements, la connexion elle-même se faisait correctement. Présentement, cet appel au serveur se fait après que l'utilisateur ait donné une note et un commentaire à un item. Ce qui m'a pris plus de temps fut de corriger les erreurs liées aux images. Le serveur n'acceptait pas les fichiers que je lui envoyais. J'ai réalisé que c'était dû au fait que Android sauve les images de la caméra avec un Uri, ce qui fait en sorte qu'elles ne sont pas reconnues comme un fichier jpg. Je dois donc créer un fichier jpg temporaire, copier les données de l'image en Uri dans celui-ci, pour ensuite l'envoyer au serveur. Ce semble être la version standard de faire les choses, je n'ai pas trouvé une autre façon de le faire dans tous les cas.

Après avoir complété l'implémentation, et avoir eu des tests concluants depuis chez moi, je suis allé faire un test pratique sur le terrain. Ceci m'a mené à poser quelques questions qui ne m'étaient pas venues à l'esprit auparavant. Par exemple, que faire si nous n'avons pas de connexion internet? Que se passe-t-il si une personne ne met pas de notes ou de commentaire? J'aborderai la première question dans la prochaine section sur le mode hors-ligne. Pour ce qui est de la seconde, la question m'est parvenue lorsque j'ai visité un endroit qui possédait des dizaines de pièces d'art l'une à côté de l'autre. Puisque j'en profitais pour promener mes chiens, je ne pouvais pas passer trop de temps à émettre des commentaires et penser à une note (ils ne supportent pas de rester immobiles trop longtemps). Alors vers la fin, je prenais simplement les photos et je passais à la prochaine. Mais cela mène à un questionnement sur la nature fondamentale de l'application. Après tout, nous voulons encourager les personnes à émettre leur opinion, pas simplement prendre une photo sans trop réfléchir. Mais nous ne pouvons pas non plus rejeter le contexte réel, comme le mien, où ce n'est pas toujours possible. Il y a aussi le fait que, dans la version que l'on avait, nous demandons une note sur 5 étoiles, et si l'utilisateur ne met rien, le programme le compte comme un zéro, ce qui affecte les

moyennes. Alors nous en sommes venus à un compromis, où il est possible de ne pas mettre de note ou de commentaire, mais l'utilisateur est encouragé de le faire par la suite. Il reste que c'est dans des cas comme celui-ci que le contexte de 2020 rend les choses compliquées, car je peux m'imaginer qu'il y a plusieurs petites problématiques comme celle-ci qui ressortiraient des essais sur le terrain.

Grâce au changement fait avec la jonction des œuvres et des lieux, il n'y a pas de différence notable lorsqu'on les envoie dans le serveur, ce qui m'a permis d'implémenter les deux en même temps. Ceci nous met donc en avance sur la version IOS, qui ne fait que les œuvres.

Créer un mode hors-ligne

Lors de sa création, l'application pouvait être utilisée sans connexion internet . Mais à un certain moment durant le développement, un changement a été apporté faisant en sorte que le programme plante complètement en cas de non connexion. J'ai donc ajouté une option dans le menu, qui offre la possibilité d'aller en mode hors-ligne, ou de retourner en ligne, s'il y a une connexion internet valide.

Pour ce faire, j'ai ajouté une variable globale dans les SavedSharePreferences (un fichier dans le téléphone où l'on peut enregistrer des données entre les sessions) qui détermine dans quel mode on se trouve. Par la suite, j'ai créé une fonction qui permet de détecter si on a une connexion valide. De là, il s'agissait simplement de rajouter des conditions aux endroits qui nécessiteraient possiblement de se connecter au serveur.

Un des problèmes majeurs de l'application était le temps de chargement. À l'ouverture, cela prenait près de 15 secondes pour pouvoir commencer à l'utiliser, ce qui est

évidemment beaucoup trop long. Ce délai provient presque entièrement du fait qu'à chaque ouverture, la database locale est repeuplée à partir de l'information des serveurs. Si l'on passe cette étape, ce qui est le cas si on est hors ligne, ce délai passe à environ 3 secondes. D'un point de vue de l'utilisateur, ce simple changement serait suffisant pour ne jamais vouloir être en ligne. Grâce au conseil de Monsieur Lapalme, nous avons conclu que l'idéal serait d'implémenter un système de datation, où l'on irait chercher que les œuvres qui n'avaient été pas été présentées lors de la dernière connexion.

Nous avons donc rajouté un nouvel attribut aux items du côté serveur, qui contient la date de la dernière modification. Du côté de l'application, j'ai rajouté une variable dans les SavedSharePreferences qui contient la dernière fois que nous sommes allés chercher les éléments dans la database. Maintenant, lorsque l'on va chercher la liste, le serveur nous retourne seulement les éléments dont la date de modification est plus récente que la date dans l'application. Ceci réduit drastiquement le temps de chargement au début de l'application.

Il y a aussi la question de la gestion des activités avec les serveurs lorsque nous sommes hors-ligne. Par exemple, que ce passe-t-il si quelqu'un est en mode hors-ligne, et qu'il prend une photo? Pour régler ce problème, j'ai implanté un nouveau *state* pour les items. Avant, nous avions l'état *null*, qui signifie qu'il n'y a pas eu d'action sur cet item, 1 signifie que l'item est ciblé, et 2 qu'il est collectionné. J'ai donc rajouté l'état 3, collectionné, mais pas dans le serveur. Lorsque la personne retourne en ligne, l'application va chercher toutes les œuvres et les lieux dont le *state* est égal à 3, et les envois dans le serveur. Leurs états sont ensuite mis à 2.

La gestion de l’information avec le serveur et hors ligne est donc complétée. Il va falloir rajouter des éléments visuels pour indiquer à l’utilisateur que le transfert est en cours. Possiblement une barre de progrès ou un message qui indique ce qui se passe.

Ces deux dernières parties du travail m’ont aidé à me familiariser avec la gestion de l’information et la communication avec des serveurs, quelque chose que je n’avais jamais eu à appliquer dans un contexte pratique avant. Cela va m’être très pratique, surtout que je me dirige vers de la création web dans le futur.

Créer un tutoriel de première utilisation

Considérant les changements apportés, et parce qu’il s’agit d’une bonne pratique de façon générale, nous avons voulu implémenter un tutoriel de première utilisation. J’ai observé que mes capacités à naviguer à travers le projet et mes connaissances en Android se sont grandement améliorées depuis le début du projet. Avec l’aide de guides trouvés en ligne, le travail s’est fait sans grandes embûches. Pour me rapprocher des standards actuels, j’ai opté pour un *ScreenSlider*, qui permet de *swipe* pour passer d’une page à une autre, qui offre une animation de transition fluide et satisfaisante.

Originellement, dans la version que j’ai trouvée en ligne, il y avait un adaptateur et un layout pour chaque page du tutoriel. J’ai amélioré le processus pour qu’il y ait un seul adaptateur. Il me serait possible de faire la même chose pour les layouts, mais si l’on veut rajouter des animations ou des éléments différents entre chaque page dans le futur, il sera plus facile de le faire si je le laisse comme cela.

Pour l’instant, le tutoriel apparaît seulement lors de la première utilisation, et peut être accédé de nouveau à partir des options. L’on pourra possiblement créer des tutoriels

multiples pour chaque page dans le futur, si l’on trouve qu’il y a trop d’informations à digérer pour l’utilisateur en ce moment.

Implémenter les Badges

Ce qui devait être mon premier grand apport à l’application s’est révélé être relégué à plus tard. Je devais me fier aux images de l’équipe de design pour implémenter la nouvelle interface des badges. Mais nous avons réalisé que les autres éléments étaient prioritaires. De plus, encore après une autre discussion, nous avons réalisé que nous pourrions fondamentalement changer la façon dont sont implémentés les badges. Présentement, les conditions pour l’obtention d’un badge sont figées dans l’application elle-même, ce qui peut être problématique, surtout si l’on veut que ce soit consistant entre les plateformes. J’ai donc proposé que l’on puisse générer les badges et les conditions à partir du serveur, pour être par la suite généré par l’application. Il faudra donc attendre d’avoir le temps, autant de mon côté que celui du serveur, pour coordonner cette méthode. C’est pourquoi les badges n’auront pas été implémentés durant cette session.

Conclusion

Je peux affirmer que la plupart des objectifs initiaux ont été complétés. Les Oeuvres et les Lieux sont implémentés également, les données sont enregistrées dans le serveur, les listes sont plus conviviales, offrent plus d'options et la structure du code rendra la tâche beaucoup plus facile dans le futur pour moi-même et ceux qui me suivront dans le développement Android. Alors qu'initialement, la version Android était en retard sur la version IOS, elle est maintenant bien en avance. Il reste beaucoup à implémenter, mais nous avons assez fait durant cette session pour créer une nouvelle version de l'application que nous mettrons sur le Google Store. Je crois que les utilisateurs seront bien satisfaits des changements apportés.

Dans l'ensemble, je crois que de participer à ce projet fut l'expérience parfaite que je souhaitais. Elle m'a permis de me familiariser avec Android, d'évoluer dans un travail d'équipe, de me forcer à respecter des échéanciers et d'en tirer plusieurs leçons provenant directement de la pratique. Ce sera un bénéfice net dans mes projets futurs dans le domaine. Je suis aussi heureux que mon apport au projet ne s'arrête pas ici, puisque l'on m'a accordé un contrat pour la prochaine session.