****

**Acclimate : rapport final**

**par**

**Jérémi Grenier-Berthiaume**

**Olivier Lepage-Applin**

**Sophie Savoie**

**Chapeauté par Mr Houari Sahraoui**

**Dans le cadre du cours**

**Projet d’informatique – IFT 3150**

**Vendredi 14 septembre 2018**

**Détails sur le problème :**

L’idée du projet est partie de l’observation de Sophie par rapport au manquement du gouvernement par rapport à une utilisation facilitée de [certaines données fournies au grand public](https://www.donneesquebec.ca/recherche/fr/dataset/carte-vigilance-multirisque-fil). En fait, plus précisément, on remarque que le contenu de [l’onglet « Applications »](https://www.donneesquebec.ca/recherche/fr/dataset/carte-vigilance-multirisque-fil/related) indique qu’ « aucune application n’a encore été développée avec ce jeu de données. »

En soit, en analysant les alertes qui étaient publiées dans leur flux RSS, on a pu remarquer que le gouvernement s’attendait à ce que les gens envoient un « gazouillis » (pour réutiliser leur mot) afin de les notifier de quelconque changement climatique en lien avec certaines de ces alertes. Il était donc claire qu’**une application Android manquait cruellement à l’outillage** entourant l’initiative du SNAP ([Système National d’Alertes au Public](https://www.publicsafety.gc.ca/cnt/mrgnc-mngmnt/mrgnc-prprdnss/ntnl-pblc-lrtng-sstm-fr.aspx?wbdisable=true)). L’image présentée au sein de ce dernier lien est un autre témoin de l’oublie quant à l’importance de faciliter une communication dans les deux sens : il faudrait aussi penser à faciliter la signalisation d’alertes au gouvernement par les citoyens.

**Répartition des tâches :**

Dès la phase d’analyse, nous avons commencé le processus de répartition des tâches. Dans le cadre de ce cours universitaire, je ne vais préciser que la répartition qui concerne la période suivant le Hackathon (HackQc 2018 – [Gagnant du 3e prix](https://devpost.com/software/hackqc-remise)) puisque la majorité du travail y a été fait :

|  |  |
| --- | --- |
| **Nom** | **Assignation principale** |
| Jérémi Grenier-Berthiaume | Application Android, WebApp |
| Olivier Lepage-Applin | Application Android |
| Sophie Savoie | Base de données MySQL, REST API |

Olivier a beaucoup plus eu à collaborer avec Sophie que moi puisqu’il s’occupait d’implémenter l’architecture des requêtes de l’application Android à l’API. Il s’occupait aussi plus généralement de ce qui faisait appel à *OSMDroid*, à l’affichage des alertes récoltées en temps réel, à la publication de nouvelles alertes par les usagers (ainsi que le système de « karma » associé) et à l’architecture entourant la sauvegarde en mémoire interne d’objet Java autres que primitifs.

**Glossaire :**

Afin de clarifier certains concepts, j’ai cru pertinent d’offrir un glossaire à mon équipe pour ce qui touchait ma partie du développement.

* **Firebase**: Ensemble de services offerts par Google pour les développeurs.
* **Admin SDK**: Permet l'accès à certaines fonctionnalités d'un API de Firebase (modifier le mot-de-passe d'un usager de la base de données d'authentification, par exemple). Authentifie l'application faisant les requêtes. L'intégration de l'Admin SDK doit se faire « en privé » (elle nécessite d’insérer dans le code de l'information sensible que le public ne doit pas connaître).
* **FCM**: Firebase Cloud Messaging. Service facilitant l'envoie de Notifications à des appareils mobiles.
* **registrationToken**: Chaque appareil mobile unique ayant téléchargé l'application reçoit un 'registrationToken' unique. C'est ce String que l'on doit utiliser pour envoyer des notifications avec FCM. (Sous certaines conditions très particulières, celui-ci peut changer.)
* **OAuth 2**: Méthode d'authentification qui permet à un usager de s'authentifier sans nécessairement envoyer ses données sensibles (tel que son mot-de-passe) au serveur auquel il fait une requête.
* **idToken**: C'est un token utilisé par le protocole OAuth 2 de Firebase pour permettre d’authentifier la requête d'un usager (la requête est envoyée avec ce token).
* **JWT**: JSON Web Token. C'est un JSON encodé en base 64. Il possède la forme "XXX.YYY.ZZZ", où XXX est le '*header*', YYY le '*payload*' et ZZZ la '*signature*'. Dans le cas de l'authentification via Firebase (où l'idToken est un JWT), le 'header' contient une clé publique de Google ainsi que l'algo utilisé pour la signature (soit RSA 256 bits, ou "RS256"), le 'payload' contient les informations en lien avec l'usager qui fait la requête (email, UID, etc.), et la 'signature' permet d'authentifier la requête.
* **UID**: Chaîne de caractères unique assignée de manière permanente à tout utilisateur s'authentifiant via Firebase à une application.

Afin de faciliter la communication au sein de ce rapport, voici aussi un glossaire des termes que nous utilisions entre nous pendant le développement du projet :

* **Token** : une String utilisée dans le fonctionnement de certaines composantes.
* **Zone Surveillée** : principalement dénommé « Monitored Zone » dans le code.
* **Bob** : blabla

**Description des accomplissements :**

[**Site web du projet**](https://www.webdepot.umontreal.ca/Usagers/p1071435/MonDepotPublic/Acclimate/index.xhtml) **(pas la WebApp) :**

J’ai fourni le patron pour les pages personnelles qui concernent le « rapport de progrès bihebdomadaire. »

Le code HTML et CSS du site provient de notre (Olivier et moi) TP1 remis dans le cadre du cours d’*Introduction au Design Web*. Le site possède un design dynamique, c’est-à-dire qu’il s’adapte à l’affichage pour quelconque support d’accès (ordinateur, tablette, téléphone mobile, etc.).

**Serveur (REST API):**

Quatre contributions directes.

* Intégration du système d’authentification du serveur
* *Parsing* des alertes de la base de données historiques du gouvernement afin de les transférer dans notre propre base de données (ce qui nous permet alors de bonifier celle-ci avec les informations que nous récoltons via l’application).
* Déblocage d’un problème de *Thread* qui rendait la requête de « POST Zone Surveillée » lente et dépendante de la grosseur de l’aire couverte (on est alors passé d’un temps d’attente parfois supérieur à 30 secondes à un temps relativement fixe de 1 seconde).

Deux propositions de **structures de données** qui ont été implémentées.

* *HashMap* des alertes pour effectuer une mise-à-jour en temps réel plus efficace des alertes et faciliter le traitement des notifications à envoyer.
* Système d’indexage (qu’on appelle « tuiles ») des zones surveillées qui permet de trouver plus rapidement à quel usager on doit envoyer une notification pour une nouvelle alerte donnée. Plutôt similaire au concept de « filtre de Bloom. »

Plus généralement, en ce qui concerne le serveur, j’ai proposé deux idées de structures de données qui ont été retouchées et implémentées : l’utilisation d’un HashMap conservé en mémoire pour rendre la mise-à-jour des alertes plus rapide en plus de permettre l’analyse des alertes nouvellement intégrées pour envoyer des notifications aux usagers, ainsi que l’intégration d’un indexage qu’on appellera « tuiles ». Cet indexage est effectué au moment de l’ajout d’une nouvelle zone surveillée par un utilisateur et va permettre, à long terme, d’accélérer les recherches effectuées par le serveur afin de trouver les utilisateurs à qui il faut envoyer une notification car une alerte est positionnée dans une de ses zones surveillées.

[ Infographie du système de tuilage ]

**WebApp :**

Comme expliqué dans le site web du projet, nous étions 5 à travailler sur le projet pendant la période précédant la fin du Hackathon. Par la suite, j’ai été assigné au développement de la *WebApp* car c’était *Charles-Philippe Lepage* qui s’en occupait avant.

Il y a donc eu un effort de *reverse engineering* pour comprendre et modifier le code qui était déjà là afin de mener à terme les points mentionnés ci-dessous (entre autres, il a fallu lire de la documentation sur **AugularJS** et **OpenLayers**) :

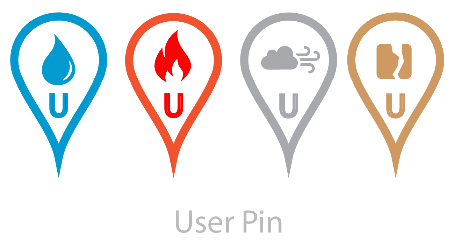
* Ajustement du positionnement des alertes lors de la décision de changer d’une référence de coordonnées (Lat, Lng) à (Lng, Lat).
* Réglage des bugs sur certaines classifications d’alertes et d’icônes.
* Implémentation des images différentes pour les pins des alertes d’usagers.
* Ajouts de catégories de classification d’alertes supplémentaires.
* Léger *refactoring* et ajout de commentaires pour augmenter la maintenabilité.
* Réglage des filtres d’affichage des alertes sur la carte.
* Changement des URLs de requêtes d’API lorsqu’on a migré le serveur.
* Ajout des descriptions des alertes d’usagers dans les infobulles.
* Implémentation de l’ajustement automatique du *zoom level* lors de la recherche d’une localisation.
* Déploiement sur une nouvelle URL et activation de la mise-à-jour automatique du site web lors d’un *commit*.

**Design / UI :**

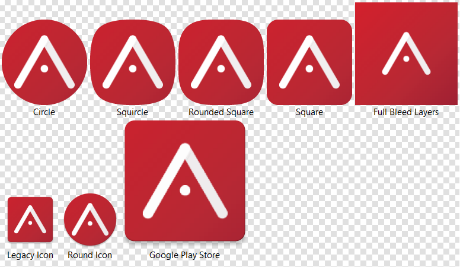
L’attrait visuel d’un logiciel est généralement important pour son succès. Charles-Philippe a été le principal designer et a fourni la majorité du matériel de base durant le Hackathon, mais j’ai aussi contribué quelque peu (en plus d’être presqu’exclusivement celui qui a intégré les dessins de manière fonctionnelle dans l’application Android).

Parmi mes différentes réalisations, on dénote :

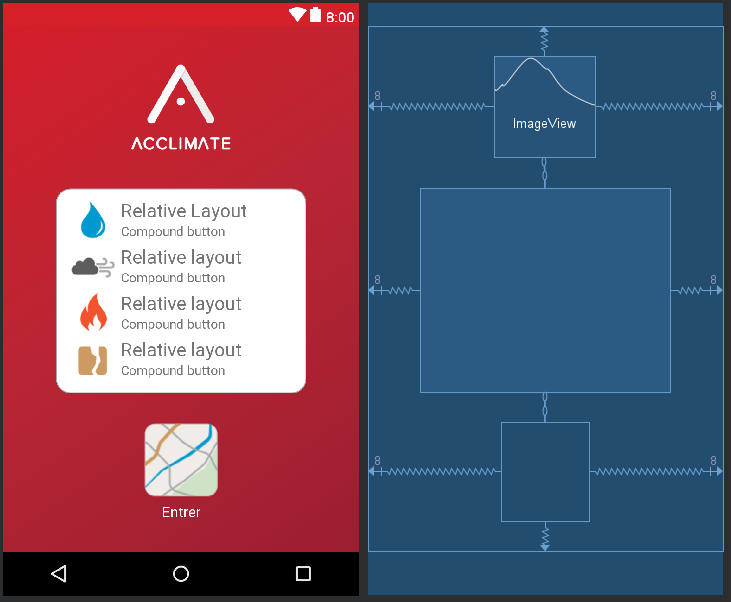
* La création des « Pins usager » qui permettent de différencier plus rapidement si une alerte provient du gouvernement ou d’un usager.

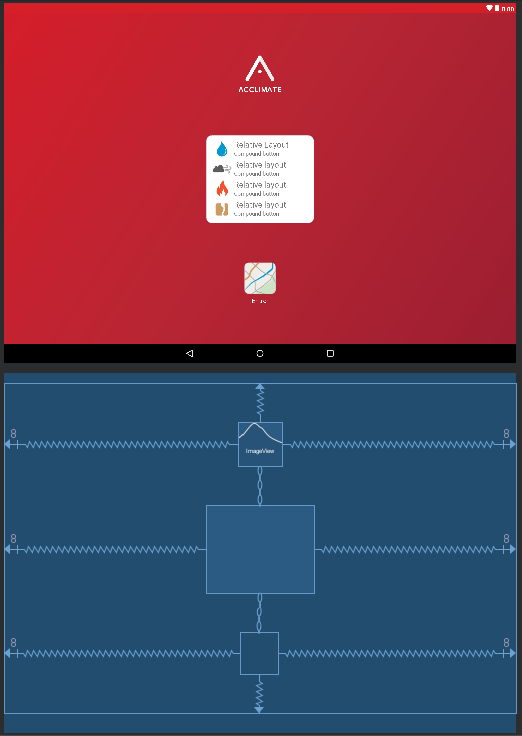


* La génération des différentes versions du Logo qui sont utilisées en fonction de différents contextes (la version d’Android du téléphone qui possède l’application, la recherche de l’application dans Google Play Store, etc.).



* Les différentes « pages » de l’application Android (*Activities* dans le jargon d’Android) qui s’adaptent automatiquement à différentes grosseurs d’écran.





**Application Android :**

Facilement 95% de mon temps a été dédié au développement de l’application Android. Mes contributions sont multiples et [mon rapport hebdomadaire](https://www.webdepot.umontreal.ca/Usagers/p1071435/MonDepotPublic/ProjInfoPerso/) pour le cours dresse une liste assez détaillée de celles-ci. Ce rapport final peut cependant s’offrir de faire une synthèse à l’aide d’une séparation par composantes. Ainsi, j’ai été le développeur principal de l’intégration de :

* Notifications
* Authentification
* Synchronisation
* Zones surveillées
* Préférences

Min SDK = 16

**Langages de programmation, outils et librairies utilisés :**

J’ai travaillé avec deux langages de programmation : **Java 8** et **JavaScript** en plus de faire appel à la **documentation** de [Java 8](https://docs.oracle.com/javase/8/docs/api/), [Android](https://developer.android.com/guide/), [Firebase](https://firebase.google.com/docs/android/setup) et [OpenLayers](http://openlayers.org/en/latest/apidoc/). Les outils et librairies suivants ont été employés :

|  |  |
| --- | --- |
| **Outils** | **Librairies** |
| **IntelliJ** 2018.2 | **AngularJS** v1.6.9 \* |
| **Android Studio** v3.1.4 | **OpenLayers** v4.6.5 (carte de la WebApp) |
| **Gradle** v4.4 | **JackSON** (JSON) |
| **Visual Studio Code** v1.26.1 | **OSMDroid** (carte dans l’application Android) |
| **GitHub Desktop** v1.3.4 | **Firebase Authentication** |
| **GitHub** | **Firebase Cloud Messaging** (notifications) |
| **BitBucket** | **Fused Location** (localisation gps) |
| **Crashlytics (Firebase)** v2.9.5 | **Lombok** (annotations) |
| **Firebase Console** | **Picasso** (images dans Android) |
| **Herokuapp** (avec **UptimeRobot**) | **FancyShowCaseView** (animations dans Android) |
| **Adobe Illustrator CS6** et  **Adobe Photoshop CS3** |  |
| **Node.js** v8.11.4 |  |

\* est en fait un *framework*

Je me dois aussi de faire une mention spéciale pour ma découverte de **Stack Overflow**. 😊

**Résultats obtenus :**

**Défis :**

Pour nous trois, cela a été notre première expérience de développement faisant appel à une base de données, une application mobile et un serveur. En fait, nous entamions à peine le troisième trimestre de ce baccalauréat au moment de nous lancer dans ce projet : il va donc sans dire que ce fut une expérience des plus enrichissantes et formatrices.

**Apprentissage :**

**Propositions d’améliorations futures :**

* Permettre aux gens de s’abonner à une publication effectuée sans l’utilisation d’internet : textos cellulaire.
* Sauvegarder certaines préférences en lien avec le Profile d’un usager au niveau de la BDD afin de permettre leur synchronisation lorsqu’un même usager utilise de multiples appareils
* Utiliser un « Naviguation Drawer » (voir <https://developer.android.com/training/implementing-navigation/nav-drawer>) pour libérer l’espace dans la « Action Bar » dans le haut de la carte (ne laisser que « Search »)
* Un « Splash Screen » pour contrôler le flot d’initiation de l’état de l’application (synchronisation des informations avec la BDD, principalement) ainsi que pour rendre le code plus aéré (maintenance facilitée)
* Stocker les dates en tant que LONG dans la BDD (pour éviter les problèmes de TimeZone, et un ‘long’ prends moins d’espace qu’une String, etc.)
* Intégrer une vérification de la version de l’application afin de pouvoir gérer correctement les synchronisations (si une nouvelle version change quelque chose qui pourrait causer un crash dans une ancienne version, par exemple…)
* Conserver une StrongReference des NetworkRequests pour ne pas que le GarbageCollector les détruise
* Bug : « back to Home » qui fait en sorte que les Overlay de la Map ne s’affichent plus
* Réduire temps de traitement des opérations par le serveur (peut même causer des erreurs lorsque trop long)
* Features de recherche sur les alertes historiques (WebApp, surtout). Peut-être analyse IA ?
* Terminer de permettre l’entrée manuelle d’un nombre de mètres pour l’ajout d’une zone circulaire
* Importance d’une bonne plateforme de communication (code-lint, renvoie vers des messages spécifiques, etc.)
* Stocker les Préférences des usagers dans notre BDD pour la synchronisation (certaines pourraient néanmoins n’être que par appareil)
* Feature de « Alertes GPS » qui utilise la localisation de l’usager pour faire des requête au serveur pour savoir s’il y a des alertes à l’intérieur d’un certain rayon (choisi par l’utilisateur). La fréquence des requêtes peut dépendre du temps ou encore de la distance parcourue (ou les deux).
* Terminer l’implémentation des Préférences (filtres d’alertes sur les notifications)
* Custom actions pour les clics des usagers sur les notifications (centrer sur une alerte, une MZ, etc.)

* ajouter lien vers /doc/presentation.pdf
* ajouter lien vers /doc/rfc/rfc6819-OAuth2\_ThreatModel.pdf
* ajouter images des /Diagrammes/images

**Annexe**

