****

**Acclimate : rapport final**

**par**

**Jérémi Grenier-Berthiaume**

**Olivier Lepage-Applin**

**Sophie Savoie**

**Chapeauté par Mr Houari Sahraoui**

**Dans le cadre du cours**

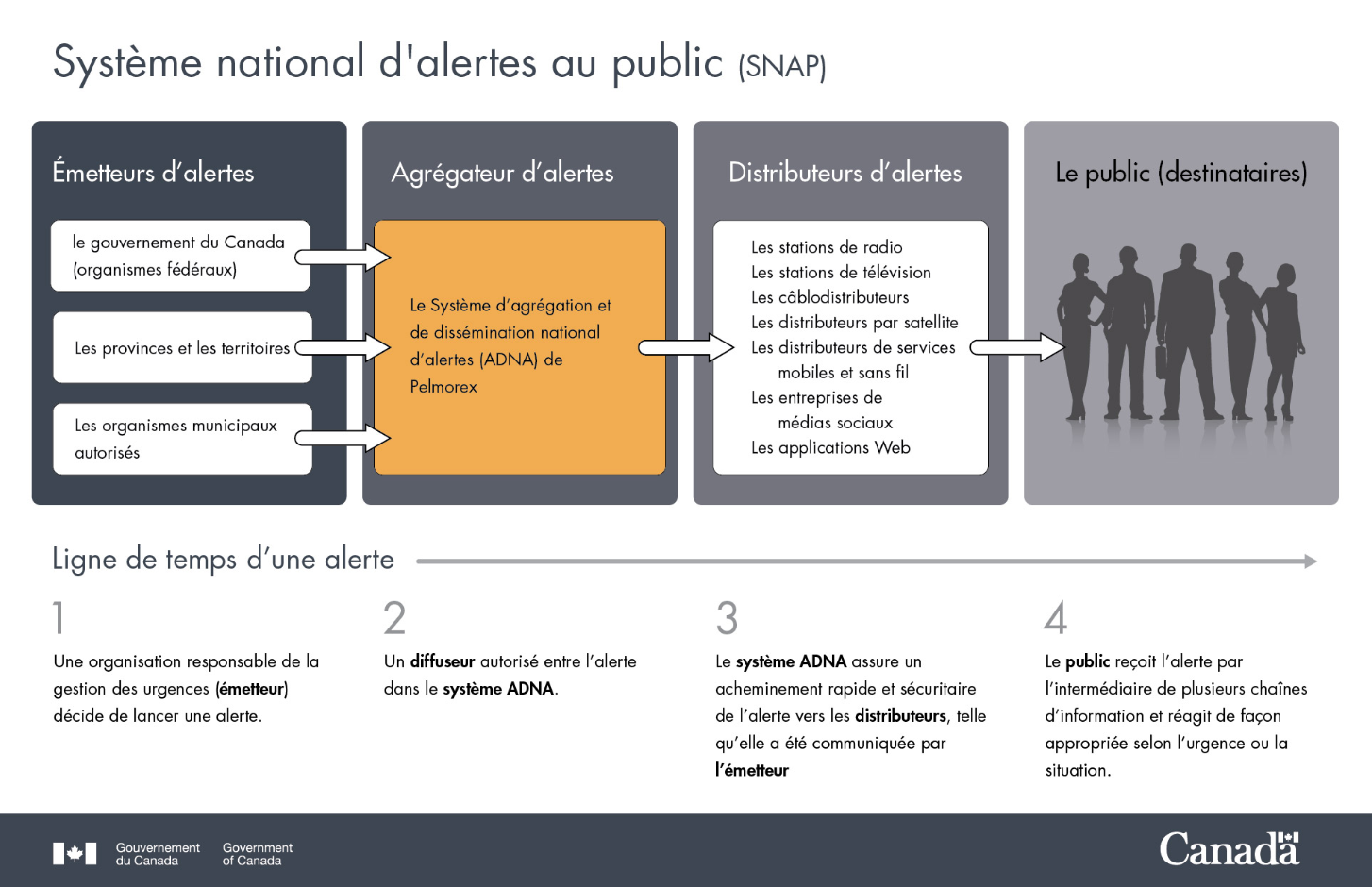
**Projet d’informatique – IFT 3150**

**Vendredi 14 septembre 2018**

**Détails sur le problème :**

L’idée du projet provient de l’observation de Sophie quant 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 le flux RSS des données présentées ci-haut, 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 des alertes publiées. 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 ci-dessous (provenant du précédent 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 plutôt que d’avoir une approche qui n’est qu’unilatéral.



**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 |

**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.
* **Token**: String devant être utilisé afin de mener à terme un certain objectif.
* **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 cependant changer.)
* **OAuth 2**: Protocole 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.

**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 auprès des serveurs de Google afin de permettre l’utilisation des fonctionnalités de l’Admin SDK de Firebase.
* Implémentation des patrons des méthodes de base en lien avec les notifications et la vérification des idTokens (ceux-ci permettent de vérifier l’authenticité et l’intégrité des requêtes envoyées par nos usagers) en s’assurant que tout soit prêt pour une intégration fluide et rapide de la part de Sophie.
* *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** pour améliorer l’efficacité du traitement des requêtes :

* *HashMap* des alertes pour effectuer une mise-à-jour en temps réel plus efficace des alertes et faciliter le traitement des notifications à envoyer. L’idée est de conserver en mémoire la version la plus à jour de la base de données (BDD) afin d’éviter d’avoir à faire une requête pour obtenir l’état actuel de la BDD. Ainsi, il est possible de n’envoyer que quelques requêtes de PATCH à la BDD plutôt que de tout recréer. Il faut savoir qu’une attention doit être portée aux alertes qui ne figurent plus dans le flux RSS du gouvernement car nous ajoutons alors celles-ci dans notre BDD que l’on dit « Historique » (ces données récoltées pourraient possiblement servir pour des projets d’apprentissage machine, par exemple).
* Système d’indexage des zones surveillées qui permet de trouver plus rapidement à quel usager on doit envoyer une notification pour une nouvelle alerte donnée. On utilise une fonction mathématique pour savoir dans quelle région géographique (ou « tuile ») tombe l’alerte, et chaque tuile contient un lien vers toutes les zones surveillées qu’elle contient (il n’y a alors pas autant de calculs et comparaisons à faire avec les coordonnées de l’alerte puisqu’on n’a plus de besoin de passer à travers l’entièreté des zones surveillées).

**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.

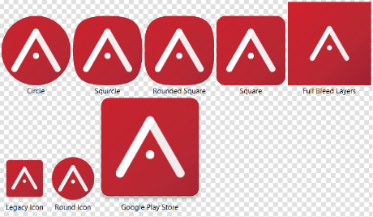
Il y a donc eu un effort pour comprendre et modifier le code qui était déjà là afin de mener à terme les points mentionnés ci-dessous (entre autres, il a fallu lire de la documentation sur **AugularJS** et **OpenLayers**) :

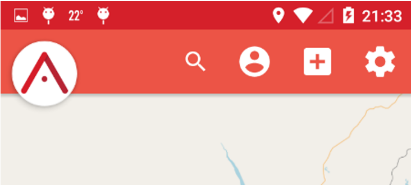
* Ajustement du positionnement des alertes lors de la décision de changer d’une référence de coordonnées (Lat, Lng) à (Lng, Lat).
* Réglage des bugs sur certaines classifications d’alertes et d’icônes.
* Implémentation des images différentes pour les pins des alertes d’usagers.
* Ajouts de catégories 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*.

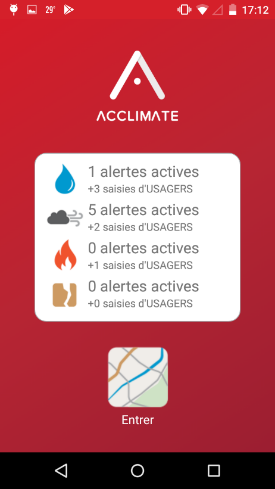
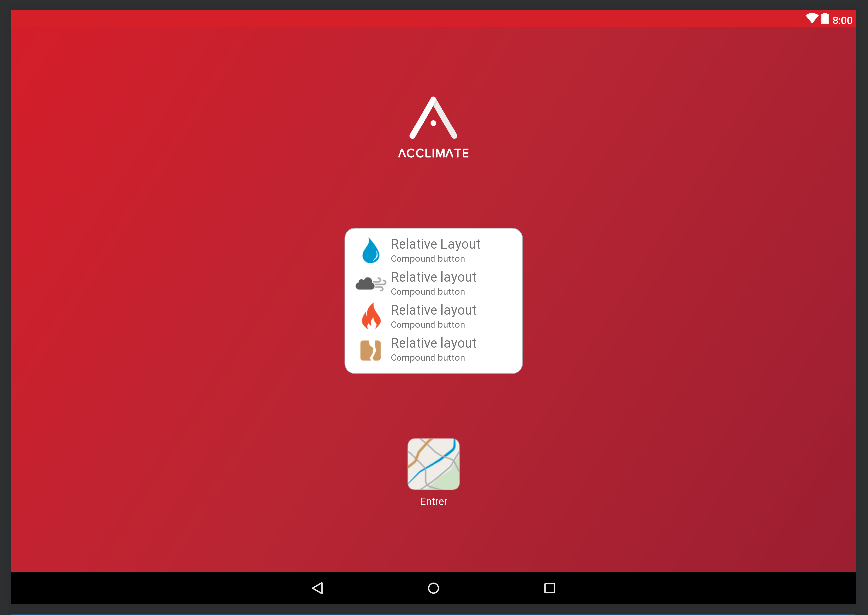
**Design / UI :**

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 :

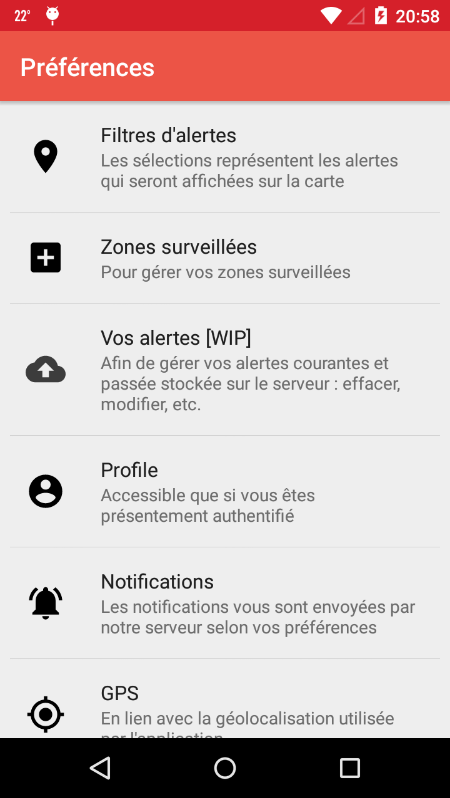
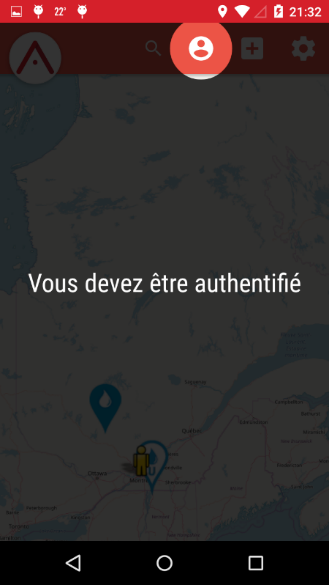
* La création des « Pins usager » qui permettent de différencier plus rapidement si une alerte provient du gouvernement ou d’un usager.



* 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.).
* La ***Toolbar*** (la barre horizontale située tout en haut de l’écran et qui contient généralement des boutons afin que l’usager puisse interagir avec l’application Android).
* Les différentes « pages » de l’application Android (*Activities* dans le jargon d’Android) qui s’adaptent automatiquement à différentes grosseurs d’écran. Par exemple, voici la même Activité avec une résolution de **768 x 1280** et puis de **2560 x 1800** pixels.

* **UI** (ici les pages d’Authentification et des Préférences, ainsi qu’un aperçu d’une animation servant à guider l’utilisateur):

**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 présenter 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
* Recherche d’adresses/endroits/etc.
* Localisation GPS

blabla

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

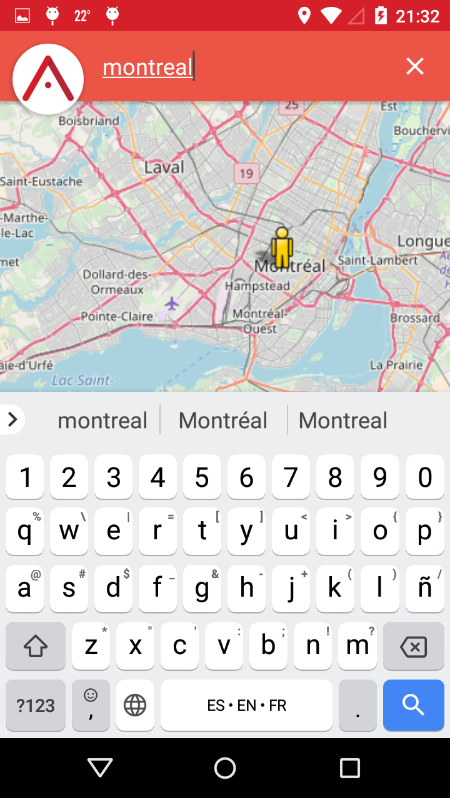
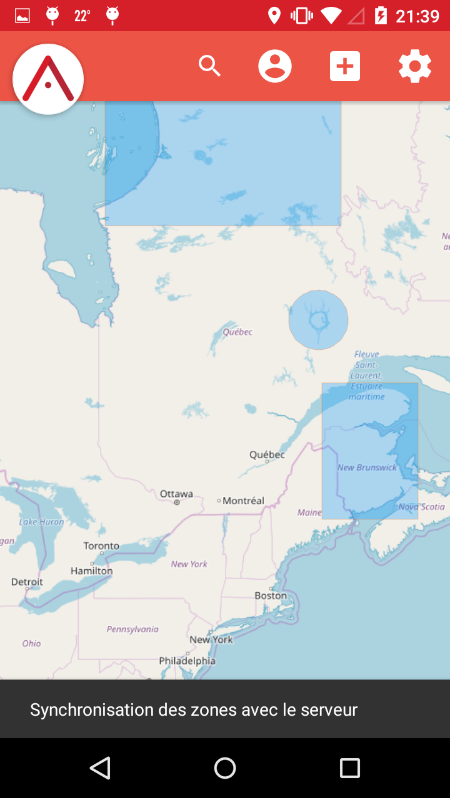
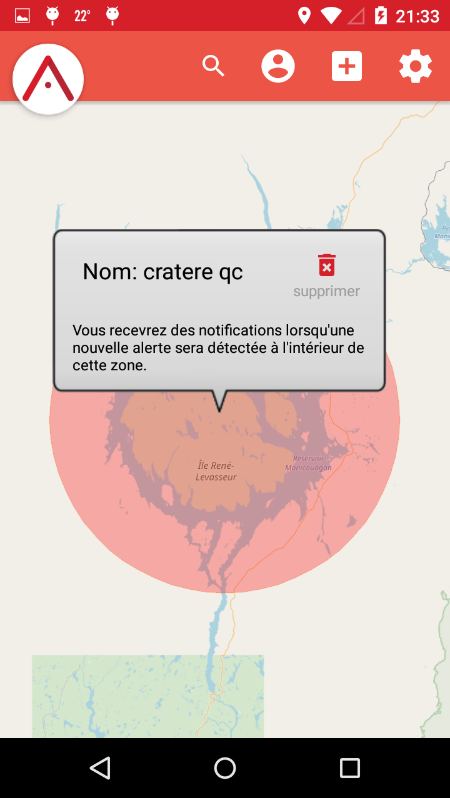
J’ai travaillé avec deux langages de programmation : **Java 8** et **JavaScript** en plus de faire appel à la **documentation** de [Java 8](https://docs.oracle.com/javase/8/docs/api/), [Android](https://developer.android.com/guide/), [Firebase](https://firebase.google.com/docs/android/setup) et [OpenLayers](http://openlayers.org/en/latest/apidoc/).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

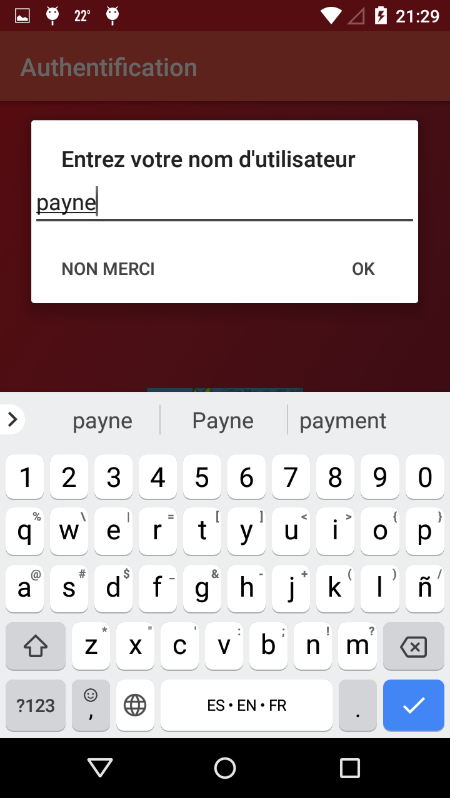
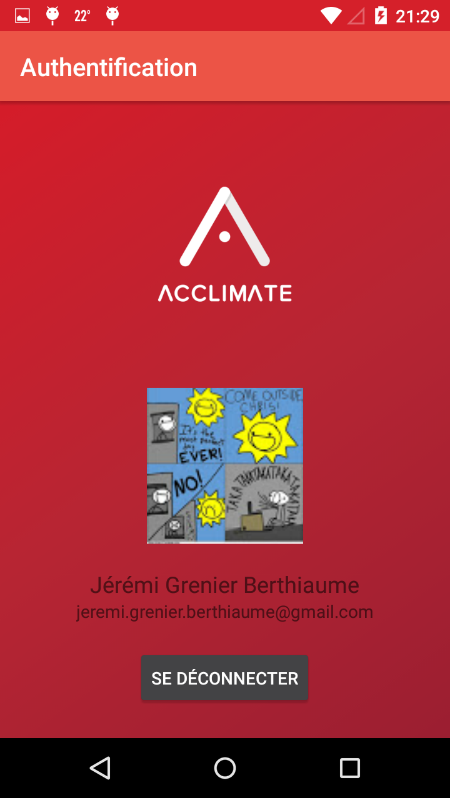
|  |  |
| --- | --- |
| **Outils** | **Librairies** |
| **IntelliJ** 2018.2 | **AngularJS** v1.6.9 \* |
| **Android Studio** v3.1.4 | **OpenLayers** v4.6.5 (carte de la WebApp) |
| **Gradle** v4.4 | **JackSON** (JSON) |
| **Visual Studio Code** v1.26.1 | **OSMDroid** (carte dans l’application Android) |
| **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 :**

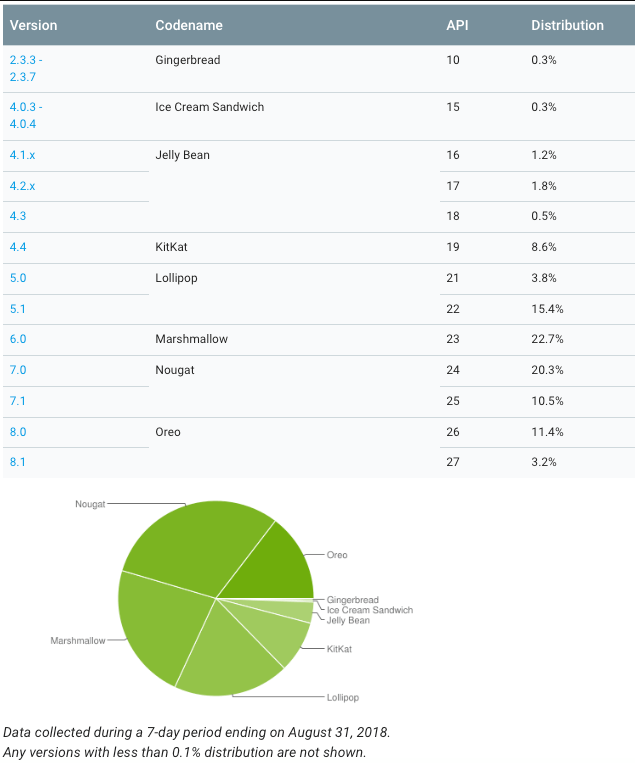
  

**Défis :**

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.

Min SDK = 16 => 99.4% du market share couvert ([source](http://www.gadgetdetail.com/android-version-market-share-distribution/))



**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
* 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 requêtes 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.)
* « Don’t remember me » option in AuthUI
* Automatically delete unverified users after X days
* Red number to indicate amount of new Alerts active on the app’s icon (like FaceBook)

**La suite du projet :**

Je pense que l’application a atteint un stade qui dépasse le « *proof of concept* » et qui pourrait donc intéresser certains partis. Son utilité peut s’étendre à plusieurs domaines et nous gardions en tête ce fait durant le développement afin de nous assurer qu’il ne serait pas trop difficile d’adapter l’application à des besoins changeants ou encore des cas d’utilisation complètement différents.

De plus, comme mentionné dans l’introduction, la motivation derrière le projet était de répondre à une lacune du gouvernement. Nous allons donc discuter de notre stratégie d’approche afin de présenter le projet pour qu’il soit possiblement intégré par le gouvernement.

Finalement, j’ai l’intention de proposer à l’équipe de soumettre notre candidature pour le [Prix pour service exemplaire en sécurité civile](https://www.publicsafety.gc.ca/cnt/mrgnc-mngmnt/xmplry-wrds/ctgrs-fr.aspx#a01) considérant qu’une des catégories concerne les « Communautés résilientes. » Indépendamment de l’obtention de ce prix, il me semble que les différents exemples de volets que pourrait couvrir un candidat potentiel sont de très bons points de départ pour tenter d’obtenir une meilleure vision à long terme du genre d’améliorations que l’on pourrait apporter à l’application.

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

**Annexe**

