
POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Département de génie informatique et génie logiciel

INF3995

**Projet de conception d'un système informatique**

Proposition répondant à l'appel d'offres
No. A2020-INF3995 du département GIGL.

***Conception d'un système* de prédiction de données sur l'usage du BIXI**

Équipe No < *01* >

*William BALEA*
Cyrille TALLA FONGANG
*Kuchue KAMGAIN-DJOKO*
Jean-Michel LASNIER
*Dalyna PAK*

24 Septembre 2020

# Table des matières

# 1. Vue d'ensemble du projet

## *1.1 But du projet, portée et objectifs (Q2.1 et Q4.1)*

La présente réponse d'appel d'offres fait suite à la demande de proposition No. A2020-INF3995 de la compagnie BIXI au sujet de la fourniture d'une application capable d'effectuer la prédiction de données sur l'usage du BIXI. Le but du projet est de mettre à la disposition des utilisateurs une solution informatique permettant premièrement de mesurer l'intérêt des usagers pour les services BIXI, deuxièmement d'obtenir des prédictions d'utilisations du système BIXI en se basant sur son utilisation au cours des dernières années.

La solution proposée intègre une application Android pour l'usager, une application PC pour l'administration du système, d'un serveur web central et d'engins de données pour la persistance des données. En ce qui concerne la portée du projet, elle est limitée à la réponse de l'appel d'offres de la Compagnie BIXI. Ainsi dans une approche graduelle (en deux phases), il est attendu à la fin du projet les livrables suivants:

- Un serveur web fonctionnel: contenu dans un conteneur Docker et permettant de gérer les opérations de sondage, d'intégrer les utilitaires Traefik et Docker-compose, de vérifier la disponibilité des engins de données; et enfin de changer le mot de passe de l'administrateur;
- Trois engins de données : contenus dans des Dockers et effectuant l'analyse des données, la production des tableaux et graphiques statistiques qui sont ensuite retournés à l'application Android; permettant la visualisation d'une station BIXI sur une carte; assurant une gestion locale des logs; permettant de faire des prédictions à un niveau raisonnable; gérer la perte des engins de données;
- Une application Android : permettant de spécifier en entrée l'adresse IP du serveur avec qui elle échange les données et d'effectuer un sondage auprès des utilisateurs; offrant une interface conviviale et ergonomique, qui permet d'afficher plusieurs écrans définis plus loin dans le document.
- Une application PC: qui permet de consulter les résultats de sondage de façon sécuritaire, d'effectuer un changement de mot de passe; de recevoir les logs des engins de données;

## *1.2 Hypothèses et contraintes (Q3.1)*

Le système envisagé dans le cadre de ce projet est divisé en cinq composantes principales. Les hypothèses et contraintes liées à sa mise en œuvre sont liées non seulement aux éléments techniques nécessaires (langage de programmation, plateforme ou environnement de développement, librairies, etc.), mais aussi à des éléments externes.

Tout d'abord, le serveur web central doit être développé en C++ dans un environnement Linux. Pour la gestion des requêtes des clients, la librairie Pistache IO

nous semble adéquate au regard de ses fonctionnalités. En effet, il s'agit d' un cadriciel REST écrit en C++ qui fournit une abstraction HTTP de bas niveau. En outre, il fournit un client HTTP et un serveur pour créer une interface REST. Au sujet de la base de données au niveau du serveur, le SGBD MySQL est le choix idéal pour ses nombreuses fonctionnalités de sécurité intégrées, sa flexibilité quant au type de données, sa performance, sa gestion des utilisateurs et les multiples contrôles d'accès. Ensuite, le logiciel des engins de données sera développé en Python 3.6 et plus. Les librairies Pytorch et Matplotlib seront intéressantes à utiliser pour la prédiction des données. De plus, les engins de données doivent rapporter des messages logs qui seront disponibles sur l'application PC seulement.

Notons que le déploiement du serveur web central et des engins de données se feront principalement à l'aide de Docker. Ceux-ci seront placés chacun dans un conteneur Docker et regroupés par la suite avec le serveur web dans un autre conteneur Docker Compose. De plus, le proxy-inverse Traefik doit être utilisé pour assurer un trafic sécurisé entre les principales composantes. Par ailleurs, le logiciel de l'application Android sera développé en Kotlin dans un environnement de développement nommé Android Studio. Grâce à ce dernier, il sera possible de créer une application Android qui fonctionne obligatoirement sur une tablette mobile. Plus précisément, le Galaxy Tab A avec Android 9. Finalement, en ce concerne le logiciel de l'application PC, l'interface graphique doit se faire sans l'aide de fureteur internet et il doit être autonome. Puisqu'il n'y a pas de contrainte de langage de programmation, le C++ et la librairie Qt semblent appropriés pour ce type de développement logiciel. L'interface usager devra être séparée selon 4 onglets distincts et notre application fonctionnera sous Windows.

Cependant, comme mentionnés plus haut, d'autres éléments externes peuvent modifier les contraintes techniques sont évoquées. Par exemple, des imprévus en cours de route qui peuvent entraîner des changements dans le projet. C'est le cas particulier de la pandémie, qui rend difficile la tenue des rencontres en présentiel. Tout se fait principalement à distance. De plus, le client pourrait apporter des changements dans les requis ou faire des ajustements de dernière minute auxquels l'équipe devra s' adapter. Ainsi, le mot clé à retenir est s'adapter.

## *1.3 Biens livrables du projet (Q2.3)*

Au terme de ce projet qui s'étend sur environ quatre mois, il est prévu de produire une solution en 3 livrables. Lesquels permettront au comité des données du BIXI d'apprécier ou d'évaluer l'état d'avancement du projet. Il est important de souligner que l'équipe « 01» s'adaptera à toutes les modifications apportées à ceux-ci par le client dans la période de réalisation du projet.

Le premier livrable est un prototype à préparer pour le 24 septembre 2020 qui permettra aux membres de l'équipe de projet de se familiariser avec les outils choisis. Ce prototype accompagnera la présente réponse d'appel d'offres. On s'attend à :

- Mettre en place un serveur web dans un conteneur Docker

- Pouvoir envoyer des requêtes HTTP à partir de l'application Android vers le serveur web avec une interface REST. La communication pourrait être qu'un simple message de log envoyé par la tablette et affiché dans le serveur.
- Une reproduction de l'exemple en Python de l'article sur les prédictions de température. Le tutoriel incitera aux développeurs de démontrer leur compréhension des opérations impliquant l'intelligence artificielle.

Le second livrable sera présenté le 29 octobre 2020. Il servira surtout au comité de données BIXI de voir l'avancée du projet, mais aussi de laisser place à certaines améliorations à la suite des commentaires de leur part pour l'équipe. Les fonctionnalités suivantes seront démontrées pour chaque composante:

- Serveur web :
    - Un serveur web qui peut recevoir via HTTPS les sondages envoyés à partir de la tablette Android.
    - Configuration de Traefik comme le proxy-inverse au serveur afin de distribuer les requêtes envoyées par l'application Android.
    - Mise en place de Docker Compose pour gérer le serveur web central et les 3 engins de données
    - Les sondages sont stockés dans une base de données MySQL.
- Engins de données :
    - Possibilité, de leur part, d'accéder et de préparer les données statistiques BIXI prétéléchargées de kaggle.com.
    - Pouvoir faire des analyses grâce au IA sur les données. Le raffinement de la prédiction n'est pas obligatoire à ce stade
    - Possibilité de créer des graphiques avec les données.
    - Retourner tous les résultats désirés à la demande de l'application Android.
    - Produire des messages de logs visibles localement dans les serveurs d'engins.
    - Mettre les engins dans des conteneurs Docker, mais pas encore de gestion de perte des engins à faire à ce stade.
- Application Android :
    - Voir toutes les différentes vues possibles pour l'usager. L'esthétisme et l'ergonomie ne sont pas la priorité à ce stade.
    - Pouvoir spécifier l'adresse IP du serveur et entrer les données du sondage
    - Envoie de requêtes et réponses des données demandées aux 3 différents engins
    - Affichage des données des engins 1 et 2 seulement selon plusieurs formats conviviaux à l'utilisateur. La carte peut être manquante à ce stade.
    - Aucune vérification de perte des engins à faire
- Application PC :
    - Les 4 onglets de l'application sont visibles et navigables.
    - Possibilité de consulter les tous les résultats de sondage dans un onglet de façon sécurisée

    - Le changement de mot de passe n'est pas encore implémenté à ce stade.
    - Les autres onglets sont occupés par une interface prête à recevoir et afficher les logs des engins de données, mais sans que l'application puisse recevoir les messages de logs.

Le dernier livrable qui traduit le projet complété sera remis pour le 1er décembre 2020. Toutes les fonctionnalités prévues par le contrat devront être présentées et fonctionnelles à savoir:

- Serveur web :
    - Vérification de la disponibilité des engins à tout moment et indication visuelle le cas échéant.
    - Changement de mot de passe possible pour l'administrateur
- Engin de données :
    - Envoie des données de recherche afin d'obtenir une visualisation d'une station BIXI sur la carte Android.
    - Raffinement de la prédiction à un niveau raisonnable.
    - Réponse face à une perte de contact des engins de données
- Application Android :
    - Complétion de toutes les vues manquantes
    - Amélioration de l'esthétisme et l'ergonomie de l'interface face aux remarques et évaluations lors du livrable intermédiaire.
    - Indication visuelle lors de la perte de contact avec les engins
- Application PC :
    - Réception des messages de logs des engins de données dans leur onglet respectif
    - Ajustement de l'interface afin d'assurer l'ergonomie pour l'usager et la cohérence avec l'application Android.
    - Interface pour changer le mot de passe de façon sécuritaire.

# 2. Organisation du projet

## *2.1 Structure d'organisation*

L'équipe de projet fonctionnera dans un mode décentralisé. Ce type de structure d'équipe sied mieux au contexte où nous sommes cinq membres dans l'équipe pour réaliser un projet qui requiert plusieurs connaissances et composantes techniques avancées. Chaque membre se concentrera sur une ou deux composantes principales du développement, mais l'organisation décentralisée nous donnera la flexibilité de réajuster nos ressources si nécessaire. Cette structure offre l'avantage d'accroître la vitesse du partage d'information et surtout, de s'assurer que les expériences et compétences de chacun seront utilisées à notre avantage. Cyrille sera notre coordonnateur (ou Scrum master) pour le projet. Son rôle sera de s'assurer du transfert d'information et d'organiser et gérer les réunions. Il accomplira parallèlement avec les autres membres de l'équipe (Dalyna, Robin, William et Jean-Michel) le rôle

de développeur-analyste. Ce rôle sert au développement des différentes fonctionnalités du produit selon les livrables et prévenir l'équipe de tout retard.

Pour ce qui est du développement logiciel du projet, tous les membres seront impliqués directement dans une ou plusieurs composantes du projet. Nous avons identifié 5 composantes principales à développer lors de ce projet: Application Android, Application PC, Serveur web, les engins de données et Docker. Cyrille et Robin s'occuperont des applications Android et PC, les engins de données seront développés par Dalyna et Jean-Michel, le serveur HTTP sera fait par Dalyna et William et le conteneur Docker sera assuré par William. Il s'agit ici simplement d'une distribution initiale des tâches et des ajustements dans l'organisation seront apportés si nécessaire. En travaillant de façon décentralisée, nous pourrons facilement venir en aide les uns aux autres et cela assurera une bonne compréhension de l'ensemble du projet pour chacun des membres. Le tableau 1 présente les rôles des membres de l'équipe.

*Tableau 1: Rôles des membres de l'équipe*

| | Cyrille | Dalyna | Jean-Michel | Kuchue | William |
|---|---|---|---|---|---|
| Coordonnateur | x | | | | |
| Développeur -Analyste | x | x | x | x | x |
| App Android | x | | | x | |
| App PC | | | | x | x |
| Engins de données | | x | x | | |
| Serveur HTTP | | x | | | x |
| Docker | | | | | x |

## *2.2 Entente contractuelle*

Le type d'entente contractuelle que nous proposons sera un contrat de livraison clé en main (prix ferme), car le contracteur (équipe de projet) doit livrer le produit final. Le paiement final sera émis lors de la livraison et de l'acceptation par le client. Voici les avantages et inconvénients de ce type d'entente contractuelle (Jérôme Collin, 2020) .

Avantages:

- Procure l'assurance des coûts finaux.
- Assure le promoteur d'un avis rapide lors de délais et de fluctuation des coûts dû à des changements.
- Requiers un suivi minimal des travaux par le promoteur.

Désavantages:

- Requiers une connaissance exacte de la demande avant d'accorder le contrat et des spécifications détaillées et vérifiables.

- Requiert un délai et des coûts supplémentaires pour monter un cahier des charges, solliciter et évaluer les soumissions.
- Les frais reliés à la soumission et les risques inhérents peuvent réduire le nombre de soumissionnaires.
- Les montants imprévus pour couvrir les travaux à haut risque peuvent faire augmenter le coût.

## *3. Solution proposée*

La figure 1 ci-après présente une vue globale des principales composantes du projet et leurs connexions.

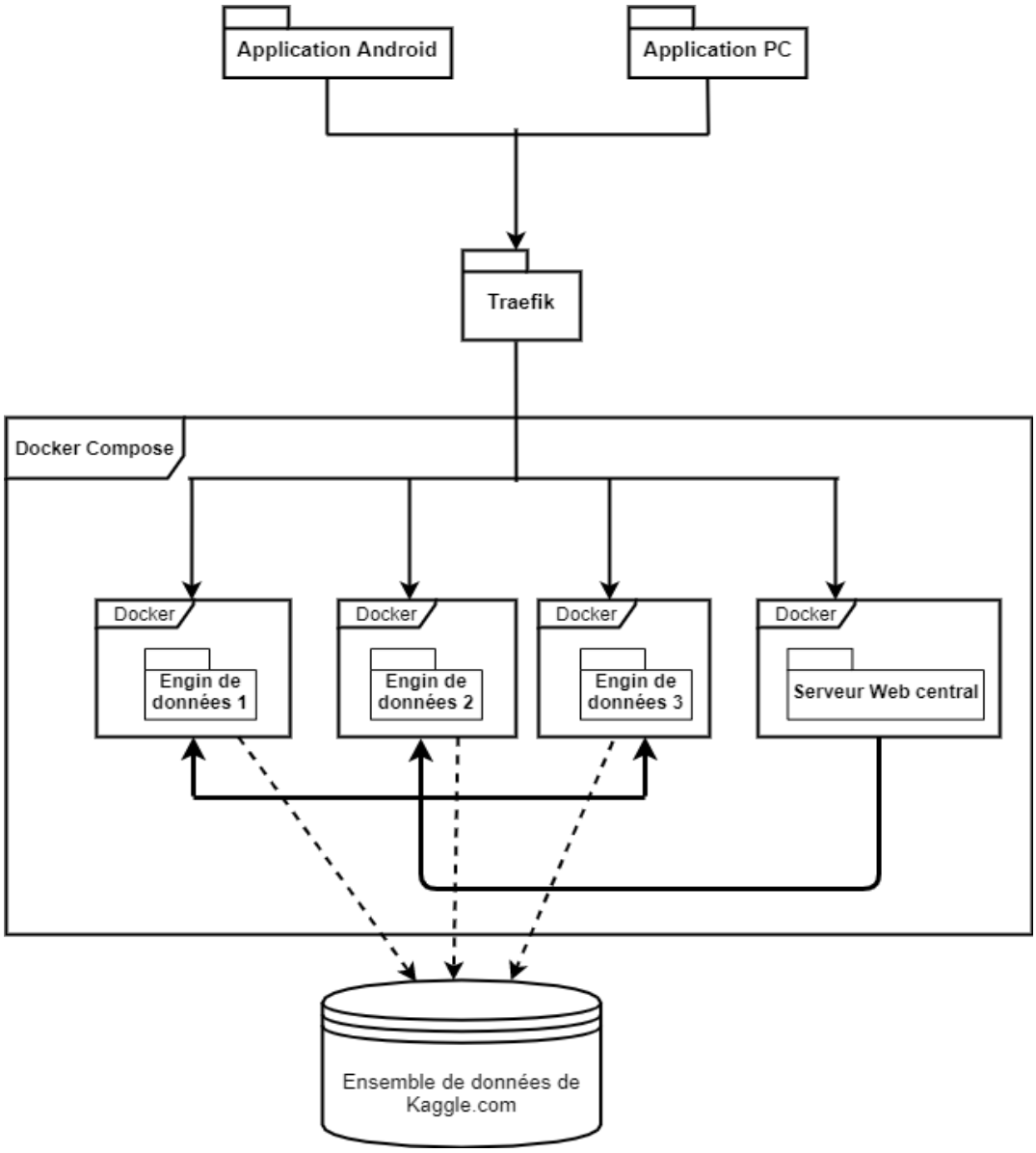


*Figure 1: Vue globale des cinq composantes du projet*

## *3.1 Architecture logicielle sur serveur (Q4.2)*

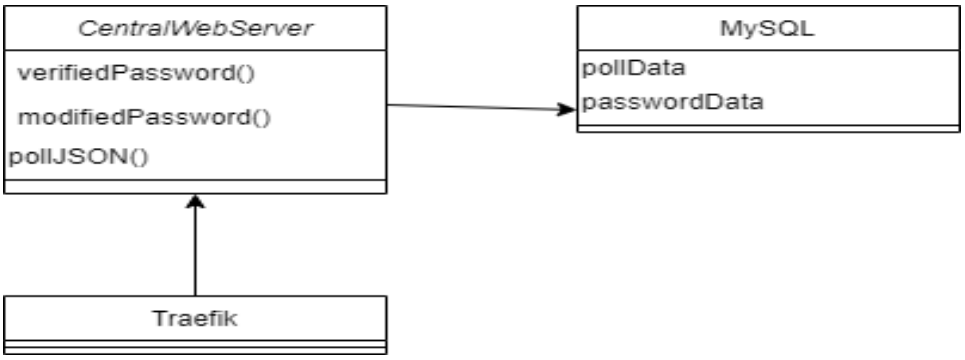


*Figure 2 : Architecture logicielle du serveur web central*

Tout d'abord, nous allons utiliser Traefik 2.2.11 pour gérer et distribuer les différentes requêtes REST aux différents conteneurs Docker qui contiennent le serveur web central et les 3 engins de données. Ce proxy-inverse servira aussi à gérer les certificats de sécurité autosignés et condamner les accès non autorisés. On s'assurera de rediriger le port 80 au port 443 pour utiliser que le HTTPS et renforcer la sécurité. On offrira une interface web de monitorage Traefik sur le port 8080. Le serveur web central sera écrit en C++ 14 avec la librairie Pistache IO comme framework REST. Le serveur web s'occupera de gérer le mot de passe administrateur, de recevoir les logs des engins et de vérifier les états des engins. Nous choisissons Pistache IO pour sa documentation détaillée et ses exemples de code. Le serveur web sera dans un Docker et les conteneurs seront gérés par Docker Compose version 1.27.3 qui est la dernière à ce jour. Lors de la perte d'activité d'un engin de donnée, le serveur web s'occupera d'aviser les applications PC et Android. Afin de stocker les résultats des sondages, des données BIXI et du mot de passe administrateur, nous allons utiliser MySQL 8.0. Voici quelques requêtes qui seront implémentées du côté serveur :

| Requête | Route HTTPS | Description |
|---|---|---|
| POST | /usager/login | Vérifier la connexion admin. |
| PUT | /usager/motdepasse | Changer le mot de passe admin. |
| PUT | /sondage | Envoyer les sondages au serveur. |
| GET | /sondage | Afficher les sondages dans l'application PC. |
| GET | /status | Assurer l'intégrité du système de données. |

## 3.2 Architecture logicielle générale des engins de données

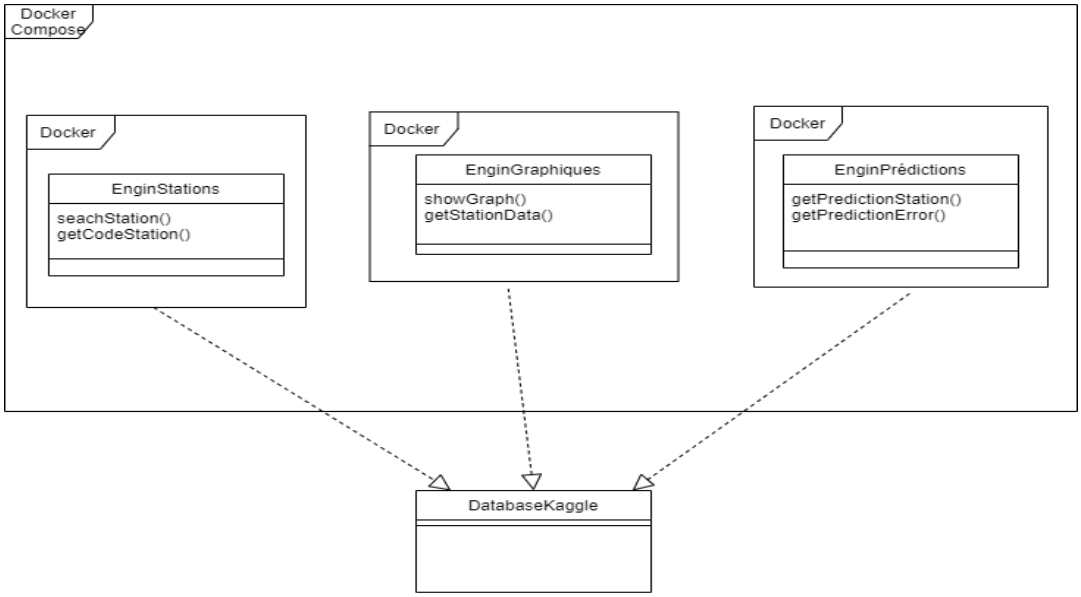


*Figure 3: Architecture logicielle générale des trois engins de données*

Notre solution implémente trois engins de données qui doivent fournir les informations et la visualisation des stations BIXI sur la carte, le deuxième engin doit fournir les données et statistiques des stations. Le troisième engin doit faire l'apprentissage et doit être capable de faire des prédictions. Pour ce faire, nous utiliserons les librairies Sklearn, Numpy, Pandas et Matplotlib 3.3.2 qui sont très efficaces pour la résolution des problèmes d'intelligence artificielle avec le langage de programmation Python (3.6 et plus). Celles-ci permettront de créer notre algorithme de prédiction en utilisant les forêts aléatoires. Les engins de données « EnginStations » et « EnginGraphiques » serviront principalement à retourner des données brutes pour donner suite aux requêtes du serveur. Les données d'utilisation du BIXI proviendront du site kaggle.com et seront enregistrées dans une base de données MySQL.

## 3.3 Architecture logicielle sur tablette (Q4.3)

En raison des nombreux avantages qu'elle offre, nous proposons une architecture MVVM (Modèle, Vue, Vue-Modèle) pour notre application Android. En effet avec MVVM, il est facile de modifier la vue sans impact sur la Vue-Modèle (et vice versa). Elle permet de tester de manière séparée les différents éléments de notre solution. Elle simplifie la gestion des affichages entre différentes rotations d'écran et enfin, elle permet une maintenance aisée du projet.
Notre solution utilisera plusieurs librairies encore appelées « Architecture Components » proposées par Android Jetpack de Google notamment: Lifecycles avec ses composants LifecycleOwner et LifecycleObserver; LiveData; Test; Repository; ViewModel; Navigation; Workmanager.

Nous utiliserons la librairie Retrofit2 (client REST) pour récupérer les données qui se trouvent sur le serveur web à l'aide de requêtes réseau. Vu les exigences du projet concernant le format des données (JSON), nous utiliserons aussi la librairie GSON. Les différents modules ou couches de notre architecture sont les suivantes :

**a)** Le Modèle: constitué principalement de la classe Repository et des deux bases des données suivantes disponibles auprès du serveur web et des engins de données :
- SurveyDatabase avec la table *« User »* pour stocker les données recueillies pour le sondage ;
- StationDatabase avec les tables suivantes:
  - *Table Station pour stocker les données relatives aux stations (coordonnées, statistiques et prédictions)*.
  - *Table « Statistics » pour stocker les statiques pour chaque station*.
  - *Table « RawData » pour stocker les données brutes pour chaque station*.
  - *Table « Prédictions » pour stocker les prédictions pour chaque station.*

**b)** La vue: qui correspond à ce qui sera affiché. Elle est constituée essentiellement des activités/fragments (qui sont nos « LifecycleOwners ») ci-après:
- « **ConfirmServerAdressActivity** »**:** Il s'agit de la page vers laquelle l'usager est dirigé lorsqu'il lance l'application. Elle lui permet de confirmer l'adresse IP du serveur.
- « **SurveyActivity** » qui permet de collecter et transmettre au serveur les informations sur le sondage éventuel de l'usager.
- « **MainActivity** » qui permet à l'utilisateur de pouvoir exploiter les fonctionnalités suivantes de l'application à savoir: rechercher une station, parcourir la liste des stations, obtenir les informations pour une station particulière et des informations globales pour l'ensemble des stations. Cette activité contient les 6 fragments :
  - « ***SearchStationFragment*** » *qui contient une barre de recherche pour effectuer la recherche d'une station.*
  - « ***ListStationFragment*** » *situé en dessous du précédent présente la liste des stations. Lorsqu'une recherche a été lancée, cette liste présente la liste des résultats de la recherche. En cliquant sur une option de menu que présente chaque station de la liste, on a la possibilité de choisir l'un des fragments (qui s'affiche dans la partie de droite) ci-après :*
  - « ***StationsCoordinatesFragment*** » *qui affiche le nom et les coordonnées (latitude et longitude) de la station.*
  - « ***RawDataFragment*** » *qui affiche les données brutes relatives au nombre de départs de parcours BIXI par heure, jours de semaine, mois pour une station suivant l'année.*
  - « ***StatisticsStationFragment*** » *qui affiche le nombre de départs de parcours BIXI par heure, jours de semaine, mois pour une station suivant l'année.*

- « ***PredictionStationFragment*** » *qui affiche une prédiction du nombre de départs de parcours BIXI par heure, jours de semaine, mois pour l'année à venir pour une station donnée suivant l'année.*

- « ***GlobalStatisticsStationActivity*** » qui présente les statistiques globales de BIXI par année.
- « ***GlobalRawDataStationActivity*** » qui présente les données brutes globales de BIXI par année.
- « ***GlobalPredictionStationActivity*** » qui présente les prédictions globales de BIXI.

**c)** La vue-modèle: qui fera le lien entre la vue et le modèle. Elle s'appuie sur la notion de «data binding » pour mettre à disposition de la vue les données du modèle. Nous utiliserons ici la librairie « LiveData » qui permet d'observer plus facilement les changements en respectant le cycle de vie de l'application. Elle est constituée essentiellement des classes suivantes:

- StationViewModel;
- StatisticsViewModel;
- SurveyViewModel;
- RawDataViewModel;
- PredictionViewModel.

Nous aurons également une tâche périodique (toutes les 15 secondes) « **CheckConnectivity** », qui l'état de la connexion du système. Nous utiliserons la librairie « Work » de Jetpack pour la gestion des tâches périodiques. En cas de non-connexion, un « Toast » sera affiché pour indiquer à l'usager un échec de connexion avec le serveur. Soulignons que le design de nos activités et fragments sera majoritairement fait à partir des outils (couleur, components...) que propose Google Material Design.

La figure ci-dessous résume l'architecture logicielle de notre application Android.

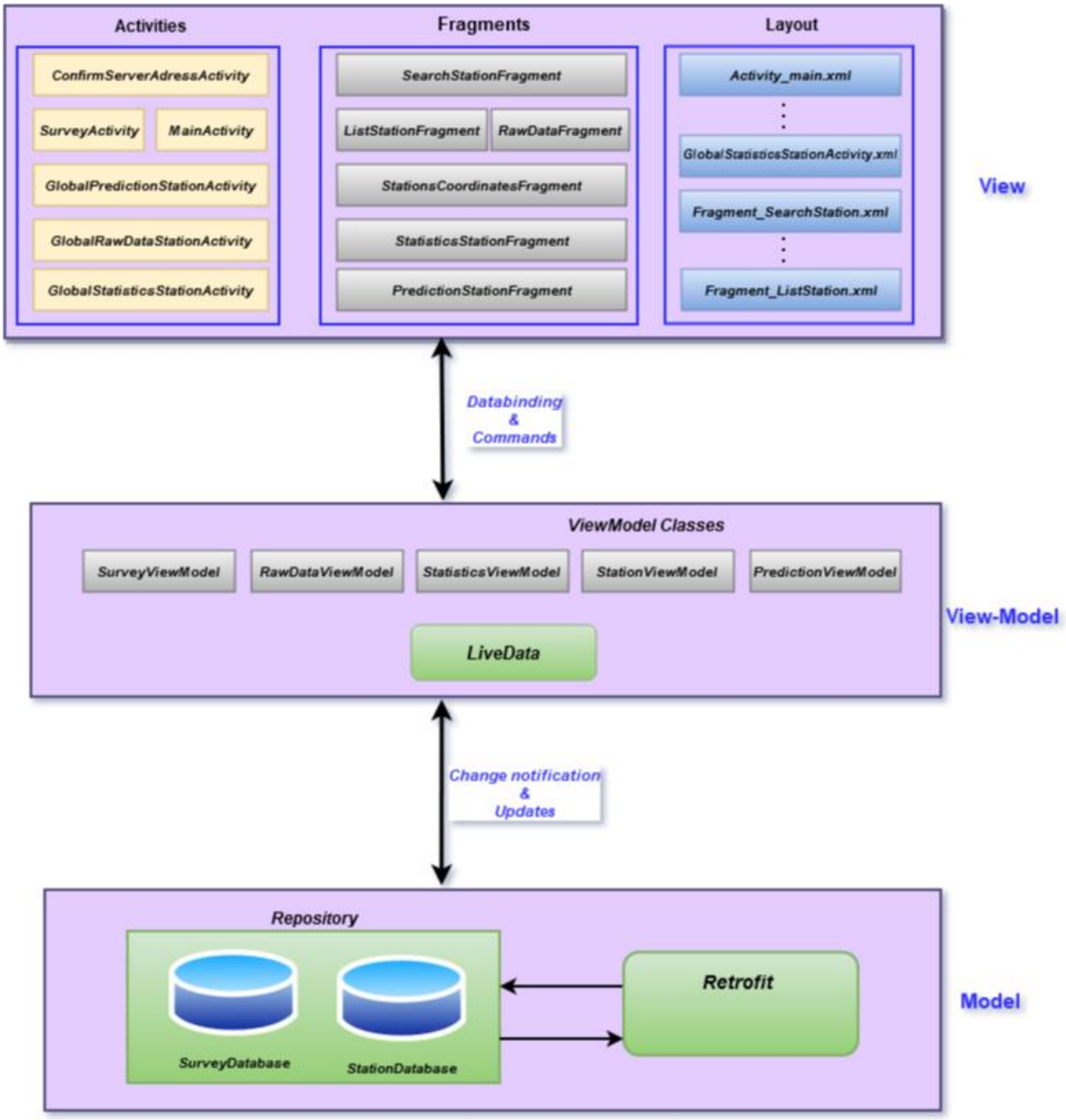


Figure 4: Architecture logicielle de l'application Android

## 3.4 Considérations logicielles de l'application sur PC

L'application PC sera faite en C++ pour Windows. La librairie utilisée pour l'interface sera la dernière version 5.15 de Qt. Afin d'avoir une interface cohérente avec l'application Android, les guides du *Material Design* de Google seront appliqués grâce aux fonctionnalités intégrées de Qt Quick Controls.

# 4. Processus de gestion

## *4.1 Estimations des coûts du projet*

Compte tenu des spécifications du projet, le coût estimatif de ce dernier est lié essentiellement à la main d'œuvre. Il s'agit des salaires des personnels de l'équipe du projet. Cette dernière est constituée principalement d'analystes développeurs et d'un coordonnateur. Chaque personnel dispose du matériel informatique (ordinateur, tablette et logiciels) nécessaire pour accomplir ses tâches. Le tableau ci-dessous présente une estimation des coûts du projet en lien avec la planification et le diagramme Gantt du projet présenté plus loin.

*Tableau 2: Estimation des coûts du projet*

| Type de personnel | Nombre de personnel | Taux horaire ($/h) | Nombre d'heures estimé de travail | Coût total |
|---|---|---|---|---|
| Coordonnateur de projet | 1 | 145 | 126 | 18 970$ |
| Développeur - Analyste | 4 | 130 | 126 | 73 080$ |
| **Coût total du Projet** | | | | **92 050$** |

## *4.2 Planification des tâches (Q2.2 et Q11.2)*

Les figures 5 et 6 ci-dessous présentent la planification /répartition des tâches du projet dans le temps et entre les membres de l'équipe.

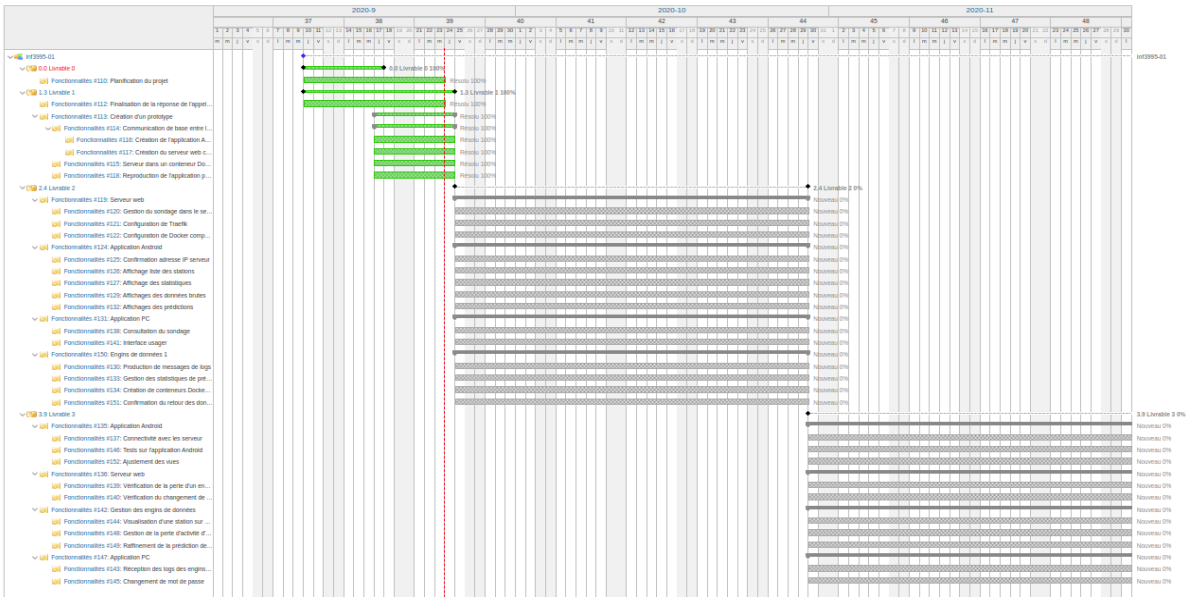

*Figure 5: Diagramme de Gantt du projet*

| # | Tracker | Statut | Priorité | Sujet | Assigné à | Mis-à-jour |
|---|---|---|---|---|---|---|
| 152 | Fonctionnalités | Nouveau | Normal | Ajustement des vues | Kuchue Kamgain-Djoko | 15/09/2020 21:14 |
| 151 | Fonctionnalités | Nouveau | Normal | Confirmation du retour des données à l'application Android | Jean-Michel Lasnier | 15/09/2020 21:12 |
| 150 | Fonctionnalités | Nouveau | Normal | Engins de données 1 | Jean-Michel Lasnier | 15/09/2020 21:07 |
| 149 | Fonctionnalités | Nouveau | Normal | Raffinement de la prédiction des données | Dalyna Pak | 15/09/2020 21:03 |
| 148 | Fonctionnalités | Nouveau | Normal | Gestion de la perte d'activité d'un engin | Kuchue Kamgain-Djoko | 15/09/2020 21:01 |
| 147 | Fonctionnalités | Nouveau | Normal | Application PC | William Balea | 15/09/2020 20:56 |
| 146 | Fonctionnalités | Nouveau | Normal | Tests sur l'application Android | Cyrille Talla-Fongang | 15/09/2020 20:49 |
| 145 | Fonctionnalités | Nouveau | Normal | Changement de mot de passe | William Balea | 23/09/2020 20:51 |
| 144 | Fonctionnalités | Nouveau | Normal | Visualisation d'une station sur une carte | Dalyna Pak | 15/09/2020 20:54 |
| 143 | Fonctionnalités | Nouveau | Normal | Réception des logs des engins de données complétée | William Balea | 23/09/2020 20:51 |
| 142 | Fonctionnalités | Nouveau | Normal | Gestion des engins de données | Jean-Michel Lasnier | 15/09/2020 20:54 |
| 141 | Fonctionnalités | Nouveau | Normal | Interface usager | William Balea | 23/09/2020 20:52 |
| 140 | Fonctionnalités | Nouveau | Normal | Vérification du changement de mot de passe | Dalyna Pak | 15/09/2020 20:53 |
| 139 | Fonctionnalités | Nouveau | Normal | Vérification de la perte d'un engin de données | Dalyna Pak | 15/09/2020 20:53 |
| 138 | Fonctionnalités | Nouveau | Normal | Consultation du sondage | Kuchue Kamgain-Djoko | 15/09/2020 20:51 |
| 137 | Fonctionnalités | Nouveau | Normal | Connectivité avec les serveur | Cyrille Talla-Fongang | 15/09/2020 20:39 |
| 136 | Fonctionnalités | Nouveau | Normal | Serveur web | Dalyna Pak | 15/09/2020 20:53 |
| 135 | Fonctionnalités | Nouveau | Normal | Application Android | Kuchue Kamgain-Djoko | 15/09/2020 20:34 |
| 134 | Fonctionnalités | Nouveau | Normal | Création de conteneurs Docker pour les engins | Dalyna Pak | 15/09/2020 21:06 |
| 133 | Fonctionnalités | Nouveau | Normal | Gestion des statistiques de prédiction | Jean-Michel Lasnier | 15/09/2020 21:05 |
| 132 | Fonctionnalités | Nouveau | Normal | Affichages des prédictions | Cyrille Talla-Fongang | 23/09/2020 19:28 |
| 131 | Fonctionnalités | Nouveau | Normal | Application PC | Kuchue Kamgain-Djoko | 15/09/2020 20:55 |
| 130 | Fonctionnalités | Nouveau | Normal | Production de messages de logs | Jean-Michel Lasnier | 15/09/2020 21:05 |
| 129 | Fonctionnalités | Nouveau | Normal | Affichages des données brutes | Cyrille Talla-Fongang | 15/09/2020 20:52 |
| 127 | Fonctionnalités | Nouveau | Normal | Affichage des statistiques | Kuchue Kamgain-Djoko | 15/09/2020 20:52 |
| 126 | Fonctionnalités | Nouveau | Normal | Affichage liste des stations | Cyrille Talla-Fongang | 15/09/2020 20:52 |
| 125 | Fonctionnalités | Nouveau | Normal | Confirmation adresse IP serveur | Kuchue Kamgain-Djoko | 15/09/2020 20:52 |
| 124 | Fonctionnalités | Nouveau | Normal | Application Android | Cyrille Talla-Fongang | 15/09/2020 20:52 |
| 122 | Fonctionnalités | Nouveau | Normal | Configuration de Docker compose | William Balea | 15/09/2020 20:51 |
| 121 | Fonctionnalités | Nouveau | Normal | Configuration de Traefik | William Balea | 15/09/2020 20:51 |
| 120 | Fonctionnalités | Nouveau | Normal | Gestion du sondage dans le serveur | Dalyna Pak | 15/09/2020 20:51 |
| 119 | Fonctionnalités | Nouveau | Normal | Serveur web | William Balea | 15/09/2020 20:51 |
| 118 | Fonctionnalités | Résolu | Normal | Reproduction de l'application python | Jean-Michel Lasnier | 23/09/2020 19:20 |
| 117 | Fonctionnalités | Résolu | Normal | Création du serveur web central | William Balea | 23/09/2020 19:17 |
| 116 | Fonctionnalités | Résolu | Normal | Création de l'application Android | Cyrille Talla-Fongang | 23/09/2020 19:16 |
| 115 | Fonctionnalités | Résolu | Normal | Serveur dans un conteneur Docker | William Balea | 23/09/2020 19:19 |
| 114 | Fonctionnalités | Résolu | Normal | Communication de base entre l'app Android et serveur web | Kuchue Kamgain-Djoko | 23/09/2020 19:18 |
| 113 | Fonctionnalités | Résolu | Normal | Création d'un prototype | Jean-Michel Lasnier | 23/09/2020 19:20 |
| 112 | Fonctionnalités | Résolu | Normal | Finalisation de la réponse de l'appel d'offre | Dalyna Pak | 23/09/2020 19:14 |
| 110 | Fonctionnalités | Résolu | Normal | Planification du projet | Cyrille Talla-Fongang | 23/09/2020 19:14 |

*Figure 6: Répartition des tâches équitable*

## 4.3 Calendrier de projet (Q3.3)

*Tableau 3: Dates cibles de terminaison des phases importantes*

| *Phases importantes* | *Date cible* |
|---|---|
| *Réunion de lancement du projet* | *Mardi 15 septembre 2020* |
| *Mise en place des outils de gestion du projet (calendrier, configuration de git...)* | *Jeudi 17 septembre 2020* |
| *Soumission d'une réponse à l'appel d'offres et démonstration d'un prototype* | *Jeudi 24 septembre 2020* |
| *Livrable intermédiaire* | *Jeudi 29 octobre 2020* |
| *Livrable final* | *Mardi 1er décembre 2020* |

## 4.4 Ressources humaines du projet

La mise en œuvre de ce projet requiert deux principaux rôles ou types de ressources humaines à savoir un coordonnateur (ou Scrum master) et des développeurs-analystes. Quantitativement parlant, notre équipe de projet comprend un coordonnateur de projet et quatre développeurs-analystes. De l'analyse des exigences du projet, il ressort un besoin des compétences suivantes:

*Tableau 4 : Compétences nécessaires pour le projet*

| Compétences du Coordonnateur | Compétence du Développeur-analyste |
|---|---|
| **Habiletés techniques**<br>• Maîtrise des outils de gestion de projet informatique (Redmine, Jira, Git…)<br>• Démontrer un esprit d'analyse et de synthèse<br>• Démontrer un esprit de leadership au sein de l'équipe<br>• Capacité à prendre des décisions et à déléguer des tâches<br>• Capable de gérer et de s'adapter durant des situations stressantes<br>• Proposer des solutions gagnantes<br>**Habiletés personnelles**<br>• Leadership<br>• Bonne communication interpersonnelle<br>• Sens de l'organisation et de la planification | **Habiletés techniques**<br>• Maîtrise du langage de programmation C/C++<br>• Maîtrise de la programmation orientée objet<br>• Connaissance du langage de programmation Java/Kotlin<br>• Connaissance de Docker<br>• Connaissance du protocole HTTP<br>• Connaissance en Web service REST<br>• Expérience avec des logiciels de gestion de projet (Jira)<br>• Connaissance de logiciels de gestion de base de données (MySQL, postgreSQL, etc.)<br>• Connaissance des systèmes d'exploitation: Linux, Windows<br>• Expérience dans le développement d'application mobile<br>• Expérience avec l'IDE Android studio<br>• Expérience en Web Design<br>**Habiletés personnelles**<br>• Autonomie<br>• Esprit d'équipe<br>• Minutieux et rigoureux<br>• Ponctualité |

Dans le cadre de notre projet, le coordonnateur devra assumer également le rôle de développeur-analyste. Il devra donc, fournir un peu plus d'efforts que ceux-ci au niveau de l'organisation de l'équipe et du projet. Voici donc les responsabilités supplémentaires qui sont les siennes:

- Organiser la réunion de lancement du projet;
- Diriger l'équipe de projet et coordonner les différentes activités qui s'y rattachent;
- Établir un plan d'exécution et un échéancier pour chaque sprint et évaluer l'atteinte des objectifs;
- Surveiller la progression du travail afin d'en assurer l'achèvement dans les délais.

Le tableau 5 ci-dessous présente les **expériences personnelles** de chacun des membres de l'équipe acquises soit dans le cadre d'un stage ou des projets intégrateurs antérieurs:

*Tableau 5 : Expériences personnelles des membres de l'équipe de projet*

| | Cyrille | Dalyna | Jean-Michel | Kuchue | William |
|---|---|---|---|---|---|
| Connaissance des méthodes agile (Scrum) | x | x | x | x | x |
| Programmation C++/C | x | x | x | x | x |
| Développement d'une application mobile sur Android studio avec Java/ Kotlin | | | | x | |
| Développement d'une application web | x | x | x | x | x |
| Engins de données / Intelligence artificielle | | | | | |
| Développement d'un serveur web | x | x | x | x | x |
| Connaissance du protocole HTTP | | | | | |
| Connaissance des conteneurs Docker | | | | x | |
| Connaissance des logiciels de gestion de base de données (MySQL, PostgreSQL | x | x | x | x | x |
| Connaissance du système d'exploitation Linux et Windows | x | x | x | x | x |
| Connaissance des logiciels de gestion de projet (Trello) | x | x | x | x | x |

# 5. Suivi de projet et contrôle

## *5.1 Contrôle de la qualité*

Le contrôle de la qualité de notre solution se fera tout au long du projet (toutes les phases) en s'appuyant sur une liste de vérification, des lignes directrices de développement et de mécanismes de programmation en binôme ou de révision en groupe.

- La liste de vérification établie dès le début du projet est constituée de l'ensemble des exigences pour chaque composante attendue de la solution. Elle sera utilisée conjointement avec notre calendrier et la planification Gantt pour s'assurer de leur conformité avant la remise de chaque livrable.

- Les lignes directrices définies dès le début permettront de s'assurer du respect des recommandations ou bonnes pratiques en matière de développement logiciel. Quelques lignes directrices sont les suivantes:
  i) Les noms des variables et fichiers doivent être en CamelCase.
  ii) Les constantes sont en lettres majuscules.
  iii) La longueur d'une ligne doit être un maximum de 140 caractères.

- Les mécanismes de révision permettront de s'assurer de la qualité de notre solution. Le premier mécanisme qui prend en compte le niveau de compétence technique des membres de notre équipe reposera sur la notion de programmation en binôme à mettre en œuvre autant que possible. Elle peut paraître parfois lente, mais elle s'avère toutefois efficace dans des contextes similaires au nôtre.
  Le second sera celui de la révision collective un ou deux jours avant la date de remise de chaque livrable. Elle consistera à passer en revue le livrable (fonctionnalités attendues, assurance qualité) pour s'assurer de la qualité de ce dernier.
  Le troisième mécanisme est celui de l'entraide au sein de l'équipe. Le principe étant de solliciter le plus tôt possible l'aide de tout membre de l'équipe en cas de difficulté.

## *5.2 Gestion de risque (Q2.6 et 11.3)*

Il est évident que durant le cheminement du projet, notre équipe rencontrera plusieurs risques. En effet, ceux-ci serviront à tester l'habileté de notre équipe à faire face à des situations imprévues et stressantes. Afin de ne pas être pris par l'élément de surprise, nous avons pris la peine d'énumérer des risques que nous pourrions probablement rencontrer lors de la réalisation du projet et les solutions à ceux-ci.

- Tout d'abord, il y a les risques concernant les retards de remise lors d'une échéance. En effet, en négligeant l'importance de la communication au sein de notre équipe, il est possible d'ignorer sans le vouloir les difficultés à progresser de l'un de nos coéquipiers. Afin de remédier à cette situation, nous comptons faire

des réunions Scrum hebdomadaires afin de s'assurer de la progression uniforme de tous les membres de groupe. Durant ces rencontres chaque membre de l'équipe devra expliquer ce qu'il a réalisé depuis la dernière rencontre Scrum et sur quoi il compte travailler à partir d'aujourd'hui. Ce risque est d'une très grande importance, car il pourrait rendre le client mécontent et augmenter les coûts du projet. Le Scrum master aura donc un grand rôle à jouer afin qu'on ne rencontre pas de retard.

- Ensuite, il est possible qu'un membre de l'équipe réalise une fonctionnalité qui ne correspond pas du tout aux attentes du client. En effet, les directives et requêtes du document "exigence technique" peuvent parfois être ambiguës ou mal comprises et mener donc à des résultats différents de ce qui était demandé. Afin de remédier à cette situation, notre équipe va s'assurer de clarifier toutes les interrogations avec le client en cas d'incompréhension d'une exigence. Par la suite, afin de nous assurer que tout le monde ait bien compris ses tâches, nous ferons un tour de table où tout le monde expliquera en détail ce qu'il doit réaliser. Ce risque est d'une très grande importance, car elle pourrait amener d'autres problèmes comme un retard sur l'échéance, l'insatisfaction du client ou une dispute au sein du groupe.

- Par ailleurs, il faudrait prendre en compte la possibilité que l'un des membres de l'équipe décide d'abandonner le projet ou de s'absenter pour plusieurs jours sans donner de nouvelle. Afin d'éviter ce risque, nous nous sommes d'abord assurés que tous les membres de l'équipe avaient l'intention de rester jusqu'à la fin du projet. Ensuite, nous nous sommes partagé nos contacts Facebook afin de pouvoir rester en communication en tout temps en cas de besoin. Donc, si une personne à une situation d'urgence et qu'il ne pourra pas être présent lors d'un événement important (réunion Scrum), il a le devoir de le faire savoir au reste de son équipe. Ce risque n'est pas vraiment contrôlable, mais grâce à une bonne communication on peut parvenir à réduire ces dégâts. Celui-ci est d'une importance capitale, car la disparition d'un membre de l'équipe peut ralentir tout le processus du projet et créer une grande situation de stress au sein de l'équipe.

## *5.3 Tests (Q4.4)*

Nous avons 4 sous-systèmes principaux à tester pour ce projet. Voici les tests que nous ferons pour chaque composante:

- Serveur
  - Pour le serveur, nous ferons des tests unitaires en utilisant la librairie Autopkgtest. Nous allons principalement tester les requêtes HTTP en amont et en aval en C++.
- Application PC:
  - Puisque nous allons utiliser principalement la librairie Qt pour développer l'interface utilisateur de l'application PC, nous nous servirons de la librairie "Qt Test" pour effectuer des tests unitaires.

- Engins de données:
  - En ce qui concerne l'engin de données 3, celui qui utilise l'intelligence artificielle, nous ferons des tests de distribution sur les arbres en donnant des valeurs en entrée et en vérifiant la réponse de sortie. Puisque ce sont des arbres de décisions et non de l'apprentissage automatique, nous pouvons attendre des résultats de vote bien précis à répétition de la forêt.
  - Pour les autres engins de données, nous ferons des tests unitaires avec des requêtes de données et ensuite vérifions les résultats.

- Application Android:
  - Pour le système Android, nous ferons des tests de la couche vue-modèle en utilisant la librairie de test "Test" fournie par Jetpack de Google.

Finalement, nous effectuerons des tests d'intégration pour nous assurer de la bonne connectivité entre tous les sous-systèmes. Nous ferons des tests (bout en bout) en boîte blanche pour s'assurer du bon fonctionnement général du projet. Nous ferons aussi des tests en boîte noire pour tester l'UX de notre application et les autres fonctionnalités pertinentes.

## *5.4 Gestion de configuration*

Pour le système de contrôle de versions, nous allons utiliser Git. Ce logiciel nous permettra de décentraliser le travail afin que tout le monde puisse avoir une copie complète du projet. De plus, ce système nous permet de gérer les différentes versions des applications et de pouvoir revenir sur des versions antérieures ou créer de nouvelles versions, sans affecter le projet présent. Lors de la publication d'une nouvelle version, au moins une autre personne de l'équipe devra vérifier que le code compile et ne brise pas d'autres fonctionnalités déjà complétées. Ainsi, tout le code source utilisé pour le projet sera stocké sur le Git de l'école qui n'est pas un répertoire public. Pour organiser le tout, chaque partie du projet aura sa propre branche : le serveur web, l'application Android et PC ainsi que les engins.

Comme mentionné à la section précédente, des tests unitaires seront utilisés pour s'assurer que les fonctionnalités ne brisent pas en cours de route.

Toutes les données recueillies des utilisateurs seront stockées sur MySQL et seront disponibles pour le serveur. Pour les données de BIXI, elles seront téléchargées de kaggle.com et stockées dans le même répertoire que les fichiers des engins.

Pour la documentation relative au code source et la conception, un fichier Readme.md sera créé un maintenu tout au long du développement. L'équipe s'assurera de mettre à jour la documentation au moins à chaque livrable afin de bien décrire les fonctionnalités et l'utilisation des logiciels.

# 6. Références (Q3.2)


1. Oktal. (2020). Pistache An elegant C++ REST framework. Disponible en ligne http://pistache.io/

2. Oracle. (2020). MySQL. Disponible en ligne https://dev.mysql.com/doc/

3. Traefik Labs. (2020). Traefik Proxy. Disponible en ligne https://doc.traefik.io/traefik/

4. Développeurs Android. (2020). Guide to app architecture. Disponible en ligne https://developer.android.com/jetpack/guide#common-principles

5. Design. (2020). Create intuitive and beautiful products with Material Design. Disponible en ligne à : https://material.io/design

6. Waytolearnx.com. (Juin 2020). Différence entre MVC et MVVM. Disponible en ligne: https://waytolearnx.com/2020/06/difference-entre-mvc-et-mvvm.html

7. Koehrsen, Will. (2017, Dec 27). *Random Forest in Python.* towards data science. Disponible en ligne: https://towardsdatascience.com/random-forest-in-python-24d0893d51c0

8. Collin J. (Septembre 2020). Notes de cours INF3995, Cycles d'acquisition et ententes contractuelles


# Annexes