République Tunisienne
Ministère de l'enseignement supérieur
et de la recherche scientifique
Université de Tunis ElManar

Département des Technologies de l'Information et de la Communication

Rapport de stage d'ingénieur
Groupe: 3ème année Télécommunications

Développement Fullstack en Spring et Ionic
Amélioration de l'application Bware

Réalisé par:

Wael ARFAOUI

Entreprise: **Poulina Group Holding**

Encadré par:

Sami BESSAIES

Année Universitaire : 2017-2018

Remerciements

C'est avec plaisir que Nous réservons cette page en signe de gratitude et de profonde reconnaissance à tous ceux qui nous ont aidé à réaliser ce projet.Nous tenons à remercier le jury pour l'honneur qu'il nous a fait pour avoir accepté de juger notre travail. Nous exprimons notre gratitude tout particulièrement à M.Sami Bessaies pour son encadrement, son aide, sa disponibilité, ses conseils fructueux et ses encouragements qu'il nous a prodigué tout au long de ce projet. Finalement, et avec beaucoup de respect, nous ne pouvons laisser cette occasion sans saluer chaleureusement tous nos enseignants de l'ENIT ainsi que nos collègues pour leur soutien moral.

Résumé

Le présent rapport décrit les différents taches effectuées tout au long de notre expérience comme stagiaire chez Poulina Group Holding. Durant ce stage de trois mois, Nous avons travaillé sur l'application Bware, application développée par Smartdata technology.
Cette application est un moyen de prévention contre les différents problèmes de santé. L'utilisateur peut surveiller tous les risques qui menacent sa santé. Que ce soit la Pollution, le Pollen, la Grippe, la Varicelle ou la Gastro-entérite, Bware" l'informe continuellement du niveau du risque.

Le travail demandé consistait à améliorer l'application en rajoutant des fonctionnalités pour les utilisateurs publiques et d'autres pour le suivi et le contrôle de l'application.

Mots clés : Santé, Prévention, épidémie, risque, pollution, REST API, Spring Fraework, Ionic framework.

Glossaire

API Application Programming Interface
CSS Cascading Style Sheet
ENIT Ecole Nationale d'Ingénieurs de Tunis
HTML Hypertext Markup Language
HTTP HyperText Transfer Protocol
JSON JavaScript Object Notation
REST Representational State Transfer
UML Unified Modeling Langage
URI Uniform Resource Identifier

Liste des figures

Liste des tableaux

Table des Matières

Introduction

Le monde entier vie sous le rythme d'une évolution et de croissance rapide mais cette croissance vient à un coût cher. Des centaines voire des milliers de menaces contre la santé et l'environnement apparaissent chaque jour.

Selon l'Organisation mondiale de la santé (OMS), 3 millions de personnes meurent tous les ans à cause de la pollution de l'air. En 2015, près de 19 milliards d'euros ont été consacrés à des soins de santé lié à la qualité de l'air et 1,2 milliard de journées de travail ont été perdues. Outre les dégâts causés par la pollution de l'air, de nouvelles virus et épidémies apparaissent annuellement contre lesquels l'immunité humaine n'est pas encore développée, ce qui résulte enfin à l'apparition des maladies et des réactions allergiques parfois mortelles.[**1**]

Pour lutter contre les risques présentés ci-dessus la plupart des personnes a recourt aux moyens classiques comme les masques, les gants et les cache-cols pour se protéger contre les différents agents contagieux.

L'inconvénient majeur de ces moyens est que les individus sont incapables de savoir au paravent le niveau de pollution ou même l'existence d'un éventuel risque à leur proximité. Et prenant en compte que la météo ne s'intéresse généralement qu'à l'état climatique, les personnes allergiques et fragiles n'ont aucun moyen pour s'informer sur l'état de l'air environnant.

La présence d'un moyen d'information sur la qualité quotidienne de l'air s'avère nécessaire surtout pour les personnes fragiles, asthmatiques ou atteintes par une allergie. Un tel outil indiquera le niveau de pollution de l'air en substances nocives, en bactéries et virus ou en pollen. La solution permettra aussi aux utilisateurs de s'inscrire à un type de risque pour recevoir des notifications de suivi ou de conseil. Ainsi une personne inscrite à un risque donné peut suivre les pics de pollution et recevoir en même temps des conseils de protection contre le risque. La solution est une application fiable et pérenne qui permettra de prévenir les utilisateurs de la pollution. Cette dernière fait l'objet de ce rapport où je présente la méthodologie adoptée pour répondre à sa finalité.

Le présent rapport décrit les étapes par lesquelles nous avons passé pour finaliser notre projet. Au début, nous présenterons le contexte général de notre application. Dans une deuxième étape, nous analyserons les besoins fonctionnels et non fonctionnels. Le troisième chapitre est consacré aux éléments de conception sur les niveaux : global et détaillé. Enfin nous avons dédié le dernier chapitre pour la réalisation dans laquelle nous présenterons les outils et l'environnement logiciel et matériel utilisés afin de réaliser notre application et nous expliquerons les interfaces de notre projet.

Chapter 1

Présentation générale du projet

Introduction

Nous consacrerons le présent chapitre à la présentation du contexte général de notre Projet, commençant par l'explication de cadre général de projet et son objectif, puis nous attaquerons la partie liée à l'étude de l'existant. Finalement une clarification sur la méthodologie de travail sera entamée.

1.1 Cadre du projet

Durant notre formation d'ingénieur nous sommes menés à effectuer un stage d'été d'au moins deux mois. Le présent stage a été élaboré à Poulina Group Holding et il consiste à améliorer l'application Bware développée par Smartdata technology.

1.1.1 Entreprise d'accueil

Le stage a été effectué au sein de Aster Training filiale du groupe Poulina.

1.1.1.1 Poulina Group Holding

Poulina est un groupe industriel et de services tunisien. Originellement spécialisé dans l'agriculture, le groupe s'est peu à peu diversifié pour devenir le premier groupe à capitaux privés du pays.

Le groupe s'est organisé au fil des années autour d'une dizaine de métiers regroupant les pôles d'activités de la holding : immobilier, travaux publics, bois et biens d'équipement, produits de grande consommation, emballage, transformation de l'acier, commerce et services, matériaux de construction et intégration avicole.

1.1.1.2 Aster Training

La société Aster training est une filiale du groupe Poulina et son capital social s'élève à 145000 DT. Créée en 1996, la société Aster training assure les activités suivantes :

- Organise des séminaires de formation en gestion, technique et informatique.
- Ingénierie informatique.
- Étude, conception et développements des logiciels informatiques.

- Étude et mise en place des réseaux informatiques.
- Infogérance.
- Audit et conseil informatiques.

1.1.2 Objectif du projet

Le but du projet est d'apporter des améliorations à l'application Bware développée par l'entreprise française Smart Data Technology. Ces amélioration toucheront la partie destinées aux clients finales, la partie serveur ainsi que la partie destinée au suivi et contrôle de l'application.

1.2 Études de l'existant

De nos jours, nous témoignons une expansion du marché des applications mobiles. Parmi ces applications on note la présence d'une catégorie destinée à la santé humaine avec un pourcentage de 1% du marché des applications et qui voit un accroissement d'à peu près 50% chaque année. Gardant à l'esprit la nécessité cruciale de ces applications pour certains utilisateurs que ce soit pour des raisons de suivi de leur performance physique ou leur état sanitaire (tension, glycémie, température interne ...), nous déduirons en que la catégorie santé et fitness est un marché très important dans le domaine des applications mobiles. Notre intérêt est d'étudier les applications de prévention contre la pollution. Ainsi plusieurs solutions s'offrent aux utilisateurs des mobiles garantissant le suivi du niveau de la pollution à savoir Air Visual, Plume Air Report.

1.2.1 Analyse de l'existant

Les solutions présentées ci-dessus ne couvrent pas l'intégralité des fonctionnalités nécessaires pour les utilisateurs mais elles en offrent d'autres telles que le partage de l'état de l'air dans les réseaux sociaux et la couverture des villes de partout dans le monde. Parmi les fonctionnalités absentes, nous citons :

- Les notifications pour Air Visual
- Les détails sur les différents type de pollution pour Plume Air Report
- La présence de la carte

1.2.2 Critique de l'existant

Les exemples cités précédemment ne permettent d'intégrer qu'une partie des fonctionnalités nécessaires pour le suivi du niveaux de la pollution dans des zones éparpillées dans le monde en raison d'une quarantaine de villes par pays. Nous citons aussi d'autres fonctionnalités absentes chez les concurrents telles que les notifications, les conseils et les détails sur les différents type de pollution.

1.2.3 Solution proposée

Notre solution consiste à couvrir les déficiences des applications présentées en mettant en place une application mobile multiplate-forme destinée aux utilisateurs français qui couvrira la totalité du territoire français mais qui permettra aussi de prévenir les utilisateurs des différents risques auxquels ils sont inscrits, d'afficher des détails sur les agents nocifs et des conseils concernant les niveaux de risque élevés.

1.3 Méthodologie de travail

Pour mener à bien notre projet nous avons opté pour une méthodologie afin d'organiser les différentes phases du projet.

Alors nous avons adopté l'approche agile qui consiste à un paradigme révolutionnaire dans le monde de gestion de projet. Cet approche diffère des méthodes classiques par la migration du centre d'intérêt des étapes par lesquels passe le projet vers les étapes par lesquels passe le produit. Ainsi, cet approche élimine l'effet tunnel pour donner davantage de visibilité et de transparence en impliquant le client dans tous les procès de la réalisation. Ce qui résulte en des majeurs changements sur tout le processus de réalisation qui devient à la fois itératif et incrémental ce qui permettra une meilleure flexibilité aux niveaux de la spécification des besoins mais en gardant un minimum de règles.

1.3.1 Méthode Scrum

Au sein de ce cadre méthodologique de gestion de projet, les acteurs ajustent empiriquement, au fil des itérations, leur propre méthode en fonction de leur contexte. On peut qualifier Scrum de simple, pragmatique, transparent et empirique. Scrum ne couvrant que les aspects de gestion de projet, c'est souvent la méthodeeXtreme Programming (XP) qui vient compléter le vide laissé en matière de pratiques de développement. XP apporte ainsi les pratiques de programmation en binôme, de développement piloté par les tests (TDD ou Test Driven Development), intégration continue, etc.
Le cycle préconisé par TDD comporte cinq étapes [**2**].

- Écrire un premier test ;
- Vérifier qu'il échoue (car le code qu'il teste n'existe pas), afin de vérifier que le test est valide ;
- Écrire juste le code suffisant pour passer le test ;
- Vérifier que le test passe ;
- Puis ré-usiner le code, c'est-à-dire l'améliorer tout en gardant les mêmes fonctionnalités

Conclusion

Dans ce chapitre, nous avons présenté le cadre général de notre stage en énonçant notre entreprise d'accueil et les objectifs envisagés de ce stage. Nous nous sommes intéressés, par la suite, à l'analyse et la critique des solutions similaires déjà existantes dans le marché et à la proposition de notre solution alternative.

Chapter 2

Analyse et spécification des besoins

Introduction

Après avoir présenté le contexte général de notre projet, nous venons d'énoncer dans cette partie les besoins auxquels notre application devra réponde. Nous énumérerons les besoins fonctionnels et non fonctionnels dans une première partie et nous exposerons par la suite la spécification de ces besoins à travers les diagrammes de cas d'utilisation.

2.1 Analyse des besoins

Ce chapitre est dédié à l'étude profonde des différents besoins sur l'échelle fonctionnelle et non fonctionnelle. Nous projetons de détailler les risques et contraintes auxquelles est soumis notre projet dans le but de répondre aux besoins.

2.1.1 Identification des acteurs

L'application doit garantir la prévention contre les risques de pollution pour l'utilisateur final et la suivi et contrôle pour les administrateurs de Smartdata.

2.1.2 Besoins fonctionnels

Nous présentons ci-dessous les besoins fonctionnels regroupés par acteur:

- Utilisateur
  - Visualisation des niveaux des risques
  - Inscription aux différents risques
  - Visualisation des détails sur les risques
  - Visualisation des états des stations
  - Visualisation des conseils
  - Inscription aux rappels santé
  - Ajout d'astuces
- Administrateur
  - Consultation des statistiques
  - Réception des Emails concernant les batchs
  - Réception des Emails concernant les notifications

2.1.3 Besoins non fonctionnels

Pour bien répondre à ses besoins notre application doit respecter les exigences techniques suivantes:

Ergonomie Le but de l'application est qu'elle soit exploitée par des simples utilisateurs qui ne sont pas forcément expérimentés dans le domaine de l'informatique, et qu'elle le soit de la façon la plus efficace. Les interfaces de l'application doivent être ergonomiques et conviviales.

Environnement et architecture L'application doit être mise en place dans une architecture répartie multi-tiers et accessible via le Web à travers les navigateurs Web.

Rapidité Notre application doit garantir un accès rapide aux données et d'une manière transparente.

Scalabilité L'application doit être facilement extensible pour pouvoir accueillir un public plus large.

Documentation L'application sera livrée avec une documentation des différentes fonctions et API qu'elle offre, nécessaire pour les futurs développeur

2.2 Spécification des besoins

Dans ce paragraphe, nous allons préciser les différents besoins de notre application en ayant recours au langage de modélisations UML. Nous illustrons les besoins par les diagrammes de cas d'utilisation.

2.2.1 Diagramme de cas d'utilisation global

Nous présentons dans cette partie le diagramme de cas d'utilisation global de notre application. La figure 2.1 représente le diagramme de cas d'utilisation global.

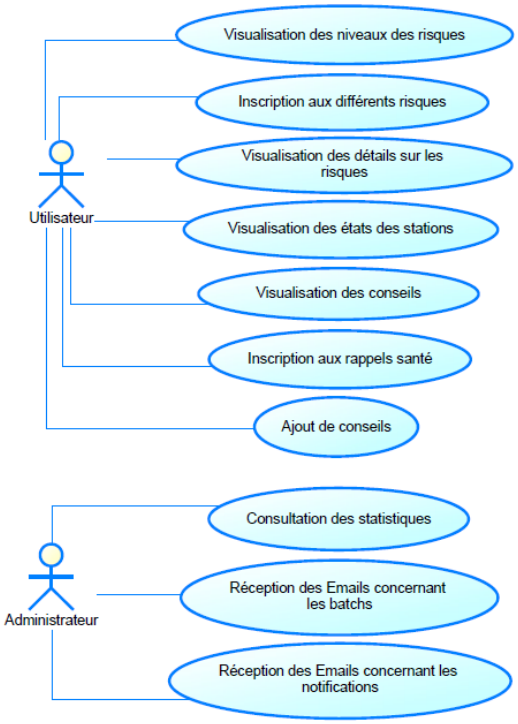

Figure 2.1: Diagramme des cas d'utilisation global

2.2.2 Raffinement des cas d'utilisation

Dans cette partie, nous présentons les diagrammes de cas d'utilisation raffinés. Nous détaillerons quelques cas d'utilisation parmi ceux qui sont précédemment présentés.

2.2.2.1 Cas d'utilisation : Ajouter astuces

La figure 2.2 représente le diagramme de cas d'utilisation de l'ajout des astuces.

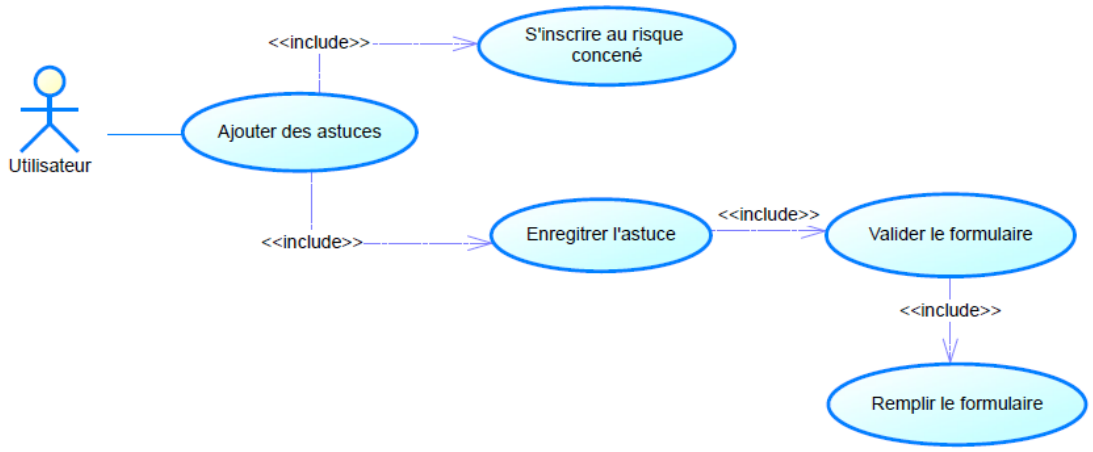

Figure 2.2: Cas d'utilisation : Ajouter des astuces

Ci-dessous une description sous forme de tableau du cas d'utilisation : Ajouter astuce.

Table 2.1: Description textuelle du cas d'utilisation : Ajouter astuce

Cas d'utilisation	Ajouter astuce
Objectif	Permettre d'ajouter des astuces
Acteur	Utilisateur
Pré-conditions	S'inscrire, remplir et valider le formulaire
Post-condition	Astuce ajoutée
Scénarios nominaux	**S1. Soumettre une astuce** L'utilisateur remplie le formulaire d'ajout d'astuces, choisit le type de risque, saisit son adresse électronique enfin il confirme l'ajout par cliquer sur le bouton enregistrer
Exceptions	**E1** Utilisateur non inscrit au risque. **E2** L'un des champs du formulaire est soit vide soit non valide.

2.2.2.2 Cas d'utilisation : visualisation des niveaux des risques

Cette partie est consacrée au raffinement du cas d'utilisation : visualisation des niveaux des risques. La figure 2.3 représente en détail le cas d'utilisation avec son acteur et ses pré-conditions.

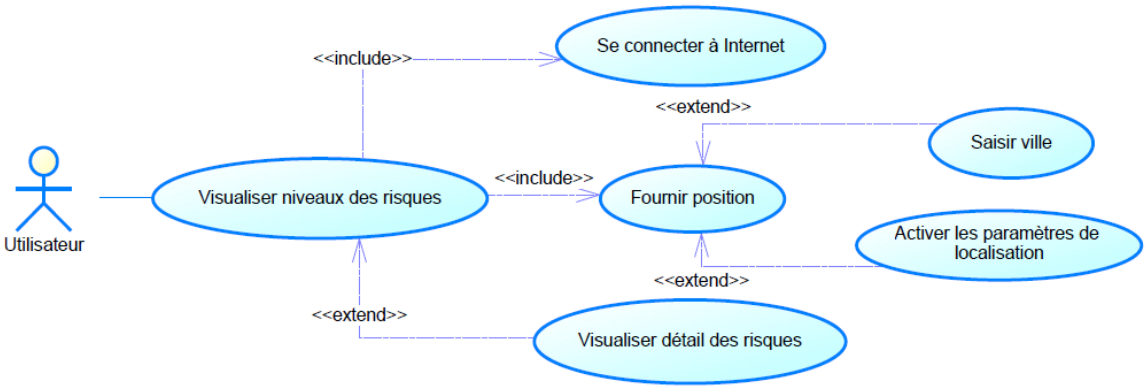

Figure 2.3: Cas d'utilisation : visualisation des niveaux des risques

Le tableau ci-après est une description textuelle du cas d'utilisation.

Table 2.2: Description textuelle du cas d'utilisation : visualiser les niveaux des risques

Cas d'utilisation	visualiser les niveaux des risques
Objectif	Permettre de consulter les niveaux des risques
Acteur	Utilisateur
Pré-conditions	S'inscrire, saisir ville, activer la géolocalisation, se connecter à Internet
Post-condition	Niveaux des risques consultés
Scénarios nominaux	**S1. Consulter le niveau des risques** L'utilisateur se connecte à internet et fournit la ville qu'il veut consulter en saisissant la ville ou en activant le système GPS. Il recevra ainsi le niveaux de risques de la ville souhaitée. **S2** Après avoir reçu les niveaux de risque, l'utilisateur peut consulter les détails sur chaque risque (polluant)
Exceptions	**E1** Utilisateur non inscrit au risque. **E2** Téléphone non connecté à Internet **E3** Ville non fourni et système GPS non activé

Conclusion

Nous avons consacré ce chapitre, à la spécification des besoins, ce qui permet de mieux comprendre les problématiques soulevées et d'avoir une vue globale sur les fonctionnalités constituantes notre projet. Nous nous intéressons dans le chapitre suivant aux éléments de conception basée sur le travail effectué dans le présent chapitre.

Chapter 3

Conception

Introduction

Dans ce chapitre, nous nous intéressons à la conception de notre projet. En effet la conception s'avère la pierre angulaire de tout projet appelant à l'ingénierie. En se basant sur la spécification détaillée des besoins, nous allons pouvoir définir tout les éléments faisant partie de la réalisation. Nous préciserons alors les éléments de la conception architecturale ainsi ceux de la conception détaillée.

3.1 Conception globale

Nonobstant la simplicité du besoin de notre application, ce dernier n'est pas aussi évident à réaliser car il fait appel à un certain niveau de complexité architecturale. Généralement notre application est composée de deux majeures parties à savoir : la partie Backend, responsable de tous les traitements possibles, la collecte des données et la réponse aux requêtes des clients, et la partie Frontend qui est la partie déployée chez les utilisateurs qui servira comme une Interface Homme Machine (IHM). La figure 3.1 illustre l'architecture de notre système. Nous y revenons en plus de détail par la suite.

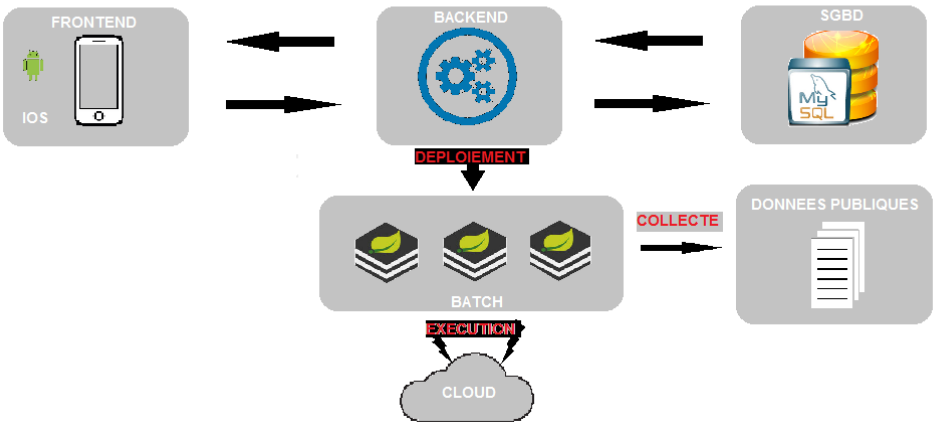

Figure 3.1: Architecture globale de l'application

La communication entre la partie front et back se fait à base de services Web REST (HTTP et JSON). En fait, la partie Frontend consomme les services web fournis par la partie Backend. (Voir figure 3.2)

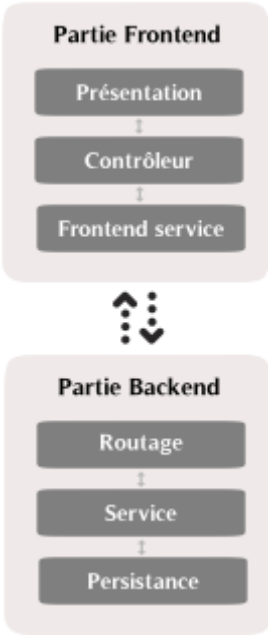

Figure 3.2: Architecture global de la couche Web

3.1.1 Partie Backend

Comme est mentionné précédemment, la partie backend est responsable de la majorité des traitements réalisés par notre application :

- Interrogation de la base de données,
- réponse aux requêtes des utilisateurs,
- envoie des notifications,
- envoie des Emails,
- collecte des données publiques,
- récupération des informations utiles

Elle composée par deux composantes à savoir le serveur HTTP et les applications Batch. Le serveur HTTP fournit des API consommables par la partie frontend par le biais du modèle architecturale REST qui permet la communication entre ces deux parties sur des sockets HTTP usant ainsi les méthodes du protocole HTTP. L'objectifs des applications Batch est d'exécuter des taches bloquantes qui nécessitent beaucoup de temps pour s'exécuter et sont indépendantes de toutes interventions humaines. Typiquement, les Batchs seront utilisés à :

- traiter des taches planifiées,
- traiter des taches volumineuses (lecture/écriture),

- traiter parallèlement des données...

Notre application utilise les Batch pour les cas d'utilisation suivants :

- Recueillir et traiter les données publiques
- Interroger les objets Netatmo
- Enregistrer les informations utiles dans la base de données
- Envoyer les notifications aux utilisateurs
- Suivre l'état du serveur Backend
- Mettre à jour le fichier log

De ce fait, les Batch sont lancés périodiquement ou selon la planification définie par des cron. Les cron sont des programmes qui permettent d'exécuter automatiquement des taches à une date et heure spécifiée à l'avance ou selon un cycle. Seulement le batch d'envoie des notifications est exécuté selon un algorithme paramétré par les niveaux quotidiens de risque.

3.1.2 Partie Frontend

La partie Frontend autrement désignée par la partie Client sera déployée sur les appareils des utilisateurs et usera les API offertes par la partie Backend pour récupérer les informations nécessaires au rendering des vues (view) de notre application.La partie client est aussi responsable des transitions entre les view. Ainsi à partir des templates préalablement définies la partie frontend crée les Interface Homme/Machine (IHM) accessibles par l'utilisateur afin de consulter les différentes fonctionnalités offertes par l'application. Outre la présentation et la création des IHM, la partie frontend fournit des services clients tels que l'accès au ressources de l'appareil téléphonique (Géolocalisation, Internet...) (voir figure 3.2).

3.2 Conception détaillée

Nous présentons dans cette partie la conception de notre application à travers le langage UML. Nous représenterons les diagrammes de modélisation UML.

3.2.1 Diagramme de paquetage

À fin d'assurer la modularité de notre système nous optons pour représenter les relations entre les différentes classes par le diagramme de paquetage. Le diagramme de paquetage assure aussi une meilleur lisibilité du diagramme des classes. Chaque module est alors représenté par un paquet selon le domaine fonctionnel comme le représente la figure 3.3.

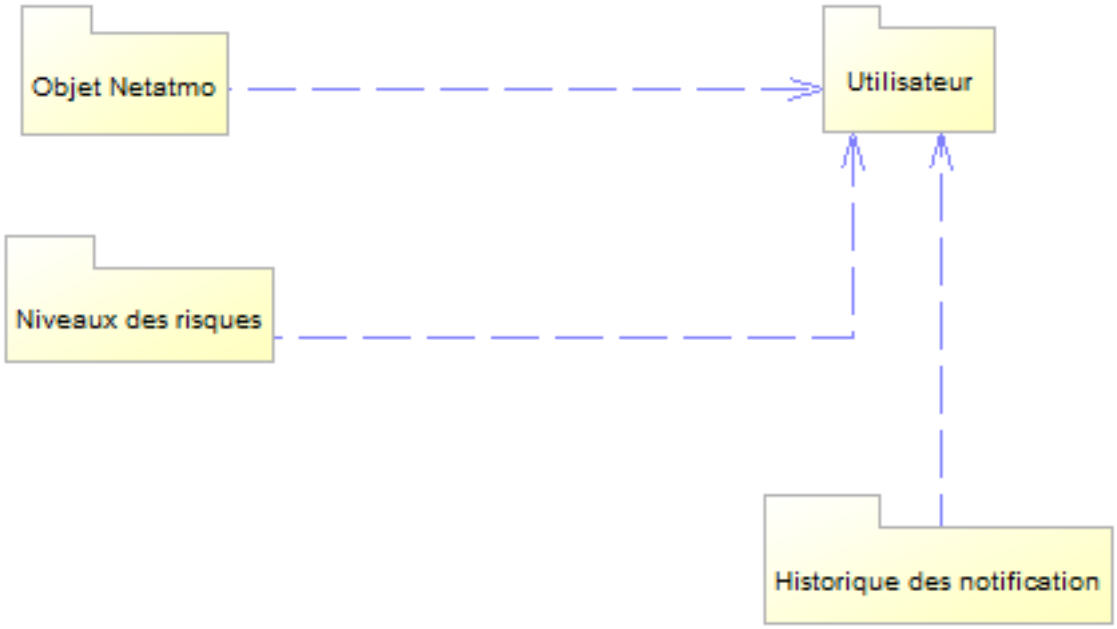

Figure 3.3: Diagramme de paquetage

3.2.2 Diagramme de classes

Arrivant à ce stade nous allons décomposer chaque paquet du diagramme de la figure 3.3 en diagramme de classes représentant ainsi les relations existante entre les classes de notre projet. Les relation entre les paquets sont représentées par les relations des classes qui les composent. Chaque classe est illustrée par ses attributs et les relations qui la relie aux classes de son paquet parent et des classes des autres paquets.

3.2.2.1 Diagramme de classes relatif au paquet : Utilisateur

Le diagramme de la figure 3.4 représente le modèle des classes du paquet : Utilisateur.

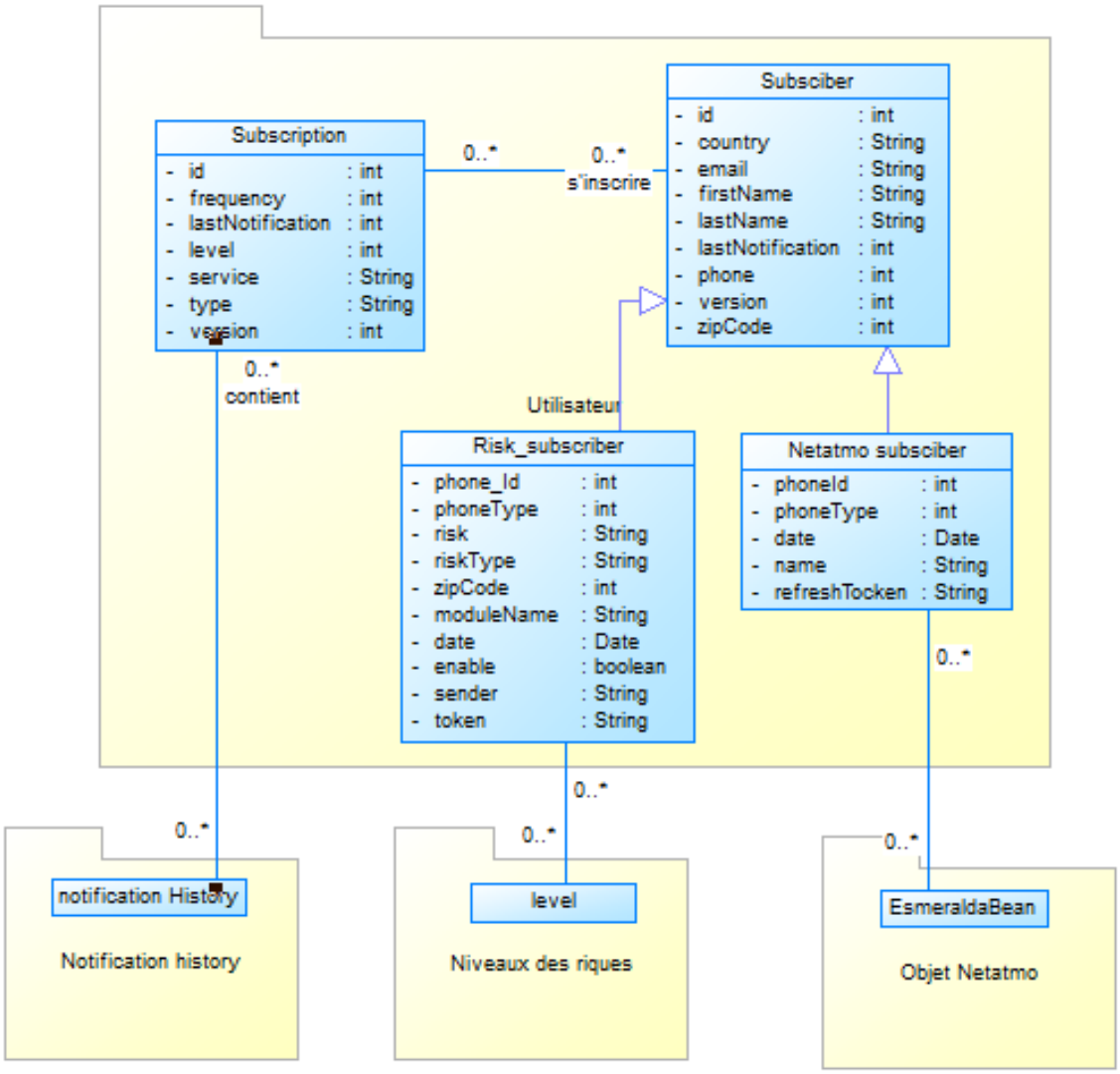

Figure 3.4: Diagramme de classes relatif au paquet : Utilisateur

3.2.2.2 Diagramme de classes relatif au paquet : Niveau des risques

Ci-après est représenté la décomposition du paquet Niveau des risques. La figure 3.5 illustre les classes membres du paquet : Niveau des risques.

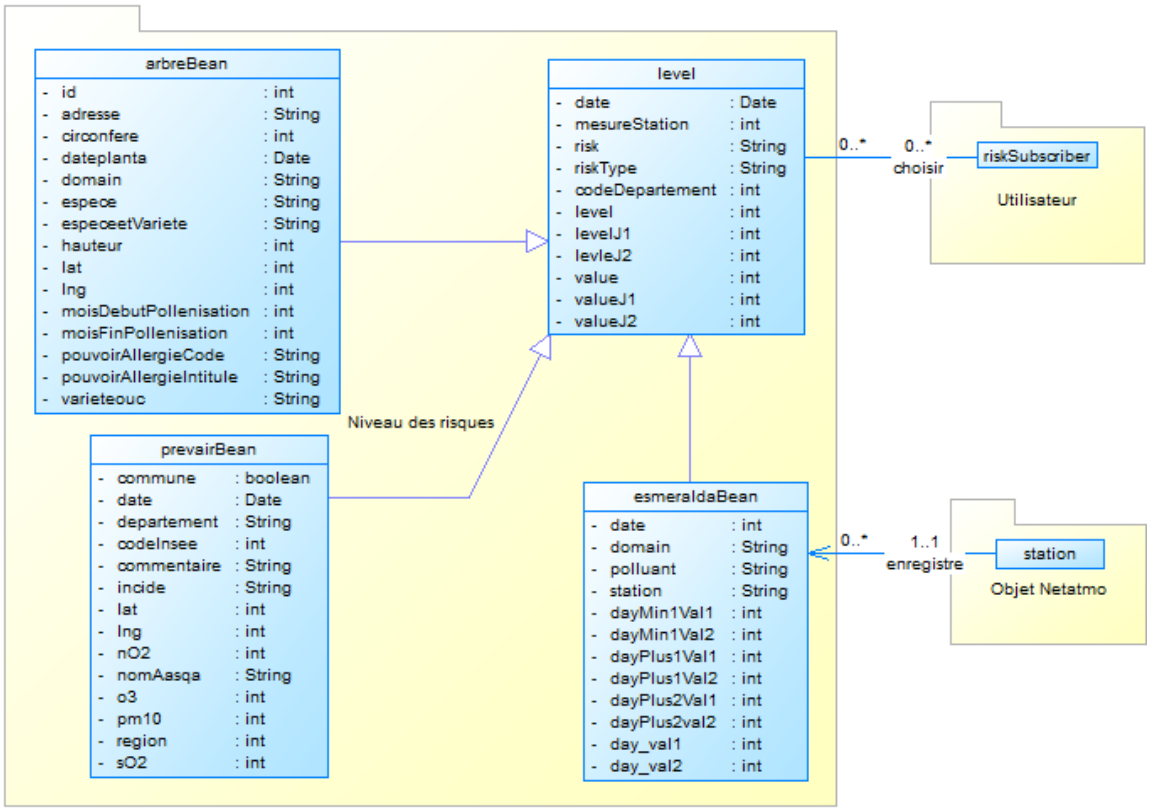

Figure 3.5: Diagramme de classes relatif au paquet : Niveau des risques

3.2.2.3 Diagramme de classes relatif au paquet : Objet Netatmo

La figure 3.6 illustre le diagramme de classes relatives au paquetage « Objet Netatmo ».

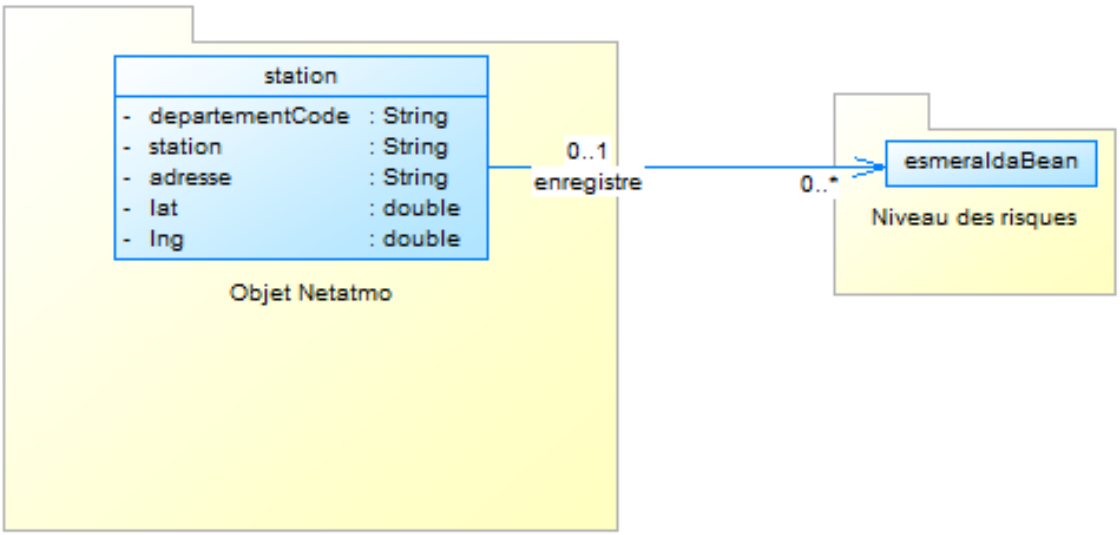

Figure 3.6: Diagramme de classes relatif au paquet : Objet Netatmo

3.2.2.4 Diagramme de classes relatif au paquet : Historique des notifications

Les classes du paquetage « Historique des notifications » sont modélisées dans la figure ci-dessous.

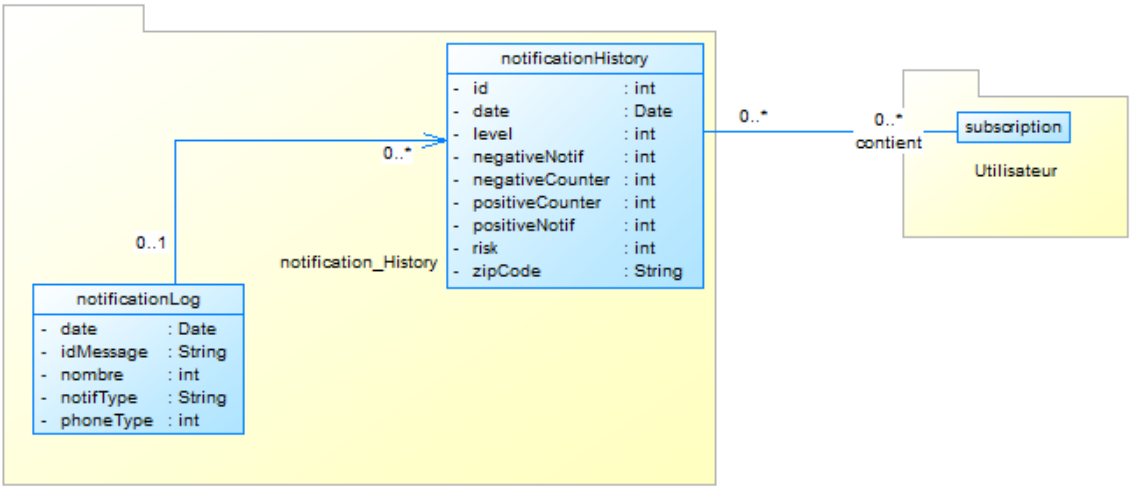

Figure 3.7: Diagramme de classes relatif au paquet : Historique des notifications

3.2.3 Digrammes de séquence

Les diagrammes de séquences permettent de donner une vue dynamique sur l'interaction entre l'acteur et le système. Nous représentons, dans cette section, des enchaînements de quelques scénarios à travers les diagrammes de séquences d'UML.

3.2.3.1 Diagramme de séquence "Ajouter astuce"

La figure 3.8 illustre le diagramme de séquence de la fonctionnalité "Ajouter astuce" dont l'acteur est l'utilisateur et la post-condition est l'ajout d'une astuce. L'ajout d'une astuce se fait à plusieurs étapes. Après avoir accéder à la page d'ajout l'utilisateur interagira avec l'IHM contenant les champs à remplir pour accomplir la tache. Il saisira l'astuce, ses cordonnées (Pseudonyme et mail) et choisira le type de risque. Une fois les champs remplis, il peut confirmer son ajout en appuyant sur le bouton "submit" qui actionnera toute la chaine d'exécution expliquée par le tableau 3.1.

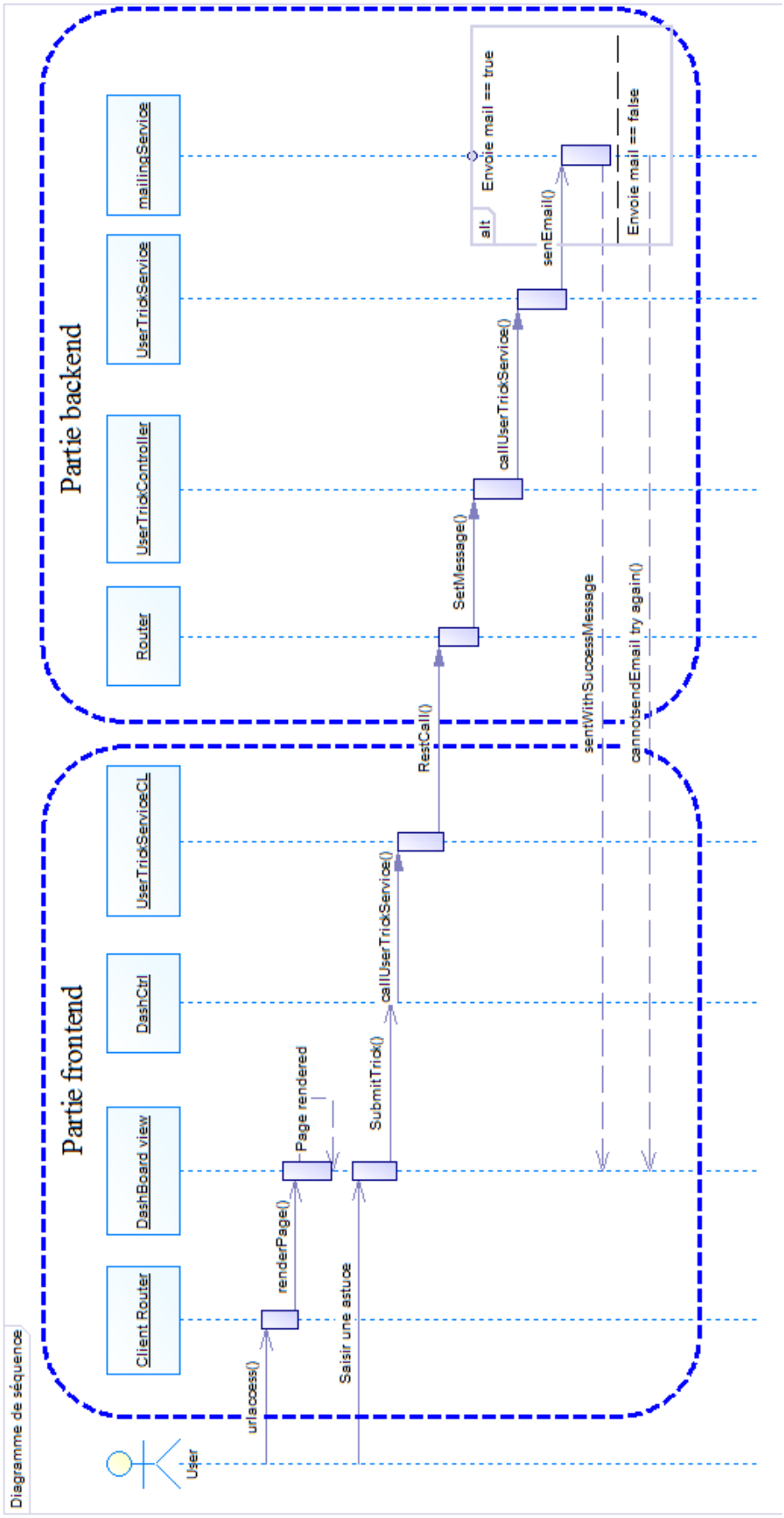

Figure 3.8: Diagramme de séquence "Ajouter astuce"

Table 3.1: Description textuelle de la chaine d'exécution de la fonctionnalité : Ajouter astuce

Fonction	**Emplacement**	**Action/Fonction appelante**	**Rôle**
submitTrick	DashCtrl	Appui sur le bouton "Submit"	-Vérifie le contrôle sur les champs -Appelle le UserTrickService
addUserTrick	UserTrickService	submitTrick	-Faire l'appel REST au serveur
setMessage	userTrickCon-troller	addUserTrick	-Appelle le service de constructionde message
buildMessage	userTrickService	setMessage	-Construire le message à envoyer pa mail
sendEmail	mailingServie	addUsertrick	-Envoyer l'astuce de l'utilisateur vers le mail de SmartData

3.2.3.2 Diagramme de séquence "Télécharger Rapport XLS"

L'acteur de cette fonctionnalité est l'administrateur. *À* travers ce cas d'utilisation, l'administrateur peut télécharger un rapport sous format .xls de l'état de l'application. Après avoir accéder à l'URI correspondante, une IHM sera affichée qui à travers laquelle peut consulter le rapport sur le Web ou le télécharger en format .xls.

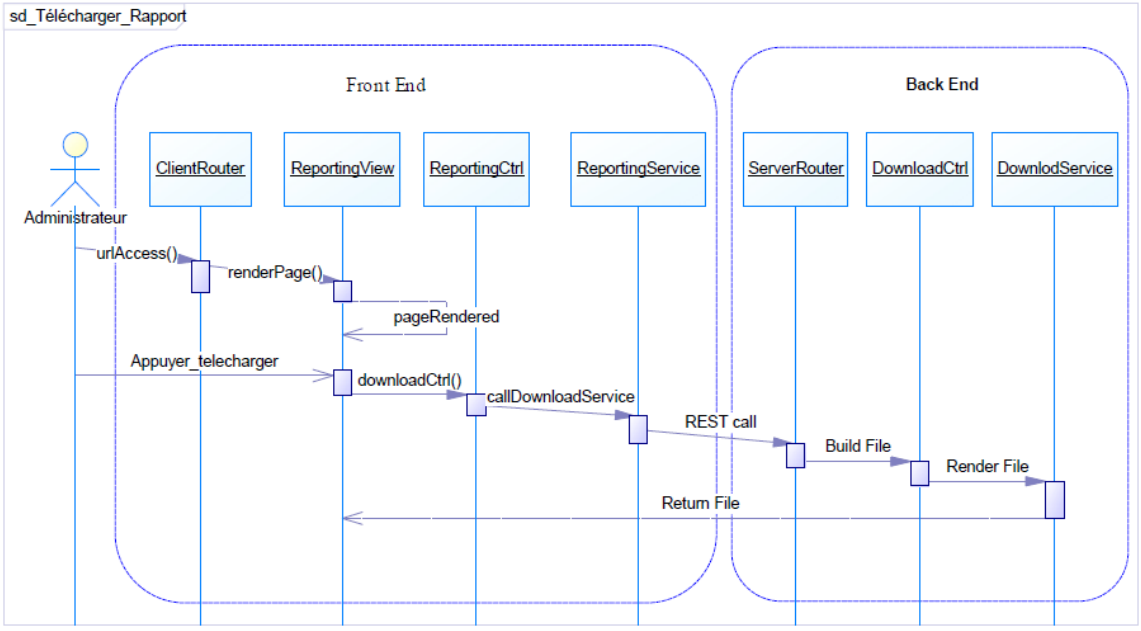

Figure 3.9: Diagramme de séquence "Télécharger Rapport XLS"

3.2.4 Diagramme d'activité

Dans cette partie nous présentons, par le modèle du diagramme d'activité, les décisions prises par le système lors de l'exécution de la fonctionnalité "Ajouter Astuce". Nous nous intéressons aux diagrammes comportementaux d'UML. Les diagrammes d'activités sont utilisés pour modéliser un workflow dans un use case ou entre plusieurs uses cases et pour spécifier une opération (logique d'une opération).

Le diagramme d'activité est le plus approprié pour modéliser la dynamique d'une tâche ou d'un use case. Nous commençons par montrer le diagramme d'activités relatives aux processus d'authentification et d'inscription puis celles qui sont relatives au processus d'ajout d'un membre.

La figure 3.10 représente le diagramme d'activité du cas d'utilisation " Ajouter astuce ".

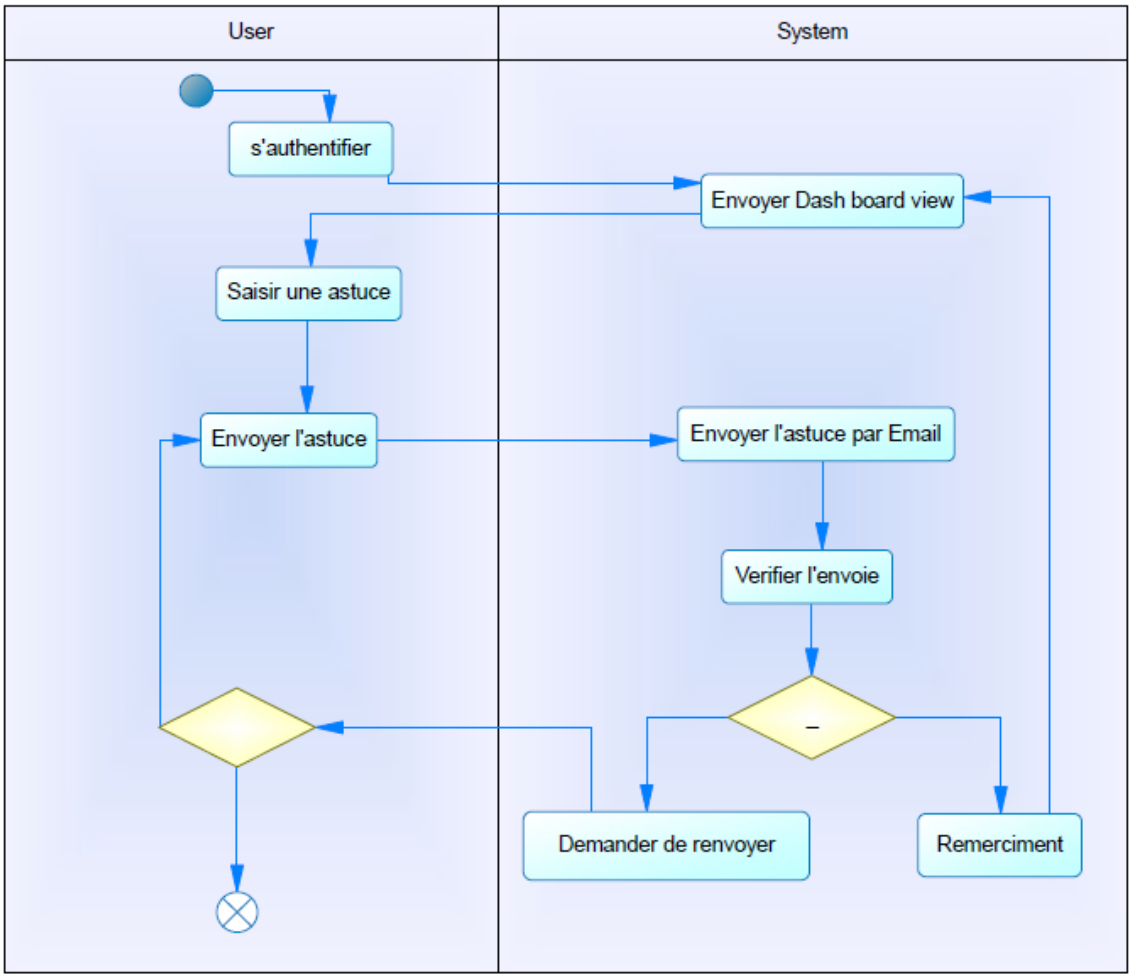

Figure 3.10: Diagramme d'activité du cas d'utilisation " Ajouter astuce "

3.2.5 Diagramme de déploiement

Un diagramme de déploiement est un diagramme UML qui fournit une représentation graphique de la configuration physique des éléments d'exécution de notre système. En phase de

production, l'infrastructure physique nécessaire se compose de quatre parties majeures:

- Le Système de Gestion de Base de Données (SGBD)
- Le serveur Web pour la gestion des requêtes, des transactions vers la base de données et le lancement des Batch
- L'infrastructure Cloud pour l'exécution des Batch
- Les parties clientes : client mobile et client léger qui consomment les API REST de la partie Backend déployée dans le serveur Web.

La figure 3.11 représente les différentes relations entre les parties du système déployés sur l'infrastructure physique ainsi que les protocoles utilisés pour assurer la communication entre les composantes de la configuration.

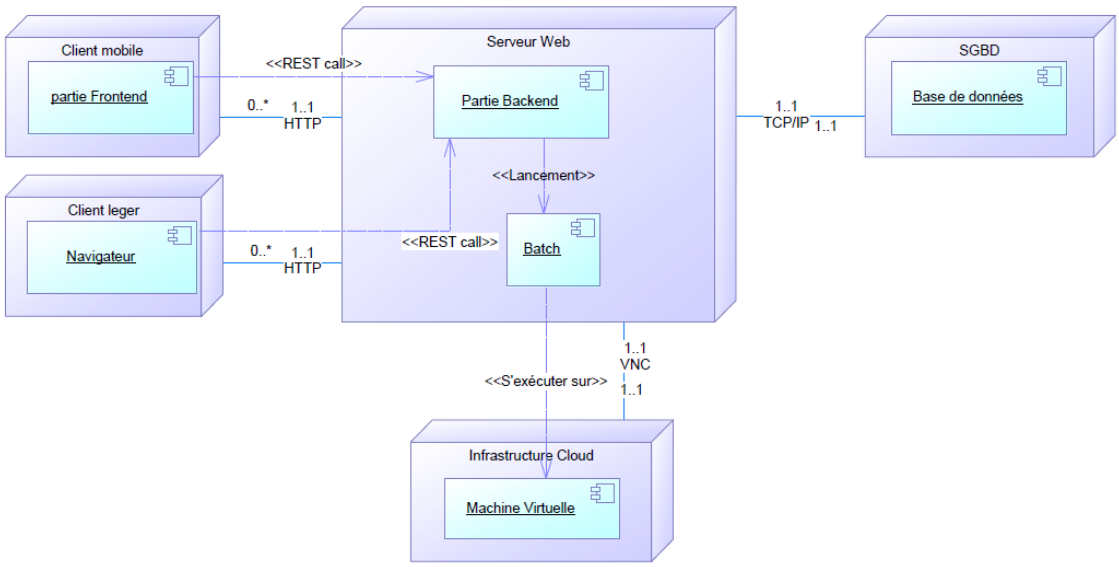

Figure 3.11: Diagramme de déploiement

Conclusion

Nous avons traité dans ce chapitre l'étude conceptuelle de notre système. Nous avons présenté, dans un premier lieu, l'architecture globale de notre application. Puis, nous avons décrit plus en détails les différentes parties du système. Nous entamons à présent la partie relative à la réalisation de notre application.

Chapter 4

Réalisation

Introduction

Une fois nous avons fixé les grandes lignes de notre application et les pilonnes conceptuelles sur les niveaux global et détaillé nous pouvons passer à l'implémentation des modèles théoriques. Pour ce faire, nous allons choisir l'environnement de développement ainsi que les technologies nécessaires. Nous passons à la fin de ce chapitre à la présentation de quelques scénarios d'exécution à travers les interfaces Homme/Machine.

4.1 Environnement de travail

Cette partie s'intéresse à l'environnement logiciel et matériel avec lequel nous avons mené notre projet.

4.1.1 Environnement physique

Nous présentons dans ce paragraphe les équipements utilisés durant toutes les phases du projet : Développement, Débogage et Test unitaire. Le tableau 4.1 présente la fiche technique des appareils utilisés.

Table 4.1: Fiche technique des équipements utilisés

Ordinateur	**Smartphone**
- Modèle: Asus - Processeur: Intel i7 - Mémoire: 12 GB RAM - Stockage interne: 1TB HDD - SE: WINDOWS 10	- Modèle: Samsung Galaxy S5 - Processeur: Quad-core 2.5 GHz Krait 400 - Mémoire: 2 GB RAM - Stockage interne: 16 GB - SE: Android 4.4.2

4.1.2 Environnement logiciel

Après la spécification des environnements physiques nécessaires à la réalisation, nous pouvons passer au choix de l'environnement logiciel. Les différents outils logiciels utilisés durant le stage pour la réalisation des fonctionnalités et la rédaction du rapport sont listés ci-après:

- **Eclipse**: Environnement de développement intégré (EDI) destiné principalement au développeur JAVA mais il est extensible vers d'autres langages tels que C/C++, PHP et plein d'autres. Nous pouvons combiner plusieurs langages sous un seul EDI à travers les plugins comme nous pouvons avoir un EDI pour chaque objectif. Nous avons utilisé Eclipse dans sa version destinée pour le Java Enterprise Edition durant le projet.
- **Visual Studio Code** : Éditeur de code source léger mais puissant qui fonctionne sur bureau et est disponible pour Windows, MacOS et Linux. Il est livré avec un support intégré pour JavaScript, TypeScript et Node.js et dispose d'un riche écosystème d'extensions pour d'autres langues (C ++, C #, Python, PHP, Go).
- **BitBucket** : Gestionnaire de versions distribuées qui permet à une équipe de coopérer ensemble dans un projets large.
- **JIRA**: Gestionnaire de taches dans une équipe de travail. Il fournit un tableau de bord pour pouvoir gérer les story dans un environnement de développement SCRUM.
- **Confluence**: Logiciel de Wiki qui permet de rédiger la documentation sur le code source délivré.
- **PowAMC**: Environnement logiciel destiné à la conception UML.
- **TeXStudio**: Éditeur de texte pour la rédaction des documents en Latex.

4.2 Choix technologiques

Cette partie s'intéresse aux choix des technologies et langages de programmation que nous allons utiliser pour parvenir à la finalité de notre projet comme le montre la figure 4.1.

Figure 4.1: Technologies utilisées

- **HTML** : langage de balisage crée et utilisé pour écrire les pages Web.
- **CSS** : langage permettent de définir des règles appliquées à un ou plusieurs documents.
- **AngularJS** : Frame-work Javascript open-source, développé par Google, permettant de réaliser des applications Web dynamique en utilisant le modèle MV-VM (Model-View View-Model).
  - **AngularUI Router** : Framework AnguarJS permettant de rajouter un router à la single page application en utilisant des vues imbriquées.
- **Ionic** : Frame-work de développement d'applications hybrides. Ce frame-work permet de développer des applications mobiles pour divers différents environnements d'appareils téléphoniques (Android, Ios et WindowsPhone) en combinant à la fois les technologies Web (HTML, CSS et JavaScript) et les technologies de développement natives des différents environnements (Java SDK, Swift, Objective C ...)[**3**].
- **Java Enterprise Edition** : Extension du Java SE contenant des bibliothèques destinée au développement des applications Web robustes.
  - **Jax-RS** : Spécification standard JEE qui fournit les méthodes du protocole HTTP pour pouvoir effectuer des appels REST. Ce standard est indépendant du format des données et est basé sur les POJO.
  - **JPA** : API qui permet le mapping relationnel des objets Java sur les classes persistantes de la base de données, utilisé pour interroger la base lors des transactions faites par soit les utilisateurs soit par l'administrateur[**6**].
  - **JDBC** : Bibliothèque qui permet l'accès au driver de la base de données. Cette bibliothèque est utilisée majoritairement dans les transactions faites lors des traitements par lots (Batch) car elle présente une légèreté de lors de l'interrogation de la base des données.
  - **Spring Batch** : Premier Framework Java pour les traitements par lots. La framewrk offre un type par défaut de traitement Batch appelé Job où le job est composé de trois step: Read, Process et Write. Un autre type offert par la framework appelé tasklet offre la possiblité de personnaliser le traitement et d'ajouter autant d'étape que l'on veut à un traitement Batch. Nous avons opté pour le deuxième type parce que la plupart des traitements par lots de notre projet ne présente pas des étapes de lecture ni d'écriture [**4**] (voir figure 4.2).
  - **Apache POI**: Bibliothèque permettant de créer des fichiers Microsoft Office (XLS, Docx, PPT, ...)
  - **StreamingOutput**: Bibliothèque permettant de télécharger (Donwnload) des fichier en streaming.
  - **Maven** : Gestionnaire de paquets pour l'environnement JEE.
- **MySQL** : Système de gestion de base de données relationnelles.
- **ServicesWeb REST** : Exposent entièrement les services comme un ensemble de ressources (URI) identifiables et accessibles par la syntaxe et la sémantique du protocole HTTP

Ci-dessous est représenté un schéma descriptif des principales éléments qui constituent une application faisant appel à un traitement par lots. Généralement, l'architecture est réalisée en trois couches (voir figure 4.2):

- Application mère ou couramment appelé projet parent qui contient tous les sous projets en particulier l'application Batch.
- Batch Core qui est la couche concernée par le lancement, la surveillance et la gestion des Batch.
- Batch infrastructure est la dernière couche qui fournit des API telles que la lecture, le processing et l'écriture des données.

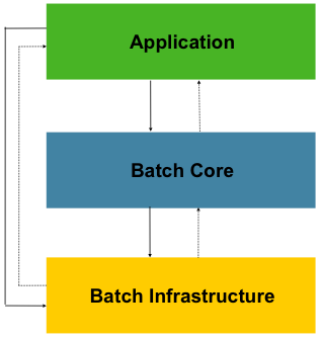

Figure 4.2: Relation entre l'application serveur et les applications Batch

4.3 Scénarios d'exécution

Ce paragraphe est consacré à la description de quelques scénarios d'exécution à travers les interfaces Homme/Machine fournies par l'application.

4.3.1 Partie: Utilisateur

L'utilisateur a un accès à l'application à travers une IHM sur son propre appareil téléphonique (smartphone). *À* l'entrée, l'application demande l'utilisateur de connecter son smartphone à Internet pour pouvoir récupérer les données du serveur comme l'indique l'interface 4.3.

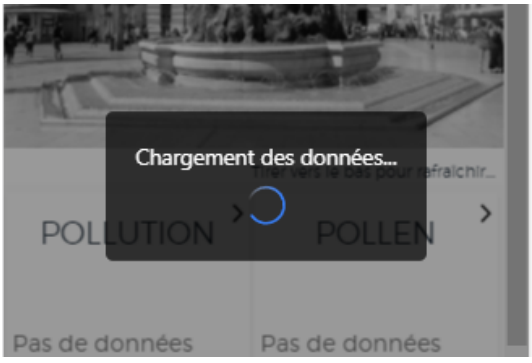

Figure 4.3: Récupération des données

Après avoir récupérer les données, l'application aura besoin de la région désirée. L'utilisateur peut alors, soit activer le service de géolocalisation, soit saisir le code ou le nom de la région comme le montre les exemples de la figure 4.4.

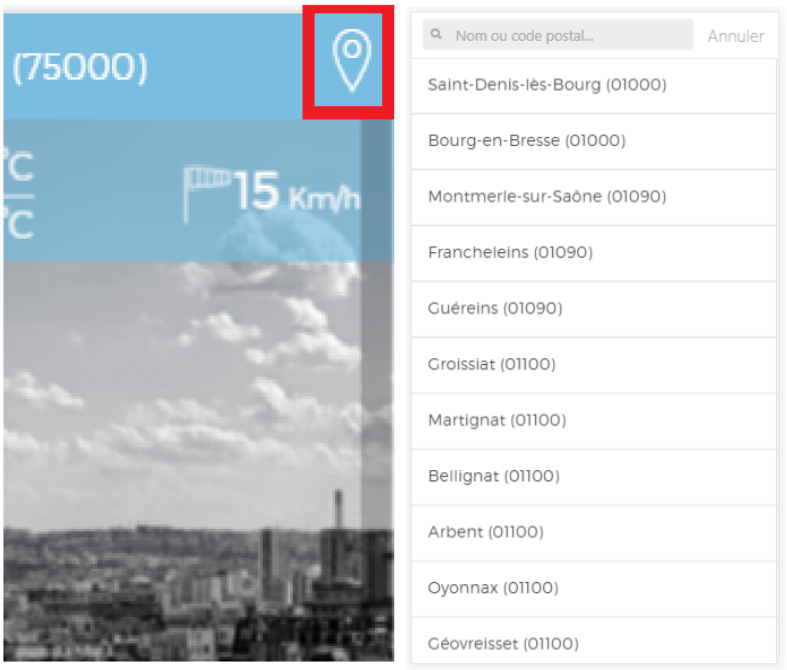

Figure 4.4: Indication de la ville : Par Service GPS / Par recherche

Une fois l'application a pu se connecter à Internet et localiser l'utilisateur elle peut maintenant afficher les niveaux des risques auxquels l'utilisateur est abonné (voir figure 4.5).

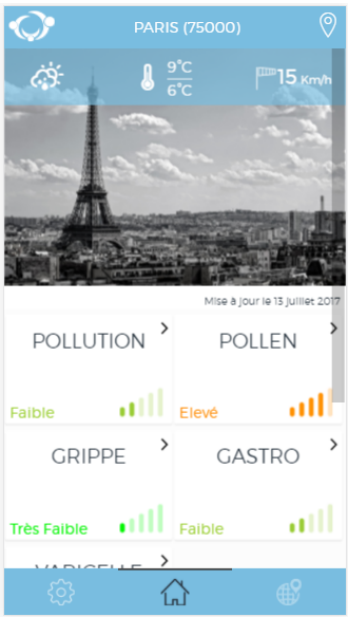

Figure 4.5: Consultation des niveaux des risques

L'utilisateur peut aussi consulter des détails et des conseils relatives aux risques (voir figure 4.6) en cliquant sur le risque désiré.

Figure 4.6: Consultation des détails et des conseils relatives à la pollution

L'application offre encore la possibilité d'ajouter des astuces à l'application. Dans le menu principal, L'utilisateur peut faire descendre l'écran vers le bas, il trouvera alors la section intitulée "pense bête" où est affiché un message aléatoire relatif au risque ayant le plus haut niveau (voir figure 4.7).

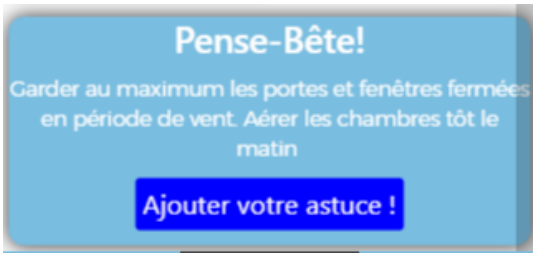

Figure 4.7: Section: Pense Bête

En cliquant sur le bouton "Ajouter votre astuce", une popup s'affiche, comme le montre la figure 4.8, donnant ainsi la possibilité de remplir les informations qui concernent l'astuce ajoutée (Pseudonyme du contributeur, Adresse mail, Type de risque et Texte de l'astuce).

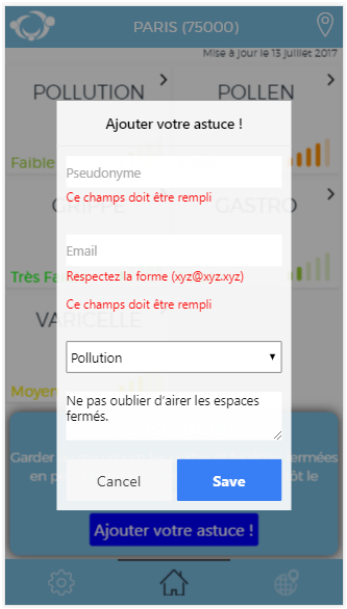

Figure 4.8: Popup: Ajout d'astuces

La figure 4.8 montre aussi le cas échéant où l'utilisateur saisit incorrectement les informations demandées. Au cas où les informations sont absentes ou incorrectes, un message d'erreur est affiché sous chaque champ erroné.

À la réception d'une notification, l'application navigue vers une interface où l'utilisateur peut choisir parmi trois conseils qui lui seront envoyé à chaque pique de niveau du risque en question. En cliquant sur le bouton "Enregistrer", il enregistre ses choix et retourne vers le menu principal (voir figure 4.9).

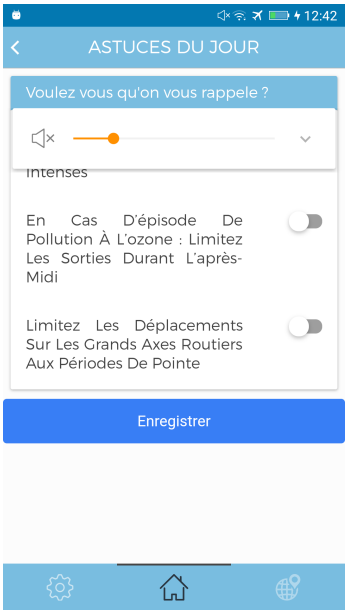

Figure 4.9: Choix des astuces "Pique des niveaux des risques"

L'utilisateur consulte la rubrique configuration pour paramétrer l'application (voir figure 4.10). Cette dernière accorde à l'utilisateur plusieurs fonctionnalités:

- S'abonner et/ou se désabonner d'un risque
- Permettre l'accès aux données personnelles (position géographique)
- S'abonner ou se désabonner aux notifications

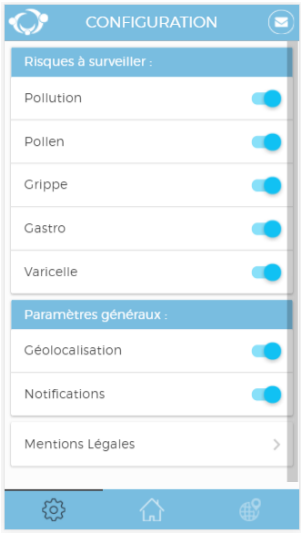

Figure 4.10: Interface Configuration

Finalement, l'utilisateur peut utiliser les boutons de navigation en bas de l'écran pour consulter les différentes interfaces.Il peut alors solliciter les stations de mesure affichées sur la carte géographique avec le niveau des risques. Il a aussi la possibilité de balayer les risques et les villes. La figure 4.11 illustre la rubrique Carte des risques.

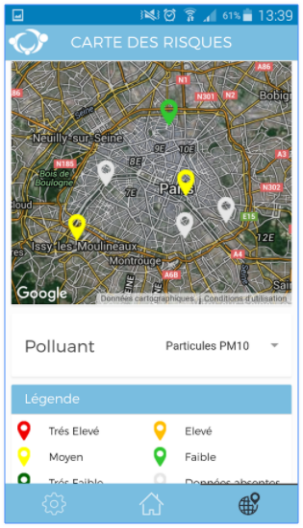

Figure 4.11: Interface Carte des risques

4.3.2 Partie: Administrateur

L'administrateur a accès à une interface pour visualiser le rapport journalier de l'application. Les informations affichées sont:

- L'historique des notifications
- Les villes les plus actives
- Les risques les plus consultés

L'administrateur peut également télécharger une version xls pour une éventuelle utilisation hors ligne, en cliquant sur le bouton "Export". Le serveur prépare un fichier xls contenant les informations actualisées du rapport, ensuite l'envoie vers la machine de l'administrateur. La figure 4.12 montre l'interface affichée à l'administrateur.

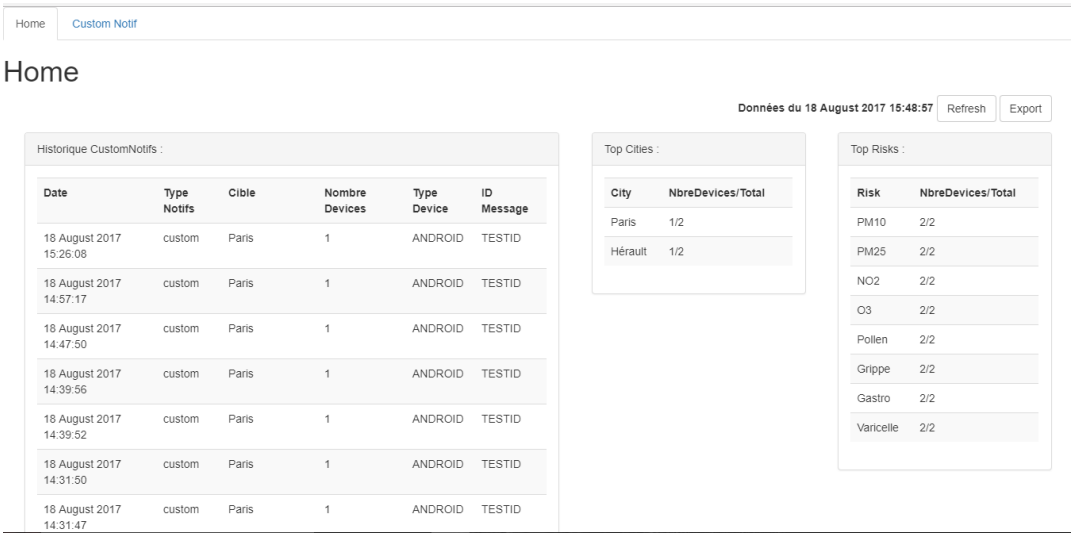

Figure 4.12: Interface Administration

L'administrateur peut aussi envoyer des notifications personnaliser aux utilisateurs (voir figure 4.13).

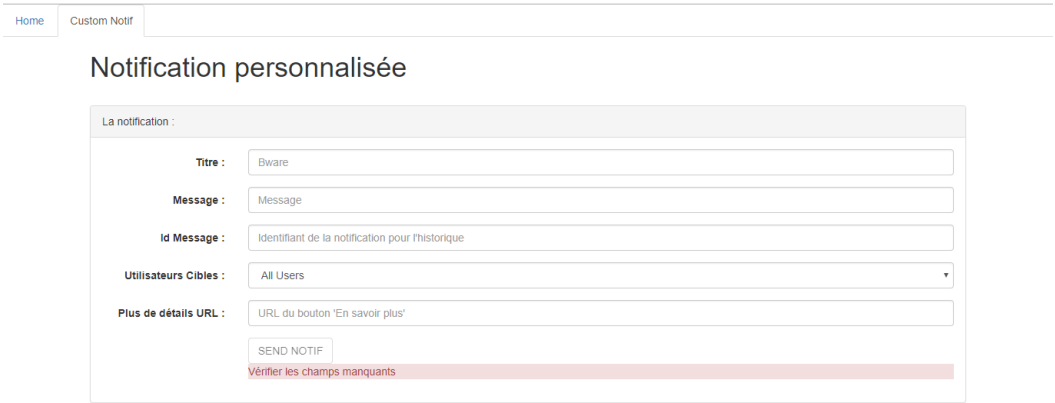

Figure 4.13: Interface Notification personnalisée

4.4 Déploiement et publication de l'application

4.4.1 Déploiement de la partie serveur

Les étapes décrites ci-dessous concernent le déploiement en phase de pré-production. Avant de déployer la partie backend dans la machine distante, il faut impérativement préparer le projet à être exécuté dans le nouvel environnement. La différence entre la phase débeugage et la phase production réside majoritairement dans les ressources. Dans la phase débeugage, le serveur Web, le SGBD et l'application cliente existent tous dans la machine du développeur par contre dans la phase pré-production on va séparer chacune de ces parties dans une machine à part comme il est indiqué dans le paragraphe 3.2.5. Cette séparation est effectuée dans le but de tester l'architecture complète avant de la faire passer vers le client final. Pour déployer la partie serveur il faut alors effectuer les étapes suivantes:

- Reconfigurer l'application avec les nouvelles adresses et ports utilisés en pré-production (Base de donnée, ports ouverts de l'application...)
- Préparer l'application à être exécuté dans le nouvel environnement (Variables d'environnement, variable système...)
- Se connecter en Secure Shell (SSH) à la machine distante par la commande
  - ssh root@<Adresse IP de la machine>
- Éteindre le serveur Web
  - /usr/share/tomcat8/bin/shutdown.sh
- Supprimer le backup existant et le remplacer par un backup de la version en cours
  - rm -rf webapps-smartdata.old
  - cp -rf webapps-smartdata webapps-smartdata.old
- Copier le Web Archive (WAR) vers la machine en utilisant le protocole Secure Copy (SCP) et le placer dans le serveur Tomcat
  - scp sd.war root@<Adresse IP de la machine>:/root/documents/2017-07-26
  - cp /root/document/2017-07-26/sd.war /var/lib/tomcat8/webapps
- Redémarrer le serveur Tomcat
  - sudo service tomcat8 start; tail -f logs/catalina.out

4.4.2 Publication de l'application dans le play store

Pour rendre accessible notre application aux utilisateurs il faut publier celle-ci dans les stores spécifiques à chaque système d'application mobile. Les étapes décrites ci-après concernent la publication dans le play store de Google spécifique à Android. Pour publier l'application mobile dans le store il faut:

- Actualiser les informations de configuration de l'application (version, IP du serveur de production...)
- Générer une version release de l'application mobile

  - ionic cordova build –release android

- Générer une clé de signature (il faut l'utiliser pour toute les versions de l'application)
  - keytool -genkey -v -keystore release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
- Signer le binaire avec la clé de signature
  - jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore release-key.keystore binary-unsigned.apk alias_name
- Ziper le binaire
  - zipalign -v 4 binary-unsigned.apk ready.apk
- Télécharger le binaire vers le play store
- Publier le binaire dans la forme désirée (Alpha, Beta, Production)

Conclusion

Dans ce chapitre, nous avons présenté l'environnement logiciel et matériel dans lequel notre projet a été élaboré. Nous avons ensuite, présenté notre travail à travers un enchaînement de quelques scénarios d'exécution illustrés par des interfaces de l'application. Finalement, nous avons présenté les étapes nécessaires pour déployer notre projet et publier l'application mobile.

Conclusion générale

Tout au long du projet nous étions menés à concevoir et à apporter des améliorations à une application mobile permettant de suivre le niveau de pollution de l'air. L'application facilitera aux utilisateurs le suivie de l'état quotidien de l'air. Elle permet aussi à l'administrateur de suivre l'état de l'application à travers des rapports à jour sous format xls qu'il peut télécharger.

Ce rapport décrit toutes les phases de la réalisation du projet. Commençant par le contexte général puis la présentation de l'objectif du projet ainsi qu'une étude menée sur les solutions existantes sur le marché de l'e-santé et les limites des solutions proposées. Ensuite nous avons passé à l'analyse des besoins fonctionnels et non-fonctionnels ce qui nous a permis de fixer le cahier des charges auquel doit répondre notre application. Ce qui nous a permis d'aborder la partie de la conception où nous avons spécifié l'architecture de notre projet et ses composantes. Enfin nous avons précisé l'environnement où nous avons réalisé notre application et les outils technologiques utilisés tout au long de la phase du développement.

Le stage ingénieur nous a permis de consolider les connaissances acquises dans notre cursus académique, en nous donnant la possibilité d'appliquer les notions théoriques du cours qui nous ont été une base solide pour comprendre l'engin du développement mobile hybride multi-plateforme.

Notre application étant fonctionnelle, mais elle présente quelques imperfections. Nous projetons de lui rajouter un module qui permettra aux utilisateurs de s'interagir et de partager les informations sur les réseaux sociaux ce qui donnera plus de visibilité à l'application. Une majeure amélioration qui peut être additionnée consiste à un module de prévision des niveaux des risques à travers des algorithmes de Machine Learning et d'Intelligence Artificiel.

Bibliography

[1] Communiqué de presse OMS: URL: http://www.who.int/mediacentre/news/releases/2016/air-pollution-estimates/fr/(visité le 10/09/2017)

[2] Guide de démarrage Scrum: URL: http://www.agiliste.fr/guide-de-demarrage-scrum/(visité le 03/09/2017)

[3] Documentation Ionic: URL: http://ionicframework.com/ (visité le 15/09/2017)

[4] Lucas Ward, Dave Syer, Thomas Risberg, Robert Kasanicky, Dan Garrette, Wayne Lund, *S*pring Batch - Reference Documentation, 2009

[5] ARNAUD COGOLUEGNES, THIERRY TEMPLIER, GARY GREGORY, OLIVIER BAZOUD, *S*pring Batch in Action, 2012

[6] Oliver Gierke, Thomas Darimont, Christoph Strobl, Mark Paluch, *S*pring Data JPA - Reference Documentation, avril 2017

[7] Jon Brisbin, Oliver Gierke, Greg Turnquist, *S*pring Data REST - Reference Documentation, avril 2017

[8] Salah BENNOUR, *T*utoriel Ionic framework, 2015

Annexes

Annexe 1
RESTful Web Services

Representational State Transfer ou communément REST est une architecture d'application basée sur le standard HTTP. REST utilise les opérations natives de HTTP : POST, GET, PUT, DELETE pour le mapping sur les quatre fonctions fondamentales d'une base de données qui sont Create, Read, Delete, et Update. Ces services peuvent ensuite être consommés par des clients authentifiés [**7**].

Les services Web REST sont utilisés pour développer des applications orientées ressources dans lesquelles chaque composant est une ressource accessible en utilisant les méthodes standards du HTTP. Les applications qui respectent les architectures orientées ressources sont appelées RESTful.

1 Caractéristiques

- Sans état : le serveur ignore l'état des clients entre les requêtes
- Cacheable : les candidats doivent être capables de garder en mémoires des informations.
- Orienté clients serveur : le client et le serveur sont les entités communiquent dans une architecture REST. Le client requière une ressource et le serveur la lui fournira, s'il est authentifié.
- Réalisé en couche : le pattern MVC doit être respecté.
- Les ressources sont accessibles à travers les méthodes du HTTP (POST, GET, PUT, DELETE)

La communication entre le serveur et les clients peut se faire sous plusieurs formats : JASON, XML, CSV... Le présent projet utilise le format JSON pour la communication client-serveur.

2 Méthodes du HTTP

- GET Accès en lecture seule à la ressource,
- PUT Création d'une nouvelle ressource,
- DELETE Suppression d'une ressource,
- POST Création d'une nouvelle ressource ou mise à jour d'une ressource existante

Table 2.1: Exemple d'une API REST

Méthode HTTP	Route	Description	Type de l'opération
GET	/Utilisateur/one	Obtenir un Utilisateur	lecture seule
GET	/Utilisateur/all	Obtenir tous les Utilisateurs	lecture seule
POST	/Utilisateur/add	Ajouter un Utilisateur	écriture seule
PUT	/Utilisateur/update	Mettre à jour un Utilisateur	écriture seule
DELETE	/Utilisateur/delete	Supprimer un Utilisateur	écriture

3 Format de données

Les différentes parties de notre application communiquent en utilisant le format de représentation de données JSON (JavaScript Object Notation – Notation Objet issue de JavaScript) qui est un format léger d'échange de données. Il est facile à lire ou à écrire pour des humains.

JSON possède deux structures de données :

- Objet : collection de pairs nom/valeur
- Liste : liste de valeurs ordonnés

la figure 3.1 représente les deux structures de données JSON.

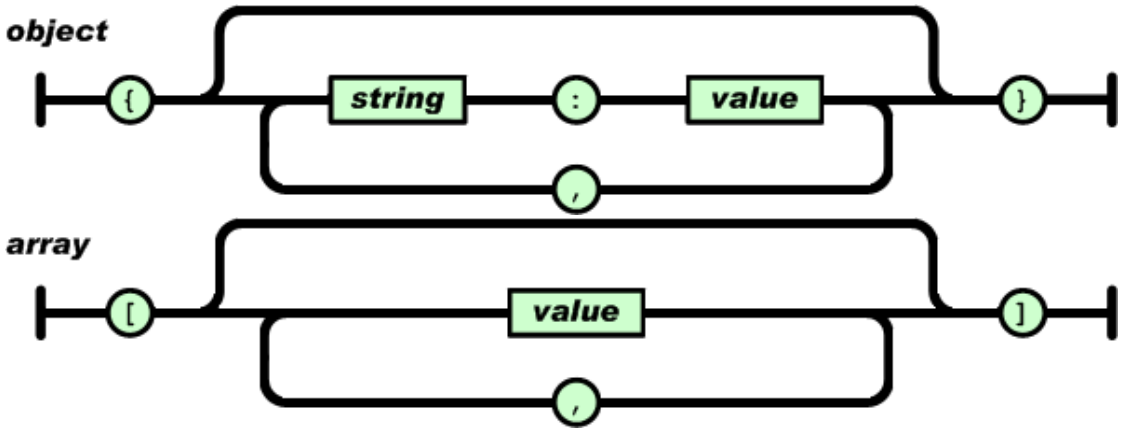

Figure 3.1: Structures de données JSON

3.1 Exemple

paramètre de la requête d'un level :

Key	Value
zipCode	75
withStations	true
withStatics	false

Figure 3.2: Paramètres d'une requête

Réponse du serveur :


```
{
    "levels": [
        {
            "date": "2017-08-22",
            "station": "Paris_04",
            "zipCode": "75",
            "risk": "pollution",
            "levelpreviewsDay": null,
            "level": 2,
            "levelJ1": 2,
            "levelJ2": 2
        },
        {
            "date": "2017-08-20",
            "station": "depID_75",
            "zipCode": "75",
            "risk": "pollen",
            "levelpreviewsDay": null,
            "level": 2,
            "levelJ1": null,
            "levelJ2": null
        },
```


Figure 3.3: Réponse du serveur