



Institut Supérieur d'Informatique  
de Modélisation et de leurs  
Applications

1 rue de la Chebarde  
TSA 60125  
CS 60026  
63 178 Aubière cedex

Rapport d'ingénieur  
Projet de 3<sup>e</sup> année

Filière architecture et génie logiciel

---

# WatchDogZZ

---

*Étudiants :*

Benjamin BARBESANGE,  
Benoît GARÇON

*Tuteur :*

Pierre COLOMB

*Tuteur ISIMA :*

Eva HASSINGER

## **Remerciements**

Avant de débiter notre étude, nous tenons à remercier M. Pierre Colomb, tuteur de ce projet, pour l'accompagnement et les réponses qu'il a su apporter à nos questions.

## **Résumé – Abstract**

# Table des matières

Remerciements	i
Résumé – Abstract	ii
Table des matières	iii
Liste des figures, tableaux, algorithmes et extraits de code	iv
Glossaire	v
<b>1 Introduction</b>	<b>1</b>
<b>2 Etudes préalables</b>	<b>2</b>
2.1 Présentation du projet . . . . .	2
2.2 Analyse de l'existant . . . . .	2
2.3 Spécifications du projet . . . . .	2
2.3.1 Architecture . . . . .	2
2.3.2 Partie serveur . . . . .	3
2.3.3 Partie Android . . . . .	5
2.3.4 Intégration continue . . . . .	6
2.4 Organisation du travail . . . . .	6
<b>3 Conception de la solution</b>	<b>8</b>
3.1 Architecture de la solution . . . . .	8
3.1.1 Web Service . . . . .	8
3.1.2 Application Android . . . . .	8
3.2 Fonctionnalités introduites . . . . .	8
3.3 Je ne sais pas . . . . .	8
<b>4 Résultats</b>	<b>8</b>
4.1 Test de la solution . . . . .	8
4.2 Améliorations possibles . . . . .	8
<b>5 Conclusion</b>	<b>9</b>
Références webographiques	i

## Liste des figures, tableaux, algorithmes et extraits de code

### Liste des figures

2.1	Architecture . . . . .	3
2.2	Diagramme de Gantt théorique . . . . .	7

### Liste des tableaux

### Liste des algorithmes

### Liste des extraits de code

2.1	Corps de la réponse serveur . . . . .	4
2.2	Corps de la requête login . . . . .	4
2.3	Corps de la requête who . . . . .	4
2.4	Corps de la requête where GET . . . . .	5
2.5	Corps de la requête where POST . . . . .	5

## Glossaire

## Introduction

Ce rapport a été rédigé dans le cadre du projet de troisième année du cycle ingénieur à l'Institut Supérieur d'Informatique de Modélisation et de leurs Applications (ISIMA) réalisé sur une durée de 120 heures par personne. Nous avons proposé de notre propre initiative le sujet de ce projet : la conception d'une carte interactive d'un établissement. Cette idée se base sur un constat très simple : il est parfois difficile de s'orienter dans un bâtiment de grande taille et de trouver une personne en mouvement en son sein. Ce projet s'inspire en grande partie de la carte du maraudeur de l'univers Harry Potter, carte sur laquelle il est possible de suivre en temps réel le déplacement de toutes les personnes dans l'enceinte de Poudlard, l'école des sorciers. L'objectif est donc de proposer et mettre en place une solution évolutive, innovante et pratique pour les utilisateurs afin de s'orienter dans les bâtiments de l'ISIMA. Nous pouvons penser que ce type de solution peut s'étendre à tout type de bâtiment au sein duquel il est autorisé et possible d'être localisé. Cette solution peut avoir des applications dans le domaine du secourisme, ce qui peut permettre aux sapeurs-pompiers de localiser des personnes facilement dans un bâtiment enfumé, permettre à des entreprises hébergeant des données sensibles de localiser ses visiteurs ou collaborateurs, ou encore d'optimiser les déplacements de personnes dans des bâtiments de grande taille comme une aide à l'orientation de médecins dans un hôpital ou de techniciens dans une usine. Nous verrons que la solution mise en place s'organise en deux principaux composants. Le premier composant consistera en un service web capable de répondre à des requêtes utilisateur de type HTTP. Ces requêtes permettront aux utilisateurs d'envoyer leur position afin de la stocker sur le serveur et de l'envoyer aux autres utilisateurs qui en font la demande. Le second composant consistera en la création d'un client du service web qui affichera à la fois les données sur les lieux mais permettra aussi le suivi et l'interaction avec ses usagers. La plateforme cible choisie pour le développement du client est une plateforme mobile afin qu'il puisse être utilisé en tout lieu et à tout moment. Ces deux parties, quoi que centrales, s'articulent au sein d'un ensemble plus complet d'outils de génie logiciel donnant à la solution une identité unique et que nous détaillerons par la suite. L'étude débutera par une partie d'études préalables plus précises sur le sujet tant au niveau du travail à fournir pour la réalisation de la solution que de l'état de l'art en la matière. Ensuite, nous détaillerons dans une seconde partie la conception de cette solution en détaillant nos méthodes et outils, pour enfin terminer ce rapport par les résultats finaux de notre travail et discuter du potentiel de notre application.

# 1 Etudes préalables

## 1.1 Présentation du projet

Ce projet à été mis en place selon notre propre initiative et nous avons donc défini les objectifs à atteindre ainsi que les fonctionnalités de nous même, avec l'aide du tuteur de projet.

Le but de ce projet est de proposer une manière simple pour tout le monde de pouvoir s'orienter dans le locaux de l'ISIMA. Une fois cette solution éprouvée avec les bâtiments de l'ISIMA, nous pouvons penser l'étendre à n'importe quel autre bâtiment donc nous pouvons avoir les plans. Cette idée est inspirée d'un film de Harry Potter, avec la fameuse carte du Maraudeur qui lui permet de suivre les déplacements de tout le monde dans Poudlard.

De manière générale, nous devons être capable de visualiser une carte de l'ISIMA, mais également de pouvoir trouver facilement un bureau ou une salle de cours. De plus, il serait intéressant de pouvoir également localiser n'importe quelle autre personne visualisant la carte en même temps. Cette visualisation doit s'actualiser assez rapidement pour que la location de personnes soit la plus précise possible pour les utilisateurs.

## 1.2 Analyse de l'existant

## 1.3 Spécifications du projet

Etant donné que peu de solutions sont disponibles sur le marché, nous avons choisi de partir de zéro et créer notre solution.

La solution la plus évidente en terme de support de visualisation pour les utilisateurs et de leur proposer une application qu'ils pourront installer sur leur smartphone, pc, montre connectée ou encore tablette. Afin de rendre cette application dynamique et de proposer un suivi de position d'utilisateurs en temps réel, il est convenu d'utiliser un Service Web. Ce service permettra également de stocker des informations utiles aux utilisateurs, ce qui permettra également d'alléger le volume de données stockées sur leurs terminaux.

### 1.3.1 Architecture

L'architecture choisie pour organiser notre solution est la suivante.



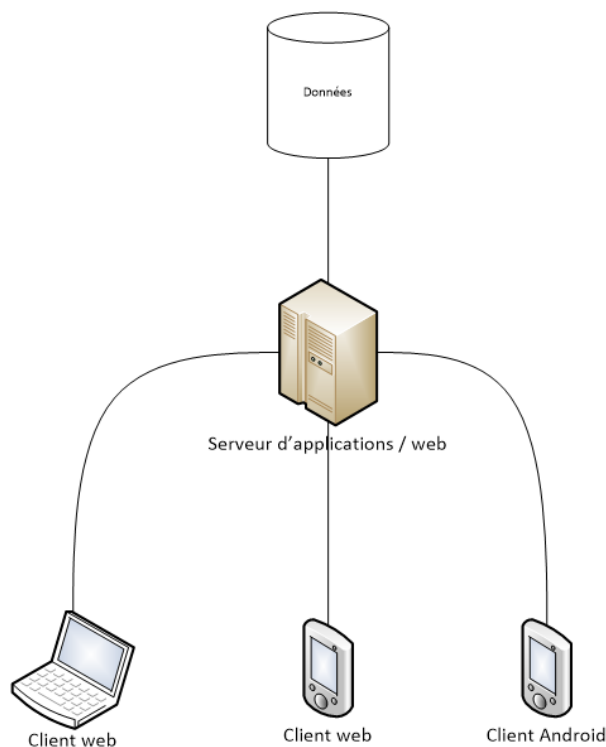


Figure 1.1 – Architecture

Nous pouvons observer dans la figure 2.1 que celle-ci ressemble à une architecture client/-serveur, ce qui est le but recherché.

### 1.3.2 Partie serveur

La partie centrale est celle du Web Service. Le serveur doit être capable de répondre aux différentes connexions et requêtes des utilisateurs (se connectant sur l'application Android ou tout autre...). Le serveur sera un service Web permettant (au minimum) :

- de s'authentifier
- d'obtenir les positions d'autres personnes connectées sur l'application
- obtenir la liste des personnes connectées
- envoyer la position actuelle de l'utilisateur connecté
- proposer l'apk

Le serveur pourra aussi en option faire :

- historique des positions
- calcul d'itinéraire
- partage de position

Le serveur comportera une base de données dans laquelle les informations des utilisateurs seront stockées (login / mot de passe). Les dernières positions des utilisateurs seront également stockées. Il y aura une table permettant d'identifier quels utilisateurs sont actuellement actifs. Il a été convenu d'utiliser NodeJS du côté serveur pour plusieurs raisons :

- rapidité de mise en place
- faible nombre d'utilisateurs

— l'un de nous a des connaissances dessus

Base de données La base de données utilisée est une base MongoDB. Ce type de base de données est très simple à mettre en place et se base sur un format JSON binaire pour stocker les différentes informations que nous souhaitons. Ceci va faire qu'il est facile d'utiliser les résultats des requêtes directement en Javascript. Requetes sur le web service Des URL sont mises à disposition par le service et permettent d'effectuer certaines tâches. Le passage de paramètres pour ces URL se fait directement dans le corps de la requête sous forme de JSON. Les réponses du service sont sous forme d'objet JSON dans le corps de la réponse. Les réponses contiennent les champs suivants :

*Code 1.1 – Corps de la réponse serveur*

```
1 {
2   'status': 'ok' / 'fail', // L'état de la requête
3   'error': 'description' // Une description de l'erreur s'il y en a
   une
4 }
```

La plupart des requêtes décrites nécessitent que l'utilisateur soit authentifié. /login La première requête à effectuer sur le service doit s'effectuer avec la méthode POST sur cette URL. Pour se connecter, l'utilisateur doit envoyer les paramètres suivants :

*Code 1.2 – Corps de la requête login*

```
1 {
2   'name': 'username', // Le nom de l'utilisateur à connecter
3   'location': [1.0, 2.0, 3.0] // La position courrante de l'
   utilisateur
4 }
```

/who Cette URL permet de récupérer en méthode GET une liste de noms de personnes actuellement en ligne. Le retour est sous la forme suivante :

*Code 1.3 – Corps de la requête who*

```
1 {
2   'list': [
3     'userName1',
4     'userName2',
5     'userName3'
6   ]
7 }
```

/where En utilisant la méthode GET, cette URL retourne la liste des utilisateurs ainsi que leur position. En utilisant la méthode POST et en passant les paramètres adéquats, l'utilisateur peut mettre à jour sa position. En méthode GET, le service renvoie une liste des utilisateurs connectés avec leurs positions :

*Code 1.4 – Corps de la requête where GET*

```
1 {
```

```
2      'list': [  
3          {  
4              'name': 'username1',  
5              'location': [1.0, 2.0, 3.0]  
6          },  
7          {  
8              'name': 'username2',  
9              'location': [1.0, 2.0, 3.0]  
10         },  
11         {  
12             'name': 'username3',  
13             'location': [1.0, 2.0, 3.0]  
14         }  
15     ]  
16 }
```

En utilisant la méthode POST, l'utilisateur met à jour sa position. Les paramètres à envoyer sont les suivants :

*Code 1.5 – Corps de la requête where POST*

```
1 {  
2     'name': 'username',  
3     'location': [1.0, 2.0, 3.0]  
4 }
```

Si l'utilisateur n'est pas connecté au moment de mettre à jour sa position, le serveur va tenter de l'authentifier et sauvegarder sa position. Partie administration Les fonctionnalités minimales pour l'application d'administration sont :

- visualisation des logs du serveur
- visualisation du contenu de la base de donnée

Ensuite les fonctionnalités avancées pourront être :

- gestion des utilisateurs

### 1.3.3 Partie Android

Une seconde partie comporte une application Android (qui pourrait être déclinée pour iOS et Windows Phone). Cette application permettra :

- de s'inscrire
- de se connecter
- envoyer sa position gps au service web
- recevoir les positions gps d'autres utilisateurs connectés
- visualiser en temps réel sur une carte les positions

L'application pourra évoluer et proposer :

- la carte en version 3D
- la carte en version VR
- l'ajout d'informations sur la carte (lieu / point de rdv)

Le choix d'une application Android se justifie par :

- la communauté Android est active
- la quantité de terminaux
- l'un de nous a des connaissances en Android

La mise à jour des positions est soit faite par l'application qui effectue une requête sur le serveur, soit c'est le serveur qui renvoie les positions des utilisateurs ayant bougé d'un delta suffisant qui permet sa mise à jour chez les utilisateurs. Les frameworks 3.X+ seront supportés pour fonctionner sur un maximum de terminaux.

### 1.3.4 Intégration continue

Un des objectifs de ce projet est de mettre en place une intégration continue et un déploiement automatique. Pour ce faire il est nécessaire d'avoir deux serveurs :

- un serveur d'application
- un serveur d'intégration

Le premier sera certainement un CaaS Amazon pour NodeJS. Le serveur d'intégration sera un serveur Travis CI puisqu'il gère à la fois le NodeJS et l'Android (nouvelle fonctionnalité). Cela permettra contrairement à un serveur Jenkins de déployer facilement sans avoir à gérer un serveur mais juste en utilisant un service.

## 1.4 Organisation du travail

L'organisation temporelle théorique du travail est décrite dans le diagramme de Gantt ci-dessous.

Diagramme de Gantt initial du projet WatchDogZZ

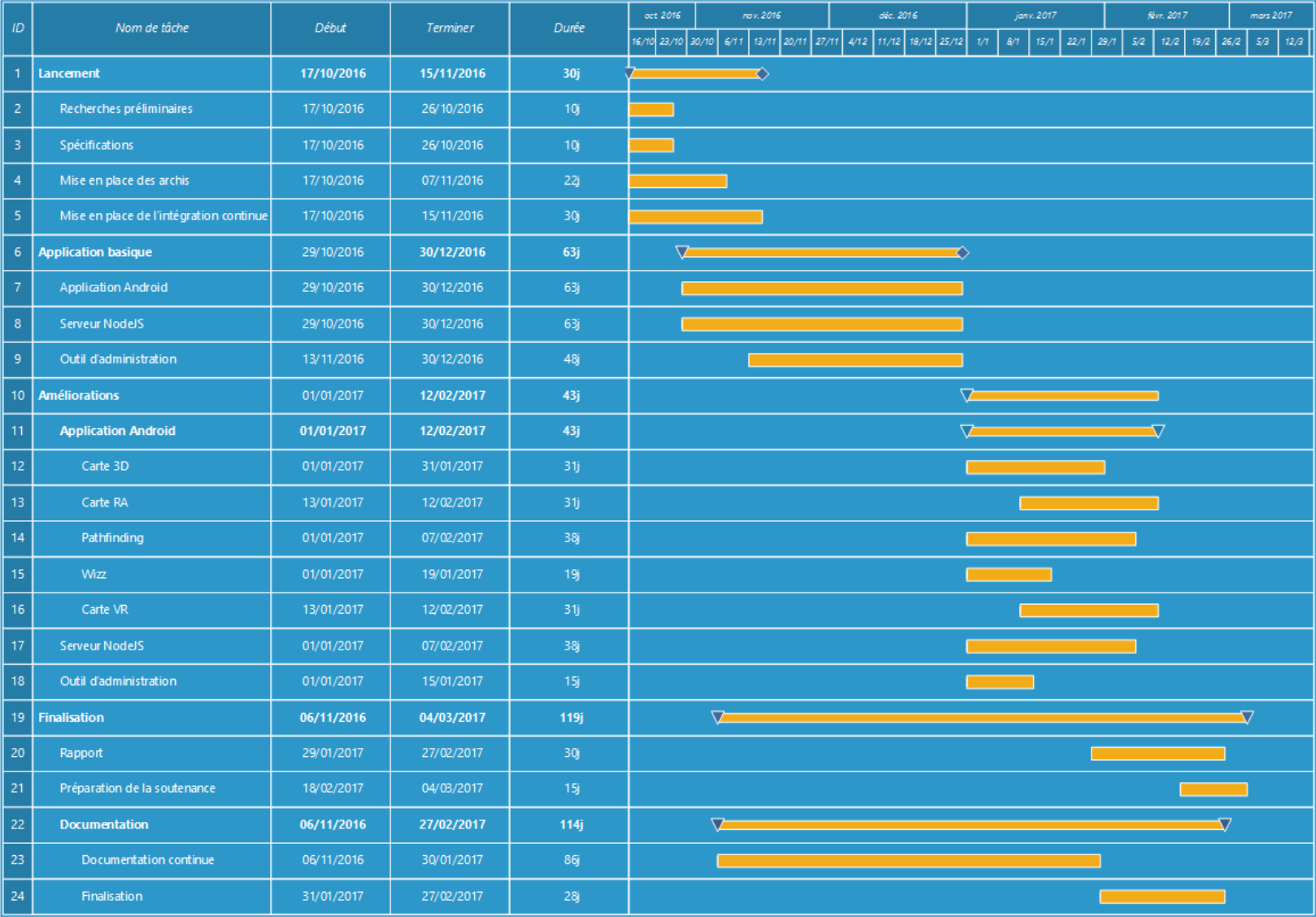


Figure 1.2 – Diagramme de Gantt théorique

suite...

## **2 Conception de la solution**

### **2.1 Architecture de la solution**

#### **2.1.1 Web Service**

#### **2.1.2 Application Android**

### **2.2 Fonctionnalités introduites**

### **2.3 Je ne sais pas**

## **3 Résultats**

### **3.1 Test de la solution**

### **3.2 Améliorations possibles**

## Conclusion

Ce projet a donc été l'occasion de concrétiser au travers de l'application WatchDogZZ notre idée personnelle. Nous avons pu développer une application complète reprenant les principes fondamentaux de la carte du maraudeur à savoir la géolocalisation d'utilisateurs dans un établissement. A ceci de nombreuses fonctionnalités ont pu être ajoutées comme le partage de position, la gestion de points d'intérêts, etc.

Ceci est rendu possible par le développement intégral des parties client et serveur. Le client est une application Android compatible avec tout dispositif (smartphone, tablette, télévision, etc.) disposant d'un système en version 12 ou supérieur. Le web service est basé sur la technologie nodeJS couplé à une base de données NoSQL MongoDB permettant la communication de données sécurisées par le protocole HTTP.

Au terme de ce projet tous les objectifs initiaux ont été atteints et de nombreuses fonctionnalités supplémentaires et de concepts originaux restent à développer. La taille du projet, ses spécifications et sa pluralité technologique en font un projet très complexe et demanderait beaucoup plus de temps pour être complet. Devant ce constat le choix judicieux a été fait en début de projet de se concentrer dans un premier temps à produire une solution primaire, élémentaire et avec seulement les fonctionnalités de bases afin d'avoir un livrable fonctionnel. Ceci a ensuite permis d'implémenter des fonctionnalités plus complexes dans des itérations agiles en assurant qu'au terme du projet nous aurions une application fonctionnelle et répondant aux critères initiaux. Ainsi nous avons pu produire WatchDogZZ avec certaines fonctionnalités supplémentaires.

De nombreux axes d'amélioration et d'évolution restent encore ouverts pour notre projet. Tout d'abord, toutes les fonctionnalités auxquelles nous avons pensé n'ont pas toutes été implémentées comme par exemple la vue en réalité virtuelle de l'établissement ou encore le calcul d'itinéraire. Cependant celles-ci restent facilement intégrables dans l'application qui possèdent tous les prérequis à leur intégration. D'un point de vue plus diffusion de l'application, deux points majeurs peuvent être améliorés. Le premier est la portabilité du client : en effet il n'est actuellement disponible que sur les appareils Android, le porter sur iOS et Windows Phone permettrait de pouvoir toucher la quasi-totalité du marché cible. Le deuxième est le passage à l'échelle du web service : les tests exécutés montrent que pour un nombre très faible d'utilisateurs les performances du service sont parfaites mais concernant un nombre d'utilisateurs plus important le comportement de notre solution nous est encore inconnu. Ces améliorations potentielles pourraient faire de WatchDogZZ une application complète et sérieuse pouvant satisfaire les cas réels présentés dans ce rapport.

## Références webographiques