



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^e 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

Résumé

La travail présenté dans ce rapport concerne l'élaboration d'une solution visant à **orienter** facilement des utilisateur au sein d'un bâtiment. Pour ceci, les utilisateurs disposent d'une **carte interactive** sur **mobile** se mettant à jour en effectuant des requêtes sur un **Service Web**. Ce travail se découpe donc en 2 parties distinctes : une partie Service Web, ainsi qu'une partie application mobile **Android**.

La partie Service Web est réalisée en utilisant le module **Express** ajouté au framework de base **NodeJS**, permettant de réaliser simplement un serveur Web. D'autres modules complémentaires viennent s'ajouter pour disposer de plus de fonctionnalités. Ce service va permettre de traiter des requêtes soumises par les clients mobiles, ainsi que de stocker des données relatives au bon fonctionnement de l'application.

La seconde partie concernant l'application Android permet à un utilisateur de se connecter au Service Web. Cette application effectuera des requêtes visant à mettre à jour la position de l'utilisateur sur le Service Web. Ainsi les applications d'autres utilisateurs seront capables de récupérer ces positions et les placer au sein d'une carte modélisant un bâtiment.

Les tests de cette solution ont pu montrer qu'il est possible d'afficher la position des utilisateurs sur une carte modélisée de l'ISIMA. Cependant, des erreurs de positionnement se retrouvent dans la position des utilisateurs, du fait de la mauvaise réception GPS par le mobile. Cette solution mise en place pourra faire l'objet d'améliorations futures telles que l'ajout d'itinéraires entre 2 utilisateurs ou vers un point d'intérêt.

Mots clés : Carte interactive, Express, NodeJS, Android, Service Web, mobile.

Abstract

English abstract

Keywords: NodeJS, Android, Web Service, mobile

Table des matières

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

Liste des figures, tableaux, algorithmes et extraits de code

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des extraits de code

2.1	Corps de la réponse serveur	2
2.2	Corps de la requête login	3
2.3	Corps de la requête who	3
2.4	Corps de la requête where GET	3
2.5	Corps de la requête where POST	4

Glossaire

Word : Definition

1 Introduction

Introduction

2 Etudes préalables

2.1 Présentation du projet

2.2 Analyse de l'existant

2.3 Spécifications du projet

2.3.1 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 2.1 – 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 2.2 – 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 courante 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 2.3 – 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 2.4 – 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 2.5 – 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

2.3.2 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.

2.3.3 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.

3 Conception de la solution

3.1 Architecture de la solution

3.1.1 Web Service

3.1.2 Application Android

3.2 Fonctionnalités introduites

3.3 Je ne sais pas

4 Résultats

4.1 Test de la solution

4.2 Améliorations possibles

5 Conclusion

Conclusion

Références webographiques

Références

- [1] NPM, INC. *Build amazing things*. Récupéré sur : <http://www.npmjs.com>. Octobre 2016 - Février 2017.
- [2] TRAVIS CI, GMBH. *Travis CI User Documentation*. Récupéré sur : <https://docs.travis-ci.com/>. Octobre 2016 - Février 2017.
- [3] TRAVIS CI, GMBH. *Travis CI - Test and Deploy Your Code with Confidence*. Récupéré sur : <https://travis-ci.org/>. Octobre 2016 - Février 2017.
- [4] GITHUB, INC. *GitHub*. Récupéré sur : <https://github.com/WatchDogZZ>. Octobre 2016 - Février 2017.
- [5] Michael KATZ. *How To Write A Simple Node.js/MongoDB Web Service for an iOS App*. Récupéré sur : <https://www.raywenderlich.com/61078/write-simple-node-jsmongodb-web-service-ios-app>. Octobre 2016 - Février 2017.
- [6] NODE.JS FOUNDATION. *Node.js*. Récupéré sur : <https://nodejs.org/en/>. Octobre 2016 - Février 2017.
- [7] MONGODB, INC. *MongoDB for GIANT Ideas / MongoDB*. Récupéré sur : <https://www.mongodb.com/>. Octobre 2016 - Février 2017.