



Soutenance de Projet Ingénieur 3ème Année

WatchDogZZ - Suivi de personnes dans les bâtiments

Benjamin BARBESANGE et Benoît GARCON

Projet de 120h
Soutenu le 21 Mars 2017

Tuteur projet : Pierre COLOMB
Référente ISIMA : Eva HASSINGER



Introduction

Contexte

- Proposition personnelle
- Domaine du tracking
- Nouvelles technologies

Introduction

Contexte

- Proposition personnelle
- Domaine du tracking
- Nouvelles technologies

Applications

- Secourisme
- Sécurité
- Optimisation de déplacements
- Analyse des déplacements dans un bâtiment

Introduction

Contexte

- Proposition personnelle
- Domaine du tracking
- Nouvelles technologies

Applications

- Secourisme
- Sécurité
- Optimisation de déplacements
- Analyse des déplacements dans un bâtiment

Objectifs

- Suivi des personnes en **temps réel**
- Architecture **simple** et **modulaire**



Plan

- 1 Présentation du projet
 - Analyse de l'existant
 - Architecture proposée
 - Spécifications
- 2 Réalisation de la solution
 - Technologies transverses
 - Technologies mobile
 - Réalisation mobile
 - Technologies serveur
 - Réalisation serveur
- 3 Résultats
 - Service à disposition
 - Client Android
 - Améliorations possibles
- 4 Conclusion



Plan

- 1 Présentation du projet
 - Analyse de l'existant
 - Architecture proposée
 - Spécifications
- 2 Réalisation de la solution
- 3 Résultats
- 4 Conclusion

Présentation du projet

Carte du Marauder Harry Potter.

Objectifs

- Visualiser un bâtiment
- Visualiser les personnes
- Opérations auxiliaires
 - Partager sa position
 - Marqueurs personnalisés
 - Itinéraires
- Données à caractère personnel

Présentation du projet

Carte du Marauder Harry Potter.

Objectifs

- Visualiser un bâtiment
- Visualiser les personnes
- Opérations auxiliaires
 - Partager sa position
 - Marqueurs personnalisés
 - Itinéraires
- Données à caractère personnel

Contraintes

- Solution évolutive
- Implémentation d'une base
- Constante amélioration (non régression)

Analyse de l'existant

Méthodes de localisation

- Intentionnelle
- Automatique
- **Open data**

Géolocalisation

- GPS intégré
- Smart*, Google Glass
- Suit les déplacements de l'utilisateur

Analyse de l'existant

Méthodes de localisation

- Intentionnelle
- Automatique
- **Open data**

Géolocalisation

- GPS intégré
- Smart*, Google Glass
- Suit les déplacements de l'utilisateur

Localiser des personnes dans des bâtiments ?

Pas de solution, nous devons créer la nôtre

Architecture proposée

Objectif

- Implémenter un service web
- Architecture en 2 parties
 - Client mobile
 - Service web

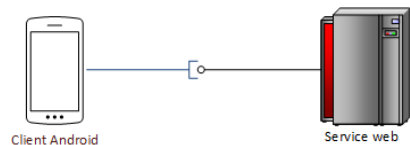


Figure : Architecture proposée

Architecture proposée

Objectif

- Implémenter un service web
- Architecture en 2 parties
 - Client mobile
 - Service web

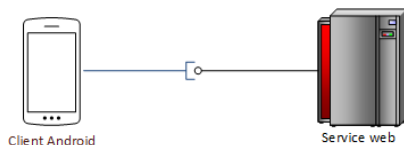


Figure : Architecture proposée

Répartition des tâches

- Benoît : Client mobile
- Benjamin : Service web

Spécifications

Service web

- REST
- Stocker et distribuer des données

Spécifications

Service web

- REST
- Stocker et distribuer des données

Fonctionnalités requises

- Connecter / déconnecter l'utilisateur du service
- Fournir la liste des personnes connectées
- Fournir la position des utilisateurs
- Mettre à jour la position d'un utilisateur

Spécifications

Service web

- REST
- Stocker et distribuer des données

Fonctionnalités requises

- Connecter / déconnecter l'utilisateur du service
- Fournir la liste des personnes connectées
- Fournir la position des utilisateurs
- Mettre à jour la position d'un utilisateur

Fonctionnalités supplémentaires

- Fournir un historique de positions
- Calculer un itinéraire
- Partager sa position (sms, mail)



Spécifications

Client mobile

- **Un client** du service
- Mobile : **portabilité**
- But : servir d'**interface** aux utilisateurs finaux

Spécifications

Client mobile

- **Un client** du service
- Mobile : **portabilité**
- But : servir d'**interface** aux utilisateurs finaux

Fonctionnalités requises

- Gérer un utilisateur
- Consommer le service
- Répondre à des critères d'**utilisabilité** et de **performances**

Spécifications

Client mobile

- **Un client** du service
- Mobile : **portabilité**
- But : servir d'**interface** aux utilisateurs finaux

Fonctionnalités requises

- Gérer un utilisateur
- Consommer le service
- Répondre à des critères d'**utilisabilité** et de **performances**

Fonctionnalités supplémentaires

- Visualiser une carte en 3D
- Visualier en réalité virtuelle
- Ajouter des informations personnalisées

Plan

- 1 Présentation du projet
- 2 **Réalisation de la solution**
 - Technologies transverses
 - Technologies mobile
 - Réalisation mobile
 - Technologies serveur
 - Réalisation serveur
- 3 Résultats
- 4 Conclusion



Technologies transverses

Travis CI

- Intégration continue
- **Tests et déploiements** automatiques

GitHub

- **Versionner** le code source
- **Planifier** et **assigner** des tâches
- Recenser les bugs



Technologies mobile

Android

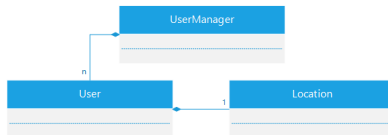
- Choix personnel
- Android studio
- Framework (version > 12)
- Gestion du réseau
 - Volley
 - Picasso
- Bibliothèque graphique : OpenGL ES



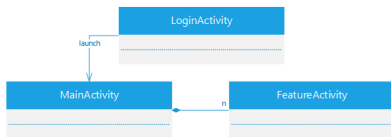
Réalisation mobile

Architecture MVC

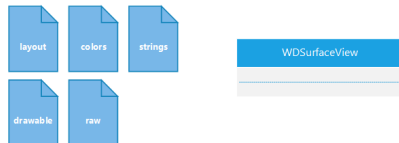
Model



Controller



View



Réalisation mobile

Composition de l'application

- 3 activités
 - Unité d'action
 - Des fragments

Communications

- Utilisation de services asynchrones
- Communication réseau

Technologies serveur

Technologies

- Framework NodeJS
- Ajout de paquets NPM
 - Express : serveur web
 - Jasmine : tests par specs
 - MongoDB : gestion de la base
 - Winston : système de log
- MongoDB : base de données



and

express



Technologies serveur

Technologies

- Framework NodeJS
- Ajout de paquets NPM
 - Express : serveur web
 - Jasmine : tests par specs
 - MongoDB : gestion de la base
 - Winston : système de log
- MongoDB : base de données



and

express

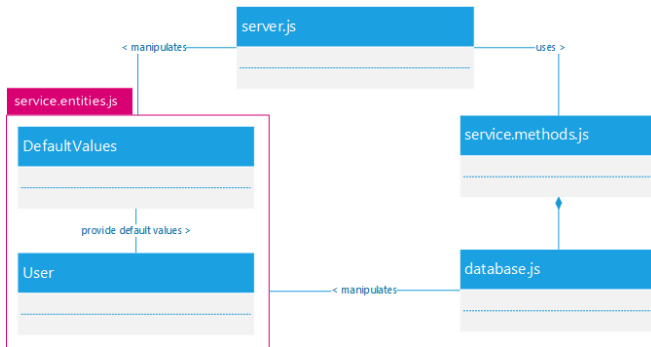


Services ajoutés

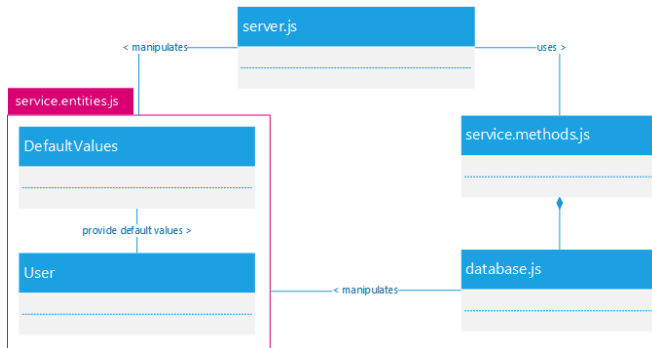
- DNS et DynDNS
- Https avec Letsencrypt
- Log d'erreurs



Réalisation serveur



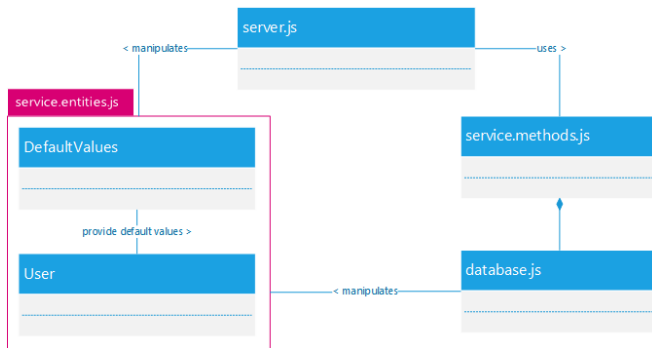
Réalisation serveur



Description

- `server.js` : code **serveur** et gestion de **requêtes**

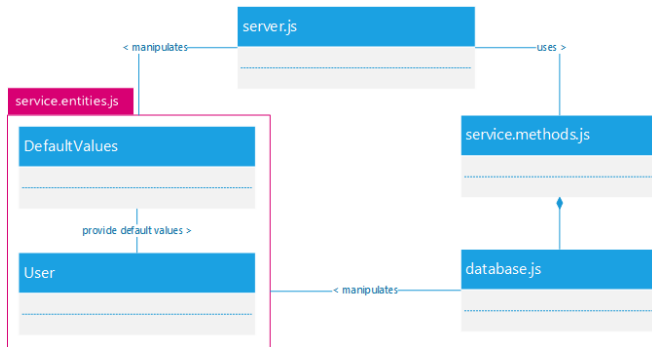
Réalisation serveur



Description

- `server.js` : code **serveur** et gestion de **requêtes**
- `service.entities.js` : gestion des **entités** et **valeurs par défaut**

Réalisation serveur

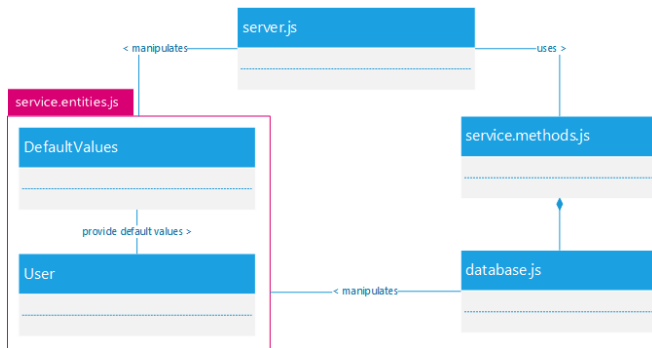


Description

- `server.js` : code **serveur** et gestion de **requêtes**
- `service.entities.js` : gestion des **entités** et **valeurs par défaut**
- `service.methods.js` : wrapper l'utilisation de la base de données



Réalisation serveur



Description

- server.js : code **serveur** et gestion de **requêtes**
- service.entities.js : gestion des **entités** et **valeurs par défaut**
- service.methods.js : wrapper l'utilisation de la base de données
- database.js : code de gestion **MongoDB**



Plan

- 1 Présentation du projet
- 2 Réalisation de la solution
- 3 Résultats**
 - Service à disposition
 - Client Android
 - Améliorations possibles
- 4 Conclusion

Service à disposition

URLs mises en place

- POST /login : se connecter
- POST /logout : se déconnecter
- GET /who : lister les personnes connectées
- POST /where : mettre à jour sa position
- GET /where : obtenir les positions des utilisateurs

Service à disposition

URLs mises en place

- POST /login : se connecter
- POST /logout : se déconnecter
- GET /who : lister les personnes connectées
- POST /where : mettre à jour sa position
- GET /where : obtenir les positions des utilisateurs

Exemple de corps de requête (POST /where)

```
{  
  "token": "1A2Z3E4R5T6Y7U8I900P",  
  "location": [  
    1.0, 2.0, 3.0  
  ]  
}
```



Client Android

Fonctionnalités

- Localisation
 - Affichage des utilisateurs connectés
 - Synchronisation de la position avec le serveur
 - Définition des points d'intérêt
 - Affichage 3D

Client Android

Fonctionnalités

- Localisation
 - Affichage des utilisateurs connectés
 - Synchronisation de la position avec le serveur
 - Définition des points d'intérêt
 - Affichage 3D
- Communication
 - Authentification Google
 - Partage de position

Client Android

Quelques écrans

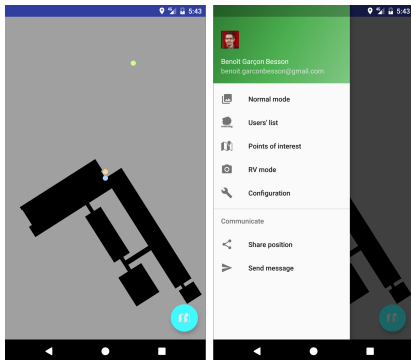


Figure : Visualisation de la carte et menu latéral

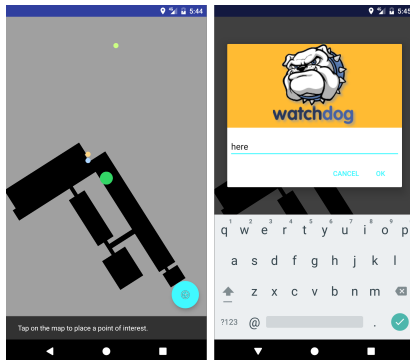


Figure : Ajout d'un point d'intérêt sur la carte



Améliorations possibles

Client

- Amélioration de l'interface
- Paramétrage de l'utilisateur
- Mode réalité virtuelle : Google cardboard
- Gestion des étages : calibration

Améliorations possibles

Client

- Amélioration de l'interface
- Paramétrage de l'utilisateur
- Mode réalité virtuelle : Google cardboard
- Gestion des étages : calibration

Serveur

- Calcul d'itinéraires
- Gestion des logs
- Alertes automatiques



Plan

- 1 Présentation du projet
- 2 Réalisation de la solution
- 3 Résultats
- 4 Conclusion

Conclusion

Rappels

- Suivi de personnes
- Service web : réceptionner et traiter des données
- Client : fournir et demander des données

Conclusion

Rappels

- Suivi de personnes
- Service web : réceptionner et traiter des données
- Client : fournir et demander des données

Points négatifs

- Client uniquement android : iOS, Windows Universal, Site web
- Déploiement sur le cloud

Conclusion

Points positifs

- Architecture modulaire
- Processus d'intégration
 - Intégration continue
 - Tests automatiques
 - Déploiement automatique



Conclusion

Points positifs

- Architecture modulaire
- Processus d'intégration
 - Intégration continue
 - Tests automatiques
 - Déploiement automatique

Perspectives

- Améliorations possibles
- Projet accessible librement sur GitHub
- Tout le monde peut y contribuer



Fin

Merci de votre attention.

