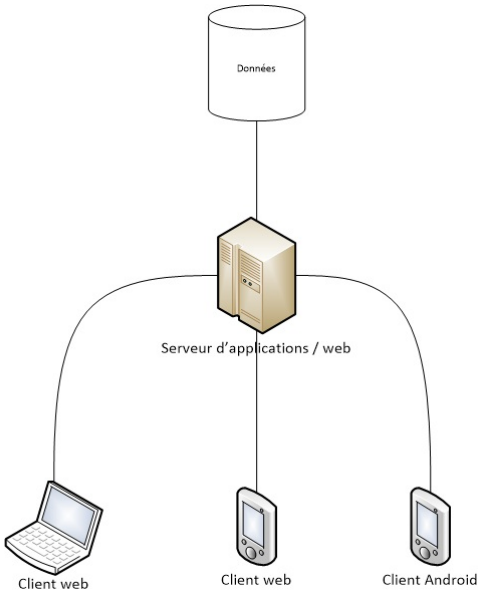
Specifications

Architecture

Nous avons choisis de traiter l'architecture du projet en trois parties. La première est la partie serveur, celle-ci sera faite en nodeJS soutenu par une base de données MongoDB. La seconde L'application android sera développé en Java et compatible pour les frameworks Android version 3.0 et supérieurs. L'interface d'administration du serveur quant à elle sera développée en AngularJS. Le serveur d'applications sera un serveur d'application compatible NodeJS sur AWS.



Partie serveur

La partie centrale est celle du Web Service. Le serveur doit etre 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 coté 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 trés 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 :

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

La plupart des requêtes décrites nécéssitent 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:

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

/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:

```
{
    'name': [
        'userName1',
        'userName2',
        'userName3'
```

```
    ]
}
```

/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 :

```
{
    'list': [
        {
            'name': 'username1',
            'location': [1.0, 2.0, 3.0]
        },
        {
            'name': 'username2',
            'location': [1.0, 2.0, 3.0]
        },
        {
            'name': 'username3',
            'location': [1.0, 2.0, 3.0]
        }
    ]
}
```

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

```
{
    'name': 'username',
    'location': [1.0, 2.0, 3.0]
}
```

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

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.

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.