

Développement d’une application mobile avec Visual Studio 2015

Rapport de stage

Version

0.9

Remise Finale

Client

C2T3

Superviseurs

Simon Boucher

Steven Goulet

Auteur

Jonathan Lafrenière

Table des matières

[1. Introduction 3](#_Toc464572178)

[1.1. Définitions, acronymes et abréviations 3](#_Toc464572179)

[2. Étude de la méthodologie de développement d’une application Web mobile à l’aide de l’IDE spécifié 3](#_Toc464572180)

[2.1. Titre Niveau 2 3](#_Toc464572181)

[2.1.1. Titre Niveau 3 3](#_Toc464572182)

[2.1.1.1. Titre Niveau 4 3](#_Toc464572183)

[3. [autre étude? Par exemple, ASP.NET] 3](#_Toc464572184)

[4. Développement de l’application Web mobile 3](#_Toc464572185)

[4.1. Contraintes rencontrées 3](#_Toc464572186)

[4.2. Problèmes rencontrés 3](#_Toc464572187)

[5. Mode d’emploi 3](#_Toc464572188)

[6. Outils de développement 3](#_Toc464572189)

[7. Bibliographie 3](#_Toc464572190)

Table des tableaux

[Tableau 1.2 : Abréviations 3](#_Toc340492962)

[Tableau 1.1 : nom du tableau 3](#_Toc340492963)

Historique du document

| Date | Auteur | Version | Sujet de la modification |
| --- | --- | --- | --- |
| 2016-10-18 | Simon Boucher | 0.0 | Création |
| 2016-10-24 | Jonathan Lafrenière | 0.1 |  |
| 2016-10-28 | Simon Boucher | 0.2 | Révision |
| 2016-10-28 | Jonathan Lafrenière | 0.3 |  |
| 2016-11-04 | Simon Boucher | 0.4 | Révision |
| 2016-11-12 | Jonathan Lafrenière | 0.5 | Début du développement |
| 2016-11-06 | Simon Boucher | 0.6 | Révision |
| 2016-11-23 | Jonathan Lafrenière | 0.7 |  |
| 2016-11-28 | Simon Boucher | 0.8 | Révision |
| 2016-12-11 | Jonathan Lafrenière | 0.9 |  |
| 2016-12-12 | Simon Boucher | 0.10 | Révision |
| 2016-12-27 | Jonathan Lafrenière | 0.11 | Remise Finale |

# Introduction

## Définitions, acronymes et abréviations

| Termes | Définitions |
| --- | --- |
| AJAX | Asynchronous JAvascript and Xml, architecture informatique permettant de mettre à jour une page en temps réel |
| API | Ensemble de classes et de méthodes et de fonctions par lequel un logiciel rend service à un autre logiciel. |
| Application Web mobile | Application Web hébergée sur un serveur, mais exécutée sur un périphérique mobile |
| Bluetooth Low Energy | Protocole de communication radiofréquence à faible consommation énergétique |
| BootStrap | Framework permettant d’adapter la présentation d’un site Web au périphérique sur lequel il est présenté (Mobile, Ordinateur, Tablette etc.. ) |
| C# | Langage de programmation orienté objet |
| CDN | Content Delivery Network : Contenu pertinent à l’application hébergé sur un autre serveur |
| Code-First | Approche permettant de générer la structure de la base de données à l’aide du modèle décrit par le code de l’application |
| DAL | Data Access Layer (Couche d’accès aux données), classe permettant l’interaction entre un contrôleur et le modèle dans une application MVC |
| Développement Front-End | Partie du développement n’impliquant que la partie visible à l’utilisateur (Pages HTML, CSS, JavaScript modifiant le DOM) |
| Développement Back-End | Partie du développement n’impliquant que le code exécuté sur le serveur (Application ASP.Net, Gestion de la base de données, SQL) |
| DOM | Document Object Model, Modèle selon lequel une page HTML ou XML est structuré. |
| Entity |  |
| GATT | Generic Attribute Profile; Protocole permettant le transfert d’information entre deux périphériques Bluetooth |
| GPS | Global Positionning System |
| IDE | Environnement de développement intégré |
| JS | JavaScript |
| JQuery | Framework Javascript permettant de grandement réduire l’écriture de code à l’aide de sélecteurs (identiques à ceux utilisés en CSS) pour modifier les attributs du DOM |
| MVC | Modèle-Vue-Contrôleur |
| Package NuGet | Package donnant accès à différentes fonctionnalités de développement sur Visual Studio |
| Périphérique mobile | Tablette ou téléphone intelligent |
| RSSI | Received Signal Strength Indication |
| SensorTag |  |
| SignalR |  |
| TI | Texas Instruments |
| Visual Studio | IDE de Microsoft |
| Visual Studio Online | Logiciel de collaboration et de gestion de versions |

Tableau 1.1 : Abréviations

| Date | Résumé de l’avancement hebdomadaire |
| --- | --- |
| 2016-10-22 | * Étude de la méthode de développement MVC * Familiarisation avec l’environnement de développement Visual Studio 2015 dans le cadre d’un projet ASP.Net MVC * Étude sur l’utilisation d’Entity Framework * Recherche sur la faisabilité des éléments demandés dans l’application finale |
| 2016-10-29 | * Recherche sur les méthodes de communication entre un serveur et une application mobile, que ce soit par le biais d’une application native ou d’une application Web * Étude de la méthode de développement MVC * Recherche en rapport aux Frameworks et APIs pouvant se montrer pertinent au développement de l’application Web * Préparation au développement de l’application * Recherche sur les méthodes sécuritaires d’exécuter des scripts ASP.Net à l’aide de requêtes AJAX |
| 2016-11-03 | Développement “Front-End”   * Recherche sur l’utilisation de l’API JavaScript Google Maps sur un navigateur * Recherche sur la géolocation * Développement d’une application pour me familiariser avec l’API Google Maps * Création de la mise en page de l’application et de différents composants à l’aide de BootStrap |
| 2016-11-10 | Recherche sur l’aspect “Back-End” du projet   * Recherche sur le moteur de base de données SqlLocalDb * Établissement de la connexion entre l’application ASP.Net et la base de donnée locale * Recherche sur la génération de la base de données à partir du modèle de l’application à l’aide d’Entity Framework (Approche Code-First) * Ajout de la partie développée la semaine passée aux vues de l’application ASP.Net |
| 2016-11-17 | * Affichage des positions enregistrées en base de données sur un tableau * Suppression de positions enregistrées en base de données * Visualisation des positions enregistrées en base de données sur la carte Google Maps * Réfactorisation du code existant à l’aide d’AJAX et JQuery |
| 2016-11-24 | * Réfactorisation du code à l’aide du modèle MVC   + Création de contrôleurs   + Fichier unique pour chaque classe du modèle * Ajout d’une classe utilisateur au modèle * Suppression des éléments redondants ou inutiles dans le code * Déplacement des scripts JavaScript vers un fichier séparé dans le répertoire « Script » du projet. * Ajouts de commentaires au code |
| 2016-12-01 | * Sélection d’appareils dans une liste générée à partir des appareils enregistrés en base de données * Enregistrement d’appareils * Exploration par rapport à l’encryption, aux sessions et à la connexion d’utilisateurs * Création d’utilisateur * Rôles d’utilisateurs (Admin/Employé) |
| 2016-12-08 | * Connexion d’utilisateurs * Gestion des droits * Restriction des actions aux utilisateurs ne possédant pas de droits suffisants ou visitant le site. * Préparation en vue de la présentation |
| 2016-12-15 | * Finalisation du rapport * Présentation * Correction de bugs faisant planter l’application si elle est exécutée dans un environnement où la langue par défaut est le français (points et virgules) * Création du manuel d’utilisation |

Tableau 1.2 : Journal d’avancement hebdomadaire

# Étude de la méthodologie de développement d’une application Web mobile à l’aide de l’IDE spécifié

Mon étude a porté sur le patron d’architecture logicielle MVC, qui permet de développer des applications structurées, faciles à entretenir et à tester. J’ai eu l’occasion d’étudier le développement d’une application ASP.Net MVC lors d’un exercice pratique comprenant Entity Framework et un projet de tests unitaires.

## Entity Framework

Entity Framework est un package[[1]](#footnote-2) permettant le développement d’applications où la base de données est représentée concrètement dans le code, par des classes, des clés primaires et étrangères et des contraintes, ce qui facilite la compréhension et les tests.

## Avantages et désavantages propre à l’utilisation d’une application Web

### Accès aux fonctionnalités de l’appareil

Une application Web n’a accès aux fonctionnalités (ex. Géo positionnement) que par le biais du navigateur. Le navigateur, étant une application installée, bénéficie de permissions lui permettant d’accéder à certaines de ces fonctionnalités, mais pas toutes. Par exemple, très peu de navigateurs donnent accès au Bluetooth depuis le navigateur Web (Chrome est le seul, et la fonctionnalité est expérimentale), alors que là, tous les navigateurs donnent accès au géo positionnement si l’utilisateur le permet. L’utilisation d’une application Web nous limite donc aux fonctionnalités incluse sur le navigateur de l’utilisateur, ce qui constitue le plus grand désavantage de l’utilisation d’une application Web par rapport à une application native. Tous les éléments demandés initialement ne sont pas réalisables si une application Web seule est utilisée sur le périphérique mobile.

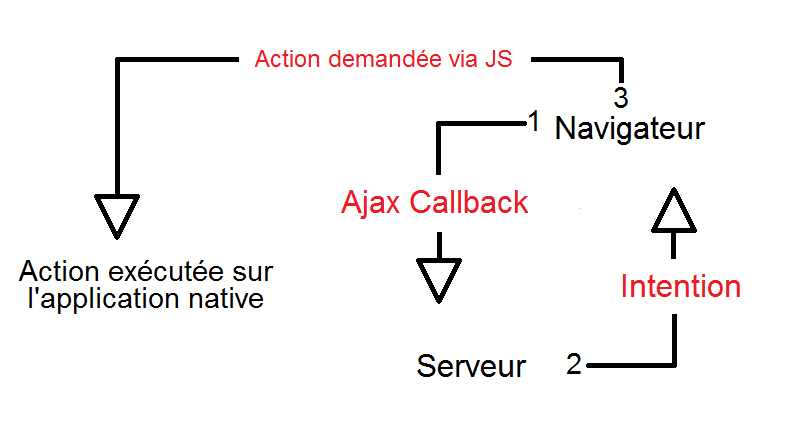
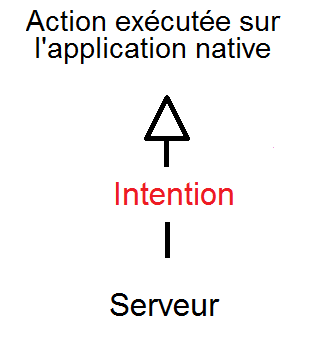
Il est cependant possible pour une application Web de communiquer une demande à une application native, qui aura un accès plus étendu aux fonctionnalités de l’appareil mobile. Le schéma suivant représente la communication entre le serveur, l’application Web et l’application mobile permettant d’interagir avec des fonctionnalités de l’appareil.

Schéma 1.1 : Interactions nécessaires à l’interaction d’une application Web avec les fonctionnalités d’un appareil mobile

Cette méthode permet deux choses, l’exécution de certaines actions lorsque l’utilisateur est sur le site Web et l’envoi “d’ordres” à l’application depuis le serveur en temps réel. Il n’est pas optimal de fonctionner ainsi, la communication entre le serveur et une application native est plus simple, comme le démontre le schéma suivant.

## Étapes importantes au développement d’une application similaire

### Création du projet

Lors de la création du projet, il est important de sélectionner “Application ASP.Net”, de choisir le modèle vide et de cocher “Inclure un projet de tests unitaires” et “MVC”.

### Packages NuGet

L’utilisation d’une base de données “Model First” nécessite l’installation du package **Entity Framework**. La base de données utilisée lors du développement est la base de données Microsoft SQL locale. Il m’a été difficile de m’y connecter initialement car je me référais à un tutoriel désuet. Il est nécessaire d’installer SqlLocalDb[[2]](#footnote-3) sur le poste pour pouvoir exécuter l’application. Il est aussi important de posséder une version du Framework .Net égale ou supérieure à 4.6[[3]](#footnote-4).

### Indiquer l’emplacement de la base de données à l’application

Pour ce faire, il faut ajouter la ligne suivante au fichier Web.Config[[4]](#footnote-5)

<connectionStrings>

<add name="BddContext" providerName="System.Data.SqlClient" connectionString="Data Source=(localdb)\mssqllocaldb;Initial Catalog=WEB\_APP\_C2T3.Models.BddContext;Integrated Security=True;MultipleActiveResultSets=True;Application Name=EntityFrameworkMUE" />

</connectionStrings>

# Étude relative au projet

## Étude de faisabilité technique

### Interface permettant d’afficher les champs suivant : coordonnées GPS (latitude et longitude), température ambiante, luminosité ambiante et l’intensité du signal cellulaire (RSSI).

Toutes ces informations sont accessibles depuis un appareil mobile, cependant, le navigateur internet ne permet pas l’accès à toutes ces données. Une application native mobile, développée grâce à Android ou Xamarin, serait plus appropriée pour cette tâche.

### Extraire la luminosité ambiante de la base de données, et l’afficher.

La lecture de données enregistrées sur une base de données ne pose aucun obstacle au développement de l’application.

### Contrôler un composant physique du périphérique mobile

Une application Web mobile ne peut pas contrôler un composant physique puisque l’application est exécutée par un navigateur Web qui n’offre pas cette fonctionnalité. Une application native mobile permet de contrôler un composant physique. Une solution hybride Web et native pourrait être une alternative fonctionnelle.

### Contrôler le composant GPS du périphérique mobile

#### Extraire les coordonnées GPS à une période d’une seconde, les afficher et les ranger dans la base de données

Cet élément est facilement réalisable sur mobile du moment que le positionnement GPS soit activé au préalable. La position d’un appareil mobile connecté au réseau cellulaire peut être évaluée très précisément dans un rayon allant de 5 à 30 mètres. Le positionnement d’un ordinateur est, cependant, fortement imprécis. L’outil étudié est Google Maps pour l’instant.

#### Activer le GPS à l’ouverture de l’application

Le navigateur web d’un appareil mobile ne peut pas forcer l’appareil à activer le positionnement GPS pour des raisons de sécurité. Il est possible de demander à un utilisateur d’activer le positionnement par le biais d’une application installée, mais pas par le biais d’une application WEB. Se référer à la section 3.1.3.

#### Désactiver le GPS à la fermeture de l’application

Se référer à la section 3.1.3.

### Contrôler le composant Bluetooth du périphérique mobile

Se référer à la section 3.1.3. L’établissement d’une connexion Bluetooth par un navigateur WEB n’est pas très répandu, cependant Google Chrome propose une fonctionnalité expérimentale sur Mac, Linux, Chrome OS et Android le permettant (Web Bluetooth).

FONCTIONNALITÉ EXPÉRIMENTALE

Pour activer la fonctionnalité :

* Ouvrir le navigateur Google Chrome
* Taper dans la barre d’URL « chrome://flags/#enable-web-bluetooth»
* Cliquer sur « Enable »

#### Établir une connexion avec le capteur Bluetooth

Se référer à la section 3.1.3. Aussi possible via le navigateur web Google Chrome.

#### Extraire la température à la période imposée par le capteur, l’afficher et la ranger dans la base de données

Possible grâce au Protocole GATT.

#### Activer le Bluetooth à l’ouverture de l’application

Impossible à cause du navigateur. Seule une application installée possède les permissions nécessaires pour demander à l’utilisateur d’ouvrir ou fermer le Bluetooth. Se référer à la section 3.1.3.

#### Désactiver le Bluetooth à la fermeture de l’application

Impossible à cause du navigateur. Seule une application installée possède les permissions nécessaires pour demander à l’utilisateur d’ouvrir ou fermer le Bluetooth. Se référer à la section 3.1.3.

### Contrôler le composant cellulaire du périphérique mobile

#### Extraire l’intensité du signal (RSSI), l’afficher et le ranger dans la base de données

Il est possible d’obtenir cette information par le biais d’une application installée, cependant un navigateur WEB n’y a pas accès. Se référer à la section 3.1.3.

### Démontrer la fonctionnalité de portabilité d’application d’un type de périphérique à un autre que prétend offrir l’environnement de développement. Porter de périphérique mobile à ordinateur conventionnel

Cet aspect du projet peut être facilement géré grâce à l’utilisation du Framework BootStrap[[5]](#footnote-6).

### Ajout d’un bouton permettant d’envoyer une alerte au serveur.

Facilement réalisable.

### Envoyer des SMS automatiquement

Cette fonctionnalité est accessible à une application native, cependant il n’est pas possible d’envoyer, de recevoir ou de lire des SMS depuis le navigateur Web. Un serveur ne peut pas envoyer de SMS sauf s’il utilise un API[[6]](#footnote-7) (souvent payant) lui permettant de le faire. Il serait cependant possible d’envoyer des SMS en effectuant une requête au serveur (à condition qu’un API permettant l’envoi de SMS y soit installée).

## Étude sur les pratiques sécuritaires d’exécution de méthodes ASP.Net grâce aux requêtes AJAX

Il est important de limiter les actions de l’utilisateur à certaines méthodes et fonctions pour éviter que la sécurité de l’application soit compromise. Pour ce faire, lorsqu’un fragment de code ASP.Net doit être exécuté suite à une action de l’utilisateur sur le site, il est important de l’appeler ainsi :

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js" type="text/javascript"></script>

<script type = "text/javascript">

function ShowCurrentTime() {

    $.ajax({

        type: "POST",

        url: "CS.aspx/GetCurrentTime",

        data: '{name: "' + $("#<%=txtUserName.ClientID%>")[0].value + '" }',

        contentType: "application/json; charset=utf-8",

        dataType: "json",

        success: OnSuccess,

        failure: function(response) {

            alert(response.d);

        }

    });

}

function OnSuccess(response) {

    alert(response.d);

}

</script>

Ce fragment de code AJAX est executé à l’aide de JQuery, il est donc important d’inclure une référence au CDN dans le code.

# Développement de l’application Web mobile

Le début du développement porte surtout sur la portabilité de l’application sur un appareil mobile. Une bonne quantité de mon temps a été investi dans la compréhension du Framework Bootstrap qui une fois mis en place dans l’application accélère grandement le développement d’une accélération utilisée sur différents supports. Le principe de base du Framework est de permettre à l’utilisateur de définir à la page si elle doit empiler ou rétrécir un élément en fonction de la taille de l’appareil. L’utilisation de Bootstrap est simple et le Framework peut être invoqué complètement à l’aide de trois liens vers des CDN[[7]](#footnote-8).

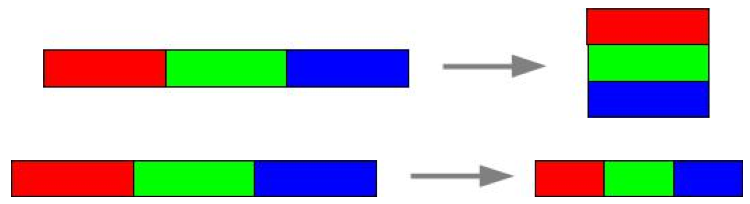


Schéma 4.1 : Empilement vs Réduction

Une fois la disposition des éléments réalisée à l’aide du Framework, je me suis lancé dans la conception de l’architecture de l’application à l’aide du modèle MVC. Cette application étant mon premier projet de taille en ASP.Net MVC, je dû m’arrêter à plusieurs étapes pour vérifier si mon application était conçue de manière performante et fidèle au modèle de développement MVC. Par exemple, je n’ai utilisé AJAX que très peu lors de la première phase de conception, et les contrôleurs étaient appelés de manière assez chaotique dans le code. La réfactorisation du code à l’aide d’appels AJAX et JQuery a grandement simplifié le code et épuré le fichier. La première écriture du code a débouché sur une application fonctionnelle, cependant toutes les actions étaient exécutées à l’aide d’un contrôleur universel et toutes les classes utilisées étaient regroupé dans le fichier « Modele.cs ». La réécriture du code a eu pour but de créer un contrôleur par objet du modèle pour faciliter la compréhension du code et la navigation à l’intérieur de la solution. Tous les scripts JavaScripts ont aussi été déplacés dans le dossier « scripts » de la solution pour réduire la quantité d’information dans les fichiers de vues, qui sont surchargés autrement.

## Contraintes rencontrées

(Ne sait pas quoi mettre ici)

## Problèmes rencontrés

### Bootstrap

L’utilisation de Bootstrap vient avec son lot de frustrations. Parfois, l’empilement de composants sur la page se comporte de manière inattendue. Il est aussi parfois complexe de définir des styles différents pour des appareils de taille différente, par exemple, centrer un élément sur mobile et justifier à gauche sur un ordinateur régulier.

### ASP.Net MVC

Le langage de programmation utilisé (C#) m’est familier et je possède certaines connaissances du modèle MVC. Mon plus grand problème repose dans la connexion entre l’application et la base de données. Beaucoup de temps y a été investi mais la connexion a finalement été établie. Un autre des problèmes rencontrés portait sur les tests d’interfaces mobiles, problème résolu grâce à la console développeur de Google Chrome (accessible via F12) qui permet de visualiser le site comme il serait vu depuis différents appareils. Le fait de travailler avec différentes technologies qui ne me sont pas familières est stimulant, mais me ralentit grandement car chaque étape me demande une mise à niveau de mes connaissances.



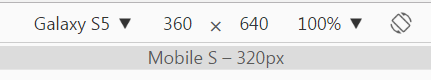


Schéma 4.1 et 4.2 : Visualiser un site web depuis un émulateur mobile

### SqlLocalDb

En raison du tutoriel désuet sur lequel je me fiais, beaucoup de temps a été perdu en tentant de connecter l’application à la base de données SQL locale qui se nommait (localdb)\v11.0 antérieurement et qui se nomme désormais (localdb)\mssqllocaldb. Ce problème me bloquait presque tout avancement et constituait le goulot d’étranglement du projet, car il ne permettait pas la connexion entre l’application et la base de données.

### Connection refused: localhost:XXXXX asp net

Ce message d’erreur s’affiche parfois lorsqu’une page de l’application est ouverte par un utilisateur. Pour résoudre le problème, il faut ouvrir le fichier contenant le projet, naviguer dans le dossier « .vs / config» et supprimer le fichier « applicationhost.config ». Ce fichier sera régénéré une fois l’application recompilée.

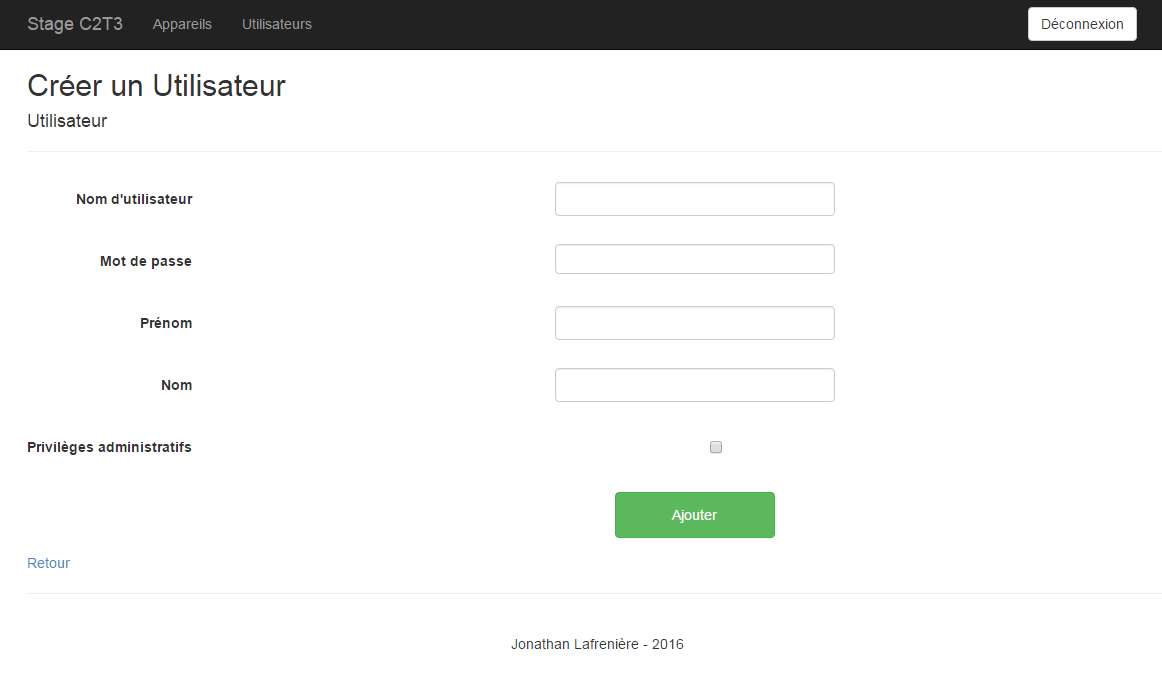
# Manuel d’utilisation

Pour avoir accès à l’entièreté de l’application, il faudra se connecter en tant qu’administrateur. Pour ce faire, un compte est généré au lancement de l’application dans le fichier WebApiConfig.cs.

Nom d’utilisateur : administrateur

Mot de passe : admin

Il sera par la suite possible d’ajouter d’autres utilisateurs via l’onglet « utilisateur » dans la barre de navigation.



Le mot de passe sera encrypté en base de données.

L’application permet d’enregistrer la position de différents appareils sur demande ou à intervalle régulier. Pour ajouter un appareil à la base de données, il faudra être administrateur et sélectionner l’onglet « Appareils ».

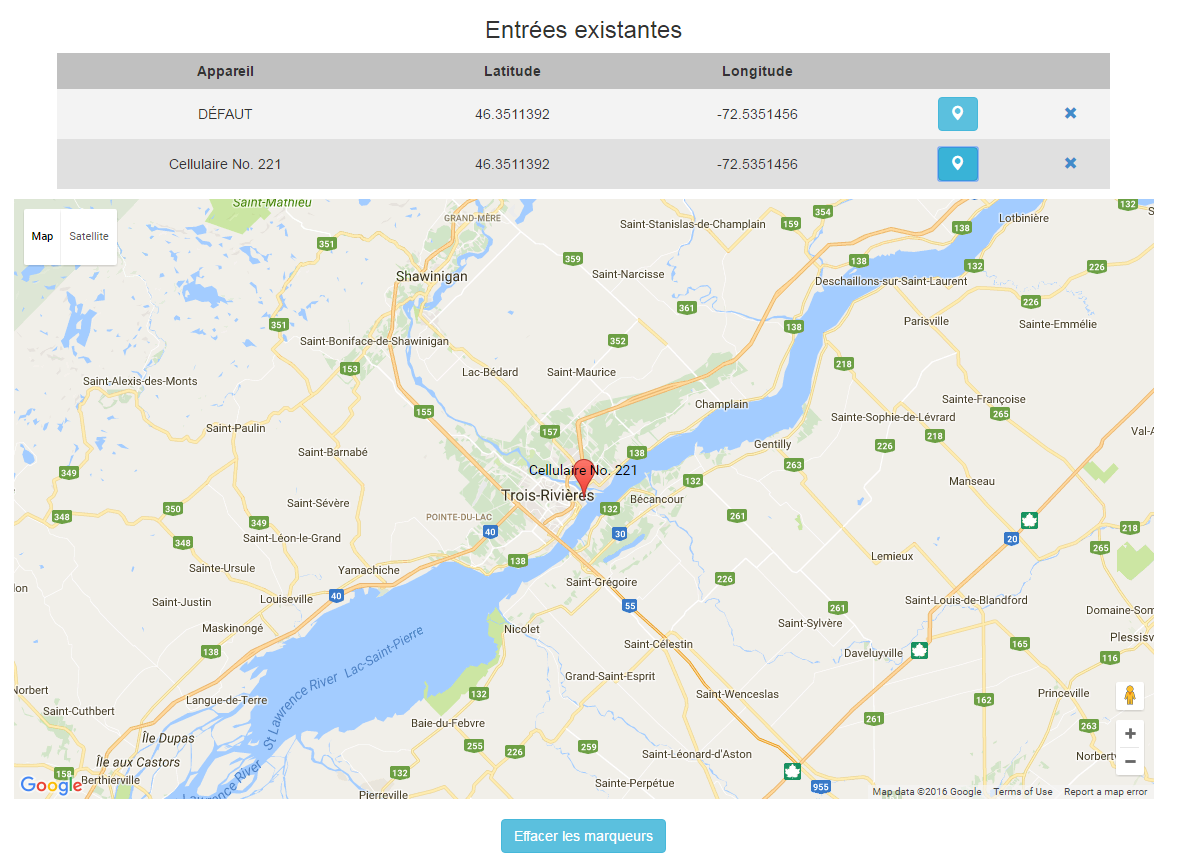


Une fois l’appareil enregistré, il sera possible de l’utiliser sur la page principale. Pour sélectionner un appareil, le sélectionner dans la liste déroulante sous l’étiquette  « Appareil ».



Le bouton enregistrer en BD enregistrera les coordonnées des champs latitude, longitude, l’heure actuelle et l’appareil actuel. Le bouton « Obtenir ma position » remplira les champs latitude et longitude avec la position de l’utilisateur estimée à l’aide du navigateur.

Une fois la position enregistrée, elle apparaitra dans la liste. Il sera possible de la visualiser sur la carte, un marqueur portant le nom de l’appareil sera placé.

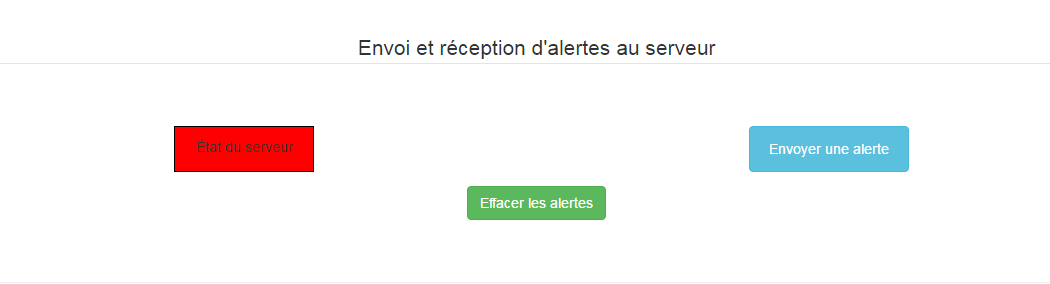


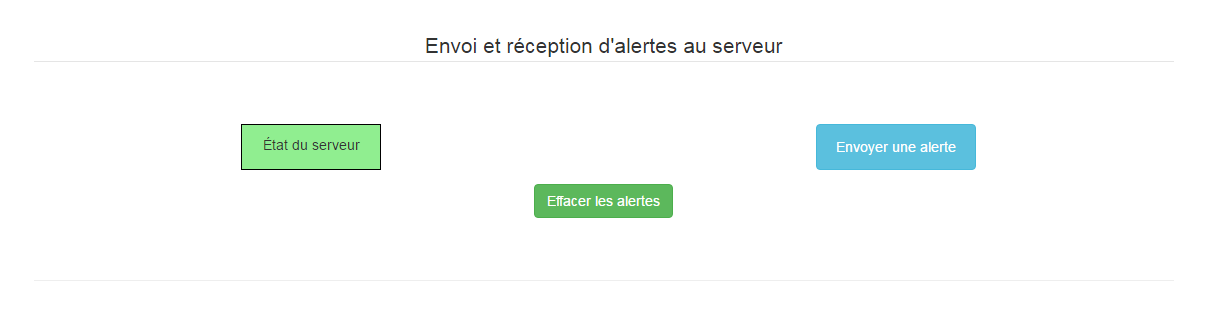
Le bouton effacer les marqueurs retirera tous les marqueurs de la carte.

Le bouton « Démarrer le tracking » enregistrera la position de l’appareil sélectionné sur la page à un interval de 5 secondes jusqu’à ce que l’utilisateur réappuie sur le bouton ou ferme la page.



Il est possible d’envoyer une alerte au serveur. Dans le cas où le serveur est en état d’alerte, le rectangle « État du serveur » sera rouge. Dans le cas où aucune alerte ne se situe sur le serveur, le rectangle sera vert. Il est possible de supprimer toutes les alertes sur le serveur à l’aide du bouton « Effacer les alertes » et d’envoyer une alerte au serveur grâce au bouton « Envoyer une alerte ».





1. https://msdn.microsoft.com/en-us/library/ee712907(v=vs.113).aspx [↑](#footnote-ref-2)
2. <https://www.microsoft.com/en-us/download/details.aspx?id=52679> – Sélectionner localdb [↑](#footnote-ref-3)
3. https://www.microsoft.com/en-ca/download/details.aspx?id=48130 [↑](#footnote-ref-4)
4. http://www.dotnetfunda.com/articles/show/3298/using-mysql-with-entity-framework-in-aspnet-mvc [↑](#footnote-ref-5)
5. <http://getbootstrap.com/2.3.2/scaffolding.html> Exemple de site utilisant un design s’adaptant aux périphériques sur lequel il est ouvert [↑](#footnote-ref-6)
6. Tinyurl.com/APISMS [↑](#footnote-ref-7)
7. https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css

   https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css

   https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js [↑](#footnote-ref-8)