Alerte Intrusion

Projet Ei4 AGI

Colin GACHET – Adrien PERRIER – Félix DELAUNAY

Année 2016 - 2017

Introduction

Nous sommes 3 étudiants en 4ème année à l'ISTIA (École d'ingénieur de l'université d'Angers) spécialisés en Automatique et Génie Informatique. Dans ce cursus il nous est imposé de réaliser un projet lié au monde de l'Automatique, de l'automatisme, et/ou de l'informatique. Sur ceux proposés par les enseignants de l'ISTIA, nous avons tous les 3 décidés de nous consacrer au projet "Alerte Intrusion" de Mehdi LHOMMEAU.

Ce projet a été réalisé entre Décembre 2016 et Avril 2017 sur une durée totale de 80 heures. Nous allons vous présenter notre travail en trois grandes parties, tout d'abord en expliquant le contexte de ce projet et de comprendre quel est son intérêt. En deuxième temps nous allons vous présenter le projet en lui-même, à savoir les différentes technologies utilisées ainsi que nos phases de développement. Pour finir nous ferons un bilan de nos travaux pour faire un état des lieux de ce qui a été effectué, mais aussi pour proposer des suites à notre projet qui n'est pas terminé.

Nous tenons à remercier Monsieur LHOMMEAU qui s'est rendu disponible pour répondre à toutes nos questions, ainsi qu'à Monsieur CHARBONNIER qui nous a aidé dans notre gestion du projet.

SOMMAIRE

Table des matières

1- PRESENTATION DU PROJET

a- Contexte

Depuis Janvier 2015, la France est sous le plan “Vigipirate”, un dispositif français de lutte contre le terrorisme. Ce plan vise à contribuer à la vigilance, à la prévention et à la protection contre la menace terroriste de l’Etat, des collectivités territoriales, des entreprises publiques et privées et des citoyens. Comme propriétés de l’Etat, les écoles primaires doivent également prendre des mesures pour prévenir de ces risques. C’est dans ce contexte qu’Hugues LECOMTE (Directeur de l’école Robert Desnos) a contacté notre enseignant Mehdi LHOMMEAU pour un projet d’alerte Intrusion pour son établissement.

En effet, à ce jour, le seul moyen pour prévenir les enseignants présents dans l’enceinte du bâtiment d’une intrusion suspecte, est l’utilisation d’un clairon par le directeur. Ce système est peu efficace pour plusieurs raisons, tout d’abord le déclencheur de l’alarme s’expose à l’intrus en soufflant dans le clairon, ce qui le met directement en danger. D’autre part, le son du clairon peu affoler les élèves qui connaissent sa signification, ce qui complique la tâche de l’enseignant qui doit essayer de calmer un maximum ses élèves.

b- Livrables attendus, nos objectifs

L’objectif de ce projet est de réaliser une application permettant de signaler une intrusion dans un établissement scolaire. Le signalement de cette intrusion doit permettre aux enseignants de l’établissement de prendre les mesures nécessaires afin de mettre les enfants de la classe, hors de danger. Le signalement pourra être réalisé de plusieurs manières, à l’aide d’une application mobile, d’un procédé physique connecté (bouton, ...). Une fois le signalement réalisé, les ordinateurs de chacune des classes devront avertir les enseignants qu’il y a une intrusion dans l’école afin qu’ils puissent prendre les mesures nécessaires.

Ces objectifs ont été proposés par notre tuteur M. LHOMMEAU. En plus de ceux-ci, il nous a été confirmé que nous avions carte blanche sur ce projet, et que nous pouvions ajouter toutes les fonctionnalités que nous voulions. Ainsi après un brainstorming lors de la première séance nous nous sommes donné les objectifs suivants :

- Création d’une application facile à déployer.
- Déclenchement de l’Alerte depuis l’application
- Affichage des mesures de sécurité
- Affichage du plan du bâtiment avec la localisation du déclenchement de l’alerte
- Page administrateur (gérer les adresses IP, le plan du bâtiment, le nombre de salle, …)

Pour répondre aux attentes du directeur de l’école primaire, nous lui avons envoyé un mail avec toutes nos idées sur le projet, ainsi que des questions pour que notre application corresponde au maximum à ce que souhaite le directeur. L’ensemble des questions est disponible en annexe. Malheureusement, le questionnaire n’a pas eu de réponse, nous sommes donc restés sur nos idées.

A terme il aurait été souhaitable d’intégrer notre application dans une école primaire, et de la rendre exploitable.

Voici le découpage des différents sprints de nos principaux travaux de développement, à savoir la page principal et la page Alarme déclenchée. Ces deux pages nécessitent une bonne

communication entre le serveur et le client, c'est pourquoi le plus gros de notre travaille s'est focalisé sur celles-ci.

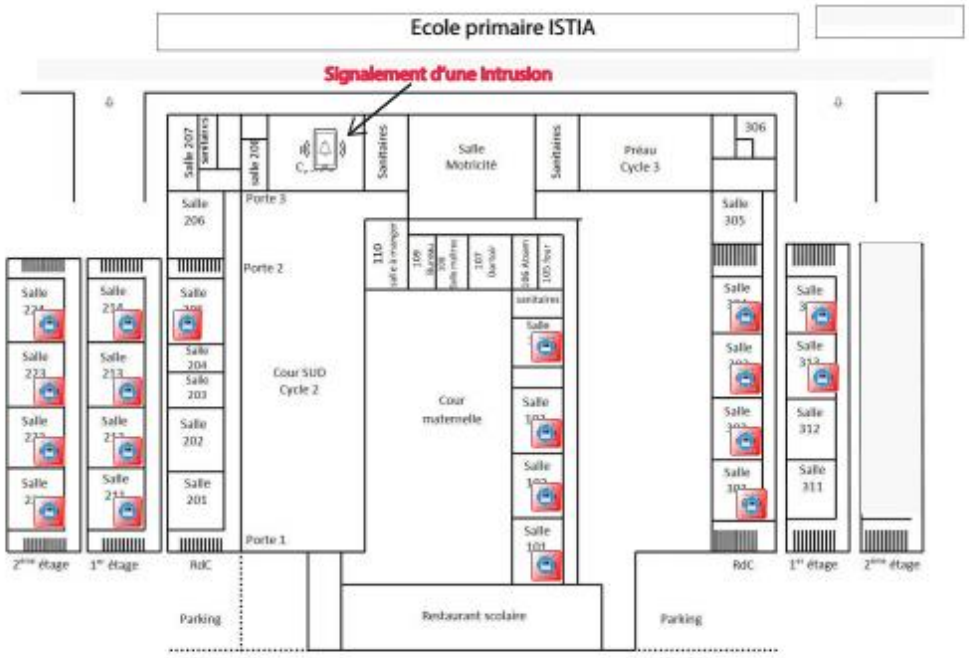


Figure 1 *Exemple de rendu de l'application*

c- Organisation du temps de travail

Pour nous organiser et nous aider à être le plus efficace possible, nous avons eu des cours de gestion de projet enseigné par M. Guillaume CHARBONNIER. Suite à cela deux méthodes de travail nous ont été proposées : une gestion de projet en mode traditionnel ou une gestion de projet en mode agile. Nous avons décidé de suivre une méthode agile qui se prêtait mieux à notre projet. En effet dès le départ nous avions une idée générale de notre produit finale, avec différentes fonctionnalités, mais celles-ci se sont vues modifiées au cours du temps.

Pour chaque tâche nous avons travaillé au fur et à mesure en scindant notre travail en différents sprints. En effet nous avons travaillé par étape, la validation du sprint précédent était nécessaire avant de passer à l'étape suivante, ceci nous a permis d'adapter nos objectifs en fonction de ce que nous avons réalisé.

Pour se répartir le travail, chaque membre du projet avait un sprint à effectuer avant de se voir en affecter un autre. Chacun a eu une ou des tâches principales à sa charge. Colin est resté focalisé sur la page principale et le serveur, qui demandaient le plus d'efforts, c'est pourquoi Adrien et Félix ont participé à certains sprints. En plus de cela, Adrien lui, s'est attaqué à la Raspberry PI, tandis que Félix a travaillé sur la librairie Electron.

Pour illustrer cette organisation, nous vous présentons ci-dessous cette méthode agile avec nos deux travaux qui nous ont demandé le plus de développement : la page Principale ainsi que la page Alarme déclenchée.

Page Principal :

Page html qui affiche un bouton et qui permet d'envoyer un message au serveur, qui lui va envoyer un message à tous les clients

La page html propose deux boutons, un qui permet d'envoyer un message au serveur et qui permet de faire clignoter du texte, et un autre bouton qui permet de stopper le clignotement.

Le bouton d'"Alarme" permet d'afficher une autre page ("page alarme déclenchée") avec les consignes de sécurité à suivre.

Envoie de l'adresse IP du client qui appuie sur le bouton vers le serveur, qui lui va renvoyer l'adresse IP à tous les clients

Ecriture et lecture dans un fichier texte qui va lier les adresses IP aux noms des salles.

Intégration d'une interface graphique à notre page principale.

Page Alarme déclenchée :

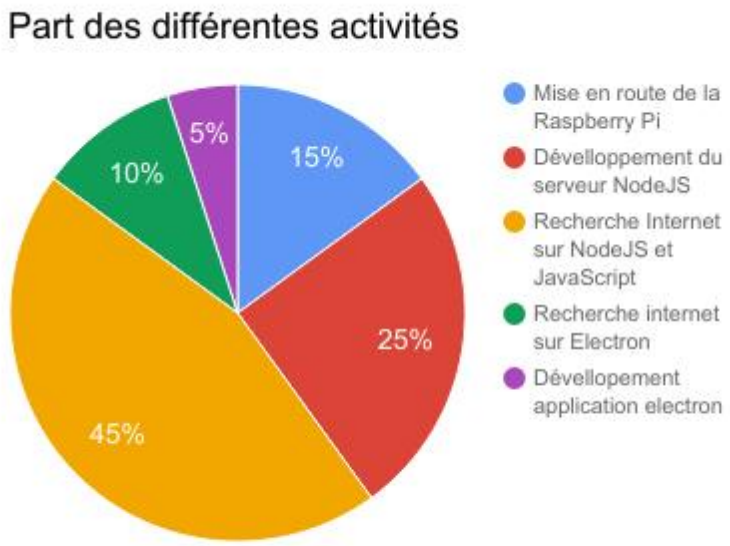
Accéder à la "page alarme déclenchée" depuis la page principale.

Ajouter un bouton qui stoppe l'alarme pour tout le monde.

Récapitulatif du calendrier de notre projet :

Tâches	Décembre		Janvier		Février			Mars			Avril	
Apprendre à utiliser Node JS												
Apprendre à utiliser Electron												
Faire la Page "Principale" / serveur node JS												
Faire Page “Administrateur”												
Faire la Page "Alarme déclenchée"												
Faire fonctionner la Raspberry Pi												
Création de l'application sous Electron												
Implémentation du serveur sur la RaspBerry et Test sur le réseau de l'ISTIA												

Après ces 80 heures de projets, nous avons constaté que nous avons passé le plus gros de notre temps à faire des recherches pour savoir comment fonctionne les différents outils auquel nous étions confrontés, à savoir NodeJS, Electron, le langage JavaScript, … . Vous pouvez voir le graphique circulaire ci-dessous qui représente la part des différentes activités de notre projet.



2- DESCRIPTION DU PROJET

Pour permettre une communication entre chaque ordinateur de l'établissement, nous avons créé un serveur auquel chaque terminal (client) peut se connecter. Pour cela chaque ordinateur sera équipé d'une application exécutable qui permettra de se connecter à la page web hébergée sur notre serveur. Nous allons décrire ces différentes parties, à commencer par la partie serveur/client qui est géré par Node.js, puis nous présenterons la Raspberry PI qui héberge notre serveur, et nous finirons par l'application Electron.

a- Serveur (nodeJS)

Notre application est constituée d'un côté serveur (Node.js) et d'un côté client (page web : html, css, js). Le choix de la technologie utilisée s'est fait sur les conseils de notre tuteur, Node.js nous permettant de créer un serveur rapide répondant à nos besoins tout en faisant découvrir un nouveau langage.

Toutes les interactions entre serveur et clients sont basées sur le système d'événements de Node.js. C'est à dire que soit le serveur soit un client va émettre un signal (un événement) et à réception de celui-ci le serveur ou le client (selon l'émetteur) va déclencher une fonction (qui peut elle même renvoyer un événement et ainsi de suite). Cela permet par exemple de détecter quand est-ce qu'un client appuie sur un bouton afin d'envoyer cette information au serveur.

Le programme du serveur est décomposé en plusieurs parties :

- **Déclaration des variables**, notamment pour pouvoir utiliser les modules de NodeJS. Parmi les modules les plus importants que nous avons utilisés il y a :
  - Express pour une gestion plus simple des routes et des fichiers statiques (css, html, js…).
  - Http pour lancer un serveur http
  - fs, string et line-by-line pour la lecture d'un fichier texte
  - socket.io (web socket) pour que la communication puisse commencer du serveur et d'un client
- **Lancement du serveur** http et de socket.io
- **Gestion des routes** (simple dans notre cas étant donné que nous n'avons qu'une véritable page)
- Gestions des différents **événements**

Le côté client est constitué d'une page html ayant deux états possibles :

Un premier état (alarme non déclenchée) avec un bouton pour déclencher l'alarme, un cadre d'avertissement et un cadre de consignes de sécurité.

***Figure 2** Etat Alarme non déclenchée*

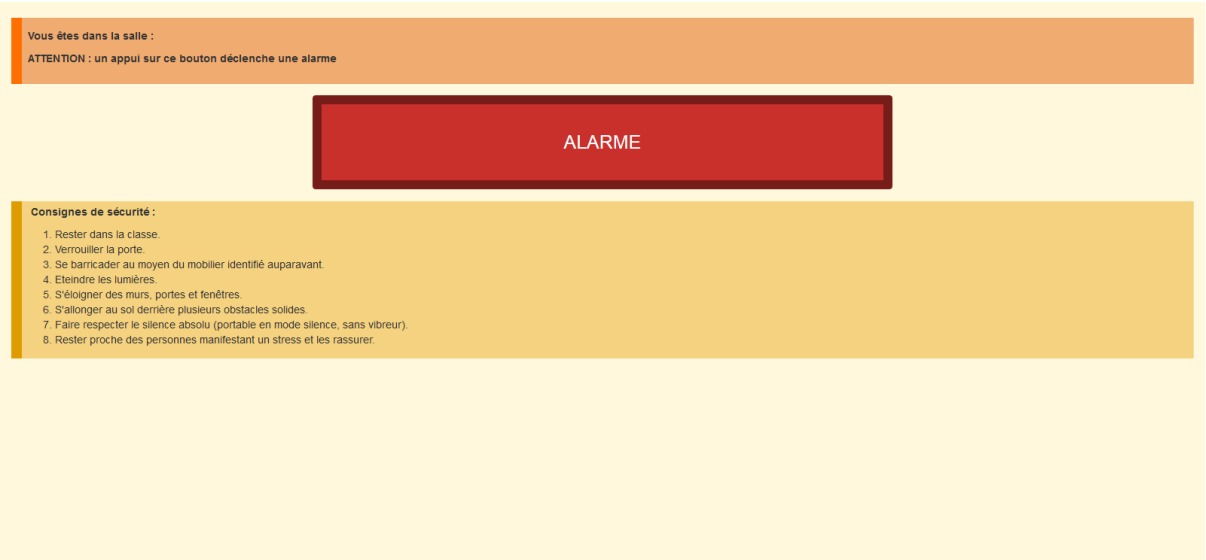


***Figure 3** Etat Alarme déclenchée*

Un deuxième état (alarme déclenchée) contenant un cadre qui indique qui a déclenché l'alarme, le cadre des consignes de sécurité et un bouton pour arrêter l'alarme.

Le passage d'un état à l'autre se fait en appuyant sur le bouton disponible sur la page. Entre les deux états, les différents éléments sont simplement rendus visibles et invisibles selon le besoin.

En ce qui concerne le design, le CSS est fait en parti avec Bootstrap qui permet site responsive et un placement des éléments simplifié.

Pour ce qui est des interactions entre le serveur et les clients (gérées dans un fichier JavaScript pour le client), nous avons mis en place deux fonctionnalités principales : le système de nommage des clients et le système d'alarme en lui-même.

- Le système de nommage permet de donner un nom aux différents clients du réseau. Il fonctionne de la manière suivante : lorsqu'un client fait une requête au serveur http on récupère son ip et vérifie si celle-ci est déjà enregistrée dans un fichier texte (contenant les ip et les noms correspondants des clients) qui sert de base de données. Si elle n'est pas enregistrée, le serveur envoie alors un événement au client concerné qui lui fait apparaître un prompt lui demandant un nom. Une fois le nom entré, celui-ci est renvoyé au serveur via un autre événement et est sauvegardé du côté client par le biais d'une variable de session. Si l'ip est enregistrée, le nom qui lui correspond est envoyé au client concerné à partir du serveur et est enregistré dans les variables de session.

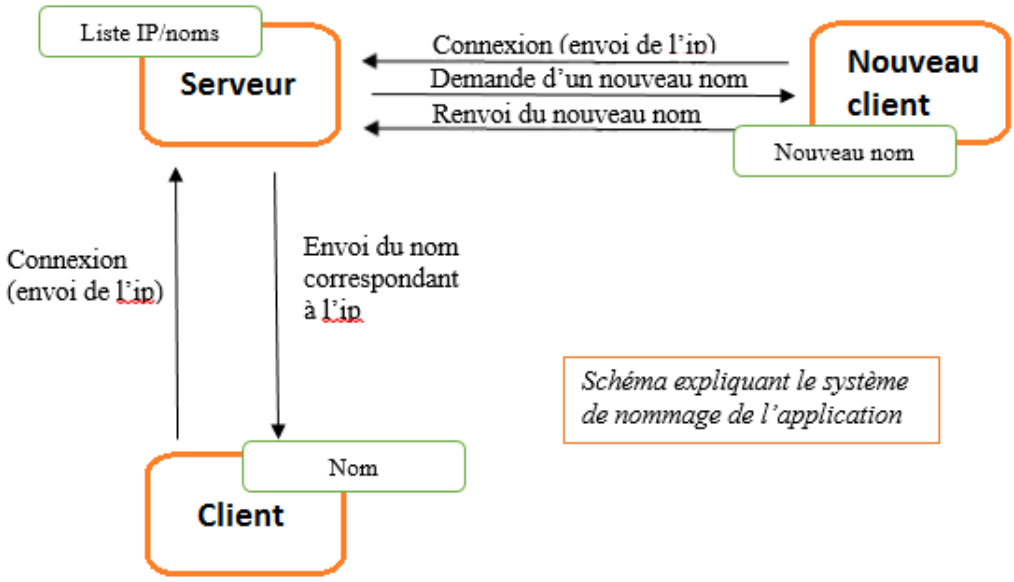


Schéma expliquant le système de nommage de l'application

- Le système d'alarme lui fonctionne de la manière suivante : lorsqu'un client déclenche une alarme (appuie sur le bouton), il envoie un événement au serveur avec son nom en paramètre qui à son tour va renvoyer un événement à tous les clients connectés. Déclenchant ainsi le passage de l'état alarme non déclenchée à l'état alarme déclenchée de la page html. Le nom du client ayant déclenchée l'alarme s'affichant sur la page. Le même processus est appliqué pour l'arrêt de l'alarme (on passe de l'état déclenché à non déclenché).

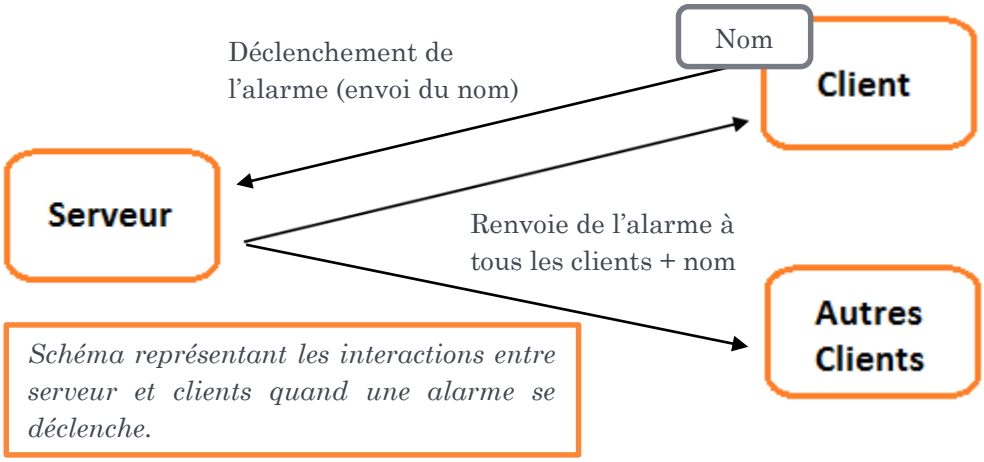


Schéma représentant les interactions entre serveur et clients quand une alarme se déclenche.

Améliorations possibles :

- Un système d'administrateur ayant des droits notamment pour gérer le fichier texte des IP/noms.
- Une meilleure visualisation pour bien voir que l'alarme a été déclenchée.

b- Hébergement

Notre serveur NodeJs a besoin de fonctionner en continue pour que l'application fonctionne. Dans le cas d'une application directe dans un établissement scolaire, il est nécessaire que le serveur reste allumé toute la journée afin que l'application puisse être utilisée à tout moment par un des enseignants.

Nous avions différentes options:

- implanter directement notre application dans le serveur de l'école
- utiliser un ordinateur de l'école
- implémenter notre application sur une carte RaspberryPI

Nous avons choisi la troisième option. La première option était difficilement réalisable, d'un point de vue pratique, nous n'aurions pas eu accès au serveur de l'établissement. Pour la seconde option, il est dommage de monopoliser un ordinateur en permanence pour une petite application.

Nous nous sommes donc tournés vers la troisième option, la carte RaspberryPI.

C'est un petit ordinateur qui permet d'utiliser différents systèmes d'exploitation libres GNU/Linux ainsi que les logiciels compatibles à la version de l'OS utilisée. La carte est fournie seule (sans boîtier, alimentation, clavier, souris ni écran) dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération.

Figure 4 *Photo de la Raspberry PI*

Pour héberger notre serveur, nous avons choisi d'utiliser une carte RaspberryPI3.

Pour rendre notre système utilisable par la carte, il nous a fallu préparer la carte. Voici la liste des modifications apportés:

- brancher tous les périphériques (écran, clavier, souries, cable éthernet)
- installation d'un os sur la carte mémoire
- définir une adresse IP fixe pour un accès internet
- ajouter Node JS
- ajouter les différents fichiers de l'application grâce au port USB
- lancer le serveur avec la console

Avantage:

A partir de ce moment, l'application fonctionne de la même manière que sur un ordinateur, il est alors possible de débrancher tous les périphériques excepté le câble RJ45 afin de minimiser l'espace physique occupé, l'application continuera de fonctionner.

Désavantage:

En cas de coupure de courant, il sera nécessaire de tout rebrancher pour pouvoir relancer le serveur.

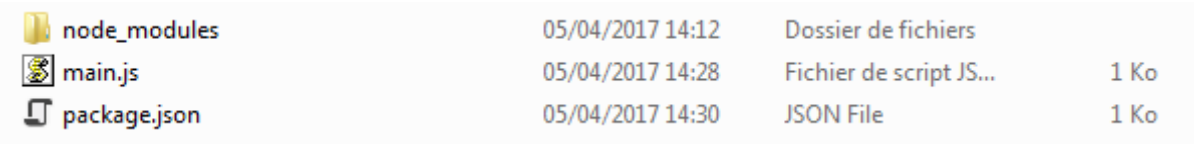
Amélioration possible:
Faire un script qui s'exécute au démarrage

c- L'application client

Notre objectif est de développer une application qui puisse être déployée sous tous les ordinateurs d'un établissement scolaire. Il nous faut donc trouver une librairie qui permette à notre application de se connecter à un serveur web, et de renvoyer une page Html, c'est pourquoi la technologie Electron a été un bon compromis pour notre projet.

Développé à partir de 2013, Electron est une librairie libre développé par GitHub. Elle a pour objectif de créer une application de bureau avec les technologies du web, à savoir HTML, CSS et JavaScript. Electron a été façonné en combinant Chromium et Node.js afin de créer des applications compatibles avec Mac, Windows et Linux. Pour notre projet, nous avons utilisé cette librairie pour développer une application qui ouvre une fenêtre avec notre page d'Alerte intrusion hébergée sur un serveur Node.js.

Pour créer une application avec Electron il nous faut au moins 2 fichiers, le package.json qui est le manifeste de notre application, et le main.js, script principal écrit en javascript, qui va permettre d'initialiser la fenêtre sous windows. Un dossier node_modules est indispensable pour pouvoir utiliser le module "electron" qui est attaché à Node.js.



Voici la description des deux fichiers principaux.

Dans notre programme "package.json", nous avons 3 clés principales à éditer pour que notre application fonctionne:

- "name", le nom de notre application
- "version", la version de notre application
- "main", le chemin vers le script principal
- La clé "test" permet de générer et de lancer l'application depuis l'invit de commande en tapant la commande "**electron .**".

```
{
  "name": "alerte_intrusion",
  "version": "1.0.0",
  "description": "signaler une intrusion dans un etablissement",
  "main": "main.js",
  "scripts": {
    "test": "electron ."
  },
  "author": "felix, colin, adrien",
  "license": "ISC",
  "devDependencies": {
    "electron": "^1.6.2"
  }
}
```

Le script principal est bien "main.js", celui-ci permet de créer et d'ouvrir une fenêtre avec tous les paramètres que nous voulons (à savoir la taille de la fenêtre, le titre de l'application, la

mise en place d'un menu ou non, etc.). Cette fenêtre affiche tout simplement la page qui est entré en paramètre grâce à son URL. Le code décrit ci-dessous permet de mieux comprendre son fonctionnement.

```
const electron = require('electron');
const app = electron.app;
const BrowserWindow = electron.BrowserWindow;

let mainWindow;

function createWindow () {

  mainWindow = new BrowserWindow({
     // on définit une taille pour notre fenêtre
    width: 1800,
    height: 1200,
    // On définit le chemin de l'icone de notre application
    icon: 'logo.png',
    // On définit le titre de l'application
    title: 'Dispositif d\'alerte intrusion ',
    maximized: false,
    center: true,});
    // Adresse URL du serveur (ici son adresse ip)
    mainWindow.loadURL(`http://172.20.82.171:6987`);

    mainWindow.on('closed', () => {
    mainWindow = null;
  });
}

//Appel de la methode createWindow lorsque l'appli demarre
app.on('ready', createWindow);

//quitter l'application quand toutes les fenetres sont ferme
app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  if (mainWindow === null) {
    createWindow();
  }
});
```

Une fois l'application terminée, il nous faut créer le fichier exécutable pour notre système d'exploitation. Pour cela il nous faut utiliser le programme electron-packager, pour le télécharger, il suffit d'entrer la ligne de commande suivante:

npm install electron-packager -g --save-dev

Maintenant il ne nous reste plus qu'à compiler notre application en tapant ceci :

electron-packager . –all

Cette ligne de commande permet de générer des dossiers pour les trois systèmes d'exploitation (Mac, Linux et Windows) sous 32 et 64 bits. Chaque dossier contient plusieurs fichiers et dossiers dont les exécutables appropriés.

Voici le rendu de notre application, une fois l'exécutable créé :

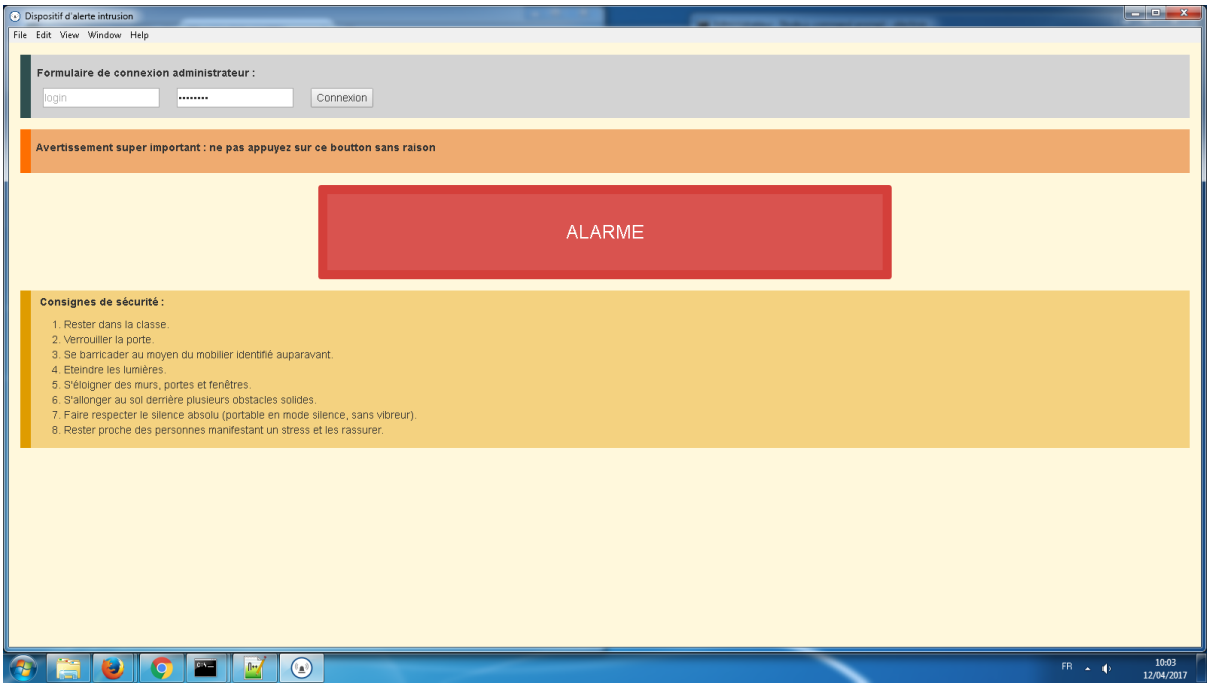


***Figure 5** Rendu de l'application*

3- BILAN

a- Principales difficultés rencontrées

Nous avons rencontrés beaucoup de difficultés durant notre projet. La première et la principale raison de ces problèmes est le fait que nous avons dû travailler sur des langages de programmation que nous ne connaissions pas du tout. Il a fallu tout apprendre pour pouvoir faire quelque chose de livrable.

NodeJS nous a pris beaucoup de temps pour dans un premier temps comprendre la programmation événementielle, ce qui est un peu déroutant au début ainsi que l'utilisation des différents fichiers au bon moment, c'est à dire la séparation du code HTML et du Java, ainsi que l'utilisation des différents module de Node JS comme la lecture et l'écriture dans fichier texte avec les fonctions de NodeJS.

Nous avions pour objectif de créer une page administrateur pour gérer toutes les adresses IP des ordinateurs connectés au serveur, gérer le plan de l'établissement, désactiver l'alarme déclenché, etc. Mais pour créer une page administrateur il faut créer un formulaire de connexion (et utiliser les sessions node.js notamment), c'est sur ce point que nous avons eu beaucoup de problèmes. Nous avons perdu beaucoup de temps à essayer de gérer la connexion d'un administrateur.

Nous sommes partis de plusieurs tutoriels de chat qui permettait de faire une authentification avec NodeJS, Express et Socket.IO. L'utilisation de cookies a permis d'utiliser des sessions et de mémoriser la connexion d'un client. Le problème est que nous n'avons pas réussi à utiliser

le système de mot de passe pour s'authentifier, seul le login (s'il était libre) permettait de se connecter. De plus la non compatibilité des versions de codes présentés nous a fait perdre du temps car il a fallu la réadapter avec la nouvelle version express 3.0.

b- Evolutions possibles de notre projet

Notre projet a bien avancé, mais n'est pas terminé, il reste beaucoup à faire, et peut encore s'améliorer sur plusieurs points. Nous avons réfléchi à plusieurs fonctionnalités qui pourraient être intéressantes de développer pour perfectionner notre outil.

Comme nous l'avions prévu au début, il a été question de rajouter un plan de l'établissement qui indique d'où est déclenchée l'alarme, ce qui permettrait à chaque enseignant de mieux se repérer dans l'espace et de voir d'où vient la menace. Cette fonctionnalité peut vite devenir un atout majeur en cas d'intrusion.

Comme pour beaucoup d'applications et de services informatiques, un accès "administrateur" est réservé pour pouvoir gérer toutes les fonctionnalités et paramètres de l'application. C'est pourquoi nous avons décidé de développer cette partie qui nous semble importante.

L'administrateur aurait pour fonction la possibilité d'ajouter et d'enlever des ordinateurs (via leur adresse IP) étant connectés au réseau et possédant l'application "Alerte Intrusion". De plus il serait possible d'affecter un nom de salle à chaque ordinateur pour pouvoir les reconnaître plus facilement (salle 101, 102, …).

L'administrateur aurait également la possibilité d'éteindre l'alarme déclenchée auparavant, en effet il n'est pas judicieux que n'importe qui puisse éteindre l'alarme.

Avec le temps et les réformes, les mesures de sécurité changent et évoluent, c'est pourquoi il peut être intéressant que l'administrateur puisse modifier les mesures de sécurité affichées.

A chaque changement de personnel, donc changement d'administrateur, il faut changer le mot de passe, mesure indispensable pour plus de sécurité. C'est pourquoi il est important d'ajouter une méthode qui permette de changer le mot de passe de l'administrateur.

Malheureusement par manque de temps, il ne nous a pas été possible de finir le développement de la partie "Administrateur".

Développer une application pour ordinateur c'est bien, mais pour mobile c'est mieux. En effet lors d'un cours, un enseignant bouge beaucoup et ne se retrouve pas souvent devant son ordinateur, alors que le téléphone portable se retrouve souvent dans sa poche, et si ce n'est pas le cas, un mode vibreur permettrait aisément d'alerter l'enseignant. C'est pourquoi il serait très intéressant de créer une application mobile permettant de se connecter au serveur web présent sur le réseau. D'après quelques recherches, Electron permettrait de répondre à nos attentes.

Une fonctionnalité ambitieuse mais qui peut être très pratique : la supervision de toutes les écoles reliées aux autorités. L'objectif serait de créer un logiciel de supervision qui puisse récolter tous les déclenchements d'alarme des écoles primaires où est déployée notre application. Cela permettrait une intervention beaucoup plus rapide et efficace des forces de l'ordre, ce qui reste le principe de notre projet.

c- Bilans personnels

Félix DELAUNAY

Pour ma part j'ai trouvé le projet intéressant, d'une part car il y a une réelle demande client avec un objectif clair et précis, pouvoir alerter tout un établissement d'une intrusion suspecte, d'autre part car ce projet nous a apporter de nombreuses compétences techniques. En effet nous avons pu nous imprégner du langage de programmation JavaScript qui était indispensable pour les technologies Electron et Node JS. C'est également sur ces deux technologies que nous

avons beaucoup appris, notamment de savoir créer et de gérer un serveur, des tâches qui n'ont pas toujours été facile. De plus un énième travail de groupe permet de se perfectionner dans la répartition des tâches, et ainsi pouvoir travailler plus efficacement.

Même si je me prédestine plus vers des métiers de l'automatisme, ce projet s'est révélé intéressant. Le manque de temps pour nous permettre de finir notre application est un peu frustrant.

Colin GACHET

En ce qui me concerne ce projet a été intéressant mais également assez frustrant. En effet d'un côté découvrir un tout nouveau langage pour moi (Node.js et en même temps Javascript) était vraiment intéressant, cependant cela a apporté aussi son lot de frustration. En effet bien que la communauté de Node.js est très active, la documentation n'est pas à mon opinion orientée vers les néophytes (notamment pour tout ce qui est serveur http, authentification…). Ce qui m'a mené plusieurs fois à butter sur certains problèmes assez longtemps.

Pour autant j'ai bien apprécié le projet, le fait de savoir qu'on aurait pu tester notre prototype dans une école a été un gros booste de motivation pour moi. Je regrette de n'avoir pas pris un peu plus d'initiatives de ce côté-là, cela nous aurait permis notamment d'y voir un peu plus clair dans ce que nous devions/voulions faire.

Dernier point sur lequel j'aimerai revenir est celui de la gestion de projet : qu'est-ce qu'on fait, quand, comment ? Je me suis rendu compte que ces questions sont vraiment importantes dans un projet et que le nôtre aurait être catastrophique si nous n'avions pas réfléchi au début.

Adrien PERRIER

Ce projet m'a permis de découvrir différentes technologies. Dans un premier temps Node JS pour la gestion Client-Serveur ainsi que pour la programmation événementielle et le javascript, bootstrap pour la partie design de la page web, le framework Électron et pour finir l'utilisation d'une Raspberry PI3 qui fait office de serveur. Ce projet m'a aussi permis de mettre directement en application des compétences acquises lors des cours de Gestion de Projet.

BIBLIOGRAPHIE

SARRION, Eric, 18 septembre 2014. **Programmation avec Node.js, Express.js et MongoDB**. Edité par Eyrolles, 586 pages. Collection Noire.

SERVICE INFORMATION DU GOUVERNEMENT. **Stop-Djihadisme**. Disponible sur : http://www.stop-djihadisme.gouv.fr/lutte-contre-terrorisme-radicalisation/mesures-lutter-contre-terrorisme/comprendre-plan-vigipirate-4

BAT', dernière mise à jour le samedi 19 novembre 2016. **Zeste de Savoir, Tutoriel Electron**. Disponible sur : https://zestedesavoir.com/tutoriels/996/vos-applications-avec-electron/

Authentification et WebSocket, avec Node.JS, Express, et Socket.IO. Disponible sur : http://naholyr.fr/2011/07/authentification-et-websocket-avec-node-js-express-et-socket-io/

Dernière mise à jour le 9 Mars 2017. **Using a package.json**. Disponible sur : https://docs.npmjs.com/getting-started/using-a-package.json

Node.js. **Node.js v7.9.0 Documentation**. Disponible sur : https://nodejs.org/api/

StrongLoop, Inc. Express. Disponible sur : http://expressjs.com/fr/

Alerte Intrusion

Résumé :

Le but de ce projet de quatrième année est de mettre en place un système d'alarme dans le cadre des nouveaux exercices de sécurité du plan Vigipirate. Le système repose sur un serveur développé sous Node.js et un client web (html, css, javascript). Un client peut déclencher une alarme qui est ensuite redirigée par le serveur vers tous les clients

Mots clés : Node.js, système d'alarme, client web, VigiPirate, sécurité

Abstract :

The goal of this fourth year project is to develop an alarm system in the context of VigiPirate's new security exercises. The system is based on a Node.js server and a web client (html, css, javascript). One client can trigger an alarm which spread then to all connected clients via the

Keys words: Node.js, alarm system, security, web client, Vigipirate