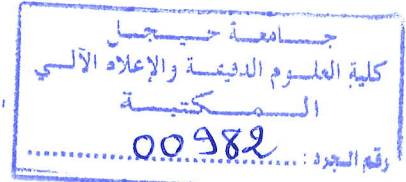


RÉPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mohammed Seddik BENYAHIA de JIJEL



Inf. ILM. 04/17

Faculté des Sciences exactes et Informatique
Département d' Informatique

Mémoire présenté pour l'obtention du diplôme de
Master en Informatique

Option
Informatique Légale et Multimédia

24
52

Conception d'un système de traces de
localisations géographiques pour une
flotte de véhicules

Proposé par : Zennir Mohamed Nadjib

Réalisé par : Bahri Tawfik, Berghida Youcef



Année universitaire : 2016/2017

Remerciements

Nous remercions tout d'abord Dieu tout puissant de nous avoir donnés le courage, la force et la patience d'achever ce modeste travail.

Nous remercions vivement notre directeur de recherche Mr ZENNIR. M.Nadjib qui nous a encadrés durant la réalisation de ce projet. Nous lui sommes reconnaissants pour son appui, sa disponibilité, ses critiques ,ses conseils qui furent précieux dans l'aboutissement de cette recherche et la patience dont il a fait preuve à notre égard. Tous nos remerciements et notre profonde gratitude à tous nos enseignants depuis le premier qui nous a accueilli à l'âge de cinq et six ans dans une classe, jusqu'à celles et ceux qui nous encadrent aujourd'hui'hui.

Un grand merci à messieurs les membres du jury pour avoir examiné minutieusement notre travail et de nous faire l'honneur de leur présence à notre présentation.

Enfin, nous tenons à remercier tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail.

REMERCIEMENTS

Résumé

En Algérie, les véhicules de transport parcourent des millions de kilomètres quotidiennement, convoyant des tonnes de marchandises. Le transport poids lourd est l'un des éléments importants de l'économie nationale et le développement dans divers domaines. Cependant, il est évident que l'Algérie dépend largement sur du transport routier (manque d'autres alternatives), ce qui conduit à l'augmentation des accidents de la route causés par ces véhicules, que ce soit en heurtant des voitures, des bus, des arbres, des murs ou même des civils. Parfois, les chauffeurs des camions fuient toutes responsabilités. Pour réduire les accidents causés par les camions, il existe des dispositifs de suivi (tracking devices) permettant aux propriétaires des entreprises de voir, en temps réel, leurs employés qui conduisent des camions sur n'importe quelle route, en utilisant le GPS, ou même savoir si l'employé conduit le véhicule d'une façon appropriée, s'il suit le chemin prévu, ou encore s'il a eu un accident, et beaucoup d'autres choses. L'inconvénient principal de ces dispositifs de suivi est qu'ils coûtent chers, aussi leur maintenance est difficile et coûteuse. Dans ce travail, nous proposons une alternative qui peut remplacer entièrement ces dispositifs. Une application Android gérée par un serveur à distance. En plus de tous les avantages des dispositifs de suivi, ce système proposé permet également l'action d'une entité gouvernementale pour intervenir en cas de coïncidence ou négation des faits. Cela coûtera moins cher que les dispositifs de suivi du marché.

Abstract

In Algeria, trucks runs millions of kilometers daily carrying tons of merchandise. The heavy transport is one of the important elements of the national economy and development in various fields. However, it's obviously that Algeria is depending on road transport in a large way (due to the absence of other alternatives), this has led to the increasing of road accidents caused by these vehicles, whether by hitting smaller cars, buses, trees, walls or even people. Sometimes trucks driver runs away from what they have done. As an alternative to reduce accidents caused by trucks, we propose tracking devices allowing companies owners to view; in real time, their employees driving trucks in any road, using GPS, or even know whether the employee is driving the truck in an appropriate way or not, is he following the designed path? The major disadvantage of these tracking devices is their prohibitive. Their maintenance is also difficult. In this work, we propose an alternative that can entirely replace these devices. An android application remotely controlled by a server. Beside all the advantages of tracking devices our proposed system includes also a possible action of a government entity which can intervenes in case of a coincidence. This system will cost less than the cheapest tracking devices in the market.

Table des matières

Remerciements	iii
Résumé	v
Abstract	vi
Table des matières	vii
Table des figures	xi
Introduction Générale	1
1 Introduction aux technologies de tracking et État de l'art	3
1 Études des marchés	3
1.1 Étude du marché des Trackers	3
1.1.1 Traceur physique	4
1.1.2 Traceur logiciel	5
1.2 Étude du marché des Smartphones	5
2 Technologies utilisées	6
2.1 Globale Positioning System	6
2.1.1 Définition du GPS	6
2.1.2 Composition du GPS	6
2.1.2.1 Segment spatial (Space Segment)	7
2.1.2.2 Segment de contrôle (Control Segment)	7
2.1.2.2.1 Station de contrôle principale (MCS)	9
2.1.2.2.2 Stations de surveillances	9
2.1.2.2.3 Antennes terrestres	9
2.1.2.3 Segment utilisateur (User Segment)	9
2.1.3 Fonctionnement du GPS	10
2.1.4 Alternatives	10
2.2 Android	11
2.2.1 Définition d'Android	11
2.2.2 Historique d'Android	11
2.2.3 Caractéristiques d'Android	11
2.2.4 Android Lollipop 5.0	12
2.2.5 Architecture d'Android	12
2.2.5.1 Noyau Linux	12
2.2.5.2 Hardware Abstraction Layer (HAL)	12
2.2.5.3 Android Runtime	14
2.2.5.4 Native C/C++ Libraries	14
2.2.5.5 Java API Framework	14
2.2.5.6 Applications système d'Android	14

3	Téléphonie mobile troisième génération (3G)	15
3.1	Définition de la 3G	15
3.2	Caractéristiques de la 3G	15
3.3	Comparatif aux autres générations	16
4	Conclusion	16
2	Le modèle Multi-vehicles Tracking	19
1	Description de MVT	19
1.1	Les Clients	19
1.1.1	Client Android	20
1.1.2	Client WEB	20
1.2	Les Serveurs dédiés	21
1.2.1	Serveur central	21
1.2.2	Serveur REDIS	22
1.2.3	Entité Gouvernementale	23
1.3	Cloud	23
2	Conception de MVT	24
2.0.1	Relation Client Android / Serveur Base de données	24
2.0.2	Relation Serveur Base de données / Serveur REDIS	25
2.0.3	Relation Client WEB / Serveur Base de données	25
2.0.4	Relation Serveur Base de données / Serveur Entité Gouvernementale	34
2.0.5	Relations des entités de création et gestion de la base de données	36
3	Développement de la solution	36
3.1	Client Android	36
3.1.1	Composition du Client	37
3.1.1.1	MainActivity/Onglet Principal	37
3.1.1.2	Settings/Onglet Secondaire	39
3.1.2	Fonctionnement du Client	41
3.1.2.1	Mode Offline	44
3.1.2.2	Mode Online	44
3.2	Serveur Central	44
3.2.1	Serveur Base De Données	50
3.2.2	Serveur REDIS	54
3.3	Site Web	56
3.3.1	Outils de développement	56
3.3.2	Présentation des interfaces	57
4	Conclusion	68
3	Signature et Validation par une Agence gouvernementale	69
1	Introduction à la Cryptographie	69
1.1	Définition	69
1.2	Cryptographie Symétrique	69
1.3	Cryptage Asymétrique	70
1.4	Hachage	71
2	L'Entité Gouvernementale	72
3	Développement de l'Entité Gouvernementale dans le système	72

4	Conclusion	74
4	Gestion des zones blanches	77
1	Les zones de couvertures des opérateurs algériens	77
2	Notre solution	77
	2.1 Algorithme des K-plus proches voisins	79
	2.2 Implémentation dans notre système	80
3	Conclusion	82
5	Testes et expérimentations de MVT	83
1	Configurer l'environnement de développement	83
2	Installer l'application sur un appareil mobile	83
3	Testes avec émulateur Android	84
	3.1 Émulateur Android	84
	3.2 Émulateur de GPS	86
	3.2.1 Côté Serveur	87
	3.2.2 Côté Client	89
	3.2.3 Simuler les zones blanches	89
4	Test réel	92
5	Les zones blanches	93
6	Les limites du positionnement géographique par 3G	95
7	Conclusion	95
	Conclusion générale et perspectives	97
	Bibliographie	99

TABLE DES MATIÈRES

Table des figures

1.1	Exemple d'un type de traceur physique disponible sur le marché	4
1.2	Schéma représentant la composition du GPS	7
1.3	Schéma représentant une constellation de 24 satellites en orbite au- tour de la terre	8
1.4	Schéma représentant les localisations des éléments du Segment de contrôle sur la carte du monde.	8
1.5	Exemple du calcul d'un point sur terre avec 3 satellites	10
1.6	Schéma représentant l'architecture complète d'Android	13
1.7	Tableau comparatif des débits moyens en download et upload des technologies	16
2.1	Schéma représentant notre modèle	20
2.2	Schéma d'une Base de donnée hiérarchique	21
2.3	Schéma représentant la structure de notre base de données	22
2.4	Schéma représentant la synchronisation entre le serveur et le cloud	23
2.5	Diagramme de cas d'utilisation UML entre le Client Android et le Serveur base de données	24
2.6	Diagramme de séquences UML entre le Client Android et le Serveur base de données	25
2.7	Diagramme de cas d'utilisation UML entre le Serveur base de données et le Serveur REDIS	26
2.8	Diagramme de séquences UML entre le Serveur base de données et le Serveur REDIS	26
2.9	Diagramme de cas d'utilisation UML entre les utilisateurs et le site web	27
2.10	Diagramme de Séquences UML pour se connecter comme Admin	28
2.11	Diagramme de Séquences UML pour se connecter comme Company	28
2.12	Diagramme de Séquences UML pour ajouter une entreprise	29
2.13	Diagramme de Séquences UML pour modifier une entreprise	29
2.14	Diagramme de Séquences UML pour supprimer une entreprise	30
2.15	Diagramme de Séquences UML pour ajouter un conducteur (Admin)	30
2.16	Diagramme de Séquences UML pour modifier un conducteur (Admin)	31
2.17	Diagramme de Séquences UML pour supprimer un conducteur (Ad- min)	31
2.18	Diagramme de Séquences UML pour ajouter un conducteur (Com- pany)	32
2.19	Diagramme de Séquences UML pour modifier un conducteur (Com- pany)	32

2.20	Diagramme de Séquences UML pour supprimer un conducteur (Company)	33
2.21	Diagramme de Séquences UML pour le suivi des conducteurs (Admin)	33
2.22	Diagramme de Séquences UML pour le suivi des conducteurs (Company)	34
2.23	Diagramme de cas d'utilisation UML entre le Serveur base de données et l'entité gouvernementale	35
2.24	Diagramme de séquences UML entre le Serveur base de données et l'entité gouvernementale	36
2.25	Diagramme d'interaction UML entre les éléments de création et gestion de la base de données	37
2.26	Interface principale de l'application Android	38
2.27	Code de la fonction getDeviceImei	38
2.28	Code de la fonction qui récupère la localisation	39
2.29	Code de la fonction qui calcule la vitesse	40
2.30	Code de la fonction batteryLevel	40
2.31	L'interface secondaire de l'application Android	41
2.32	Code du bouton Envoi	42
2.33	Code de la fonction BackgroundOperation2	43
2.34	Code de la fonction isNetworkAvailable	44
2.35	Fragment de code qui permet le fonctionnement Offline	45
2.36	Fragment de code qui de la classe DBHandler et la fonction Ajouter- Locale	46
2.37	Fragment de code de la classe BackgroundOperation en utilisant une base de données	46
2.38	Code de la Déclaration de la classe Mouvement	47
2.39	Fragment de code de la classe BackgroundOperation	47
2.40	Code de la classe MCrypte	48
2.41	Fragment de code qui permet le fonctionnement Online	49
2.42	Code qui permet la récupération des POSTs des clients	51
2.43	Code de MCrypte	52
2.44	Un des dossiers qui contiennent les documents de la base de données.	53
2.45	Un document de la base de données.	53
2.46	Code des déclarations de la bibliothèque JWT	54
2.47	Code qu'utilise la requête "Request".	54
2.48	Code qu'utilise la requête "localisations".	55
2.49	Code pour la requête d'ajout sur PHP	55
2.50	Code du fichier ajout.py	55
2.51	Code pour la requête de recherche sur PHP.	56
2.52	Code du fichier recherche.py	56
2.53	La page Login.php	57
2.54	La barre latérale gauche pour le type "Admin" à gauche et le type "Company" à droite	58
2.55	Page principale	59
2.56	La barre latérale droite	59
2.57	Code HTML qui permet de récupérer les informations des conducteurs	59
2.58	Code JavaScript qui permet d'ajouter les informations des conducteurs	60

2.59	La page companies.php	60
2.60	La page d'ajout d'entreprises, addcompany.php	60
2.61	La page drivers.php	61
2.62	La page newdriver.php	61
2.63	Code en JavaScript du bouton 'Afficher'	62
2.64	Code en JavaScript de la variable 'currentDate'.	63
2.65	Code en JavaScript qui permet de cliquer automatiquement sur le bouton 'En temps réel' chaque 20 secondes.	63
2.66	Code en JavaScript du bouton 'En temps réel'.	63
2.67	Code en JavaScript du bouton 'Arrêter'.	64
2.68	Code en JavaScript pour tracer la route sur la carte.	65
2.69	Code en JavaScript pour tracer la route sur la carte (Plusieurs conducteurs).	65
2.70	Code qui permet la récupération des données à partir de 'request.js'.	66
2.71	Code qui s'exécute si le type est "Locations"	66
2.72	Code qui s'exécute si le type est "RealTime"	67
2.73	Code qui s'exécute si \$myArray est vide	67
2.74	Code qui s'exécute si \$myArray ['response'] = 'granted'	67
2.75	Code qui s'exécute si \$myArray ['response'] = 'granted'	68
3.1	Schéma de fonctionnement de la cryptographie symétrique.	70
3.2	Schéma de fonctionnement de la cryptographie asymétrique.	70
3.3	Vue générale de MD5.	71
3.4	Schéma représentant un tour dans MD5.	72
3.5	Diagramme montrant comment des données sont signées, puis vérifiées.	73
3.6	Partie du code du script index.php qui fait appel à l'entité gouvernementale	73
3.7	Partie d'un document de la base de donnée qui contient le hash chiffré	74
3.8	Partie du code du script gov.php	75
4.1	Zone de couverture Djezzy	78
4.2	Zone de couverture Mobilis	78
4.3	Zone de couverture Ooredoo	79
4.4	Pseudo Algorithme K-PPV	79
4.5	Schéma qui représente un exemple d'une classification à base d'algorithme K-PPV	80
4.6	Fragment du code qui utilise l'algorithme K-PPV	81
5.1	Installation à partir d'une source inconnue.	84
5.2	Écran principal de l'émulateur Android NOX	85
5.3	Interface principale de l'application Android MVT	85
5.4	Configuration de l'application	86
5.5	Émulateur GPS 'Mock Locations'.	87
5.6	Application MVT, réception des coordonnées et des vitesses correspondantes.	87
5.7	La base de données hiérarchique 'database'.	88
5.8	Le fichier '00.txt' créé par le serveur	88
5.9	Créer un nouveau fichier ('01.txt').	89

TABLE DES FIGURES

5.10	Ajouter le hash à la fin du document.	89
5.11	Le suivi du simulateur GPS.	90
5.12	Une zone blanche.	90
5.13	Un marqueur avec un point d'exclamation (Vitesse > 90 km/h).	91
5.14	Essayer de suivre un conducteur hors ligne.	91
5.15	Caractéristique du smartphone Condor.	92
5.16	Caractéristique du smartphone Oppo.	92
5.17	Le suivi d'un seul conducteur.	92
5.18	Le suivi de plusieurs conducteurs.	93
5.19	Les zones blanches lorsque $K = 3$	94
5.20	Les zones blanches lorsque $K = 5$	94
5.21	Les zones blanches lorsque $K = 7$	94
5.22	Les zones blanches lorsque $K = 9$	95

Table des abréviations

- 3G** Troisième Génération. 1, 14, 15, 16, 19
- 3GPP** 3rd Generation Partnership Project. 14
- API** Application Programming Interface. 14
- CSS** Cascading Style Sheets. 56
- EDGE** Enhanced Data rates for GSM Evolution. 14
- GPRS** General Packet Radio Services. 14
- GPS** Global Positioning System. v, 1, 3, 6, 7, 8, 9, 10, 16, 19, 37, 77, 95
- GSM** Global System for Mobile Communications. 14
- HTML** HyperText Markup Language. 56, 58
- HTTP** HyperText Transfer Protocol. 50
- IMEI** International Mobile Equipment Identity. 21, 24, 37, 50
- K-PPV** K plus proches voisins. 79, 93, 97
- LTE** Long Term Evolution. 14, 15
- MVT** Multi-Vehicles Tracking. 21, 77
- OS** Operating System. 11
- PHP** Hypertext Preprocessor. 44, 50, 68
- TCAC** Taux de Croissance Annuel Composé. 3
- UMTS** Universal Mobile Telecommunications System. 14, 15

Introduction Générale

Depuis le début du troisième millénaire, l'Algérie est entrée dans un stade évolutif assez important, notamment dans le secteur du transport, et plus précisément le transport routier au point qu'il soit reconnu comme étant l'un des plus denses du continent africain avec une longueur de 112039 Km [3] (Au alentour de 2005). Celà dit, ce n'est pas l'unique secteur bénéficiant de cette évolution, le secteur des télécommunications a aussi vu une grande amélioration, surtout aux niveau technologique, ce qui a permit à la 3G de se propager dans le pays (bien que la couverture soit assez inégale pour l'instant) et qui a rendu internet accessible à tout moment pour les habitants.

L'évolution du réseau routier a permit la circulation de 5.3 millions de véhicules (remorques et motocyclettes exclues) [10]. Ce nombre important de véhicules implique linéairement autant d'accidents. En 2016, environs 3200 accidents ont eu lieu sur le territoire national et qui ont eu pour conséquences la mort de 700 personnes. Selon les statistiques, environ 85% des accidents sont d'origine humaine [6]majoritairement pour cause d'excès de vitesse. Ce problème cause la perte de plusieurs milliards de dinars aux entreprises fautives spécialisées dans les transports routiers (que ce soit transport de voyageurs ou de marchandises). Hélas, ceci n'est pas la seule préoccupation de ces mêmes sociétés.

En Algérie, les logistiques de transport sont quasiment inexistantes. Le temps que prend le déplacement d'une marchandise d'un point A vers un point B dans le territoire national diffère à chaque déplacement malgré que ce soit sur le même trajet et avec le même trafic routier. Le manque d'informations par rapport aux routes oblige, dans la majorité des cas, les conducteurs à faire des trajets plus longs, surtout lorsqu'ils doivent revenir sur leurs pas, ou à cause de travaux qui interrompent la circulation. Il faut aussi prendre en compte la possibilité d'occurrences de pannes de véhicules, surtout lorsqu'elles surviennent dans des zones isolées ou peu peuplées. Cela peut prendre des heures au conducteurs pour finalement reprendre la route.

Le fait qu'il n'y ait aucune supervision sur les moyens de transports ou leurs conducteurs impose une opacité informationnelle chez les directeurs des compagnies ou entreprises de ses derniers. Comment y remédier ?

Une solution serait d'installer des traceurs GPS (trackers) sur les véhicules et superviser leurs déplacements à distance. Ceci n'est pas rentable, que ce soit en terme de maintenance ou d'installation, car représentant un budget important pour la société, ajouter à cela, les fonctionnalités de ces dispositifs ne se limitent qu'au tracking.

Nous proposons une solution basée sur une application Android utilisant la technologie 3G pour tracer l'itinéraire de ces véhicules couplé à un serveur distant qui propose des services tels que : la surveillance de la vitesse, le calcul des trajets, la gestions des zones blanches, le signalement des pannes ou autres imprévus, la ges-

tion des périodes de pauses pour les conducteurs et finalement, une visualisation en temps réel et fidèle des trajets actuels ou passés de chaque véhicule. Il va sans dire qu'un tel outil sera influent sur le processus de prise de décision de la direction tout en ayant un impact minime sur le budget.

Tout au long de ce projet, nous allons présenter la conception, le développement et le test de cette application.

Chapitre 1

Introduction aux technologies de tracking et État de l'art

L'objectif de ce chapitre est de présenter les aspects principaux et l'état de l'art en rapport avec notre thème, ainsi que les technologies utilisées pour le tracking des véhicules. Elle contient principalement deux études du marché, la première sur les Trackers, leurs types et impacts dans le marché international, la deuxième sur les smartphones, les marques dominantes, les divers prix et une section destinée à décrire les technologies en rapport au tracking.

1 Études des marchés

1.1 Étude du marché des Trackers

Un traceur (Tracker) de véhicule est un appareil électronique placé à l'intérieur du ou des véhicules dans le but de tracer leur localisation tout en vérifiant différentes autres informations concernant l'état du véhicule (voir Figure 1.1). Avec la croissance du marché de l'automobile ces dernières années, le marché des traceurs GPS c'est vu attribué des chiffres intéressants, il est parvenu à se classer parmi le Top 3 du marché de l'électronique. La taille de ce marché a atteint 8 milliards de dollars américains en 2015, et selon des prédictions, ce chiffre est anticipé à 22 milliards de dollars américains en 2022 suivant un TCAC (Taux de Croissance Annuel Composé) de plus de 20% entre 2016 et 2023 [8]. En Algérie, la géolocalisation fait ses premiers pas. Les premiers appels d'offres intéressants voient le jour (Sonatrach). Les professionnels s'accordent à dire que moins de 10% du marché sont couverts. Les entreprises du sous-secteur sont prêtes à réagir aussitôt que les contraintes légales s'atténuent. Le ministre des Postes et Télécommunications promet de démocratiser la navigation et le GPS pour atteindre le grand public. Avec plus d'une cinquantaine de véhicules volés récupérés grâce à cette solution, l'intérêt économique est indéniable. Si les opérateurs essaient de pallier aux lacunes de la cartographie, les problèmes de débit de connexion et ses coupures se sont considérablement atténués. Les ventes de traceurs GPS a été divisé en 2 catégories après l'apparitions des Smartphones : les traceurs physique (Hardware Tracker), et les traceurs logiciels (Software Tracker).





FIGURE 1.1 – Exemple d'un type de traceur physique disponible sur le marché

1.1.1 Traceur physique

Les traceurs physiques ont permis au marché de voir le jour. Les entreprises mères de ces derniers les proposent selon différents plans :

- La vente directe : certaines entreprises vendent des dispositifs de tracking à installer dans les véhicules, il existe 2 types de dispositifs, ceux qui utilisent des puces téléphoniques pour communiquer avec leurs propriétaires grâce à des SMS (et même des appels téléphoniques dans certains dispositifs avec microphone intégré), et ceux qui utilisent des applications fournies par leurs entreprises et qui ne fonctionnent qu'avec ces mêmes dispositifs uniquement (architectures fermées).
- Abonnement : les dispositifs ne sont plus vendus intégralement, mais "loués" mensuellement (ou annuellement, tout dépend du plan choisi) ...

Le coût d'installation d'un système de traçage de véhicule dépend de plusieurs facteurs. Notamment la taille de la flotte et le nombre d'options offertes par le système. Le coût sera également différent pour l'achat ou pour la location des appareils. La plupart des sociétés estiment qu'un contrat de location de courte durée allant de 12 à 18 mois est la meilleure solution. Ainsi, si les patrons des entreprises décident de changer de prestataires pour une raison ou une autre, ils ne sont pas bloqués par un contrat de longue durée. C'est aussi avantageux comparé à l'option d'achat. En effet, dans ce cas, si ils décident de changer de prestataires, ils ne pourront alors plus utiliser leurs appareils car ils ne seront pas compatibles.

Prenons exemple de 3 fournisseurs, 1 algérien local (Sarl LVSC Méditerranée) et 2 français (Géoflotte et Géotraceur). Géotraceur contrairement aux deux autres, ne fait que vendre les traceurs hardware avec des prix allant de 12.000 dinars jusqu'à 56.000 dinars, Géoflotte quant à lui, propose un abonnement de 11.000 dinars le premier

mois puis 1.500 DA/mois, et nécessite 10.000 dinars de frais pour l'installation du traceurs ou de la balise. Pour LVSC le prix du traceur est à partir de 40.000 DA. Des modules complémentaires liés à la gestion de flottes sont payants. A cela s'ajoute un abonnement qui coûte approximativement les 1.500 DA/mois par véhicule pour le réseau mobile et dix fois plus pour le réseau satellitaire exploité généralement par des clients des zones du Sud (Les chiffres ne sont pas exacte suite aux conversions euro/dinars).

1.1.2 Traceur logiciel

Les traceurs logiciels, quant à eux, ont vu le jour avec l'apparitions des Smartphones, contrairement aux traceurs physiques, ils sont beaucoup moins chers, voir gratuit, et ils ne nécessitent que des Smartphones avec 3G (voir section 3 chapitre 2) pour fonctionner, mais leurs côté application/amateur les rends méprisé quant à leurs efficacités. Leur marché n'est pas dominé par de grandes entreprises, étant donnée que les traceurs physiques rapportent beaucoup plus, et il n'est pas précis non plus vus la facilité de conception de celle ci, toute personne s'y connaissant un peu en développement mobile pourrait développé son propre système.

1.2 Étude du marché des Smartphones

Le marché algérien des smartphones est sans surprise dirigé par les prix, où les fonctionnalités multimédia ne figurent pas parmi les priorités du consommateur. Les algériens ne semblent pas s'intéresser aux dernières sorties de téléphones ni aux grandes marques. Pourtant, il aurait fallu attendre la fin 2011 pour que celà change. La moyenne des prix des terminaux les mieux vendus, toutes gammes confondues, avait en effet avancé d'environ 4.000 DA à 9.000 DA en un an d'intervalle. Le moyen de gamme, épaulé par les téléphones double SIM, a opéré une percée certes minoritaire mais nouvelle par rapport aux périodes antérieures. Dans le segment haut de gamme, l'iPhone a longtemps mené la danse, puis le marché s'est enrichi d'offres plus compétitives. Encore une fois, l'iPhone a longtemps résisté face à une concurrence objectivement plus attractive tel que le Galaxy de Samsung ou encore la multi-gamme offerte par la marque algérienne Condor . Vers 2015, plusieurs marques de téléphonie mobile tentèrent de se positionner sur le marché algérien, et qui majoritairement sont d'origine chinoise ou française tel que Wiko, Oppo ou encore doogee. La France est représenté par 2 constructeurs, Mobiwire qui arrive à proposer des smartphones à moins de 18.000 dinars. Wiko, qui propose des smartphones à moins de 7.500 dinars mais peut atteindre les 38.000 dinars pour les plus performants. Il y'a aussi les deux géants chinois ZTE et Huawei qui ont noué des partenariats avec les opérateurs de téléphonie mobile nationaux, notamment dans le domaine des clés ou modems Internet. Avec des prix imbattables (entre 5.000 et 15.000 dinars) mais une qualité qui n'inspire pas encore pleine confiance, les smartphones de ces marques chinoises permettent de démocratiser l'usage des téléphones intelligents et la diffusion de l'Internet mobile à travers les différentes couches sociales.

En 2016, Condor a réaliser un chiffre d'affaire de presque un milliard de dollars et qui lui a attribué 50% de part du marché des smartphone en Algérie [4]. Les prix des smartphones varient selon plusieurs critères : Écran, Processeur, Mémoire vif, Mémoire de stockage, Système d'exploitation, appareil photo, et plusieurs di-

zaine d'autres caractéristiques, comme le smartphone de condor Plume P4 PGN409 qui vaut 12.900 Dinars et le Voyager2 DG310 de Doogee qui en vaut 12.800 Dinars, biens qu'il est qu'une différence de 100 Dinars, mais leurs caractéristiques sont presque toute différentes, du système d'exploitation, à la taille, à la résolution des caméra...etc. Les prix modérés des smartphones représente un plan économique non négligeable pour ce modèle contrairement au autres dispositifs.

2 Technologies utilisées

La création d'un système de tracking nécessite l'utilisation combinée de plusieurs outils ou technologies afin d'assurer un fonctionnement correcte et optimale de l'appareil. Ce nombre d'outils varie d'un tracker à un autre, c'est pour celà que certains d'entre eux sont des éléments de bases, tandis que d'autres varient en fonction des besoins du client. Pour notre modèle, la base est constitué de 3 technologies :

2.1 Globale Positioning System

Depuis le début de l'ère spatiale, la conquête de l'espace pour des fins militaires (Notamment l'espionnage) est devenue une priorité imminente. De multiples projets ont vu le jour, tel que les balises Luna (Unions Soviétique) ou encore le satellite Corona renommé Discoverer (États Unis) pour ne citer que ceux là. Ces derniers avaient tous un point en commun, essayer de récupérer la géolocalisation précise d'objectifs fixés par chaque côté (Bases, mouvements des véhicules, etc...).

Pendant la guerre froide, les États Unis ont établis un projet de recherche militaire dont le but est d'envoyer plusieurs satellites qui gravitent dans l'espace en orbite et qui émettent des ondes interceptables par des capteurs, qui permettent ensuite de détecter leurs positions géographiques. Ce projet est devenu complètement fonctionnel et publique après la fin de la guerre froide, on le nomma "le GPS" .

2.1.1 Définition du GPS

Le GPS, diminutif de Global Positioning System (ou en français : Système mondial de positionnement), est une constellation (ou groupe) de satellites qui gravitent en orbite autour de la terre, il fournit aux personnes leurs géolocalisations grâce à des capteurs GPS qui interceptent les ondes envoyées par ce dernier.

Ce système est mis en place par le département de la Défense des États-Unis à des fins militaires. Il est très rapidement apparu que des signaux transmis par les satellites pouvaient être librement reçus et exploités, et qu'ainsi un récepteur pouvait connaître sa position sur la surface de la Terre, avec une précision sans précédent, dès l'instant qu'il était équipé des circuits électroniques et du logiciel nécessaires au traitement des informations reçues. Une personne munie de ce récepteur peut ainsi se localiser et s'orienter sur terre, sur mer, dans l'air ou dans l'espace au voisinage proche de la Terre [11].

2.1.2 Composition du GPS

Le système GPS est un système plus ou moins complexe et qui se compose de trois grande parties nommée segments comme le montre la figure 1.2.

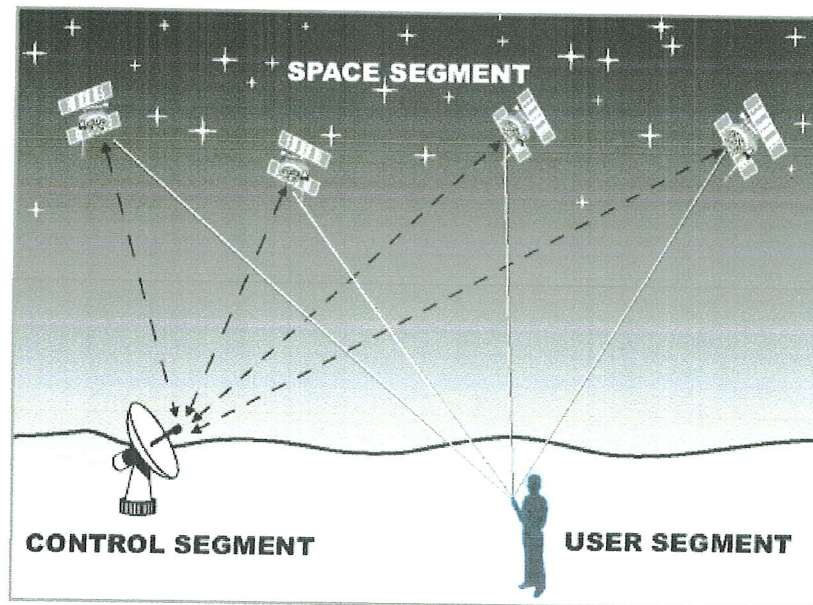


FIGURE 1.2 – Schéma représentant la composition du GPS

2.1.2.1 Segment spatial (Space Segment) Le segment spatial GPS se constitue d'une constellation de satellites transmettant des signaux radio aux utilisateurs. Les États-Unis s'engagent à maintenir la disponibilité d'au moins 24 satellites GPS opérationnels, soit 95% du temps. Pour assurer cet engagement, l'Air Force a permis le vol de 31 satellites GPS opérationnels depuis quelques années pour maintenir la couverture chaque fois que les satellites de base sont desservis ou désaffectés. Les satellites supplémentaires peuvent augmenter les performances GPS, mais ne sont pas considérés comme une partie de la constellation centrale [7]. Ces satellites volent en orbite terrestre moyenne (MEO) à une altitude d'environ 20 200 km. Chaque satellite fait le tour de la Terre deux fois par jour.

Ils sont disposés en six plans orbitaux avec des distances égales entre eux et la terre. Chaque plan contient quatre slots occupées par des satellites de base. Cet arrangement à 24 emplacements garantit la perception d'en moins 4 satellites de n'importe quel point de la terre par un utilisateur. En juin 2011, l'Air Force a complété avec succès une expansion de constellation GPS connue sous le nom de la configuration "Expandable 24". Trois des 24 slots ont été agrandies et six satellites ont été repositionnés, de sorte que trois des satellites supplémentaires sont devenus une partie de la ligne de base de la constellation. En conséquence, le GPS fonctionne efficacement comme une constellation à 27 emplacements avec une couverture améliorée dans la plupart des régions du monde.

2.1.2.2 Segment de contrôle (Control Segment) Le segment de contrôle, c'est la partie qui permet le pilotage, le contrôle, la surveillance ainsi que le bon fonctionnement du GPS. Elle se compose d'un réseau mondial d'installations au sol qui suivent les satellites GPS, surveillent leurs transmissions, effectuent des analyses et envoient des commandes et des données à la constellation. Le segment de contrôle opérationnel actuel comprend une station de contrôle maître, une station de contrôle maître alternative, 11 antennes de commande et de contrôle et 16 sites de surveillance

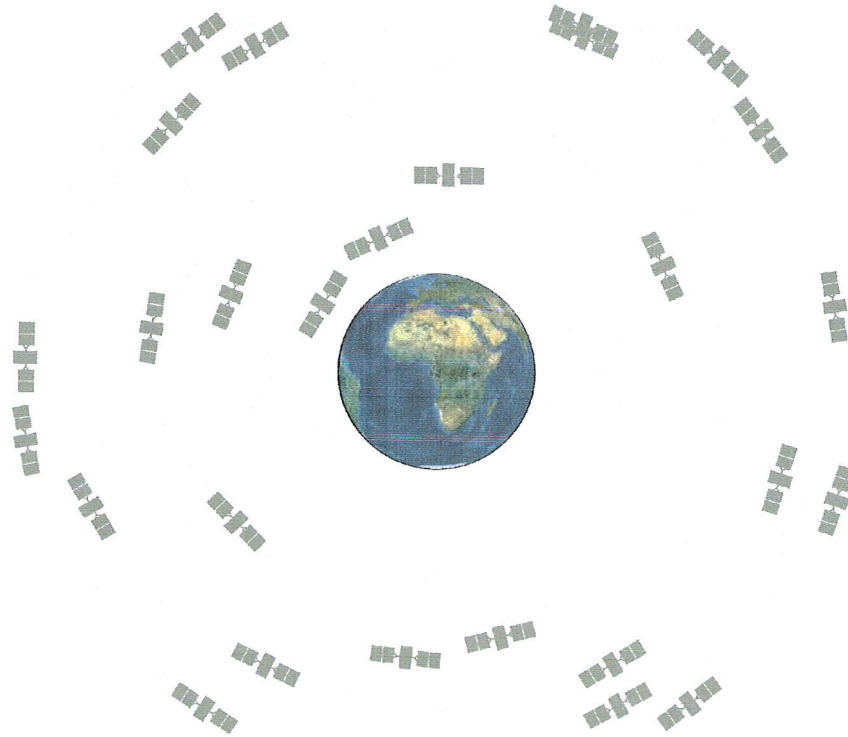


FIGURE 1.3 – Schéma représentant une constellation de 24 satellites en orbite autour de la terre

comme le montre la figure 1.4.

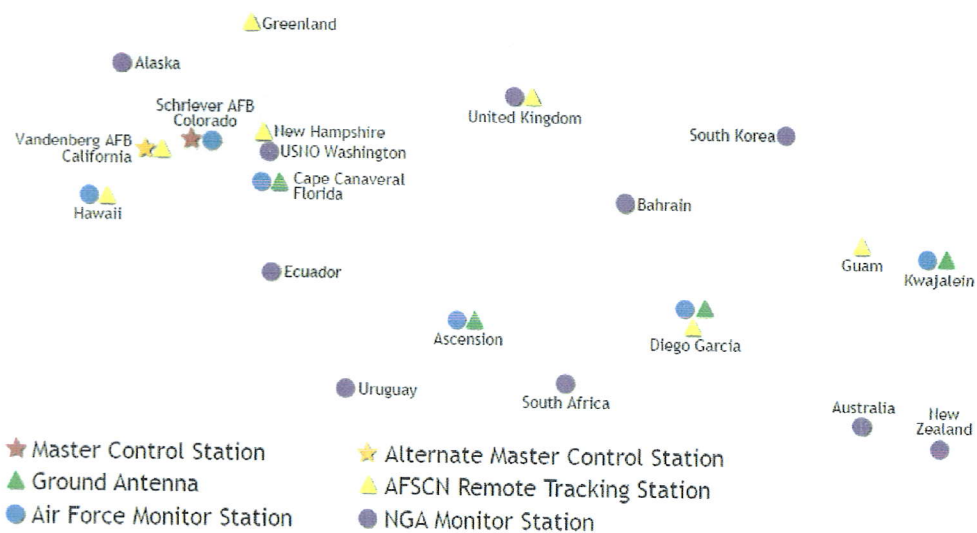


FIGURE 1.4 – Schéma représentant les localisations des éléments du Segment de contrôle sur la carte du monde.

2.1.2.2.1 Station de contrôle principale (MCS) La station de contrôle principal dans le Colorado est l'endroit où on effectue les fonctions du segment de contrôle principal, fournissant la commande et le contrôle de la constellation GPS. Elle génère et télécharge des messages de navigation et assure le fonctionnement et la précision de la constellation des satellites. Elle reçoit des informations de navigation des stations de surveillance, utilise ces informations pour calculer les emplacements précis des satellites GPS dans l'espace, puis télécharge ces données sur les satellites. Elle surveille les messages de navigation et l'intégrité du système, permettant à l'Air Force de déterminer et d'évaluer l'état de santé de la constellation GPS. L'Air Force utilise cette station pour effectuer la maintenance des satellites et la résolution d'anomalie. En cas de panne de satellite, elle peut repositionner des satellites pour maintenir une constellation GPS optimale.

2.1.2.2.2 Stations de surveillances Les stations de surveillance suivent les satellites GPS lorsqu'ils passent au-dessus et relient leurs observations à la station de contrôle principale. Les stations de surveillance collectent les données atmosphériques, les mesures de portée / porteuse et les signaux de navigation. Les sites utilisent des récepteurs GPS sophistiqués et sont exploités par le MCS. Il existe 16 stations de surveillance situées à travers le monde, dont six de l'armée de l'air et 10 de la National Geospatial-Intelligence Agency.

2.1.2.2.3 Antennes terrestres Les antennes terrestres sont utilisées pour communiquer avec les satellites GPS afin de les contrôler. Elles sont responsables des transmissions de commandes normales aux satellites. Il existe quatre sites d'antennes terrestres GPS dédiés localisés avec les stations de surveillance de Kwajalein Atoll, de l'île de l'Ascension, de Diego Garcia et de Cap Canaveral. De plus, le segment de contrôle est connecté aux sept stations de suivi à distance du Réseau de contrôle par satellite de l'Air Force dans le monde entier, augmentant la visibilité, la flexibilité et la robustesse pour la télémétrie¹, le suivi et la commande.

2.1.2.3 Segment utilisateur (User Segment) Plus connu sous le nom de récepteur GPS, c'est lui qui effectue les calculs de coordonnées à partir des informations reçues des satellites.

Il comprend l'équipement du personnel militaire et des civils qui reçoivent des signaux GPS. L'équipement militaire des utilisateurs de GPS a été intégré dans les combattants, les bombardiers, les pétroliers, les hélicoptères, les navires, les sous-marins, les citernes, les jeeps et les équipements des soldats. En plus des activités de navigation de base, les applications militaires du GPS incluent la désignation de cible, le soutien aérien rapproché, les armes «intelligentes» et les point de rendez-vous. Avec la popularité des récepteurs GPS au cours des dernières années, la communauté civile a créé son propre segment d'utilisateurs. Le GPS est utilisé par les avions et les navires pour la navigation en route. Les systèmes de suivi GPS sont utilisés pour acheminer et surveiller les fourgonnettes de livraison et les véhicules d'urgence. Dans une méthode appelée "agriculture de précision", le GPS est utilisé pour guider avec précision les machines agricoles engagées dans le labourage, la

1. télémétrie : c'est un procédé technique permettant de mesurer la distance d'un objet lointain par utilisation d'éléments optiques, acoustiques ou radioélectriques

plantation, la fertilisation et la récolte. Le GPS est disponible comme aide à la navigation automobile et est utilisé par les randonneurs et les chasseurs. La plupart des smartphones comportent une carte GPS ou des applications de navigation. Parce que l'utilisateur GPS n'a pas besoin de communiquer avec le satellite, le GPS peut servir un nombre illimité d'utilisateurs.

2.1.3 Fonctionnement du GPS

Les récepteurs GPS utilisent la trilatération² qui permet de déterminer la position de l'utilisateur, sa vitesse ainsi que son altitude. Leurs positions GPS sont définies

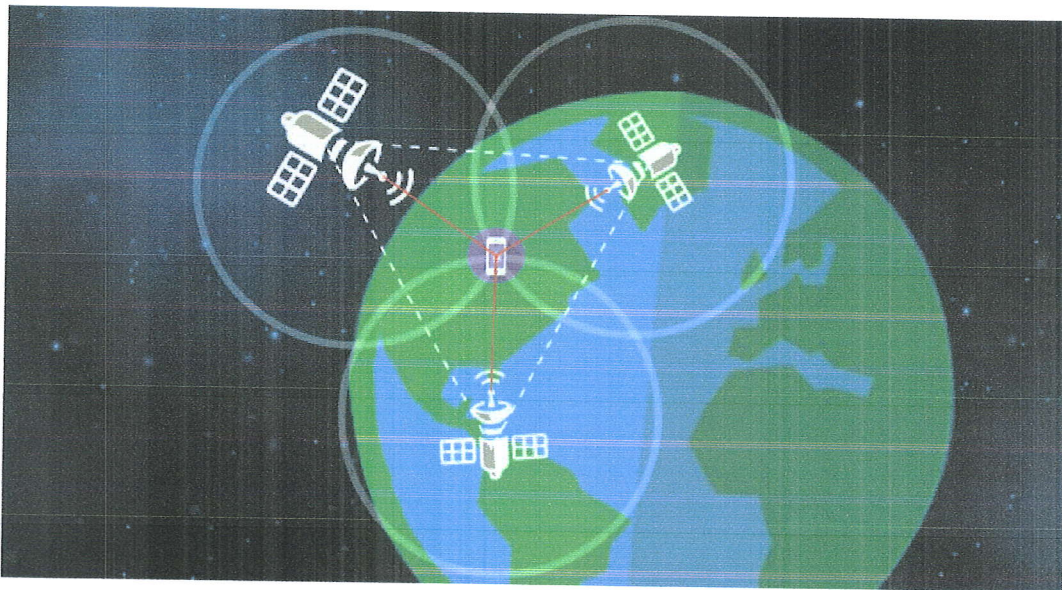


FIGURE 1.5 – Exemple du calcul d'un point sur terre avec 3 satellites

par le calcul des distances entre les récepteurs et trois satellites GPS ou plus comme dans la figure 1.5. Chaque satellite est équipé d'une horloge atomique. Lors de la première mise en marche, les dispositifs GPS subissent une période d'initialisation durant laquelle ils acquièrent les signaux des satellites et synchronisent leurs horloges avec les horloges atomiques des satellites. Ces mêmes dispositifs reçoivent et analysent constamment des ondes radio des satellites, calculant des distances précises de chaque satellite tracé.

2.1.4 Alternatives

Étant donné le nombre de satellites de différents pays qui sont en orbite autour de la Terre, plusieurs autres alternatives que le GPS ont vu le jour, notamment le GLONASS (globalnaïa navigatsionnaïa spoutnikovaïa sistema) en russe, et qui signifie système global de navigation satellitaire en français, qui est un système de navigation russe d'origine soviétique, utilisé et géré par les forces spatiales russes, il a été opérationnel en 1996, mais suite à des handicaps le système est devenu

2. La trilatération est une méthode mathématique permettant de déterminer la position relative d'un point en utilisant la géométrie des triangles. Elle utilise les distances entre un minimum de deux points de référence.

imprécis jusqu'à ce que le président de la Fédération de Russie Vladimir Putin ne le restaure et le qualifie de priorité gouvernemental en 2010. Le système Beidou, nommée aussi COMPASS, qui est un système de navigation chinois qui est prévu pour 2020, a commencé à voir le jour en 2000, et déclaré fonctionnel en 2003. Il permet la géolocalisation uniquement en Chine et des zones avoisinantes, il a une précision d'environ 100 mètres. Le système Galileo, qui est un système conçu par l'Union Européen et qui est prévu en 2020 tout comme Beidou, il a une précision de 4 mètres horizontalement et de 8 mètres en altitudes pour la version gratuite, tandis que la pour version payante, un niveau de qualité supérieur est prévu. Il existe aussi un système de navigation indien, qui est le IRNSS (Indian Regional Navigation Satellite System) ou système indien de navigation régionale par satellite qui a été prévu fin 2016, tel que Beidou, il a une couverture régional qui s'étend jusqu'à une distance de 1500 à 2000 Km de ses frontières.

2.2 Android

2.2.1 Définition d'Android

Android est un système d'exploitation « mobile » développé par Google, basé sur le noyau Linux et conçu principalement pour les appareils mobiles à écran tactile tels que les smartphones et les tablettes. L'interface utilisateur d'Android est principalement basée sur la manipulation directe. En plus des écrans tactiles, Google a développé Android TV pour les téléviseurs, Android Auto pour les voitures et Android Wear pour les montres bracelets, chacun avec une interface utilisateur spécialisée. Différents systèmes Android sont également utilisés sur les ordinateurs portables, consoles de jeux, appareils photo numériques, et autres appareils électroniques. Le système proposé sera développé dans sa version cliente pour cet OS.

2.2.2 Historique d'Android

Android Incorporation a été fondée à Palo Alto, Californie, États-Unis en Octobre 2003 par Andy Rubin, Rich Miner, Nick Sears et Chris White. Google a pris en charge Android Incorporation en août 2005, ce qui rend Android Incorporation une propriété entière de Google Incorporation. Après cela, en Novembre 2007 Android a lancé une plate-forme pour faire de nouvelles applications, jeux et autres logiciels en lançant Android bêta (kit de développement). Le premier appareil Android a été lancé le 23 Septembre 2008, le HTC Dream G1 qui utilise Android 1.0, vient après, une mise à jour Android 1.1 qui a été publiée pour T-Mobile G1 seulement. Le 5 novembre 2007, l'Open Handset Alliance, plusieurs sociétés telles que Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, T-Mobile et Texas Instruments ont développé des standards ouverts pour les appareils mobiles et leur premier produit était Android, une plate-forme de périphérique mobile construite sur le noyau Linux (version 2.6).

2.2.3 Caractéristiques d'Android

Android possède un environnement de développement complet contenant : un émulateur, un débogueur, un analyseur de mémoires et de performances. Le déve-

veloppement d'application pour cette plateforme utilise le Framework Java, et pour l'exécution de ces applications, des machines virtuelles ont été spécialement créées et qui sont Dalvik et ART.³ Pour les graphiques, il utilise la librairie graphique 2D ainsi que 3D basé sur OpenGL ES, avec la possibilité d'accélération matériel. Il supporte plusieurs formats audio/vidéo/image tel que MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF. Pour la connectivité, il supporte plusieurs systèmes tel que GSM, EDGE, 3G, 4G, Bluetooth ou encore Wi-Fi. Il utilise une base de données SQL, dont SQLite pour le stockage des données, son navigateur web est basé sur le moteur de rendu Webkit. Il est capable d'utiliser la Caméra, le GPS, l'accéléromètre, empreinte digitale....

2.2.4 Android Lollipop 5.0

Nous avons choisi de décrire exclusivement cette version, parce que c'est la plus utilisée au monde en 2016, ainsi qu'en Algérie, et c'est la version qu'en utilisera dans notre application finale. Elle a été dévoilée sous la version L le 25 juin 2014 dans le Google I/O, et elle sera enfin présentée le 15 octobre 2014 en tant que Version 5.0. Cette version a été placée sur Android TV, Android Auto et Android Wear qui, à la fin, a permis à Android de fonctionner dans les voitures, les téléphones, les montres, les TV et les tablettes.

Par rapport aux anciennes versions, cette-ci propose un nouveau design. Elle a une nouvelle interface qui facilite la navigation ; basée sur les ombres et le mouvement. Elle a un nouveau système de notifications qui s'affichent sur l'écran de verrouillage sous la forme de fiches claires et distinctes, ainsi que le nouveau système Multi-écran qui permet l'utilisation et la synchronisation d'Android dans plusieurs appareils. Il permet donc le passage d'un appareil à un autre avec possibilité de reprise d'activité récente sur le dernier.

2.2.5 Architecture d'Android

Android possède une architecture plus ou moins complexe comme le montre la figure 1.6 et qui repose sur le noyau ou kernel Linux, la figure suivante ce lit de bas en haut (du Bas niveau jusqu'au Haut niveau)

2.2.5.1 Noyau Linux La base de la plate-forme Android est le noyau Linux. Par exemple, Android Runtime (ART) s'appuie sur le noyau Linux pour les fonctionnalités sous-jacentes telles que le thread et la gestion de mémoire de bas niveau. L'utilisation d'un noyau Linux permet à Android de tirer parti des principales fonctionnalités de sécurité et permet aux fabricants de périphériques de développer des pilotes matériels pour un noyau bien connu

2.2.5.2 Hardware Abstraction Layer (HAL) La couche d'abstraction matérielle (HAL) fournit des interfaces standard qui exposent les fonctionnalités matérielles des périphériques au Framework API Java de niveau supérieur. Le HAL se compose de plusieurs modules de bibliothèque, et chacun d'eux implémente une

3. Dalvik et ART permettent de faire tourner les applications sur différents smartphones Android, quel que soit le modèle.

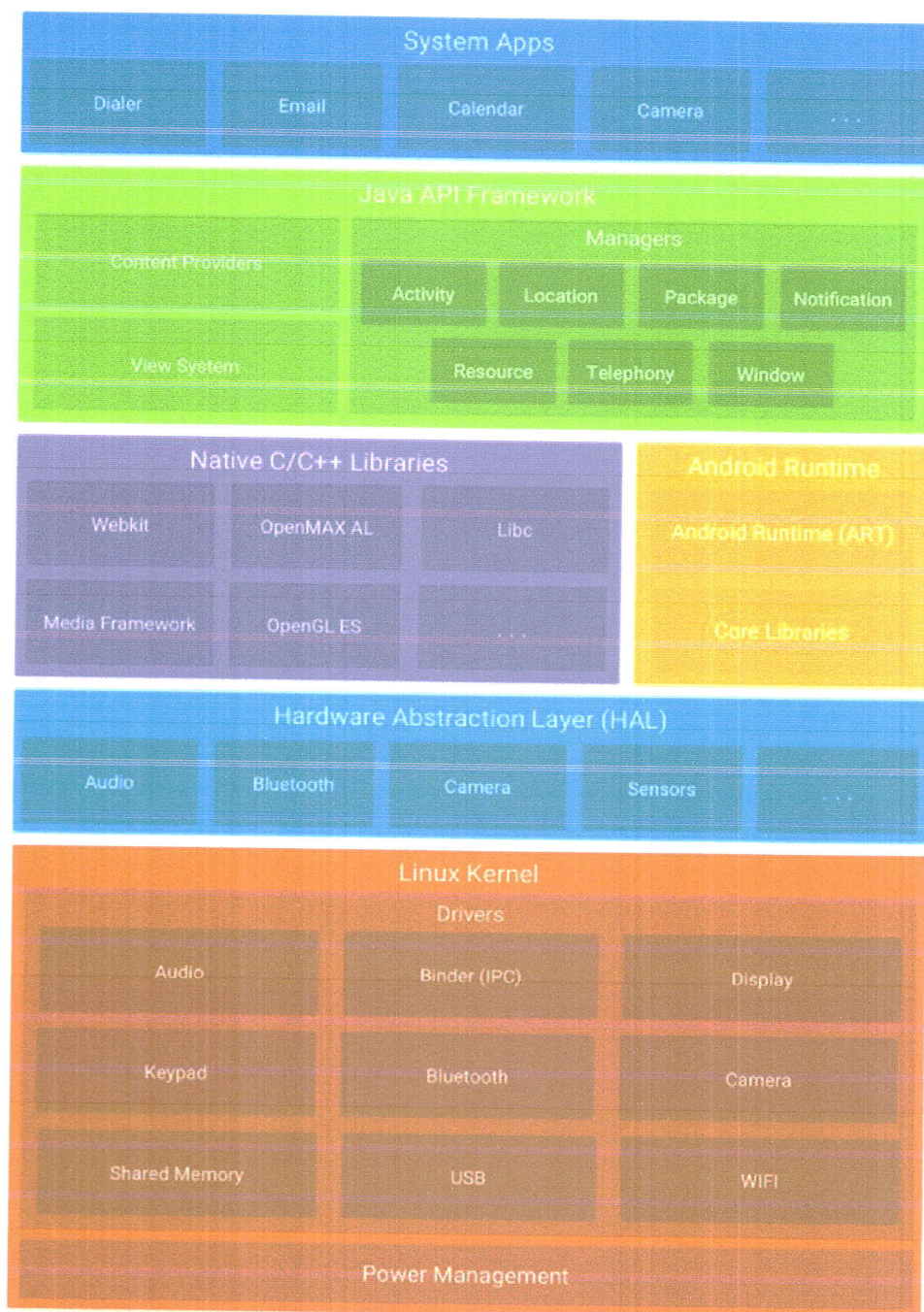


FIGURE 1.6 – Schéma représentant l'architecture complète d'Android

interface pour un type spécifique de composant matériel, tel que la caméra ou le module Bluetooth. Lorsqu'un Framework API effectue un appel pour accéder au matériel du périphérique, le système Android charge le module de bibliothèque pour ce composant matériel.

2.2.5.3 Android Runtime Pour les appareils fonctionnant sous Android version 5.0 (API niveau 21) ou supérieur, chaque application s'exécute dans son propre processus et avec sa propre instance de l'Android Runtime (ART). ART est conçu pour exécuter plusieurs machines virtuelles sur des périphériques à mémoire insuffisante en exécutant des fichiers DEX, un format de bytecode spécialement conçu pour Android optimisé pour une empreinte mémoire minimale.

2.2.5.4 Native C/C++ Libraries De nombreux composants et services principaux du système Android, tels que ART et HAL, sont construits à partir du code natif qui nécessitent des bibliothèques natives écrites en C et C++. La plate-forme Android fournit des Java framework APIs pour exposer les fonctionnalités de certaines de ces bibliothèques natives aux applications. Par exemple, vous pouvez accéder à OpenGL ES via l'API Java OpenGL afin d'ajouter une prise en charge du dessin et de la manipulation des graphiques 2D et 3D dans votre application.

2.2.5.5 Java API Framework La fonctionnalité du système d'exploitation Android est réalisée grâce à des API écrites en langage Java. Ces API forment les blocs de construction dont vous avez besoin pour créer des applications Android en simplifiant la réutilisation des composants et des services de base du système, notamment :

- Un système de visualisation riche et extensible que vous pouvez utiliser pour créer l'interface utilisateur d'une application, y compris des listes, des grilles, des zones de texte, des boutons et même un navigateur Web intégré
- Un gestionnaire de ressources, fournissant l'accès à des ressources non codées telles que des chaînes localisées, des graphiques et des fichiers de mise en page
- Un Notification Manager qui permet à toutes les applications d'afficher des alertes personnalisées dans la barre d'état
- Un gestionnaire d'activités qui gère le cycle de vie des applications.
- Des fournisseurs de contenu qui permettent aux applications d'accéder aux données provenant d'autres applications, telles que l'application Contacts, ou de partager leurs propres données ...

2.2.5.6 Applications système d'Android Android est arrivé avec un ensemble d'applications de base par défaut pour le courrier électronique, la messagerie SMS, les calendriers, la navigation sur Internet, les contacts ... etc. Une application tierce peut devenir le navigateur par défaut de l'utilisateur, SMS Messenger ou même le clavier par défaut. Les applications système fonctionnent à la fois comme des applications pour les utilisateurs et pour fournir des fonctionnalités clés auxquelles les développeurs peuvent accéder à partir de leur propre application. Par exemple, si votre application souhaite envoyer un message SMS, vous n'avez pas besoin de créer cette fonctionnalité vous-même - vous pouvez plutôt invoquer l'application SMS déjà installée pour envoyer un message au destinataire que vous spécifiez.

3 Téléphonie mobile troisième génération (3G)

Au début des années 70, l'idée d'un appareil téléphonique miniature et mobile avait commencé à devenir concrète, tel l'invention du docteur Martin Cooper, directeur de la recherche et du développement chez Motorola qui en a fait la démonstration dans les rues de New-York le 3 avril 1973 [2, p. 355], cette invention est devenu publique dans les années 80, et finalement, son usage s'est progressivement démocratisé vers les années 90, jusqu'à ce qu'il dépasse le taux d'usage des téléphones fixes au début des années 2000 et qui fini par être un gadget indispensable vers les années 2007-2010.

Tout au long de ce parcours, l'appareil téléphonique s'est vu attribuer plusieurs normes numériques. Au début, c'était des appareils analogiques avec la norme AMPS(Advanced Mobile Phone System) ou NMT(Nordic Mobile Telephone) qu'on appelle la première génération (1G), puis ils ont été remplacés par des appareils numériques utilisant la norme GSM(Global System for Mobile Communications) qui a réussi des améliorations tel que GPRS(General Packet Radio Services) et EDGE(Enhanced Data rates for GSM Evolution) connu aussi sous le nom de deuxième génération (2G), puis enfin vient ceux qui sont extrapolées du 2G par le regroupement international 3GPP et qui sont l'UMTS(Universal Mobile Telecommunications System) et le LTE(Long Term Evolution) nommé respectivement, 3G et 4G. [1] La 3G représente la norme qui a basculé l'usage de la téléphonie mobile.

3.1 Définition de la 3G

3G (diminutif de troisième génération), est la troisième génération des technologies de télécommunication mobiles sans fil utilisant la technologie UMTS. Elle est basée sur un ensemble de normes utilisées pour les appareils mobiles ainsi que les services et réseaux mobiles de télécommunication qui sont conformes aux spécifications de la norme IMT-2000 de l'UIT (Union International des télécommunications for the year 2000). Elle a été approuvée par le gouvernement ainsi que les compagnies de communication. Le premier réseau 3G a été créé par l'opérateur mobile japonais NTT DoCoMo en 1998. Les applications de la 3G se montrent souvent dans la téléphonie vocale sans fil, l'accès internet mobile, l'appel vidéo ainsi que la TV mobile. Les réseaux de télécommunications 3G assurent des services qui fournissent un taux de transfert d'informations d'au moins 200 kb/s. D'autres versions de la 3G ont vu le jour tel que la 3.5G/3G+, 3.75G/3G++, ainsi que la 3.9G (appelé LTE et qui a servi de base pour la 4G), ils profitent d'un meilleur taux de transfert que leur parent.

Son utilité dans notre travail, est qu'elle permet l'accès à internet depuis des téléphones portables Android (Smartphones), et donc elle permet une interaction avec un serveur distant avec une bonne transmission.

3.2 Caractéristiques de la 3G

Les principales caractéristiques techniques de la 3G ont été définies par l'Union Internationale des Communications (UIT) dès le tournant du siècle, avec la publication des normes IMT-2000. La future 3G, selon ce document de travail, devait constituer une norme unifiée au niveau mondial, mais aussi être compatible avec

les réseaux de seconde génération (2G) pour que les téléphones mobiles plus anciens restent joignables. Le débit exigé, quant à lui, allait de 144 Kb/s en utilisation purement nomade jusqu'à 2 Mb/s dans certaines zones restreintes. Pour atteindre ces performances, la 3G utilise une nouvelle bande de hautes fréquences, comprises d'une part entre 1 885 et 2 025 MHz et d'autre part entre 2 110 et 2 200 MHz. Apparue en 2007, la « 3G+ » ou « 3,5G » constitue une 3G dotée d'un débit amélioré, grâce à un nouveau système appelé HSDPA ou High-Speed Downlink Packet Access. Elle utilise une bande de fréquences située aux alentours de 5 000 MHz, et permet d'atteindre (dans des conditions optimales) un débit descendant compris entre 8 et 10 Mb/s. La 3G+ garde toutefois un fort lien de parenté avec la 3G dans la mesure où elle continue à utiliser le même procédé de codage, à savoir le W-CDMA (Wideband Code Division Multiple Access).

3.3 Comparatif aux autres générations

Comme cité précédemment, il existe 4 grande générations de technologies de télécommunication, chaque génération peut comporter plusieurs versions, les principales différences entre ces technologies sont généralement le débit de transmission et la base de la technologie, comme illustré dans figure 1.7.

		Real World (avg)		Theoretical (max)	
		Download	Upload	Download	Upload
2.5G	GPRS	32-48Kbps	15Kbps	114Kbps	20Kbps
2.75G	EDGE	175Kbps	30Kbps	384Kbps	60Kbps
3G	UMTS	226Kbps	30Kbps	384Kbps	64Kbps
	W-CDMA	800Kbps	60Kbps	2Mbps	153Kbps
	EV-DO Rev. A	1Mbps	500Kbps	3.1Mbps	1.8Mbps
3G	HSPA 3.6	650Kbps	260Kbps	3.6Mbps	348Kbps
	HSPA 7.2	1.4Mbps	700Kbps	7.2Mbps	2Mbps
	WiMAX	3-6Mbps	1Mbps	100Mbps+	56Mbps
Pre-4G	LTE	5-12Mbps	2-5Mbps	100Mbps+	50Mbps
	HSPA+	-	-	56Mbps	22Mbps
	HSPA 14	2Mbps	700Kbps	14Mbps	5.7Mbps
4G	WiMAX 2 (802.16m)	-	-	100Mbps mobile / 1Gbps fixed	60Mbps
	LTE Advanced	-	-	100Mbps mobile / 1Gbps fixed	-

FIGURE 1.7 – Tableau comparatif des débits moyens en download et upload des technologies

4 Conclusion

Nous avons discuté dans ce chapitre d'une étude du marché des trackers qui a montré des volumes assez importants (sachant que les exemples ont été faits pour

une seule flotte) ainsi qu'une étude du marché des smartphones. Nous avons aussi discuté des technologies utilisées pour le tracking tel que le GPS et la 3G, mais aussi d'Android.

L'idée serait l'utilisation d'un smartphone Android comme dispositif de tracking, qui utilisera un tracker logiciel (combinaison de GPS et 3G), et qui pourra minimiser les coûts d'un service de tracking hardware tout en renforçant sa sécurité et sa précision.

Chapitre 2

Le modèle Multi-vehicules Tracking

1 Description de MVT

Comme pour chaque projet, il est nécessaire d'établir un modèle fiable qui permet d'assurer le bon fonctionnement du projet. Notre modèle qui se nomme Multi-vehicules Tracking (traduit littéralement par "Le suivi de plusieurs véhicules" en Français) vise à créer un système de tracking complet dans lequel une entité gouvernementale participe également (Voir figure 2.1). Tout d'abord, le serveur sera doté d'un serveur Redis (voir section 1.2.2) dont nous discuterons plus loin ; ce serveur utilisera l'identifiant et la clé symétrique du véhicule. Ensuite, le véhicule ou le dispositif Android (voir section 1.1.1) que le véhicule utilise envoie instantanément une information composée (qu'en détaillera plus tard dans la section technique) au serveur. Celui-ci le stockera dans sa base de données (nous allons introduire notre propre modèle de base de données, voir section 1.2.1), Chaque heure, le serveur envoie un document chiffré (voir section 1) pour chaque véhicule (ID, date, somme) à l'entité gouvernementale (voir section 1.2.3) qui le déchiffre, extrait les informations et les stockent dans sa propre base de données, il chiffrera également chaque somme de document (Hash, en utilisant le MD5), le stockera et renverra le résultat de cryptage au serveur, ce dernier l'utilisera pour signer chaque document, en faisant ceci nous garantissons l'intégrité du document en cas de coïncidence. Après 24 heures, le système sauvegarde tous les documents générés au cours de cette période et les upload vers un serveur cloud¹ (un Backup par compression). L'administrateur a le droit d'accéder au serveur et à la base de données via une interface web simple (html + css + js) pour visualiser les documents générés sous forme de trajets et le suivi des véhicules.

1.1 Les Clients

Techniquement, il existe qu'un seul type de client (Mobile), et qui est l'application Android elle-même, mais étant donnée la large composition de MVT, il existe plusieurs relations Client/Serveur entre 2 serveurs, ainsi que le Client WEB.

1. Le Cloud est une technologie qui permet de mettre sur des serveurs localisés à distance des données de stockage ou des logiciels qui sont habituellement stockés sur l'ordinateur d'un utilisateur, voire sur des serveurs installés en réseau local au sein d'une entreprise.

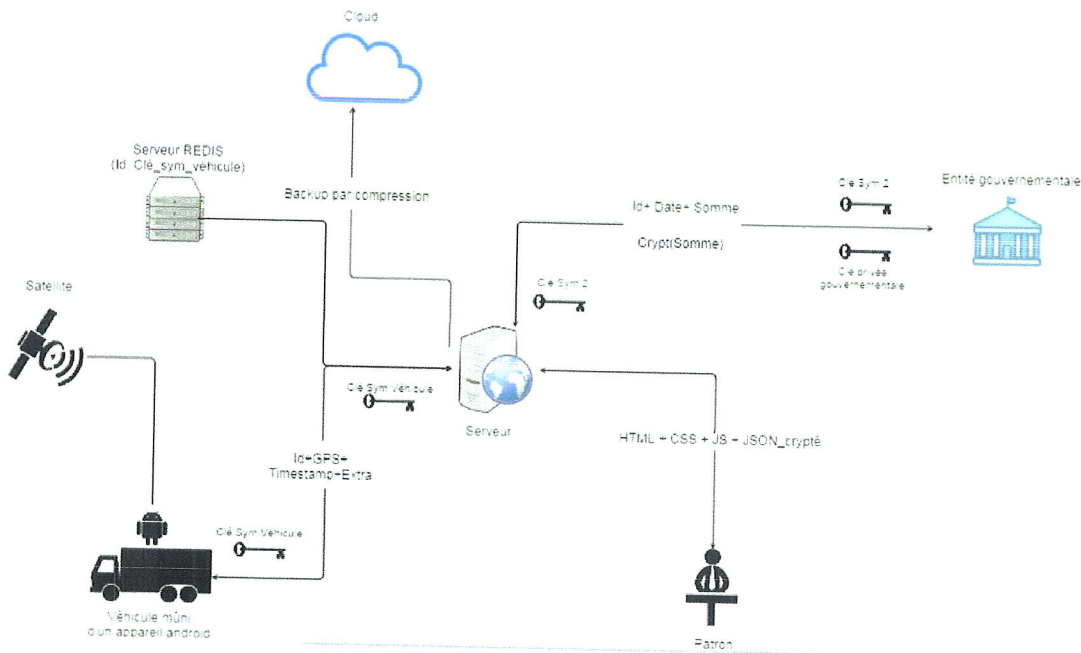


FIGURE 2.1 – Schéma représentant notre modèle

1.1.1 Client Android

C'est le client principal étant donné que c'est l'une des bases du modèle MVT, il servira de tracker GPS. Il est embarqué dans les véhicules. Ce tracker consiste en un smartphone qui possède une version Android minimum 5.0 (Lollipop). En utilisant la 2G/3G nous récupérons les positions géographiques de ce dernier, ainsi que le niveau de batterie du smartphone (qui nous sera utile pour les zones blanches). Bien sûr, à chaque fois que nous récupérons ces informations, nous notons la date courante (Date + heures), et le tout est envoyé au serveur (grâce à la 3G) qui gère de la base de données.

Étant donné la couverture imparfaite de la 3G dans le territoire national, le client subira des coupures d'internet lors des traversées des zones blanches, c'est pour cela que nous avons établi deux modes de fonctionnement pour ce client. Le premier mode, le mode "Online" est le fonctionnement basique de l'application en zone couverte. Le deuxième mode, le mode "Offline" est le mode qui permet de gérer les informations sans présence de 3G, il se base sur la base de donnée SQLite interne d'Android pour stocker les informations obtenus, puis attend la sortie de la zone blanche pour ré-envoyer au serveur.

1.1.2 Client WEB

C'est le client mis à disposition pour les gérants des sociétés et compagnies utilisant notre modèle. Il se présente comme une interface simple et facile à utiliser (qui ne nécessite pas de connaissances approfondies pour pouvoir l'utiliser). Mis à part les interfaces telles que l'inscription et la connexion, ce client possède des pages dédiées à la gestion et la supervision des véhicules disponibles pour une société donnée (la société du patron/client). Sa page principale contiendra une Map Google, avec en paramètres les véhicules voulant suivre, ainsi que 2 mode de supervision, La

supervision en temps réel, et la supervision par intervalle. (Supervision des vitesses de déplacement ainsi que des itinéraires)

La première permet la supervision en directe d'un véhicule ou de plusieurs véhicules, le gérant peut voir les déplacements actuels de ses véhicules. Tandis que la deuxième, permet de tracer l'itinéraire d'un ou de plusieurs véhicule(s) durant un intervalle définis au niveau du site web.

1.2 Les Serveurs dédiés

1.2.1 Serveur central

C'est le cœur de MVT, sa fonctionnement se décompose en 2 processus : processus axé "Base de données", et processus axé "serveur centrale". En premier lieu, il s'occupe de créer et gérer la base de donnée, pour notre modèle, nous avons choisi d'élaborer notre propre base de données (voir figure 2.3). Nous avons choisi une structure arborescente (Hiérarchique voir figure 2.2) pour permettre une organisation facile et une interrogation simple et légère de la base de donnée. Les bases de données hiérarchiques sont un type dont les données sont classées hiérarchiquement, selon une arborescence descendante. Ce modèle utilise des pointeurs entre les différents enregistrements, mais chaque enregistrement n'a qu'un seul possesseur. C'est

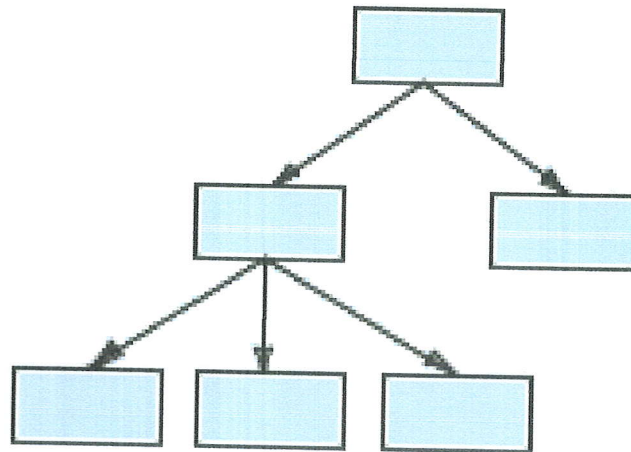


FIGURE 2.2 – Schéma d'une Base de donnée hiérarchique

un modèle adéquat avec les entreprises à structure arborescente (un grand nombre d'organisations économiques et sociales correspondent à ce modèle), ainsi il permet une implémentation facile et simple.

L'utilisation d'une base de donnée de ce type pour notre modèle est avantageuse à cause de la quantité d'informations et de leurs structures. Pour une heure d'informations, le serveur récupérera une quantités importantes, les décompose en arbres ce permet une gestion et un allégement de la base de données (Contrairement à SQL). Pour les premiers fils de la racine (Le dossier père de la base de données), nous avons les noms des sociétés, puis au niveau inférieur les véhicules (le IMEI² de leur trackers pour être exacte) auront pour fils une structure temporelle : les années, puis

2. IMEI : International Mobile Equipment Identity, est un numéro qui permet d'identifier de manière unique chacun des terminaux de téléphonie mobile (ME) GSM ou UMTS.

les mois, puis les jours, et enfin les heures qui seront les éléments qui contiennent les documents (feuilles de l'arbre). En second lieu, c'est l'élément qui relie tout les autres éléments du modèle entre eux.

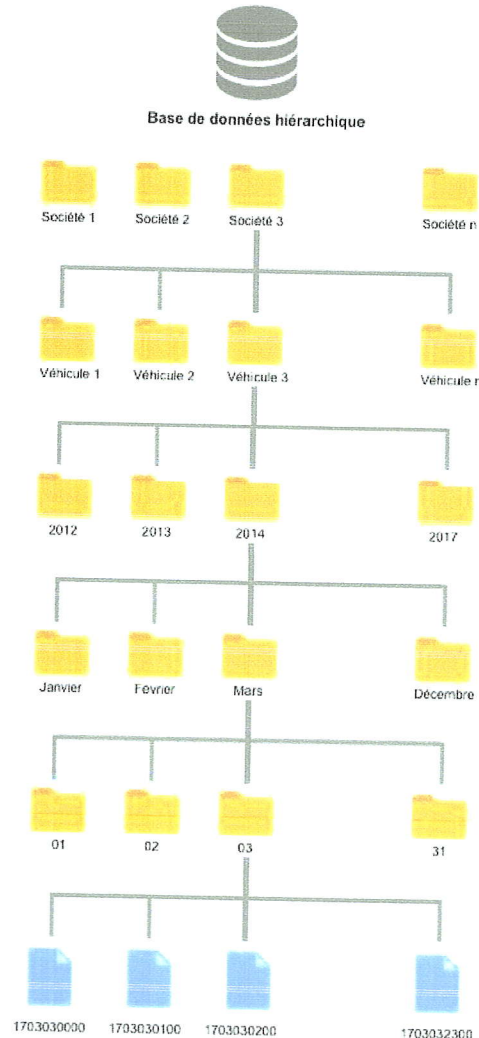


FIGURE 2.3 – Schéma représentant la structure de notre base de données

1.2.2 Serveur REDIS

Redis ou REmote DIctionary Server qui signifie serveur de dictionnaire distant, est un système de gestion de base de données clef-valeur scalable mono-thread en-mémoire (Opère dans la mémoire vif) classé comme étant NOSql (Not Only SQL)³ écrit en langage C, utilisé comme base de données ou cache. Il prend en charge plusieurs type de structures de données tel que les strings, hashes, listes, sets (ensemble), et bitmaps. Il permet l'utilisation des opérations atomiques. L'une des caractéristiques principales de Redis est qu'il conserve l'intégralité des données sur RAM, qui donne d'excellentes performances. Il peut utiliser de la mémoire virtuelle si les

3. NOSql désigne les bases de données qui ne sont pas fondées sur l'architecture classique des bases de données relationnelles

données sont volumineuses. Il offre la possibilité de capture d'état de la base afin d'éviter toute perte de données en cas d'accidents.

Il servira de base de donnée à accès rapide pour stocker les couples ID/Clé symétrique des véhicules.

1.2.3 Entité Gouvernementale

C'est la partie qui gère le côté judiciaire ou légal de l'application. Son côté principal est la signature des documents fournis par le serveur centrale. Avant la fin de l'écriture des documents, le serveur principal calcule la somme du document puis l'envoi en même temps que d'autres informations complémentaires dans un canal sécurisé avec un cryptage symétrique connu du serveur et de l'entité gouvernementale. Cette dernière sera crypté avec une clé symétrique propre à l'entité, et sera signée dans le document. Cette manipulation permet le contrôle d'intégrité et la non répudiation des documents.

Cet élément sera placé dans un serveur attaché à l'agence légale.

1.3 Cloud

Pour éviter toute perte d'informations suivant un problème techniques des serveurs, un backup est prévu en utilisant Amazon Cloud. Chaque fin de journée, la base de données sera compressé et envoyer au Cloud pour synchronisé les informations. L'utilisation de ce Cloud spécifiquement aux lieu des autres est dû aux fait que sa licence gratuite nous fournit le nécessaire dont on a besoin, étant donnée que nous utiliserons ce système une fois par 24h, celà est largement suffisant.

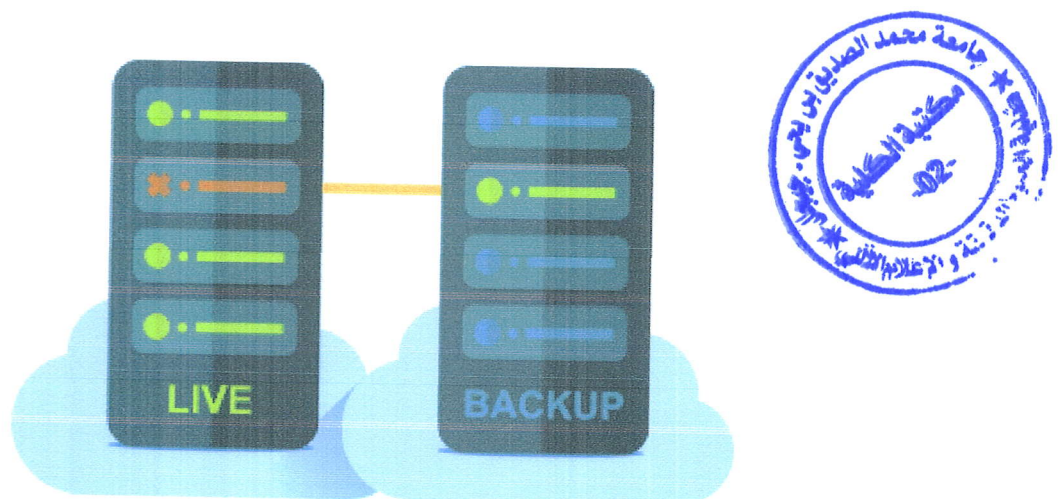


FIGURE 2.4 – Schéma représentant la synchronisation entre le serveur et le cloud

2 Conception de MVT

Comme le montre la figure 2.1, notre modèle possède plusieurs éléments, et ils sont tous reliés par le serveur central, ce qui donne naissance à plusieurs relations.

2.0.1 Relation Client Android / Serveur Base de données

C'est une relation peu complexe, elle nécessite toutefois un canal sécurisé (une Authentification) pour les communications. Tout commence avec l'authentification du Client grâce à une requête pour récupérer la clé pour les communications. Chaque x temps, le Client envoie un vecteur de (position géographique, heure actuelle du Client, identifiant IMEI, Nom de la société mère et le niveau de la batterie du Client). Ces informations seront reçues par le Serveur, qui les traite et les ajoute à la base de données. (Voir figure 2.5 et 2.6)

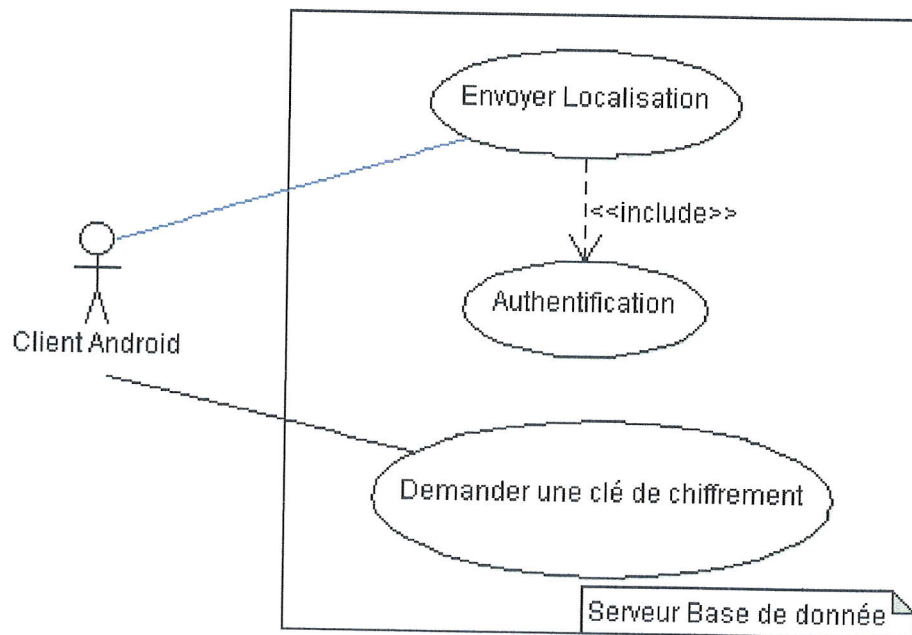


FIGURE 2.5 – Diagramme de cas d'utilisation UML entre le Client Android et le Serveur base de données

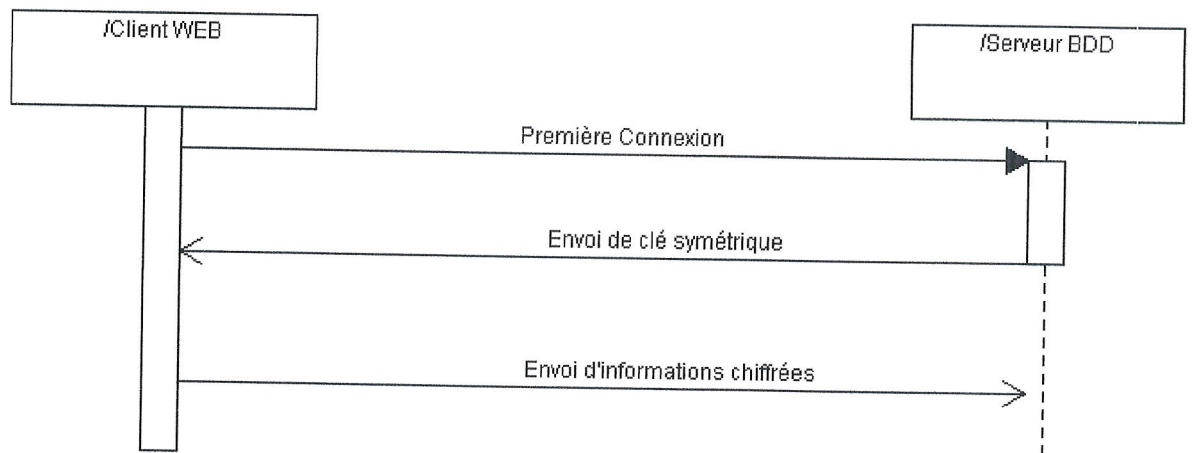


FIGURE 2.6 – Diagramme de séquences UML entre le Client Android et le Serveur base de données

2.0.2 Relation Serveur Base de données / Serveur REDIS

C'est une relation qui se produit lorsque le Serveur Base de données reçoit des informations de la part d'un Client. Grâce à l'identifiant (IMEI) du client, le Serveur base de données interroge le serveur REDIS pour récupérer la Clé de cryptage (voir section 1 chapitre 3) et ainsi, déchiffrer les informations et les stocker dans la base de données. (Voir figure 2.7 et 2.8)

2.0.3 Relation Client WEB / Serveur Base de données

Cette relation représente le point de vue d'un gérant d'une société dans notre modèle, elle lui permettra de surveiller ses véhicules à l'heure souhaitée. Ici, il existe deux types de comptes : 'Admin' ou 'Company' :

Pour se connecter, l'utilisateur doit d'abord accéder à 'login.php' où un formulaire est affiché avec deux entrées pour le nom d'utilisateur et le mot de passe. L'utilisateur entre ses informations et clique sur 'S'identifier', le système vérifiera le nom d'utilisateur et le mot de passe si ils sont correctes l'utilisateur sera redirigé vers la page d'accueil.⁹ L'utilisateur peut aussi afficher une liste de ses conducteurs en cliquant sur 'Conducteurs', un tableau avec une liste de conducteurs et ses informations sera affiché avec deux boutons 'Modifier' et 'Supprimer'. Lorsque nous cliquons sur 'Modifier', un formulaire contenant les informations du conducteur s'affiche où l'utilisateur peut modifier et valider; Lorsque nous cliquons sur 'Supprimer' : le conducteur et ses informations seront supprimés de manière permanente. Type = 'Admin' et type = 'Company' peuvent suivre un/des véhicule(s) à partir de la page d'accueil simplement en choisissant la date de début, la date de fin, le(s) véhicule(s) à suivre, le mode de suivi et cliquant sur le bouton 'Afficher'.

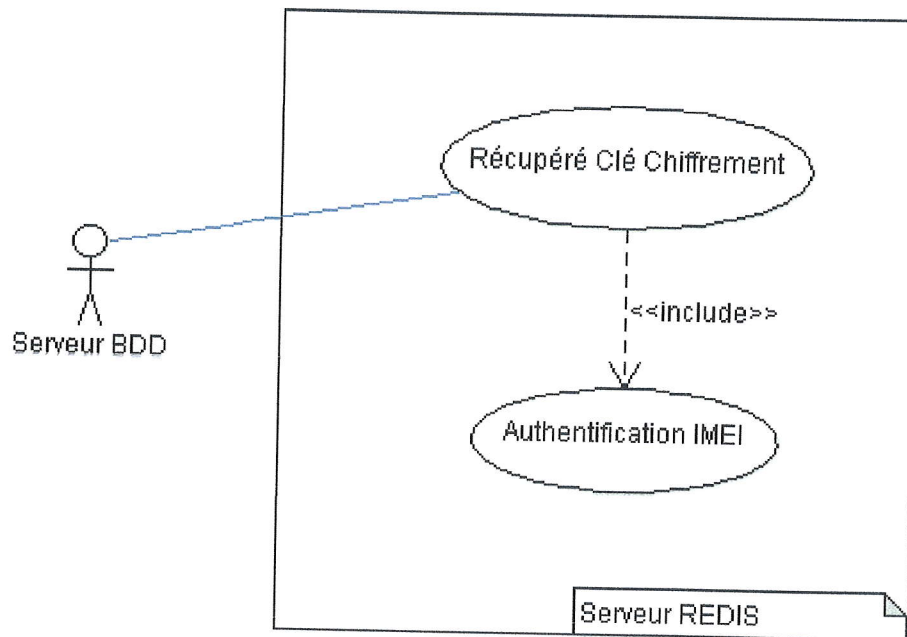


FIGURE 2.7 – Diagramme de cas d’utilisation UML entre le Serveur base de données et le Serveur REDIS

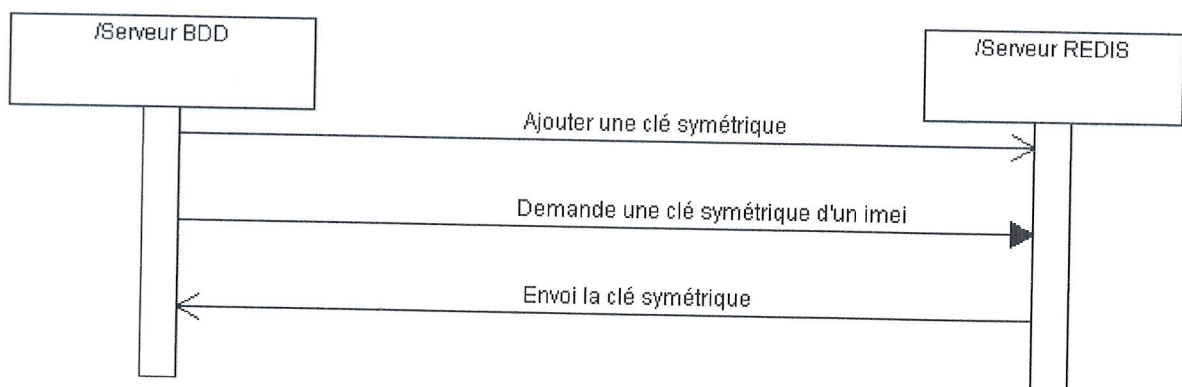


FIGURE 2.8 – Diagramme de séquences UML entre le Serveur base de données et le Serveur REDIS

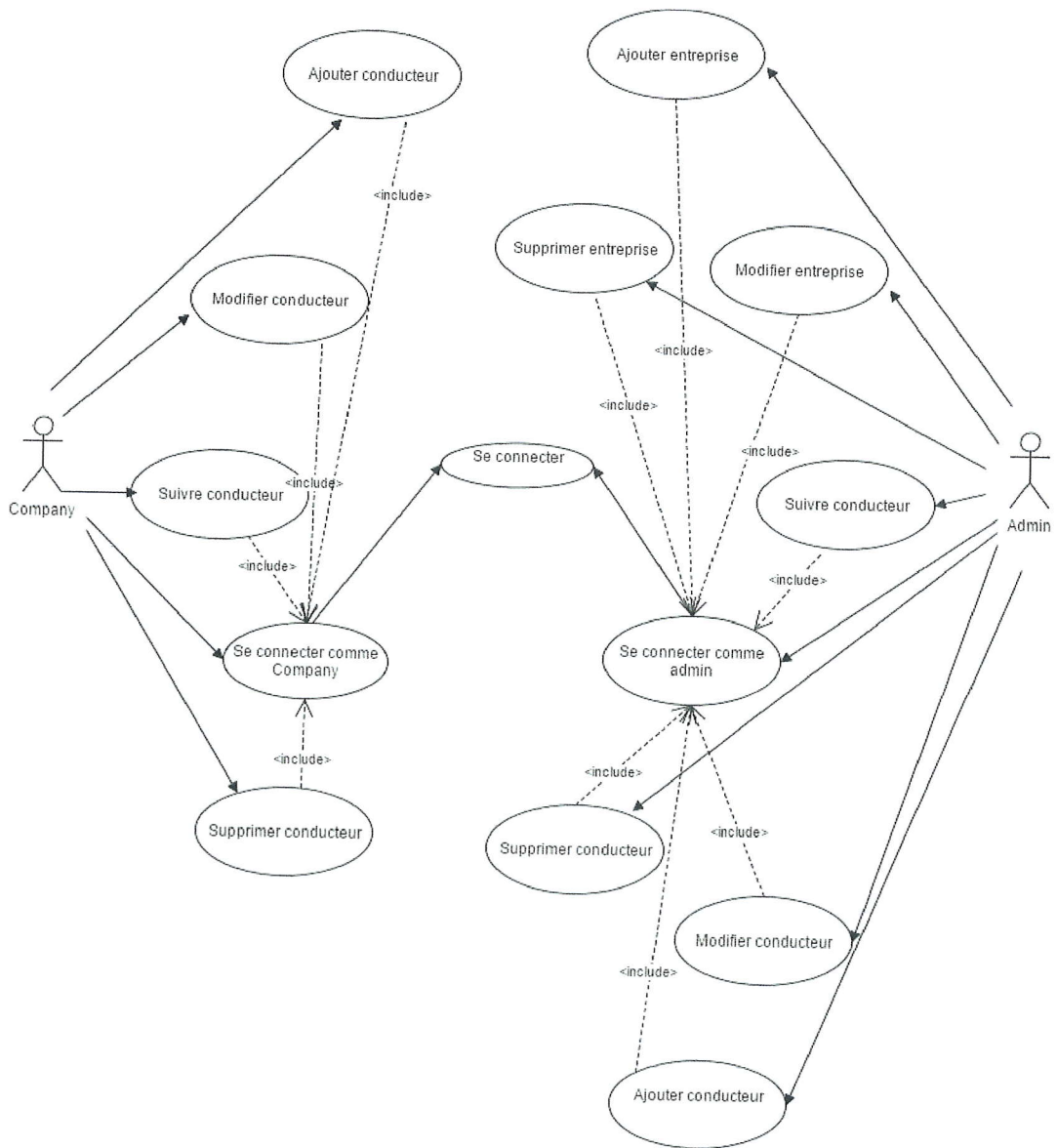


FIGURE 2.9 – Diagramme de cas d'utilisation UML entre les utilisateurs et le site web

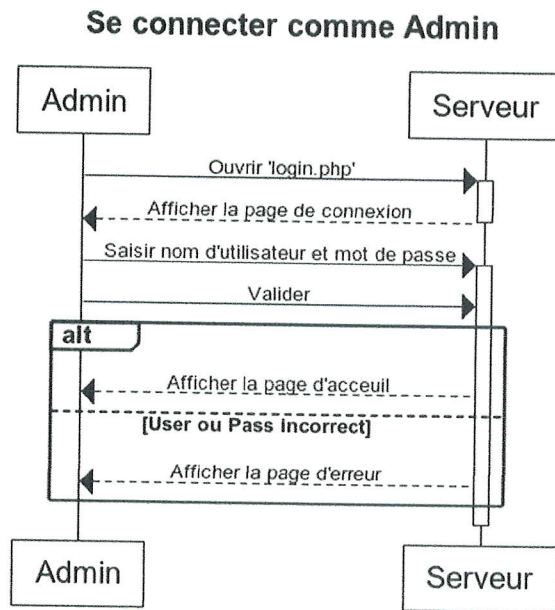


FIGURE 2.10 – Diagramme de Séquences UML pour se connecter comme Admin

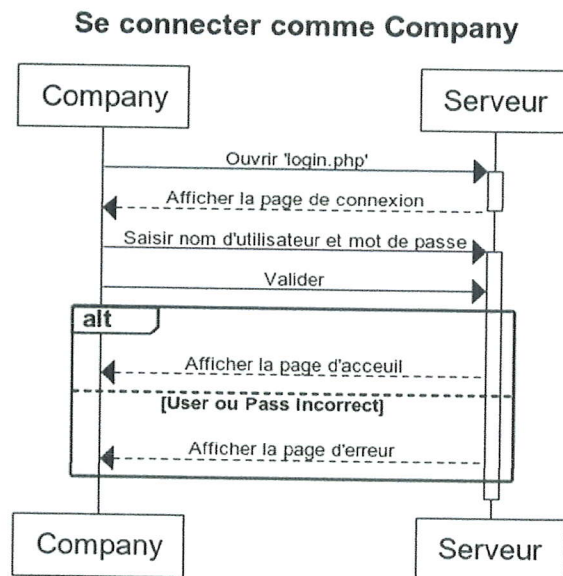


FIGURE 2.11 – Diagramme de Séquences UML pour se connecter comme Company

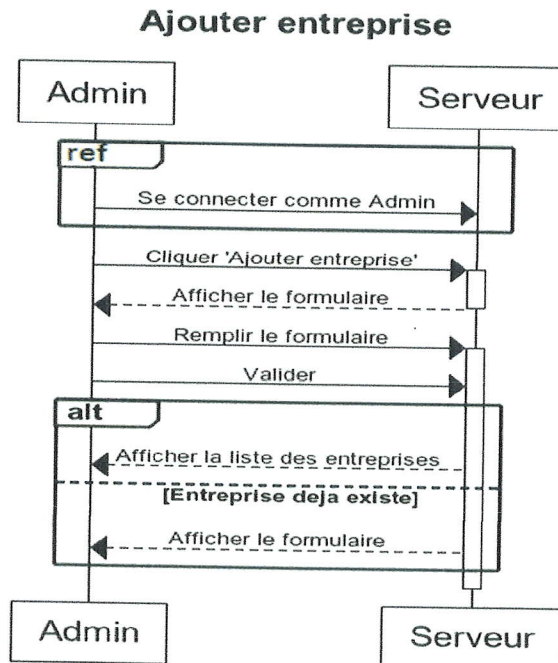


FIGURE 2.12 – Diagramme de Séquences UML pour ajouter une entreprise

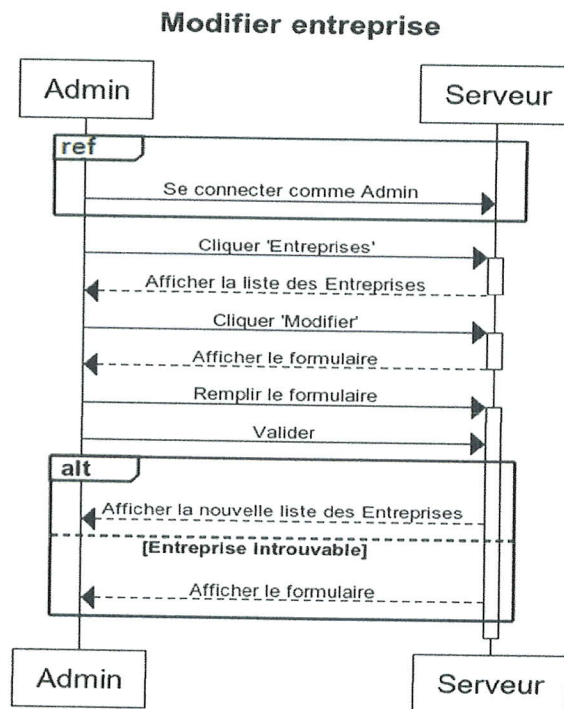


FIGURE 2.13 – Diagramme de Séquences UML pour modifier une entreprise

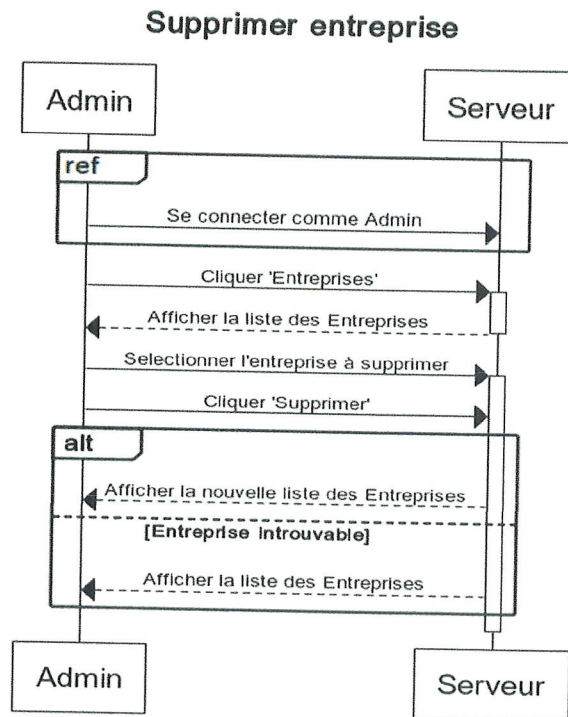


FIGURE 2.14 – Diagramme de Séquences UML pour supprimer une entreprise

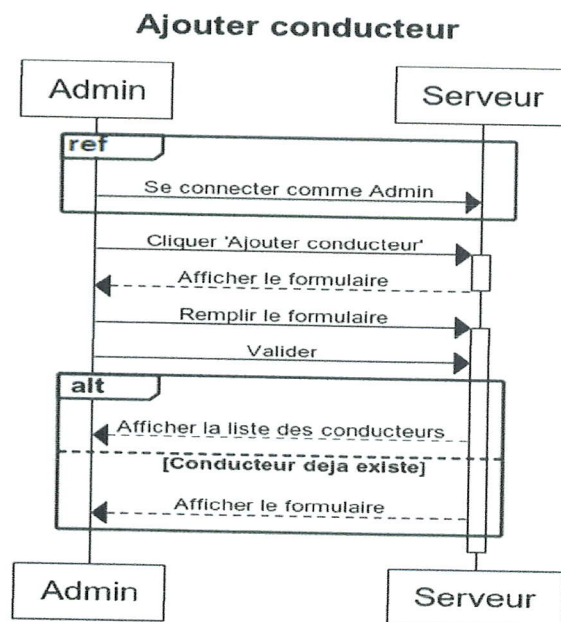


FIGURE 2.15 – Diagramme de Séquences UML pour ajouter un conducteur (Admin)

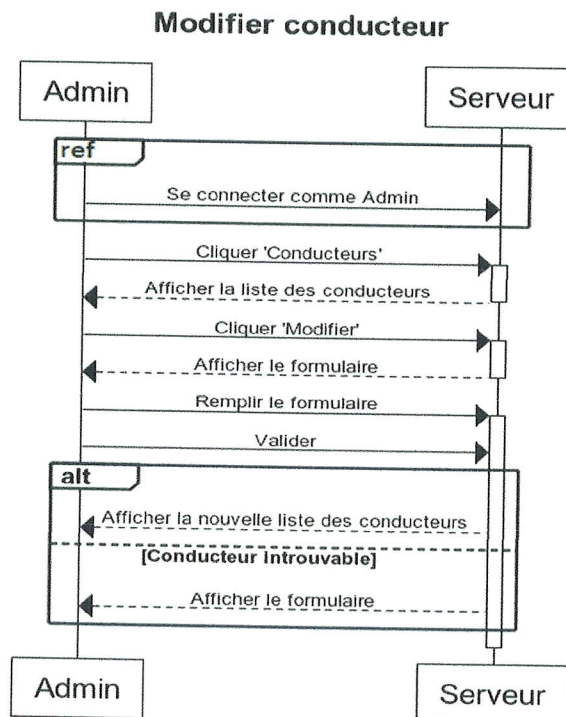


FIGURE 2.16 – Diagramme de Séquences UML pour modifier un conducteur (Admin)

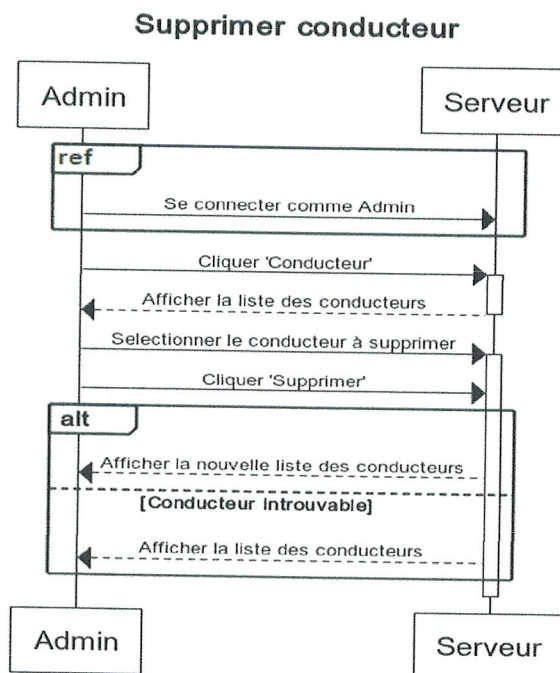


FIGURE 2.17 – Diagramme de Séquences UML pour supprimer un conducteur (Admin)

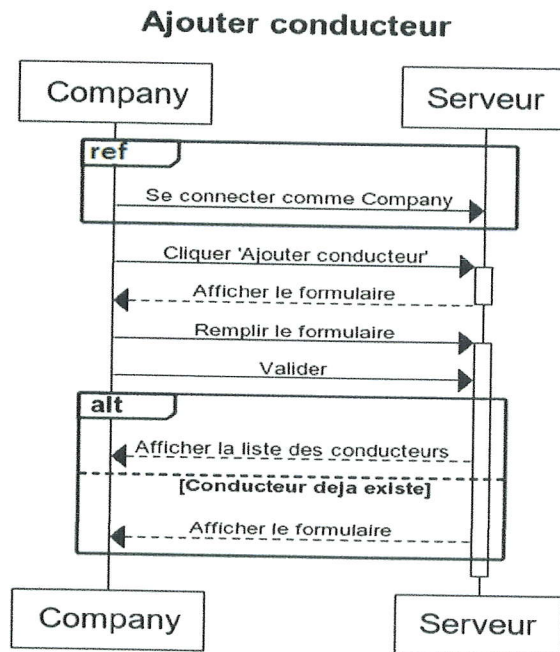


FIGURE 2.18 – Diagramme de Séquences UML pour ajouter un conducteur (Company)

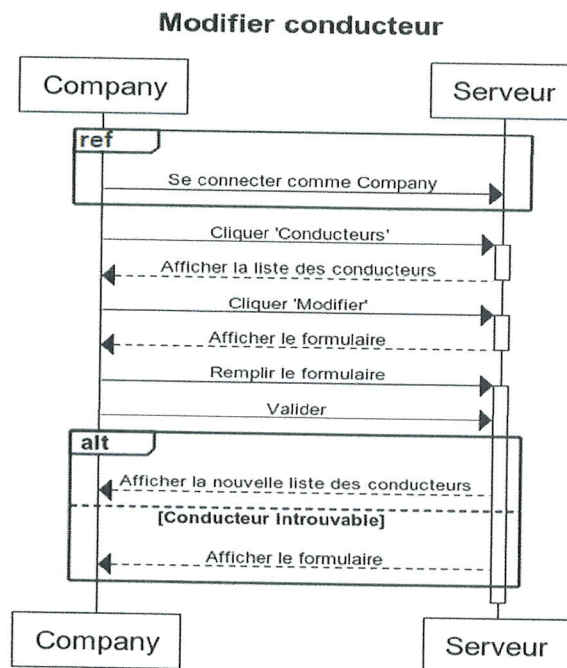


FIGURE 2.19 – Diagramme de Séquences UML pour modifier un conducteur (Company)

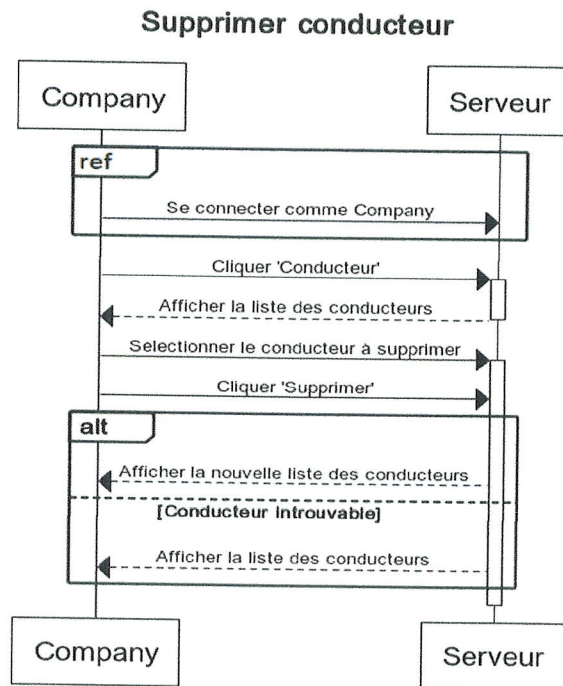


FIGURE 2.20 – Diagramme de Séquences UML pour supprimer un conducteur (Company)

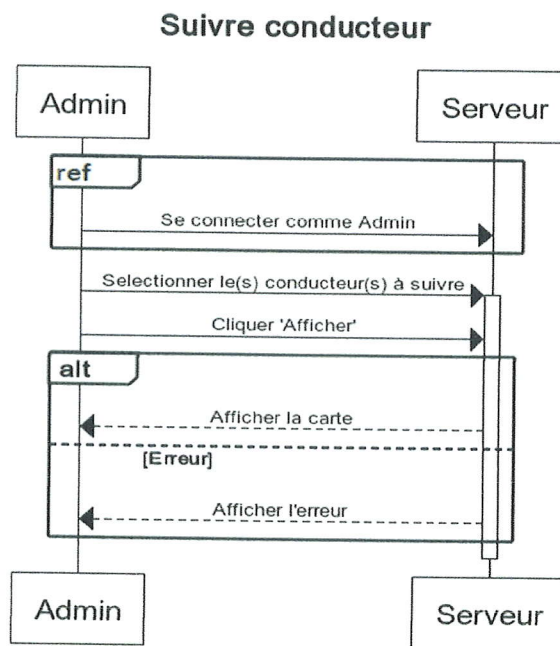


FIGURE 2.21 – Diagramme de Séquences UML pour le suivi des conducteurs (Admin)

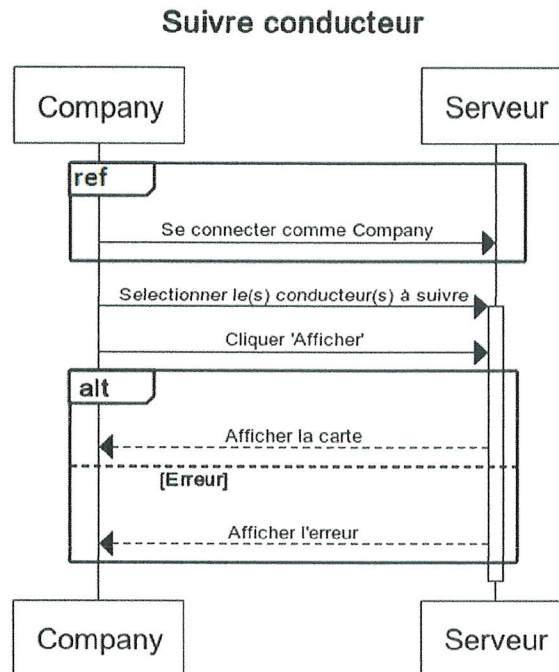


FIGURE 2.22 – Diagramme de Séquences UML pour le suivi des conducteurs (Company)

2.0.4 Relation Serveur Base de données / Serveur Entité Gouvernementale

Ici, le serveur qui gère la base de donnée est perçu comme un Client étant donnée la fonction que se dernier occupe dans cette relation. Lorsque le Serveur base de données s'apprête à terminer l'écriture d'un fichier dans la base de donnée (périodiquement à un rythme horaire) il envoie ce fichier au Serveur de l'entité gouvernementale afin que cette dernière puisse le signer. Pour cela, lorsque l'entité gouvernementale reçoit le fichier, elle calcule sa somme, le chiffre, et l'envoi au serveur base de donnée qui l'ajoute à la fin de son fichier pour le signer et permettre la vérification de son intégrité. (Voir figure 2.23 et 2.24)

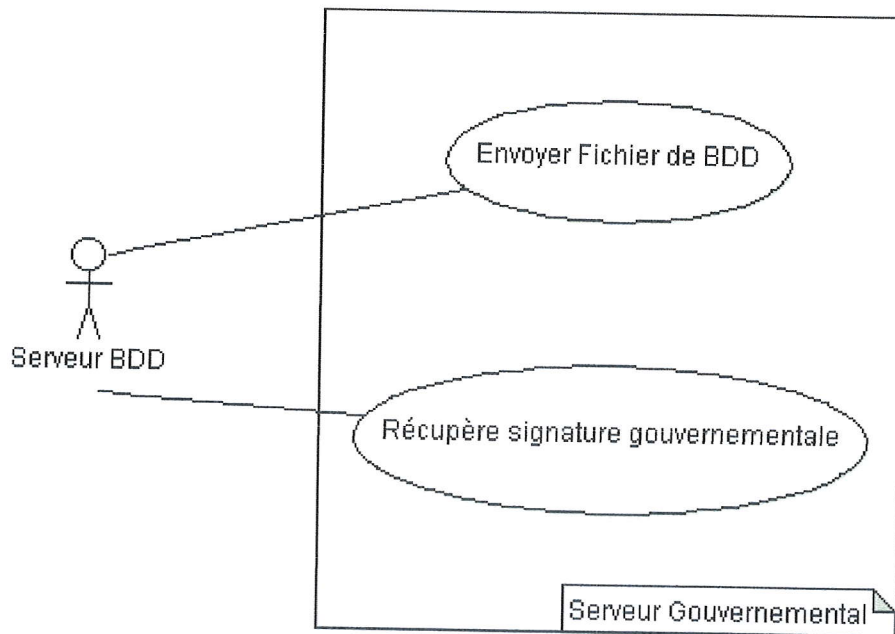


FIGURE 2.23 – Diagramme de cas d'utilisation UML entre le Serveur base de données et l'entité gouvernementale

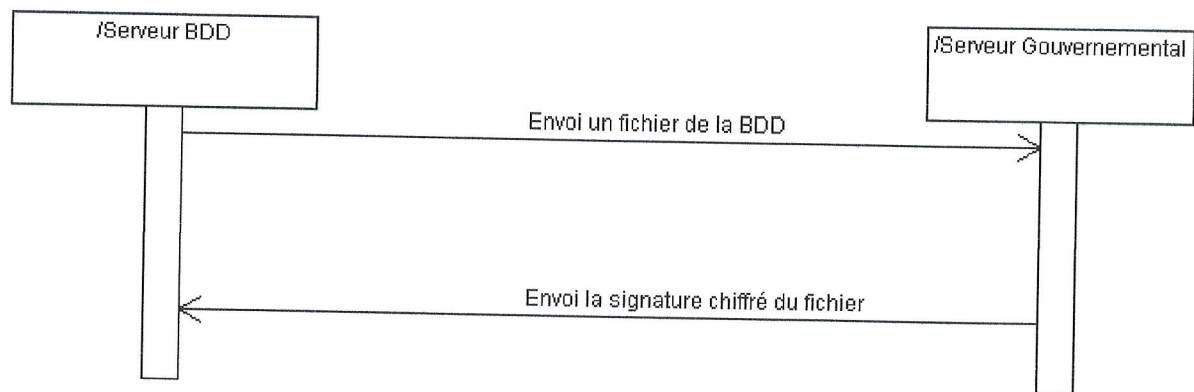


FIGURE 2.24 – Diagramme de séquences UML entre le Serveur base de données et l’entité gouvernementale

2.0.5 Relations des entités de création et gestion de la base de données

Cette interaction est le regroupement des relations qui constitue le système de création et de gestion de la base de données (Client Android, Serveur Base de données, Serveur REDIS, Serveur Gouvernementale). Elle représente le cycle de vie d’un fichier, de sa création jusqu’à sa signature juridique. (Voir figure 2.25)

3 Développement de la solution

L’utilisation de plusieurs langages de programmation et utilitaires (Environnement de travail) ont été nécessaires pour la réalisation de ce système, notamment à cause des différentes plateformes qu’utilisent les éléments du système.

3.1 Client Android

Pour remplacer les trackers utilisables dans les autres systèmes de géolocalisation classiques, nous avons utilisé un Smartphone comme client pour notre modèle, et pour le rendre opérationnel, nous avons créé une application Android qui permet de faire plusieurs tâches dont la géolocalisation et l’envoi des données au serveur.

Langage de programmation Pour la programmation, nous avons utilisé le logiciel Android Studio qui excelle dans ce domaine notamment pour sa puissance et sa facilité, mais aussi pour son support online grâce aux grandes communautés de programmeurs. C’est un logiciel de création d’application Android qui mélange Java et XML.

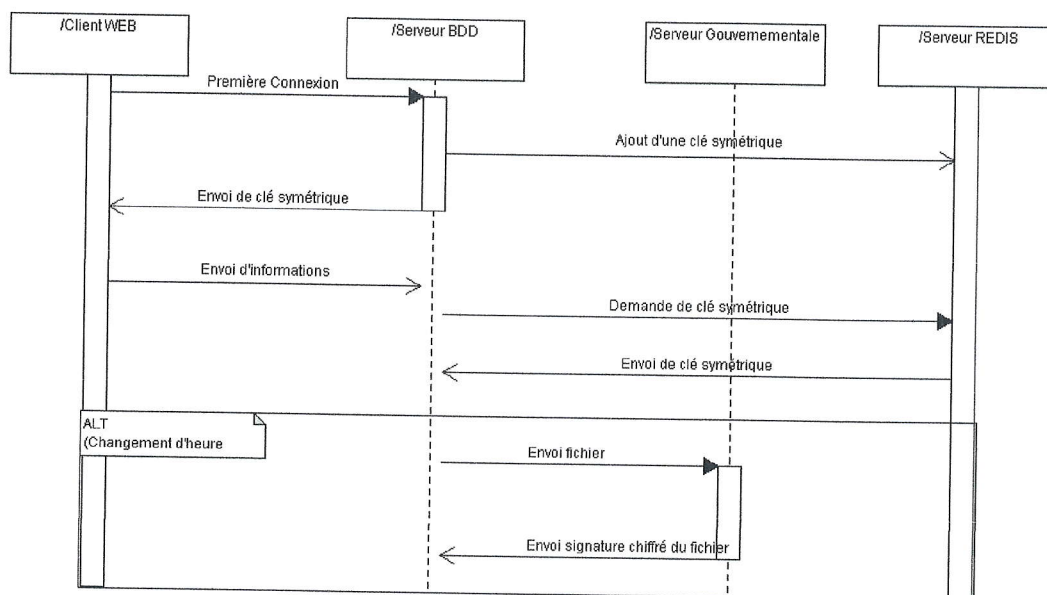


FIGURE 2.25 – Diagramme d’interaction UML entre les éléments de création et gestion de la base de données

Système de gestion de base de données Pour les bases de données, nous avons utilisées SQLite, qui est déjà inclut dans les smartphone Android et qui est léger et facilement utilisable.

3.1.1 Composition du Client

Notre application Android s’ouvre sur 2 onglets (fenêtres), un onglet principal, et un onglet pour les configurations. Pour le côté programmation, nous avons dû créer 9 classes sur Android studio : BackgroundOperation, BackgroundOperation2, DBHandler, GPS_Service, MainActivity, MCrypt, Mouvement, Settings.

3.1.1.1 MainActivity/Onglet Principal C’est la classe qui contient l’onglet principal, cet onglet contient les données envoyées au serveur, tel que l’IMEI du Smartphone, sa localisation, sa vitesse de déplacement, son niveau de batterie, ainsi qu’un bouton Configuration (Voir figure 2.26).

IMEI : Dans notre application, nous avons eue recours à une fonction prédéfinie qui permet de récupérer l’IMEI, et nous l’avons introduit dans une fonction nommé getDeviceImei(). (Voir le code 2.27)

Localisation : C’est la fonction principale de l’application, étant donnée que c’est elle qui récupère la localisation GPS. Pour avoir la localisation du Smartphone, on a créer une nouvelle classe (GPS_Service) qui permet de récupérer la localisation GPS grâce à la fonction prédéfinie LocationManager qui permet d’avoir la localisation actuelle après un changement de cette dernière. Ensuite, les localisations sont envoyés grâce à un Intent à l’onglet principale pour afficher la localisation. (Voir le code 2.28)



FIGURE 2.26 – Interface principale de l'application Android

```
private String getDeviceImei() {  
  
    mTelephonyManager = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);  
    String deviceid = mTelephonyManager.getDeviceId();  
    return deviceid;  
}
```

FIGURE 2.27 – Code de la fonction getDeviceImei

Vitesse : Pour récupérer la vitesse, nous avons calculé le rapport entre la distance

```

@Override
public void onCreate() {

    listener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            Intent i = new Intent("location_update");
            i.putExtra("coordinates", location.getLongitude()+":"+location.getLatitude());
            i.putExtra("speed", location.getSpeed());
            i.putExtra("precision", location.getAccuracy());
            sendBroadcast(i);
        }

        @Override
        public void onStatusChanged(String s, int i, Bundle bundle) {

        }

        @Override
        public void onProviderEnabled(String s) {

        }

        @Override
        public void onProviderDisabled(String s) {
            Intent i = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
            i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(i);
        }
    };

    locationManager = (LocationManager) getApplicationContext().getSystemService(Context.LOCATION_SERVICE);

    //noinspection MissingPermission
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10000, 0, listener);
}

```

FIGURE 2.28 – Code de la fonction qui récupère la localisation

parcourue entre deux changements de la localisation et le temps nécessaire à ce changement qui sera en m/h, puis divisé par 1000 pour l'obtenir en Km/h (Voir le code 2.29).

Batterie : Pour avoir le niveau de batterie, nous avons utilisé un Intent qui permet de récupérer le niveau de la batterie grâce à `Intent.ACTION_BATTERY_CHANGED`, qu'on a ensuite transformé en pourcentage. Nous avons nommé la fonction `batteryLevel()` et elle est disponible dans le `MainActivity`. (Voir le code 2.30)

3.1.1.2 Settings/Onglet Secondaire L'onglet de configuration (Voir figure 2.31) sert à fournir les données nécessaires pour le bon fonctionnement de l'application. Il dispose de 4 champs de textes :

- URL : Pour avoir l'URL du serveur.
- Société : pour le nom de la société du véhicule.
- Cryptage : la clé symétrique pour le cryptage des données vers le serveur
- Mot de passe : C'est un mot de passe connu par nous même pour éviter que les chauffeurs puissent changer ces informations.


```

coordinates.setText("" + intent.getExtras().get("coordinates"));
String coord = coordinates.getText().toString();
if (locationA.getLatitude() == 0 && locationA.getLatitude() == 0) {
    strDate = sdf.format(c.getTime());
    temps = strDate.split(" ")[1].split(":");
    tseconds = Integer.parseInt(temps[0]) * 3600 + Integer.parseInt(temps[1]) * 60 + Integer.parseInt(temps[2]);
    double tt = Double.parseDouble(coord.split(":")[1]);
    locationA.setLatitude(Double.parseDouble(coord.split(":")[1]));
    locationA.setLongitude(Double.parseDouble(coord.split(":")[0]));
    distance = 0;
    vitt = 0;
} else {
    strDate = sdf.format(c.getTime());
    temps = strDate.split(" ")[1].split(":");
    int tseconds1 = Integer.parseInt(temps[0]) * 3600 + Integer.parseInt(temps[1]) * 60 + Integer.parseInt(temps[2]);
    int dure = tseconds1 - tseconds;
    Location locationB = new Location("point B");
    locationB.setLatitude(Double.parseDouble(coord.split(":")[1]));
    locationB.setLongitude(Double.parseDouble(coord.split(":")[0]));
    distance = locationA.distanceTo(locationB);
    vitt = ((distance / dure) * 3600) / 1000;
    locationA.setLatitude(locationB.getLatitude());
    locationA.setLongitude(locationB.getLongitude());
    tseconds = tseconds1;
}
speeds.setText("" + vitt);
Local = coordinates.getText().toString();
Vite = speeds.getText().toString();

```

FIGURE 2.29 – Code de la fonction qui calcule la vitesse

```

private void batteryLevel() {
    BroadcastReceiver batteryLevelReceiver = (context, intent) -> {
        context.unregisterReceiver(this);
        int rawlevel = intent.getIntExtra("level", -1);
        int scale = intent.getIntExtra("scale", -1);
        int level = -1;
        if (rawlevel >= 0 && scale > 0) {
            level = (rawlevel * 100) / scale;
        }
        batt.setText("" + level + "%");
    };
    IntentFilter batteryLevelFilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
    registerReceiver(batteryLevelReceiver, batteryLevelFilter);
}

```

FIGURE 2.30 – Code de la fonction batteryLevel



FIGURE 2.31 – L'interface secondaire de l'application Android

Le bouton "Revenir" sert, éventuellement, à revenir à l'onglet principal, tandis que le bouton "Envoi", permet l'envoi au serveur central grâce à l'URL, l'imei du smartphone pour que le serveur génère une clé symétrique et l'envoi au client qui la récupère, et la met dans la case Cryptages. Ces champs de textes sont reliés à des variables globales qui définissent des éléments statiques de l'opération d'envoi au fil du temps (C'est les seules variables qui ne changent pas pendant l'activité de l'application).

3.1.2 Fonctionnement du Client

En premier lieu, il faut remplir les champs présents dans l'onglet de configuration (URL, Société, Mot de passe). Lors du clic sur le bouton Envoi, le client Android génère une requête POST au serveur qui contient 2 variables : Type et IMEI (Voir le code 2.32). Nous avons mis quelques conditions pour que le bouton Envoi s'exécute sans erreurs (Champs URL/Société/Mot De Passe Obligatoire).

Pour faire une requête POST sous Android Studio, nous avons créé une classe "BackgroundOperation2()" qui permet cela (voir le code 2.33). Les informations sont d'abord encodées en UTF-8⁴, puis en utilisant le "HttpURLConnection" d'Android

4. UTF-8 est un codage de caractères informatiques conçu pour coder l'ensemble des caractères du « répertoire universel de caractères codés », initialement développé par l'ISO dans la norme internationale ISO/CEI 10646, aujourd'hui totalement compatible avec le standard Unicode, en

```

save.setOnClickListener(B) > {
    try {
        try {
            m = MessageDigest.getInstance("MD5");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        m.update(password.getText().toString().getBytes(), 0, password.getText().toString().length());
        mdp = new BigInteger(1, m.digest()).toString(16);
        if (mdp.equals("25f9e794323b453885f5181f1b624d0b")) {
            if (MainActivity.url1 == null) {
                if (urlip.getText().toString().equals("")) {
                    erreur.setTextColor(Color.RED);
                    erreur.setText("Erreur, veuillez mettre une URL");
                }
            } else {
                {
                    if (MainActivity.Societe == null)
                    {
                        if (societe.getText().toString().equals(""))
                        {
                            erreur.setTextColor(Color.RED);
                            erreur.setText("Erreur, veuillez mettre le nom de la société");
                        }
                    } else
                    {
                        String passw = new BackgroundOperation2().execute().get();
                        MainActivity.password = passw.substring(0, passw.length() - 1);
                        MainActivity.Societe = societe.getText().toString();
                        MainActivity.url1 = urlip.getText().toString();
                        crypto.setText(MainActivity.password);
                        erreur.setTextColor(Color.GREEN);
                        erreur.setText("Informations modifiées avec succès");
                    }
                }
            }
        }
    }
}

```

FIGURE 2.32 – Code du bouton Envoi

```

public String postCall() throws UnsupportedEncodingException {
    URL url;
    String text;
    BufferedReader reader = null;
    String MEI = MainActivity.imei.getText().toString();
    String type = "pass";
    String data;
    data = URLEncoder.encode("imei", "UTF-8")
        + "=" + URLEncoder.encode(MEI, "UTF-8");
    data += "&" + URLEncoder.encode("request", "UTF-8") + "="
        + URLEncoder.encode(type, "UTF-8");
    try {
        url = new URL(MainActivity.url);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setDoInput(true);
        conn.setDoOutput(true);
        OutputStream os = conn.getOutputStream();
        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, "UTF-8"));
        writer.write(data);
        writer.flush();
        writer.close();
        os.close();
        reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        StringBuilder sb = new StringBuilder();
        String line = null;

        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        text = sb.toString();
    } catch (Exception e) {
        return "False";
    }
    return text;
}

```

FIGURE 2.33 – Code de la fonction BackgroundOperation2

studio, on créer une connexion asynchrone entre le client et le serveur, puis les données seront envoyées grâce au Buffers. Le client par la suite, récupère la clé de cryptographie en cas de réussite, ou "False" en cas de faillite (Serveur qui ne répond pas). Après la réception de la clé, le client pourra envoyer les données crypté au serveur sous 2 modes :

3.1.2.1 Mode Offline C'est un mode qui fonctionne lorsqu'il n'y pas d'accès internet (Hors 3G). Pour permettre celà , nous avons établit une fonction `isNetworkAvailable()` qui détecte la présence d'internet. (voir le code 2.34)

Lorsque la fonction retourne False, on utilisera le mode online. Le mode Offline

```
private boolean isNetworkAvailable() {
    ConnectivityManager connectivityManager
        = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
    return activeNetworkInfo != null && activeNetworkInfo.isConnected();
}
```

FIGURE 2.34 – Code de la fonction `isNetworkAvailable`

récupère la localisation GPS, La vitesse, la batterie, la date actuelle, le nom de la société, l'imei , les chiffres grâce à la clé symétrique, et les stocks dans SQLite intégré d'Android avec la fonction "AjouterLocale" que nous avons créer nous même, et il répétera cette action jusqu'à ce que la fonction retourne True. (Voir les codes 2.35 et 2.36)

3.1.2.2 Mode Online Le monde online a 2 fonctionnements, le premier est la vérification de la base de donnée SQLite, si elle n'est pas vide, le client envoie au serveur tous le contenu de la base SQLite (Le contenu sont les donnée stockée par l'application lorsqu'elle était en mode offline). Tout comme dans l'onglet Settings, on utilise la fonction "BackgroundOperation" qui diffère de "BackgroundOperation2" dans les informations envoyées. Toutes les informations de la Base de données sont récupérées sous forme d'objets "Mouvement" qui regroupent les informations comme attributs. (voir les codes 2.37 et 2.38) Puis, il récupère les données actuelles, et les envoie directement sans passer par la base de données, et il répétera cette opération jusqu'à ce que la fonction retourne "False". `MCrypte` est une classe que nous avons créer et qui permet le chiffrement des informations grâce à une clé symétrique, elle utilise un chiffrement AES 128bits CBC. (voir le code 2.40)

3.2 Serveur Central

Etant donnée l'architecture Client/Serveur de notre modèle, notre application Android aura besoin d'un serveur où stocker ses données, pour celà il est nécessaire de créer un serveur qui permet non seulement de communiquer avec le client Android, mais aussi avec le serveur WEB.

restant compatible avec la norme ASCII limitée à l'anglais de base, mais très largement répandue depuis des décennies.

```

} else {
    coordinates.setText("" + intent.getExtras().get("coordinates"));
    String coord = coordinates.getText().toString();
    if (locationA.getLatitude() == 0 && locationA.getLatitude() == 0) {
        strDate = sdf.format(c.getTime());
        temps = strDate.split(" ")[1].split(":");
        tseconds = Integer.parseInt(temps[0]) * 3600 + Integer.parseInt(temps[1]) * 60 + Integer.parseInt(temps[2]);
        double tt = Double.parseDouble(coord.split(":")[1]);
        locationA.setLatitude(Double.parseDouble(coord.split(":")[1]));
        locationA.setLongitude(Double.parseDouble(coord.split(":")[0]));
        distance = 0;
        vitt = 0;
    } else {
        strDate = sdf.format(c.getTime());
        temps = strDate.split(" ")[1].split(":");
        int tseconds1 = Integer.parseInt(temps[0]) * 3600 + Integer.parseInt(temps[1]) * 60 + Integer.parseInt(temps[2]);
        int dure = tseconds1 - tseconds;
        Location locationB = new Location("point B");
        locationB.setLatitude(Double.parseDouble(coord.split(":")[1]));
        locationB.setLongitude(Double.parseDouble(coord.split(":")[0]));
        distance = locationA.distanceTo(locationB);
        vitt = ((distance / dure) * 3600) / 3600;
        locationA.setLatitude(locationB.getLatitude());
        locationA.setLongitude(locationB.getLongitude());
        tseconds1 = tseconds;
    }
    speeds.setText("" + vitt);
    Local = coordinates.getText().toString();
    Vite = speeds.getText().toString();
    imeii = getDeviceImei();
    batteryLevel();
    battery = batt.getText().toString();
    db.AjouterLocal(imeii, MainActivity.Societe, Local, Vite, strDate, battery);
}

```

FIGURE 2.35 – Fragment de code qui permet le fonctionnement Offline

```

public DBHandler(Context context) { super(context, DATABASE_NAME, null, DATABASE_VERSION); }
@Override
public void onCreate(SQLiteDatabase db) {
    String Req = "CREATE TABLE " + TABLE_GPS + " (" + KEY_IMEI + " TEXT, " + KEY_Societe + " TEXT,
    db.execSQL(Req);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_GPS);
    onCreate(db);
}
public void AjouterLocal(String imei, String Societe, String Local, String Vitesse, String Date, St
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(KEY_IMEI, imei );
    values.put(KEY_Societe, Societe );
    values.put(KEY_Local, Local );
    values.put(KEY_Vitesse, Vitesse );
    values.put(KEY_DATE, Date );
    values.put(KEY_Battery, Battery);
    db.insert(TABLE_GPS, null, values);
    db.close();
}

public boolean Vide()
{
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor mCursor = db.rawQuery("SELECT * FROM " + TABLE_GPS, null);
    if (mCursor.moveToFirst()) return false;
    else return true;
}

```

FIGURE 2.36 – Fragment de code qui de la classe DBHandler et la fonction AjouterLocale

```

else {
    List<Mouvement> mvm = MainActivity.db.Lire();
    for (Mouvement mouvement : mvm) {
        String LOC = mouvement.getLocal();
        String Vit = mouvement.getVitesse();
        String MEI = mouvement.getImei();
        String strDate = mouvement.getDate();
        String soc = mouvement.getSociete();
        String battr = mouvement.getBattery();
        String type = "localisation";
        String etat = "Offline";
        MEncrypt mencrypt = new MEncrypt();
        String cle = MainActivity.password;

```

FIGURE 2.37 – Fragment de code de la classe BackgroundOperation en utilisant une base de données


```

public class Mouvement {
    private String imei;
    private String Societe;
    private String Local;
    private String Vitesse;
    private String Date;
    private String Batt;

    public Mouvement(String imei, String Societe, String Local, String Vitesse, String Date, String Batt) {
        this.imei = imei;
        this.Societe = Societe;
        this.Local = Local;
        this.Vitesse = Vitesse;
        this.Date = Date;
        this.Batt = Batt;
    }

    public String getImei() { return this.imei; }
    public String getSociete() { return this.Societe; }
    public String getLocal() { return this.Local; }
    public String getDate() { return this.Date; }
    public String getVitesse() { return this.Vitesse; }
    public String getBattery() {return this.Batt;}
}

```

FIGURE 2.38 – Code de la Déclaration de la classe Mouvement

```

URL url;
String text = "";
BufferedReader reader = null;
if (MainActivity.db.Vide()) {
    String LOC = MainActivity.Local;
    String Vit = MainActivity.Vite;
    String MEI = MainActivity.imei;
    String soc = MainActivity.Societe;
    String strDate = MainActivity.strDate;
    String cle = MainActivity.password;
    String battr = MainActivity.battery;
    String type = "localisation";
    String etat = "Online";
    MCrypt mcrypt = new MCrypt();
    try {
        LOC = MCrypt.bytesToHex( mcrypt.encrypt(LOC, cle) );
        Vit = MCrypt.bytesToHex( mcrypt.encrypt(Vit, cle) );
        soc = MCrypt.bytesToHex( mcrypt.encrypt(soc, cle) );
        strDate = MCrypt.bytesToHex( mcrypt.encrypt(strDate, cle) );
        battr = MCrypt.bytesToHex( mcrypt.encrypt(battr, cle) );
        etat = MCrypt.bytesToHex( mcrypt.encrypt(etat, cle) );
    }
}

```

FIGURE 2.39 – Fragment de code de la classe BackgroundOperation


```

public MCrypt()
{
    ivspec = new IvParameterSpec(iv.getBytes());

    //keyspec = new SecretKeySpec(SecretKey.getBytes(), "AES");

    try {
        cipher = Cipher.getInstance("AES/CBC/NoPadding");
    } catch (NoSuchAlgorithmException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (NoSuchPaddingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public byte[] encrypt(String text, String key) throws Exception
{
    keyspec = new SecretKeySpec(key.getBytes(), "AES");
    if(text == null || text.length() == 0)
        throw new Exception("Empty string");

    byte[] encrypted = null;

    try {
        cipher.init(Cipher.ENCRYPT_MODE, keyspec, ivspec);

        encrypted = cipher.doFinal(padString(text).getBytes());
    } catch (Exception e)
    {
        throw new Exception("[encrypt] " + e.getMessage());
    }

    return encrypted;
}

```

FIGURE 2.40 – Code de la classe MCrypte

```

if (!db.Vide()) {
    new BackgroundOperation().execute();
}
coordinates.setText("" + intent.getExtras().get("coordinates"));
String coord = coordinates.getText().toString();
if (locationA.getLatitude() == 0 && locationA.getLatitude() == 0) {
    strDate = sdf.format(c.getTime());
    temps = strDate.split(" ")[1].split(":");
    tseconds = Integer.parseInt(temps[0]) * 3600 + Integer.parseInt(temps[1]) * 60 + Integer.parseInt(temps[2]);
    double vt = Double.parseDouble(coord.split(":")[1]);
    locationA.setLatitude(Double.parseDouble(coord.split(":")[1]));
    locationA.setLongitude(Double.parseDouble(coord.split(":")[0]));
    distance = 0;
    vitt = 0;
} else {
    strDate = sdf.format(c.getTime());
    temps = strDate.split(" ")[1].split(":");
    int tseconds1 = Integer.parseInt(temps[0]) * 3600 + Integer.parseInt(temps[1]) * 60 + Integer.parseInt(temps[2]);
    int dure = tseconds1 - tseconds;
    Location locationB = new Location("point B");
    locationB.setLatitude(Double.parseDouble(coord.split(":")[1]));
    locationB.setLongitude(Double.parseDouble(coord.split(":")[0]));
    distance = locationA.distanceTo(locationB);
    vitt = ((distance / dure) * 3600) / 1000;
    locationA.setLatitude(locationB.getLatitude());
    locationA.setLongitude(locationB.getLongitude());
    tseconds = tseconds1;
}
speeds.setText("" + vitt);
Local = coordinates.getText().toString();
Vite = speeds.getText().toString();
imei = getDeviceImei();
batteryLevel();
battery = batt.getText().toString();
new BackgroundOperation().execute();

```

FIGURE 2.41 – Fragment de code qui permet le fonctionnement Online

Langage de programmation Pour la programmation, Nous avons eu recours au langage de programmation PHP qui est une excellente option pour de nombreuses raisons :

- Temps de chargement rapide.
- Logiciel moins coûteux : en travaillant avec PHP, la plupart des outils associés au programme sont des logiciels libres.
- Hébergement moins coûteux.
- Flexibilité de la base de données : PHP est flexible pour la connectivité de la base de données. Il peut se connecter à plusieurs bases de données, la plus utilisée est MySQL.

Nous avons aussi utilisé le langage PYTHON, qui est un langage de scripts de haut niveau exécutable par PHP, et pour accéder à la base REDIS comme dans notre cas.

Système de gestion de base de données EasyPHP est une plateforme de développement Web, permettant de faire fonctionner localement un serveur web (sans se connecter à un serveur externe). C'est un environnement comprenant deux serveurs (un serveur web Apache et un serveur de bases de données MySQL). Il permet donc d'installer en une seule fois tout le nécessaire au développement local du Web. Pour permettre l'utilisation d'EasyPHP online, nous avons utilisé par un logiciel nommé Ngrok qui permet cela.

Ngrok c'est un petit logiciel qui permet de créer un tunnel entre le Web et le serveur local si il est derrière un pare-feu. Étant donné la difficulté de paramétrer EasyPHP pour un accès depuis le Web, ngrok lui permet cela en quelques seconds, il génère un lien URL qui pourra être utilisé pour accéder au serveur local sans toucher au pare-feu.

3.2.1 Serveur Base De Données

index.php C'est le script qui permet la récupération des données de l'application Android grâce à l'opération POST du protocole HTTP, et les enregistre dans une base de donnée hiérarchiques (voir le code 2.42). En premier lieu, nous générons une clé symétrique pour le client afin de protégé les informations, pour cela nous récupérons le premier poste du Client, et grâce à une fonction qui génère une chaîne de caractères de taille aléatoire, nous générons une clé qui sera renvoyée au Client qui l'utilisera pour ses prochaines requêtes. Après cela, le Client commencera à envoyer des informations en continu, tout d'abord nous récupérons l'IMEI du Smartphone (c'est l'unique donnée qui n'est pas cryptée), puis, grâce à un script en Python qui permet d'interroger la base de donnée redis (avec la commande `shell_exec`), nous récupérerons la clé symétrique de ce smartphone qui nous servira à déchiffrer les informations grâce à la fonction `MCrypte` (même fonctionnement que sur Android, voir le code 2.43). La date courante est ensuite décomposée en 4 parties, l'année, le mois, le jour et l'heure. Nous créerons le dossier database, si il est déjà créer, nous l'ouvrons et nous créons un dossier avec le nom de la société et nous l'accédons, puis nous créerons un autre dossier avec comme nom l'IMEI du smartphone et nous accédons aussi, puis nous créons une succession de dossiers hiérarchiques basés sur la date récupérée tel que Année/Mois/Jour. Dans le dernier dossier qui est Jour voir

```

<?php
    include("MCrypt.php");
    include 'fonction.php';
    error_reporting(0);
    $mccrypt = new MCrypt();
    date_default_timezone_set('Europe/London');
    $request = urldecode($_POST['request']);
    $imei = urldecode($_POST['imei']);
    if(strcmp($request,"pass") == 0 )
    {
        $pass = createpassword(16);
        $cc = escapeshellarg($imei);
        $tt = escapeshellarg($pass);
        $reponse = shell_exec("E:\Python27\python.exe ajout.py $cc $tt");
        echo $pass;
    }
    else
    {
        if(strcmp($request,"connect") == 0 )
        {
            echo "true";
        }
        else
        {
            $key = shell_exec("E:\Python27\python.exe recherche.py $imei");
            $key = trim($key, "\n");
            $soc = $mccrypt->decrypt(urldecode($_POST['societe']),$key);
            $loc = $mccrypt->decrypt(urldecode($_POST['localisation']),$key);
            $vit = $mccrypt->decrypt(urldecode($_POST['vitesse']),$key);
            $date = $mccrypt->decrypt(urldecode($_POST['date']),$key);
            $battery = $mccrypt->decrypt(urldecode($_POST['battery']),$key);
            $etat = $mccrypt->decrypt(urldecode($_POST['etat']),$key);
            list($jj,$hh) = explode(' ', $date);
        }
    }
}

```

FIGURE 2.42 – Code qui permet la récupération des POSTs des clients


```

<?php
class MCrypt
{
    private $iv = 'dkqgendor149aalis5';
    function __construct()
    {
    }
    function encrypt($str,$key) {
        $iv = $this->iv;
        $std = mcrypt_module_open('rijndael-128', '', 'cbc', $iv);
        mcrypt_generic_init($std, $key, $iv);
        $encrypted = mcrypt_generic($std, $str);
        mcrypt_generic_deinit($std);
        mcrypt_module_close($std);
        return bin2hex($encrypted);
    }
    function decrypt($code,$key) {
        $code = $this->hex2bin($code);
        $iv = $this->iv;
        $std = mcrypt_module_open('rijndael-128', '', 'cbc', $iv);
        mcrypt_generic_init($std, $key, $iv);
        $decrypted = mdecrypt_generic($std, $code);
        mcrypt_generic_deinit($std);
        mcrypt_module_close($std);
        return utf8_encode(trim($decrypted));
    }
    protected function hex2bin($hexdata) {
        $bindata = '';
        for ($i = 0; $i < strlen($hexdata); $i += 2) {
            $bindata .= chr(hexdec(substr($hexdata, $i, 2)));
        }
        return $bindata;
    }
}
?>

```

FIGURE 2.43 – Code de MCrypte

figure 2.44, nous créerons un fichier qui a pour nom l'heure transmise, puis ce fichier aura pour contenu les autres données reçu qu'on appellera ensuite "Document". La syntaxe des fichiers sera tel que présentée dans la figure 2.45. Avec comme Header le nom de la société et l'imei, en contenu les informations reçu, et en Footer c'est le hash MD5 crypté du fichier, calculer par une entité gouvernemental abordée dans le chapitre suivant.

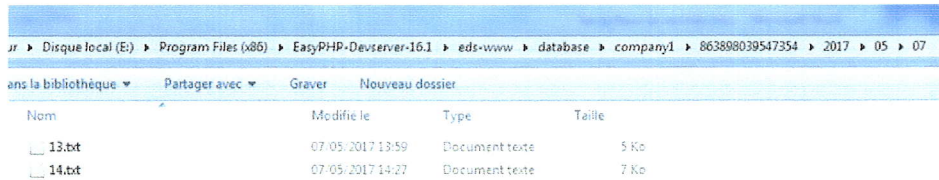


FIGURE 2.44 – Un des dossiers qui contiennent les documents de la base de données.

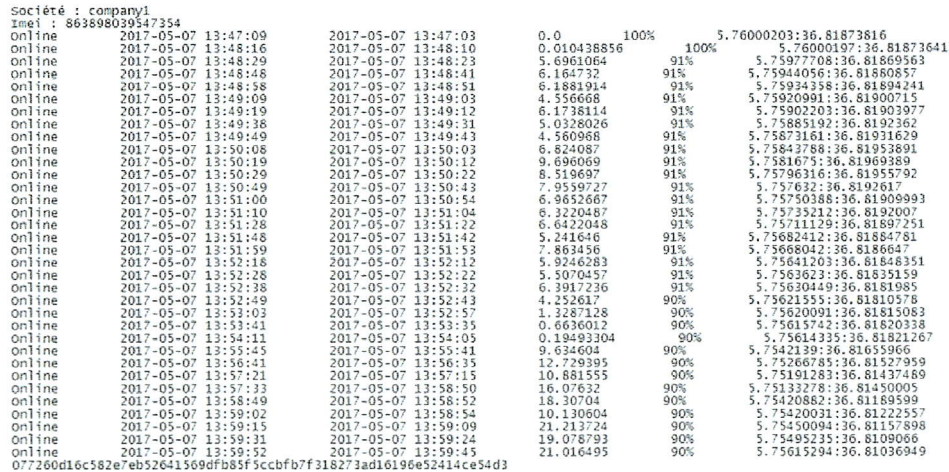


FIGURE 2.45 – Un document de la base de données.

server.php Cette page s'occupe du serveur Web qui permet au gérants des entreprises de suivre leurs véhicules. Elle a deux utilités, la première est la génération d'un token pour identifier les clients (les clients Web), la deuxième est l'envoi des données après un intervalle de temps pour les clients afin de localiser leurs véhicules. Tout d'abord, le client envoi une requête suivie d'un nombre déterminé de paramètres suivant la requête vers le serveur en Json. Dans notre modèle il existe 3 requêtes :

"Request" Elle permet de demander au serveur de créer un token et l'envoyer au client, pour celà nous avons utilisé une bibliothèque sur le net : JWT. Le serveur initialisera un décodage Json, puis créera le token grâce au nom de la compagnie, la date actuelle, ainsi qu'une clé symétrique. Le couple compagnie/token sera stocké dans la base de donnée Redis pour permettre l'identification des clients.

"localisations" cette requête permet d'envoyer les coordonnées ainsi que d'autres informations aux clients suivant un intervalle de temps. Initialement, il recherche le token (présent dans les paramètres de la requête) dans la base de donnée Redis,

```

require_once 'jwt/src/BeforeValidException.php';
require_once 'jwt/src/ExpiredException.php';
require_once 'jwt/src/SignatureInvalidException.php';
require_once 'jwt/src/JWT.php';
use \Firebase\JWT\JWT;

```

FIGURE 2.46 – Code des déclarations de la bibliothèque JWT

```

$data = file_get_contents('php://input');
$data = json_decode($data);
$type = $data->type;
switch($type)
{
    case "request" :
        $company = $data->company;
        $date = $data->date;
        $key = createpassword(24);
        $token = array(
            "company" => $company,
            "date" => $date
        );
        $jwt = JWT::encode($token, $key);
        $cc = escapeshellarg($company);
        $tt = escapeshellarg($jwt);
        $response = shell_exec("E:\Python27\python.exe ajout.py $cc $tt");
        echo $jwt;
        break;
}

```

FIGURE 2.47 – Code qu'utilise la requête "Request".

s'il ne le trouve pas, il envoie une réponse "denied" au client pour lui faire part d'un rejet de token (Validité ou non trouvable), si le token est présent dans la base de donnée Redis, le serveur soustraira les 2 dates pour obtenir le nombre d'heures (de documents voulus), puis il accédera à chaque document présent dans l'intervalle (en incrémentant l'heure à chaque fois) il stocke ses données, puis, lorsqu'il aura fini tous les fichiers, code le tout en Json et l'envoi aux clients. Pour la localisation en temps réel, le client enverra la date actuelle, et dès qu'il y'a un changement d'heure, il enverra les dates précédentes et l'actuelle pour garder une trace continue sur une carte du monde. (voir le code 2.48)

La dernière requête c'est la requête "App" abordée dans le chapitre 4.

3.2.2 Serveur REDIS

L'accès au serveur REDIS est moins complexe aux niveau du code. Il contient uniquement 2 codes en Python :

ajout.py Ce code permet l'ajout d'un couple clé/valeur sur le serveur REDIS, pour l'utiliser il suffit de lancer une requête "exec_shell" sur PHP avec 2 variables en paramètre (voir le code 2.49). Le code en Python quant à lui permet l'ajout d'un couple sur le serveur REDIS ainsi que la sauvegarde la base de données Redis actuelle (en utilisant la fonction BGSAVE()).


```

$timei = $data->imei;
$schemin = "database/$company/$timei";
$company = $data->company;
$timei = $data->imei;
$ddeb = $data->deb;
$fin = $data->fin;
$haar = nbrh($ddeb, $fin);
$stemp = $ddeb;
for($i=0;$i<=$haar;$i++)
{
    $dat = explode(" ", $stemp);
    $dat2 = explode("-", $dat[0]);

    $schemin = "database/$company/$timei/$dat2[0]/$dat2[1]/$dat2[2]/$dat[1].txt";
    $current = file_get_contents($schemin);
    $items = explode("\n", $current);
    $nbr = count($items);
    for($x=2;$x<$nbr-2;$x++)
    {
        $coor = explode(" ", $items[$x]);
        $ss = count($coor) - 1;
        $cc = explode(":", $coor[$ss]);
        $coordonnee['items'][] = array(
            'lat' => trim($cc[1], "\r"),
            'lng' => $cc[0]
        );
    }
    $stemp = gdate($stemp);
}
$coors = array($coordonnee);
$response = (object) array('response'=>'granted', 'result'=>array('coors'=>$coors));
$json = json_encode($response);
echo $json;
}
break;

```

FIGURE 2.48 – Code qu'utilise la requête "localisations".

```

$cc = escapeshellarg($company);
$tt = escapeshellarg($jwt);
$reponse = shell_exec("E:\Python27\python.exe ajout.py $cc $tt");

```

FIGURE 2.49 – Code pour la requête d'ajout sur PHP

```

import redis
import sys
r = redis.StrictRedis()
r.set(sys.argv[1], sys.argv[2])
r.bgsave()

```

FIGURE 2.50 – Code du fichier ajout.py

recherche.py Ce code permet la recherche de la valeur d'une clé dans le serveur REDIS, pour l'utiliser il suffit de lancer une requête "exec_shell" sur PHP avec une seule variable en paramètres qui est la clé (voir le code ??). Pour le code en

```
$key = shell_exec("E:\Python27\python.exe recherche.py $imei");
```

FIGURE 2.51 – Code pour la requête de recherche sur PHP.

python, il permet la recherche d'une valeur sur le serveur REDIS puis l'afficher avec un "print", et qui servira de return pour l'exec_shell.

```
import redis
import sys
r = redis.StrictRedis()
f = r.get(sys.argv[1])
print f
```

FIGURE 2.52 – Code du fichier recherche.py

3.3 Site Web

En parallèle de l'application Android et du serveur, notre modèle nécessite une interface qui interagit avec le responsable du serveur ou les gérants des entreprises. Pour cela, nous avons mis en place un site Web qui permet de le faire.

3.3.1 Outils de développement

Langage de programmation Comme cité précédemment, pour la conception du site web nous avons utilisé EasyPHP.

Bootstrap Bootstrap est un Framework open source de JavaScript développé par l'équipe de Twitter, il s'agit d'une combinaison de code HTML, CSS et JavaScript conçu pour aider à créer des composants d'interface utilisateur. Bootstrap a également été programmé pour prendre en charge HTML5 et CSS3; il s'appelle également Front-end-Framework⁵. Bootstrap est une collection gratuite d'outils pour créer des sites Web et des applications Web, il contient des modèles de conception HTML et CSS pour la typographie, les formulaires, les boutons, la navigation et d'autres composants d'interface, ainsi que des extensions JavaScript optionnelles.

Design CSS (Cascading Style Sheets) : C'est un langage informatique qui sert à décrire la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites

5. Front-end-Framework : Également appelés 'CSS Framework', ce sont des paquets contenant des codes pré-écrits et standardisés dans des fichiers et des dossiers. Ils nous donnent une base à construire notre web site tout en permettant la flexibilité avec le design final.

web et bien pris en charge par les navigateurs web dans les années 2000.

Photoshop cc (Creative Cloud) : Il s'agit d'un logiciel de retouche image avec une très grande base des filtres et des effets, il a été utilisé dans la construction des différentes icônes.

3.3.2 Présentation des interfaces

Cette partie dénombre la présentation des Scénarios applicatifs du site web (Admincp, diminutif de admin control panel). Nous allons présenter dans ce qui suit, les imprimées-écran des principales interfaces réalisées dans notre panel de contrôle et d'administration.

Login Initialement, l'utilisateur doit se connecter. Il existe deux types de comptes : Admin ou Company. Pour sécuriser les mots de passe, en particulier contre les attaques en force brute, nous avons appliqué l'algorithme de hachage Bcrypt et stocker les sorties dans la base de données. Pourquoi ? Les autres algorithmes de hachage : MD5, SHA1, SHA256 et SHA512 sont très rapides c-à-d vous pouvez calculer au moins 1 000 000 MD5 haches par seconde avec un ordinateur moderne, donc la force brute est possible contre la plupart des mots de passe généralement utilisés (voir la figure 2.53).

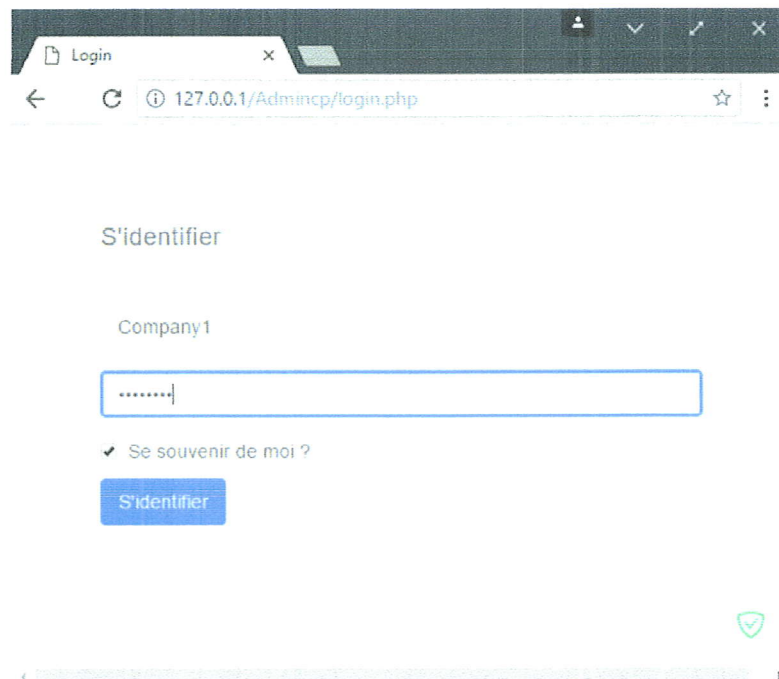


FIGURE 2.53 – La page Login.php

Admincp index C'est la page d'accueil qui s'affiche après l'identification de l'utilisateur (Login), elle est constituée de trois parties principales :

Une barre latérale gauche Située sur la colonne gauche (voir la figure 2.53), elle permet un accès rapide aux différentes pages du site web :

- **Type 'Admin'** : Accueil, Entreprises, Ajouter entreprise, Conducteurs, Ajouter Conducteur, Déconnexion.
- **Type 'Company'** : Accueil, Conducteurs, Ajouter Conducteur, Déconnexion.



FIGURE 2.54 – La barre latérale gauche pour le type "Admin" à gauche et le type "Company" à droite

Une page principale Qui permet à l'utilisateur de suivre un véhicule, en direct ou dans une période de temps. Les entreprises ne peuvent suivre que leurs véhicules tandis que l'administrateur peut suivre n'importe quel véhicule (voir figure 2.55).

Une barre latérale droite Elle contient des informations sur le conducteur suivi (voir figure 2.56). Pour le code HTML nous avons créé un nouveau 'panel' avec id = 'driversInfo', par défaut, il est vide et lorsque nous choisissons de suivre un ou plusieurs conducteurs, un script JavaScript va utiliser cet identifiant et affiche les informations des conducteurs (voir les codes 2.57 et 2.58).

Entreprises Seul le type 'Admin' peut accéder à cette page, elle contient une table des entreprises enregistrées et leurs informations : nom, adresse, numéro de téléphone ... etc. Également des liens pour modifier les informations de l'entreprise ou les supprimer de la base de données (voir figure 2.59).

Ajouter entreprise L'administrateur (type 'Admin') peut à tout moment ajouter une nouvelle entreprise au système en remplissant ses informations et les stocker dans

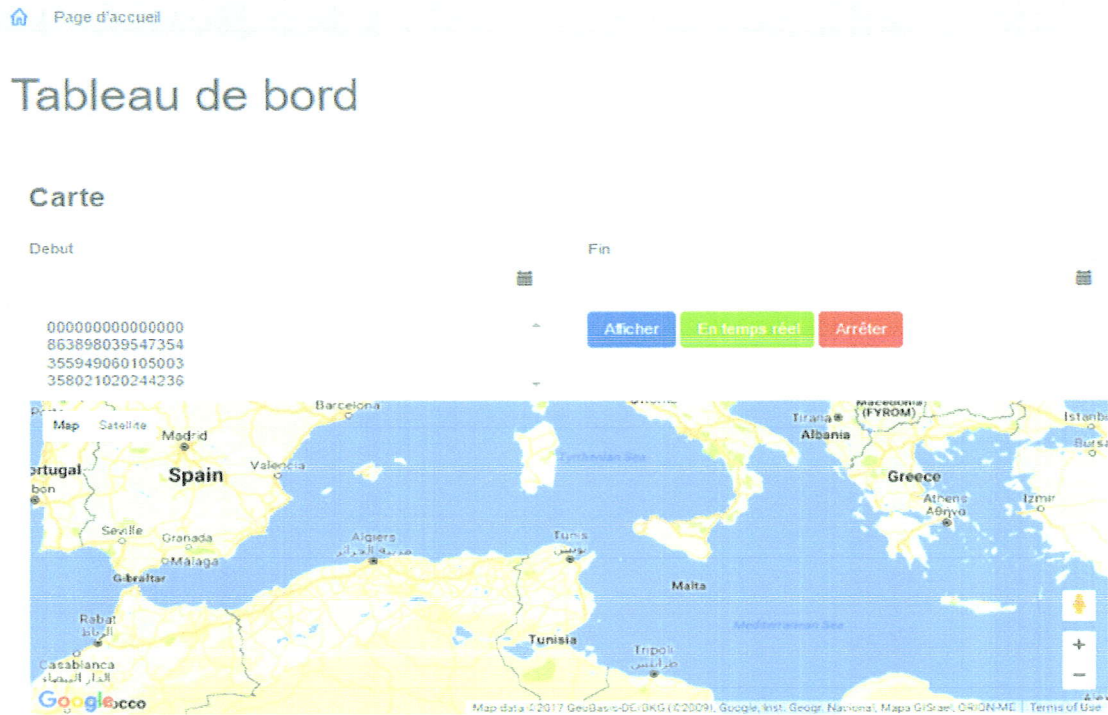


FIGURE 2.55 – Page principale

Conducteur(s)

- ↔ Youcef Berghida
- ↔ Mohammed Zennir
- ↔ Tawfiq Bahri

FIGURE 2.56 – La barre latérale droite

```
<div class="col-lg-4">  
  <div class="login-panel panel panel-default">  
    <div class="panel-heading">Conducteur (s)</div>  
    <div class="panel-body" id="driversInfo">  
  
    </div>  
  </div>  
</div>
```

FIGURE 2.57 – Code HTML qui permet de récupérer les informations des conducteurs


```
var div = document.getElementById('driversInfo');  
div.innerHTML = div.innerHTML + '<h4><i class="fa fa-arrows-h" style="
```

FIGURE 2.58 – Code JavaScript qui permet d’ajouter les informations des conducteurs



Page d'accueil

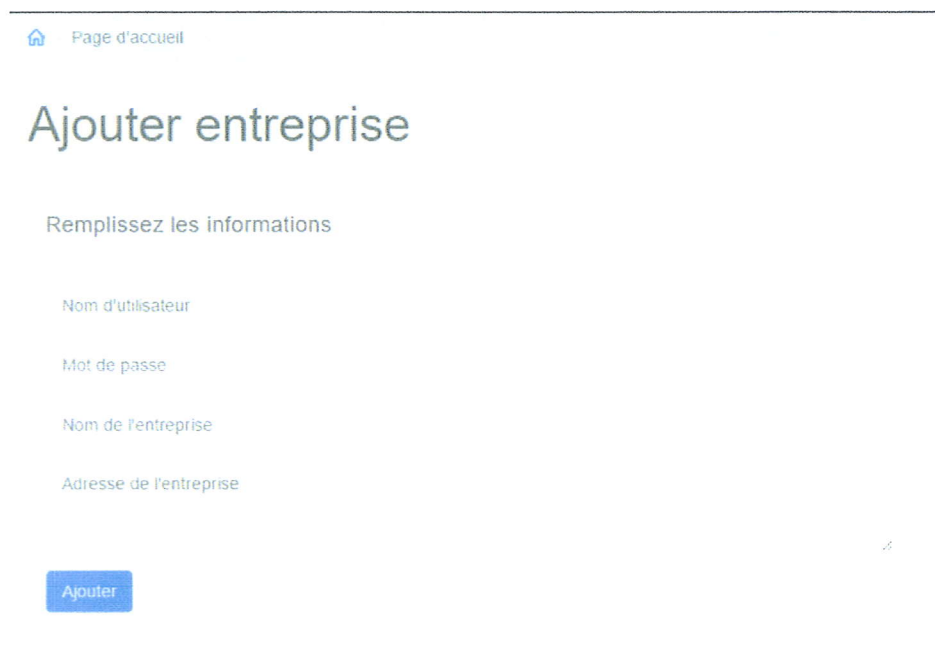
Entreprises

Liste des entreprises

ID	Nom d'utilisateur	Nom de l'entreprise	Adresse de l'entreprise	Date d'inscription		
2	company1	company 1	Jijel - Algerie	2017-04-10	Modifier	Supprimer
3	company2	company 2	Alger - Algerie	2017-04-11	Modifier	Supprimer
4	company3	company 3	Constantine - Algerie	2017-04-11	Modifier	Supprimer

FIGURE 2.59 – La page companies.php

la base de données. Ensuite, la nouvelle entreprise peut se connecter et effectuer différentes actions : ajouter des conducteurs, les suivre ...etc (voir figure 2.60).



Page d'accueil

Ajouter entreprise

Remplissez les informations

Nom d'utilisateur

Mot de passe

Nom de l'entreprise

Adresse de l'entreprise

[Ajouter](#)

FIGURE 2.60 – La page d’ajout d’entreprises, addcompany.php

Conducteurs Elle contient une table des conducteurs enregistrées et leurs informations : nom, adresse, numéro de téléphone, à quelle entreprise ils appartiennent

...etc. Il’y a également des liens pour modifier les informations des conducteurs ou les supprimer de la base de données (voir la figure 2.61). Ps : pour l’administrateur, il a l’option de sélectionner une entreprise et voir ses conducteurs.

ID	IMEI	Nom	Prenom	Numéro de téléphone	Adresse	Date d'inscription	Entreprise		
1	000000000000000	Tawfiq	Bahri	0596984344	Ferdjous	2017-04-11	company 1	Modifier	Supprimer
2	863090039547354	Yucef	Berghia	0596984344	Jijel	2017-04-12	company 1	Modifier	Supprimer
8	4589542567453	Tawfiq	B	0569895930	Alger	2017-04-12	company 3	Modifier	Supprimer
9	355949060105003	Mohammed	Zennir	123	Jijel	2017-05-08	company 1	Modifier	Supprimer
10	358021020244236	Tawfiq	Bahri	0596984344	Alger - Algerie	2017-05-18	company 1	Modifier	Supprimer

FIGURE 2.61 – La page drivers.php

Ajouter Conducteur L’administrateur (type ‘Admin’)ou l’entreprise (type ‘Company’) peut à tout moment ajouter un nouveau Conducteur au système en remplissant ses informations et les stocker dans la base de données (voir figure 2.62). Ps :

Page d'accueil

Ajouter conducteur

Remplissez les informations

company 1

IMEI

Nom

Prenom

Numéro de téléphone

Adresse

Ajouter

FIGURE 2.62 – La page newdriver.php

pour l’administrateur, il doit sélectionner à quelle entreprise le Conducteur appartient parmi une liste des entreprises enregistrées dans le système.

Le suivi des conducteurs

Dans cette partie, nous voulons être capable de suivre un ou plusieurs conducteurs, qu'ils soient en ligne en cliquant sur 'En temps réel' ou hors ligne en sélectionnant la période de temps et cliquer sur 'Afficher'. Nous voulons également afficher les résultats reçus du serveur directement sur la carte de manière dynamique, ce qui a été implémenté dans le script JavaScript nommé 'request.js'.

Request.js Le code dans le fichier JavaScript s'exécute lorsque un événement l'invoque. Dans notre cas, le clique sur 3 boutons invoquent le script : 'Afficher', 'En temps réel', 'Arrêter'.

Bouton 'Afficher' Nous ajoutons un événement "listener" sur le bouton avec id : 'btn-request' qui exécute une fonction lorsque le bouton est cliqué, il recevra : [Token (jeton), date de début, date de fin, entreprise(s), conducteur(s)] à partir des entrées en utilisant .val() et nous les reformons dans une variable nommée 'dataString' (voir le code 2.63). Ensuite, nous demandons des données du serveur, attendons sa réponse, en cas de succès, nous affichons des résultats sur la carte; sinon, nous affichons un message d'erreur, nous pouvons effectuer tout cela en utilisant AJAX (AJAX signifie Asynchronous JavaScript And XML).

- Type : 'POST' ou 'GET'.
- Url : 'send.php', c'est la page responsable de la communication avec le serveur, l'envoi des demandes, l'application des fonctions sur sa réponse pour la renvoyer au script qui effectuera également certaines fonctions aux données avant de les afficher sur la carte dans le cas de succès ou afficher des messages d'erreur en cas d'échec. 'send.php' sera abordé plus tard.
- data : dataString.
- Type : 'html' dans notre cas.
- Success : fonction (données) { MapLoad () }, Aller à la fonction mapLoad () en cas de succès.

```

$("#btn-request").click(function() {
var token = $("#input#token").val();
var startDate = $("#input#startDate").val();
var endDate = $("#input#endDate").val();
var opCompanies = $("##op-companies").val();
var opDrivers = $("#select#op-drivers").val();

var dataString = 'type=Locations' + '&token=' + token + '&startDate=' +
$.ajax({
type: "POST",
url: 'send.php',
data: dataString,
dataType: "html",
success: function(data) {
mapLoad(data);
}
})
return false;
});
    
```



FIGURE 2.63 – Code en JavaScript du bouton 'Afficher'

Boutons ‘En temps réel’ Comme précédemment, nous utilisons une variable globale nommée ‘currentDate’ qui retourne la date et l’heure exactes où le bouton a été cliqué. La deuxième date ‘endDate’ est un timer qui renvoie la date et l’heure chaque 20 secondes (voir les codes 2.64, 2.65 et 2.66).

```
// date stuff -----
var newDate = new Date();
var Year = newDate.getFullYear();
var month = newDate.getMonth()+1;
var day = newDate.getDate();
var hour = newDate.getHours();
var currentDate = day + '-' + month + '-' + Year + ' ' + hour + ':00:00';
//// end date stuff
```

FIGURE 2.64 – Code en JavaScript de la variable ‘currentDate’.

```
var timer = null,
    interval = 20000, // 20 seconds
    value = 0;
var button = document.getElementById('btn-live');
$("#btn-live").click(function(event) {
    event.preventDefault();
    if (timer !== null) return;
    timer = setInterval(function () {
        button.click();
    }, interval);
});
```

FIGURE 2.65 – Code en JavaScript qui permet de cliquer automatiquement sur le bouton ‘En temps réel’ chaque 20 secondes.

```
// Live
$("#btn-live").click(function() {
    var token = $("#inputstoken").val();
    var opCompanies = $("#op-companies").val();
    var opDrivers = $("#selectop-drivers").val();

    var newDate1 = new Date();
    var Year1 = newDate1.getFullYear();
    var month1 = newDate1.getMonth()+1;
    var day1 = newDate1.getDate();
    var hour1 = newDate1.getHours();
    var endDate = day1 + '-' + month1 + '-' + Year1 + ' ' + hour1 + ':00:00';

    var dataString = 'type=RealTime' + '&token=' + token + '&startDate=' + currentDate + '&endDate=' + endDate
    $.ajax({
        type: "POST",
        url: "send.php",
        data: dataString,
        dataType: "html",
        success: function(data){
            mapLoad(data);
        }
    })
    return false;
});
```

FIGURE 2.66 – Code en JavaScript du bouton ‘En temps réel’.

Le bouton ‘Arrêter’ Le rôle de ce bouton est d’arrêter de rafraîchir ou de mettre à jour la carte, nous avons besoin de ce bouton au cas où nous suivions un conducteur en ligne.


```

$("#btn-stop-request").click(function(event) {
    event.preventDefault();
    clearInterval(timer);
    timer = null;
});

```

FIGURE 2.67 – Code en JavaScript du bouton ‘Arrêter’.

La fonction mapLoad() Dans ce qui suit, nous allons aborder la fonction `mapLoad()` qui est responsable de la lecture des données reçues de ‘send.php’ et de les afficher sur la carte. D’abord, elle vérifiera la réponse du ‘send.php’, si `data = ‘denied’`, cela signifie qu’il y a une exception lors de la demande des emplacements. Dans ce cas, nous affichons directement un message d’erreur. Si le premier mot des données reçues est `data = ‘Offline’`, cela signifie que nous essayons de suivre un conducteur en ligne et la localisation de ce conducteur n’est pas disponible pour le moment; nous affichons un message informant l’utilisateur que le conducteur ‘X’ n’est pas actif pour le moment. Si les deux conditions ci-dessus n’ont pas été vérifiées, ça veut dire que nous avons reçu les emplacements du/des conducteur(s) et nous avons juste besoin de les afficher sur la carte. A ce niveau, les données reçues sont de la forme :[(lat 1, lng 1, speed 1, status 1), (lat 2, lng 2, speed 2, status 2), ..., (lat n, lng n, speed n, status n) conducteur1 (lat 1, lng 1, speed 1, status 1), (lat 2 , Lng 2, speed 2, status 2), ..., (lat n, lng n, speed n, status n) conducteur2 ... etc]. Nous devons les décomposer par espace et les mettre dans un tableau nommé "d". Si la taille du ‘d’ est égale à 2, nous avons des emplacements d’un seul conducteur sinon nous avons des emplacements de plusieurs conducteurs.

Un seul conducteur Dans ce cas, nous allons faire les étapes suivantes :

- Pour chaque emplacement ‘i’, nous obtenons la vitesse et le statut.
- Si `status = ‘Offline’`, nous envoyons la position actuelle au serveur et voyons si elle s’agit d’une zone blanche ou non. ‘send.php’ sera abordé plus tard.
- Si le serveur répond avec ‘Offline’, nous considérons cette position comme une zone blanche et nous l’ajoutons dans un tableau temporaire appelé ‘whiteZone’.
- Si ‘whiteZone’ n’est pas vide et le prochain `status = ‘Online’`, cela signifie que nous sommes hors de la zone blanche, donc nous mettons un marqueur sur la carte et un cercle qui désignent cette zone blanche, et ce bien sûr après avoir calculé son centre et son rayon.
- Définir une couleur pour chaque vitesse [vert, orange, rouge] et ajouter un marqueur si la vitesse est supérieure ou égale à 90 Km/h.
- Nous ajoutons la paire : (lat[i], lng[i+1]) dans un tableau nommé ‘convertedStreet’.
- A partir de ‘convertedStreet’ on trace notre route (voir le code 2.68).
- Nous ajoutons les informations du conducteur sur la barre latérale droite.

Plusieurs conducteurs Premièrement, nous définissons un tableau nommé ‘Colors’ contenant des couleurs pour chaque conducteur. Puis, pour chaque conducteur ‘j’, nous effectuons les opérations suivantes :

- Pour chaque emplacement ‘i’, nous obtenons la vitesse et le statut.

```

// drawing route
for (var l = 0; l < convertedStreet.length-1; l++) {
    var PathStyle = new google.maps.Polyline({
        path: [convertedStreet[l], convertedStreet[l+1]], // lat lng.
        strokeColor: CC[l], // Colour.
        strokeOpacity: 1.0,
        strokeWeight: 5,
        geodesic: true,
        map: map // draw on map.
    });
}

```

FIGURE 2.68 – Code en JavaScript pour tracer la route sur la carte.

- Si status = ‘Offline’, nous envoyons la position actuelle au serveur et voyons s’il s’agit d’une zone blanche ou non. ‘send.php’ sera abordé plus tard.
- Si le serveur répond avec ‘Offline’, nous considérons cette position comme une zone blanche et nous l’ajoutons dans un tableau temporaire appelé ‘whiteZone’.
- Si ‘whiteZone’ n’est pas vide et le prochain status = ‘Online’, cela signifie que nous sommes hors de la zone blanche, donc nous mettons un marqueur sur la carte et un cercle qui désignent cette zone blanche, et ce bien sûr après avoir calculé son centre et son rayon.
- Nous faisons un test sur la vitesse si elle est supérieure ou égale à 90 Km/h, nous ajoutons un marqueur.
- Chaque dix positions, nous ajoutons un marqueur indiquant la vitesse.
- Nous ajoutons la paire : (lat[i], lng[i+1]) dans un tableau nommé ‘convertedStreet’..
- A partir de ‘convertedStreet’ on trace la route de chaque conducteur avec sa couleur obtenue à partir de ‘Colors’ (voir le code 2.69).
- Nous ajoutons les informations du conducteur sur la barre latérale droite avec la couleur qui le représente sur la carte.

```

convertedStreet.push(new google.maps.LatLng(
    parseFloat(arr[i]),
    parseFloat(arr[i + 1])));
bounds.extend(convertedStreet[i/4]);

var road = new google.maps.Polyline({
    path: convertedStreet,
    strokeColor: Colors[c],
    strokeOpacity: 1,
    strokeWeight: 2
});

road.setMap(map);
map.fitBounds(bounds);

```

FIGURE 2.69 – Code en JavaScript pour tracer la route sur la carte (Plusieurs conducteurs).

Send.php C’est le responsable de la communication directe avec le serveur ; d’une manière abrégée, son rôle est :

1. Obtenir des données de ‘request.js’.

2. Les transmettre au serveur.
3. Attendre la réponse du serveur et l'envoyer à 'request.js'.

Tout d'abord, nous récupérons toutes les variables à partir de 'request.js' à l'aide de '\$_POST["variable"]'. Ensuite, nous devons savoir quel est le type de requête : 'type = Locations' ou 'type = RealTime' (voir le code 2.70). S'il s'agit du 'type = Lo-

```
$type = $_POST['type'];
$token = $_POST['token'];
$start = $_POST['startDate'];
$startDate = date("Y-m-d H", strtotime($start));
$end = $_POST['endDate'];
$endDate = date("Y-m-d H", strtotime($end));
$company = $_POST['opCompanies'];
$drivers = $_POST['opDrivers'];
$drivers = split(' ', $drivers);

$donnee = array();
```

FIGURE 2.70 – Code qui permet la récupération des données à partir de 'request.js'.

cations' : nous configurons un tableau nommé '\$arr' qui contient : 'type', 'token', 'company', 'imei', 'startDate' et 'endDate', puis on le convertit en JSON à l'aide de la fonction `json_encode()` et on l'envoie au serveur. Etant donné que la réponse du serveur sera également encodée en JSON, nous devons la décoder et la mettre dans un tableau nommé '\$myArray' (voir le code 2.71).

```
if(strcmp($type, "Locations") == 0) {
    $arr = array('type' => 'Locations',
                'token' => $token,
                'company' => $company,
                'imei' => $drivers[$j],
                'startDate' => $startDate,
                'endDate' => $endDate);

    $data = json_encode($arr);
    $out = curlPost($url, $data);
    $myArray = json_decode($out, true);
```

FIGURE 2.71 – Code qui s'exécute si le type est "Locations"

S'il s'agit du 'type = RealTime' : nous configurons un tableau nommé '\$arr' qui contient : 'type', 'token', 'company', 'imei', 'startDate', 'endDate', ensuite on le convertit en JSON en utilisant la fonction `json_encode()` et on l'envoie au serveur (voir le code 2.72).

Après avoir décodé la réponse du serveur, nous vérifions si elle est vide, cela signifie que le conducteur actuel n'est pas actif pour le moment. Nous envoyons un message de type 'Offline' à 'request.js' qui l'affichera sur la page Web (voir le code 2.73).

Maintenant, nous voyons si `$myArray ['response'] = 'granted'`, cela signifie que nous avons des emplacements reçus par le serveur, nous les mettons dans un tableau aux côtés des informations du conducteur et les envoyons à 'request.js' (voir le code 2.74).

Si `$myArray ['response'] = 'denied'`, il indique que le token (jeton) n'est pas défini ou il a expiré, dans ce cas, nous configurons un tableau de type = 'request' et l'envoyons au serveur demandant un nouveau jeton (voir le code 2.75).

```
else if(stripos($myArray['response'], 'denied') == 0 )
{
    session_start(); // start session
    print($myArray['response']);
    $Date = date('d-m-Y H:i:s');
    $arr = array ('type' => 'request', 'company' => $company , 'date' => $Date);
    $data = json_encode($arr);
    $token1 = curlPost($url,$data);
    $_SESSION['token'] = $token1; // set session
}
```

FIGURE 2.75 – Code qui s'exécute si `$myArray ['response'] = 'denied'`

4 Conclusion

Dans ce chapitre, nous avons discuté de la mise en place de notre modèle dans tout ses aspects, notamment dans l'aspect théorique, et l'aspect pratique, puis nous l'avons développé en utilisant plusieurs technologies (Java/Android Studio, PHP, Python).

Jusqu'à présent, notre système fonctionne comme un tracker classique, et pour le développer d'avantages, nous avons établi 2 autres fonctionnalités, qui sont la gestion des zones blanches qui représente l'aspect intelligent, ainsi que la gestion de l'agence gouvernementale qui représente l'aspect légale et fiable de notre modèle.

Chapitre 3

Signature et Validation par une Agence gouvernementale

Dans un système de tracking online, les informations sont facilement falsifiables étant donné le manque de notions de sécurité dans ces derniers, une solution serait de renforcer la sécurité pour empêcher les intrus de récupérer ces informations, Mais qu'en est-il des personnes malveillantes qui ne sont pas des intrus mais des utilisateurs qui ont accès au systèmes? Notre idée serait de relier notre système avec un représentant légal, tel qu'une agence gouvernementale entretenue par un huissier de justice.

1 Introduction à la Cryptographie

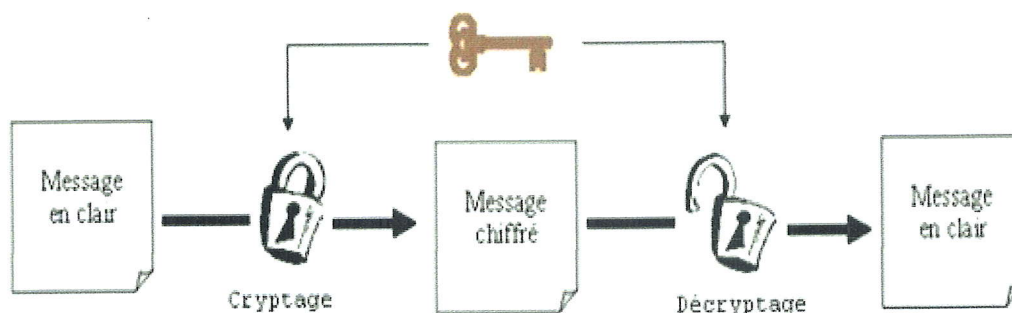
1.1 Définition

Le terme cryptographie provient des deux mots grecs anciens « Kruptos » qui signifie « cacher » et « graphein » qui signifie « écrire » [5]. Ce qui signifie littéralement, « cacher l'écriture ». C'est l'art d'écrire en chiffres ou d'une façon secrète quelconque [13]. Selon Larousse : « Ensemble des techniques de chiffrement qui assurent l'inviolabilité de textes et, en informatique, de données. » [9].

La cryptographie sera utilisée à plusieurs reprises dans notre travail pour sécuriser les informations sensibles, notamment lors des échanges d'informations entre le serveur principale et le dispositif Android, et entre le serveur principale et l'entité gouvernementale.

1.2 Cryptographie Symétrique

La cryptographie symétrique (ou cryptographie à clé secrète) est la forme la plus ancienne de cryptographie. Ce chiffrement fonctionne avec une clé secrète, bien qu'il existe certains chiffrements symétriques qui n'utilisent pas de clé, comme par exemple le chiffrement de César. Dans le cas des chiffrements avec clé, le principe est le suivant : L'émetteur du message chiffre les données grâce à une clé. Cette clé est généralement une chaîne de caractères. Le message chiffré et quasi impossible à décrypté sans la clé il est quasi impossible (le niveau d'impossibilité dépend du niveau de protection du chiffrement utilisé ainsi que de la complexité de la clé utili-



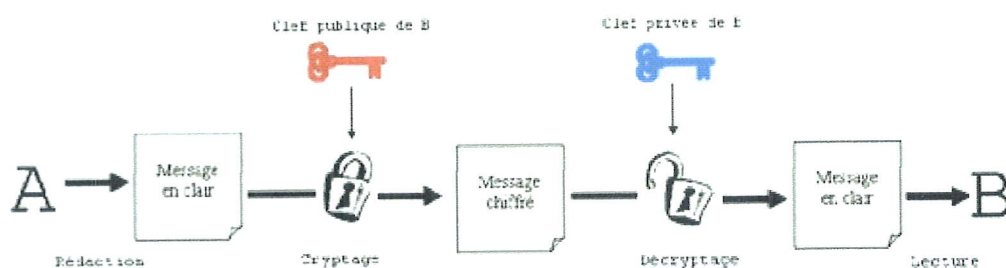
Cryptographie à clé privée : la même clé est utilisée pour crypter et décrypter.

FIGURE 3.1 – Schéma de fonctionnement de la cryptographie symétrique.

sée). L'émetteur doit donc transmettre la clé aux personnes avec qui il désire établir une communication sécurisée s'il veut que son message puisse être lu.

1.3 Cryptage Asymétrique

La cryptographie asymétrique ou cryptographie à clé publique fonctionne de façon totalement différente de la cryptographie symétrique. Si l'on peut comparer la cryptographie symétrique à un coffre-fort auquel seules les personnes possédant la clé peuvent accéder, la cryptographie asymétrique pourrait être comparée à une boîte aux lettres dans laquelle on peut déposer des informations, et seule la personne possédant la clé peut accéder au contenu de la boîte. La boîte aux lettres serait la



Cryptographie à clé publique : le message est chiffré avec la clé publique du destinataire et décrypté avec sa clé privée.

FIGURE 3.2 – Schéma de fonctionnement de la cryptographie asymétrique.

clé publique (donc accessible pour tout le monde), alors que la clé pour ouvrir la boîte serait la clé privée. En effet, dans la cryptographie asymétrique, il y a une clé publique et une clé privée.

1.4 Hachage

Les fonctions de hachage permettent de chiffrer un message sous la forme d'une chaîne de caractères de taille fixe généralement entre 128 et 512 bits, peu importe la taille du message d'origine. Elles sont comparables à une empreinte, car un message aura toujours la même empreinte en appliquant une fonction de hachage. Elles sont

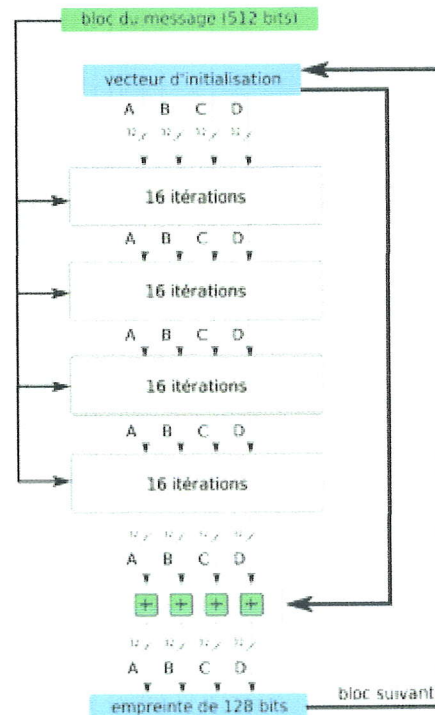


FIGURE 3.3 – Vue générale de MD5.

à sens unique, ce qui signifie qu'il est facile de hacher un message, mais qu'il n'est en principe pas possible de calculer son inverse, à moins d'utiliser une méthode de force brute. Pour qu'une fonction de hachage soit sûre, elle doit être résistante aux collisions. En partant du principe qu'il y a une infinité de messages possibles pouvant être chiffrés, il est évident que plusieurs messages vont donner le même résultat une fois hachés. La résistance est le fait que les collisions ne puissent pas être retrouvées. La fonction de hachage MD5 (Message Digest 5) a été développée par Ronald Rivest, l'un des créateurs de RSA. Elle utilise des blocs de 512 bits et génère des messages chiffrés de 128 bits. Chaque bloc est découpé en 16 sous-blocs de 32 bits (A, B, C et D). Les quatre calculs suivants sont réalisés 64 fois :

1. $F = (B \text{ AND } C) \text{ OR } (\neg(B) \text{ AND } D)$
2. $F = (D \text{ AND } B) \text{ OR } (\neg(D) \text{ AND } C)$
3. $F = B \text{ XOR } C \text{ XOR } D$
4. $F = C \text{ XOR } (B \text{ OR } \neg(D)) \dots$

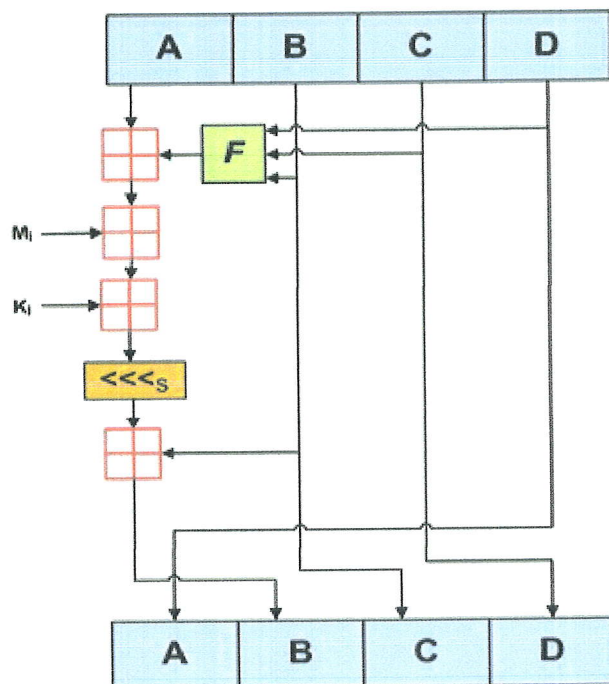


FIGURE 3.4 – Schéma représentant un tour dans MD5.

2 L'Entité Gouvernementale

C'est la partie qui s'occupe du côté légal de l'application. Son rôle est la signature des documents fournis par le serveur centrale. Enfin de documents, le serveur principal calcule la somme de ce dernier puis l'envoi ainsi que d'autres informations par un canal sécurisé avec une clé symétrique connue du serveur et de l'entité gouvernementale. Cette dernière sera crypté avec une clé symétrique propre à l'entité, et sera ajoutée dans le document. Cette manipulation permet de contrôler l'intégrité et la non répudiation des documents, étant donné qu'une bonne partie des accidents mortels sont causés par les véhicules lourds selon les statistiques (dont la majorité appartient à des entreprises). Dans certains cas, les chauffeurs des poids lourd sont coupables de délit de fuite après la collision. C'est dans ces cas que cette entité entre en jeu, elle permettra de déterminer les responsabilités de l'entreprise.

3 Développement de l'Entité Gouvernementale dans le système

Pour la mise en place du système de signatures pour l'agence gouvernementale, nous avons eu besoin d'une nouvelle machine (qui servira de serveur à l'entité) et cette dernière disposera d'un serveur REDIS pour pouvoir stocker et gérer les couples clé/valeur rapidement. Aux niveau des codes, le système n'a besoin que d'un script PHP, donc un serveur EasyPHP pour permettre un bon fonctionnement, ainsi que les deux scripts en python pour ajouter/rechercher dans le REDIS. Pour faire appel à cet organe, nous proposons un code dans le script index.php (la

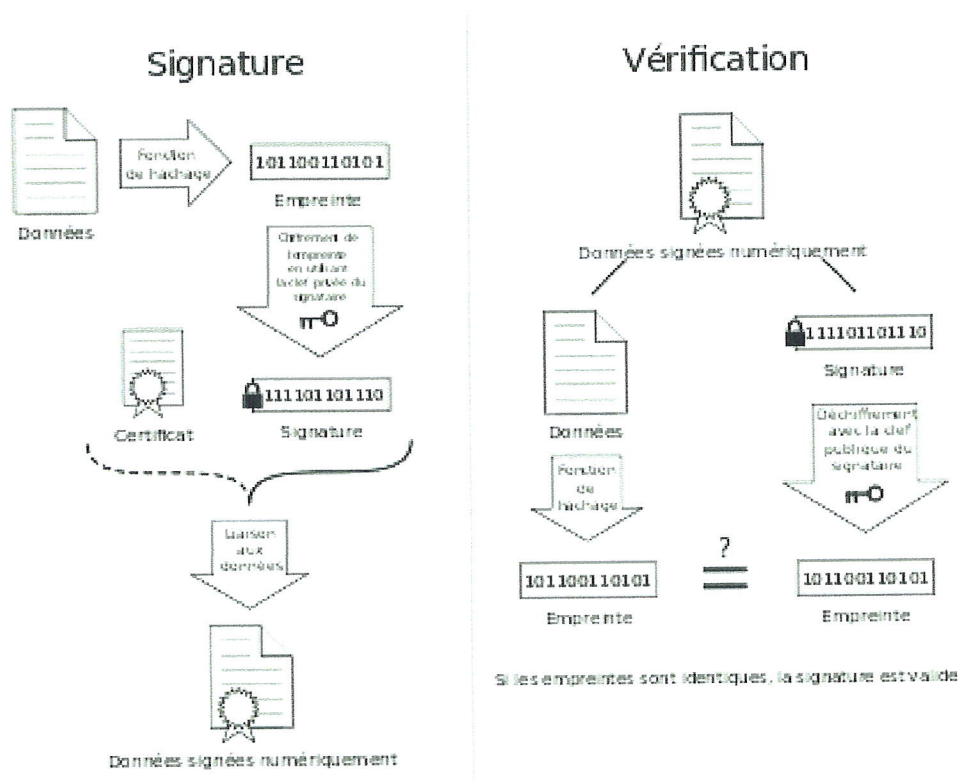


FIGURE 3.5 – Diagramme montrant comment des données sont signées, puis vérifiées.

page responsable de la création de la base de donnée) qui permet de déterminer si le document de l'heure courante est achevé en vérifiant deux conditions : La première c'est si l'heure est égale à X heure et 59 minutes , la deuxième est si on ajoute 10 seconds au seconds actuelles, la somme dépassera 60 (passage à une nouvelle heure). Si les conditions sont vérifiés, le fichier est alors chiffré avec une clé symétrique connu des deux serveurs, ensuite envoyer au serveur gouvernemental. (Voir algorithme 3.6). De son côté, le serveur gouvernementale récupère le fichier envoyé, le déchiffre grâce

```

if(strcmp($minute, "59") == 0){
    $intseconds = intval($second);
    $sec = $intseconds + 10;
    if($sec >= 60)
    {
        $clele = "123456789abcdefg";
        $dataa = array('company'=>$mccrypt->encrypt($soc,$clele),');
        $lss = json_encode($dataa);
        $ft = curlPost("http://127.0.0.1/gov.php",$lss);
        $current .= "$ft.".\r\n";
        file_put_contents($myfile, $current);
    }
}
}

```

FIGURE 3.6 – Partie du code du script index.php qui fait appel à l'entité gouvernementale

```
online      2017-05-07 13:59:52      2017-05-07 13:59:45  
077260d16c582e7eb52641569dfb85f5ccbfb7f318273ad16196e52414ce54d3
```

FIGURE 3.7 – Partie d’un document de la base de donnée qui continent le hash chiffré

à la clé symétrique, puis crée deux clés grâce à la fonction que nous avons établi précédemment, une clé de 24 bits (pas obligatoirement) pour créer un token basé sur le nom de la société et la date actuelle, ce token sera inclut au début du fichier à signer. Le serveur calculera alors le hash du fichier, le chiffre grâce à une nouvelle clé de 16 bits (obligatoirement), puis l’envoi au serveur principale (Voir algorithme 3.8). Le serveur gouvernementale stockera les 2 clés créer, la date actuelle, ainsi que le hash dans le serveur REDIS qui lui est propre, tout en ayant des syntaxes de clés REDIS qui contiennent le nom de la compagnie et l’imei du smartphone. Lorsque le serveur central recevra le hash chiffré, il l’ajoutera à la fin du fichier (voir figure 3.7).

4 Conclusion

Dans ce chapitre nous avons abordé l’élément qui augmente les capacités de notre modèle par rapport aux trackers du marché : l’agence gouvernementale. Nous avons expliqué le fonctionnement systémique de cet organe et décrit les processus à l’œuvre ainsi que les communications qui peuvent s’établir avec le serveur central de l’entreprise.

L’ajout de cette fonctionnalité permet non seulement de bien sécurisé le système, mais aussi de donnée un aspect plus fiable et professionnel vis-à-vis des autres trackers du marché.


```
$cle = createpassword(16);
$clesend = "123456789abcdefg";
$key = createpassword(24);
$date = date('d-m-Y H:i:s');
$data = file_get_contents('php://input');
$data = json_decode($data);
$company = $mcrypt->decrypt($data->company, $clesend);
$timei = $mcrypt->decrypt($data->vehicle, $clesend);
$date = $mcrypt->decrypt($data->date, $clesend);
$content = $mcrypt->decrypt($data->content, $clesend);
$token = array(
    "company" => $company,
    "date" => $date
);
$jwt = JWT::encode($token, $key);
$txt = "$jwt."."\r\n";
$txt .= $content;
$hash = md5($txt);
$md5crypt = $mcrypt->encrypt($hash, $cle);
$content = $md5crypt;
echo "$content";
$redis1 = "$company"."/".$timei"."/".$key";
$redis2 = "$company"."/".$timei"."/".$md5";
$redis3 = "$company"."/".$timei"."/".$date";
$redis4 = "$company"."/".$timei"."/".$crypto";
$redis1 = escapeshellarg($redis1);
$redis2 = escapeshellarg($redis2);
$redis3 = escapeshellarg($redis3);
$redis4 = escapeshellarg($redis4);
$key = escapeshellarg($key);
$hash = escapeshellarg($hash);
$date = escapeshellarg($date);
$exec1 = shell_exec("E:\Python27\python.exe ajout.py $redis1 $key");
$exec2 = shell_exec("E:\Python27\python.exe ajout.py $redis2 $hash");
$exec3 = shell_exec("E:\Python27\python.exe ajout.py $redis3 $date");
$exec4 = shell_exec("E:\Python27\python.exe ajout.py $redis4 $cle");
-?>
```

FIGURE 3.8 – Partie du code du script gov.php



Chapitre 4

Gestion des zones blanches

L'évolution de la téléphonie a permis plusieurs choses positives dans le monde, notamment l'augmentation des débits d'upload et de download, qui a permis lui-même le basculement de la téléphonie de génération vers génération, de la voix uniquement, à l'audiovisuelle en directe et sans perte.

Cette évolution a aussi permis l'extension continue des zones de couvertures à travers le monde, mais quant est-il des zones isolées et dépourvues de population ? Cela représentait un réel problème pour MVT, surtout pour les sociétés de livraisons officiant au national.

1 Les zones de couvertures des opérateurs algériens

Dans le domaine de la téléphonie, une zone blanche désigne une zone qui ne possède pas de réseau de téléphonie mobile et internet. Il s'agit la plupart du temps de zones très peu peuplées avec un nombre d'habitants extrêmement peu dense, n'incitant pas les opérateurs mobiles à équiper ces zones, qui ne seraient pour eux pas rentables. Cela représente un vrai handicap pour notre système. Selon les 3 opérateurs téléphoniques historiques, il y'a toujours beaucoup de zones non couvertes par la 3G (voir les figures 4.1 4.2 et 4.3) surtout si la flotte passe par ces zones (surtout dans le sud Algérien). L'idée pour contrecarrer cet handicap est de faire fonctionner l'application en "offline" lorsqu'elle pénètre une zone blanche, et tenter de calculer le périmètre de la zone, en ayant recours à un apprentissage automatique. Dans les enregistrements de la base de données, nous avons mis un couple date et heure smartphone/ date et heure serveur qui nous permettra de synchroniser les informations lorsque le système passe en online avec un système de file d'attente. Nous avons aussi introduit un champ qui nous permet de savoir si ces informations ont été enregistrées dans une zone blanche ou non. En cas de zone blanche du GPS (le satellite qui ne couvre pas cet emplacement bien que cela soit rare), il n'est malheureusement impossible de remédier à la situation.

2 Notre solution

Nous avons établi un apprentissage qui distingue, plus ou moins, les zones blanches d'autres problèmes techniques (Smartphone défectueux, un problème de

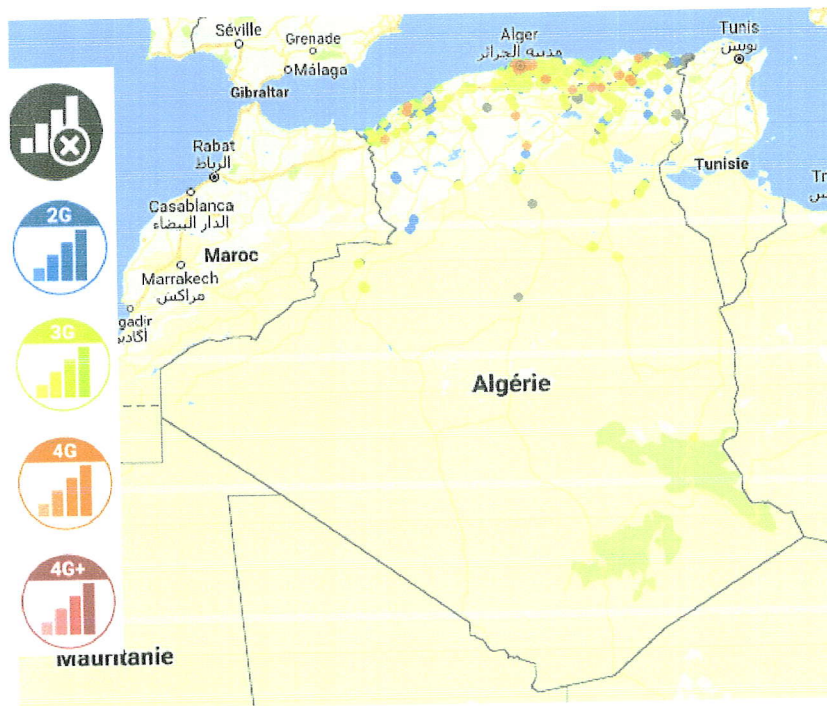


FIGURE 4.1 – Zone de couverture Djezzy

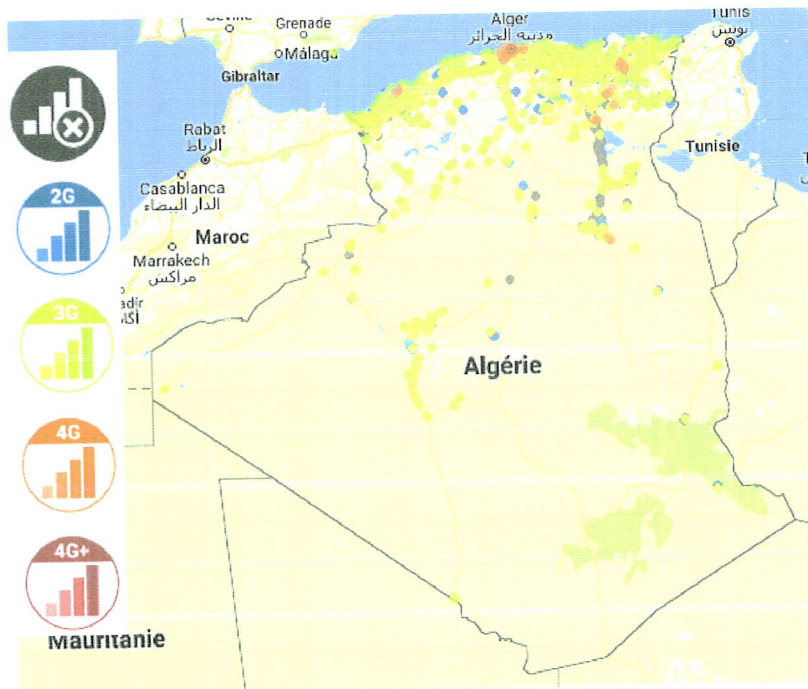


FIGURE 4.2 – Zone de couverture Mobilis

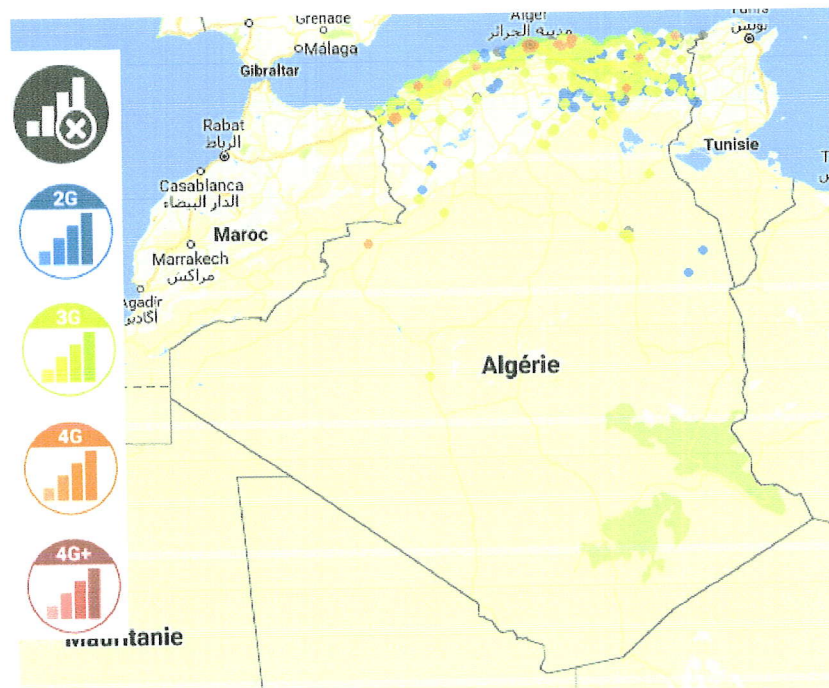


FIGURE 4.3 – Zone de couverture Ooredoo

batterie à plat, etc). Pour celà nous avons eu recours à l’algorithme des K-plus proches voisins.

2.1 Algorithme des K-plus proches voisins

L’algorithme des K-plus proches voisins (K-PPV ou K-NN en anglais pour nearest neighbors) est une méthode d’apprentissage supervisé non paramétrique, utilisée généralement pour la classification et la régression [12]. (voir pseudo algorithme 4.4). Le fonctionnement de cet algorithme repose sur le choix de l’entier K. Tout d’abord,

```

k-Nearest Neighbor
Classify (X, Y, x) // X: training data, Y: class labels of X, x: unknown sample
for i = 1 to m do
    Compute distance  $d(X_i, x)$ 
end for
Compute set I containing indices for the k smallest distances  $d(X_i, x)$ .
return majority label for  $\{Y_i \text{ where } i \in I\}$ 
    
```

FIGURE 4.4 – Pseudo Algorithme K-PPV

il calcule les distances (la distance est choisie en fonction des attentes, telle que la distance euclidienne) entre les couples présents dans la base de donnée avec un couple choisi pour déterminer son type, une fois les distances calculer, il les organise

par ordre croissant, et il prend les K premiers couples. Le couple choisi prendra le type dominant entre les K premiers couples. (voir figure 4.5)

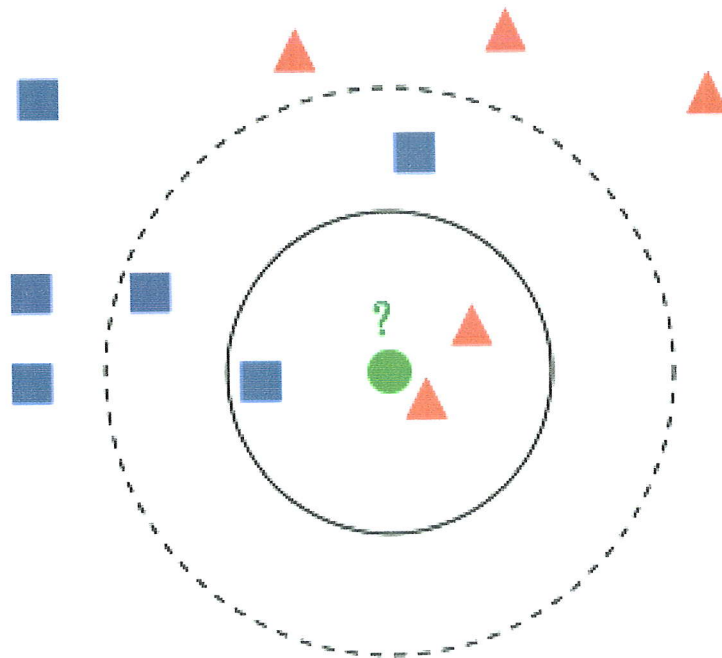


FIGURE 4.5 – Schéma qui représente un exemple d'une classification à base d'algorithme K-PPV

2.2 Implémentation dans notre système

L'algorithme des K-PPV est utilisé dans notre système pour classifier les coordonnées, c.à.d, essayer de distinguer les coordonnées présentes dans des zones blanches de celles qui se trouvent en dehors de ces dernières. Des problèmes techniques peuvent survenir pendant le fonctionnement du système, et qui pourraient conduire l'application Android à traiter de fausses informations (signaler la présence d'une zone blanche alors que ce n'est pas le cas). Cet algorithme utilisera la base de données hiérarchique comme base d'exemples, et calculera la distance entre tout les points géographiques et le point dont on cherche le type. En choisissant K, il déterminera le type du point géographique. (voir algorithme 4.6). Dans notre système, ce code sera implémenté dans le script server.php, qui lui rajoute au final une troisième fonctionnalité (que nous avons surnommé "app" diminutif de apprentissage). Le client envoi au serveur une requête avec comme type "app", la localisation à tester, ainsi que le paramètre K. Le serveur lit tous les fichiers de la base de données, les stockent dans un tableau, puis appelle l'algorithme qui calculera les distances entre le point et le contenu du tableau précédent. Il retournera un résultat qui peut être "Offline" ou "Online", dont cee dernier sera envoyer au Client après. Si le résultat est "Offline", alors le point sera défini comme étant présent dans une zone blanche, sinon, le niveau de batterie sera examiné pour déterminer si le téléphone portable fonctionne normalement ou a subit une extinction, sinon le problème sera classifié comme problème technique inconnu (la majorité des cas si non présence de réseau 2G et 3G en même temps).

```

$dist = array();
$skppv = array();
$test = array();
$ll = explode(":", $loc);
for($i=0;$i<count($coors[0]["items"]);$i++)
{
    $dist[] = array("distance"=>distance(floatval($coors[0]["items"][$i
}
function method1($a,$b)
{
    return ($a["distance"] <= $b["distance"]) ? -1 : 1;
}
usort($dist, "method1");
for($i=0;$i<$nbrk;$i++)
{
    $kppv[] = $dist[$i]["type"];
}
$res = array_count_values($kppv);
$result = null;
if($res["Offline"]>$res["Online"])
{
    $result = "Offline";
}
else
{
    if($res["Offline"]<$res["Online"])
    {
        $result = "Online";
    }
    else
    {
        $strings = array(
            'Offline',
            'Online',
        );
        $key = array_rand($strings);
        $result = $strings[$key];
    }
}
$response = (object) array('response'=>'granted', 'result'=>$result);

```

FIGURE 4.6 – Fragment du code qui utilise l’algorithme K-PPV

3 Conclusion

Dans ce chapitre, nous avons présenté la problématique des zones blanches en Algérie surtout dans les zones Sud du pays (suivant les cartes des couvertures des opérateurs). Nous avons introduit l'algorithme des k-plus proches voisins dans notre modèle afin d'établir un système qui permet de déduire les zones blanches en toute autonomie. Celà dit, le problème reste le choix du paramètre K afin d'avoir une déduction optimale, et pour établir ce dernier, une approche empirique a été menée.

Chapitre 5

Testes et expérimentations de MVT

Dans ce chapitre, nous allons présenter aux utilisateurs et aux développeurs notre application Android : ‘Multi-Vehicles Tracker’ (MVT) ainsi que ses fonctionnalités, en fournissant des informations sur le démarrage et l’installation de l’application sur les appareils mobiles. Nous allons aussi effectuer plusieurs tests sur notre modèle (serveur, application androïde et client) et voir les résultats obtenus du côté client (application web).

1 Configurer l’environnement de développement

Un développeur typique doit configurer un environnement pour le développement Android. Les étapes d’installation indiquées ci-dessous supposent que le développeur est conscient de la mise en place de l’environnement de développement pour ce type de développement. Cette application peut être implémentée sur n’importe quelle plate-forme qui fonctionne avec Java / Android. Les étapes d’installation sont :

- Installer Java JDK¹ et JRE².
- Installer ‘Android Studio’ ou ‘Eclipse IDE’.
- Installer le SDK Android.
- Installer le plug-in ADT dans Eclipse pour Android Développent.

Une fois la configuration de la machine est terminée, l’étape suivante consiste à télécharger le code source et à y apporter des modifications :

- Copiez les fichiers zip du code source sur votre machine.
- Décompressez le fichier dans votre espace de travail.
- Importez le projet dans ‘Android Studio’ ou ‘Eclipse’.

2 Installer l’application sur un appareil mobile

La première étape consiste à obtenir le fichier ‘.apk’ à partir de la source inconnue ou par une autre manière qui est l’installation à partir de ‘Google Play’ d’une façon

1. Le Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java.

2. L’environnement d’exécution Java (abr. JRE pour Java Runtime Environment), parfois nommé simplement « Java », est une famille de logiciels qui permet l’exécution des programmes écrits en langage de programmation Java1, sur différentes plateformes informatiques.

directe et simple. Si cela doit être téléchargé à partir d'une autre source (autre que 'Google Play'), les paramètres du téléphone doivent être modifiés. La figure 5.1 montre comment activer l'installation à partir d'une source inconnue.

Pour l'installation manuelle, les instructions suivantes doivent être suivies :

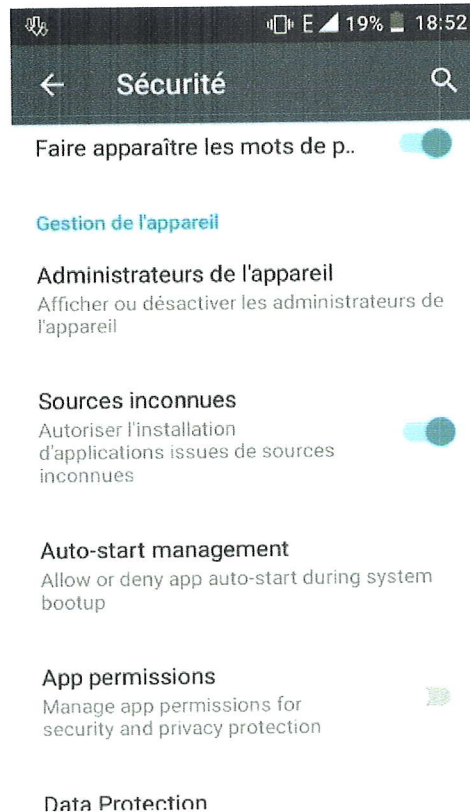


FIGURE 5.1 – Installation à partir d'une source inconnue.

- Cliquez sur le bouton Menu.
- Sélectionnez Paramètres.
- Cliquez sur Sécurité.
- Cochez l'option 'Sources inconnues'.

Maintenant, l'application MVT sera installée sur l'appareil.

3 Tests avec émulateur Android

3.1 Émulateur Android

L'émulateur Android simule un périphérique et l'affiche sur votre ordinateur. Il vous permet de prototyper, développer et tester des applications Android sans utiliser de périphérique matériel.

L'un des meilleurs émulateurs Android gratuits est 'NOX', il est basé sur Android 4.4.2 et peut être téléchargé gratuitement sur <https://www.bignox.com/>

Après avoir téléchargé 'NOX' et l'installé sur le PC, nous l'avons lancé, la figure 5.2 montre l'écran principal de l'émulateur Android 'NOX'.

L'étape suivante consiste à suivre les étapes expliquées précédemment pour instal-

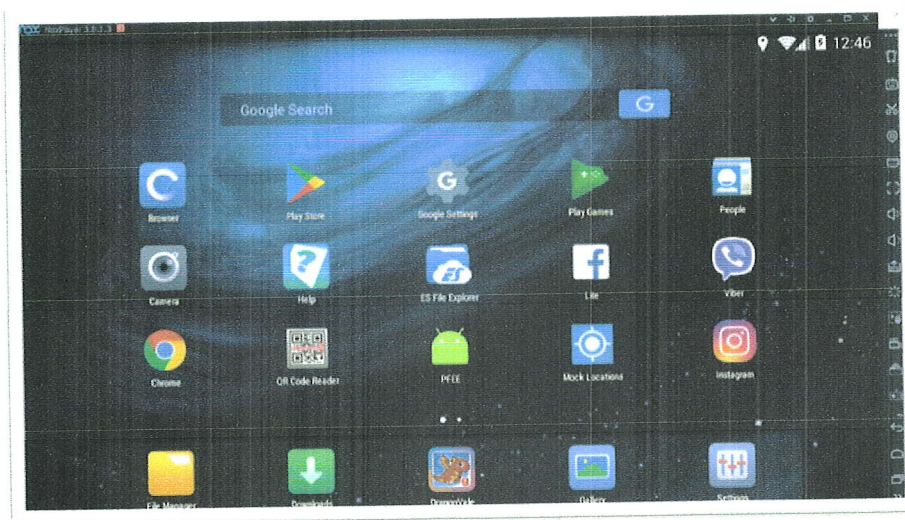


FIGURE 5.2 – Écran principal de l'émulateur Android NOX

ler l'exécutable 'apk' de notre application MVT (installation à partir d'une source inconnue) sur le simulateur Android. Nous lançons ensuite l'application, la figure 5.3 montre l'interface principale de cette dernière. Elle contient le numéro IMEI du périphérique, la localisation (une paire : Latitude, Longitude), la vitesse (en Km/h), le niveau de la batterie et un bouton de configuration.

Nous arrivons à la partie de la configuration, il est important de savoir comment

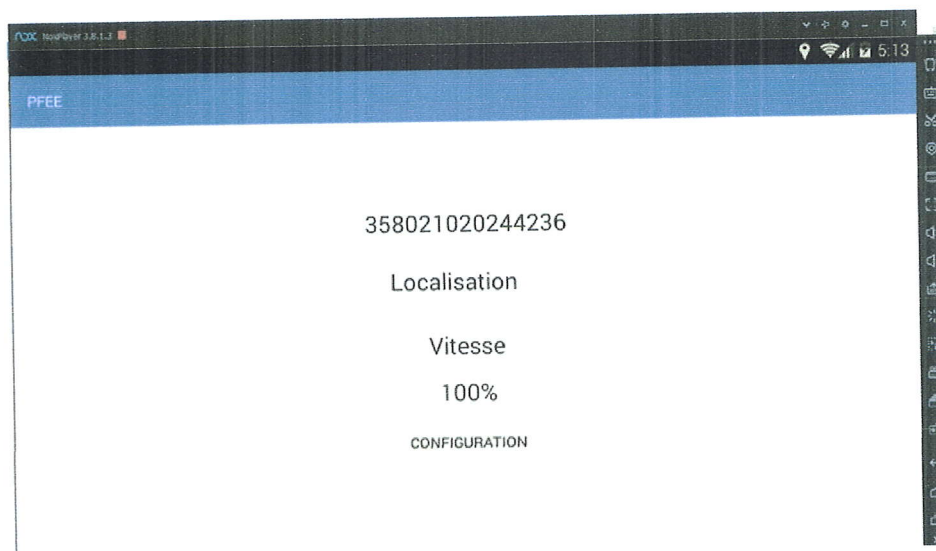


FIGURE 5.3 – Interface principale de l'application Android MVT

configurer notre application Android MVT afin qu'elle puisse fonctionner correctement, envoyer des données correctes au serveur et éviter tous types d'erreurs. Nous commençons par cliquer sur le bouton 'CONFIGURATION', une nouvelle interface sera affichée avec certaines entrées qui doivent être remplies correctement et deux boutons pour valider les modifications et pour revenir en arrière.

La première entrée est : 'URL', elle doit contenir le lien complet vers le serveur

(notre serveur est installé dans un hôte local, donc nous utilisons 'ngrok.exe' qui est un utilitaire qui permet l'accès en ligne d'un serveur locale et qui permet de générer un lien temporaire vers le serveur, voir le chapitre 2 section 3.2).

La deuxième entrée est : 'Société', elle doit contenir le nom de l'entreprise à laquelle le conducteur appartient.

La troisième entrée est : 'Cryptage', elle doit être vide, la clé est fournie par le serveur qui va générer une clé asymétrique, ce qui est très importante pour sécuriser l'échange des données entre le serveur et l'application (voir chapitre 3 section 1).

La quatrième entrée est : 'Mot de passe', elle doit contenir le mot de passe pour se connecter au serveur.

La figure 5.4 ci-dessous illustre comment configurer l'application.

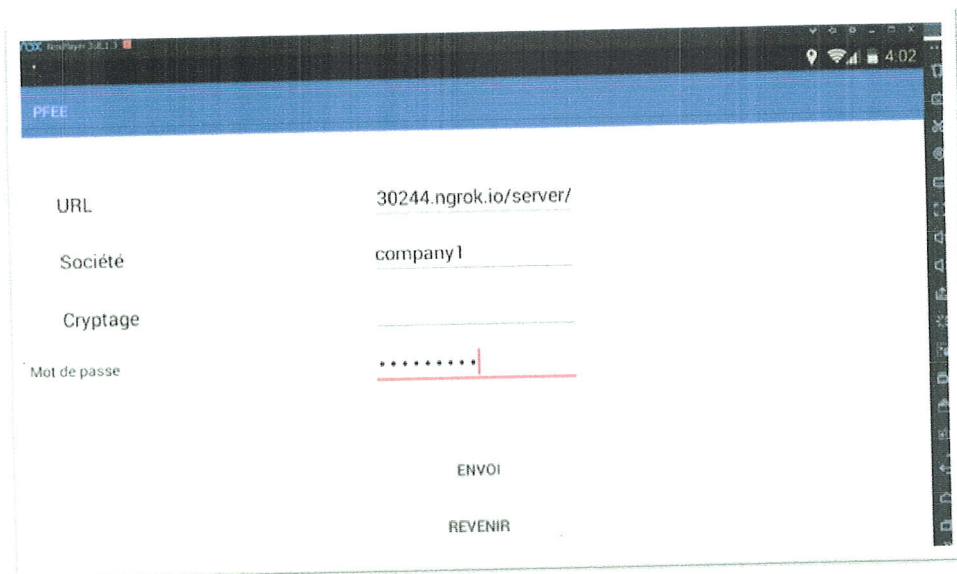


FIGURE 5.4 – Configuration de l'application

3.2 Émulateur de GPS

C'est un programme qui permet de simuler des localisations pour le smartphone en envoyant de faux emplacements. L'utilisation courante des émulateurs de GPS est pour des objectifs de tests. Ils contournent les services de restriction des GPS, triche de jeux ...etc. L'un des émulateurs GPS trouvés sur Internet est 'Mock Locations', qui d'après sa description sur Google Play Store : "il vous permet de changer les informations sur l'emplacement de votre téléphone par le GPS et l'opérateur de réseau". Il peut être téléchargé gratuitement depuis Google Play Store.

L'un des avantages les plus intéressants de 'Mock Locations' est qu'il peut dessiner un itinéraire entre un point de départ et un point final et lui donner une plage de vitesse tout en simulant le mouvement du véhicule. Après avoir installé 'Mock Locations', nous configurons ce dernier en choisissant :

- Point de départ : la ville de Ferdjioua.
- Point d'arrivée : ville de Jijel.
- Vitesse : minimum = 40 km/h, maximum = 100 km/h.



FIGURE 5.5 – Émulateur GPS ‘Mock Locations’.

Si nous revenons à notre application, nous pouvons voir que le simulateur de GPS fonctionne très bien et que nous recevons des coordonnées et des vitesses correspondantes.

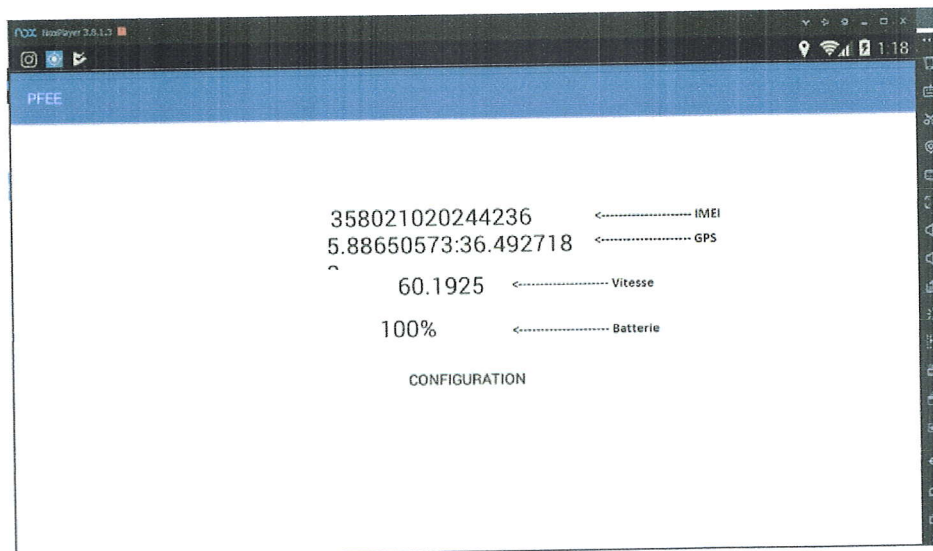


FIGURE 5.6 – Application MVT, réception des coordonnées et des vitesses correspondantes.

3.2.1 Côté Serveur

Après avoir reçu des données de l'application, le serveur doit les structurer et les stocker dans une base de données hiérarchique (voir chapitre 2 section 1.2.1), de sorte qu'il soit facile pour le client d'accéder à ces données. Le serveur va créer un document dans sa base de données. Son chemin est comme suit :

[Société] / [IMEI] / [année] / [mois] / [jour] /hour.txt.

Le fichier créé contient toutes les informations nécessaires :

3. TESTS AVEC ÉMULATEUR ANDROID

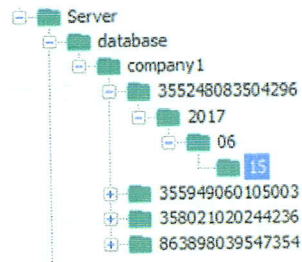


FIGURE 5.7 – La base de données hiérarchique ‘database’.

- État : ‘Online’ ou ‘Offline’.
- Date et heure du smartphone.
- Date et heure de système.
- La vitesse.
- Le niveau de la batterie.
- Les coordonnées (latitude, longitude).

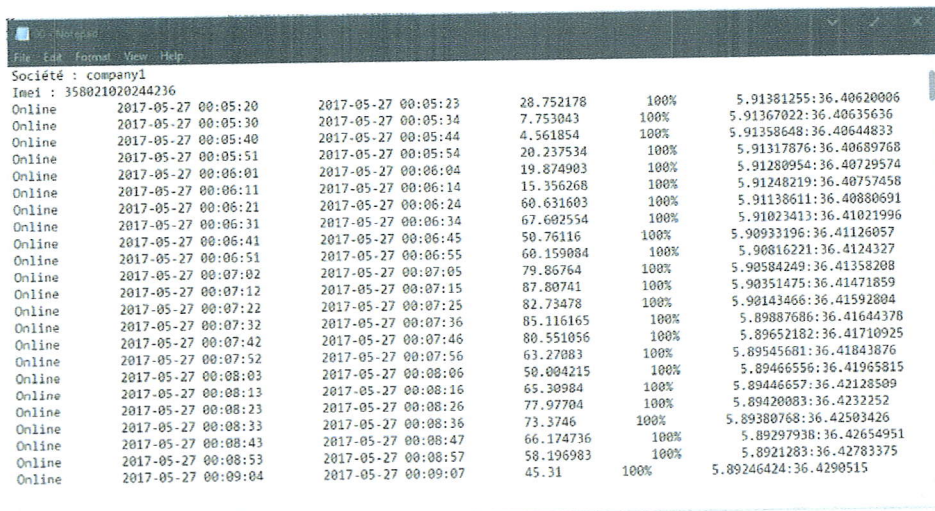


FIGURE 5.8 – Le fichier ‘00.txt’ créé par le serveur

Nous avons programmé notre serveur pour être capable de détecter les changements d’heure afin qu’il puisse créer un nouveau document dans la base de données, mais. Il ne se limite pas à cette tâche, il en a d’autres. Comme nous avons mentionné précédemment (voir chapitre 3), notre modèle implique une entité gouvernementale, et pour la maintenir à jour, nous avons mis en place un processus du côté du serveur qui :

- Après chaque heure, envoie le document précédent (‘00.txt’ dans ce test) à l’entité gouvernementale.
- Cette dernière va le signer et envoyer un hash au serveur.
- Le serveur va mettre ce hash à la fin du document. Le document est dit alors signé.

Name	Date modified	Type	Size
00	15/06/2017 00:59	Text Document	1 KB
01	15/06/2017 01:01	Text Document	2 KB

FIGURE 5.9 – Créer un nouveau fichier ('01.txt').

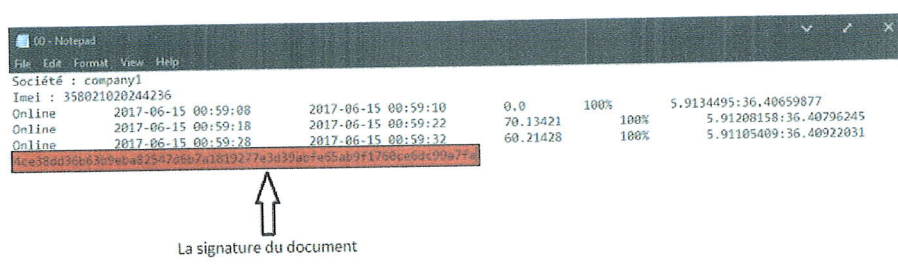


FIGURE 5.10 – Ajouter le hash à la fin du document.

3.2.2 Côté Client

La partie suivante explique les étapes qui permettent au gérant de suivre le véhicule (émulateur Android dans ce test). Après une connexion réussie, l'application Web redirige l'utilisateur vers la page d'accueil où il peut facilement suivre un ou plusieurs conducteurs, Nous devons effectuer ces deux étapes pour suivre notre émulateur Android :

- Nous gardons les deux entrées 'Début' et 'Fin' vides et nous sélectionnons l'IMEI d'émulateur Android.
- Nous cliquons sur le bouton 'En temps réel'.

Les résultats sont affichés sur la figure 5.11. Comme nous pouvons le constater sur la carte, le serveur commence à envoyer des données et le script responsable 'request.js' (voir chapitre 2 section 3.3.2 va les afficher sur la carte. On peut voir deux couleurs sur la carte qui représentent la vitesse.

- Orange : de 50 km / h à 80 km / h.
- Vert : de 0 à 50 km / h.

3.2.3 Simuler les zones blanches

Le principal inconvénient que nous allons trouver sur 'Mock Locations' est qu'il ne peut pas simuler des zones blanches, nous devons donc modifier les documents manuellement sur la base de données.

D'abord, nous ouvrons un fichier (par exemple : 01.txt) et d'une façon manuelle, on change certains mots 'Online' vers 'Offline'.

Le résultat est affiché immédiatement sur la carte sans avoir besoin de rafraîchir la

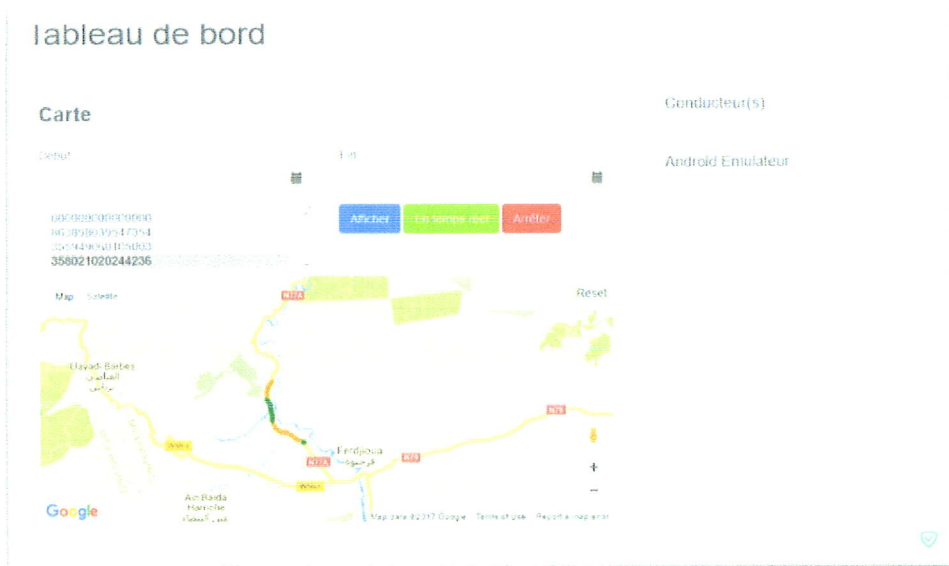


FIGURE 5.11 – Le suivi du simulateur GPS.

page web, comme le montre la figure 5.12. La carte ci-dessus représente la façon dont

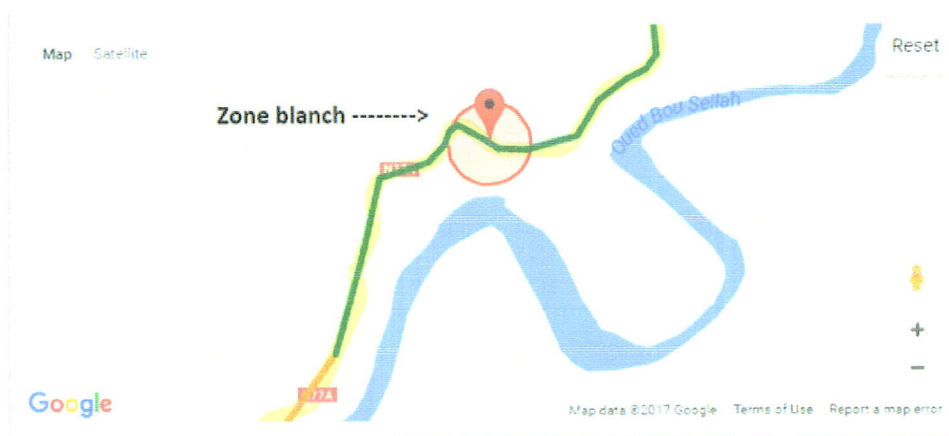


FIGURE 5.12 – Une zone blanche.

les zones blanches sont affichées, comme on peut le voir, le cercle rouge représente la zone blanche avec un marqueur centré sur ce dernier. Pour inculquer aux chauffeurs les bonnes manières d'une conduite sans danger pour eux et pour les autres utilisateurs de la route, nous avons configuré notre modèle MVT pour avertir le gérant en cas d'une négligence de la part de son chauffeur des directives données par l'établissement (mauvaise conduite, excès de vitesse, détours vains ... etc.)

Quand le chauffeur enfreint toutes les règles pour mettre en danger sa vie, l'état du matériel, les utilisateurs de la route, nous avons simulé un test sur l'émulateur Android, un marqueur avec un point d'exclamation est affiché sur la carte, en déplaçant la souris sur ce marqueur, il va afficher la vitesse (voir figure 5.13).

Si nous voulons suivre un conducteur qui n'est pas actif pour le moment, une alerte sera affichée nous informant que le conducteur 'X' n'est pas actif. (voir figure 5.14)



FIGURE 5.13 – Un marqueur avec un point d’exclamation (Vitesse > 90 km/h).

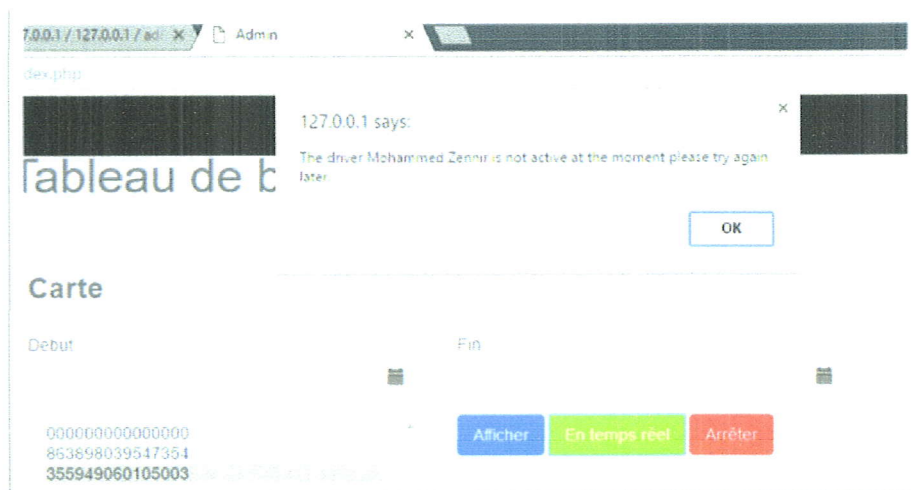


FIGURE 5.14 – Essayer de suivre un conducteur hors ligne.

4 Test réel

Nous avons fait ce test le 8 mai 2017 vers 13h00, nous avons utilisé deux voitures : le premier conducteur était muni d'un smartphone 'Condor', le second conducteur était muni d'un smartphone 'Oppo' (voir figure ?? et 5.16).

- L'adresse du serveur (obtenue à partir de ngrok.exe).
- Société (company1).
- Mot de passe pour se connecter au serveur.

CPU	Qualcomm Dual core 1.2 Ghz
RAM	1 Go
Mémoire interne	8GB
System	Android 4.4.2 Kitkat
Réseau	Wifi 802.11 b/g/n, Bluetooth, Hotknot, EDGE/GPRS/GSM/WCDMA
Batterie	Li-ion Polymer battery of 1800 mAh

FIGURE 5.15 – Caractéristique du smartphone Condor.

CPU	Qualcomm MSM8916 Snapdragon 410Quad-core 1.2 GHz Cortex-A53
RAM	2 Go
Mémoire interne	8GB
System	Android 5.1 Lollipop
Réseau	Wifi 802.11 b/g/n, EDGE/GPRS/GSM/WCDMA/LTE
Batterie	Li-ion Polymer battery of 2630mAh

FIGURE 5.16 – Caractéristique du smartphone Oppo.

Les deux conducteurs ont configuré leurs téléphones : activation des données 3G, activation du GPS, ils ont aussi installé l'application et la configuré par :

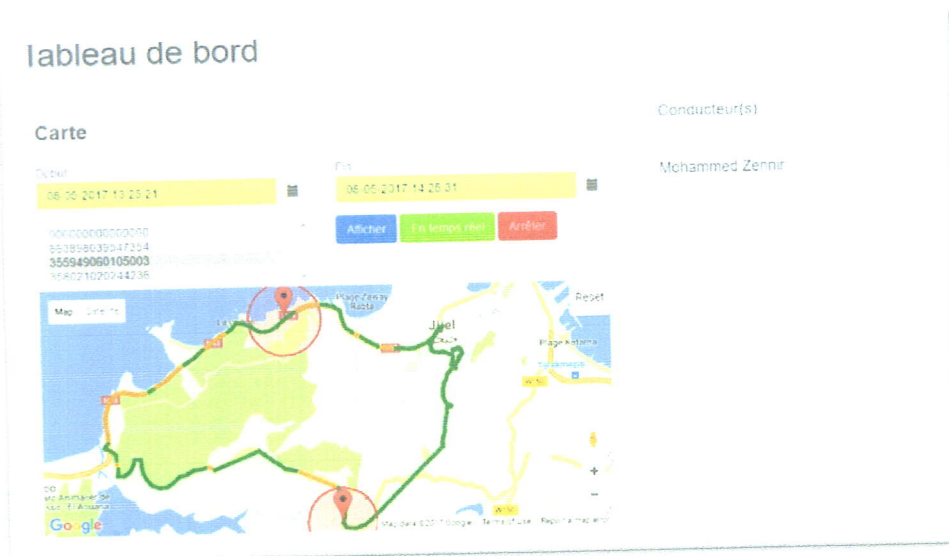


FIGURE 5.17 – Le suivi d'un seul conducteur.

Sur la carte de la figure 5.17, on remarque que le conducteur a effectué le trajet symbolisé par le tracé ci-dessus .les deux couleurs témoignent d'une bonne conduite et d'un respect des vitesses autorisées. Comme on peut constater aussi l'apparition de deux zones blanches. L'une d'entre elles se trouve au niveau de la crique ; l'autre est sur une route départementale (route des poids lourds).

Dans ce cas, nous suivons deux conducteurs. Le trajet de chacun d'eux est affiche



FIGURE 5.18 – Le suivi de plusieurs conducteurs.

d'une couleur (rouge, bleue). La couleur rouge réfère au premier conducteur ; la bleue est pour le second. Pour le premier, une petite zone blanche est apparue car elle est située au centre d'une ville où on trouve d'habitude une bonne couverture réseau. (voir figure 5.18),

5 Les zones blanches

Pour gérer les zones blanches, au niveau du serveur, nous avons mis en place un algorithme de «classification et régression» : K-PPV (voir chapitre 4), ce qui signifie que lorsque nous avons un emplacement ou 'status = Offline', cet algorithme va le vérifier et renvoyer 'Online' ou 'Offline'.

Et pour savoir quel est le meilleur k (le 'K' optimal) dans notre application, nous avons testé différentes valeurs de 'K'. Deux zones blanches apparaissent sur ce trajet (voir figure 5.19),. L'une d'entre elles se trouve au niveau de la crique ; l'autre est sur une route départementale (route des poids lourds). Pour k=5 (voir figure 5.20), les zones blanches demeurent les mêmes. Mais cette fois-ci la surface couverte par la zone blanche se trouvant au niveau de la crique a peu diminué. Pour k=7 (voir figure 5.21),, la zone blanche du même endroit a beaucoup diminué. Pour k=9 (voir figure 5.22),, la zone blanche a complètement disparu. Seule, reste celle se trouvant sur le chemin départemental(route des poids lourds).

D'après les résultats obtenus des différentes valeurs de 'K', on remarque qu'il existe



FIGURE 5.19 – Les zones blanches lorsque $K = 3$.

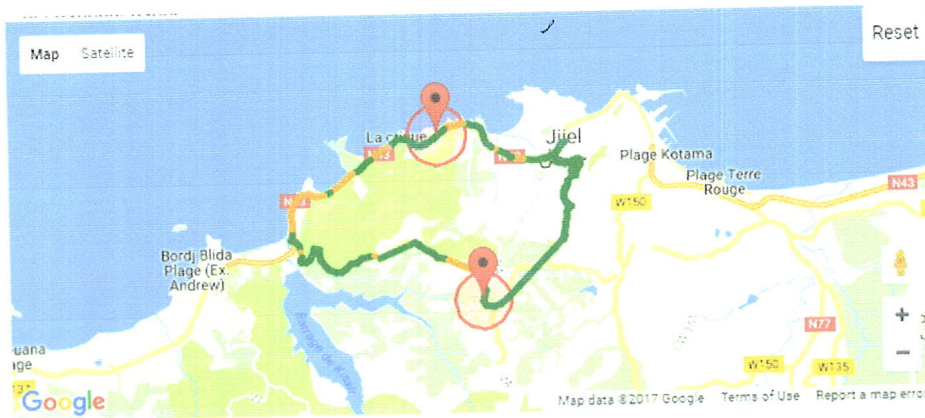


FIGURE 5.20 – Les zones blanches lorsque $K = 5$.



FIGURE 5.21 – Les zones blanches lorsque $K = 7$.





FIGURE 5.22 – Les zones blanches lorsque $K = 9$

une relation inverse entre le 'K' et le rayon de la zone blanche, c.-à-d. plus le 'k' grandit plus la zone blanche rétrécit. Celà est dû au fait que si le périmètre d'une zone blanche est petit, il faudra alors peu de temps pour traverser cette zone, donc il faudra passé par peu de positions géographiques, qui résultera en quelques points géographiques de type "offline" contrairement à la vaste majorité des points de type "online".

6 Les limites du positionnement géographique par 3G

Le peu de zone couverte 3G dans le territoire national peu provoquer des pertes d'informations importantes. La récupération de la localisation GPS nécessite des antennes de téléphonie cellulaire. Les smartphone peuvent être configuré pour utilisé le réseau 3g uniquement, mais celà provoquerais une perte complète du réseau dans des zones blanches, qui empêchera la récupération des points GPS. C'est pour celà que l'utilisation de la 2G en même temps que la 3G assurera un fonctionnement optimale du smartphone.

7 Conclusion

Dans ce chapitre nous avons abordé les résultats obtenus des testes effectués sur notre système. Nous avons d'abord testé notre application sur un émulateur Android qui nous a donner de bon résultats. Nous avons ensuite testé notre application sur deux smartphones en ville et hors-ville, et qui nous a donné d'excellents résultats aussi. Finalement, nous avons abordé les zones blanches dans les testes tout en comparant les différents résultats obtenu après chaque changement du paramètre K.

Conclusion générale et perspectives

Au cours de ce travail, nous avons étudié la problématique autour du tracking des flottes de véhicules en Algérie en minimisant les coûts pour les entreprises, tout en offrant un environnement légale à leurs fonctionnements. Nous avons étudié les marchés des traceurs et des smartphones et la diversité de ces derniers. Nous avons aussi élaboré un modèle qui repose sur plusieurs technologies, suivant un plan économique qui est à la fois peu coûteux et performant. Nous avons essayé d'établir les handicaps majeure auxquels nous feront face pendant le développement, et par la même occasion, nous avons développé quelques mesures contre celle ci ,qui se sont révélés au final, efficaces. Nous avons finalement pu développer, un système de tracking intelligent en utilisant l'algorithme des K-PPV qui permet de déduire le périmètre d'une zone blanche grâce aux types des positions géographiques proches. Ce système permet non seulement de suivre les flottes de véhicules avec précision, mais aussi avec fiabilité grâce à un serveur gouvernementale qui permet de vérifier l'exactitudes des informations présentes dans la base de données du système. Malgré tous ce que nous avons apporté comme technologies pour se système, il est cependant, et comme tout autre trackers, sensible aux zones non couverte par aucun des opérateurs, ces zones sont réputé pour ne contenir aucun réseau GPRS (minimum), et qui causera l'impossibilité de récupérer une position géographique de l'emplacement actuel.

Par ailleurs, de nombreuse améliorations sont possible pour améliorer et développer des nouvelles solutions. Dans l'acquisition des position géographique, l'algorithme de la correction des Waypoints dans Google Maps peut être utilisé pour augmenté la précision du tracking en cas d'interférences du smartphone avec d'autres dispositifs à proximité de ce dernier, ou lorsque le smartphone passe par une zone non couverte ni par la 3G ni par la 2G. Utiliser d'autres algorithmes que les K-PPV pour l'apprentissage et la gestion des zones blanches et ce qui, par conséquences, augmentera considérablement son intelligence et son autonomie. L'exploitation des K-PPV peut être optimiser en enrichissant la base de données avec une base de données hiérarchique spatiale contenant des liens symboliques vers la base principale (base temporelle). Une autre amélioration serait l'extension de ce système sur d'autre système de géolocalisation tel que GLONASS, COMPASS, Galileo ou encore l'IRNSS.

Bibliographie

- [1] COMMENTCAMARCHE.NET : Téléphonie mobile - 3g et 4g expliquées, 2016. disponible sur : <http://www.commentcamarche.net/contents/1123-telephonie-mobile-3g-et-4g-expliquees>.
- [2] Marshall Cavendish CORPORATION : *Inventors and inventions. Vol. 2.* New York : Marshall Cavendish, 2008.
- [3] Agence National de développement de L'INVESTISSEMENT : Secteur de transport, 2005. disponible sur : <http://www.andi.dz/index.php/fr/secteur-de-transport>.
- [4] Algérie ECO : Condor réalise un chiffre d'affaire de 900 millions de dollars, 2017. disponible sur : <http://www.algerie-eco.com/index.php/2017/01/28/condor-realise-chiffre-daffaire-de-900-millions-de-dollars/>.
- [5] Apprendre en LIGNE : Petite histoire de la cryptologie, 2016. disponible sur : <http://www.apprendre-en-ligne.net/crypto/histoire/>.
- [6] Algérie FOCUS : Accidents de la route, 2016. disponible sur : <http://www.algerie-focus.com/2016/04/137769/>.
- [7] Official U.S. Government information about the GLOBAL POSITIONING SYSTEM (GPS) et related TOPICS : Space segment, 2016. disponible sur : <http://www.gps.gov/systems/gps/space/>.
- [8] Global Market INSIGHTS : Vehicle tracking market size, share, 2016. disponible sur : <https://www.gminsights.com/industry-analysis/vehicle-tracking-market>.
- [9] LAROUSSE : Définition cryptographie, 2016. disponible sur : <http://www.larousse.fr/dictionnaires/francais/cryptographie/20864>.
- [10] TSA tout sur L'ALGÉRIE : Ce que l'algérie a réellement réalisé en 10 ans, 2016. disponible sur : <http://bit.ly/1QquDZ9>.
- [11] WIKIPÉDIA : Global positioning system, 2017. disponible sur : https://fr.wikipedia.org/wiki/Global_Positioning_System.
- [12] WIKIPEDIA : k-nearest neighbors algorithm, 2017. disponible sur : https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.
- [13] WIKITIONARY : cryptographie, 2016. disponible sur : <https://fr.wiktionary.org/wiki/cryptographie>.

BIBLIOGRAPHIE
