REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :…… /M2/2017

Mémoire

présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Génie Logiciel et Systèmes Distribués**

Usage du protocole MQTT dans une application de suivi.

Par :

ARADJ TAHA MOHAMED EL-AMINE

Soutenu le …\…\…... , devant le jury composé de :

		Président
Pr.KAHLOUL Laid	Pr	Rapporteur
		Examinateur

Remerciement

Je tiens de tout cœur à remercier mon encadreur monsieur KAHLOUL Laid pour ces directive strictes et objectives et sans lui je le dis haut et fort ce travail n'aura jamais vu le jour.

Je remercie aussi mes parents pour leur soutien, mes amis, et mes collègues à SECURGROUP.

Dédicace

Je dédie ce travail a tout ceux qui se levent tôt le matin pour aller étudier, travailler, réaliser un rêve.

Résumé

L'internet des objets est un domaine très vaste et qui prend une grande ampleur dans la vie quotidienne de l'humain du XXIème siècle. Parmi les domaines où il y a un grand investissement, le domaine de la géolocalisation. Entant qu'informaticiens nous devons optimiser le produit qu'on doit fournir, et la contrainte de temps est très importante c'est pour cela nous avons choisi un protocole de communication rapide et fiable. dans ce projet nous avons choisi le protocl MQTT pour l'envoi et la réception des messages.

Le présent travail consiste à réaliser une application de suivi sous android en utilisant le protocole MQTT. Pour le réaliser nous aurons besoin d'une base de donnée, un service web, un courtier (Broker) MQTT et une application android.

The Internet of Things is a very vast field which is taking on great importance in the daily life of humans in the 21st century. Among the areas where there is a large investment, the field of geolocation. As IT specialists we have to optimize the product we have to provide, and the time constraint is very important, that's why we have chosen a fast and reliable communication protocol. in this project we have chosen the MQTT protocol for sending and receiving messages.

The present work consists in creating a tracking application on android using the MQTT protocol. To achieve this we will need a database, a web service, an MQTT broker and an android application.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

À la période de la rentrée scolaire, l'inquiétude de voir son enfant partir ou rentrer tout seul de l'école où vouloir savoir où il est en permanence sont les causes de nouvelles inquiétudes pour les parents du XXIe siècle, qui oscillent entre autonomie et sécurité de leur enfants. Il faut alors trouver le bon compromis : surveiller l'enfant sans l'oppresser tout en étant rassuré et en sachant où il se trouve.

Pour remédier à cela, bon nombre de sociétés ont mis au point des traceurs GPS capables de vous avertir quand votre enfant est bien rentré ou s'il s'est éloigné d'un parcours prédéfini. La géolocalisation est au cœur de ces techniques embarquées. Elle a commencé avec des porte-clés ou des bipeurs à mettre dans les sacs à dos. Elle se peaufine désormais avec des montres plus élégantes pour les plus jeunes qu'ils pourront porter au poignet. Il faut généralement pour cela compter sur un abonnement mensuel en sus (entre 3,5 et 10 euros) pour pouvoir être averti en cas de souci, déterminer des zones de sécurité et suivre en temps réel l'enfant via son smartphone ou son ordinateur.

Mais Internet est là pour faciliter la vie quotidienne des gents vu que l'informatique et internet surtout deviennent de plus en plus une nécessité de la vie moderne, au fil du temps, on a intégré l'ordinateur dans différents objets de notre vie quotidienne. De plus, avec l'internet, ces objets peuvent se connecter et communiquer entre eux, en créant des possibilités d'intégration plus directe du monde physique dans les systèmes informatiques, et résultant en une meilleure efficacité, la précision et les avantages économiques en plus de réduire l'intervention humaine. Le concept de lier l'objet physique à Intrent est appelé Internet des objets, en l'Anglais *Internet of things*[1]

.

Le présent travail s'intéresse a réaliser une application de suivi des enfant en utilisant le protocol MQTT (Message Queue Telemetry Transport).

Le premier chapitre parlera de l'internet des objets, de la géolocalisation et en fin du protocol que nous avons choisi pour la réalisation de notre projet.

Le deuxiemme chapitre nous allons etudier la faisabilité du projet et nous allons faire la conception de ce dernier.

Le troisiemme chapitre et le dernier nous allons parler de la réalisation et l'implémentation de notre application et nous allons terminer par une conclusion générale.

1. Nous utilisons tout au long de notre mémoire l'abréviation IoT pour « Internet of Things », qui se traduit en français par l'Internet des objets

Chapitre 1

Etat de l'art

Introduction

De nos jours l'informatique prend une grande ampleur dans la vie quotidienne de chaque individu pour donner une où plusieurs solutions aux problèmes de l'être humain. Le suivi est un des domaines ou le monde a investi beaucoup d'énergie et de ressources pour satisfaire le besoin de savoir où se trouve réellement son bien ou son proche. Sur ce l'informatique a donné plusieurs solutions via des gadgets et des technologies qui ont prouvé leur fiabilité grâce a des algorithmes et des protocoles de communications et une grande concurrence entre les différentes entreprises de développement et tous ça pour offrir un large choix aux clients selon leur budget.

Dans ce chapitre nous allons parler de l'internet des objets puis nous parlerons de la géolocalisation et le protocole de communication MQTT et en finir avec une conclusion qui répondra a la question pourquoi avons-nous choisi ce protocole ?.

1.1 Internet des Objets

L'internet des objets et ses protocoles sont parmi les sujets les mieux financés dans l'industrie et étudiés dans le milieu universitaire. L'évolution rapide de l'Internet mobile, la fabrication des composants éléctronique, micro-informatique, et la machine à machine (M2M) [1] ont permis aux technologies IoT d'être au sommet de sujets médiatiques, ce qui implique qu'une grande quantité d'argent qui sera investi par l'industrie et la recherche en plus devrait venir dans les prochaines années [6].

Les technologies IoT permettent à des choses ou des appareils qui ne sont pas des ordinateurs, d'agir intelligemment et de prendre des décisions de collaboration qui sont bénéfiques pour certaines applications.

Nous présentons dans cette parie des protocoles IoT fonctionnant sur de différentes couches de la pile réseau offerts par des organismes de normalisation. Ces normes ont été proposées au cours de la dernière demi-décennie pour répondre aux besoins actuels et aux besoins futurs de l'IoT ansi que les avantages qui ont a offrir et les défis a relever.

1.1.1 Avantages

Les avantages de « IoT »[15] s'étendent à tous les domaines du mode de vie et des affaires. Voici une liste de certains des avantages que l'IoT a à offrir :

1. **Amélioration de l'engagement des clients :** Les analyses actuelles souffrent de failles importantes dans l'exactitude ; Et comme indiqué, l'engagement demeure passif. IoT aide pour atteindre un engagement plus riche et plus efficace avec le public.

1. Le "M2M", diminutif de Machine à Machine, désigne l'ensemble des Solutions et Technologies permettant à des outils, des machines, des automates, des systèmes, de communiquer entre eux de manière automatique.

2. **Optimisation de la technologie :** Les mêmes technologies et données qui améliorent l'expérience client améliorent également l'utilisation des périphériques et aident à améliorer la technologie. L'IoT déverrouille un monde de données fonctionnelles et de terrain critiques.
3. **Réduction des déchets :** L'IoT rend les zones d'amélioration claires. Les analyses actuelles nous donnent un aperçu superficiel, mais l'IoT fournit des informations réelles conduisant à une gestion plus efficace des ressources.
4. **Amélioration de la collecte de données :** La collecte de données moderne souffre de ses limites et de sa conception pour une utilisation passive. IoT le brise hors de ces espaces, et les place exactement où les humains veulent vraiment aller pour analyser notre monde. Il permet de donner une image précise de tout.

1.1.2 Défis de l'Internet des Objets(IoT)

Le développement d'une application réussie IoT n'est toujours pas une tâche facile en raison de multiples défis [17]. Ces défis comprennent : la mobilité, la fiabilité, l'évolutivité, la gestion, la disponibilité, l'interopérabilité, la sécurité et la vie privée. Dans ce qui suit, nous décrivons brièvement chacun de ces défis.

1. **La mobilité :** Les périphériques IoT doivent se déplacer librement et changer leur adresse IP et leurs réseaux en fonction de leur emplacement. De plus, la mobilité peut entraîner un changement de fournisseur de services qui peut ajouter une autre couche de complexité en raison de l'interruption du service et de la modification de la passerelle.
2. **La fiabilité :** Le système devrait fonctionner parfaitement et fournir toutes ses spécifications correctement. C'est une exigence très critique dans les applications qui nécessite des réponses d'urgence. Dans les applications IoT, le système doit être très fiable et rapide dans la collecte des données, de les communiquer et de prendre des décisions et éventuellement les mauvaises décisions peuvent conduire à des scénarios désastreux.
3. **L'évolutivité :** L'évolutivité est un autre défi des applications IoT où de millions de périphériques pourraient être connectés sur le même réseau. Gérer leur distribution n'est pas une tâche facile. En outre, les applications IoT doivent être tolérantes à de nouveaux services et périphériques qui se connectent constamment au réseau et, par conséquent, doivent être conçus pour permettre des services et des opérations extensibles.
4. **La gestion :** La gestion de tous ces périphériques et le suivi des défaillances, des configurations et des performances de ce grand nombre de périphériques est certainement un défi dans IoT. Les fournisseurs doivent gérer les défauts, la configuration, la comptabilité, la performance et la sécurité de leurs périphériques interconnectés et tenir compte de chaque aspect.
5. **La disponibilité :** La disponibilité d'IoT inclut des niveaux de logiciel et de matériel fournis à tout moment et n'importe où pour les abonnés de service. La disponibilité du logiciel signifie que le service est fourni à toute personne autorisée à l'avoir. La disponibilité matérielle signifie que les périphériques existants sont faciles d'accès et sont compatibles avec la fonctionnalité IoT et les protocoles. En

outre ces protocoles doivent être compacts pour être en mesure d'être intégrés dans les dispositifs IOT.

6. **L'interopérabilité :** L'interopérabilité signifie que les dispositifs et les protocoles hétérogènes doivent pouvoir interagir les uns avec les autres. Ceci est difficile en raison du grand nombre de plates-formes différentes utilisées dans les systèmes IoT. L'interopérabilité doit être gérée par les développeurs d'applications et les fabricants de périphériques afin de fournir les services indépendamment de la plate-forme ou des spécifications matérielles utilisées par le client, ce qui nous mène au point suivant la technologie.
7. **La technologie :** l'Internet des Objets s'appuie sur une variété de technologies et de protocoles existants ou à définir qui, combinés ensemble ouvre la porte à de nouvelles et intéressantes perspectives. Pour illustrer les aspects technologies, on peut présenter la figure suivante : D'un point de vue de l'infrastructure de communi-

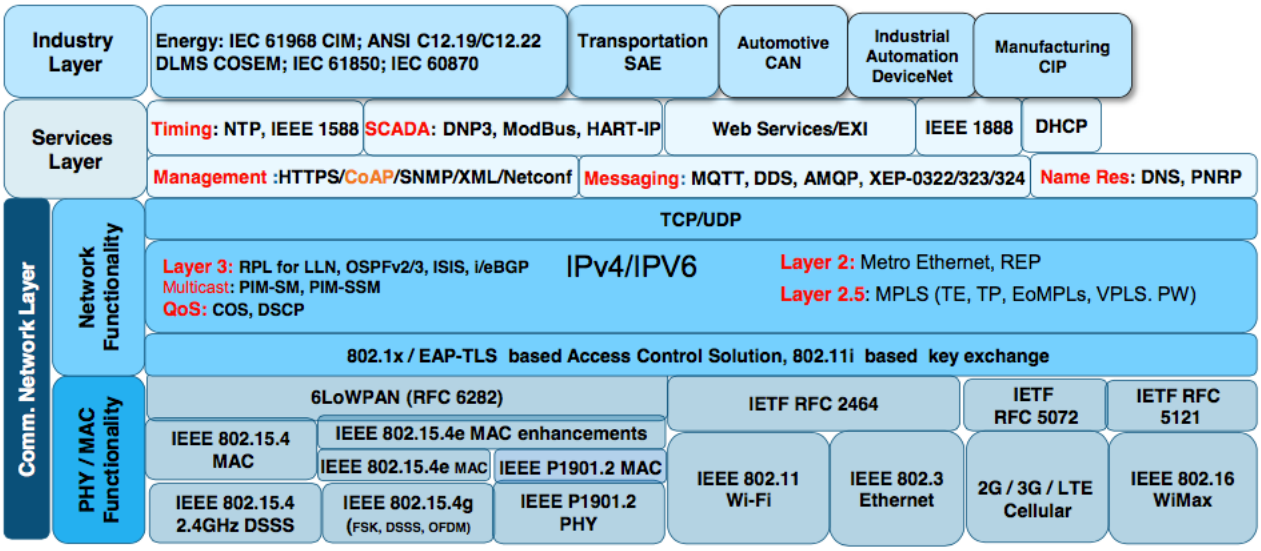


FIGURE 1.1 – IOT d'un point de vue technique [1]

cation, on trouve : les technologies d'accès au réseau et le routage des informations.

Technologies d'Accès : On peut distinguer deux types de technologies d'accès :

- **accès filaires :** Essentiellement pour l'industrie avec une adaptation de l'Ethernet pour ces besoins spécifiques.
- **accès sans fils :** via les technologies traditionnelles (Wi-Fi, 3G/4G, Bluetooth, etc.)).

8. **Routage :** Il existe de nouvelles techniques de compression pour le transport d'IPv6 sur ces nouvelles technologies sans fils. Avec le RFC 6550, l'IETF a défini un nouveau protocole de routage adapté pour ces réseaux dits « Low Power & Lossy Networks (LLN) ». Ce protocole prend en compte de nouveaux indicateurs pour définir la métrique, comme par exemple, la fiabilité des liens ou le niveau de batterie des nœuds.

9. **Sécurité des périphériques et services IoT [7]** assurer la sécurité implique la protection des dispositifs IoT et des services de l'accès non autorisé à partir de l'intérieur des dispositifs et à l'extérieur. La sécurité doit protéger les services, les ressources matérielles, les informations et les données, en transition et en stockage. On peut identifier les problèmes clés liés aux périphériques et services IoT suivant :
   - **La confidentialité des données :** représente un problème fondamental dans les dispositifs et les services IoT. Dans le contexte IoT, non seulement l'utilisateur peut accéder aux données mais également à l'objet autorisé. Cela nécessite d'aborder deux aspects importants : d'abord, le mécanisme de contrôle d'accès et d'autorisation et le second mécanisme d'authentification et de gestion d'identité. Le périphérique IoT doit pouvoir vérifier que l'entité (personne ou autre appareil) est autorisée à accéder au service.
   - **L'autorisation :** permet de déterminer si, lors de l'identification, la personne ou l'appareil est autorisé à recevoir un service. Le contrôle d'accès consiste à contrôler l'accès aux ressources en accordant ou en refusant des moyens en utilisant un large éventail de critères. L'autorisation et le contrôle d'accès sont importants pour établir une connexion sécurisée entre un certain nombre d'appareils et de services. La principale question à traiter dans ce scénario est de rendre les règles de contrôle d'accès plus faciles à créer, à comprendre et à manipuler.

     Un autre aspect qui doit être pris en considération lors de la gestion de la confidentialité est l'authentification et la gestion de l'identité. En fait, cette question est critique dans l'IoT, parce que plusieurs utilisateurs, objets / choses et périphériques doivent s'authentifier mutuellement à travers des services fiables. Le problème est de trouver une solution pour gérer l'identité de l'utilisateur, des choses / objets et des dispositifs d'une manière sécurisée.
   - **La disponibilité :** un utilisateur d'un périphérique (ou le périphérique lui-même) doit pouvoir accéder aux services à tout moment, chaque fois que nécessaire. Différents composants matériels et logiciels dans les dispositifs IoT doivent être robustes de manière à fournir des services même en présence d'entités malveillantes ou de situations défavorables. Différents systèmes ont des exigences de disponibilité différentes. Par exemple, les systèmes de surveillance des incendies ou de surveillance des soins de santé auraient probablement des exigences de disponibilité plus élevées que les capteurs de pollution routière.
   - **La non-répudiation :** la propriété de la non-répudiation produit certaines preuves dans les cas où l'utilisateur ou l'appareil ne peut pas refuser une action. La non-répudiation n'est pas considérée comme une propriété de sécurité importante pour la plupart des IoT. Il peut être applicable dans certains contextes, par exemple, les systèmes de paiement où les utilisateurs ou les fournisseurs ne peuvent pas refuser une action de paiement.

1.1.3 Les Normes et la standardisation

Comme les dispositifs IoT continuent de saturer la société, la standardisation est essentielle pour atteindre des spécifications universellement acceptées et des protocoles pour une véritable interopérabilité entre les dispositifs IoT et les applications. Les normes pu-

bliées aujourd'hui marquent une étape majeure pour l'Internet des objets en offrant la proposition de valeur unique d'une seule plate-forme d'interfonctionnement pour tous les appareils activés [10].

Près de 140 organismes dans le monde sont aujourd'hui concernés, directement ou indirectement, par la normalisation de la communication M2M. Cette phase de normalisation représente en effet l'un des facteurs cruciaux de l'évolution de l'Internet mobile vers l'Internet des Objets. Il existe des milliers de standards « spécifiques » à des contextes particuliers de l'IoT, parmi eux et en particulier ceux qui sont d'ores et déjà utilisés par l'industrie nous trouvant ceux offerts par Internet Engineering Task Force **(IETF)**, Institut des ingénieurs électriciens et électroniciens **(IEEE)**, Union internationale des télécommunications **(UIT)** et Global Standard1 **(GS1)**, Organization for the Advancement of Structured Information Standards **(OASIS)**, comme illustré dans le tableau 1.1 suivant :

ÉMETTEUR	NORME	DÉFINITION
UIT	UIT-T Y.2060	Concept IoT
	UIT-T Y.2061	Interface machine-application
IEEE	IEEE 802.15.4	Couche liaison
IETF	6LoWPAN	IPv6 over Low Wireless Personal Area Networks
	CoAP	Constrained Application Protocol
	RPL	IPv6 Routing Protocol for Low-Power and Lossy Networks
GSI	ONS	Object Naming Service
	EPC	Electronic Product Code
IETF	MQTT	Message Queue Telemetry Transport
	ADMQP	Advanced Message Queuing Protocol
	DDS	Data Diffusion Service

TABLE 1.1 – Organisme de normalisation de protocole IOT [10]

1. **UIT :** les deux recommandations, **l'UIT-T Y.2060** qui fournit une vue générale du concept de l'IoT et **l'UIT-T Y.2061** qui décrit les conditions relatives à l'interface machine orientée vers les applications de communications dans l'environnement NGN (réseaux de nouvelle génération).
2. **IEEE et l'IETF** dans le domaine des réseaux de capteurs sur protocole IP. Ces efforts se sont d'abord concrétisés par la proposition d'un modèle de couches sur le modèle OSI ainsi que de protocoles plus adaptés aux réseaux industriels que le modèle TCP/IP sur Ethernet. On trouve notamment :
   — au niveau de la couche de Liaison, le standard **IEEE 802.15.4**[14] plus adapté que l'Ethernet aux environnements industriels difficiles
   — au niveau Réseau, le standard **6LoWPAN** [12] (IPv6 over Low Power Wireless Personal Area Networks), qui a réussi à adapter le protocole IPv6 aux communications sans fil entre nœuds à très faible consommation. Ainsi 6LowPAN, est une couche d'adaptation qui permet de transporter des paquets IPv6 sur des liens 802.15.4. Sans 6LowPAN IPv6, les protocoles Internet ne fonctionneraient

pas dans ces réseaux personnels sans fil à faible consommation d'énergie qui utilise IEEE 802.15.4.
   - En ce qui concerne le routage **l'IETF** a publié en 2011 le standard **RPL** (IPv6 Routing Protocol for Low-Power and Lossy Networks).
   - au niveau Application le protocole **COAP** (Constrained Application Protocol) qui tente d'adapter **Http** , (beaucoup trop d'échange), aux contraintes des communications entre nœuds à faible consommation, ainsi que **(XMPP)** est un protocole de messagerie qui a été conçu à l'origine pour les applications de chat et d'échange de messages il a été standardisé par **l'IETF**.
3. **L'organisme GS1** : a proposé le système **EPC** (Electronic Product Code) qui est un identifiant individuel unique permettant d'identifier un produit électronique ainsi que l'architecture **EPC global Network** qui définit l'organisation des systèmes d'informations destinés à assurer l'échange des informations **EPC** au niveau global. L'un de ses principaux composants, **l'ONS** (Object Naming Service), est directement fondé sur le DNS (Domain Name System).
4. **L'OASIS** [9] : qui est un consortium sans but lucratif qui oriente les développements et l'adoption de standards «ouverts» pour la société de l'information. Les travaux de ce consortium sur l'internet des objets portent sur les technologies de réseau et de messagerie normalisées comme Message Queue Telemetry Transport **(MQTT)**, Advanced Message Queuing Protocol **(AMQP)**, ainsi que le service de diffusion de données **(DDS)**. Ces protocoles se situent à la couche application, le tableau 1.2 montre une comparaison entre ces protocole.

Protocoles	**UDP / TCP**	**Architecture**	**Sécurité et qualité de service**	**Taille en-tête (octets)**	**Max Longueur (octets)**
MQTT	TCP	Pub / Sub	Tous les deux	2	5
AMQP	TCP	Pub / Sub	Tous les deux	8	-
CoAP	UDP	Req / Res	Tous les deux	4	20 (typique)
XMPP	TCP	Tous les deux	Sécurité	-	-
DDS	TCP / UDP	Pub / Sub	QoS	-	-

Table 1.2 – Comparaison entre des protocoles IoT [17]

1.1.4 Protocoles de messagerie IoT

D'après une ressente étude publieé dans **IEEE** [11], sur les performances web pour les protocoles IoT, un ensemble de test est mené pour comparer les temps de latence et le débit des messages des protocoles de messagerie IoT, cette étude a montré que :

- Dans le premier cas de test, ils ont mesuré la latence du transfert de messages d'un éditeur à un abonné. Les temps mesurés sont indiqués dans le tableau 1.3 et la Figure 1.2. La latence la plus courte est produite par le protocole MQTT, suivi par AMQP, alors que la différence entre XMPP et DDS est négligeable. Pour le cas de test d'un seul nœud capteur, le MQTT a obtenu une latence de seulement 2,53 ms. Les latences augmentent proportionnellement avec l'augmentation de la taille des messages.

Protocol	nummber of Sensors	1 sensor Node	2 sensor Node	3 sensor Node	4 sensor Node	5 sensor Node
	Message Playload	1475-1477	2896-2901	4316-4321	5734-5746	7157-7169
MQTT	*Transfer*[ms]	**2.53**	**3.83**	**3.56**	**3.84**	**4.33**
AMQP	*Transfer*[ms]	3.99	4.25	4.61	4.82	4.93
XMPP	*Transfer*[ms]	4.11	4.97	5.28	5.89	6.34
DDS	*Transfer*[ms]	4.3	5.04	5.48	5.56	5.72

TABLE 1.3 – Mesure la latence du transfert de messages des protocoles de messagerie IoT [11]

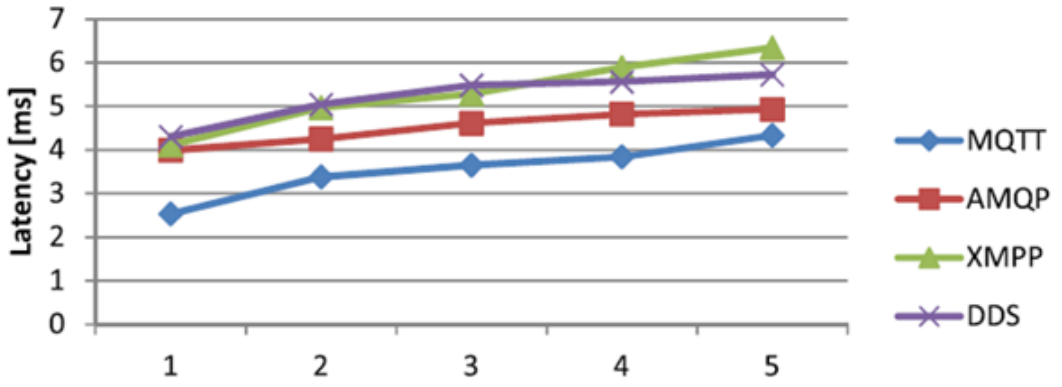


FIGURE 1.2 – Mesure de latence des protocoles de messagerie d'application web IoT [11]

— Afin de minimiser les erreurs en raison de la résolution de la minuterie, ainsi que de mesurer la capacité maximale de chaque protocole, ils ont effectué un autre test dans lequel ils ont mesuré le nombre de messages transférés aller-retour, avec le scénario suivant :
L'éditeur publie un message et à sa réception, l'abonné renvoie immédiatement ce message à l'éditeur. Le processus se poursuit car l'éditeur renvoie également immédiatement le même message à l'abonné, etc. Les résultats montrent le nombre de messages aller-retour passés dans l'intervalle de 100s Ils ont obtenu les résultats qui sont montrés dans le tableau 1.4 et la figure 1.3 .

Ces tests ont en effet des résultats très différents. Dans le cas des protocoles AMQP et XMPP, les valeurs de débit sont alignées sur le test de latence : AMQP atteint un débit de message légèrement supérieur à XMPP, 229.38 msg/s à 187.87 msg/s, et les meilleurs résultats ont été mesurés immédiatement après que le broker de messages a commencé.

Pour le protocole MQTT, le taux de débit des messages dans le cas de test des clients Java est nettement plus élevé que les protocoles AMQP et XMPP avec 302,48 msg/s, comparativement à 266,97 msg/s.

Protocol	Java – JavaScript clients		Java – Java clients		JavaScript – JavaScript clients	
	Message Throughput Rate [msg/s]	*Ttransmission* [ms]	Message Throughput Rate [msg/s]	*Ttransmission* [ms]	Message Throughput Rate [msg/s]	*Ttransmission* [ms]
MQTT(Mosquitto)	9.85	50,76	302.48	1,65	3,33	150,15
MQTT(HiveMQ)	177.37	2,82	194.60	2,57	**354,64**	**1,41**
AMQP	**229.38**	**2,18**	266.97	1,87	208,97	2,39
XMPP	187.85	2,66	196.04	2,55	195,63	2,56
DDS	X	X	**463.94**	**1,08**	181,88	2,75

TABLE 1.4 – Les mesures de taux de débit des messages des protocoles de messagerie d'application web IOT [11]

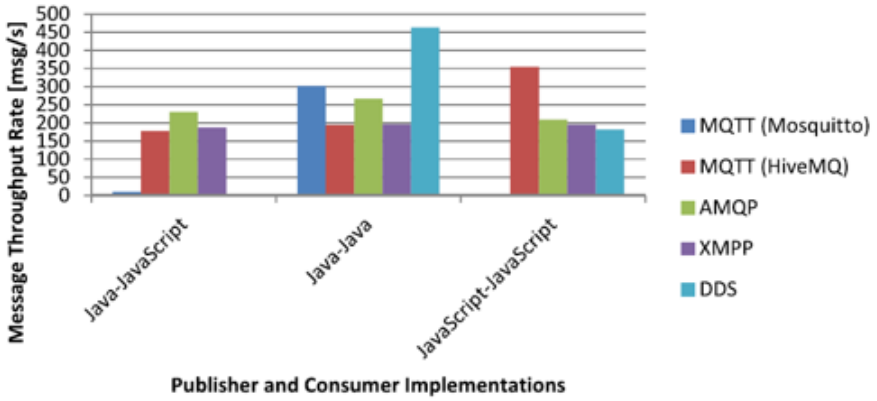


FIGURE 1.3 – Les mesures de taux de débit des messages des protocoles de messagerie d'application web IOT [11]

Le taux de rendement de message des clients JavaScript <-> JavaScript est le plus haut de tous les protocoles éprouvés, atteignant 354.60 Msg/s. Cependant, dans le test de latence, la configuration basée sur HiveMQ [2] montrait une latence sensiblement plus élevée que le test basé sur le courtier Mosquitto [3]. Le deuxième résultat inattendu est un taux de transmission de messages de plus de 50 pour le protocole DDS que pour le deuxième meilleur protocole, qui est MQTT, dans la configuration avec des clients Java Java, 463.94 msg/s à 302.48 msg/s.

L'un des aspects essentiels d'une architecture IoT est celui de la communication entre les divers composants du système global. Il existe bien sûr de multiples moyens de connecter et de faire communiquer les composants en question, mais on assiste dans ce domaine à l'émergence de certains protocoles mieux adaptés que d'autres pour remplir ce besoin. Ainsi, avec son modèle Publish/Subscribe et sa très grande légèreté, le protocole MQTT est actuellement considéré comme l'un des candidats les plus sérieux pour assurer le

2. HiveMQ est le courtier MQTT pour l'entreprise connectée. http ://www.hivemq.com/

3. Mosquitto est un iot.eclipse.org projet, est un open source (EPL / EDL autorisé) courtier de messages qui implémente le protocole MQTT versions 3.1 et 3.1.1.

transport des données au sein des architectures IoT. On en trouve de nombreuses implémentations dans divers langages de programmation, dont beaucoup sont open-source.

1.2 La géolocalisation

Ce chapitre présentera les différentes techniques mises en jeux dans la localisation d'un terminal. Les techniques de localisation sont de diverses sortes, elles peuvent être divisées en techniques basées sur le réseau, techniques utilisant des dispositifs intelligents aux endroits fixes et enfin, des techniques basées sur le GPS (voir la figure 1.4).

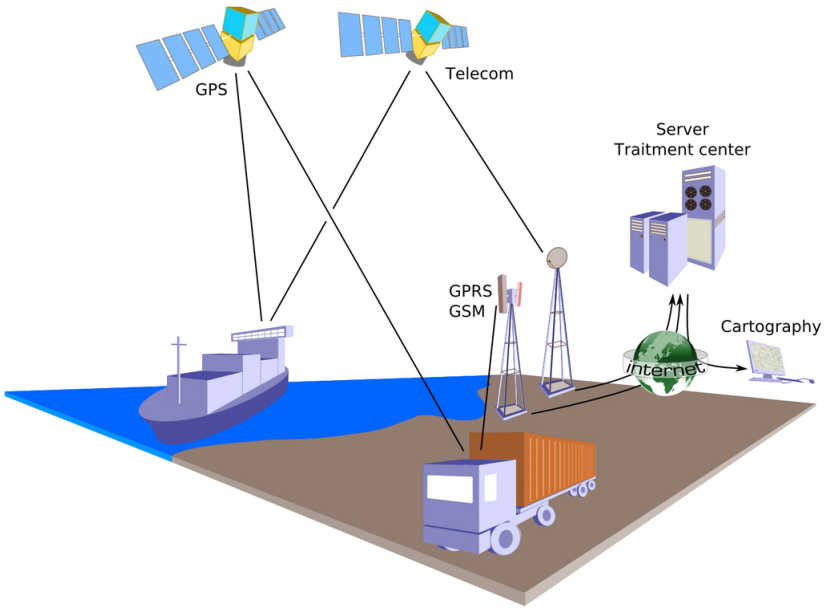


FIGURE 1.4 – Les techinique de géolocalisation [2]

.

1.2.1 Géolocalisation par Satellites

Elle consiste à calculer, grâce aux signaux émis par une constellation de satellites prévue à cet effet, la position actuelle sur la face terrestre d'un terminal équipé d'une puce compatible. Cette position est alors traduite en termes de latitude, longitude et parfois altitude et peut alors être représentée physiquement sur une carte. Le réseau satellite de positionnement le plus connu est le GPS [4].

1.2.2 Géolocalisation par GPS

Le GPS se base sur une constellation de 24 satellites, qui émettent en permanence un signal daté, et un réseau de stations au sol qui surveillent et gèrent ces satellites. La localisation est possible dès lors que quatre satellites sont visibles : il y a en effet quatre inconnues à déterminer, les trois coordonnées spatiales, ainsi que le temps, puisque le récepteur au sol n'est pas synchronisé avec les satellites. Pour ce faire, les 24 satellites du

4. Le Global Positioning System (GPS) (en français : « Système mondial de positionnement » [littéralement] ou « Géo-positionnement par satellite »), originellement connu sous le nom de Navstar GPS, est un système de positionnement par satellites appartenant au gouvernement des États-Unis. Mis en place par le département de la Défense des États-Unis à des fins militaires à partir de 1973, le système avec vingt-quatre satellites est totalement opérationnel en 1995 et s'ouvre au civil en 2000.

système sont équi-répartis sur six orbites de façon à garantir qu'au moins quatre satellites soient visibles en permanence et ce, partout sur la Terre. A partir de trois satellites, un tel récepteur est capable d'effectuer une triangulation pour déterminer sa position. Cette position est déterminée instantanément d'où la possibilité de poursuivre des cibles mobiles. Chaque mesure représente le rayon R d'une sphère centrée sur un satellite particulier.
Le récepteur GPS est sur cette sphère. Avec trois mesures, donc trois satellites, la position du récepteur se réduit à l'intersection de deux points dont l'un est très éloignée dans l'espace (voir la figure 1.5).

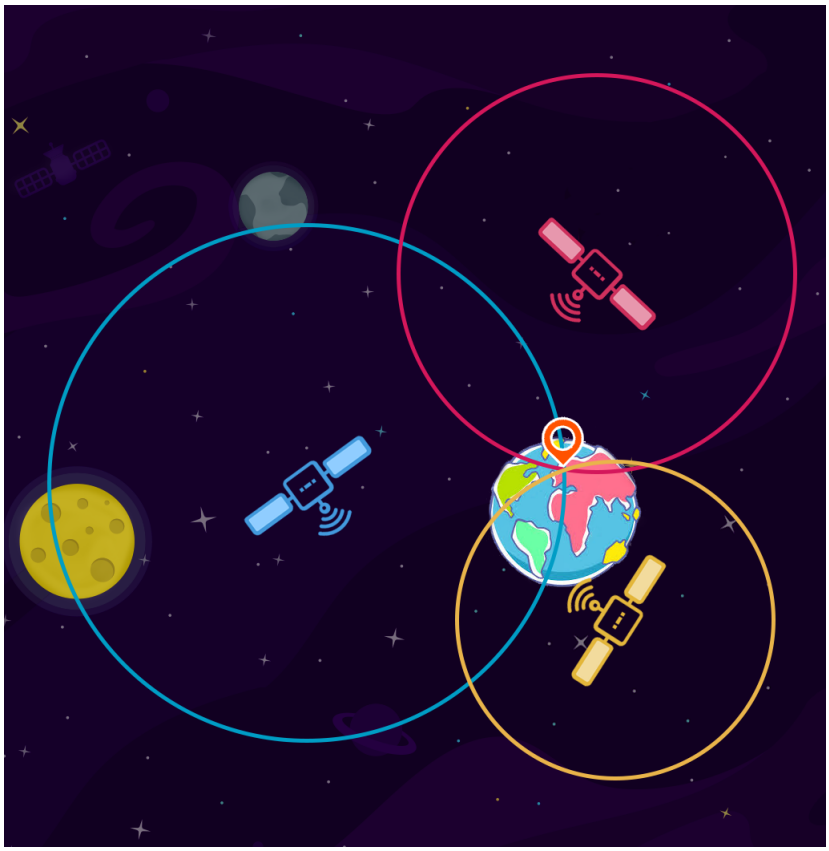


FIGURE 1.5 – Localisation du récepteur par trois satellites [3]

1.2.3 Géolocalisation par Galileo

L'architecture du futur système de positionnement par satellites européen se base sur une constellation de 30 satellites (27 actifs et 3 de secours) placés sur trois orbites circulaires d'altitude moyenne (24000 Km), des stations au sol, de centres de contrôle et des utilisateurs dotés de récepteurs mobiles.
Le principe de fonctionnement est simple : les satellites de la constellation sont équipés d'une horloge atomique mesurant le temps avec une extrême précision. Ils émettent des signaux personnalisés indiquant leur heure de départ du satellite. Le récepteur au sol, intégré par exemple dans un téléphone portable, possède pour sa part en mémoire les coordonnées précises des orbites de tous les satellites de la constellation. Il peut ainsi en lisant le signal qui arrive reconnaître le satellite émetteur, déterminer le temps mis par le

signal pour arriver jusqu'à lui et donc calculer la distance qui le sépare du satellite. Dès qu'un récepteur au sol reçoit les signaux d'au moins quatre satellites simultanément, il peut calculer sa position la plus exacte possible (précision variant de 10 à 1 m).

— Six pour les services gratuits.
— Deux pour le service commercial.
— Deux pour le service public réglementé.

1.2.4 Géolocalisation par GSM

plusieurs kilomètres, selon si le terminal se trouve en milieu urbain (où la densité d'antennes est supérieure), ou en milieu rural.
Plusieurs techniques existent. Aujourd'hui, la méthode GSM la plus utilisée est celle du Cell ID. Cette méthode consiste à récupérer les identifiants des antennes GSM auxquelles le terminal est connecté. Par la suite, grâce à une base de données faisant le lien entre les identifiants des cellules et les positions géographiques des antennes, le terminal est capable de déterminer sa position et d'émettre une estimation.
Étant donné que les bases de données Cell ID ne sont pas stockées localement dans le terminal, une connexion internet de type GPRS/EDGE ou 3G peut être nécessaire afin d'émettre une requête pour obtenir la correspondance Cell ID / Longitude Latitude.

1.2.5 Géolocalisation par WiFi

De la même façon qu'un terminal GSM peut se localiser par la méthode du Cell ID sur un réseau GSM, un terminal WiFi peut utiliser la même méthode en se basant sur les identifiants des bornes WiFi (Adresses MAC) qu'il détecte. Il existe des bases de données recensant une multitude de bornes d'accès WiFi ainsi que leur position géographique.

1.2.6 Géolocalisation par adresse IP (sur internet)

Cette méthode permet de déterminer la position géographique d'un ordinateur ou de n'importe quel terminal connecté à internet en se basant sur son adresse IP. Les adresses IP sont gérées par l'IANA, une organisation qui s'occupe de découper les blocs d'adresses IP disponibles et de les distribuer de façon très contrôlée aux pays qui en demandent. Toutes ces attributions étant très bien documentées, il est possible de savoir dans quel pays se trouve un terminal connecté à internet grâce à son adresse IP. On peut même obtenir un niveau de précision de l'ordre de la ville en se basant sur la distribution des adresses IP faite par les fournisseurs d'accès à internet.

1.2.7 Géolocalisation par RFID

La technologie RFID peut être utilisée pour la géolocalisation indoor. Pour ce faire, une série de lecteurs RFID équipés de différents types d'antennes sont positionnés de façon à couvrir l'ensemble de la zone souhaitée. La zone est alors découpée en cases dont la surface varie en fonction du nombre de lecteurs déployés et de leur puissance. Lorsqu'une personne équipée d'un tag RFID actif sera dans ces zones là, le système sera capable de

calculer sa position en se basant sur le nombre de lecteurs qui détectent le tag et de déduire la position approximative de l'individu en se référant au schéma de découpage établi.

Chacune de ces techniques se caractérise par le degré de précision de ses informations, la consommation d'énergie qu'elle engendre ou encore les conditions qui doivent être réunies pour assurer son fonctionnement (certaines capacités matérielles comme la présence d'une puce GPS, un accès à internet, tag RFID...).
Bien entendu, pour notre cas, notre technique de localisation est basée sur le GPS vu ses avantages qui font de lui un choix approprié comme technique de localisation pour randonneurs.

1.3 Le Protocole MQTT

MQTT (Message Queue Telemetry Transport) est un protocole M2M (Machine-to-Machine) open source [8] . C'est un protocole de messagerie basé sur le publish/subscrib à la fois extrêmement simple et léger idéal pour l'Internet des objets. Il a été inventé par Andy Stanford-Clark (IBM) et Arlen Nipper (Arcom, maintenant Cirrus Link) en 1999, lorsque leur cas d'utilisation était de créer un protocole avec les objectifs suivants [16] :

1. Simple à mettre en œuvre.
2. Fournir une qualité de service de livraison de données.
3. Léger et faible consommation de la bande passante.

Par la suite, la version 3.1 a été libérée de droits en 2010, et le protocole est standardisé par l'OASIS en 2014 dans sa **version 3.1.1**.[18] Il devient un standard ISO (ISO/IEC 20922) en 2016 [3].

Les qualités de MQTT font de lui un excellent candidat pour les communications au sein de l'Internet des Objets [19], mais il a également été utilisé avec succès par Facebook pour être au cœur de leur système de messagerie à cause de sa performance [19], comme il est utilisé par amazon web services dans ses passerelles pour permettre la diffusion des données à plusieurs abonnés sur un sujet donné [13].

La norme MQTT décrit le comportement attendu des brokers et clients MQTT, ainsi que les détails du format binaire employé pour les échanges.

La suite de ce chapitre synthétisera l'essentiel de cette norme afin de mieux appréhender la question de la normalisation par la suite. Nous expliquerons en détail le fonctionnement du protocole MQTT. La partie suivante sera consacrée au format binaire défini par la norme pour les paquets MQTT.

1.3.1 Mode de fonctionnement

MQTT est un protocole de publication et de souscription de message. Les clients ne communiquent pas directement entre eux, ils publient des messages sur un broker (appelé aussi courtier), les messages sont composés d'un contenu et d'un sujet (topic) [13]. Le broker garde en mémoire le dernier message pour chaque sujet. Les clients qui sont intéressés par les messages d'un sujet peuvent les récupérer en se connectant au broker. L'avantage de cette solution est qu'elle permet à plusieurs clients de communiquer même s'ils ne sont jamais connectés en même temps au broker.

Ce patterne consiste à organiser les messages par sujets (topics) et à gérer leur distribution selon le principe d'abonnement. Ainsi, toute application peut publier ses messages sur les topics de son choix, tandis que les applications intéressées par les messages d'un topic donné peuvent s'abonner pour recevoir tous les nouveaux messages publiés sur ce topic.

1.3.2 Topologies MQTT

Le standard ne définit pas de topologie particulière d'utilisation, la plus simple étant celle avec un seul broker et un certain nombre de clients. D'autres organisations restent évidement possible, avec en particulier les systèmes à base de bridge dans lesquels plusieurs brokers sont utilisés, chacun étant client des autres brokers, comme sur la Figure 1.6.

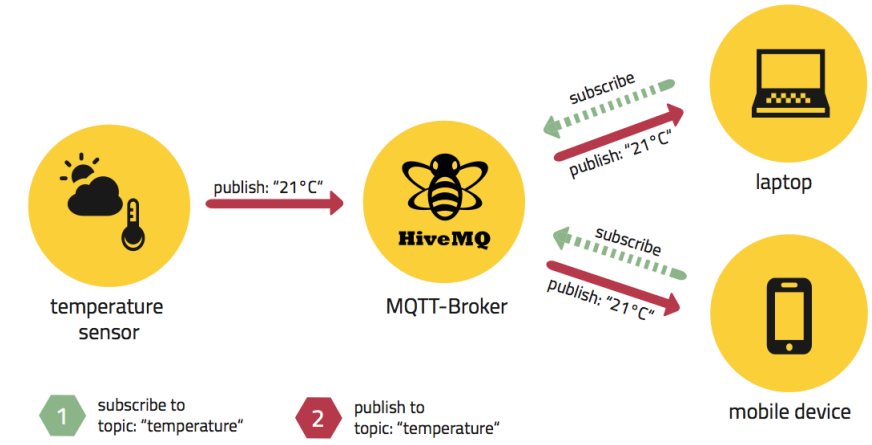


FIGURE 1.6 – Exemples de topologies MQTT [5]

1.3.3 Fonctionnement

Connexion et Déconnexion

MQTT se sert de connexions persistantes entre les clients et le broker, et exploite pour cela les protocoles de réseau garantissant un bon niveau de fiabilité comme TCP.

Avant de pouvoir envoyer des commandes de contrôle, un client doit au préalable **s'enregistrer** auprès du broker, ce qui se fait avec la commande **CONNECT**.

Divers **paramètres** de connexion peuvent alors être échangés, comme les identifiants du client ou encore le mode de persistance souhaité. Le broker doit confirmer au client que son **inscription** a bien été prise en compte, soit indiquer qu'une erreur est survenue en renvoyant un **CONNACK** accompagné d'un code de retour.

Il existe une commande **PINGREQ** permettant de faire savoir au broker que le client est toujours actif, le broker répondra de son côté avec un **PINGRESP** pour indiquer au client que la connexion est toujours active.

Lorsque le client veut se déconnecter, il envoie au préalable une commande DISCONNECT au broker pour lui faire part de son intention. Dans le cas contraire, le broker considérera la déconnexion comme anormale et agira en conséquence, en envoyant le message de volonté WILL au nom du client à tous les abonnés. Un message que Le client peut spécifier et conservera l'indicateur dans la charge utile de message CONNECT.

Abonnements et Publications

Chaque message publié est nécessairement associé à un sujet, ce qui permet sa distribution aux abonnés. Les sujets peuvent être organisés en hiérarchie arborescente, ainsi les abonnements peuvent porter sur des motifs de filtrage que nous détaillerons plus loin. La gestion des abonnements est très simple et consiste en trois commandes essentielles :

1. **SUBSCRIBE** Permet à un client de s'abonner à un topic , une fois abonné il recevra par la suite toutes les publications concernant ce sujet. Un abonnement définit également un niveau de qualité de service, dont il sera question plus bas . La

bonne réception de cette commande est confirmée par le broker par un **SUBACK** portant le même identifiant de paquet.

2. **UNSUBSCRIBE** Donne la possibilité d'annuler un abonnement, et ainsi ne plus recevoir les publications ultérieures. La bonne réception de cette commande est confirmée par le broker par un **UNSUBACK** portant le même identifiant de paquet.
3. **PUBLISH** Initié par un client, permet de publier un message qui sera transmis par le broker aux abonnés éventuels. La même commande sera envoyée par le broker aux abonnés pour délivrer le message.

Topic et motifs de filtrage

Un sujet est une chaîne UTF-8, qui est utilisé par le broker pour filtrer les messages pour chaque client connecté. Un sujet est constitué d'un ou plusieurs niveaux de sujet. Chaque niveau de sujet est séparé par une barre oblique. Voici quelques exemples de sujets :

sport/tennis/player1/score/wimbledon
myhome / RdC / living / température

A noter que chaque sujet doit avoir au moins 1 caractères pour être valide et il peut également contenir des espaces. Aussi un sujet est sensible à la casse, ce qui rend myhome / température et MyHome / Température deux sujets individuels. En outre, la barre oblique seule est un sujet valable. Un client peut souscrire à plus de sujets à la fois comme illustré dans la figure qui suit :

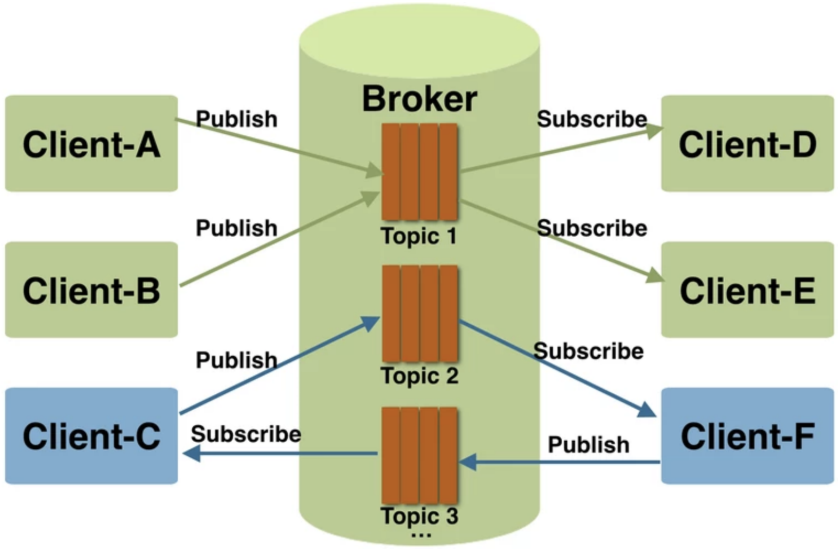


FIGURE 1.7 – Souscription au topic [4]

Afin de proposer un système de filtrage efficace sur les sujets, il est possible de définir une arborescence à l'aide du séparateur /.
Deux jokers sont réservés pour représenter un et plusieurs niveaux d'arborescence :

— + représente un niveau d'arborescence, ainsi T1/T2/T3 peut être mis en correspondance avec divers filtres tels que T1/T2/+, T1/+/T3 ou T1/+/+.
— # représente autant de niveaux que possible, et ne peut être utilisé qu'à la fin d'un motif de filtrage ; ainsi T1/# permettra de filtrer tous les sujets dont le premier niveau est T1, et # permet de recevoir tous les sujets publiés par le broker (A l'exception des sujets spéciaux commençant par $).

Qualité de service

Trois niveaux de qualité de service (QoS) sont définis pour la publication des messages :

QoS 0 : Livraison *une fois maximum.*

Les messages sont envoyés en fonction des capacités maximum du réseau TCP/IP sous-jacent. Aucune réponse n'est attendue. Aucune sémantique de relance n'est définie dans le protocole. Par conséquent, le message ne parvient pas du tout ou une seule fois au broker de destination. Le niveau QoS 0 est également connu comme fire and forget.

QoS 01 : Livraison *au moins une fois.*

L'arrivée d'un message QoS 1 au broker est reconnue. En cas d'échec identifiable de la liaison ou de l'unité d'envoi, ou bien après une certaine période de temps sans réception du message d'accusé de réception, l'expéditeur envoie à nouveau une copie du message. Par conséquent, le message est sûr d'arriver, mais il peut le faire plusieurs fois.

QoS 02 : Livraison *exactement une fois.*

Pour le niveau QoS 2, d'autres flux de protocoles sont utilisés au-delà de QoS 1 pour que des messages ne soient pas envoyés en double à l'application de réception. Il s'agit du niveau de service le plus élevé qui sert lorsque des messages en double ne sont pas appropriés. Il existe évidemment des conséquences en matière de trafic réseau, mais cet impact est souvent acceptable sachant l'importance du contenu du message.

La qualité de service peut être sélectionnée message par message, ce qui permet la publication des messages d'importance mineure avec le niveau QoS 0 et la distribution des messages plus importants avec QoS 2.

Ces niveaux sont mis en œuvre par des échanges supplémentaires entre l'expéditeur et le récepteur, et plus la qualité demandée est élevée, plus il faudra d'échanges pour valider une publication comme illustré dans la figure.

Pour tous les niveaux supérieurs à zéro, un identifiant est associé au message pour permettre son suivi, MQTT prévoyant la possibilité d'avoir au plus 65535 messages en attente (l'identifiant de message tenant sur 16 bits)

Sécurité

Trois concepts sont fondamentaux pour la sécurité MQTT sont comme suite :
— **L'identification** consiste à nommer le broker et le client auquel on donne des droits d'accès.
— **L'authentification** cherche à prouver mutuellement l'identité du client et du broker.
— **L'autorisation** consiste à gérer les droits du client.

Le broker MQTT s'identifie auprès du client avec son adresse IP et son certificat numérique. Le client MQTT utilise le protocole SSL pour authentifier le certificat envoyé par le broker.
Un client authentifie un broker à l'aide du protocole SSL.
Un broker MQTT authentifie un client à l'aide du protocole SSL, d'un mot de passe, ou des deux.
NB : L'autorisation ne fait pas partie du protocole MQTT. Elle est fournie par les brokers MQTT. Ce qui est autorisé ou non dépend de ce que fait le broker.

L'identifiant de client MQTT

Le protocole MQTT définit un «identifiant client» (ID client) qui identifie de manière unique un client dans un réseau. En termes simples, lors de la connexion à un broker, un client doit spécifier une chaîne unique qui n'est pas utilisée actuellement et ne sera pas utilisée par un autre client qui se connectera au broker MQTT. Il existe plusieurs façons de choisir un identifiant client. Voici quelques exemples :

— Un capteur installé dans un emplacement particulier peut utiliser le code d'emplacement comme ID du client.
— Un appareil mobile possédant une capacité de réseau peut choisir l'adresse MAC ou un identifiant d'appareil unique comme ID de client mais MQTT restreint la longueur d'ID de client à 23 caractères. Par conséquent, il y aura une situation où l'identification du client doit être raccourcie.

En raccourcissant l'identifiant du client, il faut s'assurer que l'ID client n'est pas identique à toute autre ID client utilisée dans le réseau. Afin de garder l'identifiant court et unique, il faut présenter un mécanisme de génération d'identifiant fiable. Par exemple, créer un identifiant client à partir du périphérique MAC 48 bits adresse. Si la taille de la transmission n'est pas un problème critique, et utiliser les 17 octets restants pour rendre l'adresse plus facile à administrer, comme un texte lisible par l'utilisateur dans l'identifiant.

Maintenant, essayons de comprendre les implications de deux clients obtenant le même identifiant client.

Le broker MQTT surveille les messages en attente d'être envoyés à un client en fonction de l'identifiant du client. Ainsi, si un client utilise QoS 1 ou QoS 2 et abonné à n'importe quel sujet et déconnecté du broker, le broker sauvera les messages arrivés pour le client pendant qu'il était déconnecté. Une fois que le client se reconnecte, le broker enverra ces messages au client. Si un autre appareil MQTT utilise le même ID de client et se connecte au broker, le broker enverra les messages qu'il avait sauvegardés.

Un autre scénario lié aux identifiants clients est la connexion en double. Disons qu'un périphérique particulier à l'aide de l'ID client DeviceA est connecté au courtier. Si un autre client vient avec le même ID de client DeviceA, le broker peut décider s'il doit permettre au nouveau client de se connecter et de déconnecter le client existant ou de garder l'ancienne connexion en vie et interdire le nouveau client. Il s'agit d'une fonctionnalité

facultative d'un broker MQTT.

1.3.4 Description informelle du protocole

nous allons proceder a la description du format binaire des paquets spécifié par la norme MQTT :

Représentations de données

1. **Bits** : Les bits dans un octet sont étiquetés de 7 à 0. Le bit numéro 7 est le bit le plus important, le bit le moins significatif est affecté au bit 0.
2. **Valeurs de données entières** : Les valeurs de données entières sont 16 bits : l'octet d'ordre supérieur précède l'octet d'ordre inférieur. Cela signifie qu'un mot de 16 bits est présenté sur le réseau comme octet le plus significatif , suivi de l'octet le moins important .
3. **Les chaines de caracteres :** Les chaines de caractères doivent toutes être encodées en utf8, et préfixées par leur longueur sur deux octets, ces chaines sont donc limitées à une longueur de
   $2^{16} - 1 = 65535$ octets.

Conclusion

Nous avons eu précédemment une vision sur l'internet des objets et ses différents protocoles et nous avons eu une idée sur le principe de la géolocalisation pour en choisir le protocole MQTT pour autant être un protocole Open Source et simple à implémenter.

Dans le prochain chapitre nous allons étudier la conception de notre projet qui sera de réaliser une application de suivi en utilisant le protocole MQTT.

Chapitre 2

Conception du système

Introduction

La conception est l'activité qui permet de transformer la description faite dans la phase d'analyse vers une description qui répond à la question comment et non plus quoi. La conception est donc l'activité qui commence à proposer les services nécessaires que le système doit offrir pour répondre aux besoins déjà évoqués dans la phase d'analyse. Elle permet donc d'élaborer l'architecture globale du système : où les différents composants et leurs relations sont développés ensuite de détailler ces composants pour aboutir à une description programmable. La conception peut être fonctionnelle ou orienté objet et ceci dépend du choix du concepteur. Dans ce chapitre, nous présentons notre conception en suivant les éléments suivant : tout d'abord on commence par présenter comment le système est sensé fonctionner. Il s'agit du modèle conceptuel présentant les grandes fonctionnalités du système ainsi que le flux de données entre ces différentes fonctionnalités. Ensuite, nous passons à une description orientée objet plus précise avec le langage UML, et enfin une description détaillée des différentes classes sera présentée.

2.1 Architecture fonctionnelle du système

2.1.1 Le Flux de données

Le Data Flow Diagram (DFD ; en français, diagramme de flux de données) est un type de représentation graphique du flux de données à travers un système d'information. Cet outil est souvent utilisé comme étape préliminaire dans la conception d'un système afin de créer un aperçu de ce système d'information. De plus, il est également utilisé pour visualiser le traitement de données (structured design).

Dans notre système le diagramme de flux de données sera représenté dans la figure 2.1

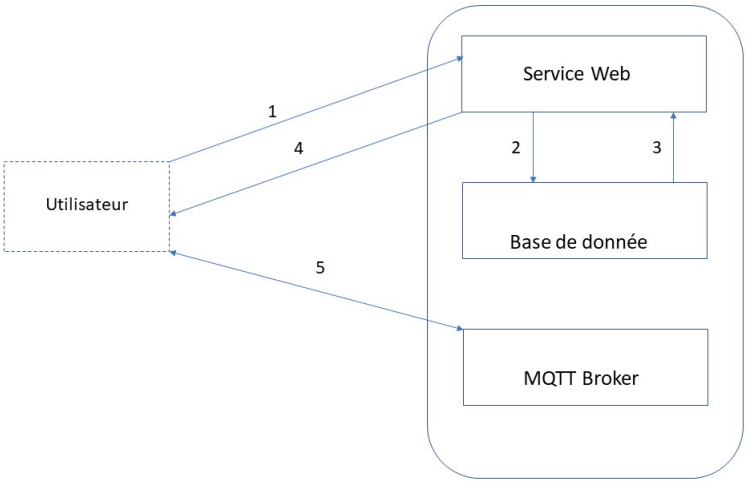


Figure 2.1 – le diagramme de flux de données (DFD)

notre système sera composé comme suit :

- L'utilisateur qui sera l'acteur externe via son terminal mobile.

• Le serveur où il y aura la base de donnée, le service web et en fin le Broker MQTT.
Donc le flux de donnée sera comme suit :

1. L'utilisateur envoie une requette d'authentification en saisissant le nom d'utilisateur et le mot passe au webservice.
2. Le webservice vérifie les informations reçues dans la base de donnée.
3. Si les informations sont correctes, les topics dans les quelles l'utilisateurs seront chargées dans le web service.
4. Les topics seront affichées dans l'interface de l'utilisateur.
5. Les messages seront envoyées et reçus entre le broker et l'utilisateur selon les topics

2.1.2 Identification des cas d'utilisation

Il est à rappeler qu'un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. La vue statique du comportement fonctionnel du système est représentée par le diagramme de cas d'utilisation de la figure 2.3

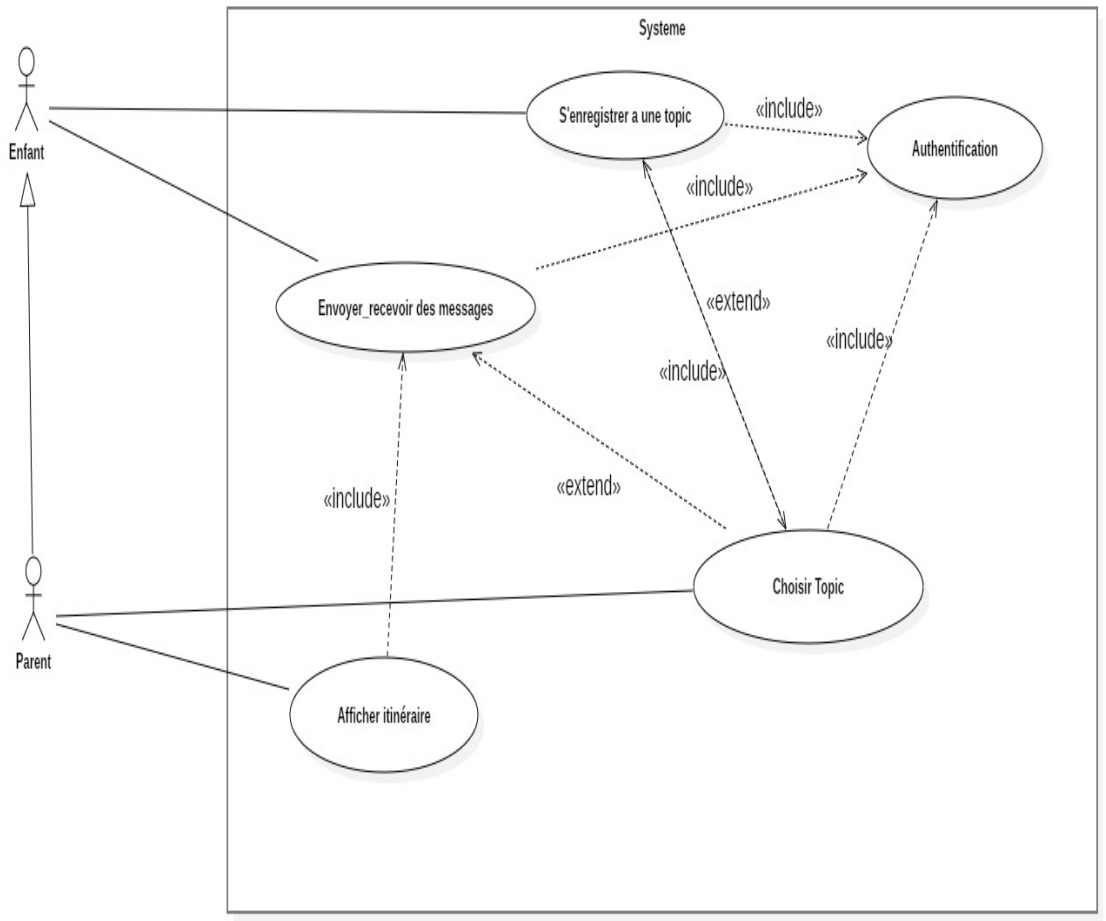


FIGURE 2.2 – Diagramme de cas d'utilisation

Description textuelle du diagramme de cas d'utilisation

La discription textuelle et est manière d'expliquer le diagramme de cas d'utilisation comme suit :

Cas d'utilisation	Authentification
Acteur Principaux	L'utilisateur et le système
Scénarion	L'utilisateur saisie l'idientifiant et le mot de passe
	Le système verifie les données saisies
	Si les données sont valides l'utilisateur sera connecté sinon un message d'erreur sera affiché

Cas d'utilisation	Choisir Topic
Acteur Principaux	Le Parent et le système
Scénarion	Le système affiche la liste des topics
	L'utilisateur choisie une parmis la liste affichée

Cas d'utilisation	Choisir Topic
Acteur Principaux	L'enfant et le système
Scénarion	Le système derrige l'enfant vers la fenetre d'attente du message

Cas d'utilisation	s'enregistrer à une topic
Acteur Principaux	le partent et le système
Scénarion	Le système derrige l'enfant vers la fenetre d'envoie du message

Cas d'utilisation	s'enregistrer à une topic
Acteur Principaux	L'enfant et le système
Scénarion	Le système derrige l'enfant vers la fenetre d'attente du message

Cas d'utilisation	envoie et reception du message
Acteur Principaux	Le parent et le systè0em
Scénarion	le parent envoie un message et attend la reception de la réponse

Cas d'utilisation	envoie et reception du message
Acteur Principaux	L'enfant et le système
Scénarion	l'enfant attend le message et l'ors de la reception du message il envoie sa gps automatiquement

Cas d'utilisation	afficher l'itinéraire
Acteur Principaux	Le parent et le système
Scénarion	l'ors de la reception du message le système affiche google maps l'itineraire

2.2 Diagrammes de séquences

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML [1].
Donc les diagrammes de séquence seront comme suit :

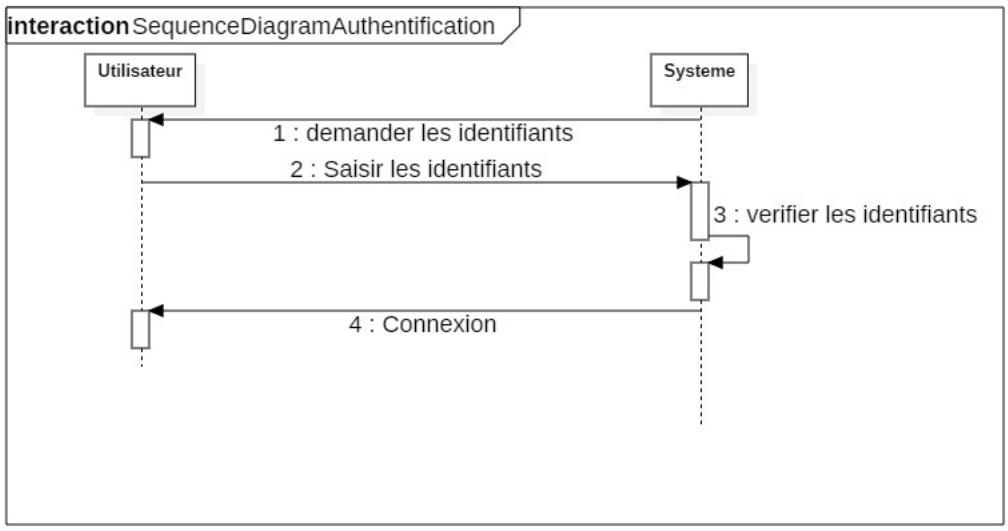


FIGURE 2.3 – le diagramme de séquence pour l'authentification

La figure 2.3 représente les séquences d'authentification dans le cas normal ou les données saisies sont correctes.

1. Le systeme affiche une fenêtre où l'utilisateur saisie l'identifiant et le mot de passe.
2. L'utilisateur saisie les données d'identification.
3. Le systeme vérifie les données.
4. Si les données sont valide l'utilisateur est connécté.

La figure 2.4 représente les séquences quand l'utilisateur est un enfant dans le cas normal ou les données saisies sont correctes.

1. Unified Modeling Language

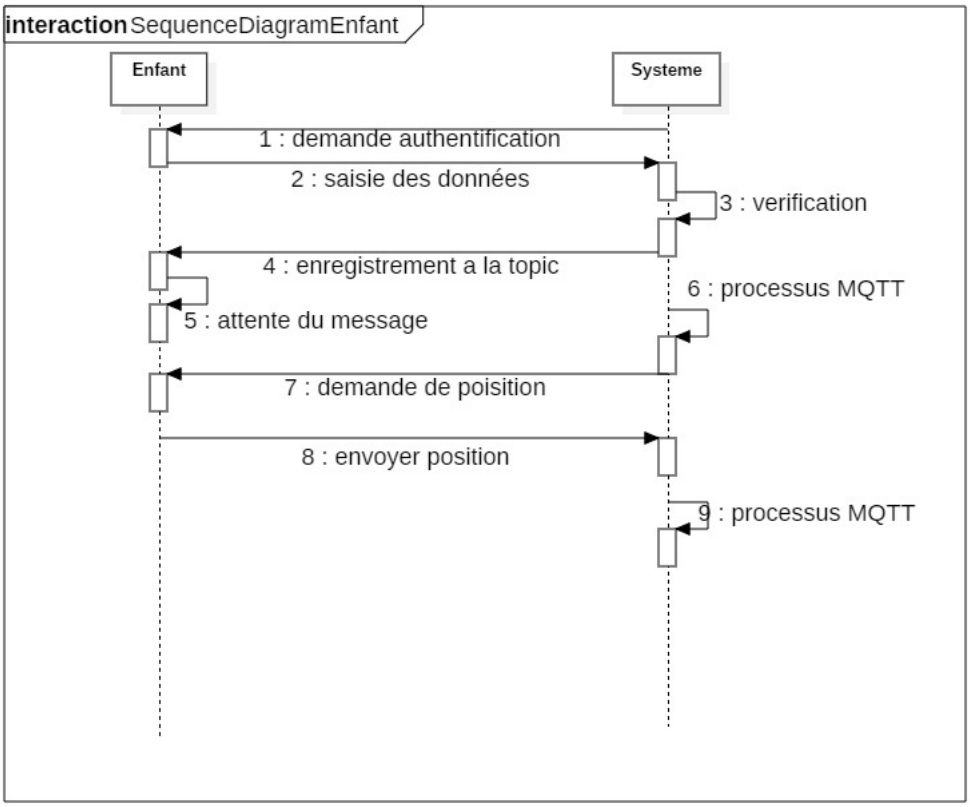


FIGURE 2.4 – le diagramme de séquence Enfant

1. Demande de saisie des données.
2. Saisie des données de connexion.
3. Le systeme vérifie les données.
4. L'enfant sera enregistré sur la topic prédéfinie.
5. L'enfant attend la demande position via un message.
6. Le Broker MQTT s'occupe de l'envoie et la reception des message.
7. Le Broker MQTT envoie la requête de position à l'enfant.
8. L'enfant Envoie la position GPS au Broker MQTT.
9. Le Broker MQTT envoie la poisition au demandeur.

La figure 2.5 représente les séquences quand l'utilisateur est un parent dans le cas normal ou les données saisies sont correctes.

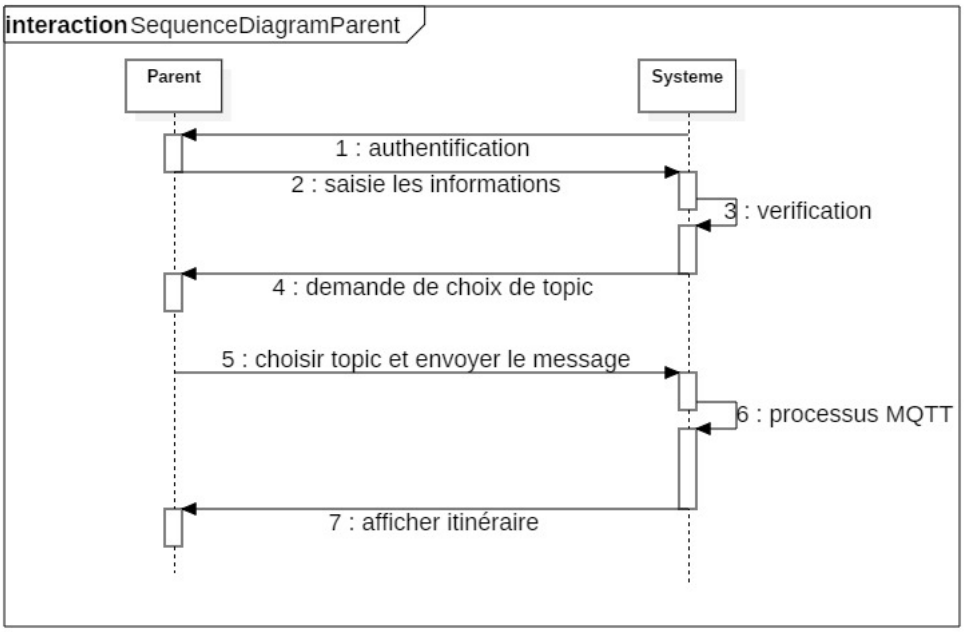


FIGURE 2.5 – le diagramme de séquence Parent

1. Demande de saisie des données.
2. Saisie des données de connexion.
3. Le systeme vérifie les données.
4. Le systeme envoie la liste des topics que le parent .
5. Le parent choisie une topic et envoie une demande de position.
6. Le Broker MQTT s'occupe de l'envoie et la réception des message.
7. lors de la réception du message reponse, la poistion GPS est affiché sur Google Maps automatiquement

2.3 Diagramme de Classes

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.

Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Les classes peuvent être liées entre elles grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, chacune de ces relations est représentée par un arc spécifique dans le diagramme de classes.

Elles sont finalement instanciées pour créer des objets (une classe est un moule à objet : elle décrit les caractéristiques des objets, les objets contiennent leurs valeurs propres pour chacune de ces caractéristiques lorsqu'ils sont instanciés).

Le diagramme de classe de notre sera représenté dans la figure 2.6

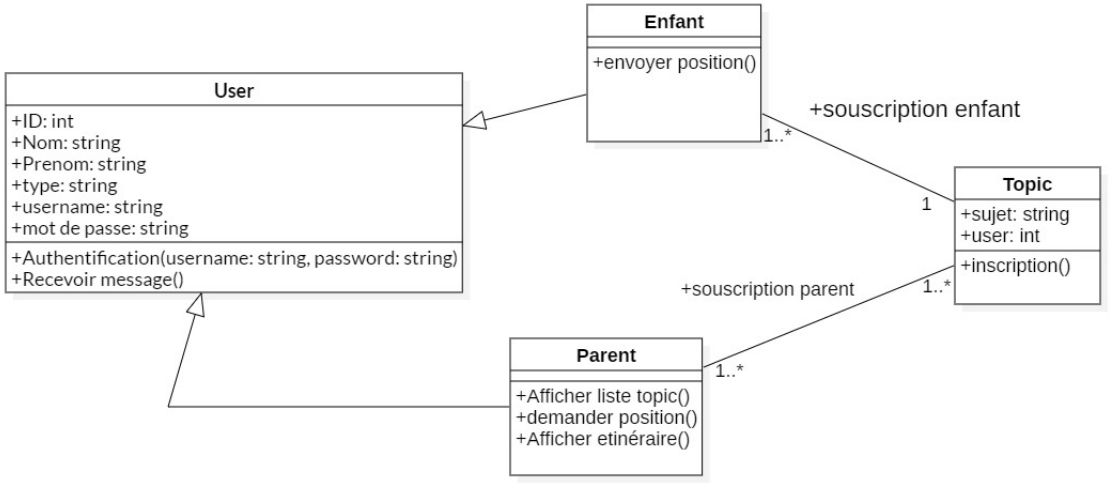


FIGURE 2.6 – le diagramme de séquence Enfant

Dans notre systeme on 4 classes. les classes Parent et Enfant heritent de la classe User qui composée des atribus suivants :

1. ID : identifiant unique du type Integer.
2. Nom : du type chaine de caractère.
3. Prenom : du type chaine de caractère.
4. Type : du type chaine de caractère.
5. Nom d'utilisateur : du type chaine de caractère.

6. Mot de passe : du type chaine de caractère.

et les methodes suivantes :

1. Authentification : qui prend les parametres : Nom d'utilisateur et Mot d epasse.
2. Recevoir un message.

Dans notre système on a 4 classes. les classes Parent et Enfant heritent de la classe User qui composée des atribus suivants :

1. ID : identifiant unique du type Integer.
2. Nom : du type chaine de caractere.
3. Prenom : du type chaine de caractere.
4. Type : du type chaine de caractere.
5. Nom d'utilisateur : du type chaine de caractere.
6. Mot de passe : du type chaine de caractere.

et les methodes suivantes :

1. Authentification : qui prend les parametres Nom d'utilisateur et Mot d epasse.
2. Recevoir un message.

la classe Enfant aura la methode envoyer une position gps et la classe parent aura les methodes suivantes :

1. Afficher la liste des topics.
2. Demander la position GPS.
3. Afficher etinéraire.

le systeme aussi aura une classe topic composée des atributs suivants :

1. Sujet : du type chaine de caractere.
2. user : du type Integer qui est l'identifiant unique.

Et la methode inscription qui liera les utilisateurs aux topics.

Les assossiations des classes seront comme suit :

1. l'enfant aura uniquement une seule topic.
2. Le parent aura une ou plusieurs topics.
3. La topic aura un ou plusieur enfants.
4. La topic aura un ou plusieur parents.

2.4 Conception détaillée

Nous allons voir dans cette section quelques algorithmes essentiels et les détailler.

Algorithm 1: Methode authentification

```
Result: Status
Lire les parametres Username et Password;
Connection a la base de donnée;
if la ligne existe then
    Status = connection;
else
    Status = erreur;
end
```

Le pseudo algorithme précedent représente la methode d'authentification qui sera comme suit :

1. La méthode prend deux parametres du type chaine de caracteres qui seront le nom d'utilisateur et le mot de passe.
2. On initialise les parametres reçu.
3. On se connecte a la base de données au serveur.
4. Si la ligne existe on se connecte.
5. Sinon un message d'erreur sera affiché.

Le pseudocode d'envoie de message de l'enfant serra comme suit :

Algorithm 2: Methode Envoie du message enfant

```
Result: Message
Se connecter au Broker MQTT;
if Connection = succes then
    x = true;
else
    x = false;
while x == true do
    attendre message ;
    if demande reçu then
        lire position GPS;
        Envoyer position GPS;
```

La methode d'envoie de message de l'enfant sera comme suit :

1. Se connecter au Broker Mqtt qui sera au serveur.
2. Attendre le message de requette.
3. Lire la position gps actuelle.
4. Envoyer la position GPS au Broker Mqtt selon la topic.

Aprés la connection du parent, exécutera le pseudocode suivant :

Algorithm 3: Algorithme Parent

```
Result: Position GPS
Se connecter au Broker MQTT;
Se connecter a la base de donnée;
if Connection = succes then
    Afficher la liste des topics ;
    choisir la topic ;
    envoyer le message ;
    x = true;
else
    x = false;
while x == true do
    if reponse reçu then
        lire position GPS;
        afficher itenéraire;
```

1. Le parent se connect au broker et la base de donnée.
2. Le parent choisi une Topic.
3. Le parent Envoi une demande de position et attend la réponse.
4. L'ors de la réception du message l'itinéraire sera affiché.

Conclusion

Dans ce Chapitre nous avons la coneption du systeme en illusrant par les diagrammes de flux de donnée, de class et de séquences et nous avons terminé par les algorithmes les plus important dans notre systeme.

Dans le chapitre suivant nous allons voir la réalisation du système

Chapitre 3

Réalisation du système

Introduction

Dans ce chapitre nous allons parler de l'implémentation de tous les composants du système et des langages de programmation que nous avons utilisé.

3.1 La Base de donnée

Une base de données est un « conteneur » stockant des données telles que des chiffres, des dates ou des mots, pouvant être retraités par des moyens informatiques pour produire une information, par exemple, des chiffres et des noms assemblés et triés pour former un annuaire. Les retraitements sont typiquement une combinaison d'opérations de recherches, de choix, de tri, de regroupement, et de concaténation.

C'est la pièce centrale d'un système d'information ou d'un système de base de données (ou base de données tout court), qui régit la collecte, le stockage, le retraitement et l'utilisation de données. Ce dispositif comporte souvent un logiciel moteur, des logiciels applicatifs, et un ensemble de règles relatives à l'accès et l'utilisation des informations.

Le système de gestion de base de données est une suite de programmes qui manipule la structure de la base de données et dirige l'accès aux données qui y sont stockées. Une base de données est composée d'une collection de fichiers, on y accède par le SGBD qui reçoit des demandes de manipulation du contenu et effectue les opérations nécessaires sur les fichiers. Il cache la complexité des opérations et offre une vue synthétique sur le contenu. Le SGBD permet à plusieurs usagers de manipuler simultanément le contenu, et peut offrir différentes vues sur un même ensemble de données.

Dans notre systeme nous avons utilisé (Microsoft SQL Server) comme SGBD et notre base de donnée sera comme suite (voir la figure 3.1) :

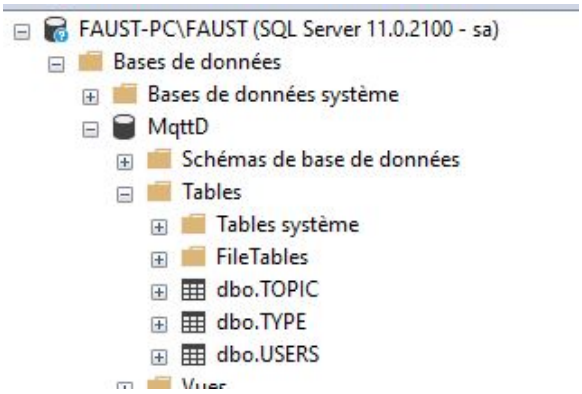


Figure 3.1 – La Base de donnée

Il est necessaire de noter que SQL Server assure la sécurité des données par l'authentification (voir la figure 3.2) et dans le cas du réseau on doit ajouter l'adresse du serveur dans le fichier host de l'ordinateur (voir la figure 3.3 et la figure 3.4)

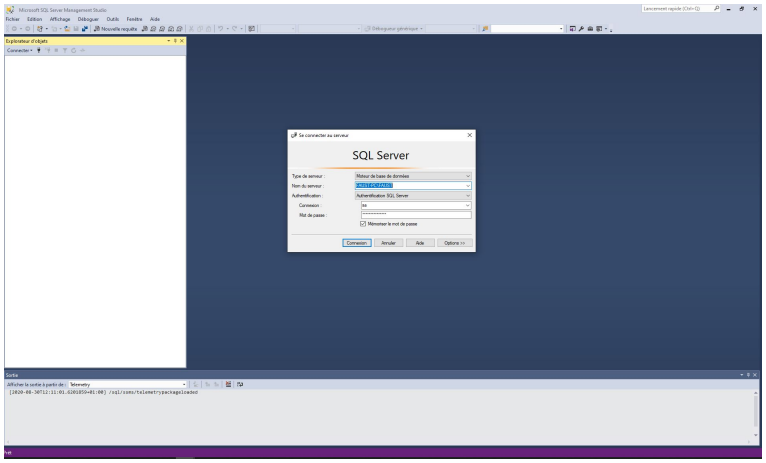


FIGURE 3.2 – Connexion SGDB

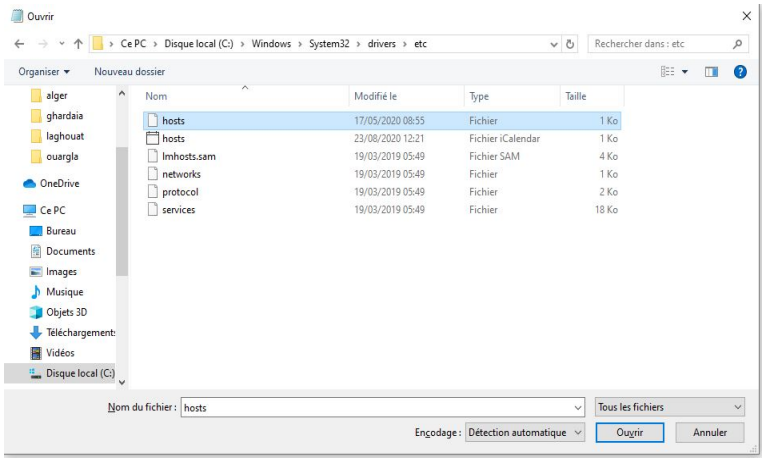


FIGURE 3.3 – chemin host

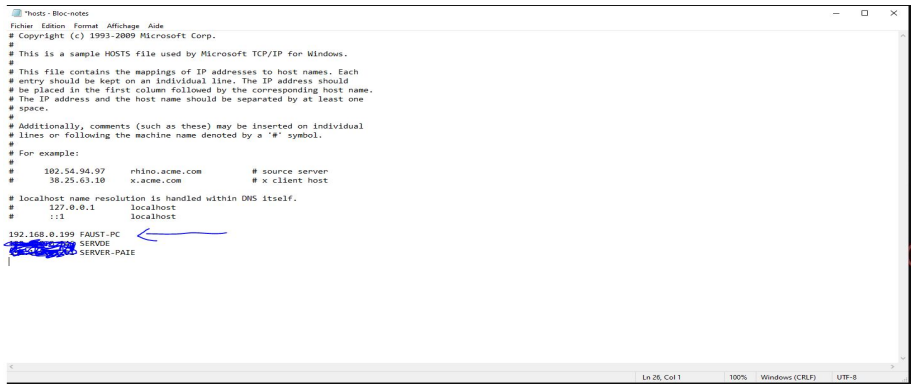
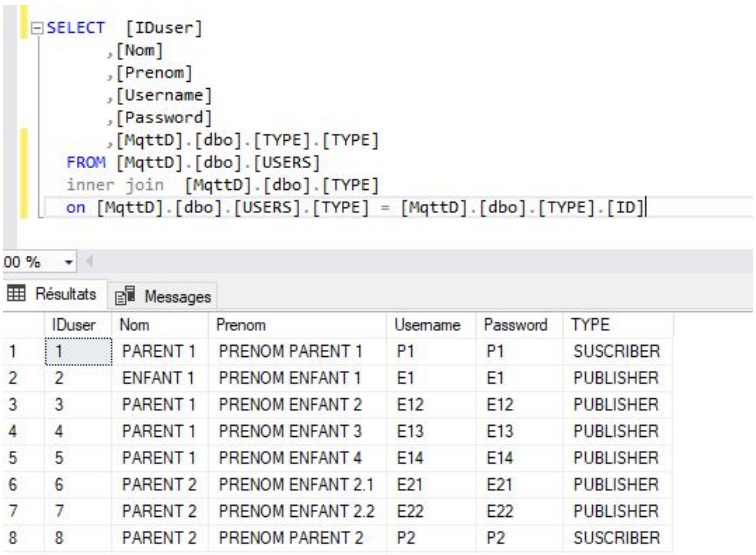


FIGURE 3.4 – fichier host

Notre base de données est composée de trois tables :

1. La table USERS composée de :

- ID user qui est la clé primaire
- Nom.
- Prenom.
- Username.
- Password
- Type qui est une clé etrangere vers la table TYPE pour eviter la redondance.



```
SELECT  [IDuser]
      ,[Nom]
      ,[Prenom]
      ,[Username]
      ,[Password]
      ,[MqttD].[dbo].[TYPE].[TYPE]
  FROM [MqttD].[dbo].[USERS]
  inner join  [MqttD].[dbo].[TYPE]
  on [MqttD].[dbo].[USERS].[TYPE] = [MqttD].[dbo].[TYPE].[ID]
```

00 %

Résultats | Messages

	IDuser	Nom	Prenom	Username	Password	TYPE
1	1	PARENT 1	PRENOM PARENT 1	P1	P1	SUSCRIBER
2	2	ENFANT 1	PRENOM ENFANT 1	E1	E1	PUBLISHER
3	3	PARENT 1	PRENOM ENFANT 2	E12	E12	PUBLISHER
4	4	PARENT 1	PRENOM ENFANT 3	E13	E13	PUBLISHER
5	5	PARENT 1	PRENOM ENFANT 4	E14	E14	PUBLISHER
6	6	PARENT 2	PRENOM ENFANT 2.1	E21	E21	PUBLISHER
7	7	PARENT 2	PRENOM ENFANT 2.2	E22	E22	PUBLISHER
8	8	PARENT 2	PRENOM PARENT 2	P2	P2	SUSCRIBER

FIGURE 3.5 – table users

2. La table TYPE :
   - ID TYPE qui est la clé primaire.
   - Type .
3. La table topic :
   - IDUSER qui est une clé etrangere vers l'utilsateur.
   - TOPIC .

```
SELECT  [IDuser]
      ,[Nom]
      ,[Prenom]
      ,[Username]
      ,[Password]
      ,[MqttD].[dbo].[TYPE].[TYPE]
  FROM [MqttD].[dbo].[USERS]
  inner join  [MqttD].[dbo].[TYPE]
  on [MqttD].[dbo].[USERS].[TYPE] = [MqttD].[dbo].[TYPE].[ID]
```

00 % — Résultats | Messages

	IDuser	Nom	Prenom	Username	Password	TYPE
1	1	PARENT 1	PRENOM PARENT 1	P1	P1	SUSCRIBER
2	2	ENFANT 1	PRENOM ENFANT 1	E1	E1	PUBLISHER
3	3	PARENT 1	PRENOM ENFANT 2	E12	E12	PUBLISHER
4	4	PARENT 1	PRENOM ENFANT 3	E13	E13	PUBLISHER
5	5	PARENT 1	PRENOM ENFANT 4	E14	E14	PUBLISHER
6	6	PARENT 2	PRENOM ENFANT 2.1	E21	E21	PUBLISHER
7	7	PARENT 2	PRENOM ENFANT 2.2	E22	E22	PUBLISHER
8	8	PARENT 2	PRENOM PARENT 2	P2	P2	SUSCRIBER

FIGURE 3.6 – table Topic

3.2 Configuration du pare-feu

Un pare-feu (de l'anglais firewall) est un logiciel et/ou un matériel permettant de faire respecter la politique de sécurité du réseau, celle-ci définissant quels sont les types de communications autorisés sur ce réseau informatique. Il surveille et contrôle les applications et les flux de données (paquets).

Le Pare-feu Windows est un pare-feu personnel édité par Microsoft. Il est inclus dans les systèmes d'exploitation Microsoft Windows XP (SP2), Windows Server 2003, Windows Vista, Windows 7, Windows 8 et Windows 10.

Dans notre serveur on doit ajouter 4 regles entrantes et sortantes pour les ports afin de permetre aux données de circuler entre les differents composants du systeme et ils seront comme suite (voir la figure 3.7) :

1. Le port 1433 pour SQL server pour pouvoir ce connecter a la base de donnée [1]
2. Le port 80 pour le protocol https.
3. Le port 7007 pour IIS afin d'heberger notre Web Service
4. Le port 1883 pour le Broker Mosquito qui est un port prédéfinie.

FIGURE 3.7 – table Topic

1. **NB :** Le port 1433 est dédié uniquement à SQL server par Microsoft

3.3 SERVICE WEB

Un service web (ou service de la toile) est un protocole d'interface informatique de la famille des technologies web permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, de manière synchrone ou asynchrone. Le protocole de communication est défini dans le cadre de la norme SOAP dans la signature du service exposé (WSDL). Actuellement, le protocole de transport est essentiellement HTTP. Le concept a été précisé et mis en œuvre dans le cadre de Web Services Activity, au W3C, particulièrement avec le protocole SOAP. Associé avec les échanges de données informatisés (EDI), le consortium ebXML l'a utilisé pour automatiser des échanges entre entreprises. Cependant le concept s'enrichit avec l'approfondissement des notions de ressource et d'état, dans le cadre du modèle REST, et l'approfondissement de la notion de service, avec le modèle SOA.

Notre Service Web sera composé de deux methodes, la première methode sera la methode authentification où pour l'invoquer on passera le nom de l'utilisateur et le mot de passe et la methode GetTopic ou on passera l'id de l'utilisateur (voir la figure 3.8).

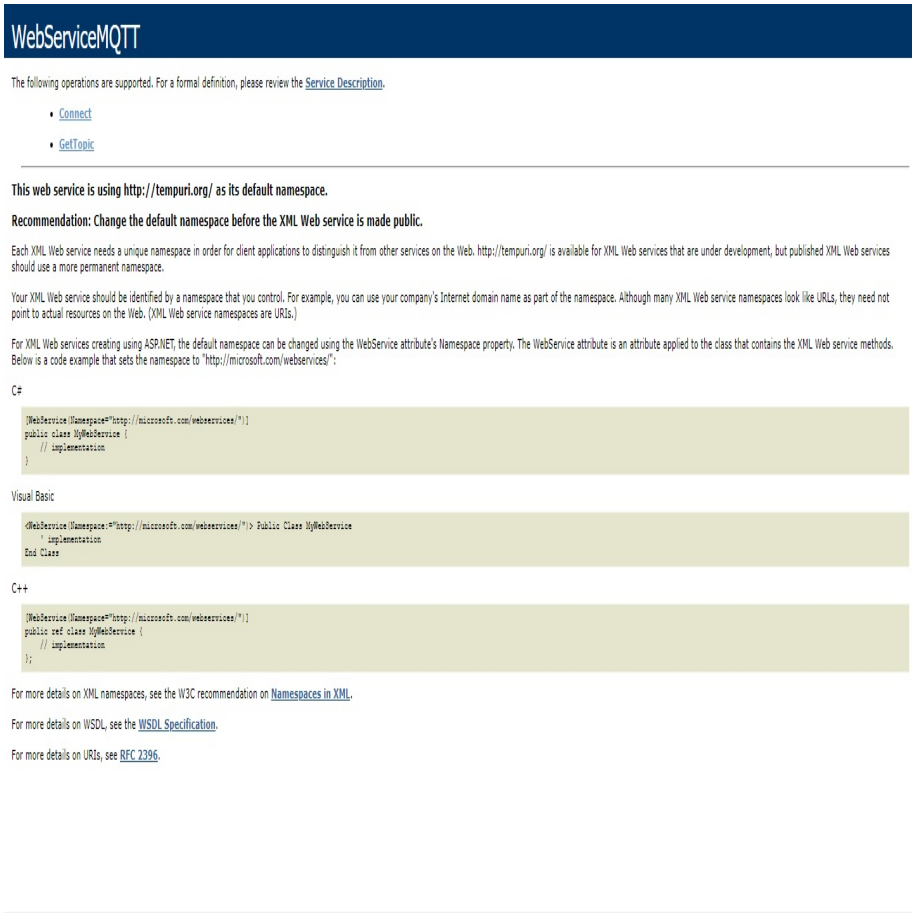


FIGURE 3.8 – Interface Service Web

Pour entamer la programmation de notre web service on doit tout d'abord configurer la chaine de connection de la base de donnée. Pour cela on doit ajouter la ligne de commande qui est dans la figure 3.9 dans le fichier Webconfig dans notre projet.

```
<configuration>
    <configSections>
    </configSections>
    <connectionStrings>
        <add name="MqttWebService.Properties.Settings.Paramètre"
             connectionString="Data Source=FAUST-PC\FAUST;Initial
             Catalog=MqttD;Persist Security Info=True;User ID=SA;Password=psqtlcpcqtl2"
            providerName="System.Data.SqlClient" />
    </connectionStrings>
    <system.web>
      <compilation debug="true" targetFramework="4.0" />
```

FIGURE 3.9 – Chaine de connection à la base de donnée

La methode d'authentification recevera deux parametres comme cité précédement, on doit initialiser la connection a la base de donnée, ensuite initialiser la requette SQL et lire les données pour enfin retourner le resultat sous Json pour optimiser la taille des paquets de données et augmenter la fluidité du systeme. Nous aurions pu choisir retourner le resultat sous XML mais nous avons préféré Json pour voir invoquer le resultat via lien HTTP direct(voir les figures 3.10, 3.11 et 3.12).

```
[WebMethod]
[ScriptMethod(UseHttpGet = true, ResponseFormat = ResponseFormat.Json)]
public void Connect(string user, string password)
{
    try
    {
        TabUsers a = new TabUsers();
        SqlConnection Con = new SqlConnection(DataBase);
        Con.Open();
        SqlCommand command = new SqlCommand();
        command.Connection = Con;
        command.CommandType = CommandType.Text;
        command.CommandText = "SELECT [IDuser] ,[Nom],[Prenom],[TYPE] FROM [MqttD].[dbo].[USERS]" +
            " where Username = '"+ user + "' and Password='"+ password + "'";
        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows)
        {
            while (reader.Read())
            {
                User temp = new User();
                temp.ID = int.Parse(reader[0].ToString());
                temp.Type = int.Parse(reader[3].ToString());
                temp.Nom = reader[1].ToString();
                temp.Prenom = reader[2].ToString();
                a.AddTemp(temp);
            }
        }
        else
        {
            a.AddTemp(null);
        }
        Con.Close();
        JavaScriptSerializer ser = new JavaScriptSerializer();
        HttpContext.Current.Response.Write(ser.Serialize(a));
    }
    catch
    {
        throw;
    }
}
```

FIGURE 3.10 – Methode d'authentification

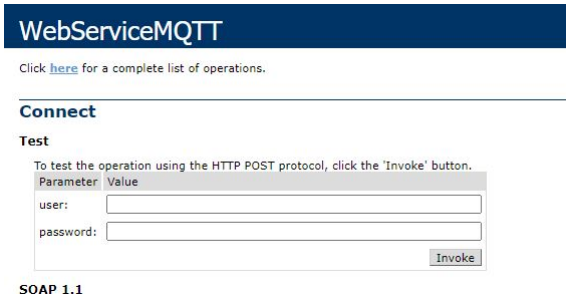


FIGURE 3.11 – Inteface Methode d'authentification

FIGURE 3.12 – Resultat Methode d'authentification

La methode GetTopic recevera L'id de l'utilisateur et retournera en suite la liste des Topics où l'utilisateur est déja enregistré. De la même façon que la methode précedente on va proceder a la programmation de la methode sauf la requete SQL et le resultat de retour (voir les figures 3.13, 3.14 et 3.15).

```
[WebMethod]
[ScriptMethod(UseHttpGet = true, ResponseFormat = ResponseFormat.Json)]
public void GetTopic(int ID)
{
    try
    {

        TabTopic a = new TabTopic();
        SqlConnection Con = new SqlConnection(DataBase);
        Con.Open();
        SqlCommand command = new SqlCommand();
        command.Connection = Con;
        command.CommandType = CommandType.Text;
        command.CommandText = "SELECT [topicc] FROM [MqttD].[dbo].[TOPIC] where iduser ="+ID;

        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows)
        {
            while (reader.Read())
            {
                Topic temp = new Topic();

                temp.topic = reader[0].ToString();
                a.AddTemp(temp);
            }
        }
        else
        {
            a.AddTemp(null);
        }
        Con.Close();
        JavaScriptSerializer ser = new JavaScriptSerializer();
        HttpContext.Current.Response.Write(ser.Serialize(a));
    }
    catch
    {
        throw;
    }
}
```

FIGURE 3.13 – Resultat Methode d'GetTopic

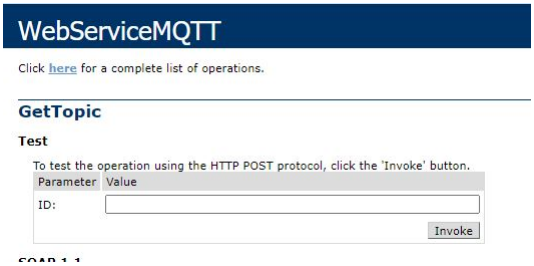


FIGURE 3.14 – Interface Methode d'GetTopic

FIGURE 3.15 – Resultat Methode d'GetTopic

Pour activer l'option de l'invoquation via lien direct on doit aller dans notre fichier Webconfig et ajouter les lignes XML dans la figure 3.16 :

```
<webServices>
  <protocols>
    <add name="HttpGet"/>
    <add name="HttpPost"/>
  </protocols>
</webServices>
```

FIGURE 3.16 – HTTP GET SET

3.4 Broker MQTT

Eclipse Mosquitto est un courtier de messages open source (sous licence EPL / EDL) qui implémente les versions 5.0, 3.1.1 et 3.1 du protocole MQTT. Mosquitto est léger et convient à tous les appareils, des ordinateurs monocarte basse consommation aux serveurs complets.

Le protocole MQTT fournit une méthode légère d'exécution de la messagerie à l'aide d'un modèle de publication / abonnement. Cela le rend adapté à la messagerie Internet des objets, par exemple avec des capteurs de faible puissance ou des appareils mobiles tels que des téléphones, des ordinateurs intégrés ou des microcontrôleurs.

Le projet Mosquitto fournit également une bibliothèque C pour l'implémentation des clients MQTT, et les très populaires clients MQTT en ligne de commande mosquitto_pub et mosquitto_sub.

Mosquitto fait partie de la Fondation Eclipse, est un projet iot.eclipse.org et est sponsorisé par cedalo.com.

Nous avons choisi Mosquitto pour sa simplicité d'imlplémentation, il suffit juste d'installer le package et ajouter la regle commme cité avant ensuite on l'execute sur le serveur (voir la figure 3.17).

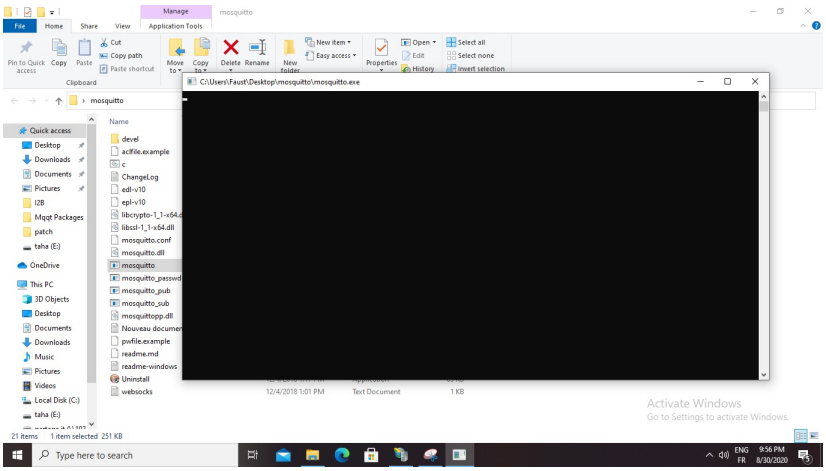


FIGURE 3.17 – Mosquitto Broker

3.5 Android

Les autorisations

Avant Android 6.0 «Marshmallow», les autorisations étaient automatiquement accordées aux applications au moment de l'exécution et elles étaient présentées lors de l'installation dans Google Play Store. Depuis Marshmallow, certaines autorisations nécessitent désormais que l'application demande l'autorisation à l'exécution par l'utilisateur. Ces autorisations peuvent également être révoquées à tout moment via le menu des paramètres d'Android. L'utilisation des autorisations sur Android est parfois abusée par les développeurs d'applications pour collecter des informations personnelles et diffuser de la publicité ; en particulier, les applications pour utiliser le flash de l'appareil photo d'un téléphone comme lampe de poche (qui sont devenues largement redondantes en raison de l'intégration de ces fonctionnalités au niveau du système sur les versions ultérieures d'Android) sont connues pour nécessiter un large éventail d'autorisations inutiles au-delà de ce qui est réellement nécessaire pour la fonctionnalité indiquée.

Dans notre applications nous aurons besoin des autorisations suivantes :

1. Internet.

2. Etat du réseau.
3. Vibrations.
4. Lecture de la memoire Interne.
5. Ecriture dans la memoire Interne.
6. Reseau Wifi.
7. Verrouillage mise en veille.
8. Emplacement GPS.

Pour cela on doit aller dans le fichier Manifest.xml (voir la figure 3.18).

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

FIGURE 3.18 – Autorisations

ClearText

Cleartext désigne toute information transmise ou stockée qui n'est pas cryptée ou destinée à être cryptée.
Lorsqu'une application communique avec des serveurs à l'aide d'un trafic réseau en texte clair, tel que HTTP, cela peut augmenter le risque d'écoute et de falsification du contenu. Des tiers peuvent injecter des données non autorisées ou divulguer des informations sur les utilisateurs. C'est pourquoi les développeurs sont encouragés à un trafic sécurisé uniquement, tel que HTTPS.
Mais juste au cas où l'utilisation du texte clair serait inévitable, les développeurs peuvent corriger l'erreur en :

1. Modification de l'attribut useCleartextTraffic dans le fichier Manifest.
2. Ajout de la configuration de sécurité réseau.

Android 6.0 a introduit l'attribut useCleartextTraffic sous l'élément application dans le manifeste Android. La valeur par défaut dans Android P est «false». La définition de ce paramètre sur true indique que l'application a l'intention d'utiliser un trafic réseau clair.

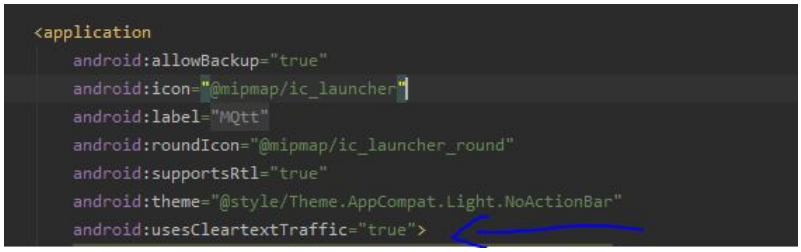


FIGURE 3.19 – Attribut Text Claire

AsyncTask

Les AsyncTask permettent une utilisation correcte et facile du ThreadUI. Cette classe permet d'effectuer des tâches de fond et de publier des résultats sans manipuler des threads et/ou des handlers.
Une tâche asynchrone est définie par un calcul qui fonctionne sur un thread en arrière-plan et dont le résultat est publié sur le ThreadUI. Une tâche asynchrone est définie par 3 types génériques, appelés : le paramètre, la progression et le résultat ainsi que 4 étapes, appelées OnPreExecute, doInBackground, onProgressUpdate et onPostExecute.
Dans notre cas on utlise les AsyncTask pour invoquer les methodes qui sont dans service web car la tache risque de prendre un temps important.
La procedure doInBackground agira de la façon suivante :

1. Recevoir l'Url du service web en parametre.
2. Ouvrir la connection et lire le contenu de la page web.
3. convertir le contenu de la page web en Json.
4. decoder le fichier Json et lire les données.

Pour illustrer nous donnons comme exemple la methode d'authentifications dans la figure suivante :

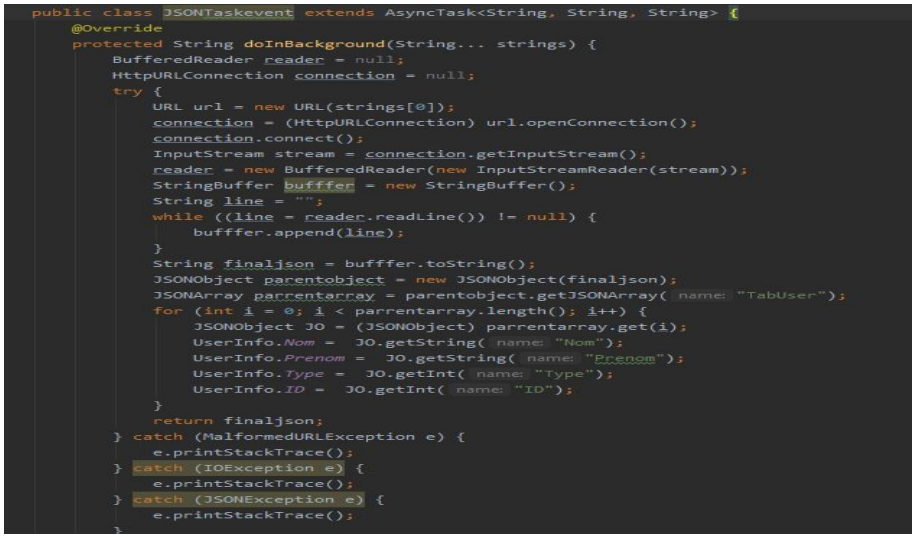


FIGURE 3.20 – AsyncTask d'authentification

Pour faire appel a l"AsyncTask" on doit lire le nom de l'utilisateur et le mot de passe puis on clique sur le bouton Connecter. ensuite les parametre seront passé a la tache et elle va executer la procedure doInBackground.

```
logtbn.setOnClickListener((v) → {
        EditText usrtxt= findViewById(R.id.UserTXT);
        EditText passtxt= findViewById(R.id.editTextTextPassword);
        username = usrtxt.getText().toString();
        password = passtxt.getText().toString();
        new JSONTaskevent().execute("http://192.168.0.199:7007/MqttWebService/WebServiceMQTT.asmx/Connect?user="+username+"&password="+password);

});
```

FIGURE 3.21 – Methode authentification android

Connection au Broker MQTT

La connection au Broker Mqtt se fait par les étapes suivantes :

1. Creer une nouvelle instance de Mqtt Android, elle prend en parametre la fenetre en cours d'execution et l'adresse IP du broker.
2. On choisi la version du protocol MQTT, dans notre cas on a choisis la version 3.1.
3. on aura en suite deux evenements :
   - onSucess : qui veut dire connection etablie.
   - onFailure : qui veut dire connection est non etablie. Un message d'erreur sera affiché.

```
client = new MqttAndroidClient(this.getApplicationContext(),  serverURI: "tcp://192.168.0.199:1883",
        clientId);
MqttConnectOptions options = new MqttConnectOptions();
options.setMqttVersion(MqttConnectOptions.MQTT_VERSION_3_1);
try {
    IMqttToken token = client.connect(options);
    token.setActionCallback(new IMqttActionListener() {
        @Override
        public void onSuccess(IMqttToken asyncActionToken) {
            // We are connected
            Toast.makeText(getApplicationContext(), topicstr, Toast.LENGTH_LONG).show();
            setsuscription(topicstr);
        }

        @Override
        public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
            // Something went wrong e.g. connection timeout or firewall problems
            Toast.makeText(getApplicationContext(),  text: "Broker non Connecté", Toast.LENGTH_LONG).show();

        }
    });
} catch (MqttException e) {
    e.printStackTrace();
}
```

FIGURE 3.22 – Connection au Broker MQTT

Evoie et reception des messages

Le broker envoie les messages a tout les souscripteurs de la topic. Donc si par exemple un parent et un enfant sont inscrits a la meme topic, lorsque le parent envoie le message, l'enfant le recevera, mais le parent aussi. Pour eviter ce probleme majeur nous avons distingué les topics en concaténant les topics avec "demande" et "réponse", pour bien illuster la situation prenons l'exemple suivant :

1. Le parent 1 est souscrit a la topic PARENT1/ENFANT1.
2. L'enfant 1 est souscrit a la topic PARENT1/ENFANT1.
3. Le parent 1 publie sur la topic PARENT1/ENFANT1demande.
4. L'enfant 1 sera souscrit a la topic PARENT1/ENFANT1demande.
5. L'enfant 1 publie sur la topic PARENT1/ENFANT1reponse.
6. Le parent 1 sera souscrit a la topic PARENT1/ENFANT1reponse.

De cette façon on assure la déliverance des message aux clients souhaités (voir les figures 3.23 et 3.24).

```
private void publish ( String topicmessage)
{
    String topic = topicmessage;
    String message = "Salut ! où est-tu ?";

    try {

        client.publish( topic: topic+"Demande", message.getBytes(), qos: 0, retained: false);
    } catch (MqttException e) {
        e.printStackTrace();
    }
}
```

FIGURE 3.23 – publication du parent

```
private void publish ( String message) {

    String topic = topicstr +"Reponse";


    try {

        client.publish(topic, message.getBytes(), qos: 1, retained: false);
        Toast.makeText(getApplicationContext(),message,Toast.LENGTH_SHORT).show();
        Toast.makeText(getApplicationContext(),topic,Toast.LENGTH_LONG).show();
    } catch (MqttException e) {
        e.printStackTrace();
    }
}
```

FIGURE 3.24 – publication de l'enfant

Google MAPS API

Google Maps est un service de cartographie en ligne. Le service a été créé par Google suite au rachat en octobre 2004 de la start-up australienne Where 2 Technologies. Lancé en février 2005 aux États-Unis et au Canada, puis en Grande-Bretagne (sous le nom de Google Local), Google Maps a été lancé mardi 25 avril 2006, simultanément en France, Allemagne, Espagne et Italie.

L'API Google Maps, maintenant appelée Google Maps Platform, héberge environ 17 API différentes, classées dans les catégories suivantes ; Cartes, lieux et itinéraires.

Après le succès des mashups d'ingénierie inverse tels que chicagocrime.org et Housingmaps.com, Google a lancé l'API Google Maps en juin 2005 pour permettre aux développeurs d'intégrer Google Maps dans leurs sites Web. C'était un service gratuit qui ne nécessitait pas de clé API jusqu'en juin 2018 (les modifications sont entrées en vigueur le 16 juillet), lorsqu'il a été annoncé qu'une clé API liée à un compte Google Cloud avec facturation activée serait nécessaire pour accéder à l'API. L'API ne contient actuellement pas d'annonces, mais Google déclare dans ses conditions d'utilisation qu'il se réserve le droit d'afficher des annonces à l'avenir.

En utilisant l'API Google Maps, il est possible d'intégrer Google Maps dans un site Web externe, sur lequel des données spécifiques au site peuvent être superposées. Bien qu'initialement uniquement une API JavaScript, l'API Maps a été étendue pour inclure une API pour les applications Adobe Flash (mais cela a été déconseillé), un service de récupération d'images cartographiques statiques et des services Web pour effectuer le géocodage, générer des itinéraires et obtenir l'altitude. profils. Plus de 1 000 000 de sites Web utilisent l'API Google Maps, ce qui en fait l'API de développement d'applications Web la plus utilisée. En septembre 2011, Google a annoncé qu'il abandonnerait l'API Google Maps pour Flash.

L'API Google Maps est gratuite à des fins commerciales, à condition que le site sur lequel elle est utilisée soit accessible au public, ne facture pas d'accès et ne génère pas plus de 25 000 accès à la carte par jour. Les sites qui ne remplissent pas ces conditions peuvent acheter l'API Google Maps for Business.

Depuis le 21 juin 2018, Google a augmenté les prix de l'API Maps et requiert un profil de facturation.

Pour pouvoir utliser Google Maps API dans notre application il faut suivre les étapes suivantes :

1. Se rendre au site ""https ://console.developers.google.com""

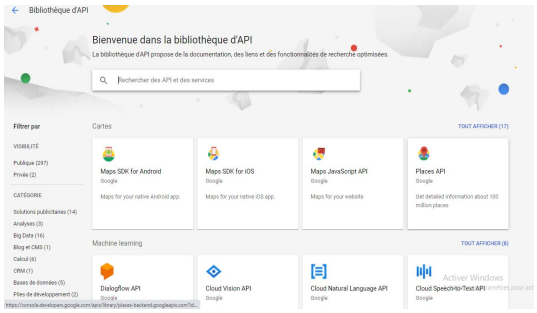


FIGURE 3.25 – Google Developer Console

2. Activer MAPS SDK pour android

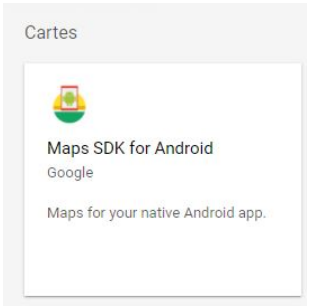


FIGURE 3.26 – MAPS SDK

3. Dans le menu identifant on clique sur créer puis Clé API.

FIGURE 3.27 – Clé API

4. On crée un fichier map_key.xml et on copie la clé pour qu'elle puisse être utilisé.

FIGURE 3.28 – Clé API en xml

5. Ajouter la valeur du fichier xml dans le fichier manifest.

FIGURE 3.29 – Ajout au fichier manifest

3.6 Résultat Final

Nous allons a présent présenter l'interface de notre application mobile.

Dans cette interface l'utilisateur doit entrer le nom d'utilisateur et le mot de passe déja enregistré dans la base de donnée.

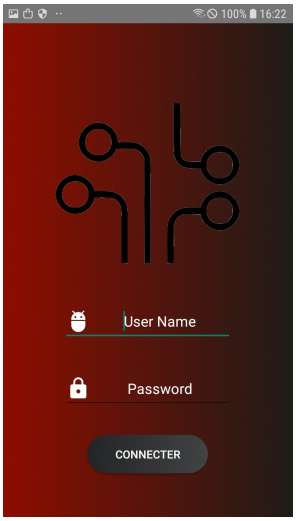


FIGURE 3.30 – Interface de connection

Le systeme distinguera le type de l'utilisateur et le dérigera vers l'interface adéquoite. S'il est un parent la figure3.35 sera affiché, sinon la figure 3.36 sera affiché.

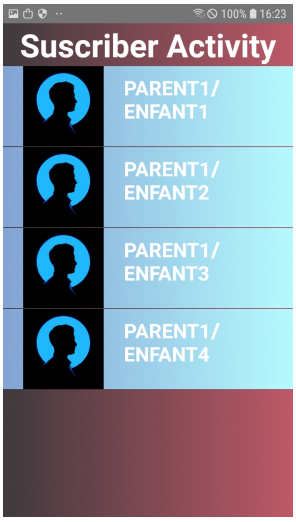


FIGURE 3.31 – Interface Parent

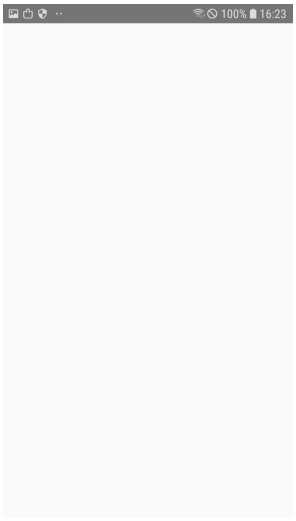


FIGURE 3.32 – Interface Enfant

Dans l'interface du parent, Le parent doit cliquer sur une topic pour envoyer un message a l'enfant voulu, ensuite l'enfant recevera un message et une notification sonnore avec une vibration. La position sera envoyée au parent automatiquement et l'interface de google MAPS sera affichée (voir la figure 3.37).

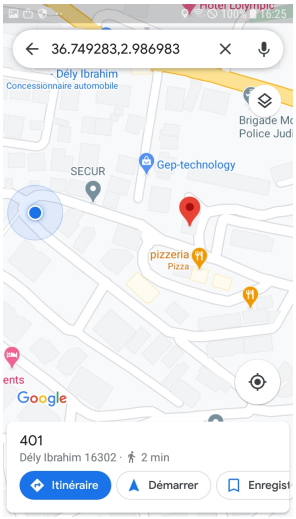


FIGURE 3.33 – Interface GOOGLE MAPS

3.7 Scénario

— Le parent et l'enfant se connecte.

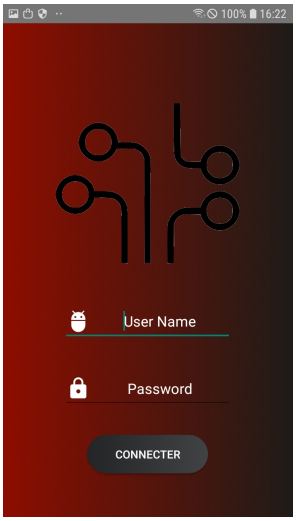


FIGURE 3.34 – Interface de connection

— Le parent choisit la topic.

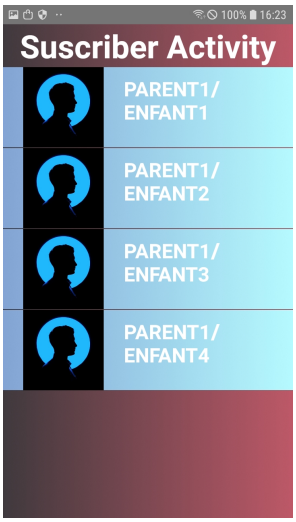


FIGURE 3.35 – Interface Parent

— l'enfant attend le message.

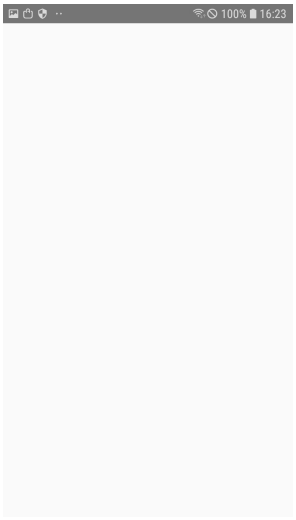


FIGURE 3.36 – Interface Enfant

— Le parent sera dérigé vers goole MAPS.

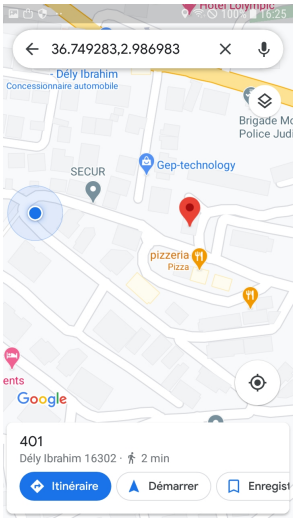


FIGURE 3.37 – Interface GOOGLE MAPS

CONCLUSION

Nous avons vu dans ce chapitre la réalisation de tout les composants de notre systeme et leur implémenation étape par étape commençant par la création de la base de données, ensuite le service web en utilisant les fichier Json pour pouvoir communiquer entre les plates formes, nous avons vu aussi comment mettre en place le broquer mosquitto et en fin nous avons vu la programmation de la l'application mobile.

Conclusion générale

Notre projet avait pour ambition de réaliser une application de suivi en utlisant le protocole MQTT tout en relevant les défis suivant :

1. La mobilité est aussuré vu que notre application est une application sur téléphone mobile.
2. la fiabilité car notre système repond aux exigence critique de la géolocalisation en utilisant la clé de GOOGLE MAPS API.
3. La gestion de tout les périphériques surtout que le broker de mosquitto admet un grand nombre de clients.
4. La disponibilité grace au service web et au broker mosquito qui sont toujours opérationnelles
5. L'interopérabilité est assuré grace a l'envoie de données via Json pour assurer la communication entre les diférents composants du sytème.
6. La sécurité est assurée par les procédure d'authentification.

Nous avons aussi vu pourquoi avons-nous choisis le protocole MQTT pour l'envoie et la réception des messages entre les utilisateurs. En effet la rapidité et la fluidité étaient remarquable en ajoutant aussi la précision de destination. Il est important a rappeler aussi que nous avons vu que l'implémentation du protocole était très simple et rapide qui ce qui donne un autre point au profit du protocole.

En conclusion nous pourront dire que le protocole MQTT est un protocole simple a utiliser et a implementer et reste l'une des meilleurs solutions a proposer dans le domaine de l'IOT.

Bibliographie

[1] Internet of things : Et d'un point de vue technique, url : http ://www.iplogos.fr.

[2] Schéma de principe de la géolocalisation par gps , https ://fr.wikipedia.org/wiki/gc3a9olocalisation/media/fichier :geolocation.png.

[3] sitionnement par satellites gnss gps galileo ,url =https ://www.reseau-teria.com/2020/01/20/positionnement-par-satellites-gnss-gps-galileo-comment-ca-marche/.

[4] Souscription au topic, url = https ://blog.engineering.publicissapient.fr/2018/04/16/internet-des-objets-quels-protocoles-applicatifs-utiliser-1-2/.

[5] Topologie mqtt par hivemeq url = https ://www.cadlog.fr/tag/iot-fr/.

[6] Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business, August 2014.

[7] Mohamed Abomhara and Geir M. Køien. Cyber security and the internet of things :vulnerabilities [On journal of cyber security, vol. 4,].

[8] Carles Anton-Haro and Mischa Dohler. *Machine-to-machine (M2M) communications : architecture, performance and applications*. Elsevier, 2014.

[9] Zoran B Babovic, Jelica Protic, and Veljko Milutinovic. Web performance evaluation for internet of things applications. *IEEE Access*, 4 :6974–6992, 2016.

[10] Dr Omar Elloumi. Iot ecosystem expands significantly with new global standards.

[11]] Z. B. Babovic et al. Web performance evaluation for iot applicationsieee access, volume 4. November 2016.

[12] Raj Jain. Networking layer protocols for internet of things : 6lowpan and rpl. 2018.

[13] Sang-hyun Kim, Dong-hwi Kim, Hyeung-seok Oh, Hyun-sig Jeon, and Hyun-ju Park. The data collection solution based on mqtt for stable iot platforms. *Journal of the Korea Institute of Information and Communication Engineering*, 20(4) :728–738, 2016.

[14] Stephane Lohier and Dominique Présent. Réseaux et transmissions, 2016.

[15] Tutorials Point. Simply easy learning. *Internet : http ://www. tutorialspoint. com/uml,[January 2013]*, 2011.

[16] Janessa Rivera and R Van der Meulen. Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business. *Connecticut, EEUU : Gartner Group*, 2014.

[17] Tara Salman and Raj Jain. Networking protocols and standards for internet of things. *Internet of Things and Data Analytics Handbook*, 2015 :215–238, 2015.

[18] MQTT Version. 3.1. 1. edited by andrew banks and rahul gupta. 29 october 2014. oasis standard, 2016.

[19] Lucy Zhang. Building facebook messenger, aug 2011. *URL https ://www. facebook. com/notes/facebook-engineering/building-facebook-messenger/10150259350998920*.