BROCHETON Nicolas

Localisation et suivi d'essaims robotiques

Projet de fin d'études

Ensta Bretagne

Promotion 2013 – Option Robotique

Résumé

La robotique en essaim se présente comme un champ prometteur de la robotique. Cette discipline met en œuvre des robots aux capacités limitées et éventuellement spécialisés dans l'exécution de certaines tâches, néanmoins en mesure d'interagir avec leurs congénères afin d'adopter des comportements d'entraide et de coopération. Elle offre une alternative intéressante aux divers robots généralistes développés ces dernières années à l'instar des robots d'exploration de Mars dont le robot Curiosity demeure le fleuron. L'emploi d'un tel nombre de robots simultanément implique le développement d'outils performants de localisation et de suivi en temps réel ou différé des éléments constituant ces meutes à des fins d'analyse des performances et des comportements.

Le projet présenté dans ce rapport vise à apporter une solution technique à cette problématique. Une première solution a d'abord été mise au point. Elle repose sur l'emploi d'une base de données, d'un client de visualisation web des positions des robots dans le temps et d'un système de balises embarquées sur les robots, en la matière des smartphones Android qui regroupent les diverses spécifications du projet et permettent d'envoyer les informations de positionnement des robots à la base de données via le réseau GSM. Dans un second temps, une API nommée Swarmon API, a été ajoutée au système initial afin d'en garantir un fonctionnement plus efficace et des possibilités d'interaction et de modification plus conséquentes. Cette deuxième architecture a offert au système et à ses futurs utilisateurs de nouvelles fonctionnalités, en à augmenter la fiabilité et devrait en assurer la pérennité du fait de facilités de gestion, de manutention et d'adaptation à de futurs projets mettant en œuvre un grand nombre de robots et nécessitant un outil performant de suivi.

Abstract

Robotic swarms are currently a promising field of robotics. It allows robots with limited capacities and often specialized in certain tasks to interact with their neighbors to adopt support and cooperation behaviors. This domain offers an interesting alternative to the non specialized robots developed during this decade like Curiosity, the last Mars Exploration rover. The use of such a number of robots involves the development of effective tools for tracking and tracing in real time these packs in order to analyze its performances and behaviors.

This project aims to provide a technical solution to this problem. A solution was first developed based on a database, a web client to visualize the successive positions of robots and an embedded tracking system on robots, in this case Android smartphones that meet project specifications and are used to send robots positioning information to the database via the mobil network. In a second step, an API called Swarmon API has been added to the system to ensure a more efficient working and more possibilities of interaction and modification. This second architecture has provided new functionalities to the system and its future users, increased its reliability and should sustain more numerous management, handling and adaptation abilities useful to future projects involving a large number of robots and requiring a powerful monitoring tool.

Remerciements

Je tiens à exprimer toute ma reconnaissance à Messieurs Jaulin et Reynet, mes tuteurs de stage pour le temps et l'attention qu'ils m'ont consacré tout au long de ce stage. Je les remercie de m'avoir encadré, orienté, parfois aidé et souvent conseillé. Vous avez largement contribué à faire de ce projet ce qu'il est désormais, une architecture opérationnelle de localisation et de suivi de meutes organisées ou non de robots.

Je souhaite également remercier tous les professeurs et personnes qui, au travers de leurs propos, de leurs écrits, de leurs conseils et de leurs critiques, ont guidé mes réflexions et donné à ce projet son visage actuel.

J'adresse enfin mes sincères remerciements à tous ceux qui, professeurs ou non, militaires ou non, ont participé de près comme de loin à faire de ma scolarité ce qu'elle a été, un important moment de formation professionnelle et personnelle.

Sommaire

Introduction

Un projet de fin d'études marque clairement le dernier pas d'une vie académique et du fait de ses tenants et aboutissants, un pas supplémentaire vers le monde professionnel qui nous tend désormais les bras. La distinction entre ces deux univers n'a cessé de s'amenuiser au fur et à mesure de ma scolarité à l'Ensta Bretagne, les années de formation tendant à largement ouvrir et immerger les étudiants dans le monde industriel et professionnel. Ce stage, ultime étape, vise à concrétiser l'achèvement d'un temps, celui de l'apprentissage au profit de l'avènement d'un autre, celui de la mobilisation et de la mise en pratique de connaissances parfois durement acquises.

Ce stage et le projet qui lui est associé sont à l'image de ce concept, ils m'ont permis de mobiliser un certain nombre des connaissances et des reflexes acquis au cours de ma formation. Ils s'inscrivent dans une thématique relativement récente, la robotique en essaim. Cette discipline inspirée des insectes sociaux, de leurs comportements et des mécanismes qui régissent leurs sociétés, vise à faire évoluer simultanément des robots à l'intelligence et aux fonctionnalités individuelles souvent limitées mais qui, collectivement à l'instar des fourmis, deviennent en mesure de réaliser des taches élaborées et complexes. Faire se mouvoir et interagir de la sorte un grand nombre de robots nécessite des outils de monitoring et de suivi relativement efficaces. C'est à cette problématique de suivi de meutes de robots que ce projet de fin d'études cherche à apporter une réponse technique fiable et performante. Bien que doté d'une problématique assez générale, ce projet aura des applications très concrètes dont le monitoring et le tracking des robots voiliers évoluant lors de la World Robotic Sailing Championship, organisée cette année par l'Ensta Bretagne. D'autre part si la solution implémentée se veut générale, ce futur champ d'emploi a nécessairement orienté certains des choix techniques réalisés, le présent rapport en montrera forcement l'influence.

Afin de répondre à ce besoin de suivi et de contrôle d'une meute de robots, la solution mise en œuvre repose sur des technologies Web accessibles en ligne. Par conséquent, le lecteur est invité à se rendre compte par lui-même du travail réalisé et du rendu final en se connectant aux différentes pages mentionnées plus à même d'exposer le travail réalisé que de simples images ou captures d'écran.

Bâtir une telle solution implique de suivre diverses étapes. Ainsi, dans un premier temps, ce rapport reviendra sur le contexte et les spécifications du projet afin d'en expliciter les enjeux techniques. Dans un second temps, l'élaboration d'une première solution technique sera présentée. Face aux limites de cette dernière en termes d'évolutivité et de performances, une deuxième approche sera ensuite privilégiée et implémentée via la mise au point d'une API dédiée. Une dernière partie viendra clore ce rapport en évoquant tout à la fois les résultats de ce projet, ses limites, ses points forts et ses faiblesses, ses pistes d'amélioration ainsi que ses conditions de mise en œuvre et d'adaptation à la World Robotic Sailing Championship (WRSC).

I. Contexte et spécifications

La présente partie vise à préciser le cadre de ce projet de fin d'études, à le replacer dans son contexte et à en établir les tenants et aboutissants aussi bien sur le plan technique qu'en termes de réalisation attendue. Intitulé "Problématique de localisation de meutes de robots", ce travail vise à mettre au point un système performant de localisation en temps réel d'une meute de robots et des outils efficaces et modulables d'observation de leur répartition. Il s'inscrit dans un contexte relativement précis, l'organisation par l'Ensta-Bretagne de l'édition 2013 de la World Robotic Sailing Championship (WRSC) ([1], [2]).

1. Contexte

Le monde actuel de la robotique est en plein essor, les avancées techniques en matière de miniaturisation, d'intelligence artificielle, de capacité de calcul des puces et circuits embarqués mais aussi de stockage de l'énergie ont permis de réaliser des robots aux caractéristiques, à l'autonomie, à l'indépendance et aux capacités croissantes. Des robots comme le robot Curiosity en sont les parfaites illustrations. Parallèlement à cet essor de robots uniques, ultraperformants, multitâches et souvent extrêmement complexes, la robotique en essaim a peu à peu gagné en importance et en consistance. Il s'agit d'un domaine récent de la robotique qui s'intéresse aux systèmes multi-robots auto-organisés et constitués d'un certain nombre de robots identiques ou spécialisés. Les commandes, les consignes et l'intelligence de telles meutes ou essaims de robots suivant leur nombre peuvent être distribuées ou centralisées selon les stratégies implémentées et les approches privilégiées. De tels systèmes permettent un gain considérable en robustesse et en résilience, la panne d'un élément n'entrainant pas nécessairement la perte d'une capacité puisque d'autres éléments de l'essaim peuvent posséder cette fonctionnalité et via des motifs de coopération largement dépasser les possibilités de manœuvre d'un système unique à l'instar de ce qui existe dans le monde animal, en particulier chez les insectes sociaux. Dans un tel contexte, il peut être relativement intéressant pour un opérateur distant et passif de pouvoir suivre le comportement des robots de l'essaim à des fins de contrôle bien sûr, mais aussi de détection d'erreurs individuelles ou collectives, ou encore l'identification de stratégies particulièrement adaptées à des situations fluctuantes.

Figure 1 : Logo de la WRSC édition 2013

L'organisation par l'Ensta-Bretagne de l'édition 2013 de la World Robotic Sailing Championship ([1]) qui aura lieu en septembre s'inscrit assez bien dans cette problématique de localisation et de suivi de meutes de robots. Cet événement a mis en exergue le besoin de connaitre en temps réel la position des différents robots voiliers, de leur mise à l'eau à leur retour au port en passant par leurs diverses phases d'évolution. Ce besoin peut se justifier de quatre manières bien distinctes. Premièrement, la connaissance en permanence de la position de chaque robot autonome offre des garanties de sécurité supplémentaires non négligeables qui viennent partiellement combler un actuel vide juridique en matière d'évolution et de mise à l'eau de drones marins. Deuxièmement, une telle solution de tracking permettrait de simplifier le travail du jury concernant la validation ou non de certaines épreuves telles le suivi d'une trajectoire ou l'évitement d'un obstacle fixe. Elle introduit par ailleurs une plus grande équité dans l'attribution des notes, le jury ne pouvant être simultanément attentif à tous les bateaux, il serait alors en mesure de revisualiser le parcours réalisé. En outre, les

tracés obtenus pourraient servir de preuve en cas de litige ou de contestation des résultats par l'une ou l'autre des équipes. Troisièmement, d'un point de vue technique, une telle solution devrait permettre aux équipes d'analyser leur prestation et d'en visualiser les points forts et les éventuelles lacunes. Dernier aspect pratique, ce projet pourrait permettre de montrer aux équipes à terre et au grand public les pérégrinations des robots des autres équipes et ainsi les autoriser à suivre en direct les évolutions de bateaux nécessairement au large et donc pas forcement entièrement visibles depuis la côte.

Dans l'histoire de la WRSC, une telle formule de tracking a déjà été employée ([3]). Basée sur une solution commerciale de suivi de flottes de véhicules terrestres adaptée au milieu marin et au contexte particulier des robots voiliers, elle semble avoir donné d'intéressants résultats mais pour diverses raisons n'a pas été reconduite les années suivantes. Ce projet de fin d'études (PFE) s'il aboutit à une solution viable techniquement pourrait permettre d'institutionnaliser le procédé dans l'organisation et le déroulement des prochaines compétitions de la WRSC et contribuerait ainsi au rayonnement de l'école. D'autre part, bien que développé pour et dans le cadre de la WRSC, ce projet est loin de se limiter au simple cadre de cette compétition ou même au milieu marin. En effet, cette solution dépasse le contexte de la robotique marine et vise à être facilement adaptable au contrôle et au suivi de meutes de drones terrestres ou aériens disposant d'un accès fiable à des solutions de géolocalisation.

2. Contexte du stage

Avant d'analyser plus en détail les exigences et les spécifications techniques du projet, une brève présentation du lieu du stage s'impose. Ce stage s'est déroulé au sein du pole STIC de l'Ensta Bretagne.

Figure 2 : L'Ensta Bretagne

Il s'agit d'un établissement public d'enseignement supérieur et de recherche sous tutelle de la Direction Générale de l'Armement (DGA) et du ministère de la Défense, un statut qu'elle partage avec 3 autres écoles : l'Ecole Polytechnique, l'ISAE (Institut Supérieur de l'Aéronautique et de l'Espace) et l'ENSTA ParisTech. Elle a vocation à former des ingénieurs civils et militaires dont les compétences répondent aux exigences des industries mécanique, électronique et informatique les plus innovantes. Créée en 1971, l'ENSTA Bretagne (anciennement ENSIETA) était destinée, à l'origine, à former exclusivement des ingénieurs militaires pour les besoins du ministère de la Défense. L'école s'est ouverte aux civils à partir de 1990 et la montée en puissance des effectifs civils s'est faite rapidement en parallèle à une décroissance des effectifs militaires liée aux restructurations de l'appareil de défense. Au cours de la même période, l'ENSIETA fait son entrée dans le monde de la recherche et accueille en 1992, son premier enseignant-chercheur. Depuis 2006, l'école dispense une formation d'ingénieur classique (cycle ENSI) et une formation d'ingénieur par alternance (cycle FIPA). En décembre 2010, l'école rejoint le groupe ENSTA et devient l'ENSTA Bretagne. Outre la formation initiale d'ingénieurs, l'ENSTA Bretagne propose diverses formations diplômantes et de nombreux stages de formation continue. Enfin, elle organise également des formations spécifiques au plan national pour la DGA et au plan international dans le cadre d'accords de coopération.

Juridiquement parlant, l'ENSTA Bretagne bénéficie du statut d'EPA (Etablissement Public à caractère Administratif), doté de l'autonomie administrative et financière, sous tutelle du ministère de la Défense. Son organisation et son fonctionnement sont régis par le Code de la Défense. Les activités de R&D menées par l'établissement au sein de ses laboratoires sont principalement axées sur les sciences et technologies de l'information et de la communication ainsi que sur la mécanique des structures navales. Elle est ainsi administrateur du "GIS Europôle Mer" et des pôles de compétitivité «Mer Bretagne» et «ID4CAR». L'Ensta Bretagne est également membres des pôles de compétitivité «Image et réseaux" et "EMC2".

Plus spécifiquement, le pôle Automatisme et Robotique est particulièrement actif dans le milieu de la robotique marine avec des projets importants développés en partenariat avec des institutions étatiques ou des entreprises. On citera à titre d'exemple le robot voilier VAIMOS développé en partenariat avec l'IFREMER ou le bon classement cette année d'une équipe du département au concours SAUC-E organisé par l'OTAN ([4]) et qui met en œuvre des SUV.

En tant qu'élève de l'Ensta-Bretagne ayant choisi la spécialité « Robotique et systèmes embarqués », le projet de monitoring et de tracking de meutes de robots qui constitue la problématique de mon PFE s'inscrit dans la thématique des travaux menés par le laboratoire d'une part et d'autre part dans la continuité directe de mes choix passés . Par conséquent, sous la direction de mon tuteur de stage M. Jaulin et avec le soutien de M. Reynet, tous deux enseignants à l'Ensta-Bretagne, le pôle robotique s'est avéré un lieu de travail particulièrement adapté à une telle réalisation.

3. Analyse du problème et de l'existant

La mise au point d'une solution de tracking en temps réel de plusieurs robots se heurte à divers obstacles et problématiques. Tout d'abord, il est nécessaire de définir la portée du projet et ses applications directes. Ce projet doit dans un premier temps servir lors de la WRSC afin de garantir le suivi des robots voiliers, permettre d'en visualiser depuis la côte les évolutions et offrir des moyens de validation des épreuves. Cependant outre ce contexte de développement et malgré son caractère relativement fort et spécialisé, il est important que le rendu final du projet puisse servir dans d'autres projets de suivi de meutes de robots, une caractéristique qu'il me faudra garder à l'esprit tout au long de l'édification de ce projet. La finalité est donc de pouvoir enregistrer et afficher, en temps réel ou différé, les positions successives d'un groupe de robots. D'un point de vue technique, il s'agit également de définir ce qu'on entend par temps réel et donc la fréquence d'obtention des données de positionnement et du rafraîchissement de celles affichées ou retournées à l'utilisateur. Par ailleurs, il faut définir les protocoles de communication entre les différentes entités composant ce projet afin de garantir la continuité de l'obtention de telles informations ainsi que leur fiabilité. Enfin, outre l'envoi de données à assurer et les protocoles de communication à définir, il s'agit de construire une solution qui demeure fonctionnelle y compris dans le cadre marin ou, du moins, dans celui de la navigation côtière. Le système qu'il s'agit de mettre au point s'apparente globalement à un dispositif de suivi de flottes. On distinguera donc la partie acquisition de la position de la partie traitement et affichage des données dans une interface client dédié ou non. De tels systèmes existent d'ores et déjà concernant ces deux parties, le paragraphe qui suit s'attache à présenter un bref et succinct tour d'horizon en la matière.

Comme mentionnées précédemment des solutions commerciales répondant partiellement à cette problématique de localisation de meutes de drones existent principalement pour le suivi des flottes de véhicules de transport ou de service. On n'en observe pour ainsi dire qu'une unique implémentation pour le monde marin, le système AIS, qui permet de connaitre en temps réel la position de tous les bateaux équipés dans le monde ([5]). Un tel système, en plus d'être partiellement inadapté du fait de ses fonctionnalités, ne se justifie certainement pas dans le cadre défini pour cette compétition. Les systèmes de suivi de flottes de véhicules existent désormais depuis plus d'une vingtaine d'années. D'abord, dédiés aux entreprises de transport de fret, ils se sont depuis étendus à tous les secteurs de l'industrie et de la société. On peut désormais suivre les véhicules de son entreprise mais aussi sa cargaison, son courrier, … sans même évoquer les diverses fonctionnalités supplémentaires offertes par les entreprises du secteur en matière de contrôle et d'observation des caractéristiques du véhicule, du comportement du conducteur, … . Ces systèmes basés sur la technologie GPS permettent de connaitre en temps réel la position de ses véhicules quelle que soient leur nature (voitures, camions, navires, …) et de l'afficher dans des logiciels propriétaires plus ou moins évolués. Le schéma ci-dessus emprunté à l'entreprise Dolink ([6]) résume assez bien le fonctionnement d'un tel système. La puce GPS incorporée dans le dispositif embarqué

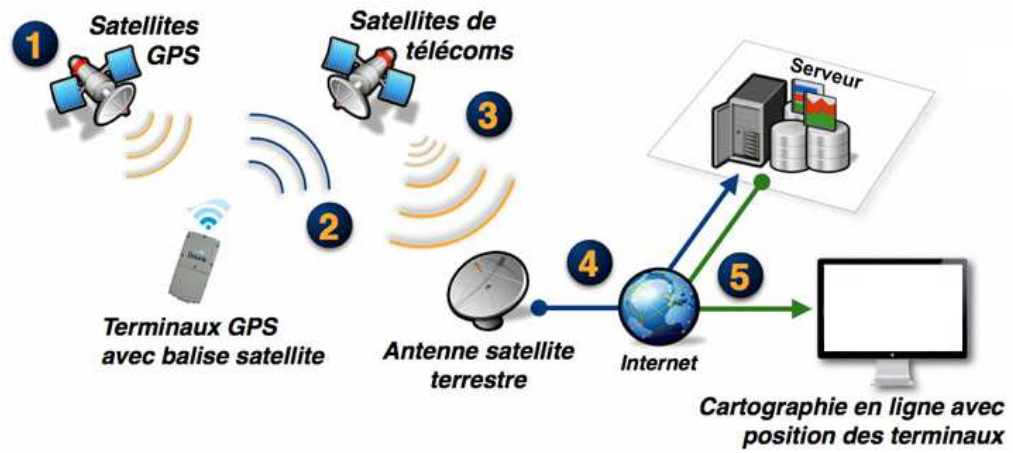


Figure 3 : Schéma du fonctionnement d'un système classique de suivi de flottes

de tracking détermine la position GPS du véhicule, puis l'envoie via des satellites de télécommunications qui transmettent le message de géolocalisation à des antennes dédiées situées aux quatre coins du monde. Celles-ci traitent et acheminent ensuite le message via Internet. Il peut alors être renvoyé vers un autre réseau (GSM, …) voire transformé (emails, SMS, appels téléphoniques préenregistrés, …). Des entreprises comme TomTom ([7]) ou MobilFleet ([8]) proposent un service de qualité à un coût variable. Au niveau des IHM et clients de visualisation, la plupart repose sur le service de cartographie de Google ([9]) qui a largement contribué à populariser les services de cartographie en ligne et présente une ergonomie et une qualité variables. De telles solutions sont certes fiables et faciles d'emploi cependant couteuses et pas nécessairement complètement adaptées à notre problématique de suivi de meutes de robots. Deux autres solutions proches et au format plus modeste, sont le système Spot qui est similaire dans son fonctionnement au système précédemment évoqué mais qui vise à être déployé par des individus ou de petits organismes, et une solution assez proche de celle souhaitée, développée par la société KaliBee ([10]). Cette dernière nommée Isea3D met à disposition du grand public et des organisateurs de régates un système de balises GPS reliées au réseau GPRS et offre surtout de visualiser les données recueillies

au sein d'un moteur virtuel en 3D accessible depuis n'importe quel navigateur Internet classique. Une particularité particulièrement intéressante demeure de proposer aux utilisateurs de charger leurs données de position acquises à l'aide d'un appareil déjà existant tels un GPSLogger, un loggerNMEA ou un PC, embarqué a bord du voilier. Par ailleurs, le service proposé par cette société inclue également une solution de localisation basée sur l'emploi d'un smartphone au travers d'une application Android. Cette dernière solution est de loin la plus proche de ce que l'on souhaite mettre en place. De plus, cette possibilité de provenance multiple des positions est une caractéristique intéressante à étudier et à essayer de conserver. La découverte in situ de leur système lors de la 45ème édition de la course croisière EDHEC organisée à Brest du 19 au 27 avril 2013, a montré au vue des technologies mises en œuvre la faisabilité technique d'un tel projet tout en mettant en exergue l'intérêt pour l'Ensta Bretagne de développer sa propre solution. Outre cette faculté à développer de tels systèmes de géolocalisation dans les délais impartis et son intérêt à opter pour cette solution, divers points viennent confirmer l'évidence d'un tel choix. D'une part, le coût facturé de la prestation ne se justifie pas pleinement au vu des compétences techniques possédées par l'école. A titre d'illustration, la solution proposée par la société Kalibee se chiffre à plusieurs centaines d'euros par jour pour une dizaine de dispositifs de tracking et l'accès au service de visualisation des trajectoires. D'autre part, l'implémentation d'une solution propriétaire autoriserait la maitrise totale des fonctionnalités de l'outil tout en en assurant l'éventuelle pérennité ainsi que son adaptation et son réemploi dans d'autres projets robotiques. Ainsi, une telle approche procure de nombreux avantages. D'une part, elle assure d'avoir un produit correspondant pleinement aux attentes spécifiées, et d'autre part, elle garantit une solution dont l'ensemble de la chaine de production et de développement sera maitrisé, le produit final pourra donc sans problème de licence ou de droit être modifié et adapté à d'autres projets.

Ainsi, bien que des solutions techniques et commerciales préexistent, l'école et plus largement les organisateurs de la WRSC gagneraient largement sur les plans promotionnels, financiers, logistiques et techniques à disposer d'un outil ouvert de suivi des robots facile à adapter et à faire évoluer. Son caractère non spécialisé au sens de réemployable, modifiable et non propriétaire devant être un gage d'une éventuelle longévité et d'un apport certain aux futures éditions. Pour l'école même, un tel projet s'il aboutit, viendrait constituer une brique logicielle intéressante dans de nombreux projets tels le « jeu des buggies » actuellement en cours de développement à l'Ensta Bretagne et qui vise à mettre en œuvre une meute de robots terrestres. Ces différentes caractéristiques justifient largement l'existence et les objectifs de ce projet tout en en clarifiant les tenants et les aboutissants.

4. Objectifs, spécifications et contraintes

Il s'agit désormais de définir les futures caractéristiques du projet, d'identifier les contraintes extérieures auxquelles il sera soumis et de définir une première architecture répondant aux besoins exprimés.

Le but de ce projet est donc l'implémentation d'une solution de tracking en temps réel d'une meute désorganisée de robots. A ce titre, il comporte deux parties, l'une tournée vers l'acquisition de la position des robots et l'autre destinée à traiter et stocker ces données. Cette distinction sera abordée par la suite.

Par ordre de priorité, on distingue les objectifs suivants, eux-mêmes pouvant être subdivisés en sous objectifs :

- Acquérir la position GPS des robots.
    - Disposer d'un module GPS embarqué.
    - Récupérer les trames NMEA émises par le GPS.
    - Enregistrer localement les données à des fins de sécurité et de redondance de l'information.
    - Garantir l'obtention des données de géolocalisation toutes les x secondes, le paramètre x devant pouvoir être modifié.
- Transmettre à un système central les données de localisation à des fins de traitement et d'affichage.
    - Définir les données à envoyer et leur format.
    - Mettre au point un protocole de communication.
    - Disposer au niveau des systèmes embarqué et central de modules de communication compatibles.
- Enregistrer les données reçues dans un système de stockage dédié et adapté (fichiers, bases de données, …).
- Créer un client de visualisation des données.
    - Etre en mesure d'afficher en temps réel ou différé les positions des différents robots voiliers sur un moniteur.
    - Assurer un niveau d'interactivité et de convivialité correct
- Garantir l'adaptabilité et la modularité du service et du programme au travers de modules pouvant être activés et/ou ajoutés facilement.

A ces multiples objectifs s'ajoutent de multiples contraintes. Tout d'abord, une contrainte liée au milieu d'emploi. Ce projet a en effet pour finalité directe d'être employé en mer ou du moins dans une zone humide. Par conséquent, le module embarqué devra pouvoir résister à d'éventuelles projections d'eau voire à une immersion temporaire, l'étanchéifier constituerait un trait intéressant offrant des garanties supplémentaires de fonctionnement et de longévité du système. En outre, une telle zone d'emploi s'avère relativement restrictive sur les systèmes de communication pouvant être mis en œuvre. Pour des raisons de simplicité, de possibilités et de conditions d'emploi, le système se limitera aux zones côtières où, à défaut de posséder une connexion Internet via un module 3G ou Wifi, le réseau GSM/GPRS demeurera accessible. Outre cette contrainte technique, une contrainte temporelle prédomine. Il s'agit d'avoir en fin de projet une solution fiable et viable à présenter afin qu'elle puisse être utilisée lors de la WRSC 2013. Ainsi, tout au long du développement, il me faudra garder à l'esprit le caractère nécessairement fonctionnel de ce projet et les attentes des utilisateurs finaux susceptibles, par ailleurs, d'évoluer au fur et à mesure de l'établissement des règles de cette édition de la WRSC, d'où la nécessité réelle d'une importante phase de tests et de retour sur expérimentation. A ces objectifs initiaux s'ajoute également le développement d'éventuels modules supplémentaires autorisant par exemple l'envoi d'instructions à la balise et leur exécution par le robot en fonction du degré de complexité et d'aboutissement atteint par le programme.

Pour parvenir à de tels résultats, une première architecture a été définie. La solution envisagée se base sur l'embarquement par chaque bateau d'une balise GPS disposant d'un module GSM/GPRS dont les facultés d'envoi des données restent plus importantes qu'une connexion

internet au travers par exemple d'un point d'accès wifi local ou d'une connexion 3G non garantie pour le moment sur l'ensemble de la communauté de communes de Brest et encore moins au milieu de la rade. Ce module de positionnement communiquera à intervalles fixes sous forme de SMS, les positions successives du bateau qui, après traitement, seront enregistrées dans une base de données et affichables sur une carte telle celles fournies par Google Maps ou l'IGN.

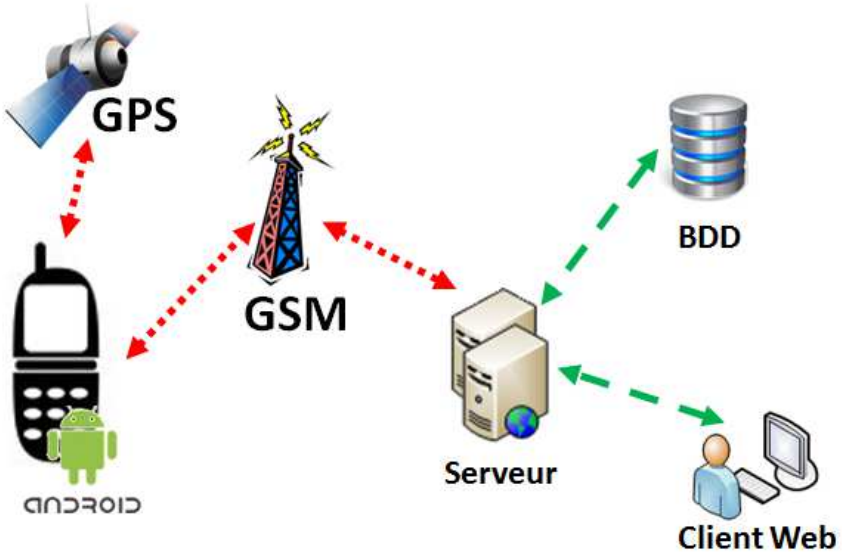


Figure 4 : Architecture globale du système de suivi

Concrètement, au vue des spécifications de la balise et afin de garantir son fonctionnement dans les délais impartis, il a été choisi d'utiliser dans un premier temps en guise de balise, un smartphone sous Android ([11]). Ce dernier, véritable concentré de technologies, réunit en effet toutes les fonctionnalités nécessaires à la mise au point de notre système de tracking. De plus, il offre des possibilités d'évolution non négligeables. Un tel choix tend à transformer une problématique de développement initialement tournée vers le hardware vers une implémentation résolument software car logicielle. En effet, ces appareils possèdent un système d'exploitation et c'est au travers de ce dernier que l'on vient interagir avec les différents capteurs embarqués. L'idée évoquée dans un premier temps de construire un prototype basé sur une carte Arduino ([12]), des modules GPS et GSM reste une piste à explorer dans le futur bien que son prix de revient demeure, pour une production unitaire, de l'ordre de celui d'un smartphone de deuxième génération. Le reste du projet à savoir le client de visualisation des données se traduit par une application de rendu web assistée dans son travail par un serveur d'application qui vient exploiter et post-traiter les informations enregistrées dans une base de données. En outre, le serveur ou un service tiers devra permettre de décortiquer les SMS envoyés par les balises et enregistrer leur contenu dans la base de données.

5. Planification et déroulement du stage

Le projet s'organise autour des grandes étapes présentées dans le diagramme de Gantt ci-dessous. Du fait des multiples technologies qu'il implique, différents langages et mécanismes de programmation devront être appréhendés et compris. Ainsi, des bases en PHP, JavaScript, Html et Css seront nécessaires pour mener à bien l'élaboration du portail internet. En outre, l'application embarquée sur le téléphone balise sera développée en Android et devra être exempte de bugs et peu gourmande en énergie, sa déclaration en tache de fond pourrait contribuer à cet état de fait. D'autre part, afin de garantir l'obtention d'un service fonctionnel en fin de projet, j'ai souhaité

concevoir au plutôt un démonstrateur. Une telle approche m'a permis une importante phase de tests et a facilité d'éventuels corrections et ajouts de fonctionnalités. Globalement, les taches bien que liées, ont eu un développement concomitant toujours dans cette optique de test et d'expérimentation. Ainsi, fort d'une base au fonctionnement connu et validé, les divers systèmes composant ce projet ont été progressivement complexifiés et se sont vus peu à peu adjoindre de nouvelles capacités.

	Nom de la tâche	Durée	Début	Fin
1	**Avant-projet**	**15 jours**	**Jeu 11/04/13**	**Mer 01/05/13**
2	Decouverte du projet	4 jours	Jeu 11/04/13	Mar 16/04/13
3	Analyse de l'existant	5 jours	Lun 15/04/13	Ven 19/04/13
4	Specification des exigences	10 jours	Lun 15/04/13	Ven 26/04/13
5	Choix d'une solution technique	7 jours	Jeu 18/04/13	Ven 26/04/13
6	Redaction d'un cahier des charges	3 jours	Lun 29/04/13	Mer 01/05/13
7				
8	**Projet**	**70 jours**	**Lun 22/04/13**	**Ven 26/07/13**
9	**Construction du portail Web**	**38 jours**	**Mer 24/04/13**	**Ven 14/06/13**
10	Mise au point de l'architecture	10 jours	Mer 24/04/13	Mar 07/05/13
11	Creation de la base de données	14 jours	Mer 24/04/13	Lun 13/05/13
12	Appropriation de l'API SMS	10 jours	Lun 29/04/13	Ven 10/05/13
13	Premiere mise en ligne	0 jour	Lun 06/05/13	Lun 06/05/13
14	Amélioration du site	23 jours	Mer 15/05/13	Ven 14/06/13
15	**Implementation du logiciel embarqué**	**30 jours**	**Lun 22/04/13**	**Ven 31/05/13**
16	Acquisition des données capteur	10 jours	Lun 22/04/13	Ven 03/05/13
17	Envoi des trames via le réseau GSM/GPRS	10 jours	Lun 22/04/13	Ven 03/05/13
18	Sauvegarde des données en interne	10 jours	Lun 22/04/13	Ven 03/05/13
19	Creation d'une interface	22 jours	Jeu 02/05/13	Ven 31/05/13
20	**Phase de tests**	**55 jours**	**Lun 29/04/13**	**Ven 12/07/13**
21	Tests unitaires	10 jours	Lun 29/04/13	Ven 10/05/13
22	Validation des fonctionnalités	3 jours	Jeu 02/05/13	Lun 06/05/13
23	Tests système V1	24 jours	Mar 07/05/13	Ven 07/06/13
24	Tests système V2	15 jours	Lun 24/06/13	Ven 12/07/13
25	**Phase d'amelioration**	**60 jours**	**Lun 06/05/13**	**Ven 26/07/13**
26	Retour sur experimentation	30 jours	Lun 06/05/13	Ven 14/06/13
27	Mise au point de modules	30 jours	Mer 22/05/13	Mar 02/07/13
28	Ajout de fonctionnalités	53 jours	Mer 15/05/13	Ven 26/07/13
29				
30	Redaction du rapport	73 jours	Mar 07/05/13	Jeu 15/08/13
31	Date de rendu du rapport	0 jour	Jeu 15/08/13	Jeu 15/08/13
32	Preparation de la soutenance	22 jours	Jeu 01/08/13	Ven 30/08/13
33	Soutenance	0 jour	Lun 02/09/13	Lun 02/09/13

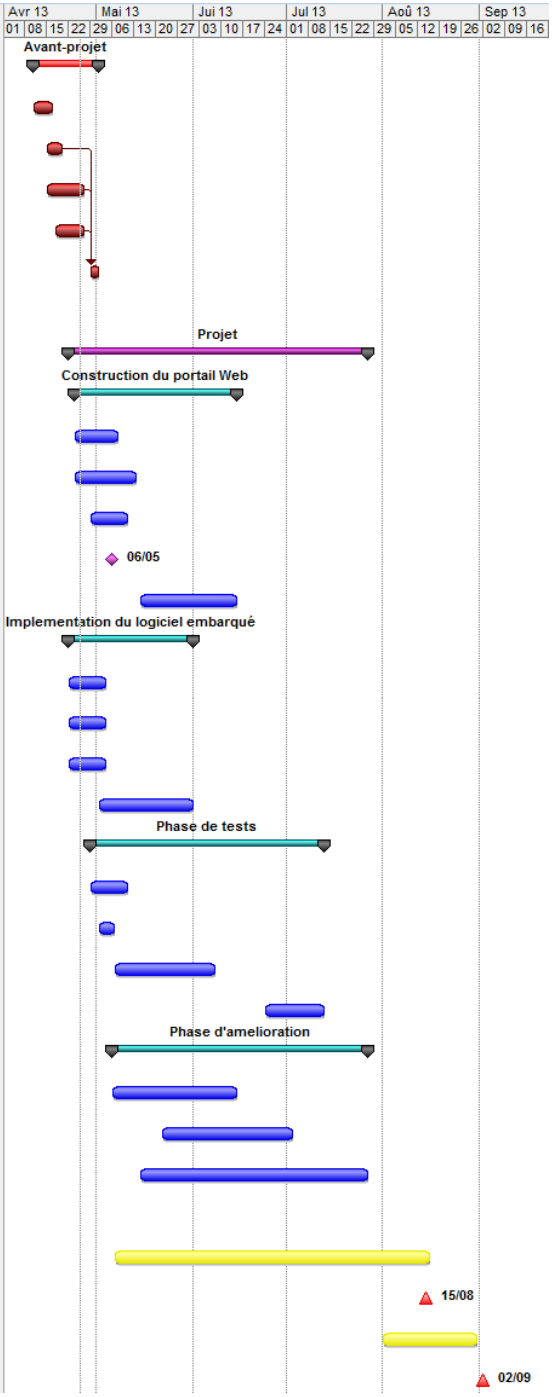


Figure 5 : Diagramme de Gantt

Dans les faits, le programme établi en début de stage a été globalement suivi. Une fois le projet appréhendé et les attentes spécifiées, une première architecture a été implémentée durant les premiers mois permettant de disposer rapidement d'un démonstrateur. Ce dernier a alors pu être

testé, ce qui a permis de mettre en exergue des erreurs de programmation et certains problèmes de fonctionnement. Il a aussi mis en lumière les lacunes et les faiblesses de la première architecture développée. Par conséquent, la phase d'amélioration initialement prévue a en fait consisté en une refonte du système au travers de la mise au point d'une API, puis à la restructuration de l'architecture du système et à l'adaptation des différents modules préexistants à cette nouvelle mouture.

A l'heure où ces lignes sont écrites, le système dans sa version beta est entré dans sa dernière phase de test prévue jusqu'à la fin du mois d'aout. Les résultats actuels des expérimentations et l'ensemble des tests réalisés tout au long du développement de ce projet m'amène, malgré l'existence d'encore quelques bugs et autres soucis techniques minimes, à envisager avec une certaine confiance la validation finale du projet et son caractère pleinement opérationnel pour la rentrée prochaine et la WRSC.

6. Moyens humains, financiers et matériels

Ce projet s'inscrit dans la préparation et l'organisation de la WRSC par l'Ensta-Bretagne. A ce titre, je dispose d'un possible soutien technique de la part de Mael Melguen, coresponsable de l'organisation de cet événement et réalisateur du site sur lequel ce projet devrait venir se loger. Les différents enseignants du pôle STIC, dont certains fortement intéressés par le traitement de cette problématique, constituent également un atout et une réelle source d'aide pour toutes les questions techniques portant sur les domaines de la robotique, de l'électronique, des applications Web ou plus généralement sur la WRSC et son déroulement.

La solution finale se veut peu couteuse. Cependant diverses possibilités de financement existent selon les besoins manifestés et la finalité des composants ou produits achetés. Ainsi, une partie du budget de la WRSC pourra être employée au développement de cette solution et surtout à sa mise en œuvre finale. En parallèle, le club robotique en plus de mettre à ma disposition des moyens techniques divers (cartes électroniques, poste de soudure, téléphones Android, …) est en mesure d'investir dans des équipements dont les capacités lui apparaissent intéressantes ou réutilisables dans un avenir plus ou moins proche.

Les contours du projet sont désormais bien définis, différentes contraintes et spécifications ont été identifiées et un calendrier des tâches à effectuer a été établi, il ne reste donc plus qu'à mettre au point l'architecture initiale envisagée. L'abondance des moyens et autres ressources mobilisables ne devant pas constituer un frein au développement de ce système.

II. Mise au point de la première architecture

L'analyse de l'existant et la définition des spécifications du système ont conduit à la définition d'une première architecture. Elle s'organise autour d'une page Internet pour afficher les données, une base de données pour les stocker et un ensemble de balises. Ce design comporte donc deux parties distinctes, un système déporté d'acquisition des positions GPS sous forme d'une application dédiée installée sur un smartphone Android d'un coté et de l'autre coté, un ensemble de structures Web dont un serveur de stockage en vue d'enregistrer les données envoyées par le dispositif embarqué et un client dédié logiciel ou accessible depuis un navigateur afin de les afficher et de les visualiser.

1. Développement du client lourd

1.1. Contraintes techniques – Outils employés

Face aux objectifs et aux diverses contraintes techniques, il s'agit d'implémenter une page internet capable d'afficher en temps réel ou différé, les trajectoires réalisées par les robots. Pour parvenir à un tel résultat, mes choix se sont orientés vers les langages HTML et PHP ([13], [14]) pour la structure de la page et les actions au chargement puis vers un célèbre service de cartographie, Google Maps, pour le fond de carte et son API JavaScript pour pouvoir interagir avec ([15]). Le choix de Google Maps peut sembler discutable puisqu'il s'agit d'une solution propriétaire et à ce titre limitée dans son utilisation. Cependant, son emploi pour un débutant est facilité par son caractère répandu et par sa documentation riche et d'une grande clarté. En outre, dans un premier temps, ce service répond complètement à mes besoins. Ce choix effectué, il va impliquer l'emploi de la technologie web Ajax ([16]) afin de mettre à jour les données représentées de manière dynamique et non via un rechargement de la page qui entrainerait l'utilisation d'un jeton de chargement de l'API Google maps dont le nombre bien qu'important demeure limitant (25000 chargements par site et par jour ([17])). En outre, cela délègue la tache de mises à jour des données au navigateur du client ce qui ne peut qu'alléger la charge de travail du serveur. L'architecture mise en place au final est la suivante (cf. figure n°6).

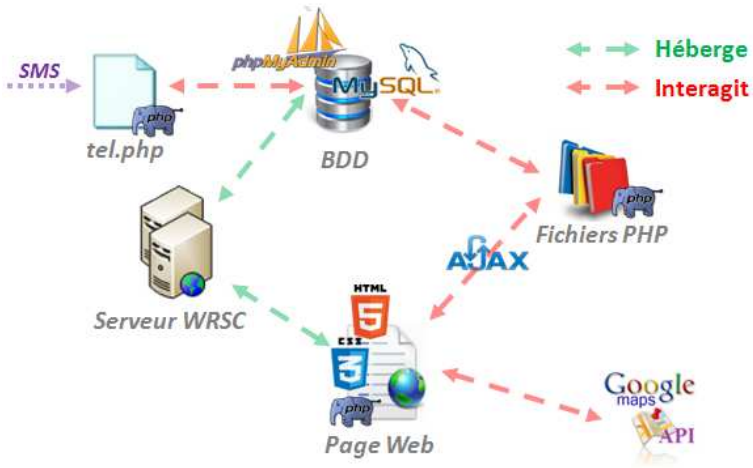


Figure 6 : Schéma de l'architecture Web

La page rédigée en PHP et en HTML affiche un fond de carte et un menu. L'utilisateur interagit avec cette page via des fonctions JavaScript et c'est également via des fonctions JavaScript qu'il charge les données stockées dans une base. Le fonctionnement de ces dernières se fondent sur la technologie Ajax et fait alors appel à une bibliothèque de fonctions écrites en PHP qui ont à charge d'interagir directement avec la base de données. A ce titre, elles contiennent des requêtes écrites en SQL et dépendent des paramètres passés dans la requête d'appel Ajax. Le menu comporte actuellement quatre onglets. Un onglet « RealTime » permettant l'affichage des trajectoires en cours de réalisation. Un sous-menu « Replay » permettant d'afficher et de rejouer les tentatives réalisées

par les robots. Un onglet « scenario » visant à autoriser la visualisation des scénarios des missions de manière distincte des tentatives et ainsi de préparer les futurs essais des robots. Et enfin, un onglet « API » afin de promouvoir l'API et son éventuel emploi auprès des différents participants de la WRSC amenés à utiliser ce client de suivi des robots voiliers. Enfin, les données envoyées via des SMS sont récupérées et traitées grâce à un script PHP spécifique, nommé « tel.php », qui entre les données directement dans la base.

Deux particularités dans le développement méritent d'être mentionnées. D'une part, l'usage d'une fonctionnalité très intéressante du langage JavaScript nommée « prototype » qui permet de modifier dynamiquement les caractéristiques d'une classe d'objets en lui ajoutant des propriétés ou des méthodes, on peut parler de « surcharge de classe ». Ainsi, dans mon cas, cela m'a permis d'étendre la classe « Marker » fournit par l'API Google Maps en lui ajoutant une date d'obtention, une donnée de vitesse caractérisant celle instantanée du robot en ce point et une propriété « azimuth » afin de connaitre la direction du robot en chaque point ainsi que les méthodes associées à ces nouveaux attributs. D'autre part, afin d'être en mesure d'afficher en temps réel, la direction et la force du vent, une autre requête Ajax a été implémentée. Cette dernière appelle une fonction PHP qui vient lire les trames METAR de l'aéroport le plus proche ([18]) et après traitement, lui retourne les données demandées qu'il ne reste plus qu'à afficher. On dispose ainsi d'informations relativement exactes sur le vent qui présentent l'avantage d'être mises à jour toutes les demi-heures. La trame METAR diffusée sur Internet via plusieurs sites donne accès à bien d'autres informations qui ne présentent pour l'instant pas d'intérêt et ne sont donc pas exploitées.

1.2. Construction de la Base de données

Afin d'être en mesure de remplir sa mission de tracking, le système doit être en mesure de stocker diverses données sur les robots voiliers à savoir principalement leur position au cours du temps mais également des données techniques sur le robot, ses caractéristiques, l'équipe qui le met en œuvre, … Pour ce faire et étant donné la taille des fichiers à conserver, il a été choisi de développer une base de données au format MySQL ([19]). Le caractère open source et populaire de ce système de gestion des bases de données demeure la garantie d'un développement sûr et facilité par une importante documentation. D'autres solutions auraient pu être retenues telles des systèmes indépendants du langage SQL comme MongoDB mais ma connaissance basique du langage SQL et la faible complexité de la base initialement mise en œuvre préconisait l'emploi d'une telle solution relativement aisée à mettre en place localement pour les tests et l'implémentation puis en ligne sur le serveur alloué à la WRSC. Dans les faits, la base de données de la première version du client a été créée et gérée à l'aide de l'outil « phpMyAdmin » ([20]), qui facilite grandement les opérations de création et de management de la base au travers principalement d'une interface graphique disponible depuis n'importe quel navigateur. La base de données a été structurée comme suit (cf. figure n°7). On peut ainsi distinguer quatre tables en relation les unes avec les autres. La table «Robots » regroupe les différents appareils de tracking et leurs caractéristiques comme le nom de l'équipe possédant le robot, l'icone à employer pour afficher le robot sur le client, la couleur de la trajectoire, l'adresse du site web de l'équipe, le numéro de téléphone du dispositif de tracking android et enfin le statut du dit dispositif soit son état activé ou non. Un robot se déplaçant possède des coordonnées enregistrées dans la table « coords ». Un « coord » se compose d'un identifiant unique, d'une latitude, d'une longitude, d'une date et heure d'acquisition du point, d'un cap , d'une vitesse instantanée calculée lors de la prise du point GPS, du nom du robot ayant réalisé cette

mesure, du nom de la mission qu'il était alors occupé à réaliser et de quelques paramètres dédiés à l'extrapolation de trajectoire. Un robot s'il se meut, réalise obligatoirement une mission qui est

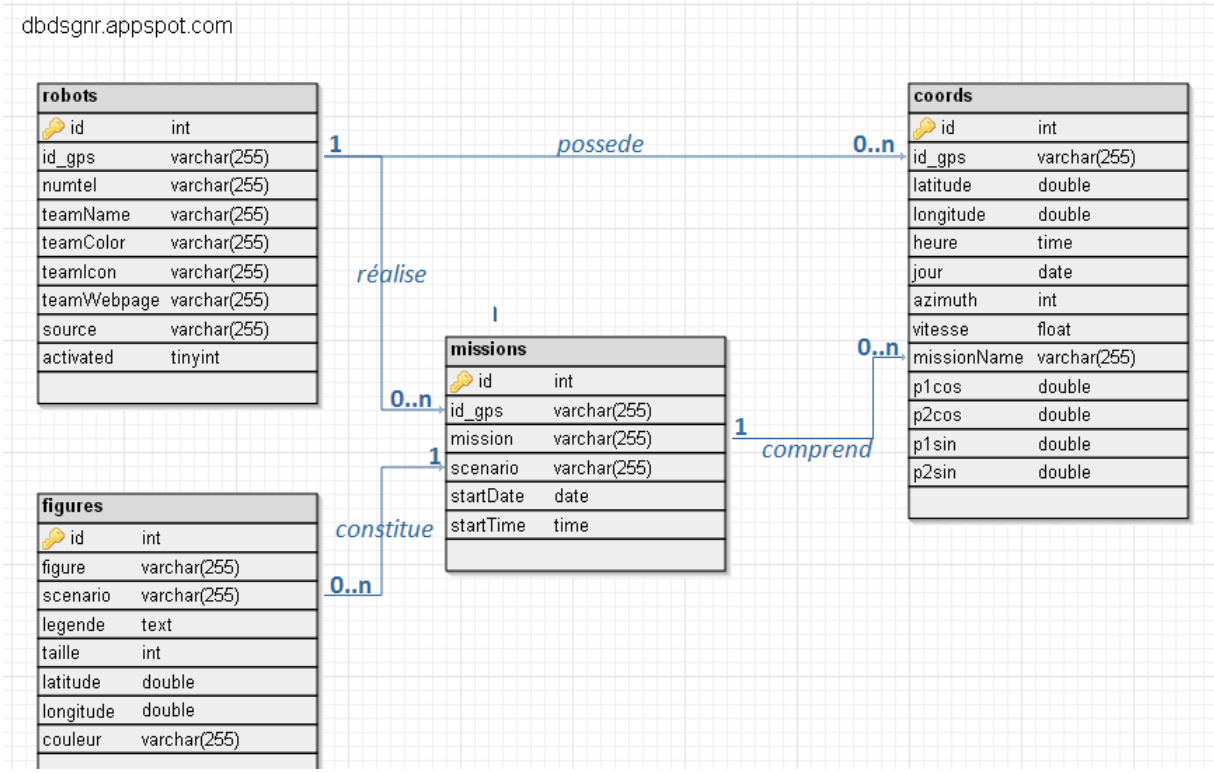


Figure 7 : Architecture de la base de données

caractérisée par un identifiant unique, un nom, une date et une heure de départ, et qui est éventuellement liée à un scénario. Une mission regroupe et se compose de coordonnées géographiques. La dernière table n'est autre que le contenu d'un scenario à savoir l'ensemble des figures qu'il contient. Dans le contexte de la WRSC, 5 types de figures ont été crées : les balises, les murs, les portes, les zones et les trajectoires. Chacune de ces figures possèdent sa propre représentation au niveau du client. Cette dernière se veut caractéristique de l'élément concerné et dans une certaine mesure respectueuse du code de navigation, ainsi les portes sont définies à l'aide de deux balises et d'un trait les reliant, la couleur des balises indiquant le sens de passage de la porte (cf. figure n°8). Afin de les enregistrer le schéma suivant a été retenu, dans la table « figures » on enregistre un à un les points les définissant à l'aide des champs latitude, longitude, couleur et légende. Le champ figure comprend le type de la figure, celui intitulé scenario le nom du scenario et la taille permet de connaitre le nombre de points composant la figure explosée dans la base afin d'être en mesure de la recréer même si un scénario comporte plusieurs figures similaires. Parallèlement, un petit utilitaire écrit en JavaScript et en html a été crée afin de pouvoir facilement rédiger des scenarios et les enregistrer dans la base de données.

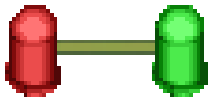


Figure 8 : Visuel d'une porte

1.3. Css et charte graphique

Une fois l'architecture du site construite, les interactions activées et les fonctionnalités implémentées, l'IHM et son aspect visuel ont du être développés afin de ne pas perdre en qualité et en utilisabilité par une présentation peu claire, peu attractive voire rebutante car à la fois trop basique et ne répondant pas aux standards esthétiques du web ou de ce qu'un utilisateur est en mesure d'attendre visuellement de n'importe quelle page internet. Il aurait en effet été dommage

que dépourvu d'une ergonomie correcte et d'un aspect graphique élégant, le site ait perdu en intérêt et en utilisation, sans même évoquer la mauvaise publicité réalisée pour l'école qui demeure la principale organisatrice de l'événement dans lequel ce projet se doit d'être dans un premier temps employé. Depuis le début des années 2000, une pratique en matière de mise en page des sites web s'est très largement répandue à savoir l'usage des feuilles de style en cascade ou Cascading Style Sheets (abrégées en CSS) ([21]). Cette méthodologie basée sur un langage particulier permet de dépeindre le rendu visuel d'un site indépendamment de la structure du document écrite pour sa part en HTML la plupart du temps. La présentation est ainsi réalisée en parallèle de la page et il devient plus aisé de définir une charte graphique et donc une cohérence visuelle à l'ensemble de son site. Etant donné que ce projet vise à offrir une fonctionnalité de tracking pour les robots voiliers de la WRSC 2013, la charte graphique établie se base sur celle du site web de l'événement à savoir des tons bleutés afin de ne pas jurer et montrer l'inscription du service de tracking dans la réalisation de l'événement. Globalement, le fichier css rédigé s'attache donc à garder une couleur de fond similaire et le liseré bleu clair qui entoure les différents cadres présents sur le site officiel. Les couleurs et l'aspect visuel des autres éléments ont été choisis en fonction de ces deux règles initiales afin de maintenir l'harmonie de l'ensemble et son ergonomie. Le résultat obtenu visible sur la figure n°9 est globalement un bon compromis entre cohérence graphique avec le site officiel, ergonomie et facilités d'emploi, homogénéité, esthétique visuelle et fonctionnalités du client d'affichage des trajectoires des robots voiliers. La mise au point d'un tel fichier fut un travail relativement long voire pénible constitué d'une multitude de retouches et de corrections minimes afin d'aboutir à quelque chose d'élégant et en accord avec le cachet visuel que je souhaitais donner au site web.

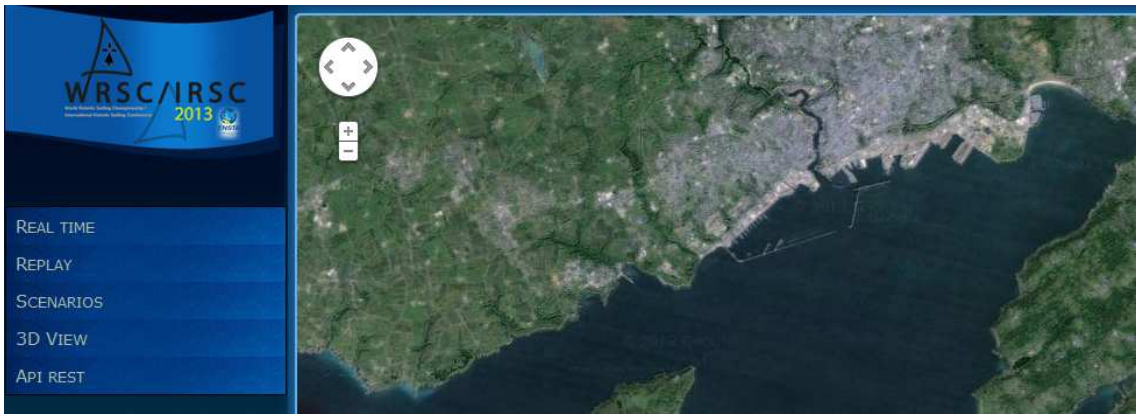


Figure 9 : Aperçu du client Web

1.4. Gestion et traitement des SMS

Initialement quatre solutions ont été envisagées pour permettre la récupération des SMS, leur traitement étant systématiquement, et ce quelque soit l'outil employé, assuré via un script écrit le plus souvent en PHP qui, une fois appelé par le système, a à charge d'extraire les données du SMS et d'en réaliser l'enregistrement dans la base de données adéquate. On distingue les quatre approches suivantes : la mise au point d'un SMS Center ([22]), l'acquisition d'un numéro dédié auprès d'un opérateur téléphonique qui mettrait une partie des fonctionnalités de son SMS Center à la disposition de ce projet, la location d'un numéro court auprès d'un entreprise de promotion par SMS qui offrirait des possibilités similaires au numéro dédié et la transformation d'un téléphone Android en SMS Center léger grâce à une application spécifique. Les deux premières propositions se sont avérées irréalisables car couteuse et très technique pour la première et simplement onéreuse

pour la seconde. Les deux solutions restantes ont été étudiées. Pour le moment, seule la dernière a été testée bien que des démarches aient été entreprises afin d'essayer l'autre solution et une société ciblée. Cependant, faute de temps et disposant d'une solution fonctionnelle, la dernière approche a pour l'instant été privilégiée.

Dans la première version du système, et de manière similaire dans les suivantes, la gestion et le traitement des SMS envoyés par la balise GPS sont assurés par un script PHP. Appelé au travers d'une application Android nommée SMSSync ([23]) qui permet de transformer le téléphone en un SMS Center léger et d'émettre des requêtes de traitement via l'appel à des scripts, il vient interpréter le contenu du SMS, en extrait les données de géolocalisation et accède directement à la base de données afin de les stocker. Suivant le contenu du SMS reçu et les mots-clés employés, trois comportements sont envisageables soit le lancement d'une mission c'est-à-dire d'une tentative par un robot de réaliser une certaine trajectoire, soit l'enregistrement de données de localisation correspondant au chemin réalisé par un robot, soit enfin, la notification de l'achèvement d'une mission par un robot. A l'usage, ce script s'est avéré parfaitement effectif même si le fait d'interagir directement avec la base via des requêtes SQL puisse se révéler une porte à d'éventuelles injections de code malicieux ou corrompu.

2. Le système embarqué

Une fois, la partie hardware définie et le choix d'utiliser des smartphones confirmé, il a fallu développer la partie software qui allait réaliser l'ensemble du travail d'acquisition et de transmission de la position GPS du robot, OS oblige.

Les spécifications sont les suivantes, le programme doit être en mesure de récupérer la position du téléphone via la puce GPS du téléphone et communiquer ces données pour le moment par SMS. Par ailleurs, l'utilisateur doit pouvoir paramétrer son système et en modifier certaines caractéristiques telles l'id du téléphone ou le numéro d'envoi des données pour l'adapter à son propre contexte d'utilisation. Le programme ou plutôt l'application a été implémentée sous Android, un environnement mélangeant du code Java et des fichiers xml. Elle est développée pour la version 2.3 d'Android répondant au doux nom de « GingerBread ». Elle est compatible avec les versions ultérieures mais également avec la version 2.2 bien que dans ce cas, son fonctionnement ne soit pas parfaitement identique du fait de changements d'implémentation de certaines fonctionnalités dans les versions suivantes de l'environnement. Cependant, l'application demeure globalement fonctionnelle.

Le code de l'application de tracking s'organise comme suit : un manifeste ou fichier xml destiné à paramétrer l'application et définir ses droits vis-à-vis du système, un ensemble de ressources comprenant les fenêtres de l'application, l'interface utilisateur et diverses images et autres fichiers de personnalisation du contenu affiché et enfin deux packages de classes java, l'un étant une librairie de fonctions mathématiques permettant de résoudre des calculs d'algèbre linéaire tandis que l'autre constitue le cœur de l'application. Ce package se compose de trois classes au fonctionnement décrit ci-après. La librairie mathématique sera, quant à elle, présentée ultérieurement dans ce rapport et ne vient qu'ajouter une fonctionnalité supplémentaire à l'application. En aucune façon, elle ne lui est indispensable (cf. partie extrapolation de trajectoire pour en connaitre l'utilité et l'emploi). Concernant le package central, on distingue donc trois classes respectivement nommées « MainActivity », « ServiceLocalisation » et « UserSettings ». On peut

préciser qu'historiquement l'application ne comportait qu'une seule classe démarrant une activité qui réalisait l'ensemble des tâches et qu'au fur et à mesure du développement et après les premiers essais, les deux autres ont été ajoutées pour en étendre les capacités. L'architecture finale du système est celle mentionnée dans le diagramme de classes simplifié ci-contre.

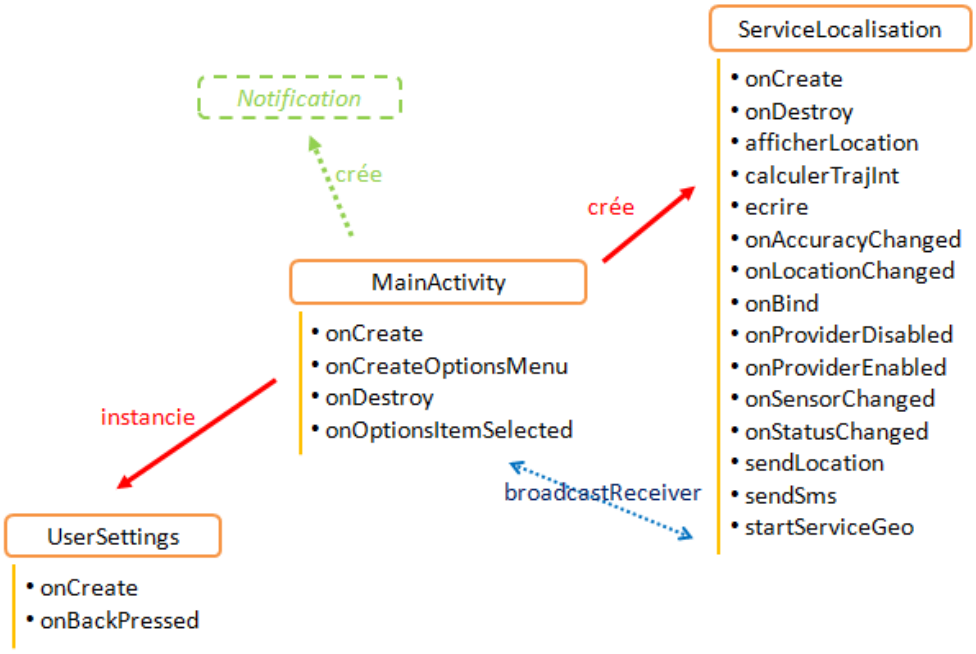


Figure 10 : Diagramme de classes de l'application embarquée

La classe « MainActivity » implémente l'IHM du programme et permet de lancer les différentes fonctionnalités du programme. L'application comporte une unique fenêtre (cf. figure n°11) composée de plusieurs champs permettant à l'utilisateur de connaitre, une fois le service lancé, sa position GPS, son azimut et sa vitesse ainsi que deux boutons permettant d'initier le service ou de l'arrêter. C'est également cette classe qui crée la notification de fonctionnement du service qui apparait dans la barre de notifications située en haut des téléphones et le menu des préférences utilisateur. Ce menu accessible par l'appui sur le bouton du même nom des téléphones permet de paramétrer l'application en précisant le nom ou id du téléphone, le numéro auquel doivent être envoyées les données GPS émises sous forme de SMS avec une trame bien précise, l'activation ou non de cette fonctionnalité d'émission des SMS et dans un futur plus ou moins proche, l'émission de requêtes HTTP en direction de l'API qui a été dans un second temps développée. Ce menu est géré par la classe « UserSettings » et un layout xml spécifique permettant sous Android de définir de tels objets. Ses données sont accessibles au travers des préférences partagées du programme qui sont une entité particulière du langage Android permettant grâce à un « preferenceManager » de connaitre dans l'ensemble du programme les valeurs de ces champs.

Si la classe « MainActivity » est comme son nom l'indique une activité au sens d'Android, la dernière classe, et de loin la plus intéressante, est un service. C'est elle qui accomplit l'ensemble des taches qui échoient à notre programme de tracking à savoir la récupération des données GPS, leur enregistrement sur une carte SD et leur éventuel envoi via un SMS. L'emploi d'une telle classe et d'une telle fonctionnalité se justifient par la nécessité pour le programme de continuer à fonctionner quand l'écran se verrouille et se met en veille, et pour l'utilisateur de pouvoir éteindre l'écran lorsqu'il ne consulte pas les données de position afin d'économiser la batterie du téléphone.

Figure 11 : Ecran d'accueil de l'application

Dans la partie qui suit, je vais revenir plus en détail sur les fonctions et les mécanismes qui permettent à cette classe de remplir ses objectifs. L'obtention des données GPS est réalisée au travers d'un LocationManager afin d'autoriser et de gérer l'accès au capteur et d'un locationListener qui nécessite et génère une série de méthodes afin de pouvoir utiliser au travers de l'OS les capteurs mis en jeu pour l'acquisition du signal GPS et la traduction de sa trame. Le compas est quant à lui récupéré au travers d'un sensorManager, un pendant du LocationManager qui couvre cependant davantage de types de capteurs embarqués. Les données GPS sont également enregistrées sur une carte SD de façon systématique tandis que leur envoi est optionnel.

La première fonction appelée est la fonction « onCreate » qui est systématiquement appelée à la création d'une activité ou d'un service (cf. cycle de vie d'un service sous Android en annexes). Dans cette dernière, on vient d'abord instancier les différents utilitaires (locationManager, sensorManager, …) nous permettant de récupérer les données des capteurs puis on crée le fichier de logs au niveau de la carte SD. Le fichier est au format csv afin d'être facilement exploitable dans n'importe quel logiciel de tableurs et aisément exportable, les convertisseurs depuis le format csv vers d'autres formats étant nombreux sur internet. Le preferenceManager est ensuite initialisé afin de connaitre les paramètres utilisateurs qui sont accessibles dans le menu Options. Enfin, on appelle la fonction startServiceGeo qui lance véritablement l'exécution du service.

Avant de revenir sur le comportement de cette fonction, la fonction onDestroy va être brièvement évoquée. Comme son nom l'indique, elle est automatiquement appelée à la destruction du service et à ce titre, va nous permettre de le quitter correctement et proprement. Pendant de la fonction onCreate, elle va désinstancier les différents utilitaires utilisés pour les capteurs et proprement libérer les ressources système, capteurs et mémoire employées lors de l'exécution de cette application.

La fonction startServiceGeo se résume à un genre particulier de timer qui vient tous les intervalles de temps fixé par l'utilisateur relever la position GPS. Si elle détecte un changement de position, elle fait alors appel à la fonction de traitement onLocationChanged. Cette dernière va tout d'abord chercher à rafraichir les données affichées dans l'activité principale en émettant à son intention un intent contenant les données de localisation. Concrètement, il s'agit d'un genre de hashtables qui associe des couples de clés-valeurs et permet leur transmission au sein du système entre des instances non directement liées ou connectées. Dans un second temps, cette fonction appelle la fonction « ecrire » qui vient inscrire, dans le fichier csv de la carte SD, les données nouvellement acquises selon la trame définie (id, date-heure, latitude, longitude, vitesse, azimut, p1cos, p1sin, p2cos, p2sin) puis si nécessaire la fonction sendLocation qui, au travers d'un smsManager, autorise la rédaction et l'envoi d'un SMS au numéro enregistré dans les paramètres utilisateurs. Le contenu classique d'un SMS est le suivant, vient d'abord l'identifiant du téléphone ou « idGps », puis la date et l'heure d'acquisition du point, la latitude, la longitude, la vitesse, le cap, 4 paramètres utiles à l'extrapolation de trajectoire notés p1cos, p1sin, p2cos, p2sin (cf. II.3.) et un label missionName qui permet de connaitre le nom de la mission pour le système d'acquisition. Ce dernier paramètre est devenu caduque et inutile une fois l'API créée. Cependant à des fins de test et de

validation et étant donné que la limitation de 160 caractères n'était pas atteinte, il a été temporairement conservé. La récupération du cap étant réalisée à l'aide du sensorManager avec une fréquence plus élevée (1 Hz contre 0.22 Hz par défaut pour le GPS), ses valeurs sont localement stockées dans un tableau et ponctuellement récupérées lors de l'envoi. L'implémentation d'un tel mécanisme ainsi que la fonction « calculerTrajInt » s'explique par la mise au point d'une méthode d'extrapolation de la trajectoire, à des fins majoritairement de gain en précision, justifiée et expliquée dans la partie s'intitulant « Extrapolation de trajectoires » (cf. II.3.).

Enfin au niveau de la classe « MainActivity », un « BroadcastReceiver » a été instancié afin de faire dialoguer les deux classes et ainsi de pouvoir récupérer les données GPS acquises par l'une puis transmises via un « intent » au système et enfin afficher dans les champs adéquats de l'autre.

Au delà d'une architecture de programme relativement simple dans sa conception et son implémentation, cette application se veut à la fois ergonomique et hautement personnalisable au travers d'un vaste choix d'options. Son développement, ses conditions d'emploi et la volonté de pouvoir mettre en veille l'écran afin de réduire son cout énergétique en fonctionnement ont induit des choix techniques qui ont conduit à la mise au point d'un service. Ce dernier, s'il remplit pleinement sa mission, n'en demeure pas moins un objet particulier du langage et de la programmation Android et à ce titre a fortement influencé l'allure générale de l'application finale. Malgré tout, l'ajout de fonctionnalités reste envisageable au travers de nouvelles fonctions ou de nouvelles classes.

3. Extrapolation de trajectoires

3.1. Un compromis SMS – batterie – précision

Les premiers tests du système embarqué ont montré qu'il fallait en moyenne 10 secondes au système Android pour acquérir un point GPS une fois celui-ci initialisé et qu'en dessous de 25 secondes entre chaque mesure les coordonnées obtenues perdaient nettement en précision. En outre, il est inconcevable d'envoyer toutes les secondes un SMS avec des données pour de simples raisons énergétiques. Il s'agissait donc de trouver un compromis entre précision, nombre de SMS envoyés à la minute et consommation de la batterie. Enfin, un SMS classique se limite à 160 caractères ce qui implique de trouver des solutions pour garder le message court tout en augmentant sa teneur et le nombre d'informations transmises. L'idée mise en place repose sur le calcul par intervalles et l'utilisation du compas intégré aux smartphones modernes. Grâce à ces deux outils, une technique d'extrapolation de la trajectoire a pu être mise au point. Son emploi entraine un gain non négligeable en précision comme le montrent les résultats présentés ci-dessous et vient globalement pallier la fréquence relativement faible des mesures GPS.

3.2. Méthodologie et protocole

L'idée est relativement simple, on enregistre via le téléphone le cap suivi entre deux points GPS avec une fréquence plus élevée et connaissant la vitesse instantanée en ces deux points, on vient approximer via un calcul par intervalles les positions occupées par le système entre les deux instants de relevé GPS de sa position permettant de prendre en compte les erreurs de mesure mais également d'en corriger une partie lors d'une boucle de calcul inverse (Partie « backward » du calcul par intervalles). Dans les faits, l'implémentation est un peu plus délicate. De même qu'on ne peut émettre la position GPS toutes les minutes pour des raisons d'abord énergétiques, envoyer dans un

même message, une position GPS prise toutes les 30s et une mesure de cap prise toutes les secondes avec une précision de 2 flottants, implique très vite le dépassement de la limite fixée de 160 caractères. Envoyer plusieurs messages n'est pas un palliatif plus acceptable. Enfin, il s'agit de définir qui du téléphone ou du client doit effectuer le travail d'extrapolation de la trajectoire. La démarche adoptée est la suivante : au téléphone reviennent les tâches d'enregistrer les données capteurs et d'en condenser l'information de manière à pouvoir être émise dans un unique message et au client, il appartient de réaliser les calculs d'extrapolation à partir des données reçues.

3.3. Code et implémentation

La partie qui suit s'attache à expliquer concrètement la solution mise en œuvre tant au niveau du téléphone qu'au niveau du client d'affichage. Dans un premier temps, on s'attachera à présenter les calculs théoriques réalisés puis on reviendra sur l'implémentation pratique de cette solution et sur les outils nécessaires à sa mise en œuvre. Certains calculs moins importants sont renvoyés en annexe.

o Protocole et calculs

Afin de correctement présenter l'intégration de ces calculs dans notre système, les calculs ont été décomposés en grandes étapes présentées et explicitées ci-après dans leur ordre de traitement.

1. Acquisition du cap sur un intervalle de temps

Entre deux relevés de la position GPS, le cap est régulièrement mesuré et enregistré. On obtient une courbe du cap en fonction du temps ou au choix du nombre de mesures.

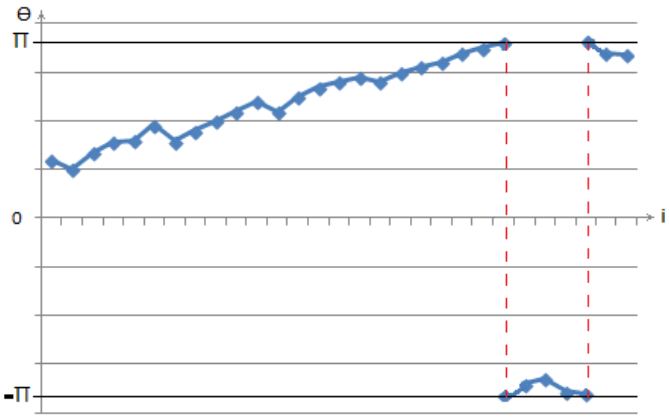


Figure 12 : Cap au cours du temps

2. Passage au sinus et au cosinus

On utilise ensuite une astuce de calcul courante en robotique, on passe au sinus et au cosinus de l'angle mesuré précédemment pour des soucis de continuité.

3. Approximation polynomiale des fonctions sin et cos à l'ordre 3

On réalise une approximation polynomiale à l'ordre 3 du sinus et du cosinus d'où :

$$\sin\theta_i = \alpha_3 t_i^3 + \alpha_2 t_i^2 + \alpha_1 t_i + \alpha_0 \; avec \; \alpha_j, les \; inconnues$$

Donc, rapporté sur l'intervalle de temps séparant deux prises GPS, on obtient la formule algébrique suivante à la k-ième itération de la formule :

$$T_k A_k = M_{sin,k} \ (de\ même\ pour\ M_{cos,k})$$

$$avec\ A_k = (\alpha_k), k \in [\![0,3]\!]\ et\ M_{sin,k}, (\sin\theta_i)_{i \in [\![0,n-1]\!]}\ et\ n, le\ nombre\ de\ mesures$$

4. Approche indicielle et normalisation

Avant de poursuivre les calculs et de chercher à déterminer les inconnues, on privilégie une approche indicielle qui ne change rien à une approche temporelle les mesures étant dans tous les cas prises régulièrement. Puis, on normalise entre 0 et 1 afin de s'affranchir du nombre variable de mesures réalisées entre deux prises de position GPS dont la fréquence a tendance à légèrement fluctuer et par conséquent de ne pas avoir à envoyer cette donnée supplémentaire. On a alors :

$$t_i = \frac{i}{n-1} \text{ où } n, le\ nombre\ de\ mesures\ effectuées$$

5. Conditions aux limites

Le cap étant parfaitement mesuré et transmis à chaque prise de la position GPS, on dispose de deux conditions aux limites correspondant aux mesures de l'azimut à l'instant actuel notée C_0 et lors de la précédente mesure notée C_1. A l'aide de ces conditions, on réduit l'équation à 4 inconnues à une équation à 2 inconnues (cf. annexes). On a désormais :

$$T_k A_k = M_{sin,k}\ avec\ A_k = (\alpha_k), k \in [\![1,2]\!]$$

$$et \begin{cases} \alpha_0 = sinC_0 \\ \alpha_3 = sinC_1 - sinC_0 - \alpha_1 - \alpha_2 \end{cases}$$

Les calculs sont similaires pour le cosinus.

6. Résolution de l'équation

L'équation est résolue algébriquement ou directement du fait du faible nombre d'inconnues.

$$A_k = \left(T_k^T T_k\right)^{-1} T_k^T\ b \quad (cf.\ annexes)$$

7. Changement de repère « LatLng vers XY »

On dispose actuellement d'une latitude et d'une longitude. Afin d'être en mesure d'effectuer correctement les calculs d'extrapolation, on se place dans un repère cartésien Est – Nord – Centre de la terre centré sur le robot ou en un point proche. On se place donc dans un repère local tangent à la surface de la terre. Les hypothèses de la transformation sont les suivantes, la terre est considérée localement sphérique (rayon localement constant) et on se situe au niveau de la mer (altitude nulle). Dans ce repère plan, les points sont représentés par une abscisse et une ordonnée. L'expression de la matrice de rotation représentant ce changement de repère permet de définir les relations de passage d'un système à l'autre. La formule employée est extraite du cours de robotique ([24]), elle demeure correcte pour des points située à moins d'une centaine de km du centre du repère.

$$T : \begin{cases} \tilde{x} = R_t \frac{\pi}{180} \cos(\frac{\pi}{180} l_y)(l_x - l_{x0}) \\ \tilde{y} = R_t \frac{\pi}{180} (l_y - l_{y0}) \end{cases}$$

avec l_{y0}, la latitude du centre du repère, l_{x0}, sa longitude et (l_x, l_y), le point mesuré.

Dans ce repère R_{ENC}, on est en mesure d'effectuer des calculs classiques de géométrie plane.

8. Calcul des vitesses intermédiaires V_i

On ne connait la vitesse du robot qu'au niveau des extrémités c'est-à-dire lors des mesures GPS. On calcule donc les vitesses intermédiaires par approximation affine. On a alors :

$$v = (V_1 - V_0)t + V_0 \; pour \; t \in [0,1]$$

9. Normalisation sin et cos

Les approximations successives et les imprécisions nous obligent pour plus d'exactitude à normaliser les sinus et cosinus.

$$\widehat{sin_k} = \frac{sin_k}{\sqrt{cos_k^2 + sin_k^2}}$$

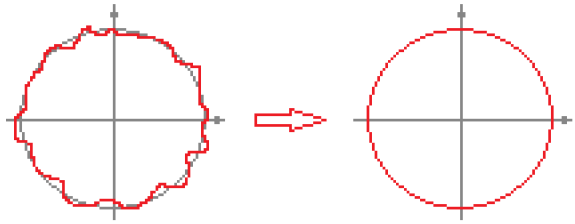


Figure 13 : Normalisation des cos et sin

10. Schéma d'Euler et calcul par intervalles

L'objectif est d'extrapoler à partir des caps suivis et de quelques autres données, la trajectoire suivie entre deux points GPS. Pour ce faire, on applique la formule d'Euler à notre problème de la façon suivante :

$$\begin{cases} [x_{k+1}] = [x_k] + [v_k][sin_k] \\ [y_{k+1}] = [y_k] + [v_k][cos_k] \end{cases}$$

Le passage par des intervalles permet de prendre en considération les imprécisions d'origine diverse présentes dans ce calcul. Les points GPS sont considérés comme ayant une précision de 10 mètres et les vitesses une précision de 0.1 m/s. On rappelle par ailleurs que l'angle considéré est un cap d'où la forme inusuelle de la formule. Il est mesuré par rapport à l'axe des ordonnées représentant le Nord.

11. Extrapolation de la trajectoire

A partir du schéma d'Euler, on est désormais en mesure d'extrapoler les points intermédiaires. Le calcul par intervalles renvoie, une fois les boucles forward et backward effectuées, des intervalles dont les centres correspondent aux points intermédiaires recherchés.

12. Changement de repère «XY vers LatLng »

On repasse dans le repère géographique afin de pouvoir placer les nouveaux points sur une carte à l'aide de la transformation inverse.

o Validation à l'aide d'un programme QT

Une première version simplifiée de ces calculs a été insérée dans un programme QT ([25]) afin d'en valider l'exactitude et le fonctionnement. Reposant sur IBEX ([26]), une bibliothèque C++ de calculs par intervalles, il a permis de s'assurer de l'intérêt d'une telle approche (cf. figure présentant les résultats). Sur ce graphique, on peut observer les deux positions GPS mesurées, en rouge les résultats de la boucle forward et en bleu, le chemin inverse qui affine les intervalles précédemment obtenus. Au final, la trajectoire courbe effectivement réalisée par le robot devient visible.

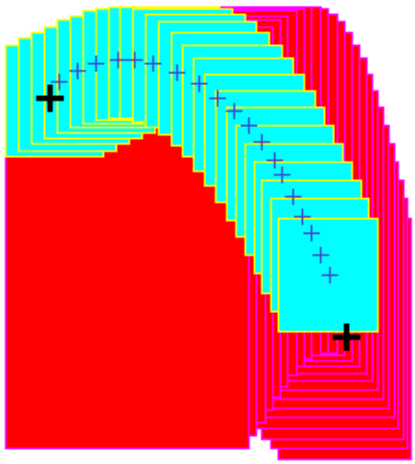


Figure 14 : Démonstrateur QT

o Au niveau du téléphone

Les étapes 1 à 6 sont réalisées par le téléphone. Le cap est acquis via le compas du téléphone toutes les secondes. La bibliothèque Jama qui est initialement une bibliothèque Java permettant la réalisation de calculs algébriques a été assez facilement adaptée à Android. L'emploi d'une telle bibliothèque de fonctions, d'une telle machinerie peut sembler exagéré puisque les conditions aux limites permettent de ramener ce problème à une équation à deux inconnues. Cependant son usage offrait deux avantages au prix d'un léger gain en poids mémoire de l'application. D'une part, elle permettait de directement implémenter les calculs présentés ci-dessus sans chercher à les adapter ou à les optimiser et ainsi, laisser des méthodes éprouvées les résoudre d'où moins d'erreurs de bord non nécessairement détectées lors de tests. D'autre part, elle permettait de conserver un caractère générique aux calculs et donc se révélait facilement modifiable si, par exemple, on souhaitait pour des raisons de précision augmenter l'ordre des polynômes approximant le sinus et le cosinus de l'angle comme cela aurait pu se révéler utile par la suite. Dans les faits, la fonction « calculerTrajInt » se charge de calculer les deux inconnues pour le sinus et le cosinus en suivant les étapes précédemment présentées. Il en résulte 4 paramètres qui sont ajoutés à la trame du SMS et transmis au serveur.

o Au niveau du client

Les étapes restantes sont traitées par le client. Il lui revient donc d'exploiter les paramètres calculés par le téléphone afin d'extrapoler concrètement la trajectoire et en afficher le résultat soit les différents points intermédiaires entre deux points mesurés à l'aide du capteur GPS. Le client doit donc effectuer un changement de repère, extrapoler les points de la trajectoire suivie puis repasser dans le repère géographique afin d'afficher les points nouvellement calculés sur la carte. Afin de réaliser au niveau du client ces calculs par intervalles, une bibliothèque dédiée présentée ci-après a été implémentée.

3.4. Bibliothèque JS

L'extrapolation par intervalles des trajectoires étant réalisée au niveau du client, il fallait pouvoir disposer d'une bibliothèque de fonctions en JavaScript permettant de tels calculs. Faute d'en avoir trouvée une prête à l'emploi, j'en du en créer une. Pour des raisons de temps de développement, je me suis limité aux fonctions dont j'avais besoin tout en m'assurant qu'il serait possible d'ajouter facilement de nouvelles fonctionnalités à cette bibliothèque. Une classe Interval a

donc été créée possédant les attributs bornes inferieure et supérieure, et des méthodes basiques telles les fonctions inter ou mid. Le langage Javascript n'autorisant pas la surcharge des opérateurs, ces derniers ont été transformés en fonctions rendant l'écriture des calculs moins fluide. Ainsi, la somme des intervalles I1 et I2 s'écrit : I3 = I1.add(I2) ou l'inverse. La fonction « add » renvoyant un nouvel intervalle, on est assuré par définition du constructeur que ses bornes seront correctement ordonnées. L'absence d'objet Infini spécifique dans la bibliothèque Maths de JavaScript m'a poussé à considérer la valeur 999 comme la plus grande valeur envisageable bien que dans le cas présent, cela n'est pour ainsi dire aucune importance. Les opérations et les fonctions implémentées sont résumées dans le tableau ci-joint.

Nom	Fonction
Interval	**Constructeur**
inter	Calcule l'intersection entre deux intervalles
c_mul	Permet de multiplier un intervalle par une constante
i_mul	Permet de multiplier deux intervalles
i_add	Réalise la somme de deux intervalles
i_minus	Réalise la différence de deux intervalles
i_mid	Renvoie la valeur du milieu d'un intervalle
i_print	Permet d'afficher un intervalle

Avant de déployer cette bibliothèque et de l'intégrer à mon client, il m'a fallu la tester en profondeur. Pour ce faire, une page HTML de test a été développée. Il s'agit d'un formulaire avec deux champs permettant d'indiquer des intervalles et un ensemble de checkbox afin de choisir la fonction à essayer. Pour créer un ensemble vide, il suffit de mettre un v dans l'une des cases permettant d'instancier l'intervalle. Le bouton « submit » permet de lancer le test et d'afficher le résultat (cf. annexes). Les résultats ont été validés à l'aide d'une batterie de calculs réalisés à la main et confirmés avec une bibliothèque C++ de calculs par intervalles (IBEX) implémenté dans un programme sous QT. Les résultats des deux programmes ont ensuite été comparés et quelques erreurs ont ainsi pu être détectées et par conséquent corrigées. Les fonctions d'interpolation ont également été testées en comparant leurs résultats avec ceux obtenus par le programme QT. Une fois cette tache effectuée, la bibliothèque a été validée et jointe au client.

4. Tests, limites et bilan

Cette première architecture a subi un certain nombre de tests. Ils ont mis en exergue des erreurs de programmation qui ont ainsi pu être corrigées mais ils ont également permis d'adapter l'architecture aux enjeux visés et de mieux définir et cerner les fonctionnalités souhaitées.

L'ajout de la méthode d'extrapolation constitue un plus intéressant et permet un gain considérable en précision comme en atteste la figure ci-dessous. On observe à gauche le tracé sans la méthode et à droite avec, sachant que le robot se déplace dans le premier couleur de la piste. Bien que le tracé obtenu comporte encore des approximations, on peut constater que la méthode d'extrapolation comble de manière vraisemblable et assez correcte les tronçons inconnus. On peut par ailleurs mentionner que l'application de cette technique est facultative, le programme ne l'utilise que si les champs et les variables impliqués sont renseignés. Ainsi, si dans le futur, de nouvelles approches offrent de meilleurs résultats en termes de précision et de compromis énergétique, il sera

possible de ne plus la mettre en œuvre tout en conservant le client existant. Les essais effectués ont aussi été l'occasion de constater les limites de la théorie par rapport à la pratique. En effet, si sur le papier, cette méthode est en mesure de détecter et de rendre compte de la réalisation d'une boucle. Dans les faits, suite principalement à des problèmes de calcul de la vitesse effective en chaque point, il a été observé que de tels mouvements n'étaient que rarement transmis et donc observables.

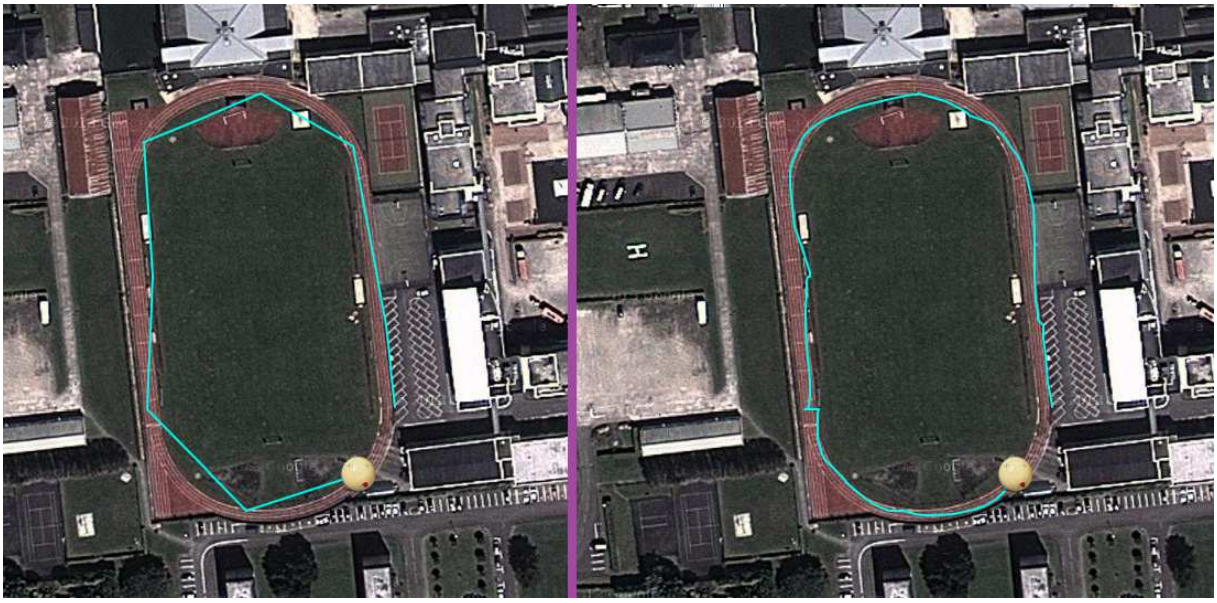


Figure 15 : Apport du procédé d'extrapolation

Les essais réalisés autour de la piste qui constitue un excellent point de référence, les premiers essais en mer et les phases d'expérimentation faites à Angers lors des journées démonstrateurs ([27]) ont permis de révéler différentes difficultés techniques. On citera par exemple la nécessité d'automatiser le lancement d'une mission par le robot concerné plutôt qu'une activation manuelle souvent délicate et source d'incohérence ou encore le besoin de pouvoir récupérer plus facilement certaines données plus souvent requises.

Les tests à Angers ont cependant permis de constater la relative précision du système de balises Android malgré une fréquence plus faible d'obtention des points qu'un module GPS classique. Pour aboutir à une telle conclusion, les traces GPS des capteurs du robot voilier VAIMOS ([28]) ont été comparées à celles de la balise Android embarquée à son bord. Globalement, bien que le module GPS du smartphone demeure de moins bonne qualité que celui du voilier et donc moins précis et rigoureux de ses mesures, les traces correspondent assez bien du moins dans la partie où le système a fonctionné. Il s'est en effet inexplicablement arrêté en plein milieu d'une série de mesures. Cette erreur jamais reproduite incombe probablement à un manque d'espace mémoire au niveau du téléphone qui a déclenché un comportement inattendu.

En bref, cette première phase d'expérimentation a révélé des résultats globalement prometteurs cependant elle a aussi permis de faire un premier bilan du projet. En l'état et étant donné l'architecture choisie et implémentée, la mise en place de futures options et autres services parait fortement délicate sans remettre entièrement en cause la structure même du système, un aspect pour le moins problématique et rédhibitoire à l'ajout de toute nouvelle fonctionnalité, d'où la recherche de nouvelles solutions et la mise au point d'une nouvelle architecture pour le système.

III. Développement d'une API

1. Concept et motivations

Malgré le caractère globalement fonctionnel du client mis en place, je n'ai pu que constater assez rapidement qu'il ne répondait que partiellement à la problématique de ce PFE. Il offrait certes la possibilité de visualiser en temps réel ou différé la trajectoire réalisée par de multiples robots cependant il était difficile de lui ajouter des fonctionnalités sans un important travail de fond ou même de réutiliser le système mis en place simplement et rapidement sans connaitre le code et les subtilités de sa programmation. Je n'évoque même pas le caractère inaccessible de la base de données pour toute entité autre que le client de visualisation. Ainsi, il répondait au besoin de la WRSC d'un outil de monitoring et de suivi des robots voiliers mais pas à une problématique plus large de contrôle et de visualisation de meutes de robots. Pour pallier ce défaut, l'idée fut de développer une API dédiée qui constituerait une brique logicielle facilement réutilisable pour d'autres projets ayant les mêmes problématiques de fond. Une API (Application Programming Interface) ou interface de programmation en français est un ensemble de méthodes et de classes qui offre un panel de services à d'autres logiciels. Cette bibliothèque de fonctions serait plus facilement réutilisable et adaptable que l'actuel client désormais dit « lourd ». En outre, cette approche s'inscrit davantage dans la politique actuelle de développement Web ou logiciel qui consiste à bâtir des briques ou des modules puis à les assembler selon le besoin de chaque projet ou client.

De plus, la programmation d'une telle interface autorise d'autres entités à interagir avec le système créé dans un cadre préalablement défini et contrôlé. Ainsi, une fois cette solution développée, non seulement les équipes dont les robots ont été suivis pourront facilement récupérer leurs données de localisation que ce soit en temps réel ou après la fin d'une épreuve, mais en plus le système embarqué de géolocalisation qui, dans la première approche, demeurait le seul capable d'envoyer des données à la base, peut désormais être remplacé par n'importe quel système dument identifié en mesure de dialoguer avec l'API. On gagne ainsi en utilisabilité, en visibilité et en performance. Les possibilités d'emploi croissent sensiblement, l'API devenant une porte et un moyen d'interaction avec la base de données et ses constituants (cf. figure n°16). Dans l'idéal, le client lourd n'est plus alors que soit une fonctionnalité de l'API, soit un client de cette même API profitant des services qu'elle propose. Chaque système devient alors libre de taper la base de données au travers de l'API et selon ses termes d'utilisation.

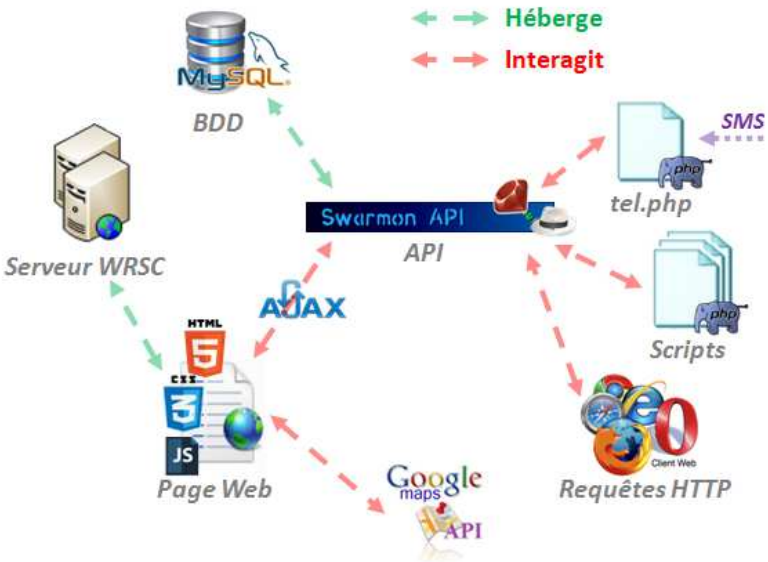


Figure 16 : Nouvelle architecture Web

Il existe de multiples méthodes pour développer une API et définir les interactions que l'on peut entretenir avec elle. Face aux contraintes de ce projet, aux propositions de mes encadrants, à la nature des données et des fonctionnalités offertes et à la politique de développement et d'utilisation

future que je souhaitais mettre en place, une API de type REST ([29]) convenait parfaitement. En effet, les données de localisation, les caractéristiques des robots ou des équipes, les scenarios à réaliser sont autant de ressources dans le contexte REST. Une telle API répond à un certain nombre de règles telles que définies par Roy Fielding en 2009 ([30]) et vient s'appuyer sur le protocole http dont elle utilise par exemple les codes de réponse pour indiquer le comportement de l'API et la disponibilité des données demandées. En outre, une telle API permet d'accéder à chaque ressource au travers d'un URL précis éventuellement paramétrable. Enfin, la documentation d'une API REST est partie intégrante de l'API elle-même, une contrainte additionnelle parfaitement en accord avec une politique d'ouverture de l'API.

2. Développement de l'API

Une API comme d'ailleurs un site web ou un serveur sont principalement des tables de routage qui donnent accès à des ressources stockées localement ou non. Ainsi, mettre au point une telle API supposait de créer un serveur web présentant une table de routes et renvoyant sur requêtes, des données.

Pour parvenir à un tel résultat efficacement et rapidement, étant donné le besoin et les futures spécifications du site web, j'ai choisi d'utiliser le framework Ruby Sinatra ([31]) qui permet de créer sans trop de difficultés des sites web de petite taille et qui est plus simple d'emploi que son grand frère « Ruby on Rails » ([32]). Ce framework permet, avec une architecture minimaliste et légère, de développer des applications web relativement complexes et complètes. L'architecture de l'application est la suivante :

- views/ Dossier permettant de stocker proprement les templates.
- public/ Dossier de stockage des ressources statiques.
- tmp/ Dossier nécessaire pour la gestion du cache par exemple.
- log/ Dossier dédié aux fichiers de logs.
- config.ru Fichier apporté permettant de paramétrer le framework.
- app.rb Fichier principal comprenant le code de notre application soit la table de routage du site.

Outre cette architecture relativement épurée et un site web robuste et performant, Sinatra offre l'avantage de pouvoir profiter de toutes les gems de Ruby, que ce soit pour les procédures d'authentification, les moteurs de templates, la gestion de la base de données, … bref un vaste panel d'options et de facilités.

Avec un tel framework, router une requête Get sur une ressource donnée est assez facile, il suffit de définir dans le code de l'application le chemin correspondant à la ressource et de mentionner le comportement souhaité pour une telle requête. On peut bien entendu lui passer des paramètres voire créer des URL génériques. Décrire le code de l'application ne présentant guère d'intérêt autre que purement technique, le lecteur est invité à se référer à la documentation de l'API pour en connaitre les fonctionnalités voire à la forge du projet pour toute visualisation directe du code. La documentation de l'API présente également quelques exemples d'emploi auxquels le lecteur est invité à se référer (http://haggis.ensta-bretagne.fr:3000/).

Cette API visant principalement à être scriptée c'est-à-dire que les requêtes seront davantage émises depuis un programme que depuis un navigateur, il est intéressant de mentionner que l'utilisateur peut choisir le type ou plutôt le format des données qu'il souhaite lui voir envoyer. Pour ce faire, il lui faut préciser dans le champ « Accept » du header de sa requête http (cf. protocole http [33]) le format de retour des données. Cinq formats sont actuellement disponibles : json, xml, csv, html par défaut et x-bni, un mime-type ([34]) personnel proche du format json, l'utilisateur pouvant ainsi choisir le format le plus adéquat à l'utilisation future des données. Des données au format csv pourront être facilement placées dans un fichier Excel tandis que celles au format json seront très aisément exploitables dans un code d'analyse ou de traitement automatique.

Cette API est accessible en ligne sur un serveur spécifique qu'il a fallu créer et paramétrer. Le serveur est déployé sur une machine Debian mise en place par l'Ensta-Bretagne. Une fois l'environnement installé et paramétré, tâches réalisées par M. Reynet dont à nouveau je le remercie, les différents composants nécessaires au fonctionnement du serveur ont pu être à leur tour mis en place. Ainsi, RVM ([35]) a été installé permettant de disposer de Ruby sur la machine et de mettre les différentes gems nécessaires au fonctionnement de mon application. Seul détail d'importance, j'ai choisi d'utiliser le serveur Ruby Thin qui, en plus de présenter des caractéristiques de performance intéressantes ([36]), fonctionne nativement avec le framework Sinatra. Pour des raisons évidentes de sécurité, la plupart des ports du serveur ont été désactivés hormis les ports 80 et 3000, utilisés respectivement pour les requêtes http et pour l'accès au serveur. Enfin, dans l'optique de pouvoir se connecter au serveur et le paramétrer, les requêtes SSH ont été autorisées sur le port 80.

3. Base de données et ORM

Bien que cette API donne accès à des ressources, il faut cependant pouvoir les stocker. Pour ce faire, une base de données similaire dans son fonctionnement et son implémentation à celle employée dans la première version a été développé (cf. figure n°17 pour le modèle relationnel et cf. annexes pour le schéma structurel classique). On retrouve ainsi les tables Robot et Coord. La table Figures a laissé la place à une table plus générique nommée Scénarios qui regroupe toutes les informations sur les différents scénarios ou missions réalisables. La table Missions a été renommée Attempts mais contient les mêmes informations sur les diverses tentatives d'un robot à réaliser tel ou tel scenario. Les tables Teams et Users ont été ajoutées. La première afin d'éviter la répétition de certaines informations au niveau de la table Robots, une équipe ou une meute donnée pouvant posséder plusieurs appareils; tandis que la seconde a été créée afin de gérer les droits d'accès et d'utilisation de l'API. Comme précédemment le moteur de gestion choisi est MySQL néanmoins grâce à la mise en place et l'emploi d'un ORM (**Object-Relational Mapping**) nommé Datamapper ([37]) plus aucune requête SQL ne sera directement effectuée.

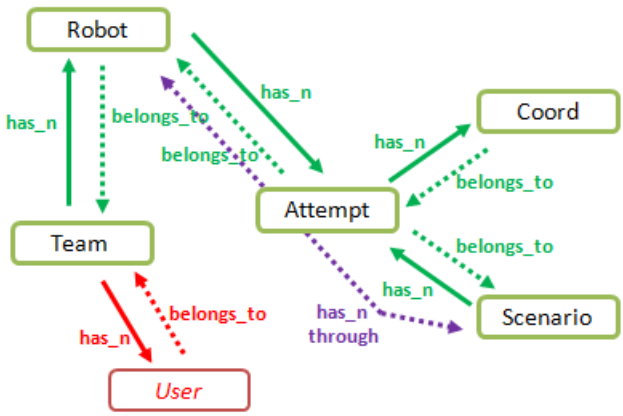


Figure 17 : Modèle relationnel de la BDD

Le fonctionnement d'un ORM consiste à venir « mapper » la base de données afin de donner l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle. Pour parvenir à un tel résultat, il définit des correspondances entre cette base de données et les objets du langage utilisé. On établit ainsi une passerelle orientée objet entre la base de données et le programme, et on s'affranchit globalement du langage de la base de données.

Les avantages sont multiples :

- Simplification des interactions avec la base de données : les données sont désormais interprétées comme des objets. Pour le développeur, l'emploi est plus que facilité étant donné qu'il n'a plus à écrire de commandes SQL parfois complexes ou alambiquées.
- Possibilités d'effectuer des validations de données avant de les intégrer à la base : on peut par exemple vérifier le type de la donnée à enregistrer ou son format, s'agit-il d'une adresse email, d'un numéro de téléphone ou d'un nom ?
- Expression claire et aisée des relations au sein de la base de données : on manipule désormais des objets.
- Indépendance de l'outil vis à vis du langage de la base de données : il est donc très aisé de migrer vers un autre format au prix simplement de l'installation d'un autre adaptateur pour réaliser une base en SQLlite ou MongoDB.
- Sécurisation du système : en n'écrivant pour ainsi dire jamais du code SQL directement, bien qu'un module le permette dans le cas de requêtes délicates à réaliser autrement, les injections de code SQL malicieux ou corrompu sont pour ainsi dire impossibles.

4. Tests, authentification et sécurité

Un des avantages d'utiliser le framework Sinatra est qu'il se base sur une architecture Rack ([38]).Il est par conséquent possible de lui adjoindre des modules Rack afin de réaliser telle ou telle fonction. Dans le cas des procédures de test et de validation du fonctionnement de l'API, un tel module a été déployé permettant de confirmer le bon fonctionnement des différents services proposés par l'API et l'obtention des résultats escomptés au format précisé dans l'entête de chaque requête http. Un autre avantage des tests unitaires de ce type est de permettre en cas de modification du code d'en revalider le bon fonctionnement rapidement et efficacement au prix de quelques adaptations du fichier de test. Par ailleurs, une fois chaque fonction testée et validée, des scripts de simulation ont été implémentés afin de tester d'une part le fonctionnement global de l'API en conditions d'usage et d'autre part, de pouvoir simuler à volonté le mouvement en temps réel de robots virtuels. Pour ce faire, les scripts ont été rédigés en Ruby à l'aide du gem « net/http » ([39]) qui permet d'émettre des requêtes http et donc d'interagir avec l'API Swarmon. De plus, les logs envoyés proviennent de trajectoires réelles réalisées par le robot voilier VAIMOS d'où des vérifications plus complètes qu'avec des coordonnées aléatoires. Les tests ainsi effectués ont permis de corriger quelques erreurs et de valider finalement cette API.

Une fois le fonctionnement pré-testé, la question de la sécurité des données et de l'API a du être soulevée. Tout le monde ne devait pas pouvoir avoir accès aux données et encore moins en enregistrer, une surcouche de protection et d'authentification a donc été ajoutée. Elle se base sur le protocole d'authentification « Basic » nativement inclus dans le protocole http. Chaque utilisateur identifié se voit allouer un nom d'utilisateur et une clé d'API qu'il lui faut mentionner à chaque requête s'il souhaite pouvoir interagir avec l'API. Au delà de cette protection basique, un système de

rang est utilisé pour donner à chaque intervenant plus ou moins de droits et de possibilités d'emploi de l'API (cf. documentation officielle de l'API pour le détail). Par ailleurs, afin d'éviter qu'une équipe, soit un groupe identifié d'utilisateurs possédant une meute de robots enregistrés, puisse involontairement ou intentionnellement altérer les données d'un autre groupe, il a été informatiquement interdit qu'une équipe puisse modifier les données d'une autre. Pour ce faire, à chaque tentative de connexion et de manipulation de l'API, outre le test d'identification, on vient s'assurer que la requête de modification de données concerne le robot de l'équipe à laquelle l'utilisateur appartient. Enfin, pour des raisons de sécurité, de débuggage et de contrôle de l'activité de l'API, un module de logs a été ajouté permettant de conserver un historique précis des actions réalisées. Pour optimiser l'emploi de ce fichier, une consigne CRON ([40]) a été implémenté dans le serveur afin d'archiver tous les jours, l'activité de la journée passée dans un fichier distinct.

5. Documentation et présentation

La rédaction d'une documentation complète et précise de l'API s'est doublement justifiée. D'une part, dans les API de type REST, la documentation se doit d'être présente et d'expliquer clairement les fonctionnalités de l'API et son mode d'emploi ce qui est conforme à ma volonté d'ouverture et de diffusion de l'API au moins aux participants de la WRSC. D'autre part, ce fut un exercice de style intéressant permettant une remise en cause du code rédigé et un questionnement sur le caractère employable des méthodes proposées. Au final, cela a permis quelques corrections sur la forme plus que sur le fond et ainsi un gain de clarté. Cela a également mis en exergue la nécessité de présenter la base de données sous-jacente afin que les utilisateurs sachent quelles données ils sont en mesure de sauvegarder ou de récupérer et selon quelle organisation. Bien qu'en anglais, la documentation de l'API constitue désormais une base solide pour saisir son mode d'emploi, lui assurer une certaine pérennité et en demeure la meilleure vitrine (http://haggis.ensta-bretagne.fr:3000/).

Un autre aspect, graphique cette fois, se devait également de ne pas être négligé. En effet, face au travail de développement fourni, le fond ne devait pas être desservi par la forme, comme ce fut précédemment le cas pour le client de visualisation. Par conséquent, la mise en page a été soignée. Une charte graphique a été établie et un fichier css rédigé pour l'appliquer. Les couleurs choisies sont relativement proche de celles du client lourd bien que l'API se veuille indépendante de la WRSC. Un des avantages de Sinatra est d'autoriser l'usage de moteurs de templates, dans le cas présent erb ([41]), qui permettent de créer un layout de fond et ainsi de donner rapidement un cachet et une uniformité à l'ensemble des pages du site. Toujours dans cette optique et dans un souci d'achèvement, l'API s'est vu doter d'un nom, Swarmon API, pour Swarn Monitoring et une bannière à ce nom a été créée. Une fois encore, le lecteur est convié à se rendre sur le site de l'API pour se rendre compte de visu des éléments évoqués dans le présent paragraphe.

Une dernière question demeurait à traiter, à savoir sous quelle licence ce travail allait être diffusé. Crée dans une volonté open source et loin initialement de telles questions, il s'agissait d'en définir la portée et les possibilités de réemploi. Apres discussions et mûres réflexions, ce projet sera sous licence GPL 3 ([42]) ce qui donnera aux utilisateurs la possibilité de réutiliser le code mais en en conservant la structure et l'intégralité ainsi qu'en en citant la source et l'auteur.

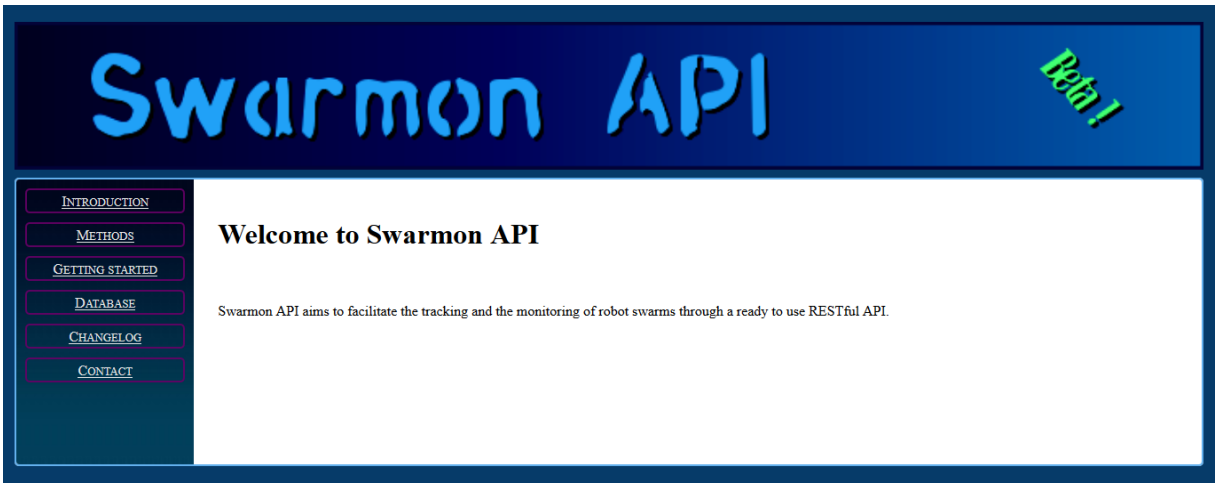


Figure 18 : Aspect visuel de l'API

6. Adaptation et mise à jour de l'existant

Une fois l'API implémentée, le client lourd perdait en l'état de son intérêt et il devenait donc nécessaire de l'adapter d'autant plus que la base de données avait connu quelques changements. L'idéal demeurant qu'il cesse d'interroger directement la base de données pour passer par les services de l'API et ainsi profiter de tous les avantages qu'ils peuvent offrir en matière de sécurité, validation des données, format, … . Le client lourd a donc été considérablement modifié et adapté à la nouvelle architecture du système. Parallèlement, ma meilleure connaissance et compréhension des outils impliqués a également induit une remise en cause de certaines implémentations et la modification voire la suppression de certaines fonctions ou méthodes au profit d'autres plus adaptées aux tâches à accomplir et bien souvent plus génériques donc plus à même d'inscrire ce projet dans une certaine durée. C'est par exemple le cas des scenarios qui, tout d'abord enregistrés dans une table de la base de données, sont désormais des fichiers KML, un format répandu d'affichage de données géographiques. De même, l'affichage des points, balises, portes, … autrefois régi par des fonctions personnelles et posés directement sur le fond de carte, fait désormais appel au système de couches présent dans la grande majorité des services de cartographie qu'ils soient propriétaires comme Google Maps ou le GéoPortail, ou libres tels OpenStreetMaps. Une autre façon de donner davantage de modularité et d'ouverture à ce système.

Lors de l'adaptation du client, une problématique nouvelle a été soulevée, celle de l'accès « cross domain » des scripts JavaScript et plus particulièrement des requêtes Ajax. Un code JavaScript est dit « cross-domain » lorsqu'il est hébergé sur un certain serveur et émet des requêtes http vers d'autres domaines. Par défaut, ce comportement est interdit par les navigateurs pour des raisons de sécurité. Il était donc impossible pour notre client lourd affiché dans un navigateur de dialoguer avec l'API faute d'être situé sur le même serveur. Plusieurs solutions permettent de contourner ce problème. La première consiste à intégrer notre client à l'API, solution que je n'ai pas souhaité retenir préférant conserver une certaine indépendance entre l'outil de visualisation des trajectoires et l'API au rôle, à mon sens, plus général. En outre, l'API se voulant ouverte, quiconque cherchant à développer son propre outil d'affichage ou n'importe quel outil intégré à un navigateur se serait heurté à cet écueil ce qui ne me convenait pas. Une autre approche revient à dire au navigateur que désormais de telles interactions entre les domaines sont possibles. Cette solution n'est pas implémentable et pour cause, elle implique que l'utilisateur final doit posséder les bons paramètres

pour pleinement utiliser l'API. D'autres approches existent dont certaines basées sur la nouvelle définition de l'objet Ajax XMLHttpRequest ([43]) encore en cours d'élaboration par la W3C et à ce titre, encore d'emploi instable. Pour ma part, faute d'une solution miracle qu'une courte recherche bibliographique a révélé utopique, j'ai profité d'un module Sinatra, le gem sinatra-cross_origin ([44]), pour simplifier une partie du travail en autorisant au niveau du serveur le traitement des requêtes cross-domain et j'ai ajouté au programme du serveur une route « options ». Cet ajout qui permet de confirmer l'autorisation de réaliser certaines requêtes http depuis d'autres domaines s'explique par le comportement des navigateurs lorsqu'ils rencontrent de telles requêtes. En effet, ces derniers commencent par émettre vers le serveur distant visé une requête http Options demandant au serveur si ce dernier autorise ou non les requêtes cross-domain et dans quelles conditions. Suivant la réponse du serveur, ils émettent alors la requête initiale ou s'abstiennent, la difficulté technique résidant dans le formatage correct de cette réponse.

Le script tel.php a aussi du être adapté. Si son rôle et son fonctionnement demeure le même, il utilise désormais l'API pour transmettre, à la base de données, les informations de géopositionnement extraites des SMS.

Par ailleurs, un outil de visualisation sous la forme d'un client léger a été développé et intégré directement à l'API afin d'offrir aux utilisateurs la possibilité de visualiser facilement les robots en cours d'évolution. Ce client peut être paramétré de plusieurs façons. Dans l'URL, les coordonnées géographiques du centre de la zone d'évolution des robots doivent être systématiquement indiquées permettant ainsi de facilement changer de zone. Plusieurs robots peuvent être simultanément représentés. En option, un scenario mentionné dans les paramètres de la requête peut être affiché. En outre, cette ressource peut être sollicitée depuis un autre site et affichée sur n'importe quelle page Web à condition de définir un objet html div afin de l'accueillir et de lui attribuer ses dimensions. Cet aspect du client léger peut permettre la promotion de la WRSC ou de l'API et constitue un outil de visualisation prêt à l'emploi.

Au final, cette API globalement respectueuse des fondements du standard REST offre des fonctionnalités intéressantes de monitoring de meutes de robots mais également des possibilités de réemploi et d'extension de ses capacités plus aisées que pour le client lourd qui apparaissait ultra-spécialisé. Sa structure finale tire parti de différents modules Rack qui viennent lui ajouter des fonctionnalités de log, d'authentification, … Une fois complètement testée et améliorée par le retour d'expérience de ses premiers utilisateurs, elle pourra constituer une brique logicielle performante pour de futurs projets de monitoring de meutes de robots.

IV. Tests, bilan et mise en œuvre future

1. Résultats et bilan du projet

Ce projet s'est étalé sur une durée d'environ 6 mois. Si sa conclusion en est globalement heureuse, puisque le projet mis au point est dans sa grande majorité fonctionnel, je m'aperçois avec le recul qu'il m'a environ fallu un peu plus de deux mois et demi pour être pleinement opérationnel et en mesure de définir une architecture répondant aux problématiques soulevées par le sujet et développer un système correspondant à ses spécifications. La démarche adoptée qui a consisté à mettre rapidement sur pied une première architecture s'est avérée plutôt payante, elle m'a permis d'acquérir les bases informatiques nécessaires à l'élaboration de ce projet, principalement en matière de langages employés, et de peu à peu préciser mes objectifs réels et la forme finale que je souhaitais donner à mon système. Fort d'une première base fonctionnelle, il a été plus évident de rechercher et d'investiguer de nouvelles voies en vue d'améliorer le système et de l'optimiser pour ses futures missions. Ainsi, d'une première architecture résolument destinée à la WRSC, il a été possible d'aboutir à une solution plus généraliste basée sur une API, une idée évoquée très tôt dans la définition du projet mais qui n'a pris tout son sens qu'une fois ce dernier amorcé et d'autres pistes plus directes et globalement plus simples en termes d'implémentation essayées. De manière générale, et c'est là sans doute l'un des intérêts majeurs autres que techniques de ce projet, il a nécessité une perpétuelle remise en cause personnelle et de continuelles interrogations quant à l'architecture et les choix techniques réalisés au fur et à mesure d'une meilleure compréhension des outils employés, de la découverte de leurs possibilités et de solutions plus élégantes voire plus performantes. Cette spécificité des projets informatiques dont elle constitue à la fois la force et la faiblesse du fait des possibilités quasi infinies que ce domaine propose, en demeure un aspect extrêmement enrichissant sur les plans professionnel et personnel. D'autre part, l'ouverture d'esprit indispensable à la bonne progression du projet, l'écoute des critiques et des observations de mes encadrants et l'autonomie dans laquelle ce stage s'est déroulé ont constitué des points importants qui ont exigé compétences, organisation, initiative, rigueur et discipline, un état d'esprit que je serai amené à encore développer dans un futur proche.

Ce projet a clairement été l'occasion d'une découverte théorique et pratique des technologies du Web, des technologies bien souvent extrêmement prometteuses et amenées à fortement se développer dans les années à venir. Je pense notamment au langage JavaScript et aux différentes interactions qu'il autorise avec Internet et ses divers constituants, ou encore à WebGL, une technologie et un savoir-faire d'avenir. Néanmoins, en dépit des nouveaux langages découverts et des nouvelles techniques employées, ce projet n'a pas été dénué de liens avec la formation reçue, il a également nécessité des allers-retours permanents entre les connaissances acquises tout au long de ma scolarité et le savoir petit à petit découvert et mis en œuvre. En outre, les modes de raisonnement, la nécessité d'une bonne organisation et d'un certain recul acquis durant les précédents projets ont pleinement pris leur sens durant ce projet.

L'API mise en place autorise des interactions que la première architecture était loin de tolérer. On peut ainsi désormais connecter facilement au système des simulateurs et mélanger robots virtuels et réels, une opportunité qui ouvre de nouveaux champs de recherche bien que les interactions entre chacun demeurent à définir. De même, le rôle de l'API reste pour l'instant relativement passif, elle permet de visualiser les trajectoires des robots. On peut imaginer la mise au

point d'un module au niveau du client d'affichage qui viendrait détecter les risques de collision entre deux robots en extrapolant par exemple les trajectoires futures des robots et en avertirait un opérateur voire même les robots concernés à condition d'ajouter un module de communication au système. Cependant, cette nouvelle architecture bâtie autour de l'API rend plus aisée de tels ajouts et modifications. Dans la même optique, l'emploi d'un ORM, en l'occurrence DataMapper, permet de s'affranchir d'un type précis de base de données et facilite la migration sous d'autres systèmes et organisations au gré des utilisateurs et des besoins finaux.

La base de données demeure néanmoins un des aspects de ce projet sujets à discussions. Si elle est certes adaptée au projet actuel et à sa future utilisation, son emploi dans un contexte plus général ou pour plusieurs projets en simultané risque de nécessiter certaines modifications. Dans un tel cadre, il s'agira de choisir entre conserver une base unique impliquant des moyens d'identification de l'appartenance des données à des projets distincts et des règles d'accès plus contraignantes ou alors l'attribution à chaque projet de bases de données similaires dans leur architecture mais spécifiques et dans ce cas, d'être en mesure soit d'adapter dynamiquement les routes soit de lier la base aux identifiants de chaque utilisateur et de ne la charger que ponctuellement, une fonctionnalité plus aisée à implémenter via des requêtes SQL directes sur les bases qu'à l'aide d'outils tels que DataMapper.

La phase de tests de l'API et de ses fonctionnalités est actuellement en cours. Les tests unitaires présentés tout au long de ce rapport conjointement aux procédés mis en place se sont avérés extrêmement utiles pour corriger quelques erreurs de développement et vérifier le bon fonctionnement des méthodes et l'obtention des résultats attendus. Une fois ces derniers validés, une première série de tests de la plateforme a été réalisée afin de confirmer que le système est globalement opérationnel. Cette campagne menée en Juillet à la fin du développement de l'API a donné toute satisfaction et a permis l'ajout de quelques fonctionnalités et autres raccourcis pour accéder plus facilement aux ressources de l'API à savoir les données de géopositionnement et toutes les informations liées. Depuis début Aout, la plateforme dans sa version beta est ouverte aux équipes de la WRSC qui souhaitent l'essayer. Actuellement, deux équipes se sont manifestées et vont se voir octroyer une clé d'API afin de pouvoir découvrir et expérimenter l'API. Leurs retours d'expérience constitueront un bon moyen de parachever cette plateforme et de s'assurer qu'elle remplit correctement son rôle. Il est en effet probable que malgré les tests entrepris, certaines situations de bugs ou d'emploi n'aient pas été envisagées ou abordées. Ainsi, durant les dernières semaines d'août, il me faudra réagir rapidement afin de régler les derniers bugs et détails de l'API, m'assurer que les spécifications sont bien toutes entièrement respectées et opérationnelles, et éventuellement ajouter à la demande des utilisateurs quelques améliorations à l'API afin d'accroitre son expérience utilisateur.

Enfin, la méthode d'extrapolation des trajectoires mise au point et basée sur des méthodes de calculs par intervalles constitue un moyen original et performant de pallier la difficulté d'augmenter la fréquence d'obtention des données au niveau de la balise Android et de condenser l'information en permettant de diminuer le nombre de messages et de caractères nécessaires pour rendre compte de la trajectoire réalisée. Dans le futur, cette méthode gagnerait à être conservée même si d'autres systèmes de balises étaient déployés ou élaborés, elle reste un moyen efficace et élégant de résoudre des problèmes de localisation lorsque peu d'informations peuvent être échangées en termes aussi bien de fréquence que de contenu.

2. Pistes d'amélioration possibles

Ce projet n'est néanmoins pas entièrement achevé et de multiples pistes demeurent à explorer afin d'en accroitre les fonctionnalités voire d'en étendre le domaine d'emploi. Ci-dessous sont présentées quelques unes des pistes entrevues, celles qui présentent un intérêt certain ou offrent des capacités jusqu'alors négligées ou non explorées faute de temps ou d'une architecture à même de les recevoir.

La première amélioration envisageable porte sur le moteur de cartographie. Actuellement, le système se base sur une solution propriétaire, Google Maps, si cette approche convient amplement au projet et à sa formulation actuelle, il n'en reste pas moins qu'une solution basée sur des outils open source gagnerait considérablement en visibilité, en contrôle des données et des flux d'information et en possibilités d'amélioration et de modification. A ce titre, le projet OpenStreetMap ([45]) associé à la librairie JavaScript OpenLayers ([46]) pourrait constituer une excellente alternative. Si OpenLayers, une bibliothèque de fonctions JavaScript autorisant la mise en place d'applications clientes Web cartographiques fluides, s'avérerait un parfait remplaçant de l'API dédié de Google Maps et permettrait l'affichage et la personnalisation des divers marqueurs et symboles sur des cartes, c'est sans doute le projet OpenStreetMaps qui demeurait l'aspect le plus intéressant de cette alternative. OpenStreetMaps vise à constituer une base de données cartographiques mondiale libre et offre des possibilités d'affichage, de contrôle des données envoyées et de personnalisation des cartes sans commune mesure avec une solution plus stable mais propriétaire. Internet regorge de données numériques de cartographie plus ou moins facilement accessibles par l'intermédiaire d'outils d'édition et sous forme de couches de données. Cette solution possède un fonctionnement en couches ou layers similaires à celui de Google Maps, le caractère libre et open source en plus.

Une deuxième amélioration possible repose sur l'usage des nouvelles technologies du web pour afficher les cartes en 3 dimensions. Les fonds de carte fournies par la NASA en matière de relief ([47]) offriraient un moyen de disposer des données d'altimétrie dans le monde même si leur détérioration volontaire soulève un léger problème de précision variable suivant l'échelle employée (de l'ordre de 90m en Europe contre 30m aux Etats-Unis). Cet aspect n'étant toutefois pas nécessairement primordial pour rendre compte du relief existant. Ces données associées à un moteur graphique WebGL ([48]) directement basé sur le célèbre standard graphique OpenGL permettraient de visualiser en 3 dimensions dans un navigateur, le déplacement des voiliers ou d'une meute de drones. En termes d'ergonomie, d'interactivité et de visualisation des trajectoires effectuées, une telle fonctionnalité pourrait s'avérer extrêmement intéressante à développer.

Figure 19 : WebGL, le futur d'Internet ?

Une autre piste d'amélioration de l'API serait la mise au point d'un flux GéoRSS ([49]). GéoRSS est un standard web permettant d'inclure des données cartographiques au sein d'un flux RSS. Ces flux sont bien souvent nativement compris et interprétés par les applications de cartographie web, c'est ainsi le cas de Google Maps mais aussi d'OpenStreetMaps qui dispose également d'un lecteur de flux RSS adapté. Une telle solution offrirait un autre canal de diffusion des données facilement actualisable au niveau du serveur ainsi que l'opportunité de mieux réguler

l'accès aux données des outils de visualisation des dites données. Ainsi, l'accès actuellement libre aux informations de géolocalisation pourrait être considérablement restreint, seul l'accès au flux GéoRSS serait alors ouvert à tous. L'autre avantage d'un tel dispositif demeure que l'utilisateur lambda c'est-à-dire sans droits particuliers n'interroge plus directement la base de données mais un fichier XML spécifique au contenu entièrement contrôlé.

Deux derniers points restent à évoquer. D'une part, l'éventuelle mise au point d'un système de géopositionnement dédié, le système actuel ayant montré quelques limites, l'emploi de composants précis aux performances connues permettrait de résoudre certains des problèmes rencontrés tout en diminuant en contrepartie les possibilités de rajouter facilement de nouvelles fonctionnalités au niveau de la balise. On s'heurte là aux limites des systèmes dédiés par rapport à ceux plus généralistes. D'autant plus que dans ce cas, on vient comparer développement hardware sur puce et développement software au travers d'un OS qui n'offrent pas du tout les mêmes facilités d'implémentation, de manipulation et de modification. Cette solution technique au coût globalement équivalent si un certain nombre d'unités étaient produites, viendrait agrandir et diversifier le déjà long catalogue des appareils de géopositionnement compatibles avec ce système et dans le cadre d'une commercialisation, offrir un outil simple d'utilisation et directement prêt à l'emploi largement basé sur le principe du « Plug and Play » souvent cher aux utilisateurs actuels.

D'autre part, le système bien que dans sa forme actuelle limité à la visualisation des données et des positions des robots composant l'essaim, pourrait se voir adjoindre un module de contrôle individuel ou collectif du comportement des robots. L'ajout d'un interpréteur de commandes au langage potentiellement spécifique au niveau du smartphone et d'une carte agissant sur les actionneurs du robot permettrait l'envoi et l'exécution de consignes plus ou moins complexes via, par exemple, des SMS ou des requêtes HTTP. Suivant le temps restant, un « concept proof » sera éventuellement mis au point à l'aide d'un smartphone Android et d'une carte IOIO ([50]). Cet objectif de monitoring actif venant s'inscrire dans la droite ligne de ce projet de suivi et de contrôle de meutes de robots et lui offrant une cohérence certaine. Les interactions de cette piste avec le système existant restent néanmoins à définir.

Ces nombreuses pistes restant à explorer montrent que ce projet bien que fonctionnel est loin d'être entièrement achevé, de nombreuses améliorations et optimisations pouvant encore lui être apportées. Cependant, il compose une base logicielle solide à laquelle il demeure aisé d'ajouter des fonctionnalités.

3. La WRSC, une mise en œuvre prochaine

La WRSC constitue l'échéance la plus importante de ce projet. Les organisateurs au sein de l'école comptent sur ma solution de localisation et de suivi de robots afin d'être en mesure de suivre en temps réel les trajectoires des robots voiliers et de pouvoir en analyser lors des épreuves, les tracés à des fins d'attribution des scores. La course sera tout d'abord brièvement présentée puis on reviendra sur ma propre participation à l'événement et l'adaptation de cet outil à ce contexte particulier.

La WRSC (World Robotic Sailing Championship) est une compétition de voiliers autonomes ayant pour objectifs de stimuler le développement de la robotique marine. C'est une compétition dérivée du défi Microtransat, une course transatlantique à la voile pour les robots autonomes. Elle

Figure 20 : Affiche de la WRSC 2013

vise à promouvoir le développement de robots autonomes propulsés par le vent, à travers une série d'épreuves de navigation et des défis. Elle est systématiquement précédée d'une conférence, l'IRSC qui offre aux chercheurs travaillant sur des problématiques liées à la navigation autonome, l'opportunité d'échanger des idées lors d'une conférence scientifique. La sixième édition de cette course se déroule cette année à Brest et est organisée par l'Ensta Bretagne.

Après ce bref rappel des tenants et aboutissants de la compétition afin de bien resituer le contexte qui a baigné ce stage, quelques lignes sur ses implications annexes sur le déroulement de ce stage de fin d'études s'imposent. Globalement, il a tendu à orienter le système et a lui donner une note marine. Même si le choix de garder une solution le plus globale possible a essayé d'être respecté, certains attributs et éléments du système dénotent clairement de l'influence que ce contexte particulier a pu avoir sur ce projet et sa progression. L'exemple le plus flagrant demeure le client de visualisation aux fonctionnalités relativement typées et à la charte graphique définitivement marine. Cette influence a cependant perdu en importance avec la mise en place de l'API, le client lourd n'étant en effet plus la pièce maitresse du projet mais simplement un module facile à modifier voire à transformer.

Le passage par l'API facilite de plus considérablement les interactions avec les données stockées dans la base. Ainsi, un script de validation des missions est en cours de mise au point. Ce dernier dédié exclusivement à la WRSC et écrit en Ruby doit permettre de valider la réussite ou plutôt le pourcentage de réussite d'un robot voilier à diverses épreuves suivant ses logs de position. Pour ses calculs, il se base sur les règles édictées pour cette édition de la WRSC ([51]) et se veut aisé à paramétrer. Les conditions du concours n'étant en effet pas entièrement connues, il doit pouvoir être adapté aux divers aléas climatiques et logistiques qu'il ne manquera pas de rencontrer. Dans les faits, il consiste à projeter dans un repère plan tangent à la zone de manœuvre, les cordonnées géographiques dessinant les trajectoires des robots puis à réaliser divers calculs afin de valider ou non l'épreuve et de définir conformément aux règles de calcul explicitées dans le règlement, les points obtenus à l'épreuve. Un tel script a deux finalités principales, primo, permettre une plus grande objectivité dans l'attribution des notes par les juges qui n'auront plus dans cette tache qu'un rôle secondaire et secundo, son élaboration a permis de confirmer le caractère rigoureux des règles et l'inexistence de moyens flagrants de tricher ou de biaiser les résultats.

Enfin, outre l'activité de développement d'un système de localisation, ma participation à l'organisation de la WRSC comprend également la relecture de trois articles soumis à la conférence précédant la WRSC et la réalisation d'un soutien logistique durant l'événement avec, d'une part, le suivi et la mise en place de mon système et, d'autre part, un support plus général pour l'organisation et le bon déroulement de l'événement. La relecture des articles s'est avérée riche d'enseignements à la fois techniques, les problématiques évoquées étant bien souvent extrêmement intéressantes et à la pointe des recherches en matière de robotique marine mais aussi sur le plan personnel via la confrontation à d'autres modes de raisonnement, de recherche et de développement de projets. J'ai essayé de rendre objectivement des critiques et des observations pertinentes qui, je l'espère, ont contribué à améliorer la clarté et la qualité des articles présentés bien souvent déjà d'excellente facture.

Conclusion

Arrivé au terme de ce projet et face aux objectifs fixés, je dispose désormais d'un outil de localisation et de suivi de meutes de robots opérationnel. L'architecture proposée permet de répondre à la fois à la problématique de ce stage et au besoin d'une solution de tracking pour la WRSC qui aura lieu début septembre.

Le système développé se décompose en deux parties distinctes qui communiquent entre elles via le réseau GSM. On dispose, d'un coté, d'un ensemble de smartphones Android équipés d'une application spécifique, qui les apparente à des balises GPS dotées d'un module de transmission GSM. De l'autre coté, on a une structure de récupération, de traitement et d'enregistrement des données de géopositionnement organisée autour d'une API nommée Swarmon. Sur cette dernière, il est possible de venir greffer divers modules apportant des fonctionnalités supplémentaires au système. On peut ainsi actuellement dénombrer un client de visualisation des données, un script de validation des parcours réalisés par les robots de la WRSC et un module de traitement des SMS envoyées à l'API.

Cette solution repose sur l'usage des dernières technologies Web telles le HTML 5 et le CSS 3, et se veut à la fois modulable, modulaire, facile à adapter, à faire évoluer et à maintenir. Des caractéristique qui, je l'espère, lui garantiront une certaine pérennité et un réemploi dans de futurs projets possédant des enjeux similaires ou nécessitant des outils de suivi de meutes de robots.

Même si du fait des spécificités de ma formation, l'heure n'est plus nécessairement à la maturation d'un projet professionnel désormais bien défini et établi, ce stage n'en demeure pas moins l'opportunité de confirmer et de renforcer des choix professionnels et d'acquérir des connaissances et des compétences en lien avec ce domaine. Ce projet a ainsi été l'occasion d'apprendre de nouveaux langages, de développer de nouvelles compétences principalement informatiques mais aussi de mettre en pratique celles acquises durant ma scolarité. Il reflète assez bien la nécessité pour un ingénieur de se tenir au courant des diverses évolutions technologiques et de perpétuellement se former.

Bien que, dans sa forme actuelle, il constitue une réponse globalement pertinente à la problématique de ce stage, ce projet demeure perfectible, de nombreuses pistes restant à explorer. Il n'en demeure pas moins une brique logicielle substantielle apte à être employée dans d'autres projets.

Sources bibliographiques et références

[1] http://www.ensta-bretagne.eu/wrsc13/

[2] http://en.wikipedia.org/wiki/World_Robotic_Sailing_Championship

[3] http://paginas.fe.up.pt/~jca/wrsc/

[4] http://sauc-europe.org/

[5] http://www.marinetraffic.com/ais/fr/

[6] http://www.dolink.fr/technologie

[7] http://business.tomtom.com/en_us/fleet-management/vehicle-tracking

[8] http://www.mobilefleet.es/

[9] https://maps.google.fr/

[10] http://isea3d.com/

[11] http://developer.android.com/about/index.html

[12] http://www.arduino.cc/

[13] http://www.w3schools.com/html/

[14] http://php.net/

[15] https://developers.google.com/maps/documentation/javascript/reference?hl=fr

[16] https://developer.mozilla.org/en/docs/AJAX

[17] https://developers.google.com/maps/documentation/javascript/usage?hl=fr

[18] http://www.wunderground.com/metarFAQ.asp

FEDERAL METEOROLOGICAL HANDBOOK No. 1 (http://www.ofcm.gov/fmh-1/pdf/FMH1.pdf)

[19] http://www.mysql.fr/

[20] http://www.phpmyadmin.net/

[21] http://www.w3.org/Style/CSS/

[22] http://www.developershome.com/sms/sms_tutorial.asp?page=smsc

[23] http://smssync.ushahidi.com/

[24] http://www.ensta-bretagne.fr/jaulin/polyrobots.pdf

[25] http://qt-project.org/

[26] http://www.emn.fr/z-info/ibex/

[27] http://lisa.univ-angers.fr/Demonstrateurs2013/

[28] http://www.fondriest.com/pdf/airmar_wx_spec.pdf

[29] http://www.figer.com/publications/REST.htm

[30] “Architectural Styles and the Design of Network-based Software Architectures”, Roy Thomas Fielding, 2000 (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)

[31] http://www.sinatrarb.com/

http://sinatra-book.gittr.com/

“Sinatra: Up and Running”, Alan Harris - Konstantin Haase, O'Reilly Media, November 2011.

[32] http://rubyonrails.org/

[33] http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
[34] http://fr.wikipedia.org/wiki/Type_MIME
[35] https://rvm.io/
[36] http://code.macournoyer.com/thin/
https://github.com/macournoyer/thin/
[37] http://datamapper.org/
[38] http://rack.rubyforge.org/doc/SPEC.html
http://rack.github.io/
[39] http://ruby-doc.org/stdlib-2.0/libdoc/net/http/rdoc/Net/HTTP.html
[40] http://manpagesfr.free.fr/man/man5/crontab.5.html
[41] http://www.stuartellis.eu/articles/erb/
http://ruby-doc.org/stdlib-2.0/libdoc/erb/rdoc/ERB.html
[42] http://gplv3.fsf.org/
[43] http://www.w3.org/TR/XMLHttpRequest/
https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
[44] https://github.com/britg/sinatra-cross_origin
[45] http://wiki.openstreetmap.org/wiki/Main_Page
[46] http://openlayers.org/
[47] http://www2.jpl.nasa.gov/srtm/
[48] http://www.khronos.org/webgl/
http://learningwebgl.com/blog/
"WebGL: Up and Running - Building 3D Graphics for the Web", Tony Parisi, O'Reilly Media, 2012
[49] http://georss.org/
[50] https://www.sparkfun.com/products/retired/10748
[51] http://www.ensta-bretagne.eu/wrsc13/index.php/rulesen

Annexes

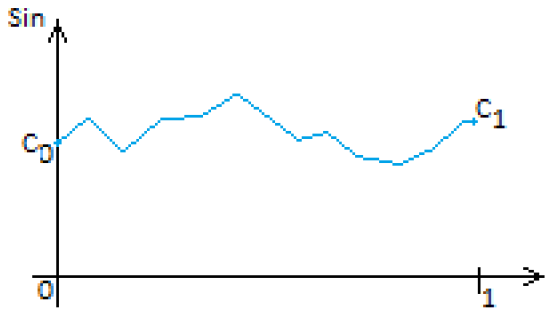
Calculs intermédiaires

Initialement, on a :

$$T_4A_4 = M_{sin}$$

$$[n,4] * [4,1] = [n,1] \text{ où } n, \textit{le nombre de mesures du cap réalisées}$$

Or,



D'où,

$$C_0 = \alpha_0 \; en \; t_i = 0$$

$$\alpha_3 = C_1 - \alpha_2 - \alpha_1 - C_0 \; en \; t_i = 1$$

On injecte ces conditions aux limites dans les équations, on a alors :

$$\sin\theta_i - C_0 = (C_1 - \alpha_1 - \alpha_2 - C_0)t_i^3 + \alpha_2 t_i^2 + \alpha_1 t_i$$

$$\sin\theta_i - C_0 = (C_1 - C_0)t_i^3 + \alpha_2(t_i^2 - t_i^3) + \alpha_1(t_i - t_i^3)$$

D'où,

$$T_2A_2 = M_{sin} - V_{C_0} - (C_1 - C_0)V_{t_i^3}$$

$$T_2A_2 = b \; avec \; A_2 = \begin{pmatrix}\alpha_1 \\ \alpha_2\end{pmatrix} \; et \; T_2 = \begin{pmatrix}(t_i - t_i^3) & (t_i^2 - t_i^3)\end{pmatrix}_{i \in [\![0,n-1]\!]}$$

Donc

$$T_2A_2 = b \iff A_2 = (T_2^T T_2)^{-1} T_2^T b$$

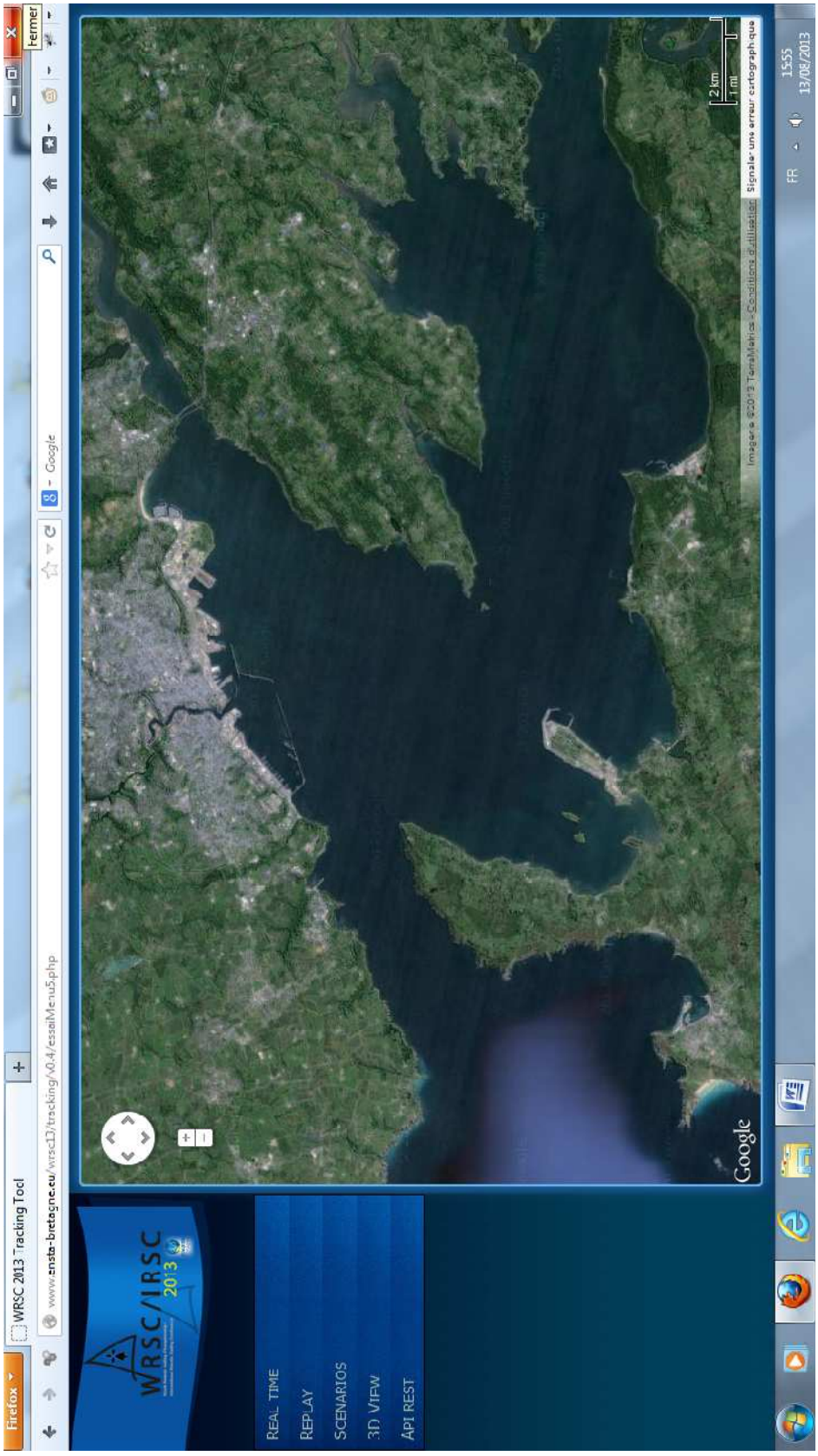


Figure 21 : Dernière version du client "lourd"

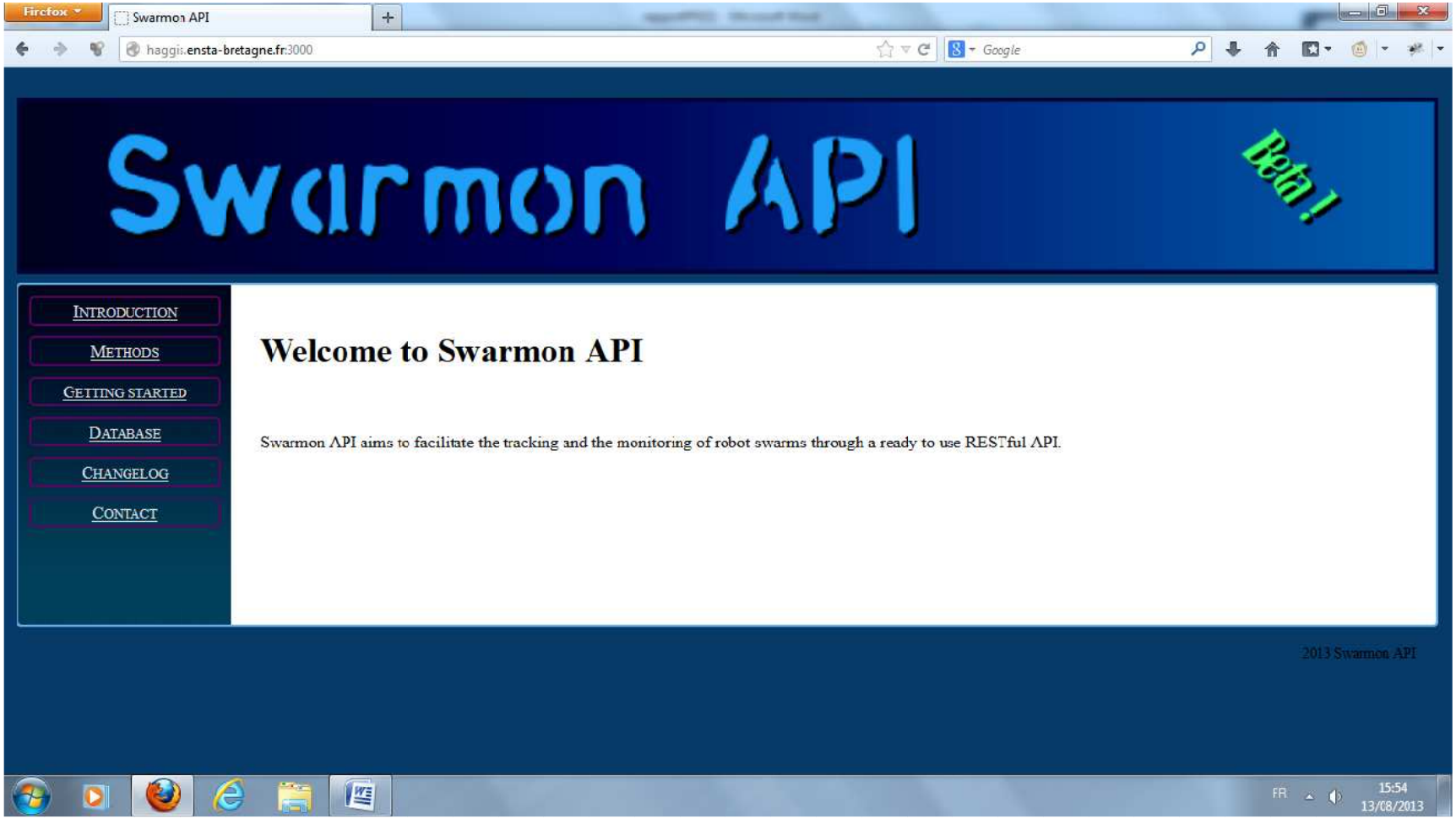


Figure 22 : Page d'accueil de l'API

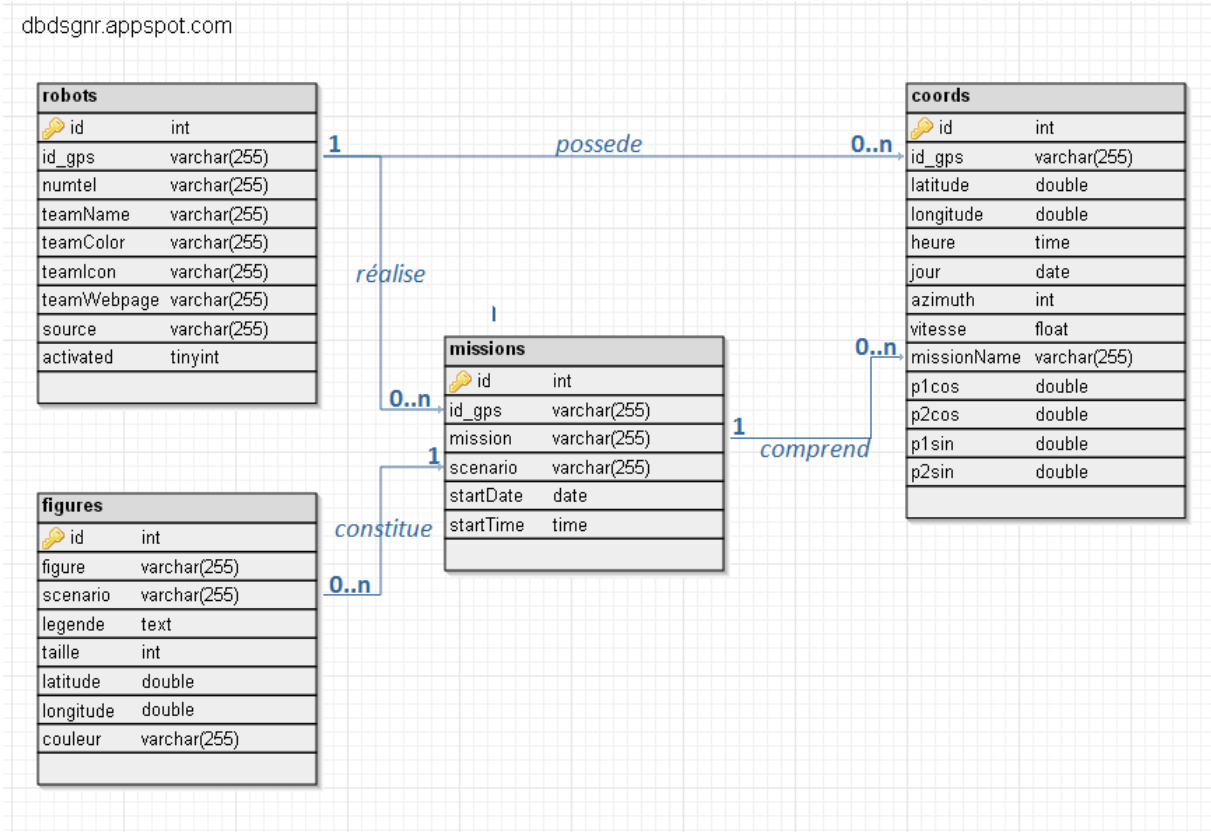


Figure 23 : Première architecture de la BDD

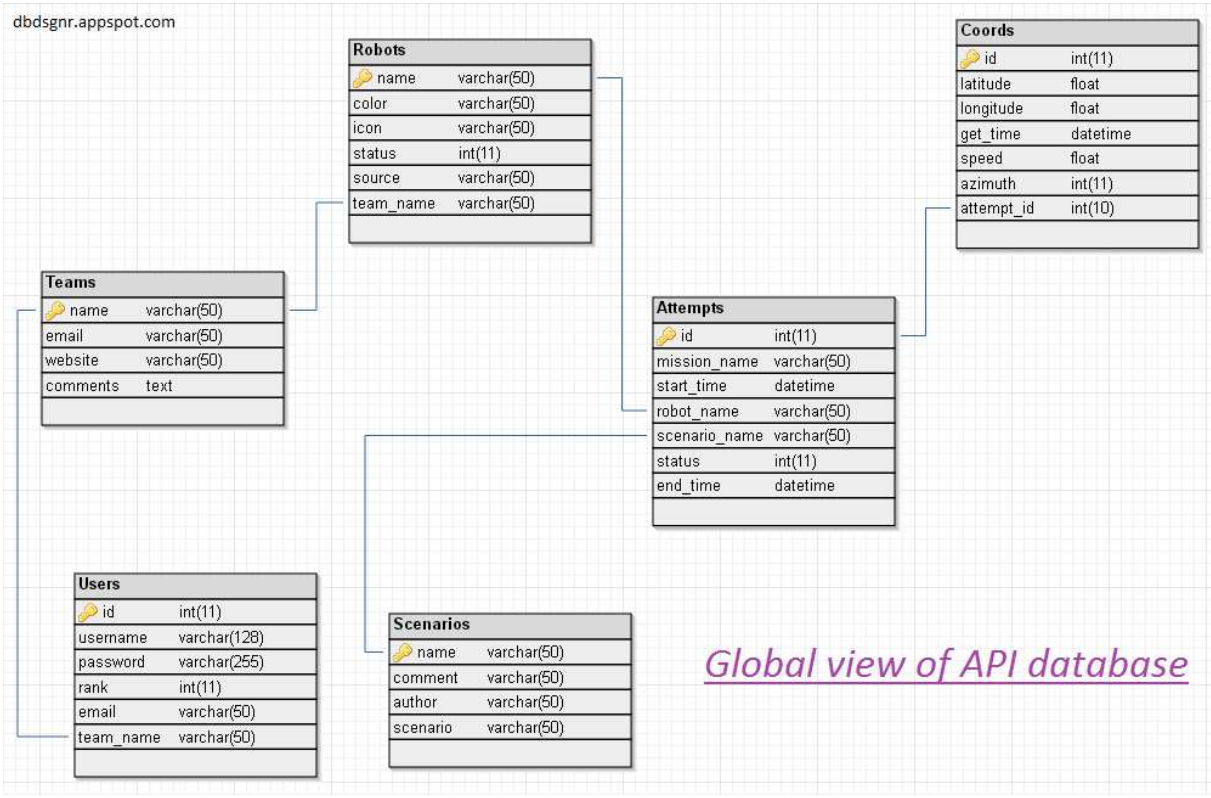


Figure 24 : Architecture finale de la BDD

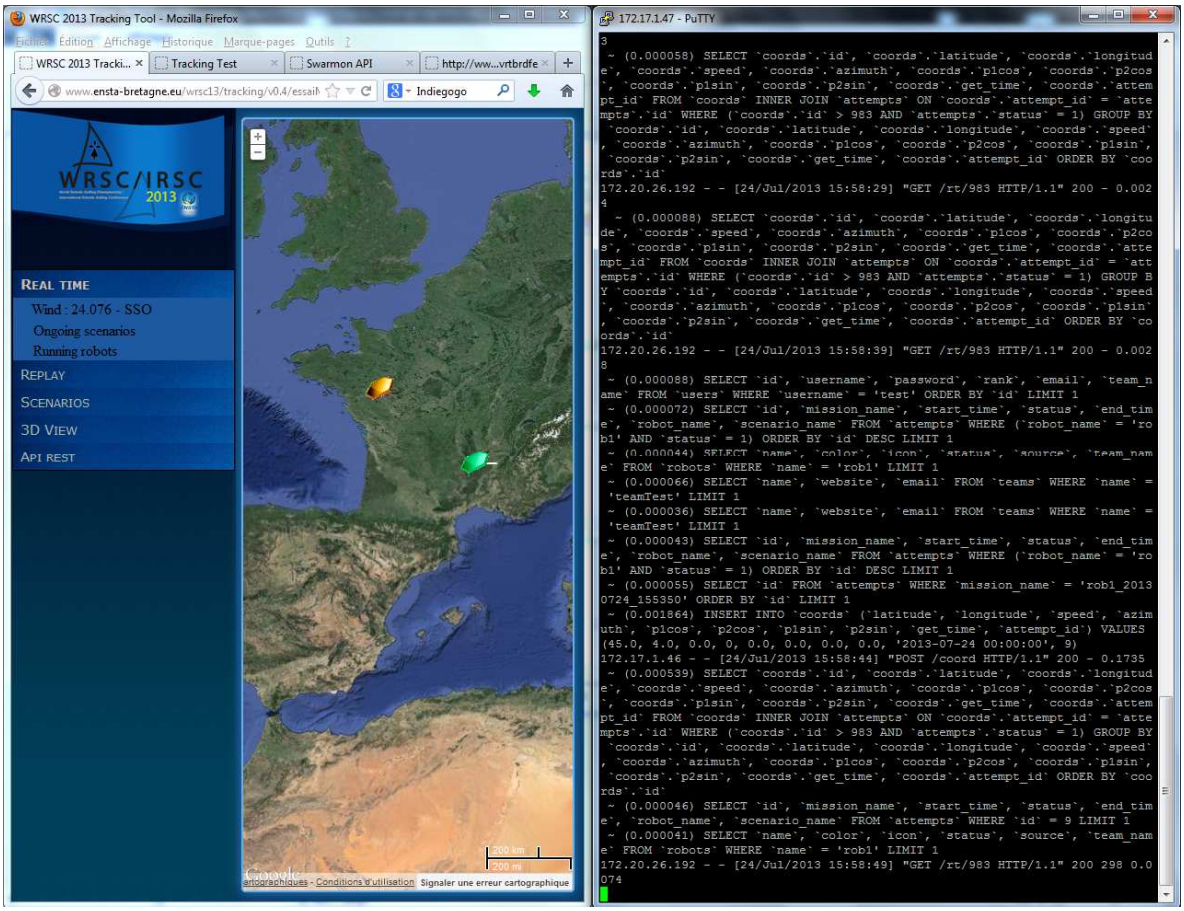
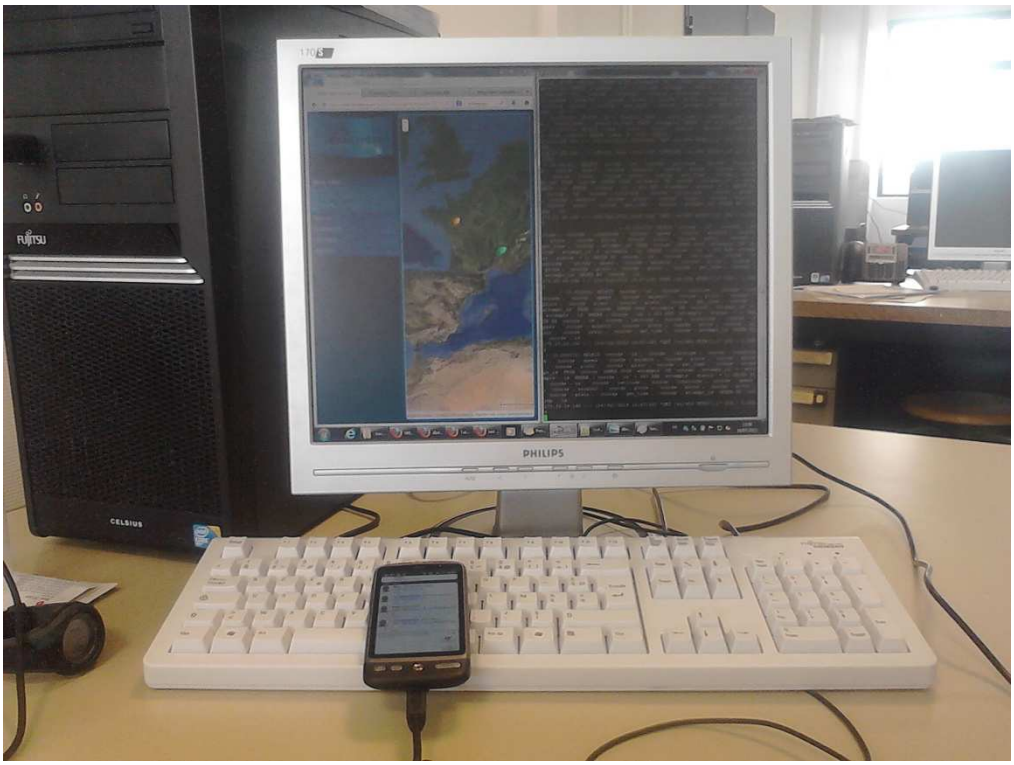


Figure 25 : Système en fonctionnement



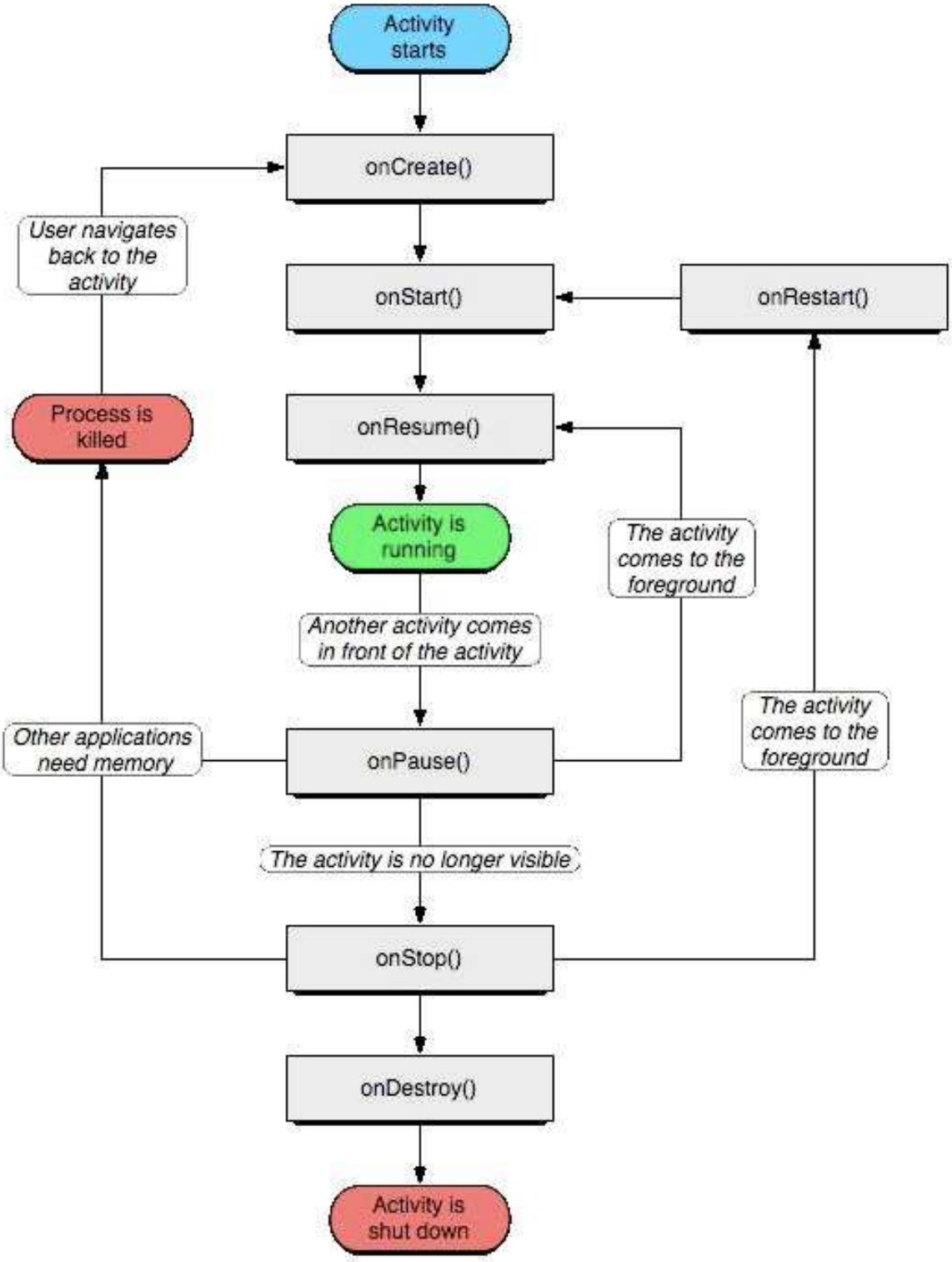


Figure 26 : Cycle d'une activité Android

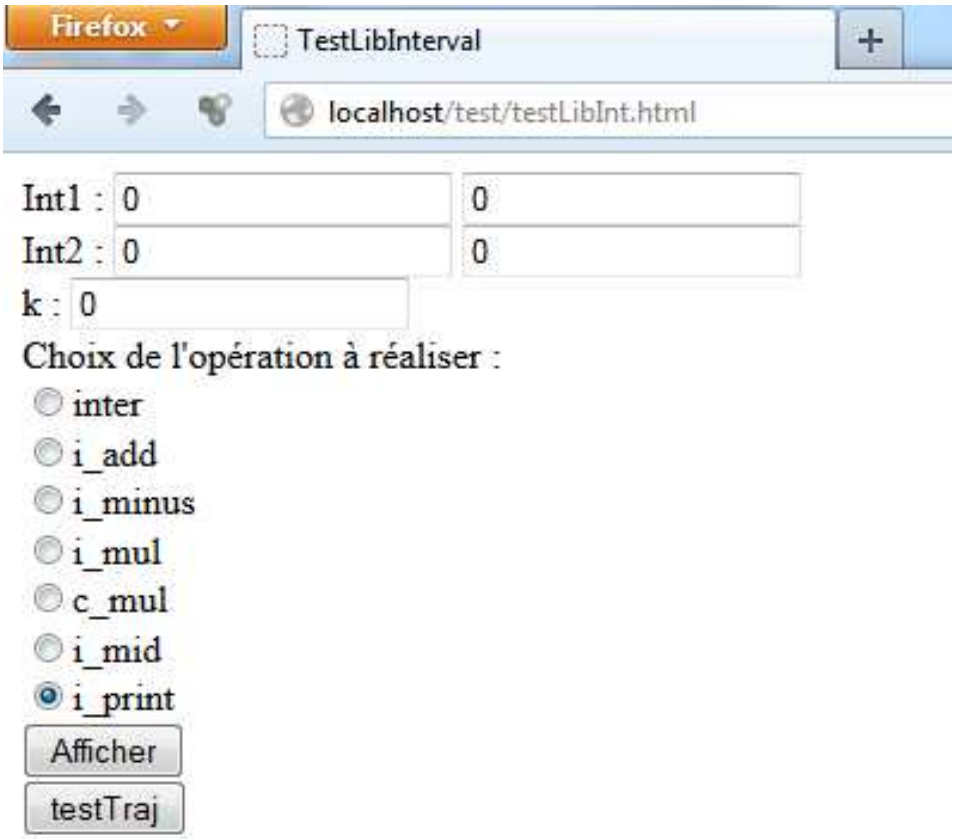


Figure 27 : Outil de test de la bibliothèque JavaScript