Université Nice SOPHIA ANTIPOLIS

Institut Universitaire de Technologie Nice Côte d'Azur

Rapport de stage pour l'obtention du diplôme universitaire technologique
Session 2011 – 2013

Création d'un outil interactif dédié au contrôle à distance de l'instrumentation PICARDSOL

Présenté par Gauthier Cibert--Volpé

Observatoire de la CÔTE d'AZUR

Remerciements

Je tiens à remercier dans un premier temps, toute l'équipe pédagogique de l'IUT de Nice et les intervenants professionnels responsables de ma formation, pour avoir assuré la partie théorique de celle-ci.
Je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance aux personnes suivantes, pour l'expérience enrichissante et pleine d'intérêt qu'elles m'ont fait vivre durant ces dix semaines au sein de l'entreprise :

- Tout le personnel de l'Observatoire de la Côte d'Azur pour m'avoir permis d'être à l'aise dès le début de mon stage.
- L'équipe PICARD-SOL pour m'avoir accueilli. Et plus précisément :
  - Catherine Renaud pour avoir porté autant d'intérêt à mon stage et à sa poursuite et pour m'avoir aidé dans mon projet tout au long de ce dernier.
  - Frédéric Morand pour m'avoir beaucoup appris sur des domaines ne relevant pas forcément de l'informatique.
  - Thierry Corbard pour m'avoir permis de monter aussi souvent sur le plateau de Calern
  - Rabah Ikhlef et Mamar Fodil pour m'avoir accueilli dans leurs bureaux et m'avoir permis de passer ce stage dans la bonne humeur
  - Et tant d'autres que j'oublie.

Résumé

Pour l’obtention du Diplôme Universitaire de Technologie (DUT) en Informatique, un stage de fin de parcours d’une durée minimale de dix semaines est obligatoire. En ce qui me concerne, il a été effectué du 8 avril au 14 juin 2013 à l’Observatoire de la Côte d’Azur et plus précisément au sein de l'équipe PICARD-SOL.

L’instrumentation PICARD-SOL est chargée d'effectuer des mesures au sol sur le plateau de Calern de l'Observatoire de la Côte d'Azur dans le cadre de la mission spatiale PICARD. Son objectif est de mesurer le diamètre solaire simultanément au sol et dans l'espace afin de corréler les mesures au sol et celles faites dans l’espace et d'apprendre à interpréter les mesures du sol grâce à une analyse détaillée de la turbulence atmosphérique.

Le stage porte sur la spécification, le développement et la validation d’un outil permettant aux membres de l'équipe de l’observatoire de surveiller et commander à distance des instruments servant à la mesure du diamètre solaire et à l’étude de la turbulence atmosphérique. L'objectif est, à terme, de faciliter le processus d'observation.

Ces instruments se trouvant à Calern, les phases d’analyse et de déploiement s’y sont donc effectuées.

Pour y parvenir, j’ai dû utiliser beaucoup de mes connaissances acquises à l'Institut Universitaire Technologique mais j'ai également été amené à découvrir des concepts et méthodes nouveaux. Et j'ai travaillé avec des outils nouveaux, dont je parle dans ce rapport, qui m'ont aussi beaucoup apporté .

Les fonctionnalités essentielles ont pu être implémentées et testées à la fin du stage.

Summary

To obtain a DUT in computer science, an internship during a minimum of ten weeks is required. In my case, it was conducted from April 8 to June 14, 2013 at the Observatory of Cote d'Azur and more specifically within the PICARD-SOL team.

The instrumentation PICARD-SOL is responsible for conducting ground-based measurements on the board Calern the Observatory of Cote d'Azur in the PICARD space mission. Its aim is to measure the solar diameter simultaneously from ground and space in order to correlate the ground measurements and those made in space and learn to interpret ground measurements through a detailed analysis of atmospheric turbulence.

The internship focuses on the specification, development and validation of a tool for some users of the observatory to remotely monitor and control instruments for measuring the solar diameter and the study of atmospheric turbulence. The objective is to greatly facilitate the observation process.

These instruments are located at Calern, analysis and deployment phases were therefore made at this place.

To achieve this, I used most of the knowledge I acquired during my studies, but I also was led to discover new concepts and methods. I also worked with new tools to me.

The essential features have been implemented and tested at the end of the internship.

Table des matières

Introduction

Ce document constitue le rapport de stage de DUT informatique effectué du 8 avril au 14 juin 2013 au sein de l'Observatoire de la Côte d'Azur.
Ce projet porte sur la création d'un outil dédié à la facilitation des observations de l'équipe PICARD-SOL. L'objectif est d'assister au maximum ces dernières pour permettre aux observateurs de se concentrer sur d'autres tâches.

Le fonctionnement prévu rend difficile le déploiement de clients dédiés. Il s'agit donc d'une interface web accessible depuis un navigateur web pour permettre aux utilisateurs du système de ne plus avoir à être physiquement présents à proximité des instruments (mis à part une personne physiquement présente dans le cas d'une intervention d'urgence).

Pour cela, les principales étapes de mon travail ont été la phase d'analyse (une discussion avec l'équipe a permis l'établissement de l'expression du besoin puis du cahier des charges), la phase de conception puis la programmation développée en parallèle de la phase de tests.

Sont présentés dans un premier temps l'Observatoire de la Côte d'Azur (puis plus précisément l'instrumentation concernée par le projet) puis le projet dans sa globalité avant d'aborder les différentes phases qui ont accompagnés la réalisation du projet.

L’Observatoire de la Côte d’Azur

L’Observatoire de la Côte d’Azur (OCA) est un établissement public à caractère administratif (EPA).
L’OCA regroupe et pilote les activités de recherches en Sciences de la Planète et de l’Univers de la région azuréenne effectuées au sein de 3 unités de recherche : Artémis, Géoazur et Lagrange.
L’observatoire de la Côte d’Azur est installé sur 4 sites : le plateau de Calern, le Mont Gros où se trouve le siège social (site historique de l’Observatoire de la Côte d'Azur), le campus de Valrose à Nice et les locaux de Sophia-Antipolis.

Lagrange

Le laboratoire J.-L. LAGRANGE est une Unité Mixte de Recherche de l’Observatoire de la Côte d’Azur, du CNRS et de l’Université de Nice-Sophia Antipolis. C’est un laboratoire pluridisciplinaire qui regroupe des équipes d’astrophysique (planétologie, physique stellaire et solaire, galaxies et cosmologie), de mécanique des fluides et de traitement du signal et images.
Des compétences transverses en instrumentation pour l’observation astronomique à haute résolution spatiale et en calcul à haute performance sont au cœur des capacités des équipes à développer de nouvelles théories et modèles pour les confronter à des observations acquises sur les grands télescopes au sol et dans l’espace. Le laboratoire LAGRANGE a été formé au 1er janvier 2012 par la fusion du Laboratoire FIZEAU et du Laboratoire CASSIOPEE. Le personnel ainsi que les moyens matériels du laboratoire sont actuellement répartis sur 3 sites : Observatoire de Nice au Mont Gros, Université de Nice sur le campus de Valrose et plateau de Calern

PICARD-SOL

Présentation

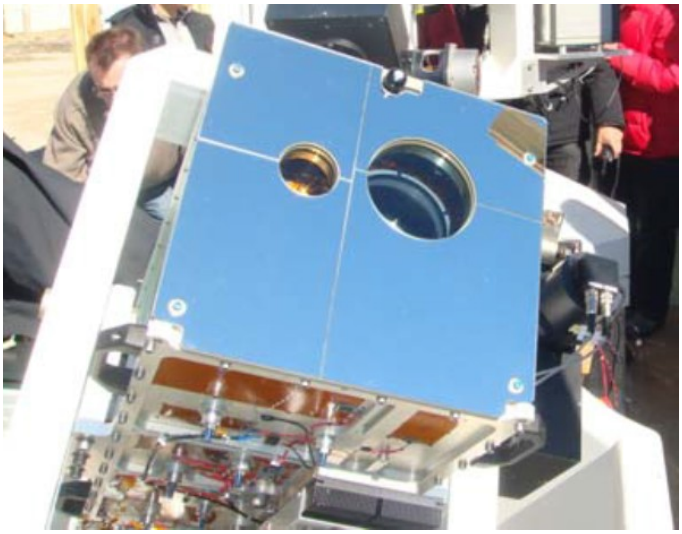
PICARDSOL est un ensemble d'instruments basés au sol qui sont exploités sur le plateau de Calern de l'Observatoire de la Côte d'Azur dans le cadre de la mission spatiale PICARD. Son objectif est de mesurer le diamètre solaire simultanément au sol et dans l'espace et d'apprendre à interpréter les mesures du sol grâce à une analyse détaillée de la turbulence atmosphérique.

Intérêt

Pour éviter l'influence de la turbulence atmosphérique, des mesures sont prises en orbite. Cependant, il est nécessaire de comprendre et d'interpréter les mesures effectuées à partir du sol, qui sont plus faciles à mettre en œuvre et donc beaucoup moins coûteuses.
C'est pourquoi un important programme de mesures au sol est associé à l'opération de spatiale avant, pendant et après la mission PICARD : PICARD-SOL.

Instruments

SODISM-II :
SODISM 2 est un télescope imageur de 11 cm de diamètre doté d'une caméra CCD fr 2048x2048 pixels. Il permet d'enregistrer des images du soleil à 5 longueurs d'onde différentes. C'est la copie conforme de SODISM, télescope embarqué sur le satellite picard lancé en juin 2010.
SODISM2 est installé dans une cuve lui permettant de fonctionner dans des conditions les plus similaires à celles de l'espace.
Il est opérationnel depuis mai 2011 et fournit quotidiennement des séries d'images qui sont utilisées pour mesurer, après traitement, le rayon solaire dans les différentes longueurs d'onde.



MISOLFA :
MISOLFA est un instrument destiné à qualifier les observations diurnes. C'est un télescope de 26 cm de diamètre associé à une boite focale à 2 voies qui enregistrent les effets de la turbulence atmosphérique sur le bord solaire.
La voie image permet de former, à l'aide d'une caméra CCD, des images de 2 bords opposés du soleil.
La voie pupille permet de mesurer, à l'aide de fibres optiques couplées à des photo-détecteurs, les fluctuations de l'intensité lumineuse reçue par le télescope.

Procédure d'observation

Les observations se déroulent en trois phases :

- Mise en route : Cette phase consiste en la mise en service de chaque télescope et monture. Elle nécessite une attention particulière concernant la cuve de dans laquelle SODISM2 est intégré (mise en service de la régulation de température, gestion de la pression).
- Observation:
  La phase observation est relativement simple et répétitive elle consiste à surveiller régulièrement les dernières images fournies par l'instrument lors des différents plans (dont la durée est d'environ 3 heures) et de constater si le soleil est correctement centré. S'il ne l'est pas, il faut alors appliquer manuellement une correction pour la monture en rentrant les coordonnées correctives à appliquer. Il faut également surveiller en permanence l'état des paramètres instrumentaux.
- Fermeture : Cette phase correspond à la mise à l'arrêt manuelle de tout le matériel de l'instrumentation.

Le projet ne s'intéresse qu'à la phase d'observation car les phases de mise en route et de fermeture ne sont pas contrôlables à distance.

Présentation du projet

Objet du stage

Les observations effectuées dans le cadre de l'instrumentation PICARD-SOL nécessitent une présence humaine. Actuellement, les observateurs sont techniquement contraints d'être présents à proximité des instruments pour effectuer des corrections, surveiller les observations.
L'objet de ce stage est de spécifier et développer un outil permettant de s'affranchir de cette contrainte et d'assister au mieux ces derniers pour leur permettre à terme dde se consacrer à d'autres tâches pendant la phase d'observation (traitement de données, …).

On peut donc décrire le contenu du projet en discernant trois parties :

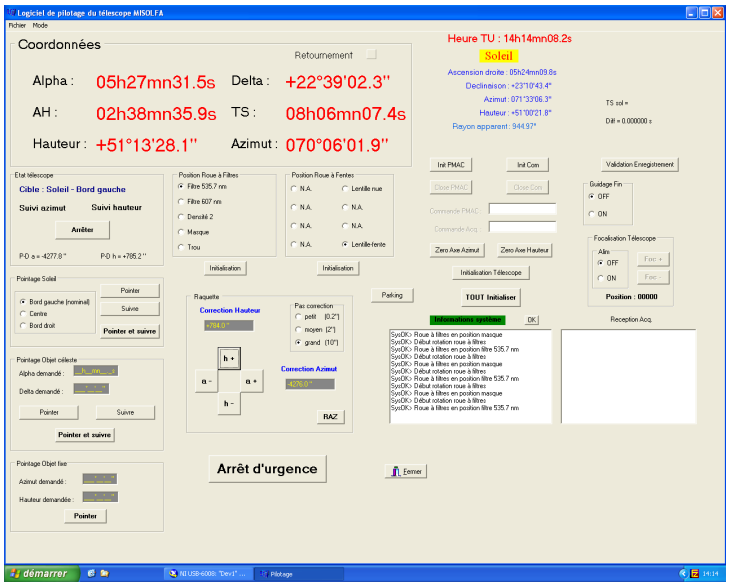
- Récolter les informations à transmettre aux utilisateurs
- Distribuer les informations
- Transmettre d'éventuelles commandes aux systèmes déjà en place

Nous avons convenu que cet outil s'utiliserait par le biais d'un navigateur de manière à ce qu'il soit disponible facilement et sans installation.

Ce qui était en déjà place

Les instruments se commandent à partir des logiciels présents sur les ordinateurs de l'instrumentation.

Les logiciels d'acquisition et de pilotage déjà en place permettent un contrôle total des instruments.
Les concepteurs de ces derniers ont prévu la greffe d'un tel outil puisque le squelette d'une connexion avec cet outil est déjà implémenté.



Cahier des charges

Besoins

Besoins et contraintes :
À l'établissement de la première version du cahier des charges, certains besoins ont été dégagés :
Des besoins principaux :

- B1 : Contrôler rapidement qu'une séance se déroule correctement.
- B2 : Consulter les informations relatives aux instruments
- B3 : Commander les instruments

Et d'autres secondaires :

- B4 : Voir qui est en train de visualiser (pour un observateur et un administrateur)
- B5 : Tenir un journal (quelles commandes ont été effectuées, par quel observateur …)

Fonctions

Ces besoins ont été traduits en fonctions à implémenter :

- Fonctions de service principales :
  - F1 : Authentifier un utilisateur selon son statut (Administrateur, Observateur, Visualisateur)
  - F2 : Permettre à un utilisateur de visualiser en temps réel certaines informations (déterminées selon son statut)
  - F3 : Permettre à un utilisateur d'exécuter certaine commandes (déterminées selon son statut)
  - F4 : Transmettre les commandes reçues et informer l'utilisateur de l'exécution de ces dernières
  - F5 : Notifier sonorement et/ou visuellement à l'utilisateur qu'un événement requiert son attention.

Caractérisation des fonctions

Les fonctions peuvent se décrire plus précisément de la façon suivante :

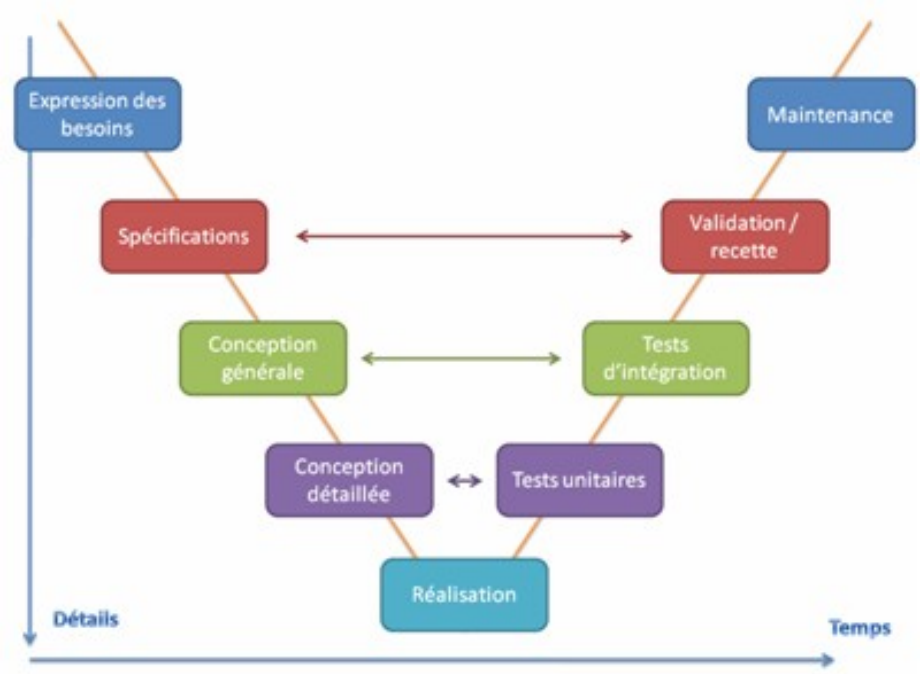
- F1 : Authentifier un utilisateur selon son statut (Administrateur, Observateur, Visualisateur)
    - *Buts : Limiter le droit d'interaction selon le statut de l'utilisateur. Il est prévu que le contrôle se fasse par identifiants (du système d'annuaire de l'observatoire)*
- *F2 : Permettre à un utilisateur de visualiser en temps direct certaines informations (déterminées selon son statut)*
    - *Buts : Les informations que pourra consulter un utilisateur dépendront de son statut. Typiquement, ici, un visualisateur aura un degré de visualisation moins important qu'un observateur qui, lui-même, aura un degré de visualisation moins important qu'un administrateur*
- *F3 : Permettre à un utilisateur de demander certains contrôles (déterminées selon son statut)*
    - *Buts :* Associer les différentes commandes à chacun des types d'utilisateur.
- *F4 : Transmettre les contrôles reçus et informer l'utilisateur quant à l'exécution de ces derniers.*
    - *Buts : Faire en sorte que les commandes requises par l'utilisateur soient transmises au système déjà existant et offrir une confirmation de la bonne exécution ou, a contrario une notification d'erreur.*
- *F5 : Notifier auditivement et/ou visuellement l'utilisateur d'une situation qui requiert son attention.*
    - *Buts : Il est important que l'utilisateur soit averti dans certains cas (fin d'un plan, alarme ...)*

Planning prévisionnel

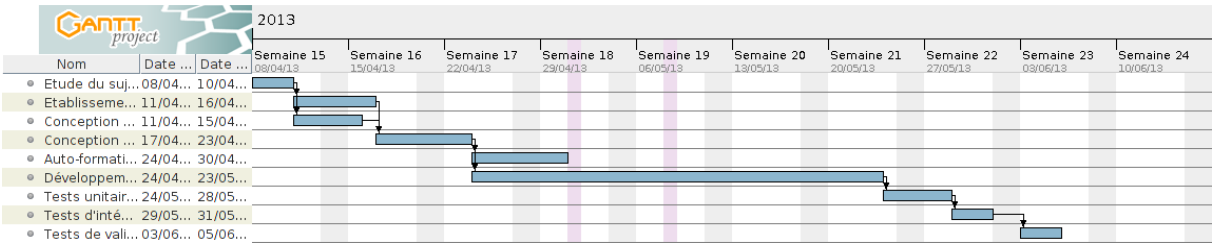
Conformément aux méthodes de gestion de projet apprises dans le cadre de l'IUT, un processus d'organisation et de planification a été effectué avant la réalisation du projet.

Le cycle en V

Le cycle de vie du logiciel désigne toutes les étapes du développement d'un logiciel, de la définition des besoins jusqu'à la maintenance. L'objectif du cycle en V est de définir des étapes intermédiaires pour parvenir à la validation du logiciel. Il s'agit donc d'une perpétuelle vérification de la conformité du logiciel avec les besoins exprimés par la maîtrise d'œuvre. Le modèle du cycle en V est le modèle structurel de développement utilisé pour ce projet.



Planning prévisionnel



Ce diagramme représente la charge de travail prévisionnelle, après élaboration du cahier des charges. On peut voir les phases d'auto-formation et de développement s'effectuer simultanément après la conception.

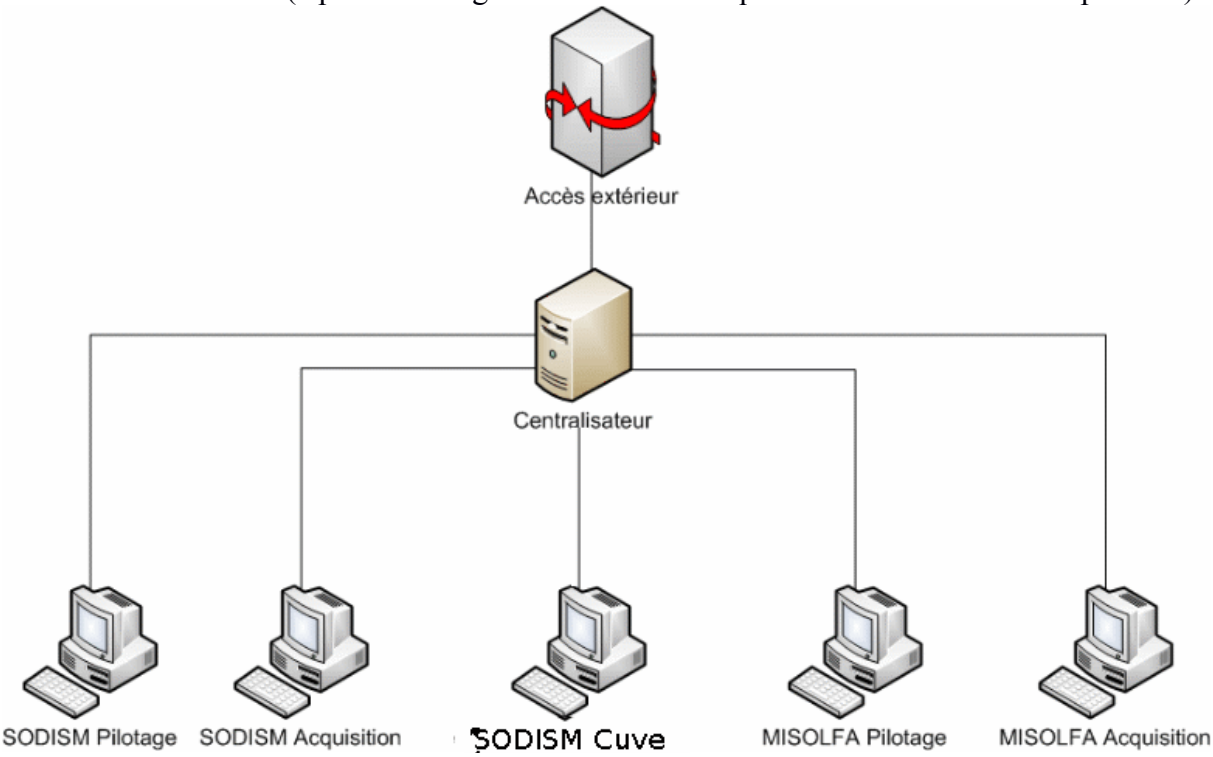
Analyse et conception

Analyse

Architecture en place

L'instrumentation PICARDSOL est composée de 6 ordinateurs :

- Pilotage SODISM
- Cuve SODISM (Sert à la gestion de l'environnement de SODISM : Pression, température, ...)
- Acquisition SODISM
- Pilotage MISOLFA
- Acquisition MISOLFA
- Centralisateur (a pour but de gérer les données acquises et de les mettre à disposition)



Le centralisateur étant le cœur de la manipulation, ce dernier est voué à accueillir le serveur d'information

Sources d'information

Les sources d'information sont au nombre de trois :

- Fichier House Keeping : Fichier texte contenant toute les informations relatives à l'environnement d'un instrument et mis à jour périodiquement. Il peut se trouver dans différents formats selon l'origine du fichier (logiciel de contrôle pilotant le sous-instrument concerné).
- Fichier fits (images) : Résultat des acquisitions au format .fit déjà récupérés par ftp grâce à un daemon. Pour qu'ils soient exploitables, il faut les convertir en un format géré par la plupart des navigateurs. Pour SODISM, les acquisitions ne se font pas en temps réel. Par conséquent un fichier fit à peine reçu peut correspondre à une acquisition effectuée 5 minutes plus tôt.
- Liaison directe avec un ordinateur de PICARDSOL : Les logiciels s'occupant de gérer les instruments ont été prévus pour établir une connexion (par sockets), échanger des données, gérer des commandes etc.

Statut des utilisateurs

Après étude, nous avons défini que les utilisateurs doivent être séparés selon le niveau de privilèges auxquels ils ont accès. On parle alors de « statut » d'utilisateur :

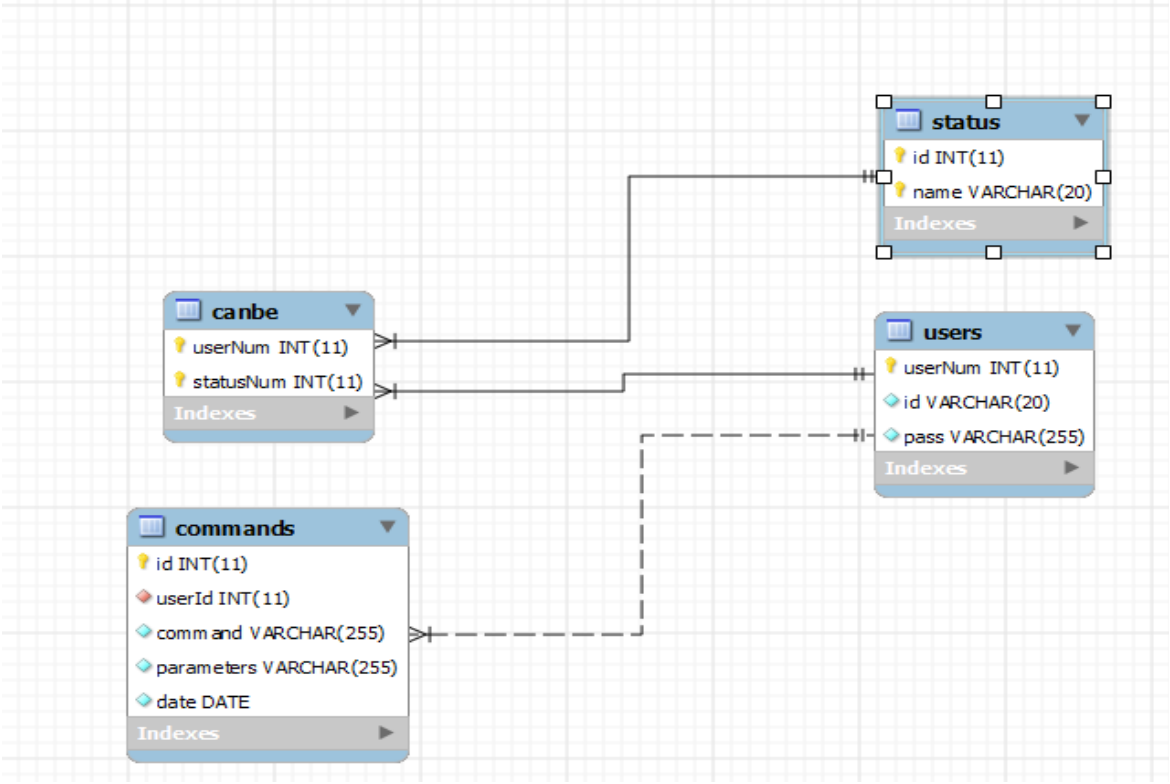
- Administrateur :
- Observateur : Il ne doit y en avoir qu'un connecté au système à un instant donné (la personne chargée des observations)
- Visualisateur : Il peut uniquement consulter les informations mais ne peut pas envoyer de commandes. Il doit quand même être référencé auprès du système.

Pour des raisons pratiques (un id pour un utilisateur), un utilisateur peut endosser différents statuts (Administrateur, Observateur, Visualisateur). Typiquement, un observateur peut endosser le rôle d'un visualisateur (par exemple lorsqu'un observateur est connecté) et plusieurs utilisateurs peuvent être visualisateurs lorsqu'une observation est en cours.

Conception

Base de données

La base de données sert à stocker identifiants et mots de passe des différents utilisateurs et modéliser les rôles qu'ils peuvent endosser. Elle sert également à stocker les informations concernant commandes qui sont retransmises par le serveur d'information et ainsi à garder une trace de celles-ci.



Sur cette vue relationnelle, on distingue quatre tables :

- users : Cette table sert à stocker l'identifiant et le mot de passe d'un utilisateur
- status : Cette table recense tous les statuts qu'un utilisateur peut endosser
- canbe : Cette table sert à associer un utilisateur à tous les statuts qu'il peut endosser
- commands : Cette table remplace le journal. A chaque commande émise par l'utilisateur, une entrée est ajoutée et permet d'identifier l'origine de la commande, le nom de la commande, les paramètres et l'heure précise où elle a été transmise.

Séparation du serveur d'information et du serveur web

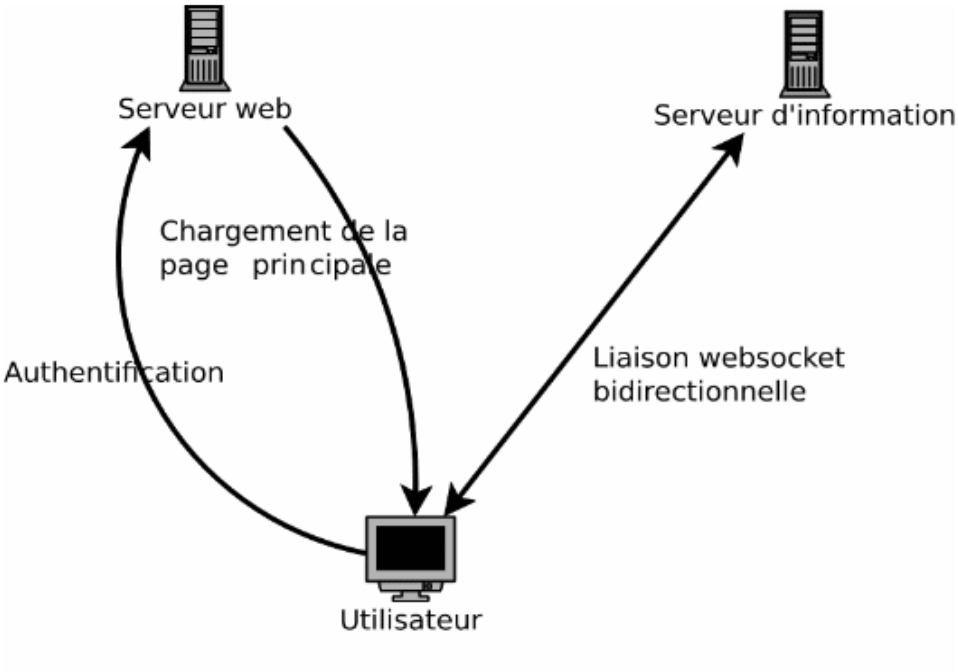
La solution finale a deux rôles bien distincts. En effet, la solution est une page web et les informations doivent être récoltées et mises à jour de façon continue. Un simple serveur HTTP pourrait à lui seul effectuer la tâche mais la conception d'une telle solution serait confuse.

De plus ce serait ignorer les besoins d'instantanéité exprimés par la maîtrise d'ouvrage.

Une séparation s'avère donc nécessaire avec, d'une part, un serveur web dont le rôle est simplement de transmettre une page web et permettre l'authentification d'un utilisateur et d'autre part, un serveur qui est chargé de récolter les informations (localisé sur place donc) et de les transmettre à l'utilisateur. Ce dernier a été baptisé « Serveur d'information ».

En plus d'être plus cohérente, cette structure permet une portabilité accrue du système (permettant par exemple de localiser le serveur web et le serveur d'information sur différentes machines, sur différent réseaux...).
Par conséquent, deux serveurs sont lancés :

- Le serveur web : Il se contente d'authentifier l'utilisateur au moyen d'un formulaire et de générer une page selon le statut de ce dernier.
- Le serveur d'information : Une fois l'utilisateur authentifié et la page chargée, le navigateur établit une connexion (websocket) avec le serveur d'information qui prend le relais et transmet les informations à l'utilisateur.



Ce schéma illustre le mode de communication du système.

L'utilisateur s'authentifie auprès du serveur web qui lui enverra alors une page rédigée selon le statut de l'utilisateur. Une fois cette page chargée, elle va établir un canal de communication avec le serveur d'information et communiquer selon un certain protocole.

Choix des websockets

Le projet nécessite l'établissement d'un canal de communication bidirectionnel sécurisé entre un navigateur web et un serveur. Plusieurs solutions ont été étudiées (AJAX, Silverlight, flash) et nous avons établi que le moyen le plus propre est de faire usage des web sockets.

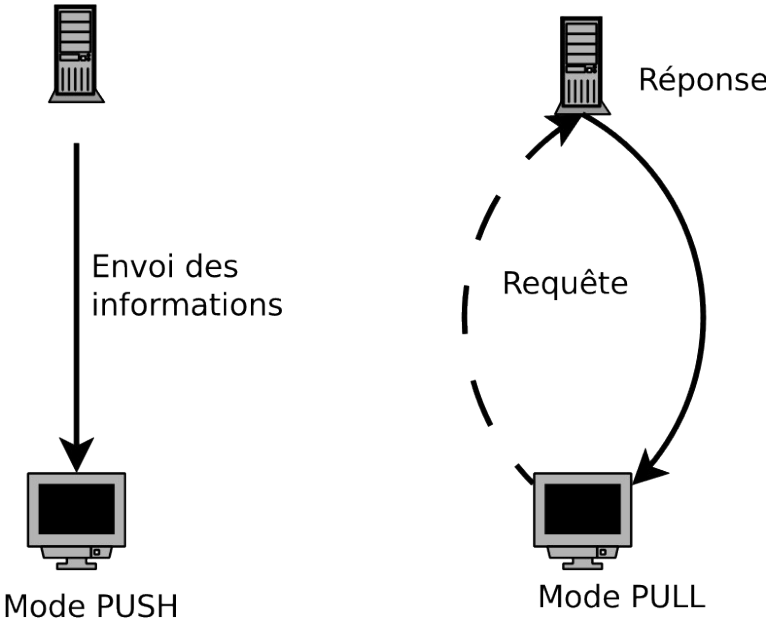
En effet, ces derniers, à travers le protocole wss sont idéaux pour le fonctionnement prévu.

Protocole d'échange entre le client et le serveur

Le canal établi, nous avons défini un protocole de communication entre le client et le serveur. Avec ce protocole, deux modes ont été retenus:

- Mode push : Le serveur envoie les informations au client sans que celui-ci ne demande rien (généralement à intervalles réguliers).
- Mode pull : Le client envoie une requête au serveur qui lui envoie l'information demandée. Exemples types : un bouton « Rafraîchir », l'envoi d'une commande.

Ces deux modes permettent la bonne communication des informations et des commandes dans les deux sens.

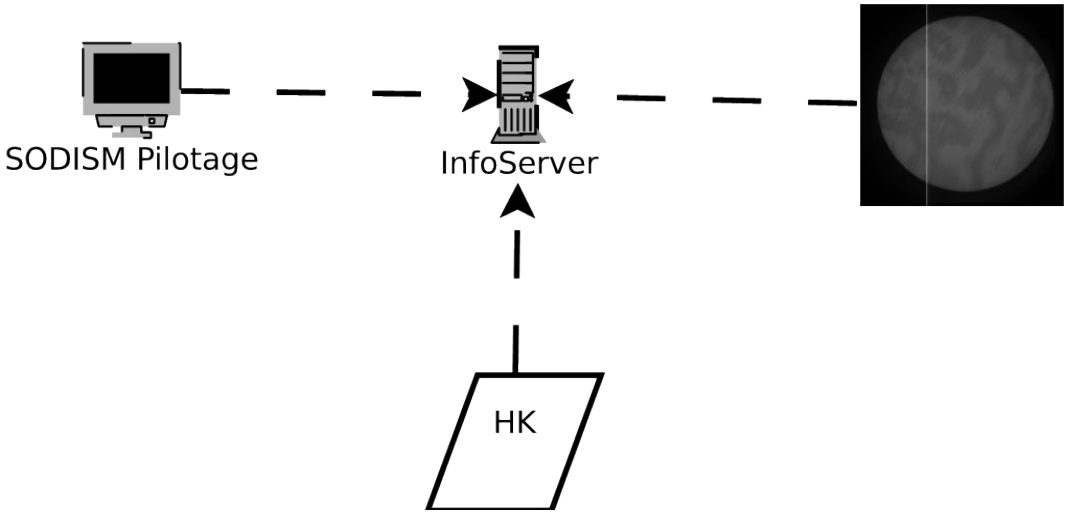


Le serveur et le client échangent donc des informations grâce à des objets JSON dont la structure est spécifiée en annexe.

La récupération des informations

Le serveur d'information se doit de récolter les informations à divers endroits et selon diverses sources:

- Fichier HK (House Keeping): Ces fichiers textes se trouvent déjà sur le centralisateur.
- Fichier fits : Ces fichiers se trouvent déjà sur le centralisateur.
- Liaison avec un ordinateur de PICARDSOL : Le serveur peut être configuré pour accepter des connexions avec un ordinateur de pilotage par exemple afin de récupérer les coordonnées visées et effectives, l'ascension droite/ déclinaison.



Alarmes

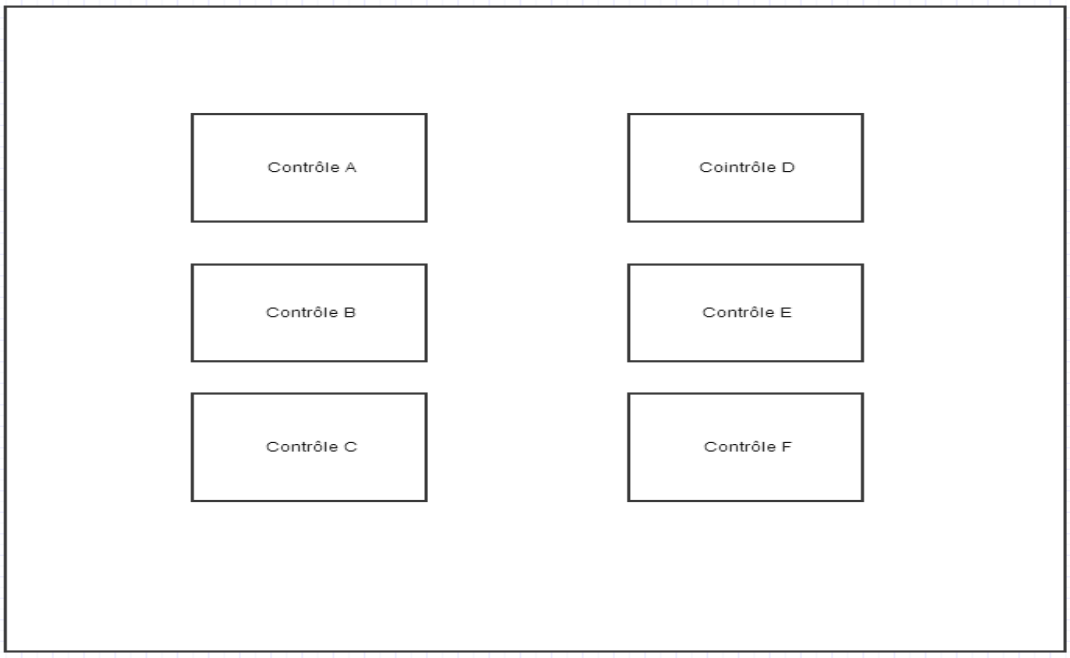
Les alarmes représentent une partie importante du projet. Elles servent à avertir l'utilisateur d'un événement qui sollicite une attention particulière. Elle peuvent être déclenchée par, par exemple, une température ou pression trop élevée, un dysfonctionnement dans le système…

L'outil à développer doit par conséquent implémenter une gestion totale et sécurisée des alarmes.

Ces dernières peuvent provenir de différentes sources. Elles peuvent provenir d'un fichier HK ou encore d'un message de la part d'un ordinateur de pilotage, d'acquisition ou cuve.

Concepts

Après étude, je me suis rendu compte que la page demandée pouvait être déclinée en modules indépendants. J'ai donc décidé de découper la page en modules appelés « Contrôles ». L'idée est donc de disposer les modules sur la page ainsi :



Chaque contrôle a son propre code HTML et JavaScript qui est indépendant des autres.

Il existe deux types de contrôles :

- Les contrôles Vue (sert typiquement à afficher une information)
- Les contrôles Commande (sert à envoyer une commande mais peut aussi servir à afficher une information)

La répartition des types de contrôles selon le type d'utilisateur se trouve en annexe.

Côté serveur d'information, il faut des entités capables de récolter des informations de manière indépendantes et parallèle. J'ai donc dû concevoir des « InfoSeeker ». Ces entités peuvent être déclinées en deux types : les InfoSeeker temporisés qui sont conçus pour se mettre à jour de façon régulière et les autres qui, quant à eux, sont conçus pour envoyer une information sur un événement (fichier modifié, certaines valeurs dépassent un seuil ...).
Le diagramme concernant la partie InfoSeeker se trouve en annexe.

Les InfoSeekers génèrent des événements lorsqu'ils estiment que l'information doit être transmise aux utilisateurs. Le serveur d'information réagit à ces événements et les distribue aux utilisateurs selon leur statut.

La gestion des commandes est assez délicate. En effet, la transmission des commandes par le serveur d'information dépend de plusieurs paramètre :
Une restriction basée sur le statut d'utilisateur et donc sur les permissions est appliquée. Mais il arrive que les commandes nécessitent une gestion plus approfondie. Les gestionnaires de commande (ou « CommandManagers ») servent à pallier ce problème et permettre une gestion plus spécifique des commandes.

L'exemple le plus représentatif de cette nécessité est la gestion des commandes de correction. La demande de correction pose problème si une demande ultérieure n'est pas achevée. Ici, l'utilisateur est notifié de cet incident et la commande n'est pas transmise.
Par conséquent un CommandManager a dû être mis en place. Ce gestionnaire agit comme un automate à état fini. Lorsqu'il transmet une correction, il interdit les suivantes tant qu'il n'a pas reçu un signal de fin de correction de la part du pilotage de SODISM et que l'acquisition suivante est arrivée. C'est seulement à ces conditions qu'il acceptera de transmettre une nouvelle correction.

Le diagramme de classe de la partie CommandManager se trouve en annexe

Choix des outils

Serveur d'information

Bien entendu, le serveur d'information se doit d'être compatible avec le centralisateur. Ce dernier se trouvant sous une distribution Linux, plusieurs choix ont été étudiés :

- C (Bas niveau)
- C++ (Langage inconnu)
- PHP (Pas très adapté)
- JavaScript (au travers de Node.js par exemple)
- C# (langage enseigné en deuxième année, selon moi un des meilleurs, nombreuses bibliothèques et sources disponible pour les websockets)

J'ai opté le C# qui se trouve être le langage étudié en deuxième année de DUT Informatique et avec lequel je me sentais à l'aise.
Du code C# (ou tout autre langage du framework .NET) peut très bien être compilé et exécuté sur une machine qui fonctionne sous une autre OS que Windows (ANNEXE)

Serveur web

Pour le serveur web j'ai décidé d'utiliser le langage PHP. Ce langage permettant à la fois une création d'interfaces dynamiques et une gestion des données, son utilisation est toute indiquée.

L'interface web a été créée grâce au triplet HTML/CSS/JavaScript.

L'utilisation de ces langages permet la création simple et efficace de pages web. La maniabilité et la lisibilité du HTML en font un outil de choix pour le développement des interfaces. Son association avec le CSS permet une gestion des styles graphiques encore plus simplifiée, grâce à la création de feuilles de styles. L'utilisation d'une librairie CSS telle que Bootstrap permet d'accélérer grandement le travail.

Enfin, le JavaScript est un langage de programmation Web qui permet d'ajouter de l'interactivité aux pages Web.

Il est un des langages les plus utilisés sur Internet, puisqu'il fonctionne sur quasiment tous les navigateurs à ce jour. L'utilisation d'un framework tel que jQuery simplifie grandement le développement.

De plus, le JavaScript permet l'ouverture et la gestion simplifiée, au travers d'une interface de programmation, de websocket.

SGBD

Concernant le Système de Gestion de Base de Données, mon choix s'est porté vers MySQL qui est le système de gestion de base de données le plus populaire, intuitif et simple d'accès ; il est tout indiqué pour le projet d'autant plus que c'est un logiciel libre.

De plus, un outil tel que PHPMyAdmin permet une création et administration de la base de données simple, intuitive et efficace. Il a d'ailleurs été l'outil employé pour la création de la base de données.

Développement

Outillage mis en œuvre

Lors du développement du serveur d'information, le principal outil mis en œuvre a été le logiciel MonoDevelop.

Il s'agit d'un environnement de développement intégré (ou IDE) utilisé pour la création, édition, compilation et débogage d'applications écrites avec le framework .NET. Il fait quasiment partie intégrante du projet Mono qui est une plateforme qui consiste à porter .NET sur toute les systèmes faisant de .NET un très sérieux concurrent du java.

Le langage de base supporté est le C#.Le serveur Web, quant à lui, n'a nécessité l'utilisation que d'un éditeur de texte spécialisé dans les langages web.

La création et l'administration de la base de données se sont faites avec phpMyAdmin, un logiciel dédié à cette utilisation.

Problèmes rencontrés/ Solution apportées

Durant la phase de développement, j'ai dû faire face à plusieurs difficultés.

Gestion des erreurs

Le serveur d'information peut avoir à traiter différents cas d'erreur de différentes natures et à différents niveaux (recevoir une commande à transmettre à un ordinateur qui ne s'est pas encore connecté, fichier HK absent ou dont la permission de lecture n'est pas accordée …)

Pour gérer les erreurs, les design patterns fail-fast & Conditional warnings ont été employés. Une technique appelée « empilage d'exception » a également été utilisée.

Configuration de l'application

La configuration représente une partie importante du projet. Pour le rendre flexible, un maximum d'éléments doivent être configurables. Les sources de données, les connexions, les gestionnaires de commande, les InfoSeekers et bien sûr les éléments de base (informations de connexion à la base de données, port d'écoute, nombre maximum de clients, filtre IP ...)

Concernant la configuration de l'application, j'ai donc utilisé le framework de configuration intégré à .NET. En plus d'être normalisé, il est éditable avec un simple éditeur de texte ou un éditeur de documents XML.

La configuration a aussi beaucoup nécessité de la réflexivité (instanciation dynamique de classe grâce aux classes Type et Activator de .NET).

Son avec JavaScript

Lors d'événements importants qui nécessitent une attention particulière de la part de l'utilisateur, l'interface génère un son pour attirer son attention. Cette interface est écrite en html et en javaScript. Or la gestion du son avec ces langages est loin d'être simple.

Pour pallier le problème du son, un outil qui s'appelle SoundManager2 a été utilisé. Ce dernier utilise HTML5 et, de manière détournée, adobe flash player. Cela implique que l'utilisateur doit avoir l'extension d'adobe pour être notifié sonorement.

Sécurité

Une faille de sécurité s'est révélée lors de la phase de développement. L'authentification s'effectuait uniquement sur le serveur web, or, cette situation représentait un risque. Une personne mal intentionnée aurait pu (non sans difficultés) se substituer à un observateur et agir de façon malveillante.

Pour résoudre le problème de sécurité, une double authentification s'opère. Il y a donc une authentification normale de la part du serveur web puis une seconde authentification (invisible pour l'utilisateur) auprès du serveur d'information.

Communication avec les utilisateurs

Pour faciliter les échanges entre le navigateur d'un utilisateur et le serveur d'information, une couche d'abstraction a été ajoutée grâce à l'emploi d'une librairie open source dédiée tout spécialement à l'utilisation des WebSockets avec .NET et plus précisément avec C# (SuperWebSocket).

Côté client, ce travail n'était pas nécessaire puisque les WebSockets sont gérés nativement par le JavaScript.

Configuration de l'application

Choix pour la modularité et l’évolutivité

Tout le projet est réalisé dans l'optique d'une évolution future. En effet, il a été produit de manière à ce que l'on puisse :

- Ajouter facilement une source de données (fichier HK, connexion à un ordinateur de contrôle)
- Ajouter un autre type de donnée.
- Ajouter des contrôles
- Ajouter/ configurer un gestionnaire de commande
- Ajouter un autre statut d'utilisateur
- Modifier les permissions selon le statut d'utilisateur

Tests

Tests unitaires

Les tests unitaires sont des tests que le programmeur effectue sur des portions de programme. Ces tests permettent de contrôler le fonctionnement du module. En effet, le programmeur définit l'état et les sorties après exécution, et vérifie que les résultats de son module correspondent aux résultats définis.

La plupart des éléments ont passé les tests avec succès, les autres ne peuvent pas être testés unitairement et nécessitent des tests d'intégration.

L'ensemble des tests unitaires du serveur d'information ont été écrits grâce à l'outil Nunit spécialement conçu pour les tests en .NET.

Tests d'intégration

Suite aux tests unitaires ont lieu les tests d'intégration.

Ces tests ont principalement été effectués lors de l'intégration de la solution sur le centralisateur. Les tests d'intégration ont donné des résultats concluants.

En effet, la connexion à la base de données était effective et les deux serveurs ont tourné sans aucun imprévu.

Une fois ces tests terminés, il faut s'occuper des tests de validation.

Tests de validation

Les tests de validation vérifient la conformité du logiciel ou du projet par rapport au cahier des charges du client.

La difficulté des tests de validation dans ce projet vient du fait que l'instrumentation PICARD-SOL dépend de la météo. En effet, si le ciel est dégagé, les observations sont lancées et on ne peut donc pas les interrompre pour les tests.

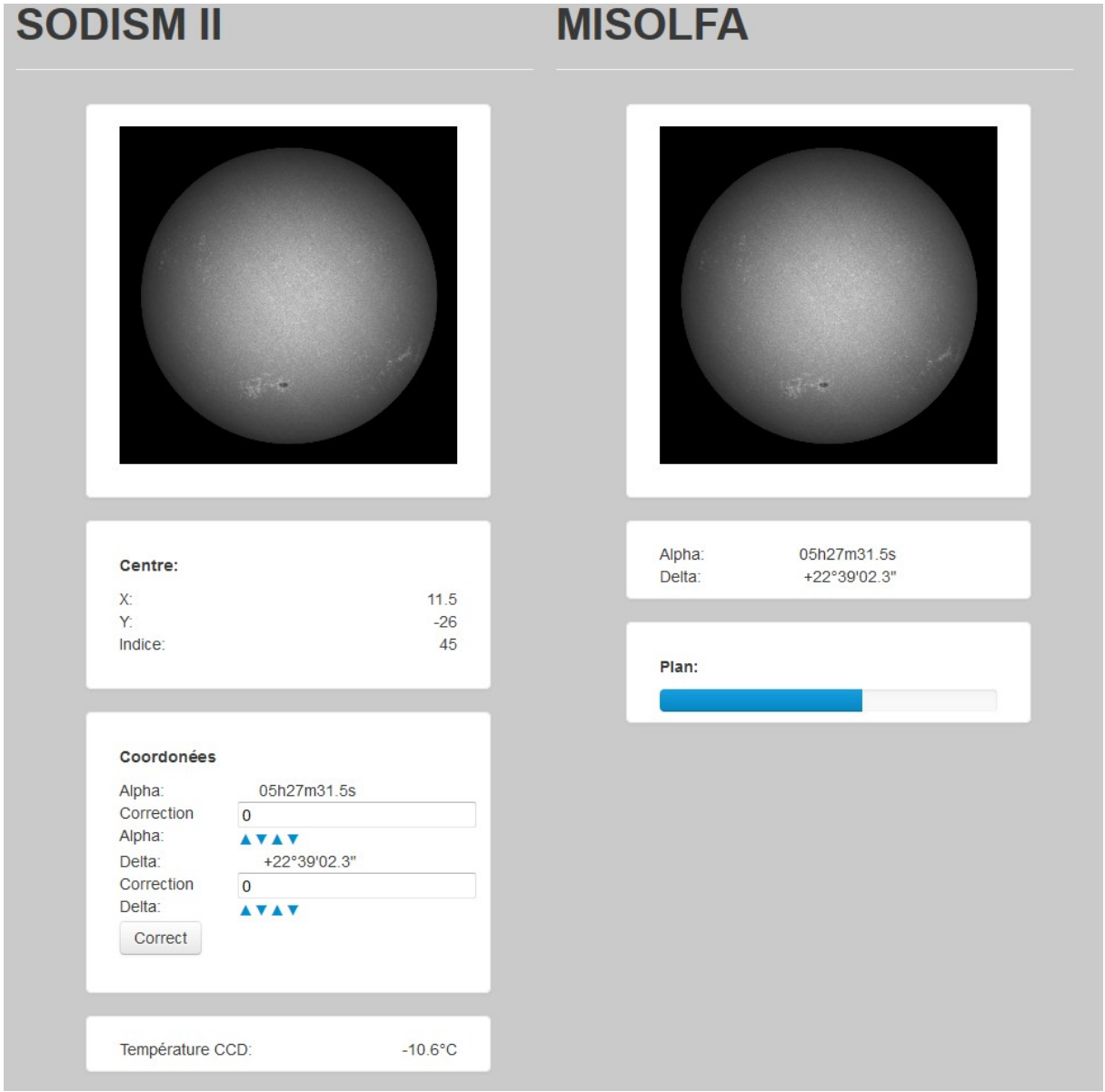
Certains contrôles ont pu être testés et validés mais d'autre nécessitent encore une validation future.

Résultats

État final

Le serveur d'information se lance et se commande à partir d'une console.

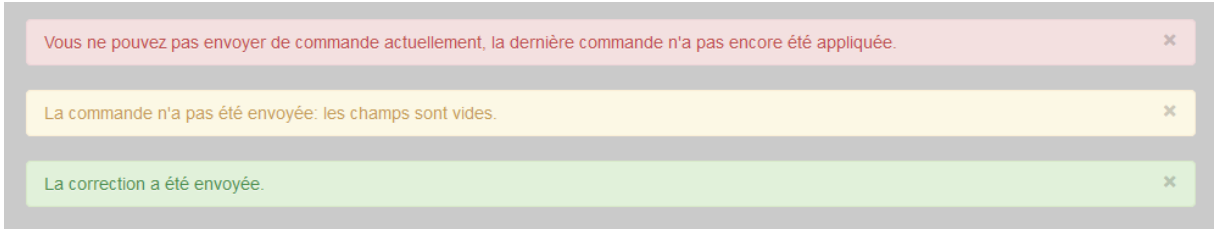
Pour l'interface web :



Les images sont les même car ce sont celles utilisées pour les tests. On peut voir la différence entre un contrôle «Vue » et un contrôle « Commande » (les contrôles de correction).

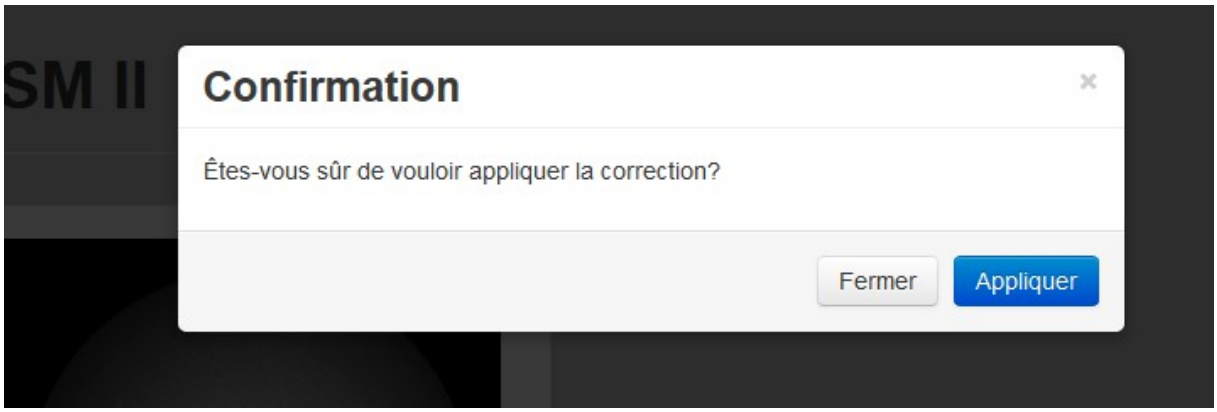
Cette interface variera bien-sûr en fonction du type d'utilisateur qui se connecte. Il s'agissait ici d'un observateur.

Voici des exemples des notifications qu'un utilisateur est susceptible de recevoir.



Ces notifications apparaissent dans la zone de notification située en haut de la page. Au moment de leur apparition, un son est joué selon le type d'alerte (configurable) et la page scroll jusqu'aux alertes.

Pour certains contrôles, une confirmation peut être demandée :



Quantification

Serveur d’information :	
Nombre de classes	34
Complexité cyclomatique moyenne	26
Nombre de lignes de code	2341
Pourcentage de commentaire	21,3%
Serveur web :	
Nombre de contrôles	12
Nombre de lignes php	267
Nombre de lignes Javascript	328
Nombre de lignes html	186

Perspectives d’évolution

Plusieurs évolutions sont envisageables. On peut par exemple imaginer une interface graphique pour le serveur d’information, une interface de gestion pour l’administration du système plus intuitive ou encore une gestion plus avancée des permissions.

Originalement, la base de données des utilisateurs devait être fusionnée à l'annuaire (LDAP) de l'observatoire. Cette idée a été abandonnée car le système actuel était voué à périr. Il sera possible, une fois le nouveau système en place, de reprendre et mettre en application cette idée.

Bien sûr on peut également penser à l’ajout de contrôles ou de source d’informations (tâche rendue facile par l’architecture du projet).

Conclusion

Ce stage d'obtention de DUT, effectué au sein de Observatoire de la Côte d'Azur a consisté à concevoir, développer et valider un outil de commande à distance de l'instrumentation PICARD-SOL avec pour finalité de faciliter le processus d'observation de celle-ci.

Durant ce stage, certaines libertés (notamment dans le choix des outils) m'ont permis de me familiariser avec les prises de décisions. Bien que la collaboration de toute l'équipe ait été forte, j'ai dû faire preuve d'une certaine autonomie.
J'ai aussi été amené à découvrir et exploiter des concepts et outils qui m'étaient inconnus dans différents domaines, notamment grâce aux WebSockets, ce qui est assez nouveau dans le monde de l'informatique, ou grâce à Mono.

Ce stage a été l'occasion pour moi de faire mes premiers pas dans le milieu professionnel et a constitué une occasion de mettre en pratique les compétences acquises grâce à la formation DUT Informatique.

En définitive, c'est une expérience très enrichissante qui m'a permis d'acquérir de nombreuses connaissances complémentaires à ma formation et une certaine expérience qui saura, j'en suis sûr, s'avérer utile.

J'aurai le plaisir de poursuivre ce stage au mois de juillet toujours au sein de l'équipe PICARD-SOL afin d'achever la validation de ce projet et faire évoluer un outil pour automatiser la gestion de la cuve de SODISM.

Glossaire

PHP	Langage de scripts libre principalement utilisé pour produire des pages Web dynamiques.
phpMyAdmin	Outil permettant la gestion graphique d'une base de données MySQL.
SGBD	Un Système de Gestion de Base de Données permet la manipulation de bases de données.
SQL	Langage normalisé qui sert à effectuer des opérations sur des bases de données.
C#	Langage isssu du framework .NET.
Web Socket	Technique permettant d'établir un canal de communication bidirectionnel notamment à travers un navigateur grâce à JavaScrit.
JavaScript	Langage de programmation de scripts principalement utilisé dans les pages web pour les dynamiser.
JQuery	Bibliothèque JavaScript ayant pour but de grandement simplifier des commandes communes de JavaScript.
JQuery UI	Extension de jQuery concernant les interfaces utilisateur.
SoundManager2	Framework audio multiplateforme utilisant HTML5 et Flash utilisable à travers une interface de programmation JavaScript.
Mono	Implémentation open source du framework .NET. Il permet de compiler du code .NET et de l'exécuter sur quasiment tous les OS.
HTML	HyperText Markup Language. Langage permettant de modéliser une page web.
CSS	Cascading Style Sheets. Langage qui sert à décrire la présentation des documents HTML (l'apparence qu'ils auront).
Twitter BootStrap	Collection d'outils utiles à la création de sites et applications web. (Framework CSS/JavaScript)
HTTP	Hypertext Transfer Protocol est un protocole de communication client-serveur développé pour le World Wide Web
MISOLFA	Moniteur d'Images SoLaires Franco-Algérien
PICARD	Télescope spatiale du CNES envoyé en juin 2010
SODISM	Solar Diameter and Surface Mapper. Télescope actuellement embarqué sur le satellite PICARD pour effectuer des observations du diamètre solaire.
SODISM2	Réplique de SODISM
House Keeping	Fichier contenant toute les informations relatives à l'environnement d'un instrument

Annexes

ANNEXE I : Rôles des utilisateurs

SODISM :

	Administrateur	Observateur	Visualisateur
Dernière image	V	V	V
Coordonnées	C	C	V
État	V	V	V
Alarmes	V	V	V
Tensions	C	C	
Température CCD	V	V	V
Pression cuve	V	V	V
Température de l'eau	V	V	V
Fonctionnement moteur	C	V (sauf arrêt urgence)	

MISOLFA :

	Administrateur	Observateur	Visualiseur
Dernière image	V	V	V
Coordonnées	C	C	V
État	V	V	V
Alarmes	V	V	V
État des roues	V	V	V
Focalisation	C	V	
Séquences et plans	C	C	V (plan en cours)
Signaux voie pupille	V	V	V
Tensions	V	V	
Température CCD	V	V	
Température peltier/pupille	V	V	
Fonctionnement moteur	V	V (sauf arrêt urgence)	
Gain/filtre	C	V	

« V » : Vue
« C » : Contrôle
« » : Aucun accès

ANNEXE II Structure de données (des échanges)

Messages JSON.

I. Client => Serveur

{

« CommandType » : CommandType

…

}

enum CommandType {

Register = 0,

Command,

InformationRequest

}

Dans le cas de Register :

But : Authentifier un utilisateur avec ses identifiants. Pas besoin du statut qui sera récupéré au moment du contrôle.

« id » : id

« hashedPass » : hashedpass

Dans le cas de Command:

But : Commande provenant de l'utilisateur (Coordonnées …).

« id » : id

Ici, l'id est en fait un mot clé unique à chaque commande. Par exemple « coordinates » …

Dans le cas d'InformationRequest:

But : Requête d'information spontanée de l'utilisateur. Sert par exemple à la mise à jour immédiate d'une information dont le cycle de rafraichissement est long.

« id » : idInfo

II. Serveur => Client

{

« ResponseType »: ResponseType

…

}

enum ResponseType {

Message = 2

Error = 255

}

Dans le cas de Message:

But : Transmettre un message (que ce soit une information, une alerte...)

« id » : idInfo

Le reste de la structure dépendra de l'id.

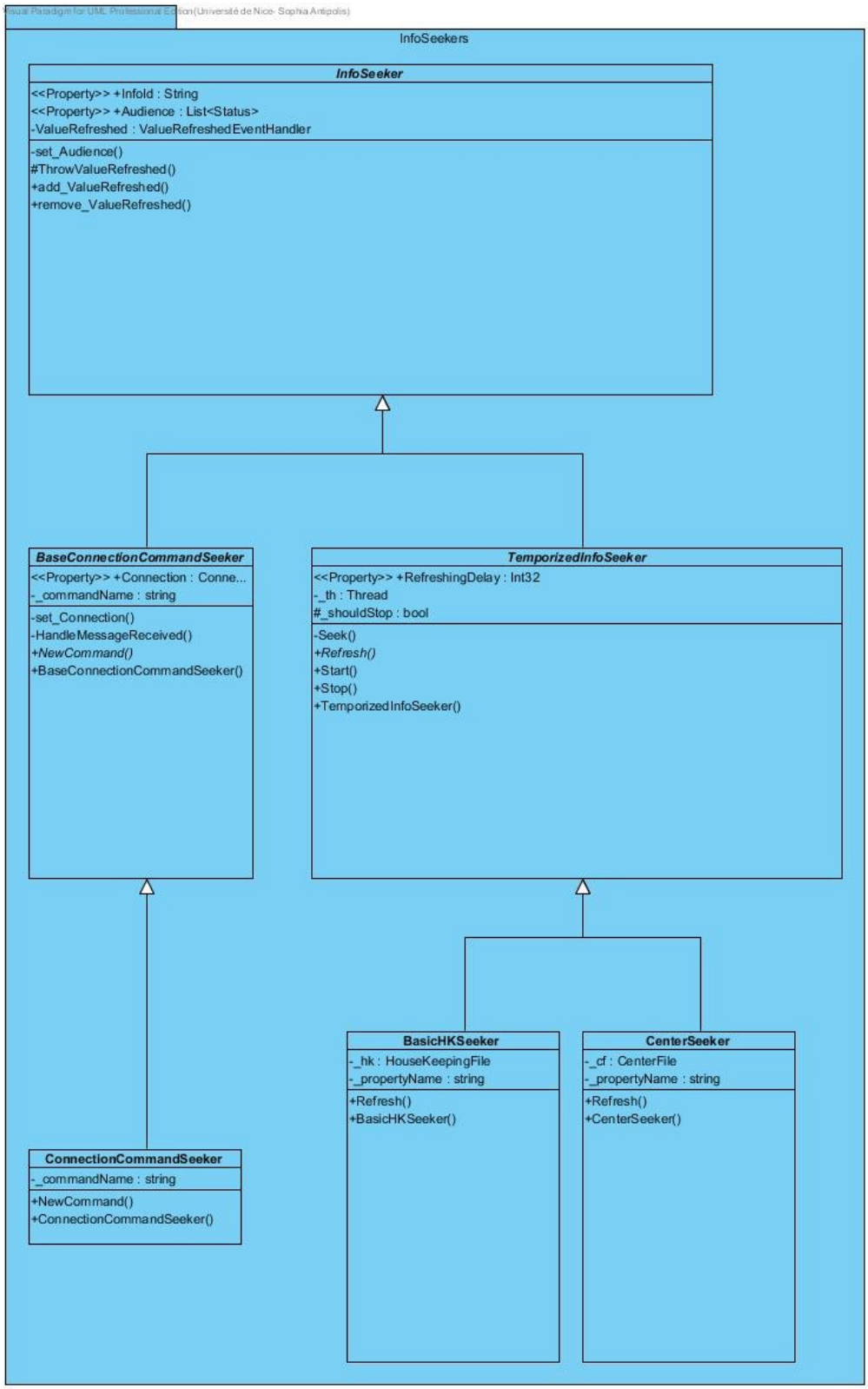
Dans le cas d’Error:

But : Transmettre un message (que ce soit une information, une alerte...)

«Error» : Message de l'erreur

ANNEXE III : Diagrammes de classe

InfoSeekers :



CommandManagers :